International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 296~307
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp296-307

Cooperative hierarchical based edge-computing approach for resources allocation of distributed mobile and IoT applications

Maha Aljarah[1], Mohammad Shurman[2], Sharhabeel H. Alnabelsi[3]
[1]Computer Engineering Department, Jordan University of Science and Technology, Jordan
[2]Network Engineering and Security Department, Jordan University of Science and Technology, Jordan
[3]Computer Engineering Department, Faculty of Eng. Technology, Al-Balqa Applied University, Jordan
[3]Computer and Networks Eng. Department, Al Ain University of Science and Technology, United Arab Emirates

Article Info

Article history:

Received Feb 23, 2019
Revised Aug 27, 2019
Accepted Aug 30, 2019

Keywords:

Edge computing
Internet of things (IoT)
Mobile edge computing
Modules placement
Resources management

ABSTRACT

Using mobile and Internet of Things (IoT) applications is becoming very popular and obtained researchers' interest and commercial investment, in order to fulfill future vision and the requirements for smart cities. These applications have common demands such as fast response, distributed nature, and awareness of service location. However, these requirements' nature cannot be satisfied by central systems services that reside in the clouds. Therefore, edge computing paradigm has emerged to satisfy such demands, by providing an extension for cloud resources at the network edge, and consequently, they become closer to end-user devices. In this paper, exploiting edge resources is studied; therefore, a cooperative-hierarchical approach for executing the pre-partitioned applications' modules between edges resources is proposed, in order to reduce traffic between the network core and the cloud, where this proposed approach has a polynomial-time complexity. Furthermore, edge computing increases the efficiency of providing services, and improves end-user experience. To validate our proposed cooperative-hierarchical approach for modules placement between edge nodes' resources, iFogSim toolkit is used. The obtained simulation results show that the proposed approach reduces network's load and the total delay compared to a baseline approach for modules' placement, moreover, it increases the network's overall throughput.

Corresponding Author:

Mohammad Shurman,
Network Engineering and Security Department,
Jordan University of Science and Technology,
Irbid, Jordan.
Email:alshurman@just.edu.jo

1. INTRODUCTION

First, let us define Internet of Things (IoT), as devices that can receive or transmit data using transceivers, such as wireless sensors, vehicles, actuators, smart grids, or any smart device, in a way that connects devices together. Edge computing paradigm (or similar platforms as Mobile Edge Computing (MEC), fog computing or cloudlet) was introduced, due to the fact that the mobile and IoT applications demands and resources requirements have increased, e.g.; computing platforms [1-5]. Consequently, the necessity of a new platform that supports these applications demands within a close proximity of computational and storage resources, within a geographically distributed area, are really a necessity. In contrast to the current cloud paradigm, a centralized infrastructure is more suitable for regular personal computer's applications rather than the requirements nature of mobile and IoT devices' applications.

Fog computing paradigm is introduced by Cisco [6], in order to satisfy crucial demands for IoT services such as mobility support, high geographically distribution of real-time, and location-based applications. Mobile applications produce high network traffic that causes congestions, collisions, and unbearable delay, especially, if this traffic is forwarded through the network's core.

Cloudlet is the computation resources that located closer to mobile-user devices, in order to perform high computing power requirements [7]. Other enabling technology for IoT is 5G technology that is expected to provide massive connections for machines' communication to satisfy the mobility for high-density devices, scalability, and low latency requirements of IoT environment. Edge computing is considered as a key technology to boost 5G environment with the computational and storage resources [8-12]. MEC is also considered another key technology for 5G, enabling Software Defined Network (SDN) [13] and Network Function Virtualization (NFV), in order to apply different network functions by utilizing edge node with multiple virtual machines, and consequently, providing end-devices with various functions.

Content Delivery Networks (CDNs) is the inspiration for edge computing paradigm, such that the too frequently accessed web contents are pre-fetched to edge nodes that are closer to end-user devices. Edge computing extends this idea to use it in cloud's infrastructure, such that edge nodes or cloudlets execute the code instead of the cloud only if they have the required resources [14]. In caches, data pre-fetching can be in a proactive manner [15], where this technique can be applied to mobile users such that fog servers provide services based on users locations, in order to increase system's response time.

Edge computing is also capable of coping with internet shift usage model, moving from host-centric design, which uses routing information, to Information-Centric Networking (ICN) design [16], by distributing a massive amount of information through network nodes. This information will be accessed regardless of location, leading to a fast information retrieval from the closest location [17].

In this work, we focus on exploiting edge resources efficiently; especially, these days the IoT devices are growing rapidly. In order to solve this problem, we introduce a cooperative-hierarchical approach for executing the pre-partitioned applications' modules between neighbor-edge nodes in a distributed manner using edges resources, in order to increase resources utilization. Also, the effect of positions' for neighboring nodes compared to the position of the cloud (proximity distance) on performance is studied.

The rest of this work is organized as follows: Section 2 presents our main contributions. In Section 3, a comprehensive overview of related work is presented. Section 4 explains the proposed protocol for cooperative-hierarchical approach that based on edge computing. Section 5 presents the simulation results and their insights. Finally, Section 6 discusses conclusions and the future work.

2. CONTRIBUTIONS

The contributions of this work are mainly summarized as follows:

a. A module placement algorithm is proposed, such that modules are executed cooperatively between neighbor-edge nodes by utilizing their resources rather than using farther cloud resources.
b. Simulation of practical scenarios is conducted, using iFogSim toolkit, in order to examine the performance of our proposed algorithm.
c. Studying the variation effect for the positions' of the neighboring nodes compared to the position of the cloud (proximity distance) on performance of our proposed modules' placement approach.

The simulation results demonstrate the proposed scheme increases edge resources utilization, reduces delay as compared to the scenario when modules are executed on the cloud itself. Furthermore, the proposed approach reduces the traffic that goes through the network core towards the cloud and enhances user experience by delivering a faster service.

3. RELATED WORK

Extensive research effort has been introduced on challenging issues related to the deployment of edge computing, such as designing the underlying platform, resource management, communication, energy consumption, security, and privacy. The architecture of edge computing platform has two main models: hierarchical based and software defined-based. Tong *et al*. [18] proposed a hierarchical geo-distributed edge cloud platform to handle peak loads at different tiers in the hierarchy. They compared their hierarchical design with a flat-edge cloud design, based on their results the hierarchical design achieved 25% reduction in the average programs execution delay compared to the flat-edge cloud design. Jararweh *et al*. [19] integrated a hierarchical design of cloudlet servers deployed close to end users with the MEC servers at the base station. This integration overcomes the limited coverage area problem of Wi-Fi access points used by the cloudlet. Furthermore, controlling tasks is distributed between local cloudlet controllers instead of a single MEC controller. Regarding the second model type of edge computing, the software defined based architecture,

authors of [20] proposed a theoretical-software defined based framework. Their framework integrates mobile-edge servers with software defined controlling system of local and global layers. Both layers are provided with multiple controlling units including network entities, storage, security, computational resources, data aggregation and an IoT unit for controlling and monitoring IoT sensors and actuators. Specifically, local layer serves time sensitive applications in the local domain, while the global layer serves applications that require data aggregation from several mobile edge servers.

Du *et al.* [21] designed an application-specific MEC platform that optimizes the Mobile Virtual Network Operators (MVNOs) by applying software defined data plane paradigm. Authors of [22] integrate MEC with the emerging technologies of Software Defined Network (SDN) and Network Functions Virtualization (NFV), in order to achieve better edge and IoT resources deployment and management. In the same context, Ravindran *et al.* [23] merged SDN, NFV and ICN to design an architecture for edge-cloud services. Moreover, authors in [24] developed adaptive routing schemes for both ordinary and emergent delay requirement of data transmission in industrial IoT framework that combines edge computing and software defined network. Another work that combines SDN with MEC to support reliability, agility, responsiveness, and application specific requirement of dynamic vehicular network is proposed in [25].

Regarding communication stage, Channel State Information (CSI) is considered by [26] for control and management schemes in MEC systems, taking into consideration environment effects, e.g.; fading, on wireless channel state, which affects transmission data rate, and transmission energy consumption regarding the allowed transmission latency and suitability of channel state for appropriate computation offloading decision by the controller.

Researchers have paid an attention to management and edge resources' provisioning, Hu *et al.* [27] examine offloading computation impact from the mobile devices to edge's node compared to the exsiting system which offload tasks' processing to the cloud. The results show an improvement in energy consumption for mobile devices and response time depending on the offloading distance. Authors in [28] proposed a dynamic services migration policy in mobile edge-clouds, using a distance-based Markov Decision Process (MDP) framework. This framework approximates user's mobility movement model in 2-D space. They confirmed the superiority of their model analytically and experimentally over various baseline models, such as: no migration, always migrate, and myopic. In [29], a framework is presented for resources management in datacenters, in which resources assignment is enhanced based on users' behavior, type of service, and price.

Resources provisioning approach is proposed in [30], its goal is to maximize fog resources utilization. A hierarchal-fog framework is introduced that provides resource-controlling services in both the cloud and the fog. In this framework, fog colony has a set of fog cells, modules of IoT, and one fog-control node. It has three layers, such that the first layer has multiple fog colonies. The second layer has a fog-control node that orchestrating and connects fog colonies together in the first layer. The third layer has the fog-cloud control middleware that controls cloud services. Control nodes provision cloud resources in order to maximize their utilization, also guarantee end-user requested service and forward it to a higher network level.

Kapsalis *et al.* [31] proposed a fog platform that has four layers: First layer, or the lowest layer, contains IoT devices. The second layer contains gateways named hub layer that connects devices in first layer. The third layer is the fog layer that has a fog broker and fog servers that manages resources. The fourth layer is the cloud layer. Authors proposed a workload-balancer module implemented in the fog broker, its role is to balance fog resources utilization based on the remaining batteries power for mobile devices, latency, and current utilization.

A Message Queuing Telemetry Transport (MQTT) protocol is used for communication in WSNs [32], where exchanged messages contain a code that will be executed, required data, and metadata that specify tasks characteristics. The MQTT protocol is used for communication in the proposed SDN-based fog computing system [33]. A customized integration of edge switches, named fog nodes, with SDN controller and a MQTT broker is developed. Fog node receives MQTT massages from the IoT devices and publishes their data with a certain topic, or type. After that, the fog node transmits the message to devices requesting the same type. Moreover, they proposed conducting analytical analysis in the SDN controller such that it acts as the central node.

Addressing the energy consumption problem, authors of [34] proposed an approach for dynamic tasks assignment in heterogeneous- mobile-embedded systems and cores in mobile clouds that reduces the total energy cost using cyber-enabled applications. Furthermore, for green computing, authors of [35] proposed a mechanism to prevent energy wasting in mobile devices while using dynamic wireless communication. This mechanism is based on cloudlets layer between mobile devices and the cloud, performing a dynamic search to achieve better communication experience between mobile devices and the cloud. Moreover, authors in [36] proposed an energy aware scheme for load balancing and scheduling in

smart manufacturing based on fog platform. Regarding security, authors of [37] proposed a solution for security and privacy concerns, in order to maximize security and privacy levels for transmission, while maintaining a successful connection probability within a given timing constraints using multi-channel communication.

4. OUR PROPOSED APPROACH

The main goal of this work is increasing edges' resources utilization using a cooperative-hierarchical based approach; therefore, this work proposes: (a) An edge-computing platform architecture that facilitates communication between edge's entities. (b) An approach with a polynomial-time complexity, in order to distribute the pre-partitioned application modules between edge resources.

4.1. System architecture

In this subsection, the platform for edge computing and edge node structure is discussed as follows:

4.1.1. Platform for edge computing

The structure of the proposed platform, as illustrated in Figure 1, consists the core network and three layers: (1)- End layer, (2)- Edge layer, (3)- Cloud layer. The end layer is the lowest layer that has mobile and IoT devices, e.g.; sensors, with poor or limited resources, e.g.; computing platform and memory, where nodes in this layer cannot perform tasks with a heavy computation.

Above the end layer is the edge layer, or the middle layer, that has a geographically distributed edge nodes with computation, communication, and memory resources in a close proximity to the end layer devices. This layer can perform tasks independently, and running pre-defined applications on the behalf of the lower layer, or the end layer, devices in a fast manner to reduce latency, and also to provide efficient streaming without the assistance of the cloud layer.

The upper layer is named cloud layer, which is the farthest layer from users-devices in this platform. Due to this architecture nature, this work goal is to minimize communication between the cloud layer and other layers, namely end and edge layers, in order to reduce number of routing requests through the core network. As a result, the traffic amount through the core network, used bandwidth, and congestion are all reduced. On the other hand, cloud resources could be used as data storage for a long-term to be analyzed in depth. Moreover, if edge-nodes' resources are not capable of executing some tasks, thus, the edge node forwards these heavy processing tasks' requests to the cloud as a final resort.

4.1.2. Edge node structure

Each edge node owns storage, computation, and communication resources. These resources can be heterogeneous and distributed in a hierarchical way, reducing the distance to the cloud to a certain extent, where less powerful resources are located closer to end devices. Moving away from end devices, the more powerful resources become available. In the proposed approach, as discussed in details in our proposed approach description, Subsection 4.2. Tasks are distributed such that light-computation tasks are executed close to the end devices, while tasks with heavy computation requirements are executed by upper level resources, as shown in Figure 2.

Edge nodes might vary in their available resources, and hence depending on the type of task's requirements, the suitable edge node is selected. Edge nodes that provides heavy computational services, such as gaming or augmented reality, should be provided with more resources than nodes that provide light-computation services. This design shows the necessity of the proposed cooperative-hierarchical based approach, e.g.; if a node with light resources is flooded with task requests and has a neighbor node that has more resources that currently are underutilized. As a result, some tasks are forwarded to this neighbor node rather than forwarding them far to the cloud layer. Clearly, this reduces traffic in the core network and enhances response time. For example, when deploying systems in wide regions, such as a smart driving assistance, they can benefit from our proposed approach.

When several edge nodes can provide the same requested service, they might be distributed to cover a wide region; however, the amount of traffic is not the same for different areas. When employing our proposed approach between neighboring nodes, the utilization of edge nodes' resources is increased and interacting with the cloud layer is reduced, as demonstrated in Subsection 4.2.2. In this work, it is assumed that each edge node also consists an edge-orchestration node that represents the main component in the edge node. This node buffers information about each running task, such as its running host, the current state of computation and memory resources, in addition to information about the connection state with neighbor-edge nodes and the cloud.

As shown in Figure 2, there is a node named orchestration node that is responsible for resources management, making decisions regarding executing tasks on its node's resources, or transferring this task to a neighbor node or even forward it to the cloud layer.

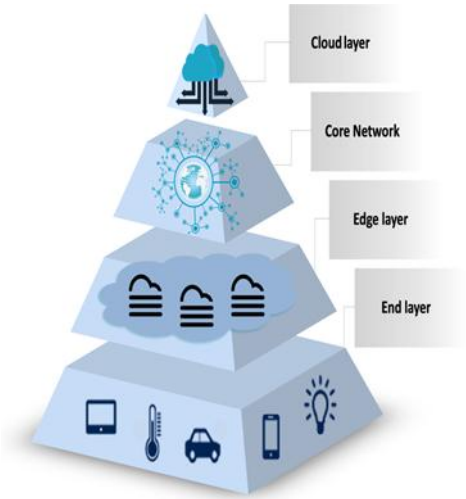

Figure 1. Platform with cloud and edge infrastructures

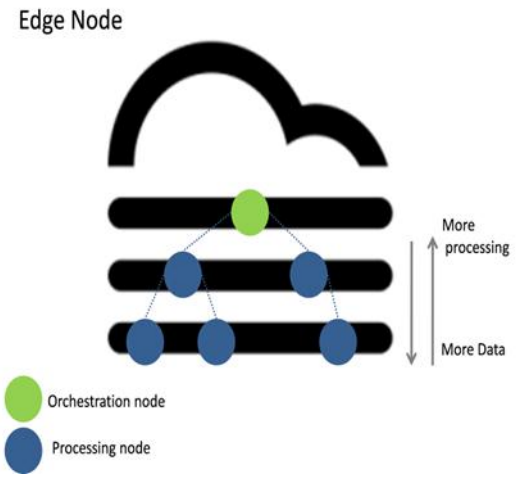

Figure 2. Edge node structure

4.2. Our proposed approach description

In order to improve edge-node resources utilization, a cooperative-hierarchical approach is proposed in order to distribute and execute application modules between neighbor-edge nodes.

4.2.1. Application

An application to be executed on edge's resources, it must be pre-partitioned into modules; such that each module conducts a specific function. The proposed approach does not partition the application, but some other entity, e.g.; compiler, is responsible for dividing the application into modules. The complexity of each module is defined by number of parameters, such as required RAM size, required CPU time, and transmitted data size. In order to represent the application using a directed graph, a distributed data flow model is used [38], such that each module is represented by a vertex and a directed edge represents the dependency and data flow direction between modules.

4.2.2. Algorithm for modules placement

The pre-partitioned application modules are received by the edge-orchestration node, in order to be distributed to the available resources based on their needs of memory and processing requirements. The baseline module placement method, that used in *iFogSim* simulator, depends on spanning the constructed graph from the bottom to the top within the same edge node, that is searching for a suitable resource to assign each module. However, if there are no resources satisfy the module needs, therefore, it will be forwarded to the cloud. In [30], a system model is proposed that has utilization and latency metrics for the resources inside the same node, however, the utilization and the delay caused by neighbor nodes are not considered. In this proposed work, the platform is spanned horizontally searching for suitable resources in neighboring nodes, as assigning the module to the cloud is the last resort. The proposed mechanism for module's placement is shown in the below flow chart, shown in Figure 3, and is explained as follows:

1. In the edge node, the orchestration node keeps leaf-to-root paths for the hosts, also it keeps tracking the current CPU utilization (busy time) of each host. The orchestration node represents hosts inside the edge node by a tree structure based on their computing power, such that the least powerful hosts are the leaves and the most powerful host is the root. Apparently, a leaf-to-root path has the possible hosts of executing a module; where the orchestration node visits each path from leaf-to-root. As a result, modules are executed on hosts with less power, if they satisfy modules requirements, that closer to end devices, before moving them to the upper level in order to look for available resources.
2. The orchestration node schedules the order of executing the modules in the edge node based on the dependencies between them, e.g.; a module may need some modules results as input data.

3. The orchestration node visits leaf-to-root paths of hosts for each module, in order to check the feasibility of executing the module on this host, e.g.; if available CPU time is enough.
4. If the host is able to execute this module, thereby the orchestration node forwards this module to this host and also it updates its information about CPU's load for this host and this module.
5. Else, if the hosts in this level (lowest level) along the path do not have enough resources to execute the module, therefore, the orchestration node checks the upper level for other capable hosts, if exist.
6. The orchestration node checks all levels of resources in order to execute the module inside the current-edge node, an that before considering executing it on a neighbor-edge node or on the cloud. If the orchestration node is unable to execute the module inside its corresponding edge-node, due to the module's excessive resources requirements, e.g.; computing power, which is currently unavailable at hosts inside the current-edge node. Consequently, the orchestration node will sort its neighbor-edge nodes and the cloud based on the estimated delay to reach each of them.
7. The orchestration node starts with the neighbor-edge node that has the minimum-reachability estimated delay, and transmits a request for executing the module to this neighbor-edge node.
8. In the proposed approach, one level of cooperation between edge nodes is employed, e.g.; if the neighbor-orchestration node receives a request from other edge node for module execution, it checks the ability of module execution only at its highest level of resources.
9. If the original-orchestration node receives a positive response from a neighbor-edge node, it forwards all required data for executing this module to this neighbor-edge node.
10. On the other hand, if the original-orchestration node does not receive a positive response from any neighbor, therefore, it will forward this module to the cloud as the last resort.

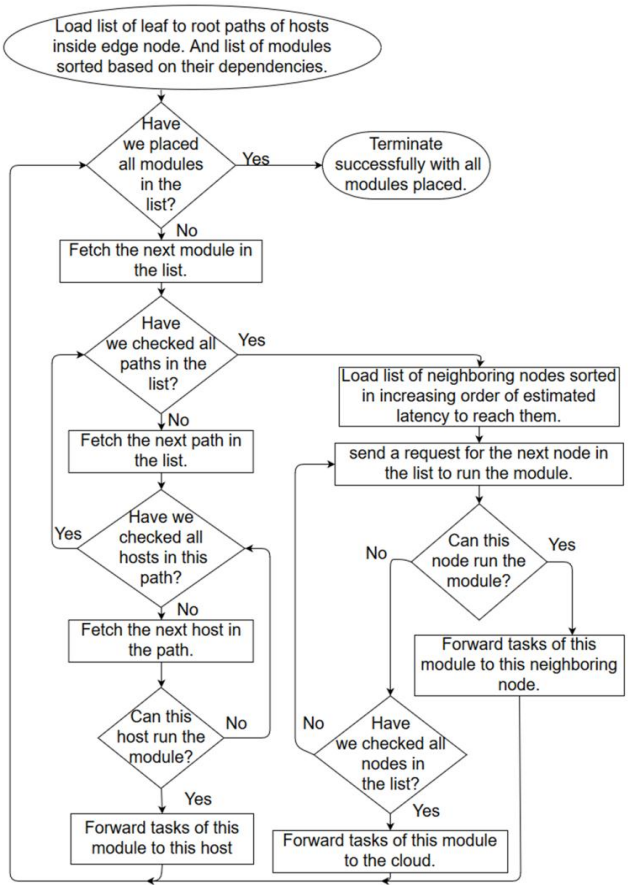

Figure 3. Flow chart of the module placement approach in edge node

Apparently, the complexity of this proposed approach is polynomial in time, which is O(M*P*N), where M, P, N are number of modules, number of paths within the edge node itself, and number of neighbor-edge nodes, respectively.

5. SIMULATIONS RESULTS

To evaluate the proposed Cooperative-Hierarchical based Edge-Computing approach for modules-placement against a baseline approach, the *iFogSim* simulation is employed [38] which is a CloudSim extension [39] for fog computing.

5.1. Simulation scenarios

The case study provided by the simulator [38] is employed for our experiments, which is an implementation for Electroencephalography (EEG) online Tractor Beam Game, in which each player wears electroencephalogram headset that equipped with a sensor, in order to transmit user's brain state to the application on the user's smart phone. The application shows all players surrounding an object, such that each player tries to pull the object toward him. The object's movement depends on the concentration level of each player; consequently, the player that has the highest concentration level will succeed to pull the object toward his direction.

The application is partitioned into three different modules: (1)- Client, (2)- Concentration calculator, (3)- Coordinator. The client module receives EEG sensor readings and pre-processes them, and then forwards them to the concentration-calculator module. Next, the concentration-calculator module finds the value for the player's concentration level and transmits this value back to the client module. After that, the client module transmits the found concentration level to the actuator's display.

At the same time, the module of concentration-calculator transmits the results periodically to the coordinator module, which is responsible for global-game state sharing with other clients. Accordingly, their displayed global state of the game is updated. Figure 4 demonstrates the dependencies and data flow between EEG-game modules [38]. Tuples are the task's specifications and data transferred between modules that used for processing by the receiving module. Each tuple has a type, in our scenario, tuples are sent by sensors to the client module are EEG type that contain EEG-sensor readings. While tuples that transmitted from the client to the concentration-calculator module, and that after conducting sensors' readings pre-processing, have the sensor type and so on, as illustrated in Figure 4. Generally, each tuple type has different specifications, such as the required task's execution time, data size in bytes before execution, and other required characteristics that needed to execute the task.

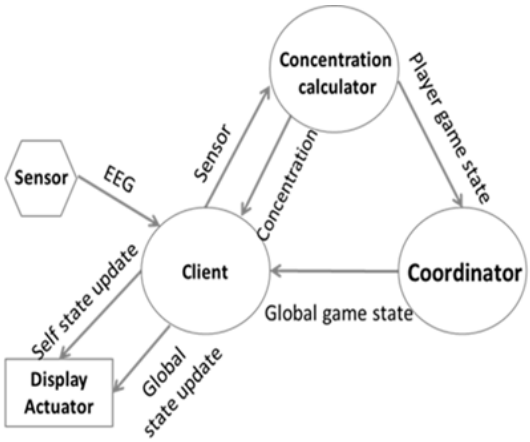

Figure 4. EEG-Tractor Beam Game modules [38]

We developed a platform for the experiments that contains two edge nodes, where each edge node has one orchestration node, one edge server and a group of mobile devices that varied through experiments from 10 to 100 devices. Figure 5(a) shows that the proposed edge-computing platform with the orchestration nodes and the communication delay between components. It is implemented using *iFogSim* toolkit. Figure 5(b) demonstrates the developed platform without having orchestration nodes, this platform is employed as the baseline model for comparison purpose. The configurations and characteristics for devices in both developed platforms are unified, in order to fairly compare our proposed cooperative-hierarchical modules placement method to the baseline placement method.

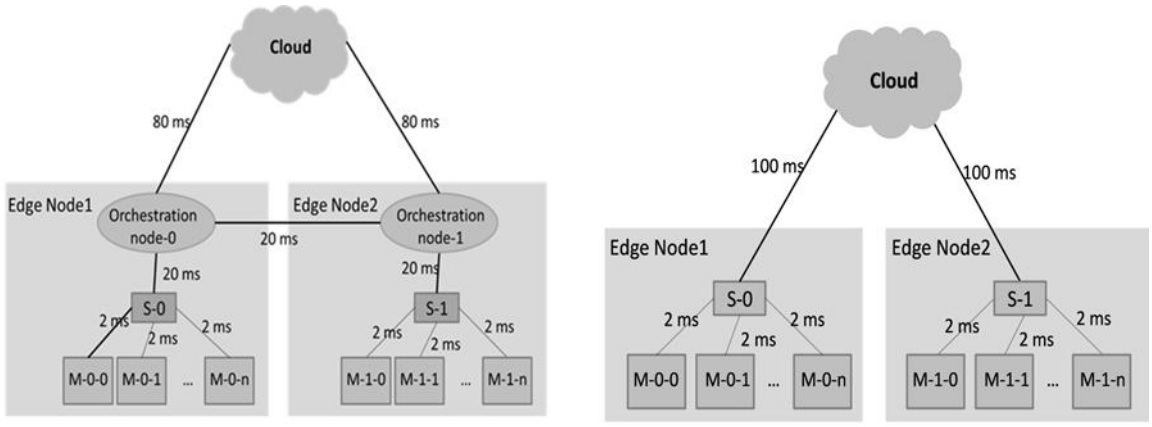

Figure 5. (a) Edge-computing developed platform with orchestration nodes (our proposed approach); (b) Edge-computing developed platform without orchestration nodes (baseline approach)

5.2. Performance metrics

To evaluate our proposed approach, we introduced the following performance metrics:

1. Total delay: is the total time delay that required for executing all tuples (tasks) including communication delay, due to the fact that these tuples travel from one module to another on different hosts and/or different edge nodes. Therefore, the total time delay includes network communication, processing, congestion, and collision time.
2. Throughput: is the total number of data tuples executed per time unit. It is evaluated by using (1), where T denotes total number of the executed tuples and D denotes the total delay.

$$Throughput = \frac{T}{D} \quad (1)$$

3. Network load: is the amount of data tuples that travels in the network and occupies it for a certain amount of time. It is evaluated by using (2), where Nd_i denotes the total delay of the i^{th} tuple and Nw_i denotes the i^{th} tuple size.

$$Network\ load = \sum_{i=1}^{T} Nd_i \times Nw_i \quad (2)$$

Notice that the network load includes congestion and collisions. Therefore, network load reduction leads to reduce collisions and traffic congestion. As above (1) and (2) demonstrate, throughput is inversely proportional to the total delay, while network load is proportional to the network overall delay. The simulation results in Subsection 5.3 prove that the throughput is influenced by reducing total delay for tuples execution more than network load. For consistency, these metrics are measured for the same number of tuples executed in both systems implementation: the proposed approach and the baseline approach, as their architectures are shown in Figure 5(a) and Figure 5(b), respectively. In simulation, when number of devices in edge nodes is increased, clearly, more tuples are produced, and hence, the assessment validity for introduced performance metrics is enhanced.

5.3. Simulation results and discussion

In our proposed cooperative-hierarchical placement approach, all mobile devices have same capabilities allows them to execute only the clients' modules, while the edge servers have different capabilities such that some servers can execute the concentration-calculator module while others cannot. Therefore, in the proposed placement approach, a server with enough computing power can execute the entities of this module. On the other hand, regarding the baseline placement method, all entities of this module are executed at the cloud which may cause a higher communication delay.

A preliminary result of this work is published in ICECTA'2017 [40], the initial enhancement results are presented for the proposed approach when 40% of nodes are closer than the cloud. However, in this extended work, more extensive results are presented as follows:

1) An extensive simulation is conducted when 30% and 70% of edge nodes are closer, to mobile and IoT devices in the end layer, than the cloud.
2) Extensive experiments are conducted for total delay, throughput, and network load metrics.
3) Moreover, this work demonstrates how different performance metrics are affected by neighbour-nodes' position or edge nodes proximity to end devices.

5.3.1. Average delay

Delay enhancement results is shown in Figure 6, for example, results show that if 10 devices emanating data, a 60% reduction in latency is achieved when placing modules on neighbor-edge nodes that are 40% closer to devices than the cloud. This reduction in delay is due to reduction of traffic directed through the network core towards the cloud, imposing less network congestion, leading to a lower delay, and that is due to employing our proposed approach rather than the baseline approach.

Delay reduction percentage is going down to 36% when there are 100 devices emanating data, because congestion and collision probability will increase in the network causing more delay. Clearly, delay reduction decreases as number of devices increases. That makes sense, due to the fact that when number of devices increases, available edge nodes resources become not sufficient to satisfy their resources requirements. Therefore, some tuples are forwarded to neighbor-edge nodes or even to the cloud resources through the network core. As a result, the overall communication overhead and data transmission time increase, in other words, average delay increases. Furthermore, Figure 6 shows that when 70% of neighbor-edge nodes are closer to mobile IoT devices than the cloud, enhancement results are better than when 40% of neighbor-edge nodes are closer. In contrast to the scenario when only 30% of neighbor-edge nodes are closer to devices, delay enhancement is less.

5.3.2. Throughput

Figure 7 proves that our proposed approach improves the throughput significantly. It is interesting to see that the proposed approach enhancement ratio is more than double the enhancement for delay, because both network delay and congestion are reduced. Also, when the number of devices increases, more tuples are generated, and thus, throughput enhancement is degraded. An interesting observation is noticed, as the number of devices increases, the enhancement reaches a saturation level. Even with this saturation, the proposed approach outperforms the baseline approach.

5.3.3. Network load

Network load enhancement means reduction in network congestion and collisions. As predicted, the proposed approach decreases the network load, since fewer packets will be routed through the core network towards the cloud. Figure 8 illustrates that reduction in network load is inversely proportional to number of devices in the platform. A noteworthy observation about network load is as number of devices increases the enhancement decreases, due to the fact that more tuples are generated that cause more network congestion and collision possibility.

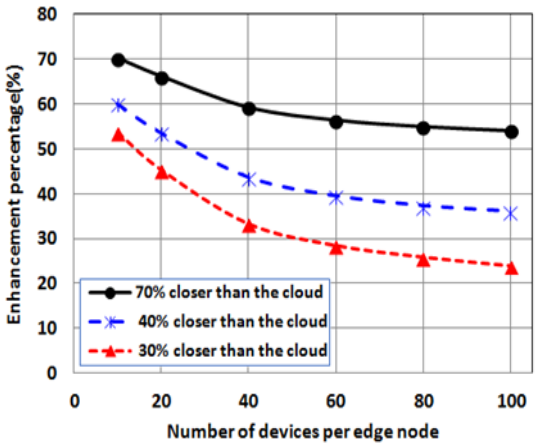

Figure 6. Delay enhancement

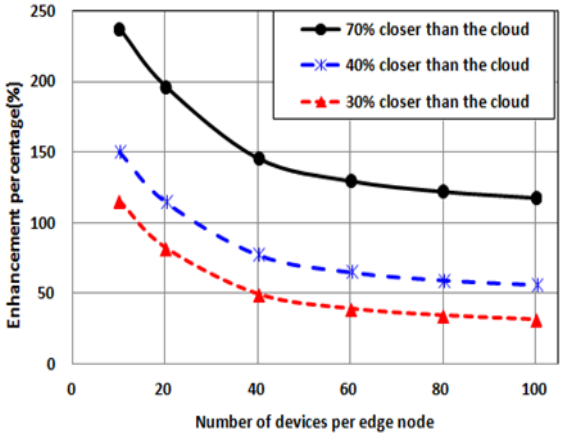

Figure 7. Throughput enhancement

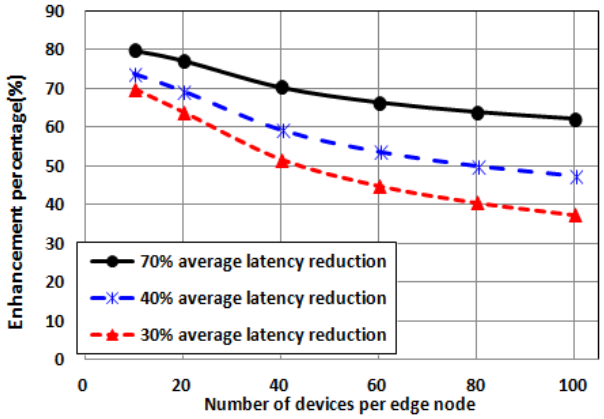

Figure 8. Network load enhancement

6. CONCLUSIONS AND FUTURE WORK

Edge computing has a crucial role for supporting the operation of mobile and Internet of Things (IoT)-based applications. Edge computing supports applications that requires high computing capabilities, and/or fast response demands; also it plays a major role for applications that based on location awareness. This work goal is to improve edge resources utilization; therefore, we proposed a cooperative-hierarchical based approach for modules' placement in edge node and between neighbor-edge nodes, where this approach has a polynomial-time complexity. Also, this work introduced the architecture that facilitates communication between edge nodes. In simulation experiments, *iFogSim* toolkit [38] is used. The proposed module placement algorithm is executed only during application deployment phase. Simulation results show that adopting the proposed approach improves the overall of platform performance by reducing the overall latency, reducing network congestion, and improving throughput. Moreover, the end-user experience is improved. As a future work, we plan to study cases with a dynamic environment and study the effect of rapid changes in the environment, and how this influences end-user experience.

REFERENCES

[1] T. Francis, "A Comparison of Cloud Execution Mechanisms Fog, Edge, and Clone Cloud Computing," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 4646-4653, 2018.
[2] Y. Pradhananga and P. Rajarajeswari, "Tiarrah Computing: The Next Generation of Computing," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 1247-1255, 2018.
[3] P. Prakash, *et al*., "Fog Computing: Issues, Challenges and Future Directions," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 3669-3673, 2017.
[4] K. Qaddoum, *et al*., "Elastic neural network method for load prediction in cloud computing grid," *International Journal of Electrical and Computer Engineering*, vol. 9, pp. 1201-1208, 2019.
[5] K. Sumalatha and M. S. Anbarasi, "A review on various optimization techniques of resource provisioning in cloud computing," *International Journal of Electrical and Computer Engineering*, vol. 9, pp. 629-634, 2019.
[6] F. Bonomi, *et al*., "Fog Computing and Its Role in the Internet of Things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Finland, 2012.
[7] M. Satyanarayanan, *et al*., "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, 2009.
[8] S. Li, *et al*., "5G Internet of Things: A survey," *Journal of Industrial Information Integration*, vol. 10, 2018.
[9] J. Cheng, *et al*., "Industrial IoT in 5G environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, 2018.
[10] Y. C. Hu, *et al*., "Mo-bile-edge computing-introductory technical white paper," *WhitePaper, Mobile-edge Computing (MEC) industry initiative*, 2014.
[11] Y. C. Hu, *et al*., "Mobile edge computing: A key technology towards 5G," *ETSI White Paper*, vol. 11, 2015.
[12] Shi W. and Dustdar S., "The promise of edge computing," *Journal of Computer*, vol. 49, 2016.
[13] C. Y. Chang, *et al*., "MEC architectural implications for LTE/LTE-A networks," *in Proc. ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, New York, 2016.
[14] S. K. Sharma and X. Wang, "Live Data Analytics with Collaborative Edge and Cloud Processing in Wireless IoT Networks," *IEEE Access*, vol. 5, 2017.
[15] T. H. Luan, *et al*., "Fog Computing: Focusing on Mobile Users at the Edge," *arXiv pre-print technical report*, 2015.
[16] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the Internet," *USENIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, 2001.

[17] G. Xylomenos, *et al*., "A survey of informationcentric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, 2014.
[18] Tong L., *et al*., "A hierarchical edge cloud architecture for mobile computing," *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, 2016.
[19] Y. Jararweh, *et al*., "The future of mobile cloud computing: integrating cloudlets and mobile edge computing," *23rd ICT*, Greece, 2016.
[20] Y. Jararweh, *et al*., "SDMEC: software defined system for mobile edge computing," *IEEE IC2EW Workshop*, 2016.
[21] P. Du and A. Nakao, "Application specific mobile edge computing through network softwarization," *5th IEEE International Conference on Cloud Networking (Cloudnet)*, Italy, 2016.
[22] O. Salman, *et al*., "Edge computing enabling the Internet of Things," *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Italy, 2015.
[23] Ravindran R., *et al*., "Towards software defined icn based edge-cloud services," *IEEE 2nd International Conference on Cloud Networking (CloudNet)*, USA, 2013.
[24] X. Li, *et al*., "Adaptive transmission optimization in SDN-based industrial internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 5, 2018.
[25] J. Liu, *et al*., "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Communications Magazine*, vol. 55, 2017.
[26] Mao Y., *et al*., "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys &Tutorials*, vol. 19, 2017.
[27] W. Hu, *et al*., "Quantifying the Impact of Edge Computing on Mobile Applications," *Proceedings of the 7th ACM SIGOPSAsia-Pacific Workshop on Systems*, Hong Kong, 2016.
[28] S. Wang, *et al*., "Dynamic service migration in mobile edge-clouds," *IFIP Networking Conference, New Jersey*, 2015.
[29] M. Aazam and E. N. Huh, "Dynamic resource provisioning through fog micro datacenter," *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), USA*, 2015.
[30] O. Skarlat, *et al*., "Resource provisioning for iot services in the fog," *IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, China, 2016.
[31] A. Kapsalis, *et al*., "A Cooperative Fog Approach for Effective Work-load Balancing," *IEEE Cloud Computing*, vol. 4, 2017.
[32] U. Hunkeler, *et al*., "MQTT-S: A publish/subscribe protocol for Wireless Sensor Networks," *3rd international conference on Communication systems software and middleware and workshops, COMSWARE 2008*, India, 2008.
[33] Y. Xu, *et al*., "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery," *IEEE 8th International Conference on Communication Systems and Networks (COMSNETS), India*, 2016.
[34] K. Gai, *et al*., "Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing," *Journal of Parallel and Distributed Computing*, vol. 111, 2018.
[35] K. Gai, *et al*., "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, 2016.
[36] J. Wang, *et al*., "Fog Computing for Energy-aware Load Balancing and Scheduling in Smart Factory," *IEEE Transactions on Industrial Informatics*, vol. 14, 2018.
[37] K. Gai, *et al*., "Privacy-preserving multi-channel communication in Edge-of-Things," *Future Generation Computer Systems*, vol. 85, 2018.
[38] H. Gupta, *et al*., "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments," *Software: Practice and Experience*, vol. 47, 2017.
[39] R. N. Calheiros, *et al*., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Journal of Software: Practice and experience*, vol. 41, 2011.
[40] M. Shurman and M. Aljarah, "Collaborative execution of distributed mobile and IoT applications running at the edge," *International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2017.

BIOGRAPHIES OF AUTHORS

Maha Aljarah is a M.Sc. student in Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan. Her research interests are IoT and Fog Computing.
Email: mkaljarah16@cit.just.edu.jo

Dr. Mohammad M. Shurman received his B.Sc. degree in Electrical and Computer Engineering from Jordan University of Science and Technology, Jordan, M.Sc. and Ph.D. degrees in Computer Eng.-Wireless Networks from University of Alabama-Huntsville, USA, in 2000, 2003, and 2006, respectively. Currently, he is an associate professor in Network Eng. and Security Dept., JUST, Jordan. His research interests include wireless ad-hoc networks, security, WSNs, network coding, mobile networks, SDN, cognitive radio, 4G and 5G technologies.
Email: alshurman@just.edu.jo

Dr. Sharhabeel Alnabelsi is an associate professor at Computer Engineering Dept. at Al-Balqa Applied University, Amman, Jordan. Also, he is an associate professor in Computer and Networks Eng. dept. at Al Ain University of Science and Technology, UAE. He received his Ph.D. in Computer Engineering from Iowa State University, USA, 2012. Also, he received his M.Sc. in Computer Engineering from The University of Alabama in Huntsville, USA, 2007. His research interests are cognitive radio networks, wireless sensors networks, network optimization, and cloud computing. He is a member of honorary societies including Eta Kappa Nu and Phi Kappa Phi.
Email: alnabsh1@bau.edu.jo, sharhabeel.alnabelsi@aau.ac.ae