**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 13, No. 6, December 2023, pp. 7099~7107
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i6.pp7099-7107 

# Resource allocation for fog computing based on software-defined networks

**Sepideh Sheikhi Nejad[1], Ahmad Khademzadeh[2], Amir Masoud Rahmani[3], Ali Broumandnia[1]**
[1]Department of Computer Engineering, Islamic Azad University South Tehran Branch, Tehran, Iran
[2]Department of Computer Engineering, Research Center ITRC, Tehran, Iran
[3]Department of Computer Engineering Science and Research Branch, Islamic Azad University, Tehran, Iran

**Article Info**

***Article history:***

Received Aug 26, 2022
Revised Mar 15, 2023
Accepted Apr 3, 2023

***Keywords:***

Fog computing
Fog node
Resource allocation
Software defined network
Task mapping

**ABSTRACT**

With the emergence of cloud computing as a processing backbone for internet of thing (IoT), fog computing has been proposed as a solution for delay-sensitive applications. According to fog computing, this is done by placing computing servers near IoT. IoT networks are inherently very dynamic, and their topology and resources may be changed drastically in a short period. So, using the traditional networking paradigm to build their communication backbone, may lower network performance and higher network configuration convergence latency. So, it seems to be more beneficial to employ a software-defined network paradigm to implement their communication network. In software-defined networking (SDN), separating the network's control and data forwarding plane makes it possible to manage the network in a centralized way. Managing a network using a centralized controller can make it more flexible and agile in response to any possible network topology and state changes. This paper presents a software-defined fog platform to host real-time applications in IoT. The effectiveness of the mechanism has been evaluated by conducting a series of simulations. The results of the simulations show that the proposed mechanism is able to find near to optimal solutions in a very lower execution time compared to the brute force method.



***Corresponding Author:***

Ahmad Khademzadeh
Department of Computer Engineering, Research Center ITRC
Tehran, Iran
Email: a.khademzadeh@itrc.ac.ir

## 1. INTRODUCTION

Internet of thing (IoT) comprises many smart devices that are connected via wired or wireless connections and are also connected to a cloud data center. The deployment of a cloud computing data center at the core of the IoT network has advantages such as ubiquitous access, unlimited scalability, and elasticity [1]. However, due to the centralization caused by deploying a central cloud data center in IoT and the geographical distance of cloud data centers from IoT devices, the links connecting the IoT de-vices and cloud data centers may become performance bottlenecks. Such performance bottlenecks can increase the execution latency of computation-al tasks submitted by IoT devices to the cloud data center, making it challenging to host delay-sensitive applications in such an IoT network. To mitigate these challenges, a new paradigm called "fog computing" [2] has been proposed in recent years. The idea behind fog computing is to reduce the average execution latency of tasks by placing a set of fog computing servers between IoT devices and cloud data centers.

If an IoT-based fog computing model is implemented using traditional networking paradigms, the convergence to a new desirable configuration will be time-consuming, making it challenging to quickly adapt the platform to host new services with a short lifespan. As such, it is crucial to adjust the networking paradigm to make it agile enough to update its configuration to handle such services. According to the software-defined networking (SDN) paradigm, to address these challenges and leverage the features of SDNs, it seems to be a promising solution to implement the network of IoT-based fog computing models.

The concept of SDN has been proposed in [3] and has garnered significant attention from both industry and academia [4]. So, using the SDN idea to implement the network IoT-based fog computing model has been considered in several types of research. Therefore, the implementation of an IoT-based fog computing model using the SDN paradigm has been considered in various research studies. Sood *et al.* [3] examined current efforts to merge SDN and IoT. Gupta *et al.* [5] proposed a middleware based on SDN-cloud fog computing. Hakiri *et al.* [6] proposed a novel architecture for controlling wireless fog-based SDN, in order to reduce delay. In [7] studied a software-defined fog computing in IoT architecture for resource management. Misra and Saha [8] studied a greedy heuristic scheme for multi-hop task offloading in IoT-based fog computing via software-defined methods. Misra and Bera [9] proposed mobility-aware task offloading in software-defined vehicular networks to optimize the computational offloading and network latency in vehicular networks. This scheme is based on SDN and has a node selection and task computation phase.

Therefore, due to the advantages of SDN and following the aforementioned research works, in this paper, we consider the platform of software-defined IoT-based fog computing to address the problem of processing delay-sensitive applications on this platform.

- Analyzing the network between fog servers to find all possible paths between every pair of fog servers and indexing them as a hypergraph to facilitate the assigning process.
- Selecting a mapping between the task graph and the constructed hypergraph, leading to the task's lowest execution latency. A set of simulations have been conducted to evaluate the effectiveness of the proposed method, and the proposed method's performance is compared to the exhaustive optimal search method.

The main contribution of this paper is to extend the previously proposed task processing latency models proposed in [6]–[9] to consider the latency of processing tasks with multi-node weighted directed graphs. The necessity of considering such tasks arises from the fact that there may be situations in which a single network fog server cannot handle the submitted task, and the task must be partitioned into dependent sub-tasks. The directed graph of the task would model the dependency between sub-tasks, and the graph nodes would denote each sub-task. Therefore, this graph should be assigned to a connected set of fog servers so that the processing latency of the task falls within an acceptable range according to the timing constraints of the submitted real-time task. In light of this, the proposed task offloading method in this paper is composed of two parts. The first part is similar to previously proposed methods for offloading tasks from IoT devices to fog servers. The second part deals with assigning the task graph to a suitable subset of fog servers.

The problem of assigning the multi-node task graph to the cluster of fog servers can be modeled as a variation of the well-known sub-graph isomorphism problem, which is NP-hard [10]. Thus, the second part of the proposed method is designed based on a greedy approach that achieves optimal solutions with lower execution time than exhaustive optimal search. To this end, the second part of the proposed method takes the following actions: i) finding the critical path in the task graph, ii) analyzing the network between fog servers to find all possible paths between every pair of fog servers and indexing them as a hypergraph to facilitate the assigning process, and iii) selecting a mapping between the task graph and the constructed hypergraph, leading to the task's lowest execution latency.

## 2. RELATED WORK

This section provides an overview of related literature on the task offloading problem in IoT-based fog computing and software-defined fog computing. Specifically, with regard to the main contribution of this paper, which pertains to the mapping of undirected multi-node task graphs to fog servers, a brief review of related works in the field of task graph mapping is also presented. Subsection 2.1 primarily examines research conducted on task offloading in IoT-based fog computing, while subsection 2.2 examines literature addressing task offloading in software-defined fog computing. Finally, subsection 2.3 offers a succinct overview of the concept of task graph mapping.

### 2.1. Task offloading in IoT based fog computing

To address task offloading in the fog computing environment, Sood and Singh in [11], proposed a priority-based resource allocation scheme. Liu *et al.* [12] studied offloading processes in a fog computing system with mobile devices by utilizing queuing theory to form a theoretical foundation for formulating a multi-objective optimization problem to minimize energy consumption, execution delay, and payment cost.

They proposed a task offloading method based on finding the optimal offloading probability and transmitting power for each mobile device. Wang *et al.* [13] proposed a resource management framework equipped with mechanisms for provisioning and auto-scaling edge node resources. Shojafar *et al.* [14] considered the resource scheduling challenges as a part of task offloading in IoT-based fog computing in vehicular networks. Zeng *et al.* [15] proposed an innovative algorithm for scheduling tasks and resource management with minimized task completion time in fog computing based on software-defined embedded systems. Gu *et al.* [16] considered the integration of fog computing and medical cyber-physical devices, and have proposed an algorithm for jointly optimize base station association, task distribution, and virtual machine placement to minimize the cost of this network. Pham-Nguyen and Tran-Minh [17] considered the service deployment problem as a multi-objective optimization that minimizes the overall response time of an application. Huang *et al.* [18] have proposed the task offloading problem in IoT-based fog computing with deep reinforcement learning in single -nodes task graphs.

### 2.2. Software-defined fog platform and task assignment

To address the issue of task offloading in the fog SDN, Bu and Wang [4] proposed a novel networking for edge computing patterns using the idea of software defined networking. Huang *et al.* [18] considered a SDN-based mobile edge computing framework to provide a higher level of data-plane flexibility and programmability. The network deployment and conditions of the proposed framework. In [19] studied a offloading model cooperative software-defined for device-to-device communication in advanced long-term evolution (LTE) networks. Misra and Saha [8] proposed an integer linear programming formulation for the task offloading problem in IoT-based fog computing with a software-defined access network. In addition, Misra and Bera in [9] considered optimizing the computational offloading and network latency in vehicular networks with SDN access networks.

### 2.3. Task graph mapping

Mirza *et al.* [20] proposed a systematic review in the scope of mapping and scheduling data flow graphs in streaming applications. Sugiarto *et al.* [21] presented an efficient mapping strategy for a task graph on a machine based on spiking neural network (SNN) architecture. Simon *et al.* [22] proposed a directed cycling graph scheduling algorithm over multiprocessor system-on-chips intending to minimize energy consumption. Taura and Chien [23] have presented a graph-theoretic formulation of task scheduling problems and have proposed a heuristic algorithm based on their proposed model.

## 3. THE PROPOSED SOFTWARE-DEFINED PLATFORM

We propose a software-defined fog platform as shown in Figure 1, the requests of each IoT device would be submitted to the fog-cloud network through base stations in the form of a multi-nodes weighted directed task graph. The base stations and the fog domains are SDN-enabled and can be monitored and managed by the SDN controller through its southbound Application programming interface (APIs). This module aims to reduce task execution latency by forwarding tasks to proper base stations and fog domains. The problem is formulated as integer programming and is presented below.

The physical network is shown as a $G =< I, V, L >$ where $I$ is the set of IoT devices, $V$ denotes a set of nodes including base stations and fog servers and, $L$ denotes the set of communication links between the nodes is proposed in [21]. The computational capacity of network nodes is denoted by $W = \{ w_1, \dots, w_N\}$ where $w_i$ is the processing capacity of $i^{th}$ node of the network and $N = |V|$. Furthermore, a bandwidth of network links is presented by $B = \{b_1, .., b_M\}$ where $b_j$ is the bandwidth of $j^{th}$ link of network and $M = |L|$. Following this notation, it is implicitly assumed that the base station nodes are seen the same as the fog servers while they have no processing capacity by default.

Let $T = \left\{t_1^1, \dots., t_{R_1}^1, \dots, t_1^K, \dots., t_{R_k}^K\right\}$ be the set of all tasks submitted by IoT devices where $t_s^o$ is the $o^{th}$ task of the $s^{th}$ IoT device, and $R_k$ is the number of tasks submitted by $K^{th}$ IoT device. Each $t_s^o$ is by itself a directed acyclic graph (DAG). So, each task is shown as $t_s^o =< V_s^o, L_s^o >$. $V_s^o$ denotes a set of nodes in each task and $L_s^o$ denotes the set of communication links between the nodes. Each task has a processing requirement and communication requirement, the processing requirements of task nodes is denoted by $P_s^o =< p_{s_1}^o, \dots, p_{s_H}^o >$ where $p_{s_f}^o$ is the processing requirement of $f^{th}$ node of the task and $H = |V_s^o|$. Furthermore, the communication requirement of task links is presented by $C_s^o =< c_{s_1}^o, \dots, c_{s_Z}^o >$ where $c_{s_q}^o$ is the communication requirement of $q^{th}$ link of task and $Z = |L_s^o|$. We calculate the maximum delay taken to process a task. Maximum delay (Mdf) to service a task in a fog domain is expressed as (1):

$$\mathrm{M_{df}} = \mathrm{D_p} + \mathrm{D_t} + \mathrm{D_c} \quad (1)$$

$D_p$ is the sum of all propagation delay, and $D_t$ is the transmission delay, and *Dc* is the multi-node task graph processing delay which includes both queuing delay and multi-node task graph processing delay.

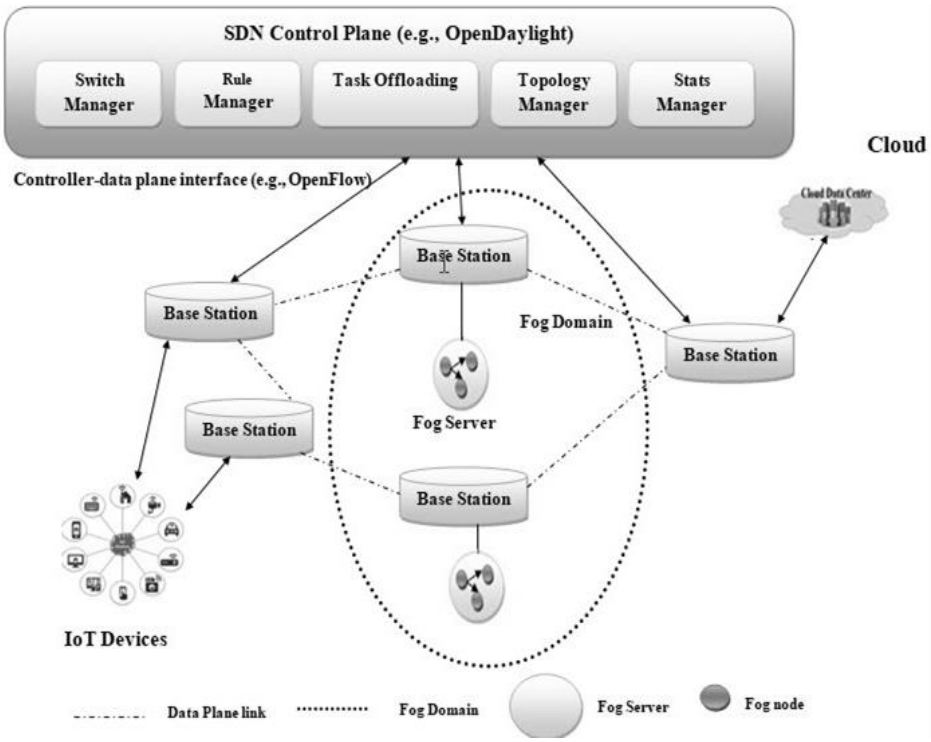



Figure 1. The architecture of the SDN fog platform

The multi-node task graph processing delay is the total time taken by the fog domain to compute a task. Let $x_i^f$ be the mapping parameter indicating the hosting the $f^{th}$ node of the $t_s^o$ task by the $i^{th}$ node of the *G*. Furthermore let $PH_{u,v}$ be the set of all possible paths between nodes $u, v \in G$ . Besides let $y_p^q$ be the parameter indicating the mapping of the $q^{th}$ link of the $t_s^o$ task to the path $p \in PH_{u,v}$ . So, the multi-node task graph processing delay for processing the task $t_s^o$ can be computed as:

$$D_c = \sum_{f\in V_s^o, i\in V, q\in L_s^o} x_i^f . D_{f,i}^{que} + x_i^f . D_{f,i}^{proc} + y_p^q . \sum_{\substack{J\in p, \\ p\in PH_{u,v}, u,v\in G, \\ (x_u^\alpha=1 \text{ and } x_v^\beta=1), q==(\alpha,\beta)}} D_{q,J}^t$$

where $D_{f,i}^{que}$ and $D_{f,i}^{proc}$ are in order the queuing latency and the processing latency of the $f^{th}$ node of the $t_s^o$ task by the $i^{th}$ node of the *G*, and $D_{q,J}^t$ is transmission delay over $J^{th}$ link of the *G* and is member of $p \in PH_{u,v}$ hosting the $q^{th}$ link of the $t_s^o$ the task with starting and ending nodes hosted by $v \in G$.

Therefore, the optimization objective function can be defined as (4)-(10).

$$P: \text{Minimize } M_{df} \tag{4}$$

$$\text{s. t: } x_i^f \in \{0,1\} \tag{5}$$

$$y_p^q \in \{0,1\} \tag{6}$$

$$x_i^f = \begin{cases} 1 & \text{hosting the } f^{th} \text{ node of the } t_s^o \text{ task by the } i^{th} \text{ node of the G} \\ 0 & \text{otherwisw} \end{cases} \tag{7}$$

$$y_p^q = \begin{cases} 1 & \text{mapping of the } q^{th} \text{ link of the } t_s^o \text{ task to the path } p \in PH_{u,v} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\forall \vartheta \in V, \forall t_s^o \ \sum_{\gamma \in V_s^o} p_{s,\gamma}^o x_\vartheta^\gamma \leq w_\vartheta \quad (9)$$

$$\forall J \in B, \forall t_s^o \sum_{\substack{J \in p, \\ p \in PH_{u,v}, u,v \in G, \\ (x_u^\alpha = 1 \text{ and } x_v^\beta = 1), q == (\alpha,\beta)}} c_{s_q}^o y_p^q \leq b_J \quad (10)$$

Constraints (5), (7) means, $x_i^f$ either gets a value of zero or a value of 1. If its value is 1, it means that the $f^{th}$ node of the task $t_s^o$ is mapped to the $i^{th}$ node of the $G$. Constraints (6), (8), means, $y_p^q$ either gets a value of zero or a value of 1. If its value is 1, it means that the $q^{th}$ link of the task $t_s^o$ is mapped to the path $p \in PH_{u,v}$, in which $p$ passes through the $J^{th}$ link of the $G$. Constraint (9) stand for the processing capacity limitation of the $i^{th}$ node of the $G$. Constraint (10) stands for bandwidth capacity limitation of the $J^{th}$ link of the $G$.

## 4. THE PROPOSED ALGORITHM

We introduce a heuristic greedy algorithm called SDN BSA. The proposed algorithm is an adaptation of the BSA algorithm presented in [24]. This algorithm starts by scheduling all the nodes to one fog node in a virtual way. It then improves the schedule by migrating the nodes to other fog nodes.

It should be noted that each link of the task graph may be mapped to a path on the fog domain. To make it possible to use the mapping technique of BSA in our presented problem a preprocessing step should be done on the fog domain topology. This preprocessing step indexes all possible paths between each pair of fog servers in the fog domain. By doing so, a hypergraph of the fog domain topology will be constructed in which each node is a fog server and each link represents a physical path over the fog domain. The paths represented by hypergraph links do not include any duplicate fog servers or physical links. The paths between each pair of fog servers can be found by depth first search (DFS).

Upon receiving a task by a base station, it will be forwarded to the SDN controller for making decisions about its mapping. Benefiting this it can make a central decision about the task mapping. To do this the controller designates a fog server as the "Admin node" of the mapping.

To minimize the overall task execution time, it is required to minimize the execution time of the longest path of the task graph. To do so, a function will scan the task graph and find its longest path. All fog servers will be checked for their available computational resources to host accumulated computational demands of the nodes in the longest path. If there is such a fog server it will be determined as the admin node and all of the longest path nodes will be mapped to this node. If there is not enough room over none of the fog servers to host all of the longest path nodes, a part of the longest path will be mapped to neighbor fog servers regarding their available resource and the delay constraint of the task.

After determining the admin node, the mapping of each task node will be done according to its data dependency on its previous nodes in the task graph and availability of the resources on the fog servers and their connections. While implementing the algorithm, we have a large data-producing parent that the volume of data they send to their child node is the maximum, here it is better to put the child next to these parents to minimize latency. Each fog node also has its computation capacity and bandwidth (communication capacity). Now, based on the selected admin node and capacity of the fog node, the node in the task graph maps to the fog node, after mapping the resources, the mapped value is reduced from the fog node capacity, and then the management module updates the fog node capacity. The mapping algorithm 1 is described as a pseudo-code as:

Algorithm 1: SDN BSA

```
SDN BSA Algorithm:
0. Preprocess the physical network topology and constructed the hypergraph.
1. Partitioning of task graph into sub-tasks
   a. Select Critical Path(CP)
   b. Select (CPN ancestors)
   c. Add CPN near family to CPN
      - Select one of the parents of the first node at CPN, if all parents of the selected
        node are in CPN add selected in CPN. Else, select one of the parents of the
        selected node with the farthest distance from the first node and call this routine
        for the newly selected node in a recursive way.
      - Run i for other CPN nodes.
   d. Add CPN Root cousins to CPN ancestor, CPN Root cousin is a node that is left out of
   CPN after completion of c.
2. Select Admin node in the hypergraph
   a. The admin node in the hypergraph has the most number of links to the other nodes with
   the ability to host task nodes.
3. Assign all CPN ancestors to the admin node
4. Migrate the task nodes on CPN ancestors to adjacent fog servers using the following
   routine:
```

a. For each task node that must be migrated to other fog servers, the following conditions should hold:
*(Start of time node in adjacent fog node-max (start of time node in admin node, the data arrival time of node receive from its parent))>=delay of processing task graph nodes and transmission required data on nodes for forwarding target fog server.*

## 5. RESULT AND PERFORMANCE OF THE METHOD

In this section, a series of simulations have been carried out, and the results of the simulations are presented. These simulations are coded using Python 3.8. A random topology generator is implemented to create the fog node networks and SDN controllers. Additionally, a random task graph generator has been developed for sequential generation of task graphs. All coding runs on a system with 8 GB of RAM and a Core i7 CPU. For each node in the fog network, computation node frequencies of [0.2, 1.5 GHz] and bandwidths of [250 kbps, 54 Mbps] are considered. The transmission rate between the fog nodes is expected to be higher, approximately 100 Mbps, the average packet size [0, 1 KB, 80 KB]. For each task, computation node frequencies of [0.1, 0.5 GHz] and bandwidths of [150 kbps, 10 Mbps] are considered, as per reference [25]. The simulation parameters, such as the fog network size, the values of task node and fog node capacity, and the size of the task graph, are also reported for each experiment.

Simulation 1: the first experiment presents the results of the analysis of the working time of the SDN-BSA algorithm. The effect of the estimation on the algorithm's total working time is explained in the subsequent section. The reported results are then evaluated. The average mapping time plays an important role in the application of SDN-BSA. In this part, the average time of the proposed SDN-BSA algorithm is compared to the comprehensive execution time of the mapping algorithm. As shown in Figure 2, in experiment 1, due to the exponential growth of the average execution time of the comprehensive implementation for the size of the task graph, the two algorithms are implemented in a network of size 3. The parameters used in the fog network and task diagram are shown in Table 1. A series of sequences consisting of 3 tasks each is applied to both algorithms, and the average working time of each algorithm is measured. The size of applied tasks varies from 1 to 3. The fog networks and task graphs are randomly generated.

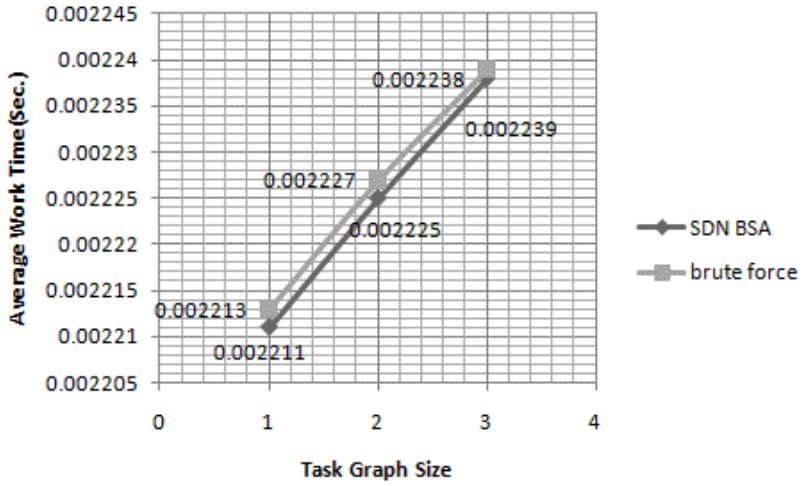



Figure 2. Average of working time

Table 1. The parameters of task graph and fog network

| | Computation capacity | Communicational capacity |
|---|---|---|
| Fog node | [0.2, 1.5 GHz] | [250 kbps, 54 Mbps] |
| | Computation demand | Communicational demand |
| Task node | [0.1, 0.5 GHz] | [150 kbps, 10 Mbps] |

Simulation 2: to evaluate the average working time of the proposed SDN-BSA algorithm for larger samples, we analyze the proposed algorithm on task graphs with sizes between 20 and 140. The computation capacity and communication capacity for the fog network and task graph are according to Table 2. As shown in Figure 3, the average working time increases with an increasing task graph size.

Table 2. The parameters of task graph

| | Size | Computation capacity |
|---|---|---|
| Fog network | related to the selected topology | [0.2, 1.5 GHz] |
| | Size | Computation demand |
| Task graph | [20, 140] | [0.1, 0.5 GHz] |

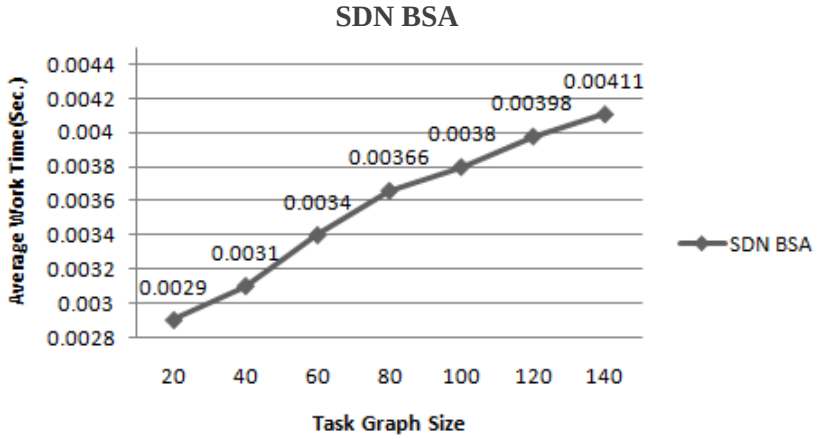



Figure 3. Average of working time

As shown in Figure 4, to demonstrate the superiority of the proposed SDN-BSA algorithm, we compare the total delay obtained from the proposed algorithm with the brute force algorithm. The parameters used in Figure 4 are extracted from Table 3. The figure shows the comprehensive results of the SDN-BSA algorithm.

Simulation 3: to confirm the SDN-BSA, the overall delay obtained by this algorithm is compared to the delay of the exhaustive. Figure 4 shows the results of this simulation using the benchmarks with the parameters listed in Table 3. As shown in Figure 4, the delay gained by algorithm SDN-BSA approves the results of exhaustive.

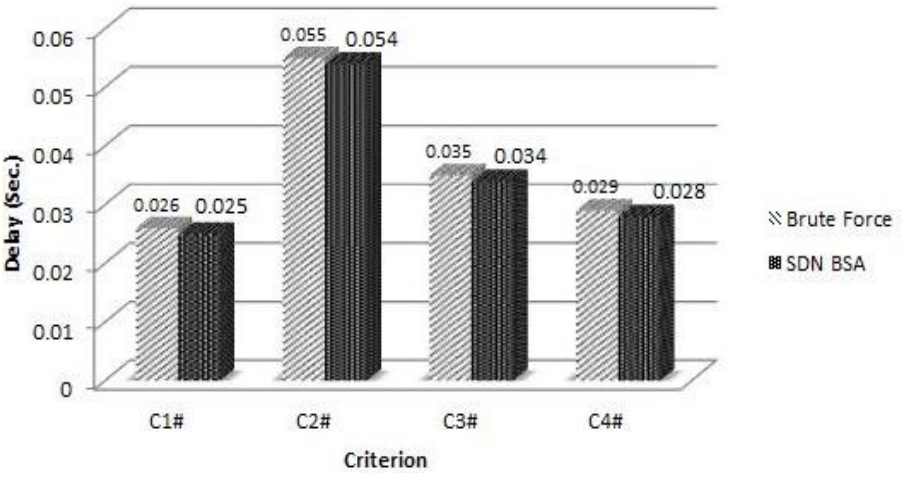



Figure 4. The results of simulation 3

Table 3. The critical parameters in simulation 3

| Criterion | C#1 | C#2 | C#3 | C#4 |
|---|---|---|---|---|
| Min of node computation demand in task graph | 0.1 GHz | 0.2 GHz | 0.3 GHz | 0.1 GHz |
| Max of node computation demand n task graph | 0.4GHz | 0.5GHz | 0.5 GHz | 0.5 GHz |
| Min of link communication demand in task graph | 150 kbps | 160 kbps | 165 kbps | 180 kbps |
| Max of link communication demand in task graph | 5 Mbps | 8 Mbps | 10 Mbps | 10 Mbps |

## 6. CONCLUSION

In summary, this paper presents a new approach for task offloading in the SDN-Fog platform by proposing a formal model to address the delay-sensitive task offloading problem. A brute force technique and a heuristic task assignment technique were proposed and evaluated through simulations. The results show that the proposed heuristic method, based on constructing a hypergraph of the underlying network, is superior to the brute force technique and is capable of reducing delay by 22% and 6% compared to Detour and Soft-VAN, respectively. This research contributes to the field of IoT and fog computing by proposing a new approach for task offloading in SDN-Fog platforms that addresses the challenges of delay-sensitive applications.

## REFERENCES


[1] F. A. Zaman, A. Jarray, and A. Karmouch, "Software defined network-based edge cloud resource allocation framework," *IEEE Access*, vol. 7, pp. 10672–10690, 2019, doi: 10.1109/ACCESS.2018.2889943.

[2] OpenFog Consortium Architecture Working Group, "OpenFog reference architecture for fog computing," *OPFRA001*, vol. 20817, 2017.

[3] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for internet-of-things: a review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, Aug. 2016, doi: 10.1109/JIOT.2015.2480421.

[4] C. Bu and J. Wang, "Computing tasks assignment optimization among edge computing servers via SDN," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1190–1206, May 2021, doi: 10.1007/s12083-021-01081-x.

[5] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh, "SDFog: a software defined computing architecture for QoS aware service orchestration over edge devices," *arXiv preprint arXiv: 1609.01190*, Sep. 2016.

[6] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing wireless fog networks using software-defined networking," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, Oct. 2017, pp. 1149–1156, doi: 10.1109/AICCSA.2017.9.

[7] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for IoT," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, Jan. 2017, doi: 10.1007/s11277-016-3845-0.

[8] S. Misra and N. Saha, "Detour: dynamic task offloading in software-defined fog for IoT applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, May 2019, doi: 10.1109/JSAC.2019.2906793.

[9] S. Misra and S. Bera, "Soft-VAN: mobility-aware task offloading in software-defined vehicular network," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2071–2078, Feb. 2020, doi: 10.1109/TVT.2019.2958740.

[10] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," *Journal of Graph Algorithms and Applications*, vol. 3, no. 3, pp. 1–27, 1999, doi: 10.7155/jgaa.00014.

[11] S. K. Sood and K. D. Singh, "SNA based resource optimization in optical network using fog and cloud computing," *Optical Switching and Networking*, vol. 33, pp. 114–121, Jul. 2019, doi: 10.1016/j.osn.2017.12.007.

[12] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, Feb. 2018, doi: 10.1109/JIOT.2017.2780236.

[13] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: a framework for Edge NOde resource management," *IEEE Transactions on Services Computing*, vol. 13, no. 6, pp. 1086–1099, 2020, doi: 10.1109/TSC.2017.2753775.

[14] M. Shojafar, N. Cordeschi, and E. Baccarelli, "Energy-efficient adaptive resource management for real-time vehicular cloud services," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 196–209, Jan. 2019, doi: 10.1109/TCC.2016.2551747.

[15] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016, doi: 10.1109/TC.2016.2536019.

[16] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, Jan. 2017, doi: 10.1109/TETC.2015.2508382.

[17] H.-N. Pham-Nguyen and Q. Tran-Minh, "Dynamic resource provisioning on fog landscapes," *Security and Communication Networks*, pp. 1–15, May 2019, doi: 10.1155/2019/1798391.

[18] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing," in *Machine Learning and Intelligent Communications*, Springer International Publishing, 2018, pp. 33–42.

[19] Y. Cui *et al.*, "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1746–1760, Jun. 2017, doi: 10.1109/TNET.2017.2650964.

[20] U. M. Mirza, M. A. Arslan, G. Cedersjo, S. M. Sulaman, and J. W. Janneck, "Mapping and scheduling of dataflow graphs-a systematic map," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Nov. 2014, pp. 1843–1847, doi: 10.1109/ACSSC.2014.7094787.

[21] I. Sugiarto, P. Campos, N. Dahir, G. Tempesti, and S. Furber, "Optimized task graph mapping on a many-core neuromorphic supercomputer," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2017, pp. 1–7, doi: 10.1109/HPEC.2017.8091066.

[22] B. Simon, J. Falk, N. Megow, and J. Teich, "Energy minimization in DAG scheduling on MPSoCs at run-time: theory and practice," *arXiv preprint arXiv:1912.09170*, Dec. 2019.

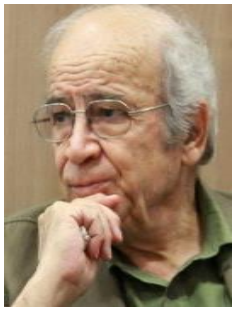
[23] K. Taura and A. Chien, "Heuristic algorithm for mapping communicating tasks on heterogeneous resources," in *Proceedings of the Heterogeneous Computing Workshop, HCW*, 2000, pp. 102–115, doi: 10.1109/hcw.2000.843736.

[24] I. Ahmad and Y. K. Kwok, "On parallelizing the multiprocessor scheduling problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 4, pp. 414–432, 1999, doi: 10.1109/71.762819.

[25] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via *F*og-2- *F*og collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, Nov. 2019, doi: 10.1016/j.future.2019.05.015.

## BIOGRAPHIES OF AUTHORS

**Sepideh Sheikhi Nejad** received her B.S. degree in Computer Engineering from Azad University, Ashtian, Iran, in 2006 and her M.S. degree in Computer Software Engineering from University of Azad, south branch, Tehran, Iran, in 2010. She is currently a Ph.D. candidate at the University of Azad, south branch. Fog computing and software defined networking are her major fields of research. She can be contacted at email: s.sheikhynejad9834@gmail.com.



**Ahmad Khademzadeh** received his B.Sc. degree in Applied Physics from Ferdowsi University, Mashhad, Iran, in 1969 and his M.Sc. and Ph.D. degrees respectively in Digital Communications and Information Theory and Error Control Coding from the University of Kent, Canterbury, UK. He is currently the Head of Education, National and International Scientific Cooperation Department Affairs, and in the meantime the head of Post Graduate Department at the ICT Research Institute (ITRC). He was the head of the Test Engineering Group and the director of the Computer and Communication Department at ITRC. Dr. Khademzadeh is the chair of the IEEE Iran Section Standards Committee and also a lecturer at Tehran Universities. He is a committee member of the Iranian Electrical Engineering Conference Permanent Committee. He can be contacted at email: a.khademzadeh@itrc.ac.ir.

**Amir Masoud Rahmani** received his B.S. in Computer Engineering from Amir Kabir University, Tehran, in 1996, the M.S. in Computer Engineering from Sharif University of Technology, Tehran, in 1998, and the Ph.D. degree in Computer Engineering from IAU University, Tehran, in 2005. Currently, he is a Professor in the Department of Computer Engineering at IAU University. He is the author/co-author of more than 150 publications in technical journals and conferences. His research interests are in the areas of distributed systems, ad hoc and wireless sensor networks, and evolutionary computing. He can be contacted at email: rahmani@srbiau.ac.ir.

**Ali Broumandnia** received the B.Sc. degree from the Isfahan University of Technology in 1991, M.Sc. degree from Iran University of Science and Technology in 1995, both in hardware engineering, and a Ph.D. degree in computer engineering from Tehran Islamic Azad University-Science and Research Branch in 2006. From 1993 through 1995, he worked on intelligent transportation control with image processing and designed the Automatic License Plate Recognition for Tehran Control Traffic Company. He has published over 30 computer books, journals, and conference papers. He is interested in Persian/Arabic character recognition and segmentation, Persian/Arabic document segmentation, medical imaging, signal and image processing, and wavelet analysis. He is a reviewer of some international journals and conferences. He can be contacted at email: broumandnia@gmail.com.