IJICTR
International Journal of Information & Communication Technology Research

Volume 15- Number 4 – 2023 (41 -52)

Resource Allocation Optimization in Fog Architecture Based Software-Defined Networks

Sepideh Sheikhi Nejad
Department of Computer Engineering,
Islamic Azad University South Branch,
Tehran, Iran
s.sheikhynejad9834@gmail.com

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

Amir Masoud Rahmani
Future Technology Research Center,
National Yunlin University of Science and Technology,
Taiwan
rahmani74@yahoo.com

Ali Broumandnia
Department of Computer Engineering,
Islamic Azad University South Branch,
Tehran, Iran
broumandnia@gmail.com

Received: 30 July 2022 – Revised: 25 December 2022 - Accepted: 2 March 2023

Abstract—**As a growing of IoT devices, new computing paradigms such as fog computing are emerging. Fog computing is more suitable for real-time processing due to the proximity of resources to IoT layer devices. Service providers must dynamically update the hardware and software parameters of the network infrastructure. Software defined network (SDN) proposed as a new network paradigm, whose separate control layer from data layer and provides flexible network management. This paper presents a software-defined fog platform to host real-time applications in IoT. Then, we propose a novel resource allocation method. This method involves scheduling multi-node real-time task graphs over the fog to minimize task execution latency. The proposed method is designed to benefit the centralized structure of SDN. The simulation results show that the proposed method can find near to optimal solutions in a very lower execution time than the brute force method.**

Keywords: Software-defined network, fog computing, Multi-nodes weighted directed task graph, Task assigning, task offloading

Article type: Research Article

Publisher: ICT Research Institute

* Corresponding Author

[DOI: 10.61186/itrc.15.4.41]

I. INTRODUCTION

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 the geographical distance of cloud data centers from IoT devices, the links connecting the IoT devices and cloud data centers may become performance bottlenecks. Such performance bottlenecks can increase the execution latency.

To mitigate these challenges, a new paradigm called "fog computing" (2) has been proposed in recent years. In a fog-based network, each IoT device is connected to local computational domains, each comprising a set of computational nodes or fog servers. By deploying the computational resources at the network's edge, it becomes possible to offload tasks to the fog servers near IoT devices, reducing the average round-trip time compared to the original IoT architecture. 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), (5). The primary advantage of SDN in comparison to traditional networks is its ability to manage the data forwarding process in a logically centralized manner. Therefore, the implementation of an IoT-based fog computing model using the SDN paradigm has been considered in various research studies. Sood et al. (4) examined current efforts to merge SDN and IoT and noted the benefits of such a combination for information acquisition, analysis, decision-making, scalability, and security in IoT. Gupta et al. (6) proposed a middleware based on SDN-cloud fog computing that provides services to the heterogeneous fog infrastructure and enables applications to orchestrate fog services while considering end-to-end Quality of Service (QoS) requirements. This article aims to extend an integrated system to an SDN-cloud-fog-based approach. Hakiri et al. (7) proposed a novel architecture for controlling wireless fog-based SDN, in order to reduce delay and enable suitable load-balancing among fog devices. The proposed scheme in (7), the SDN controller, combined both wireless routing protocols and OpenFlow to collect values from the wireless devices to facilitate optimal path selection among the wireless fog nodes. Tomovic et al. (8) proposed a software-defined fog computing architecture for IoT resource management to improve the latency of an IoT network. The authors of (8) highlighted the advantages of the SDN-fog interplay in terms of network scalability, real-time data delivery, and mobility. Misra et al. (9) studied a greedy heuristic scheme for multi-hop task offloading in IoT-based fog computing via software-defined methods. Additionally, Misra et al. in (10) 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 .Rahimi et al. (30) proposed an effective solution for traffic management and dynamic allocation of radio resources in 5G networks based on the use of SDN in the fog architecture of radio access networks with the aim of reducing energy consumption. This method increases user satisfaction in performing real-time tasks. 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. To this end, a novel method is proposed to take advantage of SDN to collect network information using a Southbound API, which relies on the overall view of the network and offloads delay-sensitive tasks for processing to reduce task processing latency and meet the timing constraints of the submitted real-time application. The proposed method achieves this goal by minimizing different parameters of the task processing latency. The main contribution of this paper is to extend the previously proposed task processing latency models proposed in (7), (8), (9), (10) 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.

Based on this, we propose a delay model, which includes the following parameters:

a) The sum of all propagation delays

b) The sum of all transmission delays

c) Queuing delay

d) Multi-node task graph processing delay

As stated earlier, the proposed method aims to reduce the task processing latency parameters a, b, and c, building upon previous proposed task offloading techniques. The second part of the proposed method addresses the last parameter of the delay model, which is one of the main contributions of this paper. 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 (11). 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

[DOI: 10.61186/itrc.15.4.41]

end, the second part of the proposed method takes the following actions:

Finding the critical path in the task graph

- 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 rest of this paper is organized as follows:

The second section of the paper is dedicated to a review of related works. The proposed Software-defined fog platform and formal mathematical expression of the platform are presented in the paper's third section. In Section IV, the proposed algorithm is presented. In Section V, the results of the performance evaluation of the proposed method are reported. Finally, the concluding notes and future directions of extending the presented work are covered in Section VI.

II. 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 A primarily examines research conducted on task offloading in IoT-based fog computing, while Subsection B examines literature addressing task offloading in software-defined fog computing. Finally, Subsection C offers a succinct overview of the concept of task graph mapping.

A. Task Offloading in IoT based fog computing

To address task offloading in the fog computing environment, various techniques have been proposed in the literature. Sood and Si (12) proposed a priority-based resource allocation scheme for submitted jobs and a deadlock-removing method in IoT-based fog computing with optical connections to minimize the response time of these jobs. Liu et al. (13) 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. (14) proposed a resource management framework equipped with methods for provisioning and auto-scaling edge node resources. Shojafar et al. (15) considered the resource scheduling challenges part of task offloading in IoT-based fog computing in vehicular networks. They presented an energy-efficient adaptive resource scheduler for fog Network Centers in vehicular networks. The goal of their work was to apply the TCP/IP connections' locally measured states to maximize the overall communication-plus-computing energy efficiency while meeting the application-induced hard Quality of Service (QoS) requirements on the minimum transmission rates and maximum delays and delay-jitters. Zeng et al. (16) 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. (17) considered the integration of fog computing and medical cyber-physical devices and proposed an algorithm for jointly optimizing base station association, task distribution, and virtual machine placement to minimize the cost of this network. Nguyen et al. (18) considered the service deployment problem a multi-objective optimization that minimizes the overall response time of an application with awareness of network usage and server usage to prove the effectiveness of their proposed foggy service deployment strategy. As previously stated, the works (12)-(18) have considered the task offloading problem in IoT-based fog computing, but none of them have addressed the problem of scheduling multi-node task graphs as a part of the task offloading problem. However, scheduling multi-node task graphs has been considered in some works, such as (19). Bitam et al. in (19) introduced a meta-heuristic approach based on swarm optimization for scheduling multi-node jobs over IoT-based fog networks. The tasks considered in (19) are assumed to be a set of independent sub-tasks, so the work presented in (19) does not address the problem of scheduling tasks composed of dependent sub-tasks modeled as multi-node directed graphs.

B. Software-defined fog platform and task offloading

To address the issue of task offloading in the fog SDN, Chao Bu et al. [20] proposed a novel networking approach for edge computing patterns using the idea of SDN. In this platform, tasks are assigned based on the network's global view provided by a central SDN controller. The logically centralized controller is constructed through collaboration between multiple, physically distributed edge computing servers. Using this novel networking approach, a more efficient method was proposed to optimize task assignment and minimize task processing delay. They proposed a model for task assignment among edge computing servers via SDN. A. Huang et al. (21) considered an SDN-based mobile edge computing framework to provide higher data-plane flexibility and programmability. The network deployment and conditions of the proposed framework (21) can be reconfigured at runtime to improve network latency. Cui et al. in (22) proposed a software-defined cooperative offloading model for device-to-device communication in advanced LTE networks. Furthermore, they proposed a new online task scheduling algorithm (22) to minimize the energy consumption of a mobile device. Misra et al. (9) proposed an Integer Linear Programming formulation for the task offloading problem in IoT-based fog computing with a software-defined access network. They also suggested a greedy heuristic task offloading algorithm to solve the problem of delay, energy consumption, multi-hop paths, and dynamic network conditions such as link utilization and SDN rule

[DOI: 10.61186/itrc.15.4.41]

capacity. Additionally, Misra et al. (10) considered optimizing computational offloading and network latency in vehicular networks with SDN access networks. Alomari et al. in (31) proposed a comprehensive study on the role of software defined networks (SDN) in the ease of managing cloud and fog computing networks, improving network performance, reducing energy consumption, and reducing latency. In this study, algorithms, architecture, simulation environment and data have been investigated. A comparison between the proposed method in this paper and some of the related works is presented in Table I.

C. Task Graph Mapping

As previously stated, the proposed task offloading method in this paper deals with offloading multi-node task graphs. Therefore, it includes a task graph mapping component that maps the task graph to a subset of fog servers. This subsection covers research works that address this problem. To address this issue, Mirza et al. (23) proposed a systematic review of the mapping and scheduling of data flow graphs in streaming applications. They considered the problem of executing these applications over a multiprocessor platform to efficiently implement latency, throughput, power, and energy consumption. Saguaro et al. (24) presented an efficient mapping strategy for a task graph on a machine based on Spiking Neural Network Architecture. This strategy is suitable for mapping large task graph networks and reduces communication latency. Paone et al. (25) introduced a task-mapping method to maximize the overall application throughput by utilizing concurrency in the task graph. Simon et al. (26) proposed a directed cycling graph scheduling algorithm over multiprocessor system-on-chips to minimize energy consumption. Taura et al. (27) presented a graph-theoretic formulation of task scheduling problems and proposed a heuristic algorithm based on their proposed model. The algorithm proposed in (27) is designed to run the entire data-processing pipeline with good throughput regarding parallelism and communication messages

III. THE PROPOSED SOFTWARE-DEFINED PLATFORM

We propose a software-defined fog platform as shown in Figure 1, consisting of a set of base stations, a set of IoT devices (NI), a set of fog nodes (NF) with a standard structure proposed in [24], and a set of cloud nodes (NC). In this platform, IoT devices act as clients of fog systems. Each IoT device uses a communication protocol (such as IEEE 802.15.4, Wi-Fi, Bluetooth, MQTT, etc.) to interact with base stations. The requests of each IoT device are submitted to the fog–cloud network through base stations in the form of a multi-node weighted directed task graph. Each fog domain comprises several distributed fog servers that are ideally located "next" to data sources (IoT devices). This set of fog servers refines and processes the request submitted by the IoT devices. The fog may reduce the amount of data transmitted to the cloud data center by preparing these data. The base stations and the fog domains are SDN-enabled and are monitored and managed by the SDN controller through its southbound APIs.

The SDN controller acts in a centralized way based on the network's global state. The SDN controller can collect global information of the IoT-based fog computing network, including the processing load, network traffic, available processing and communication resources, and delay of each fog node and link. The central controller unit, or SDN, can make logical and correct decisions about the transmission of each offloaded task graph and, accordingly, place the suitable flow rules in the active SDN base stations and fog servers.

The logic of handling task offloading requests is implemented as an application in the SDN controller, denoted as the task offloading module in Fig. 1. This module aims to reduce task execution latency by forwarding tasks to proper base stations and fog domains.

As stated earlier, it is assumed that some IoT devices may submit tasks with a non-reducible multi-node graph structure. To deal with such tasks, several fog nodes must act collaboratively. These fog servers may be clustered in some logical/physical fog domains accessible through one or more base stations. Then, IoT devices that want to offload their tasks submit their requests to their nearest base station. After that, the SDN controller decides to send the task to the appropriate fog server based on the features and priority of the submitted request and the network's global state.

Besides, Fig. 2 shows the sequence diagram of the task offloading process in the proposed platform. In this diagram, an IoT device submits a task to a base station first. After that, the base station forwards the submitted task to the SDN controller. The task offloading module determines the suitable fog nodes, the paths between them, and the paths connecting the fog domain to the cloud data center to host the task to reduce the task execution latency. There may be some situations in which, regarding the processing requirements of the task graph nodes, it is required to partition the task graph and host some of its nodes in a cloud data center while the fog servers would handle other nodes of the task graph. In the end, the result of executing task graph nodes is aggregated in the fog domain, and finally, the computation results are forwarded to the IoT device.

[DOI: 10.61186/itrc.15.4.41]

TABLE I. COMPARISON OF OUR PROPOSED WITH PREVIOUS WORK

Related work	Year	Delay	SDN	Rule Capacity	Task Assignment	Resource Allocation	Task Graph with weight node & link	Multi-Task nodes in task graph	Physical Graph with weight node & link	IoT	Fog computing	Cloud computing
Sood & Singh.[4]	2021	√	×	×	×	√	×	×	×	√	√	√
Liu et al.[13]	2018	√	×	×	×	√	×	×	×	√	√	√
Wang et al.[14]	2018	√	×	×	×	√	×	×	×	√	√	√
Shojafar et al.[15]	2019	√	×	×	×	√	×	×	×	√	√	√
Zeng.[16]	2016	√	√	√	√	√	×	×	×	√	√	√
Gu.[17]	2017	×	×	×	×	√	×	×	×	√	√	√
Nguyen et al.[18]	2019	×	×	×	√	√	×	×	×	√	√	√
Chao Bu et al.[5]	2021	√	√	√	√	×	×	×	×	×	√	×
Cui et al.[22]	2021	√	×	×	√	×	√	×	×	×	×	×
Bu et al.[20]	2017	×	×	×	√	×	√	×	×	×	×	×
Misra & Saha[9] (Detour)	2019	√	√	√	√	×	×	×	×	√	√	×
Misra & Bera[10] (Soft-VAN)	2020	√	√	√	√	×	×	×	×	×	√	×
SDN-BSA (Proposed platform)	2021	√	√	√	√	√	√	√	√	√	√	√

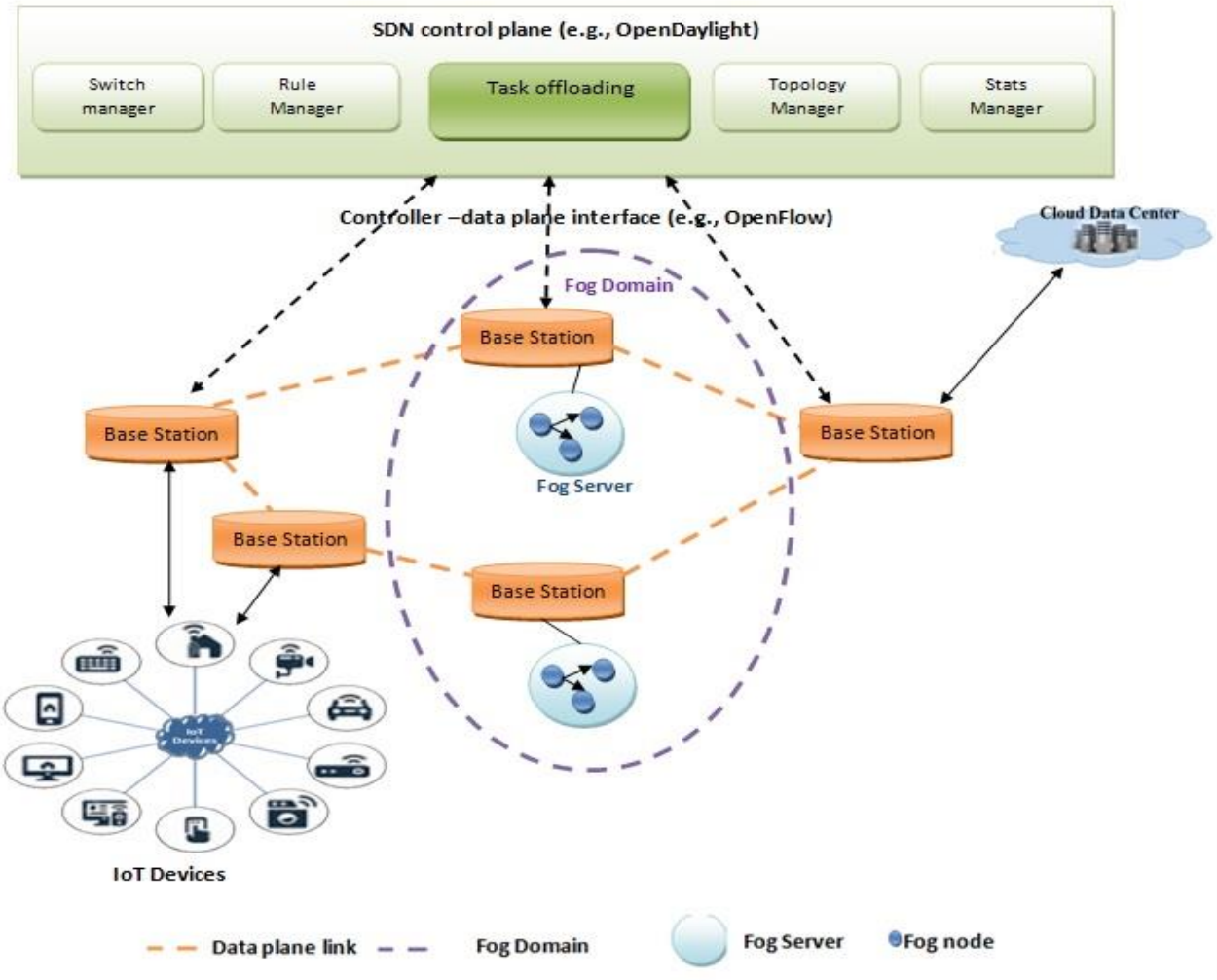

Figure 1. The architecture of the SDN fog platform

[DOI: 10.61186/itrc.15.4.41]

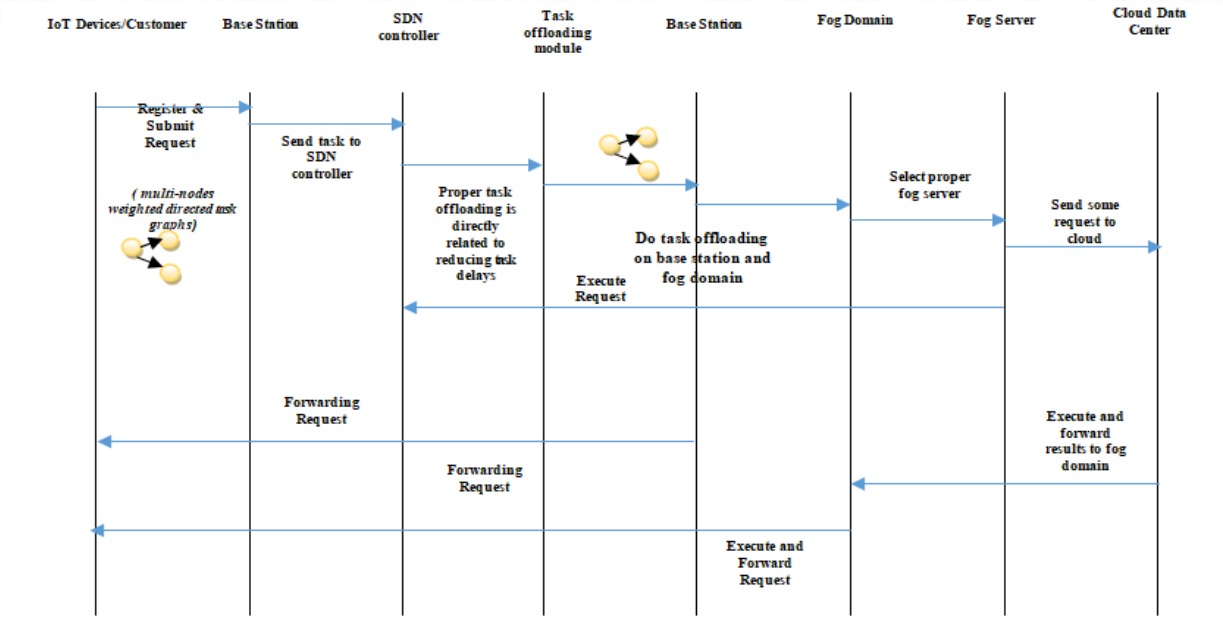

Figure 2. Sequence diagram of the proposed method

TABLE II. SUMMARY OF KEY NOTATIONS

Notation	Definition
G = <I, V, L>	Physical network with the node-set V and link set L
$t_s^o =< V_s^o, L_s^o >$	The task graph with a set of task node V_s^o and link L_s^o
w_i	Processing capacity of i^{th} node of the network
b_j	The bandwidth of j^{th} link of network
$P_{s_i}^o$	Processing requirement of i^{th} node of the task
$C_{s_j}^o$	Communication requirement of j^{th} link of task
D_p	The sum of all propagation delays
D_t	The sum of all transmission delays
D_c	The multi-node task graph processing delays
$D_{f,i}^{que}$	The queuing latency of the f^{th} node of the t_s^o task by the i^{th} node of the G
$D_{f,i}^{proc}$	The processing latency of the f^{th} node of the t_s^o task by the i^{th} node of the G
CP	A critical path of the task graph
CPN ancestors	Task node series
CPN near family	The internal node in the task graph
CPN Root cousin	External node

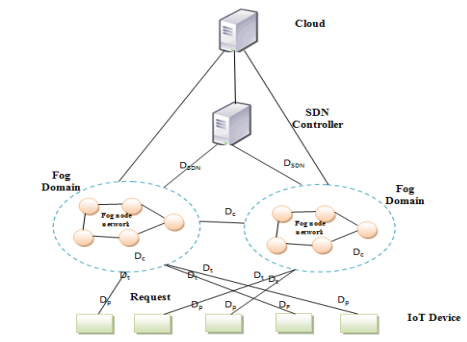

Figure 3. Undirected diagram of SDN fog platform

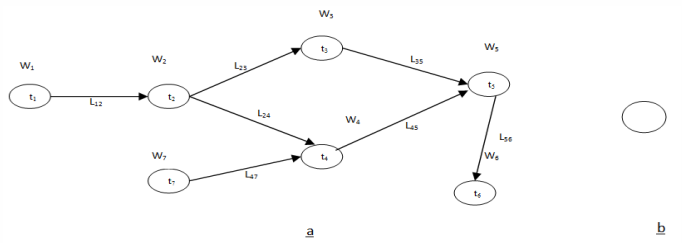

Figure 4. (a) shown a multi-nodes weighted task graph, $t_s^o =< V_s^o, L_s^o >$, and (b) shown the single task graph

[DOI: 10.61186/itrc.15.4.41]

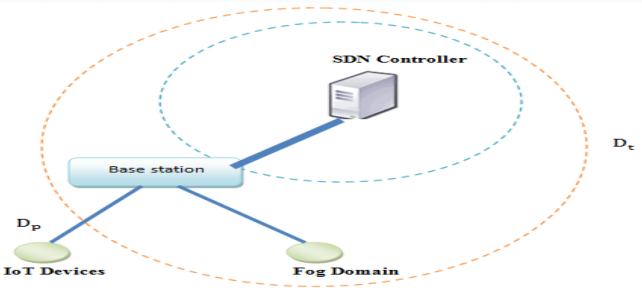

Figure 5. Transmission delay in SDN fog platform

A.delay objective function

The problem of reducing task offloading latency in the proposed software-defined fog platform is formulated as integer programming and is presented in figure3. The key notations are summarized in Table II.

The physical network is modeled as a graph 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. 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 having no processing capacity by default.

Let $T = \{t_1^1, \dots., t_{R_1}^1, \dots, t_1^K, \dots., t_{R_k}^K\}$ 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 are 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$Z = |L_s^o|$.

Figure 4.a shows a multi-nodes weighted directed task graph, and Figure 4 b. shows a single task graph used by previous researches done in the field.

We calculate the maximum delay taken to process a task. Maximum delay (M_{df}) to service a task in a fog domain is expressed as follows:

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

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

Propagation delay is the time required to transmit all data packets of a task over a physical link from an IoT device and a base station.

Transmission delay is when a base station task is taken to a fog domain to transmit the data packets over the fog node network. It should be noted that D_t is the maximum delay between a fog domain and a base station plus the maximum delay which the base station should wait to get a response from the SDN controller. As shown in figure 6, the maximum SDN response delay is denoted by D_c, as a part of D_t. Since, upon receiving the first packet of the task to a base station, it will be forwarded to the SDN controller to find out how the task should be processed. The SDN response delay would be nonzero for the first packet of the task, but it would be zero for the following packet of the task.

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 follows:

$$D_c = \sum_{f \in V_s^o, i \in V, q \in L_s^o} x_i^f . D_{f,i}^{que} + x_i^f . D_{f,i}^{proc} + y_p^q . \sum_{\substack{J \in p, \\ p \in PH_{u,v}, u,v \in G, \\ (x_u^\alpha = 1 \text{ and } x_v^\beta = 1), q == (\alpha,\beta)}} D_{q,J}^t \quad (2)$$

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 task with starting and ending nodes hosted by $v \in G$.

B.The mathematical and formal expression of the problem

We address the Integer programming problem with achieving to minimize delay taken to process a task (M_{df}). Therefore, the optimization objective function

[DOI: 10.61186/itrc.15.4.41]

can be defined as follows:P:

Minimize M_{df} (3)

s. t: $x_i^f \in \{0,1\}$ (4)

$y_p^q \in \{0,1\}$ (5)

$$x_i^f = \begin{cases} 1 & \text{hosting the } f^{th} \text{ node of the } t_s^o \text{ task by the } i^{th} \text{ node of the G} \\ 0 & otherwise \end{cases} \quad (6)$$

$$y_p^q = \begin{cases} 1 & \text{mapping of the } q^{th} \text{ link of the } t_s^o \text{ task to the path p} \in PH_{u,v} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$\forall \vartheta \in V, \forall t_s^o \ \sum_{\gamma \in V_s^o} p_{s,\gamma}^o x_\vartheta^\gamma \leq w_\vartheta (8)$$

$$\forall J \in B, \forall t_s^o \sum_{\substack{J \in p, \\ p \in PH_{u,v}, u,v \in G, \\ (x_u^\alpha = 1 \text{ and } x_v^\beta = 1), q == (\alpha,\beta)}} c_{s_q}^o y_p^q \leq b_J (9)$$

Constraints (4, 6) 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 (5, 7), 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 pathp $\in PH_{u,v}$, in which p passes through the J^{th} link of the G.

Constraint (8) stand for the processing capacity limitation of thei^{th} node of the G.

Constraint (9) stands for bandwidth capacity limitation of the J^{th} link of the G.

TABLE III. COMPARISON OF OUR PROPOSED PLATFORM WITH PREVIOUS SDN PLATFORM

Algorithm name Parameters	**SDN-BSA (Proposed platform)**	**Soft-VAN[10]**	Detour[9]
Objective Function	Accomplishment delayed sensitive tasks	Minimize task computation delay	Reducing the average delay and energy consumption
Definition of delay	a)sum of all propagation delays b)sum of all transmission delays c) Queuing delay d) Multinode task graph processing delay	a)uploading delay transmission delay, propagation delay queuing delay processing delay b)downloading delay propagation delay and transmission delay	a) time to transmit the data to the associated access point b) propagation delay from access point to the fog node c) queuing delay at fog node d) task execution time at the fog nodes. Calculate the delay as a weighted average of the local delay or send it to the fog node
Type of Task Graph	directed acyclic graph(DAG) A series of tasks are dependent on each other, and the first one has to be done and the other then the sequence is different.	Single node - undirected graph	Single node - undirected graph
Physical Network	G =<I,V,L> 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. The computational capacity of network nodes is denoted by W, and the bandwidth of network links is presented by B.	G = (N; L), where N and L denote the set of all RSUs and links between the RSUs	directed graph G = (A∪F; L) where L denotes the set of links between access points(A) and fog nodes (F)

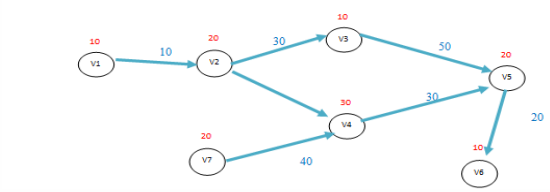

Figure 6. An example of the task graph

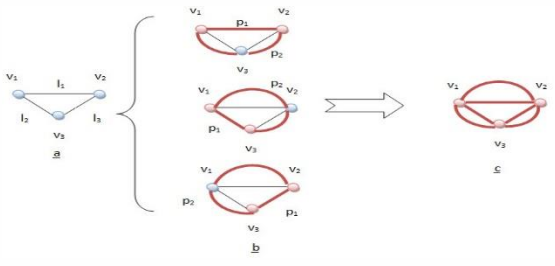

Figure 7. (a) shown a physical graph, $G =< V, L >$, (b) shown the paths between two nodes of the physical graph, and (c) shown the hypergraph

[DOI: 10.61186/itrc.15.4.41]

IV. THE PROPOSED ALGORITHM

The problem modeled by (4), is NP-hard. So, it is not possible to find the optimal solution in polynomial time. Regarding this, we introduce a heuristic greedy algorithm called SDN-BSA. The proposed algorithm is an adaptation of the BSA algorithm presented in [29]. 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. The SDN-BSA handles the tasks submitted by IoT devices which are multi-nodes weighted directed graphs. A sample of these graphs is shown in figure 6. 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. 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) with O (N+M) time complexity. Constructing hypergraph will be done by Task Offloading application at SDN controller and would not affect the actual topology of the physical network. The capacity of hypergraph nodes is equal to their counterparts at the actual fog domain. The capacity of hypergraph links is equal to the capacity of the physical link, with the lowest capacity between all physical links forming their counterpart paths.

Upon receiving a task by a base station, it will be forwarded to the SDN controller for making decisions about its mapping. SDN controller has a holistic view of the network topology and state. Benefiting this, it can make a central decision about the task mapping. The controller designates a fog server as the "Admin node" of the mapping to do this. The procedure of appointing a fog server as the admin node will be covered in the sequel.

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 any 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 the availability of the resources on the fog servers and their connections. The mapping algorithm is described as a pseudo-code as follows:

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**
  - **i. 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 recursive. If two parents have the same distance from the selected node, selected the parent with a smaller distance from the exit node.**
  - **ii. 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 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.

While implementing the algorithm, we have a large data-producing parent whose 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. Then the management module updates the fog node capacity.

New computational capacity of fog node =(old computational capacity of fog node) – (computational demand of task node). The new communicational capacity of fog node =(old communicational capacity of fog node) – (communicational demand of task node)

TABLE IV. THE PARAMETERS OF BENCHMARKS IN SIMULATION 1,2

	Size	Computation capacity	Communicational capacity
fog network	[25,160]	[0.2 GHz , 1.5 GHz]	[250kbps , 54Mbps]
	Size	Computation demand	Communicational demand
Task graph	[20,140]	[0.1 GHz,0.5GHz]	[150kbps, 10Mbps]

[DOI: 10.61186/itrc.15.4.41]

TABLE V. 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

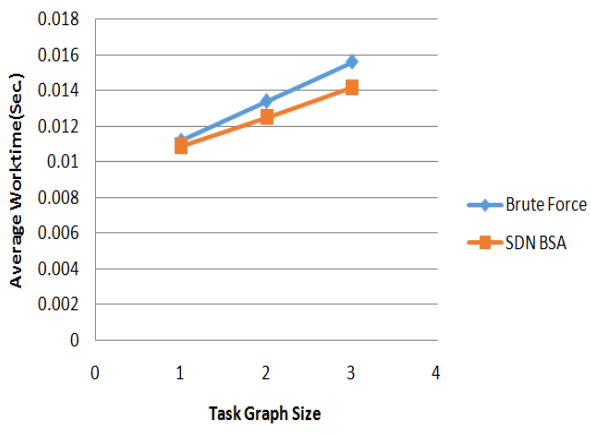

Figure 8. Average of working time in simulation1

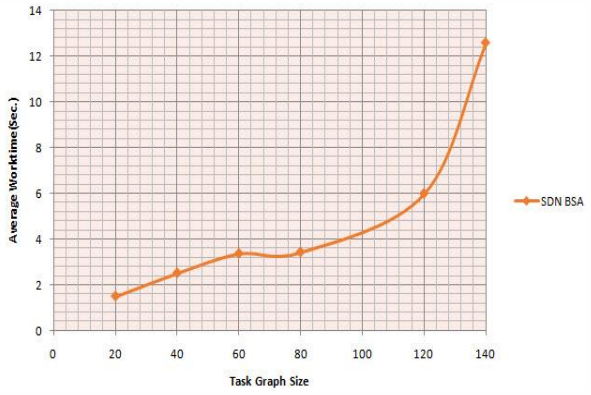

Figure 9. Average of working time in simulation 2

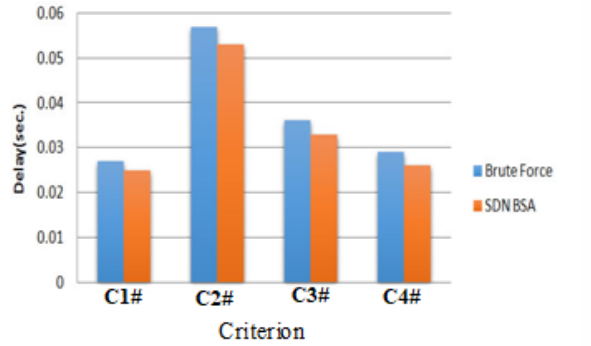

Figure 10. The resual of Simulation 3

V. 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 8GB of RAM and a Core i7 CPU. For each node in the fog network, computation node frequencies of [0.2 GHz, 1.5 GHz] and bandwidths of [250kbps, 54Mbps] 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 GHz, 0.5 GHz] and bandwidths of [150 kbps, 10 Mbps] are considered, as per reference [29]. 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 8, 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 IV. 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.

Simulation 2: To evaluate the average working time of the proposed SDN-BSA algorithm for large 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 IV. As shown in Figure 9, the average working time increases with an increasing task graph size.

Simulation 3: To confirm the SDN-BSA, the overall delay obtained by this algorithm is compared to the

[DOI: 10.61186/itrc.15.4.41]

delay of the exhaustive. Fig. 10 shows the results of this simulation using the benchmarks with the parameters listed in Table V. As Shown in Fig. 10, the delay gained by algorithm SDN-BSA approves the results of exhaustive.

VI. 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. 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 and J.A.Karmouch,"SDN-based edge cloud resource allocation framework," IEEE Access, vol. 7, pp.10672–10690, 2019.

[2] "Openfog Reference Architecture for fog computing," Openfog Consortium, Tech. Rep., 2017. [Online]. Available: https://www. openfogconsortium.org

[3] N.McKeown,"Software-defined networking, INFOCOM Keynote Talk 17 (2009) 30–32.

[4] K. Sood, S. Yu, and Y. Xiang, "Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review," IEEE Internet of Things J., vol. 3, no. 4, pp. 453–463, 2016.

[5] Ch. Bu and J. Wang," computing tasks assignment optimization among edge computing servers via SDN", Springer Peer-to-Peer Networking and Applications,2021.

[6] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh. (2016) SDfog: A Software-Defined computing Architecture for QoS Aware Service Orchestration over Edge Devices. [Online]. Available: arXiv:1609.01190

[7] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing Wireless fog Networks using Software-Defined Networking," in Proc. IEEE/ACS Int. Conf. Computer Systems and Applications, 2017, pp. 1149–1156

[8] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for IoT," Springer Wireless Personal Communications, vol. 92, no. 1, pp. 181–196, 2017.

[9] S. Misra and N. Saha, "Detour: dynamic task offloading in software defined fog for IoT applications," IEEE J. on Selected Areas in Communications, vol. 37, no. 5, pp. 1159–1166, May 2019.

[10] S. Misra and S. Bera, "Soft-VAN: Mobility-aware task offloading in software-defined vehicular network," IEEE Trans. Veh. Technol., vol. 69, no. 2, pp. 2071–2078, Feb. 2020.

[11] Eppstein, D. Subgraph Isomorphism in Planar Graphs and Related Problems. J. Graph. Algorithms Appl. 1999, 3, 1–27.

[12] S.K.Sood and K.D.Singh, "SNA Based Resource Optimization in Optical Network using fog and cloud computing, "Optical Switching and Networking.2017.

[13] L.Liu,Z.Chang,X.Guo,S.Mao,andT.Ristaniemi,"Multiobjective Optimization for Computation Offloading in fog computing,"IEEE Internet of Things Journal,vol.5,no.1,pp.283-294,2018.

[14] N.Wang, B.Varghese, M.Matthaiou, and D.S.Nikolopoulos, "ENORM: A Framework For Edge Node Resource Management, "IEEE Transactions on Services computing, pp.1-1,2018.

[15] 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.196-209,2019.

[16] 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.3702-3712,2016.

[17] 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.108-119,2017.

[18] Pham-Nguyen, H.N., Tran-Minh, Q., 2019. Dynamic resource provisioning on fog landscapes. Security and Communication Networks 2019.

[19] S.Bitam, S.Zeadally, and A.Mellouk, "fog computing job scheduling optimization based on bees swarm," Enterprise Information Systems, vol.12, no.4, pp.373-397,2018/04/21 2018

[20] C. Bu, Jinsong Wang. "computing tasks assignment optimization among edge computing servers via SDN," Springer Peer-to-Peer Networking and Applications, vol. 25, no. 3, pp. 1746–1760, 2017.

[21] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep Reinforcement Learning- Based Task Offloading and Resource Allocation for Mobile Edge computing," in EAI Int. Conf. on Machine Learning and Intelligent Communications, 2018, pp. 33–42.

[22] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software Defined Cooperative Offloading for Mobile cloudlets," IEEE/ACM Transactions on Networking, vol. 14. pp. 1190–1206, 2021.

[23] Mirza, U. M., Arslan, M. A., Cedersjo, G., Sulaman, S. M., and Janneck, J. W. (2014). Mapping and scheduling of dataflow graphs—a systematic map. In Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers page 1843–1847. IEEE.

[24] I. Sugiarto, P. Campos, N. Dahir, G. Tempesti and S. Furber, "Task graph mapping of general purpose applications on a neuromorphic platform", Future Technologies Conference 2017 (FTC 2017 accepted), November 2017.

[25] E. Paone, F. Robino, G. Palermo, V. Zaccaria, I. Sander, and C. Silvano, "Customization of OpenCL applications for efficient task mapping under heterogeneous platform constraints," in Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015, pp. 736–741.

[26] B. Simon, J. Falk, N. Megow, and J.Teich," Energy Minimization in DAG Scheduling on MPSoCs at Run-Time: Theory and Practice" arXiv.1912.09170v1, Dec.2019.

[27] K. Taura, A. Chien: A Heuristic Algorithm for Mapping Communicating Tasks on Heterogeneous Resources. 9th Heterogeneous computing Workshop, Cancun, Mexico (May 2000). Of the ACM HotSDN, 2012, pp. 115–120.

[28] AHMAD, I. AND KWOK, Y.-K. 1999. On parallelizing the multiprocessor scheduling problem. IEEE Trans. Parallel Distrib. Syst. 10, 4 (Apr.), 414-432.

[29] Al-Khawaja M, Baker T, Al-Libawy H, Maamar Z, Aloqaily M, Jararweh Y. Improving fog computing performance via fog-2-fog collaboration. Future Generation Comput Syst. 2019;100:266-280. https://doi.org/10.1016/j.future.2019.05.015.

[30] Rahimi, Payam, Chrysostomos Chrysostomou, Haris Pervaiz, Vasos Vassiliou, and Qiang Ni. "dynamic resource allocation for SDN-based virtual Fog-RAN 5G-and-beyond networks." In 2021 IEEE Global Communications Conference (GLOBECOM), pp. 01-06. IEEE, 2021.

[31] Alomari, Amirah, Shamala K. Subramaniam, Normalia Samian, Rohaya Latip, and Zuriati Zukarnain. "Resource management in SDN-based cloud and SDN-based fog computing: taxonomy study." Symmetry 13, no. 5 (2021): 734.

Sepideh Sheikhi Nejad received her B.Sc. degree in Computer Engineering from Islamic Azad University, Ashtian, Iran, in 2006 and her M.Sc. degree in Computer Software Engineering from Islamic Azad University, South Branch, Tehran, Iran, in 2010. She is currently a Ph.D. candidate at Islamic Azad University, South Branch. Fog Computing and SDN, are her major fields of research.

[DOI: 10.61186/itrc.15.4.41]

Ahmad Khadem-Zadeh was born in Mashhad, Iran, in 1943. He 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 & 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 IEEE Iran Section Standards Committee Chair and also a lecturer at Tehran Universities. He is a committee member of the Iranian Electrical Engineering Conference Permanent Committee. Dr. Khadem-Zadeh has been received four distinguished national and international awards including Khwarizmi International Award and has been selected as the National Outstanding Researcher of the Iran Ministry of Information and Communication Technology.

Amir Masoud Rahmani received his B.Sc. in Computer Engineering from Amirkabir University of Technology, Tehran, in 1996, the M.Sc. 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. He is the author/co-author of more than 350 publications in technical journals and conferences. His research interests are in Distributed Systems, Ad Hoc, Wireless Sensor Networks and Evolutionary Computing.

Ali Broumandnia was born in Isfahan, Iran. He 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 the reviewer of some international journals and conferences.

[DOI: 10.61186/itrc.15.4.41]