LOAD BALANCING TECHNIQUES FOR FOG COMPUTING INTEGRATED TO D2D NETWORKS

Mustafa Khaleel HAMADANI[1], Eugen BORCOCI [2]

Device-to-Device (D2D) communication and Fog Computing (FC) are two emerging paradigms proposed to overcome the increased data traffic. D2D allows a direct connection between users in proximity without the involvement of the Base Station. FC brings the cloud resources close to the users to reduce the communication latency and increase computing efficiency. This work addresses the load balancing issue in FC, and two algorithms are proposed; First Come First Served and Max-Min. The simulation results show that the Max-Min performs better than the FCFS in terms of the maximum completion time of all computing nodes, Execution Time, and Throughput.

Keywords: D2D, Fog, Cloud, FCFS, Max-Min, Makespan.

1. Introduction

The high development of mobile communication technology is a reality. Today, billions of different devices (e.g., mobile devices, autonomous vehicles, sensors, IoT, and others) with various applications such as video streaming, vehicular applications, social networks, mobile terminal applications, and many others are available on top of the internet. Subsequently, a tremendous amount of data are generated and has to be processed while needing vast computing capacity, while some of them having low latency constraints. However, mobile devices have limited computation capacity and battery life. To address the computation issue, there are several options, such as cloud computing, edge, and fog computing.

Cloud computing provides virtually unlimited computation resources (i.e., processing power) and storage capabilities offered too many remote devices (e.g., mobile devices). However, the location of the centralized cloud servers located far away from the end-users results in a high transmission latency, which can affect the performance of the services provided to end-users [1].

Fog computing (FC) has emerged as a promising paradigm to overcome the computation and latency issues mentioned above [2]. Fog computing extended the cloud computing resources (e.g., computation and storage) close to the edge of the network [3]. Fog computing is deployed as a set of servers with limited

[1] PhD student, University POLITEHNICA of Bucharest, Romania, e-mail: mkhaleel190@gmail.com
[2] Prof., University POLITEHNICA of Bucharest, Romania, e-mail: eugen.borcoci@elcom.pub.ro

capabilities (e.g., CPU); these servers are installed at appropriate locations and provide computing resource with lower delay in comparison to remote cloud servers [4].

Device-to-Device (D2D) communication is defined as the direct connection between devices in proximity of each other, without the involvement of wireless infrastructure nodes (e.g., Base Station or Access Point). D2D communication provides a lower communication latency between devices located in the proximity of each other; therefore, D2D is considered a key enabler for future wireless communication (e.g., in 5G technology) [5]. Examples of main use-cases of D2D, such as sharing video content, online gaming, caching and others, are described in [6]. Also, D2D is a powerful solution to serve vehicle-to-vehicle (V2V) communications.

In D2D communication, the participants could be grouped into D2D clusters (e.g., sharing similar content such as online games). Each D2D cluster has a cluster head (CH) that communicates directly with nodes belonging to the wireless infrastructure. The cluster head election procedure is out of the scope of this work.

Traditionally, each D2D cluster generates service tasks (e.g., asking for processing) that will be offloaded to the remote cloud servers. However, the transmission latency is high, due to the remote location of the cloud servers; this will affect the Quality of Experience (QoE) observed by the end-users. Hence, offloading the service tasks to closer fog serves for processing, will reduce the average computation time and improve the overall network performance.

The main contributions of this work are:

- The D2D Service task has been modeled via the expected time to compute (ETC) model and definition of the main objectives.
- Two Tasks offloading techniques are proposed to achieve load balancing in the fog computing nodes.
- Conduct a simulation experiment for task offloading techniques under several performance metrics.

The paper structure is the following: The studying and analysis of previous works are given in Section 2. Section 3 describes the proposed system model. The problem formulation is provided in Section 4. The tasks scheduling techniques description is contained in Section 5. The simulation experiment results are provided in Section 6. The conclusion and a few directions for the future work are given in Section 7.

2. Related Works

In recent years, researchers carried out research on the load balancing problem in cloud computing technologies. A few research works covered an

SDN-based load balancing issue in fog computing. Authors in [7] proposed a low-latency hybrid cloud-fog network architecture for medical big data, the main objective to optimize the processing delay for in business environment and hence, reducing the computing load. The fog computing is deployed at the hospital at network equipment such as routers and switches. The authors applied the Bat algorithm to solve the optimization problem in medical big data scenarios. The limitation of their work-oriented to medical applications and no centralized approach (i.e., SDN) considered in the work. A Hill Climbing Load Balancing (HCLB) technique proposed in [8], the proposed technique optimize Response Time (RT) of the requests and minimizes the processing time as a metric for load balancing techniques in the network. Their work is oriented to Micro Grids (MG) network. the centralised approach (i.e., SDN) does not consider in their proposed technique. The work in [9] proposed a simple Tabu Search method for optimal load balancing between cloud and fog nodes under resource constraints. Their simulation results improved memory usage and minimized computational costs. Another work related to the optimal service placement issue with considering the application load distribution in edge computing environment proposed in [10]. The proposed technique trade-off between the minimization service request deadline and the service migrations. Similar to the above works, the centralised approach does not adapt in their work. Finally, a comparison between load balancing algorithms proposed in literature such as Round Robin (RR), Throttled, Particle Swarm Optimization (PSO) and Active VM Load Balancing Algorithm (AVMLB) provided in [11]. The simulation results show that The AVMLB outperforms all mentioned.

After studying and analysing the previous research works and additionally based works in [12] and [13], we proposed the SDN-based load balancing for Fog nodes oriented to the Device-to-Device network. the proposed techniques based on dynamic centralised load balancing approach, when the SDN controllers responsible for monitoring and controlling the fog nodes and due to global network knowledge; two load balancing algorithms running on the top of SDN architecture to distribute the workload (e.g., video, online games) among the fog nodes. The objective of the intorduced algorithms is to ensure that all fog nodes approximately handle an equal amount of workload (i.e., all fog nodes equally balanced). several of performance metrics including a maximum completion time of all computing nodes, Execution Time, and Throughput.

3. System Model

In this paper it is assumed that the control logic for the edge part of the overall system is based on Software Defined Networking (SDN) concepts. The system architecture comprises three types of functional entities (we can call them

layers): *Infrastructure layer* (it consists of D2D clusters and Base Stations), *Fog Nodes layer*, *Control layer*, as illustrated in Fig.1. The "SDN controllers" is actually an architectural control plane. The other entities/layers (Infrastructure, Fog Nodes, and CRS) contain both data and control functions.

The infrastructure contains the D2D clusters and Base Stations (BS). The BSs manage the D2D clusters and provide cellular connections to D2D users. The BSs are connected to the Fog nodes networks via high-speed connections (e.g., channels on optical fiber). The D2D clusters will be formed by considering a special criterion, e.g., based on similar content (e.g., online gaming). Each cluster has a Cluster Head (CH); however, the Cluster head election procedure is out of the scope of this work. The CH is responsible for offloading computation-intensive tasks to the fog nodes network through a BS. After processing, it distributes the computation results received from Fog nodes, among the cluster members.

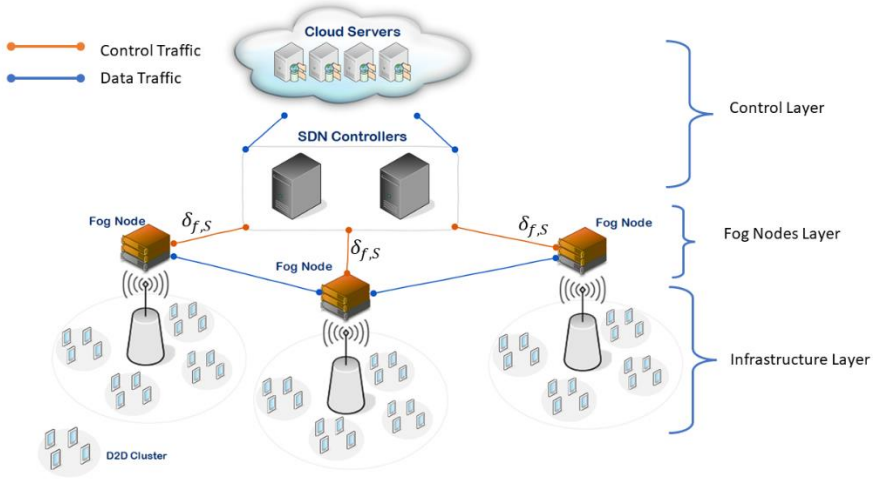

Fig. 1 The System Architecture

The Fog nodes (FN) layer consists of several nodes that provide fog services (e.g., computation and storage) to the D2D clusters. Futher, a single D2D cluster can be connected to one Fog node via the the Base Station. The FNs are connected to elements of the above layers (i.e., SDN controllers and Cloud Servers), as shown in Fig.1.

The control plane in this architecture can be based on Software Defined Networking (SDN) technology. Via OpenFlow control protocol, the fog nodes receive the behavior rules from SDN controllers. Additionally, information such as the current load, the number of tasks in processing, and waiting status will be transferred to the SDN controllers and Cloud Servers in order that the latter can

achieve global information knowledge. The $\delta_{f,s}$ represents the communication latency between the fog nodes and the SDN controllers.

The SDN Control plane consists of several SDN controllers that interact with the FNs through OpenFlow protocol for setting the rules to control the fog nodes network. The SDN controller builds a global knowledge view of the network based on the status information received from the Fog nodes. Hence, an optimal load balancing strategy could be formulated by the SDN controller based on the global network view.

Finally, the remote cloud servers (RCS) layer comprises several data centers that provide almost unlimited computation and storage capabilities to various applications.

The system architecture shown in Fig.1 could be represented as a weighted undirected graph $G = (V, E)$, where V is the set of nodes and $\boldsymbol{V} = \{F, S, C\}$. The Fog nodes set is represented as $F = \{ f_1, \ldots, f_m \}$; S represent the SDN controllers where $S = \{s_1, s_2, \ldots, s_j\}$, and C is cloud remote servers set where $C = \{c_1, c_2, \ldots, c_c\}$. $E = \{e_{a,b}\}$ represents the set of edges between two nodes and $\boldsymbol{e_{a,b}}$ denotes the link between two nodes a and b.

Each fog node consists of n Virtual Machines (VM); The set of VMs is denoted by $N = \{VM_1, VM_2, \ldots, VM_w\}$. Each particular virtual machine VM_v ($VM_v \in N$) has limited capability of computing resources such as CPU, memory, and storage capacity [14]. Each VM_v belonging to the set is described by the following attributes [15]:

$$VM_v = \{VM_v^{Id}, VM_v^{MIPS}\} \tag{1}$$

where VM_v^{Id} denotes a unique identifier of the VM_v and VM_v^{MIPS} is the processing capability of VM_v. In the rest of the text the processing capability of VM_v will be also shortly denoted as P_v.

The D2D clusters $\{D_1, D_2, \ldots, D_i\}$ generate computational service tasks denoted as $\{\lambda_1, \lambda_2, \ldots, \lambda_i\}$, and the service tasks are submitted to the fog nodes network, and no service migration is considered in this work. Each services task λ_k ($\lambda_k \in$ Service Tasks) can be described following a model taken from [15]:

$$\lambda_k = \{\lambda_k^{Id}, \lambda_k^{Length}\} \tag{2}$$

Where,

λ_k^{Id} is a service task unique identifier of the service tasks λ_k.

λ_k^{Length} is the size of the service task λ_k, stated as a number of requests per second.

The ETC concept introduced in [9] is used to describe the service task model for D2D clusters. The ETC can be represented as a matrix (of execution times) given that specific task can be assigned to a particular VM as illustrated in formula (3). So, the execution times is the expected time to complete a task λ_k on VM_v as illustrated in (4).

$$ETC = \begin{bmatrix} ETC_{1,1} & ETC_{1,2} & \cdots & ETC_{1,w} \\ ETC_{2,1} & ETC_{2,2} & \cdots & ETC_{2,w} \\ ETC_{3,1} & ETC_{3,2} & \cdots & ETC_{3,w} \\ \vdots & \vdots & \cdots & \vdots \\ ETC_{k,1} & ETC_{k,2} & \cdots & ETC_{i,w} \end{bmatrix} \quad (3)$$

where $ETC_{k,v}$ defined the expected time to compute a task λ_k on VM_v. The $ETC_{k,v}$ could be calculated as follow:

$$ETC_{k,v} = \frac{\lambda_k^{Length}}{P_v} \quad (4)$$

In ETC matrix, the row *i* indicates a specific service task (λ_k) and the column k represent a specific virtual machine VM_v. For example, $ETC_{1,2}$ expressed the required time to complete a task λ_1 on VM_2, $ETC_{2,4}$ expressed the required time to complete a task λ_2 on VM_4 , and so on.

The purpose of this work is to find an optimal mapping of service tasks to VMs such that the workload is approximately equal among all the nodes; thus, load balancing is achieved.

4. Optimization Problem Formulation

The load balancing problem can be described as finding an optimal assigning of service task k on virtual machine v to satisfy one or more objectives; the most objective studied in the literature is the *makespan* (MS), which is defined as maximum execution time in any node [16]. The execution time (ET) of all VM can be determine as in equation (5).

$$ET_{k,v} = \sum_{k=1}^{i} \sum_{v=1}^{w} x_{k,v} \, . ETC_{k,v} \quad (5)$$

Where ($k \in$ Service Tasks, and $1 < k < i$) and ($v \in w$, and and $1 < v < w$), and $x_{k,v}$ equal 1 when a service task is allocated to a specific VM and otherwise $x_{k,v} = 0$.

Hence, the makespan (MS) can be computed as:

$$MS = maximum\ (ET_{kv}) \tag{6}$$

Finally, the load balancing problem for fog computing can be formulated as:

$$minimize\ (MS) \tag{7}$$

Finally, the load balancing problem for fog computing can be described as is finding an optimal mapping between the service tasks and VMs that minimizes the maximum execution time (i.e., makespan), as shown in 5. Thus, in the next section, two tasks scheduling techniques are proposed to solve the (7).

5. Load Balancing Techniques

This section gives an overview of the common scheduling techniques to solve 7, namely, First Come First Serve (FCFS) and Max-Min algorithms. The working of these algorithms gives as follow:

- First Come First Served (FCFS) is the simple and basic tasks scheduling technique in which the tasks that arrived first are executed first, and the second task has to wait until the execution of the first tasks is completed [17]. This algorithm has less complexity than other scheduling techniques. However, FCFS suffer from high waiting time and does not consider other characteristics such as tasks priority.
- In the Max-Min technique, the expected completion time of service tasks is calculated, and the service task that has maximum expected completion time is chosen and assigned to the corresponding VM. The main advantage of Max-Min is efficient resource utilization. Further, one of the drawbacks of this algorithm is the high waiting time for tasks with small and medium completion times.

Algorithm 1 The Pseudo code of Max-Min algorithm

Input: number of Service tasks, number of VMs.

Output: Minimum completion time of service tasks.

Begin

Step 1: Add the Serive tasks to a list.
Step 2: Comput the completion time for each service tasks.
Step 3: Assgin a service task with maximum completion time to VMs with minumm competion time.
Step 4: Update the Service tasks list.
Step 5: Repest Step 1 to 4 until all service tasks in the list have been assigned.

End of the algorithm.

6. Simulation Results

This section performed simulation experiments to study the proposed algorithms mentioned above; several metrics included the makespan (MS), resource utilization, and execution time. We considered CloudSim as a simulation tool [18] for this work. The simulation scenario assumed that the number of D2D clusters varied from 100 to 1000 (e.g., 100, 200, 300, etc.), and these clusters generate a service task with a length of about 2500 requests per second. Moreover, the number of fog nodes is about 10 nodes and each node and the number of virtual machines varied from 10, 15, and 20. Table 1 shown the simulation parameters used in this work.

Table 1 Simulation Parameters

Simulation Parameters	Description
Number of D2D Clusters	100, 200, 300, and 1000
Number of Fog Nodes	10 Nodes
Virtual Machines	10, 15, 20
Service Tasks Size	2500 request per second
Virtual Machine Process Power	1000 *MIPS*

The performance metrics that considered given as follow:

- The Makespan represents the completion of all the service tasks execution in the system. The smaller value of the makespan is desirable.
- The Execution Time is the time that taken by the service task to complete. The lowest value, the better performance of the algorithm.
- Throughput is defined as the number of service tasks executed during a unit of time (e.g., second). The higher throughput is also desirable, and the throughput can be determined as follow:

$$Throughput = \frac{i}{makespan} \tag{8}$$

Where i represented the number of services tasks.

Fig.2 shown the behavior of FCFS and Max-Min algorithms; the Max-Min provided a lower value of makespan compared to FCFS. For example, when the number of D2D clusters equals 500 clusters, the value of makespan equal to 0.02 seconds (Max-Min), and FCFS about 0.03 seconds (FCFS).

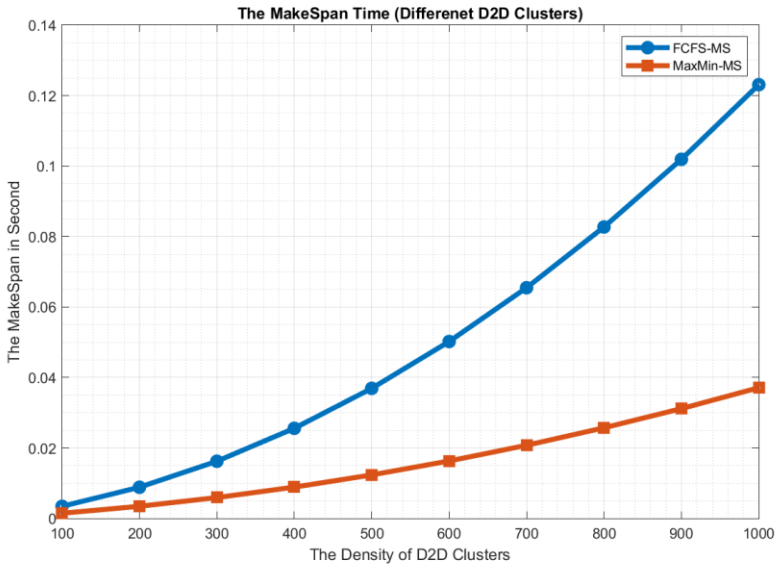

Fig. 2 The Makespan for Different D2D Clusters

The Max-Min provides a lower service task execution time, where FCFS has higher execution, as illustrated in Fig.3. Thus, poor performance of FCFS technique. For example, where the number of D2D Clusters equals 500; the FCFS needs about 0.07 seconds to execute a service task where the Max-Min takes about 0.05 seconds for service task execution.

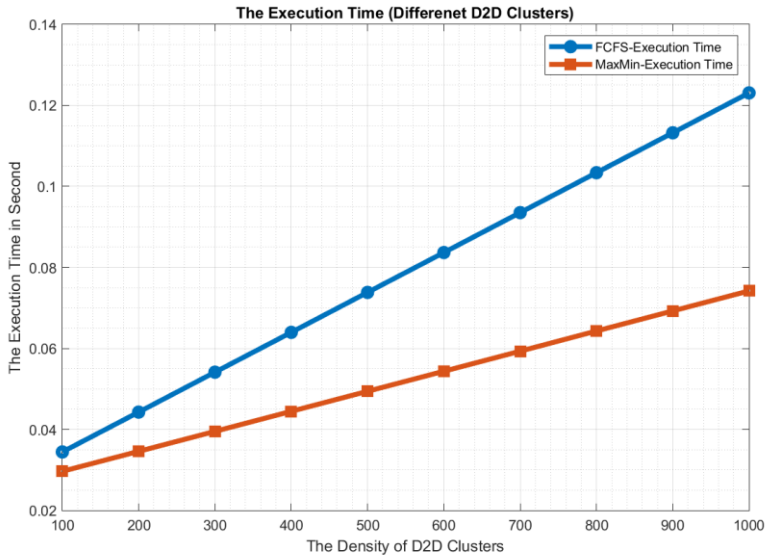

Fig. 3 The Execution Time for Different D2D Clusters

As illustrated in Fig.4, the Max-Min perform better than the FCFS algorithm in term of throughput.

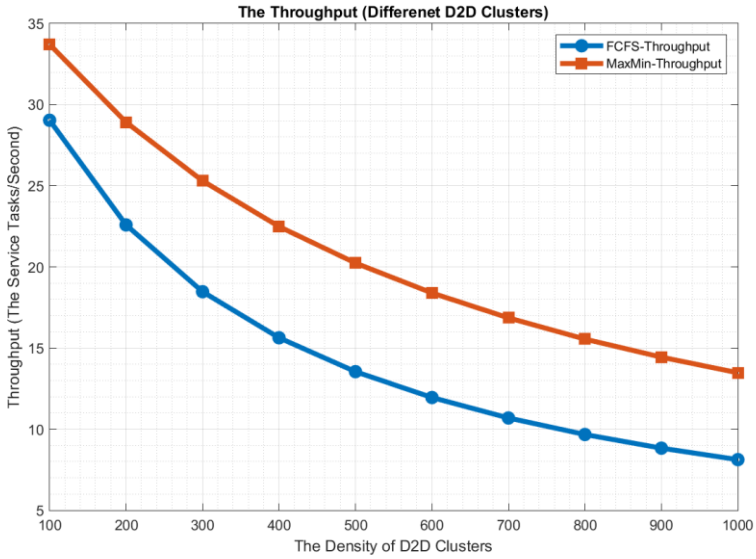

Fig. 4 The Throughput for Different D2D Clusters

For example, when the number of D2D cluster equal 500, the value of throughput equal to 20 services per second, and in case of FCFS; the throughput equal 14 tasks per second

7. Conclusion and Future Work

In this work, we have studied various load balancing techniques for fog computing. From the simulation results, we observed that overall the Max-Min algorithm performs better than the FCFS algorithm. As we notice from Fig.2, the Max-Min provides a lower makespan value; when the number of D2D clusters is equal to 500, the value of makespan with Max-Min algorithm is equal to 0.02 seconds where the FCFS gives the makespan value above 0.03 second.

The FCFS algorithm has poor performance in terms of the execution time, as has been noticed in Fig.3, where the number of D2D Clusters is equal to 500; the FCFS executes a service task within 0.07 seconds and in the case of Max-Min, equal to 0.05 seconds.

The Max-Min algorithm provides higher throughput than the FCFS, as seen in Fig.4, at 500 D2D clusters; the throughput value is equal to 20 service tasks in the case of the Max-Min algorithm, and with FCFS, the throughput equal about 14 service tasks per second. In general, the Max-Min performs better than the FCFS under mentioned performance metrics. In future work, a meta-heuristic approach will be considered for load balancing; other performance metrics such as resource utilization and the degree of Imbalance will be considered in the algorithm design.

R E F E R E N C E S

[1] A. N. Toosi, R. N. Calheiros, and R. Buyya, “Interconnected cloud computing environments: Challenges, taxonomy, and survey,” ACM Comput. Surv. CSUR, vol. 47, no. 1, pp. 1–47, 2014.

[2] D. Wang, Z. Liu, X. Wang, and Y. Lan, “Mobility-aware task offloading and migration schemes in fog computing networks,” IEEE Access, vol. 7, pp. 43356–43368, 2019.

[3] S. Antonio, “Cisco delivers vision of fog computing to accelerate value from billions of connected devices,” Cisco San Franc. CA USA, 2014.

[4] L. Liu, Z. Chang, and X. Guo, “Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices,” IEEE Internet Things J., vol. 5, no. 3, pp. 1869–1879, 2018.

[5] U. N. Kar and D. K. Sanyal, “An overview of device-to-device communication in cellular networks,” ICT Express, vol. 4, no. 4, pp. 203–208, 2018.

[6] S. Jayakumar and S. Nandakumar, “A review on resource allocation techniques in D2D communication for 5G and B5G technology,” Peer--Peer Netw. Appl., vol. 14, no. 1, pp. 243–269, 2021.

[7] J. Yang, “Low-latency cloud-fog network architecture and its load balancing strategy for medical big data,” J. Ambient Intell. Humaniz. Comput., pp. 1–10, 2020.

[8] M. Zahid, N. Javaid, K. Ansar, K. Hassan, M. K. Khan, and M. Waqas, “Hill climbing load balancing algorithm on fog computing,” in International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2018, pp. 238–251.
[9] N. Téllez, M. Jimeno, A. Salazar, and E. Nino-Ruiz, “A tabu search method for load balancing in fog computing,” Int J Artif Intell, vol. 16, no. 2, 2018.
[10] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, “Dynamic Service Placement and Load Distribution in Edge Computing,” in 2020 16th International Conference on Network and Service Management (CNSM), 2020, pp. 1–9.
[11] S. H. Abbasi, N. Javaid, M. H. Ashraf, M. Mehmood, M. Naeem, and M. Rehman, “Load stabilizing in fog computing environment using load balancing algorithm,” in International Conference on Broadband and Wireless Computing, Communication and Applications, 2018, pp. 737–750.
[12] C. Shi, Z. Ren, and X. He, “Research on load balancing for software defined cloud-fog network in real-time mobile face recognition,” in International Conference on Communicatins and Networking in China, 2016, pp. 121–131.
[13] I. Strumberger, M. Tuba, N. Bacanin, and E. Tuba, “Cloudlet scheduling by hybridized monarch butterfly optimization algorithm,” J. Sens. Actuator Netw., vol. 8, no. 3, p. 44, 2019.
[14] I. Attiya and X. Zhang, “D-choices scheduling: a randomized load balancing algorithm for scheduling in the cloud,” J. Comput. Theor. Nanosci., vol. 14, no. 9, pp. 4183–4190, 2017.
[15] I. Strumberger, M. Tuba, N. Bacanin, and E. Tuba, “Cloudlet scheduling by hybridized monarch butterfly optimization algorithm,” J. Sens. Actuator Netw., vol. 8, no. 3, p. 44, 2019.
[16] B. Sahoo, “Dynamic load balancing strategies in heterogeneous distributed system,” PhD Thesis, 2013.
[17] I. M. Ibrahim, “Task scheduling algorithms in cloud computing: A review,” Turk. J. Comput. Math. Educ. TURCOMAT, vol. 12, no. 4, pp. 1041–1053, 2021.
[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Softw. Pract. Exp., vol. 41, no. 1, pp. 23–50, 2011.