IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 16, 2020, accepted January 13, 2021, date of publication January 25, 2021, date of current version February 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054420

Multi-Level Resource Sharing Framework Using Collaborative Fog Environment for Smart Cities

TARIQ QAYYUM[1], ZOUHEIR TRABELSI[2], ASAD WAQAR MALIK[1], (Senior Member, IEEE), AND KADHIM HAYAWI[3], (Member, IEEE)

[1]Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[2]College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
[3]College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates

Corresponding author: Zouheir Trabelsi (trabelsi@uaeu.ac.ae)

This work was supported by the United Arab Emirates (UAE) University UAEU Program for Advanced Research (UPAR) Research Grant Program under Grant 31T122.

ABSTRACT Fog computing has proved its importance over legacy cloud architectures for computation, storage, and communication where edge devices are used to facilitate the delay-sensitive applications. The inception of fog nodes has brought computing intelligence close to the end-devices. Many fog computing frameworks have been proposed where edge devices are used for computation. In this paper, we proposed a simulation framework for fog devices that can use end devices to handle the peak computation load to provide better Quality of Services (QoS). The regional fog nodes are deployed at network edge locations which are used as an intelligent agent to handle the computation requests by either scheduling them on local servers, cloud data centers, or at the under-utilized end-user devices. The proposed device-to-device resource sharing model relies on Ant Colony Optimization (ACO) and Earliest Deadline First(EDF) Algorithm to provide a better quality of service using device available at multi-layer design. The concept of using IoT devices as fog nodes has improved the performance of legacy fog based systems. The proposed work is benchmarked in terms of system cost, efficiency, energy, and quality of service. Further, the proposed framework is with xFogSim in terms of task efficiency.

INDEX TERMS Fog computing, IoT, resource management, fog simulators, OMNeT++.

I. INTRODUCTION

With greater needs for sensor devices and the internet of things, a huge amount of data is produced which requires extensive computing resources. The concept of 5G has further increased the data generation rate with high-speed data rate between the device-to-device communication is used. Therefore, compute extensive tasks can be offloaded to cloud data centers with an additional cost of multi-hop network latency, end-to-end delay, bandwidth, and congestion. However, to facilitate the delay-sensitive tasks such as medical, power, gas and oil leaks, the concept of fog computing is adopted that helps in performing task computations on near fog devices. Thus, reducing the network load while sending the data to cloud data centers. This inclusion helps in reducing task latency, end-to-end delay, and bandwidth. In a multi-agent fog environment, the concept of fog broker is used which manages the local fog nodes, defines task execution policies, and communicates with other devices [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Zehua Guo.

A generic layered architecture of the fog computing environment is given in Fig. 1. There are three layers in conventional fog networks. The lower level comprises IoT devices that request resources from gateway broker nodes. The second step comprises gateway nodes that connect the end-user devices with fog nodes and also perform resource management. In case, the gateway broker does not have vacant resources it requests nearby brokers for resources. The broker returns the task results to IoT devices. The application that can be facilitated through fog nodes is security systems, healthcare systems, or facial detection, etc. The gateway broker can schedule the task to local resources; however, doing that constantly can result in long queues which can degrade the system performance [2].

In recent years, most of the work has been focused on scheduling algorithms, task distribution, and the energy preservation of handheld devices. The adoption of 5G has changed this concept of using central nodes for compute-intensive tasks execution. In 5G, device-to-device communication is used to achieve high throughput, low power consumption, and wide-area coverage [3]. In the

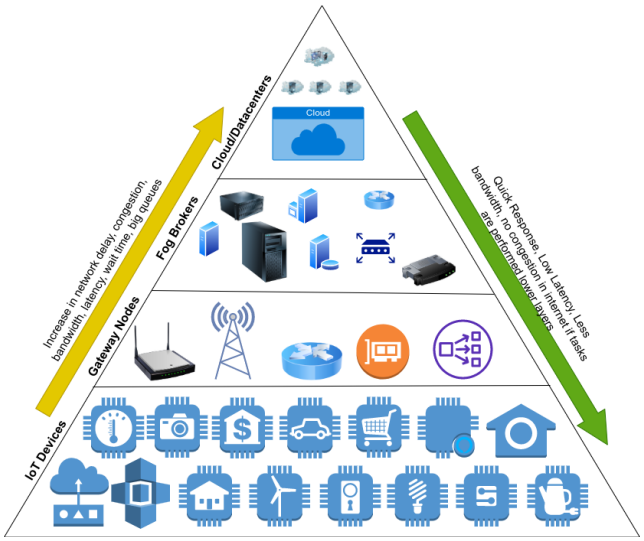


FIGURE 1. Fog computing architecture pyramid diagram.

proposed work, the concept of device-to-device communication is used which enhances overall system performance and reduces waiting time as well as network congestion [4].

In this article, we proposed a simulation framework for fog networks that inherit the concept of the device to device communication to offload the task to the nearby end device. These end devices are named as volunteer nodes because they volunteer their resources and communicate directly with task requesting nodes to achieve the device-to-device communication feature of 5G. There are gateway nodes act as a broker node to resolve the incoming computing requests. The dedicated fog devices and volunteer devices are used to perform computation. The framework is dynamic, devices having sufficient energy, and available resources can participate in the resource sharing model where the broker nodes allocate requests to the most capable device, keeping in view the energy and other constraints. When a task is performed, the volunteer computing device sends results directly to the requesting device and acknowledges to the broker device. The other salient features of the framework are:

- Efficient energy management module which enables the system to preserve energy through task scheduling over devices having significant energy.
- Safe handover mechanism under mobile environment, especially, When both end user device and volunteer fog nodes are mobile, the localization becomes very complex.
- Configurable mobility module that allow researchers to include their own mobility models through the extensive markup language.
- Inclusion of collaborative geographically distributed fog locations to ensure the availability of resources.
- Addition of backend datacenters for batch processing tasks having less stringent deadlines, and computation intensive.
- Quality of service (QoS) and resource localization based allocation algorithm to ensure that the task gets its required QoS with least possible communication delay to improve overall system performance.

The rest of the paper is structured as follows: Section II covers the related work. The system model is presented in Section III. Section IV covers the proposed framework in detail and system evaluation is covered in Section V. Finally, the conclusion and future directions are presented in Section VII.

II. LITERATURE REVIEW

This section covers the existing work on fog computing frameworks and simulators.

A. FOG COMPUTING FRAMEWORKS

YAFS *et al.* [5] proposed a simulation framework to enable users to simulate fog networks in the context of node placement. The authors allow users to place fog nodes based workstations at different network topologies to perform result comparison.

Belli *et al.* [24] proposed a fog based mobile crowdsensing platform for the efficient users' recruitment process. They utilized the concept of using mobile nodes to perform specific tasks in a complex fog network. The mobile nodes advertise their battery and CPU power and the fog node takes the offloading decision. However, the fog nodes are not utilized for computation purpose; thus, only act as a task distributor.

Tom *et al.* [25] proposed a routing protocol labeled as Low Power Lossy Area Network for smart metering. The authors incorporated a middle node i.e. aggregator between meters and fog nodes. Using the intermediate node, the authors claim to improve the packet delivery ratio by 35% and decreased end-to-end delay by 13%. The simulation is performed using Contiki OS [26].

Natesha *et al.* [27] proposed an IoT application module placement strategy for fog network environment. The authors benchmark the proposed work using iFogSim and claim a significant decrease in delay and power consumption.

Huang *et al.* [28] proposed a mathematical model to generate automated test cases based on path coverage for a dynamic fog environment.

Avaisi *et al.* [29] proposed an efficient car parking system that uses fog computing concepts to reduce network latency and delay. The solution is tested using iFogSim and compared with a similar solution in a cloud-based environment. Further, significant improvement has been observed in terms of latency and performance.

Rehman *et al.* [30] proposed a trust-based task mapping and resource sharing framework for vehicular fog networks. They applied the concept of 5G and utilized the concept of trust matrix to form an off-street cluster of vehicles. A task mapping algorithm is also proposed which enhances the efficiency of resource sharing.

Nashaat *et al.* [31] proposed an IoT application placement algorithm for a fog environment. The authors classified the algorithm into two phases. The first phase is to prioritize IoT application requests based on application usage, user expectation, and environment run-time context. It used QoS

violation as feedback, in the second phase it places the request to fog node based on its computing capability, proximity, and its response time. This model showed improved system performance plus a minute increase in power consumption. Dar *et al.* [32] proposed and designed a delay-aware accident detection and response system. The authors believe that in developing countries, economy scale vehicles are used which are not digitally equipped. Therefore, an Android phone is used where the application detects accidents through its sensors. This information is sent to the nearest hospital.

Deep *et al.* [33] proposed a 5G optical fog node which creates cyberspace near IoT devices. The devices can efficiently upload tasks to the fog node and gets the response within the time-bound.

Rafique *et al.* [34] proposed a resource allocation strategy in fog network. A hybrid modified cat swarm optimization algorithm (MCSO) and a modified particle swarm optimization algorithm (MPSO) is used for task allocation at the fog nodes. They distributed resources based on incoming demands. The main objective is to reduce delay and response time.

Ammad *et al.* [35] proposed a new multi-level energy-efficient framework for fog and IoT based smart networks. Two additional layers, energy-efficient hardware, and policy layers are introduced in the IoT-fog-cloud network to make energy-aware decisions. The energy required to execute a particular task is also estimated. Further, various case studies are discussed to evaluate the performance of the proposed framework which includes the smart airport, smart agriculture, smart hospital, and smart parking.

Coutinho *et al.* [36] proposed a toolkit for fog computing emulation. They designed the fogbed in a virtualized environment which enables users to understand fog concepts and test fog components using third-party available tools.

B. FOG SIMULATORS

RECAP [6], is a fog/cloud simulation framework which can be used to simulate large scale fog, edge, and cloud network scenarios for applications related to control and decision. The simulator provides support in predicting the impact on Quality of service (QoS), workload, and resources.

iFogSim [14] is a simulation toolkit for fog network environments. It adopted the concepts from CloudSim. The iFogSim has three main components, i.e. physical, management, and logical. Buyya *et al.* [37] elaborated the concept of iFogSim and its usage in various applications.

piFogBed [38] is the first fog computing testbed which is based on Raspberrypi and can easily be configured to use with a realistic fog network. However, for the evaluation, the authors simulated a medical monitoring system.

MobFogSim- [8] is an iFogSim based simulator that enables the modeling of the node's mobility and service migration. In this simulator, the impact of mobility on service migration is studied using an experiment-based approach. MockFog [39] is a freely available simulator where users can simulate small scale fog networks, and test the performance by setting up the network characteristics. Héctor [40] is a simulator for automatic testing of IoT devices. The simulator generates virtual testbeds with customizable network characteristics.

PureEdgeSim [16] is used for performance evaluation of cloud, edge, and fog networks and simulation of resource management strategies. A Fuzzy Decision Tree-based resource allocation algorithm is also proposed which enables the system to adopt IoT environment changes. Finally, a case study is simulated to evaluate the simulator's performance and valuable results are collected.

SatEdgeSim [41] is PureEdgeSim based simulator for modeling satellite edge computing-based networks. The impact of various task deployment strategies is studied. A comparison is also given between conventional task deployment strategies and proposed strategies. The results show that the proposed model is efficient in terms of delay, energy consumption, and task success rate. xFogSim [23] is a simulator designed in OMNeT++[1] which can be used to simulate distributed fog networks. Medium-scale networks are simulated and it has a very efficient task distribution algorithm that chooses the best fog node depending on the cost, availability, and response time. The simulator allows researchers to simulate complex network scenarios, and evaluate the network performance.

The proposed system has many additional features as compared to xFogSim. In exFogsim, only the dedicated fog devices are used to compute tasks but in the proposed framework the concept of device-to-device communication is applied which increases the available system resources because here some user nodes will be acting as fog devices to compute tasks. An efficient and collaborative resource sharing algorithm is proposed that chooses the best available option for task computation from volunteer, fog, and cloud. The task acceptance ratio is increased because no task is dropped due to a limited number of fog devices. An efficient energy management module is deployed. Table 1 shows a detailed comparison of the proposed simulation framework with many existing fog simulators.

III. SYSTEM MODEL

The system architecture is represented in Fig. 2. There is G_n number of fog locations and each fog location is equipped with W_m fog gateway nodes. Whereas, the F_n fog devices are dedicated computing nodes. There are U_n heterogeneous devices which include IoT sensors, mobile nodes, user devices, vehicles, roadside sensors. Except for the sensors, all can work in two modes simultaneously i.e. computation seeker and computation provider. These mobile nodes update the gateway nodes about their available energy, and computation power through heartbeat messages. When a user node U_i generates computation request x_i and send it to nearby gateway fog node W_i available in their region. The gateway nodes forward this request to dedicated fog

[1] https://omnetpp.org/

TABLE 1. Fog simulators comparison.

Simulator	GNU GPL	Platform Independent	Fog Federation	Mobility Support	Customize-able Algorithm	Energy	Cloud Integration	Device-Layer as fog
YAFS [5]	✓	✓	✗	✗	✗	✓	✗	✗
RECAP [6]	✓	✓	✗	✗	✗	✓	✓	✗
FogTorch [7]	✓	✓	✗	✗	✗	✗	✗	✗
MobFogSim [8]	✓	✓	✗	✓	✗	✗	✗	✗
FogDirSim [9]	✓	✓	✗	✗	✗	✗	✗	✗
PFogSim [10]	✓	✓	✗	✓	✗	✗	✗	✗
Edge-Fog cloud [11]	✓	✓	✗	Limited	✗	✗	✗	✗
SimpleIoTSimulator [12]	✗	✓	✗	✓	✗	✗	✗	✗
OPNET [13]	✓	✓	✗	✓	✗	✓	✗	✗
iFogSim [14]	✓	✓	✗	✗	✗	✗	✓	✗
EdgeNetworkCloudSim [15]	✓	✓	✗	✗	✗	✗	✗	✗
PureEdgeSim [16]	✓	✓	✗	✗	✗	✗	✗	✗
MyiFogSim [17]	✓	✓	✗	✗	✗	✗	✗	✗
iFogSimWithDataPlacement [18]	✓	✓	✗	✗	✓	✗	✓	✗
EdgeCloudSim [19]	✓	✓	✗	✓	✗	✗	✓	✗
StormOnEdge/SpanEdge [20]	✓	✓	✗	✓	✗	✗	✓	✗
DockerSim [21]	✓	✓	✗	✓	✓	✗	✗	✗
EmuFog [22]	✓	✗	✗	✗	✓	✗	✗	✗
xFogSim [23]	✓	✓	✓	✓	✓	Limited	✓	✗
Proposed Framework	✓	✓	✓	✓	✓	✓	✓	✓

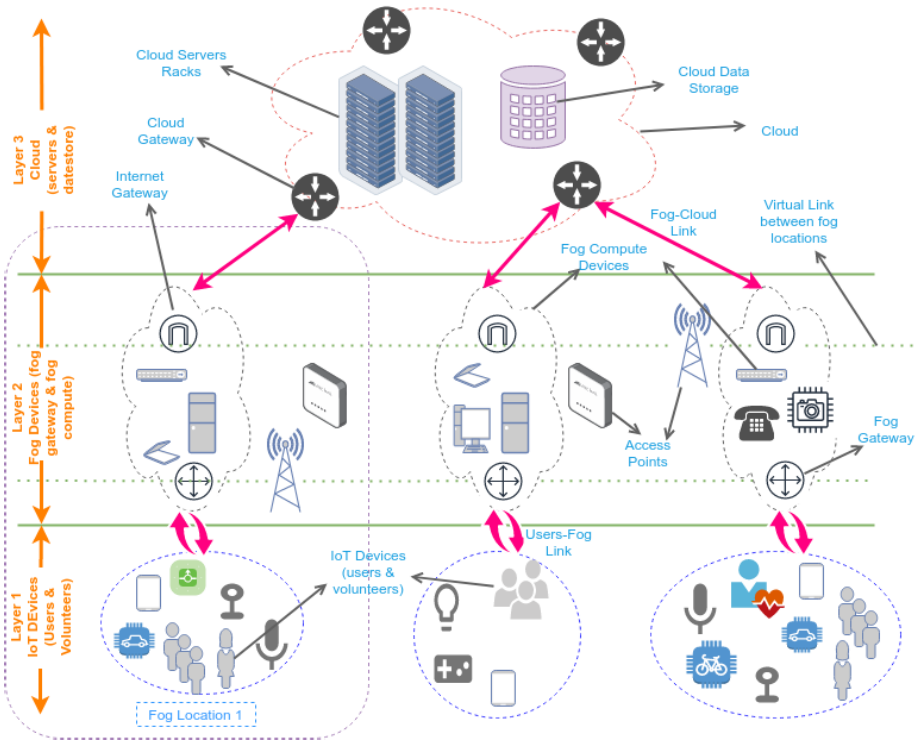


FIGURE 2. Proposed system architecture–devices communicate with fog devices, offload tasks. The fog nodes are connected with cloud data centers. Fog gateways are the intelligent devices decides whether offload tasks to dedicated fog nodes in its location, neighbor locations or nearby volunteer user nodes.

node F_i or forward to a volunteer user node V_i. If a task is completed on the dedicated fog node, it sends the result to the requesting node directly and acknowledges the gateway node about this task expectation to ensure desirability and avoid data loss. The desirability means whether a task received its desired quality of service or not. The same happens if the task is computed on the user node. In both scenarios, systems ensure device-to-device communication to reduce extra delay

TABLE 2. Summary of notations.

Sr.	Symbol	Definition
1	G_n	Distributed Fog Locations
2	F_n	Dedicated Fog Nodes
3	W_n	Gateway Nodes
4	U_n	User nodes seeking computation
5	V_n	User nodes volunteer resources
6	$\mathcal{D}_i^F$	Computation delay (queue + service)
7	ν_i	Service rate at fog node F_i
8	u_i	Average arrival rate at F_i
9	$\mathcal{D}_{x_i}^L$	Network delay to local fog broker
10	$\mathcal{D}_{x_i}^V$	Network delay to volunteer node
11	$\mathcal{D}_{x_i}^{G_j}$	Network to remote location
12	x_i	Computation request
13	γ	size of the request x in Bytes
14	$\mathcal{E}_{Net}^i$	The energy required to transmit a Byte data
15	U	Network element utilization rate
16	Ψ_{idle}^i	Network element's idle power
17	ζ_{idle}^i	Network element's idle network capacity
18	Ψ_{max}^i	Network element's maximum power
19	ζ_{max}^i	Network element's maximum network capacity
20	$\mathcal{E}_\eta$	Energy consumption when task is computed in cloud
21	$\mathcal{E}_R$	Energy consumption in case nearby fog location
22	$\mathcal{E}_{\alpha,x_i}^i$	Energy consumed by access network
23	$\mathcal{T}_\gamma$	Time required to process γ sized request
24	$\mathcal{T}_{act,\gamma}$	Time consumed to process active request of size γ

and to reduce the burden at the centralized fog node. All this communication is being performed staying in the LAN network. In some cases, the system can communicate with cloud servers through a wide area network.

In the proposed work M/M/c model is adopted [42] where the system follows a queue with multiple computing nodes. However, the arrival rate is simulated using the Poisson process. The total arrival rate λ_T at fog device F_i is given in eq. 1.

$$\lambda_T = \sum_{F_i \in F_n} \lambda_{F_i}. \tag{1}$$

where λ_{F_i} is the average arrival rate at *ith* fog node F. The system will accept new task request if the total arrival rate λ_T is less the total workload capacity of the system ν_T. According to Liu *et al.* [43] the decision ι is made by equ 2.

$$\iota = \begin{cases} 1.0 & \lambda_T \leq \nu_T \\ \dfrac{\nu_T}{\lambda_T} & \lambda_T > \nu_T \end{cases} \tag{2}$$

The task execution rate $\mathcal{D}_{x_i}^F$ is determined using [44].

$$\mathcal{D}_{x_i}^F = \iota.\lambda_T = \begin{cases} \lambda_T & \lambda_T \leq \nu_T \\ \nu_T & \lambda_T > \nu_T \end{cases} \tag{3}$$

The computational delay for a request x_i is calculated using eq. 4:

$$\mathcal{D}_{x_i} = \mathcal{D}_{x_i}^F + \mathcal{D}_n. \tag{4}$$

Considering the system is first to come first serve based queue and $\mathcal{D}_{x_i}^F$ means task execution delay which is a combination of both queuing delay and the service delay. The notation $\mathcal{D}_n$ denotes the network delay. There are three possibilities in our proposed work, i). The task is computed on a dedicated fog device in a nearby fog location ii). the task is computed on local dedicated fog devices and iii). the task is computed on a volunteer user node. If the task is executed on a dedicated fog device within the same fog location then the network delay is calculated using eq. 5 where d_{u,F_i} is the network delay between user and fog device.

$$D_n^L = d_{u,F_i}. \tag{5}$$

The eq. 6 is used to calculate the network delay when a task is executed on a remote fog node.

$$\mathcal{D}_n^R = d_{u,F_i} + \sum_{f \in F_r} d_{F_i,f}. \tag{6}$$

Here d_{u,F_i} means the delay between user and local fog device, and $d_{F_i,f}$ is the average delay between local fog device and the remote fog device. In eq. 7, the network delay is given when a task is computed on a local volunteer node [45].

$$\mathcal{D}_n^V = \mathcal{L}_{u_i,w} + \mathcal{L}_{u_j,w} + \mathcal{L}_{u_i,u_j}. \tag{7}$$

This is the best-case scenario where $\mathcal{L}_{u_i,w}$ is the latency between *ith* user and gateway, $\mathcal{L}_{u_j,w}$ is the latency between *jth* user and gateway, and $\mathcal{L}_{u_i,u_j}$ is the latency between *ith* and *jth* user for the *jth* user result transmission.

The distance between a user node and volunteer node is determined by standard distance equation 8 where p(x,y) and v(x,y) are the coordinates of user u and volunteer node v respectively in a two-dimensional plane.

$$d_v \leftarrow \sqrt{(p_y - v_y)^2 + (p_x - v_x)^2} \tag{8}$$

The energy of the system is divided into three categories. i). energy consumed by the access network devices (α) e.g. switches/routers when a request x_i of size γ is transmitted in a fog network $\mathcal{E}_{\alpha,x_i}^F$ ii). energy consumed by the fog device, volunteer user device, or cloud servers for request processing (ψ) i.e. $\mathcal{E}_{\psi,x_i}^F$. iii). sometimes additional data is required that consume network energy for this request i.e. $\beta * \mathcal{E}_{\rho,x_i}^F$ where β is the ratio $N_{sent}/N_{received}$ of the number of the request sent/received to the cloud. According to Deng *et al.* [45], the total energy consumed by fog is given in eq. 9.

$$\mathcal{E}_F = \mathcal{E}_{\alpha,x_i}^F + \mathcal{E}_{\psi,x_i}^F + \mathcal{E}_{\alpha 2,x_i}^F + \beta * \mathcal{E}_{\rho,x_i}^F \tag{9}$$

Here $\mathcal{E}_{\alpha,x_i}^F$ is the energy consumed by access network elements to upload request x_i to fog device, $\mathcal{E}_{\psi,x_i}^F$ is the energy consumption during process and storage operation at fog device, and $\mathcal{E}_{\alpha 2,x_i}^F$ is the energy consumed by network elements when the result is returned to requesting user. When task is computed at volunteer user device, the energy consumption is computed in eq. 10

$$\mathcal{E}_F = \mathcal{E}_{\alpha,x_i}^F + \mathcal{E}_{\psi,x_i}^F + \mathcal{E}_{\alpha 2,x_i}^F \tag{10}$$

The system has a feature of leasing resources from nearby fog locations, so the energy consumption when a request is computed in a remote location is given in eq. 11

$$\mathcal{E}_R = \mathcal{E}_{\alpha,x_i}^R + \mathcal{E}_{\psi,x_i}^R + \mathcal{E}_{\alpha 2,x_i}^R. \tag{11}$$

where $\mathcal{E}^{R}_{\alpha,x_i}$ is the energy consumed by network nodes to transmit data to a remote location, $\mathcal{E}^{R}_{\psi,x_i}$ and $\mathcal{E}^{R}_{\alpha 2,x_i}$ are the energies consumed during processing request and returning result to requesting user respectively. In some rare cases, the request is processed in cloud servers. Hence the energy consumed in that case is given in eq. 12

$$\mathcal{E}_{\eta} = \mathcal{E}^{\eta}_{\alpha,x_i} + \mathcal{E}^{\eta}_{\psi,x_i} + \mathcal{E}^{\eta}_{\alpha 2,x_i}. \tag{12}$$

The access energy consumed by the network elements e.g. switches/routers is calculated in equation 13

$$\mathcal{E}^{i}_{\alpha,x_i} = \gamma * \mathcal{E}^{i}_{Net} + \gamma * (\Psi^{i}_{idle}/\zeta^{i}_{idle} + \Psi^{i}_{max}/U * \zeta^{i}_{max}) \tag{13}$$

The energy consumption for execution of request x_i at *ith* volunteer node is given in eq.14

$$\mathcal{E}^{i}_{\psi,x_i} = \gamma * \mathcal{E}^{i}_{\psi} \tag{14}$$

and the energy consumption during task execution at a fog device i is calculated using eq. 15

$$\mathcal{E}^{i}_{\psi} = \Psi^{i}_{idle} \cdot \frac{\mathcal{T}_{\gamma}}{\mathcal{T}_{Total}} + \Psi^{i}_{max} \cdot \frac{\mathcal{T}_{act,\gamma}}{\mathcal{T}_{Total}} \tag{15}$$

In next section, system components are discussed in details.

IV. PROPOSED FRAMEWORK

The proposed simulation framework consists of mobile and static nodes. Let's assume, there are n fog locations. Each fog node has a broker node that can resolve the incoming computing requests. Moreover, there are few mobile devices which act as a fog node to share their resources. Whereas, the requests are generated from user devices that only send requests which are resolved through fog broker. The user device sends job tasks to the fog broker with the strict QoS constraints. The fog broker schedules the incoming computation jobs to a local fog node or volunteer device. A detailed flow of tasks and their control is shown in Algorithm 3.

A. DESIGN COMPONENTS AND UTILITIES

The core components of the proposed framework include user devices, volunteer user devices, fog gateway, centralized fog nodes, and backend cloud data center.

User devices – are the nodes which include IoT sensors, mobile devices, and other handheld devices. These devices communicate with gateways devices to resolve computation requests.

Volunteer user devices – volunteer devices are the mobile fog nodes which can share their resources on request. The volunteer devices broadcast their resource information along with available energies to nearby fog gateway nodes. This received information is used to select the most suitable device for the request. The volunteer node executes the assigned task and returns the result directly to the requesting user nodes. Thus, for result delivery no intermediate node is considered, this is to reduce the delay and network congestion. The task execution at volunteer node is illustrated in algorithm 2.

Fog gateway – is responsible for receiving resources requests, heartbeats messages from the user, and volunteer devices. The gateway selects the suitable device for the incoming request and forwards the task to the selected device accordingly. The potential computation devices include dedicated fog nodes available at fog gateways, neighboring fog locations, cloud data centers, or volunteer end devices. The resource allocation algorithm is responsible for selecting the computation device.

Dedicated Fog Servers – refer to the nodes available at each fog location having significant computing resources with no energy constraints. These servers can accommodate multiple requests at a time. The fog servers share their resource information along with task residence time with gateway nodes. This information is used for computation device selection.

Cloud data centers – This module can handle compute and data-intensive tasks. Moreover, with specific APIs, sensor data can be stored in the data centers. Further, this module is added to support batch processing with no stringent deadlines; whereas, these tasks may require resources for concurrent execution. In such cases, the framework provides an interface to upload those tasks to the cloud where more compute resources are available. Such tasks may include machine learning and artificial intelligence algorithms where the intermediate results are the input for the next layer. In such situations, the cloud data centers perform the required computations and share the results with fog nodes for better intelligent decisions.

Mobility Models & Handover – The mobility is an important aspect of performance degradation in a mobile environment. The fast-moving vehicles or devices can very rapidly disassociate from the network [46]. Therefore, data loss and re-transmissions decrease overall network performance/efficiency. In the proposed framework, devices can adopt different mobility models. The list of mobility modules that exist is Linear, Circular, Gauss Markova, Turtle, etc. Moreover, to handle the impact of various mobility modules, a handover module is inherited to help device association through access points to prevent data loss.

B. PROPOSED RESOURCE SHARING TECHNIQUES

An adoptive resources scheduling algorithm is proposed that inherits properties from two popular algorithms Ant Colony Optimization (ACO) Algorithm and Earliest Deadline First (EDF) Algorithm. In the traditional ACO algorithm, the ants search for food, which is evaluated in terms of quantity and quality before sharing information with others [47]. Similarly, in the proposed algorithm, the broker sends jobs to different nodes, which send an acknowledgment back after completing the job. The broker evaluates the performance of the device in terms of time, the energy consumed, the queue size, and the estimated waiting time for the next job. This helps the broker maintaining the list of available nodes in terms of QoS. The entire process happens asynchronously. The proposed resource manager execution is shown in algorithm 1. This is the main decision module which helps in selecting the most suitable devices for a given

task execution request to meet the service quality. The QoS varies depend upon the task stringent deadlines. Therefore, the task having hard deadlines are more suitable to execute on the nearby devices, as it can reduce the communication delay as well as the queuing time. However, as the framework relay on volunteer devices, and in case if no device is available, the system negotiate with the source device for additional time so that the task can execute over fog node. If the source device can tolerate additional delay, the task is scheduled at the fog node. Moreover, the task with flexible or no deadline, the selection modules still check the available devices, their computing power and energy. Thus, device having sufficient energy can be opted for the execution. This selection is performed based on distance (d_v using eq. 8) from the source/ requesting device. However, to balanced the workload and to reduce the network traffic, the no deadline tasks are send to cloud data center.

When a task is received at a volunteer/fog/cloud device, it is added to a queue. The queue is ordered using the Earliest Deadline First (EDF) Algorithm [48]. This can help in reducing the re-transmission; on deadline (timer) expiry the system sends a control message to acquire consent to execute the task otherwise, it can simply drop the task. Thus, this way it reduces the retransmission of tasks, and also the failed tasks are dropped from the queue without execution to save time for other tasks to meet the deadline. As illustrated in algorithm 2, an idle device having the queue size zero, the device starts executing the incoming job and sends the result directly to the requesting source device and it also sends an acknowledgment to the broker gateway. This acknowledgment is fruitful in two ways, first, the broker keeps track of each job, means no job will be starved in the queue. Second, the gateway broker ranks this fog node in its list according to the performance which helps in improving the future performance of the system. If the queue size is not zero the incoming job is placed in the queue according to EDF policy. Algorithm 3 gives an overview of the system which describes the processing of the task life cycle.

V. EVALUATION

The proposed simulation framework is evaluated on Ubuntu 16.04 by simulating a smart traffic case scenario.

Case Study – A smart traffic case scenario where roadside units work as fog brokers, dedicated fog nodes are deployed in parking, security cameras, and sensors. The vehicles and pedestrians are used as users and volunteer nodes. In case of any computation requirement, the devices send computation requests to the nearby roadside unit which allocates resources using the list of potential devices. The request size and other networks/ system parameters are given in Table 3. The proposed framework is evaluated in terms of CPU, memory usage, network delay with varying nodes, task acceptance rate, packet drop ratio, execution time, and residual energy. The results are discussed below:

Graphical User Interface Drawing Delay – As the proposed framework is developed as an extension of OMNeT++; therefore, it can run in both Graphical User Interface (GUI) and command line interface mode. However, if the GUI based environment is triggered, GUI setup delay is observed due to components drawing on the screen. Figure 3 depicts the setup delay with increasing

Algorithm 1 Adaptive Resource Sharing Algorithm

Input V: list of volunteer nodes
x_i: incoming request
p: requesting node's position (x,y)
F: List of fog nodes
QoS: Quality of Service
Output S: optimal solution for job execution

$d_{min} \leftarrow +\infty$ ▷ distance between requesting and volunteer node
$D_{min} \leftarrow +\infty$ ▷ computational and queuing delay of fog/volunteer node for request x_i
$S \leftarrow \{\phi\}$
switch QoS **do**
  case 2
    if $Vis\{\phi\}$ **then** $QoS \leftarrow 1$
    else
      for each $v \in V$ **do**
        if $\mathcal{E}_v < \mathcal{E}_F^{request}$ **then** ▷ v have enough energy equ 9
          $d_v \leftarrow Distance(p, v)$ ▷ calculate distance between user and volunteer node using equation 8
          $D_v \leftarrow D_{x_i}^v$ ▷ From equ 4
        end if
        if $s_v < \alpha d_v + \beta D_v$ **then** ▷ α and β are constants
          $S \leftarrow s_v$
        end if
      end for
      return S
    end if
  case 1
    for each $f \in F$ **do**
      $D_f \leftarrow D_{x_i}^f$ ▷ From equ 4
      if $s_f < D_f$ **then**
        $S \leftarrow s_f$
      end if
    end for
    return S
  case 0
    Forward request to cloud servers

Algorithm 2 Earliest-Deadline-Based Task Execution

Input v: fog node; u: user node; x incoming job
Output Nil

$\tau \leftarrow 0$
while *true* **do**
  if v **InIdle then**
    $job \leftarrow Recv(x)$ ▷ Receive job
    $r \leftarrow Exec(job)$ ▷ Execute job
    Send(r, u) ▷ Send result to user node
    Send(Ack,b) ▷ Send acknowledgement to broker
  else
    $Queue \leftarrow Recv(x)$ ▷ Receive and queue job
  end if
  if $queue \neq \{\phi\}$ **then**
    $job \leftarrow Queue.pop()$
    $r \leftarrow Exec(job)$ ▷ Execute job
    Send(r, u) ▷ Send result to user node
    Send(Ack,b) ▷ Send acknowledgement to broker
  end if
  if τ Expires **then**
    $\tau \leftarrow 0$ ▷ Reset timer
    Send(m_{HB},b) ▷ Send heartbeat to broker
  end if
  Send($Beacon$,b) ▷ Update broker about resources
end while=0

Algorithm 3 Gateway Broker– The System Life-cycle

Input *Msg* incoming message
Output Nil

V: list of volunteer nodes
R : *Listofjobs*
p: requesting node's position (x,y)
F: List of fog nodes
QoS: Quality of Service
while *true* **do**
  switch *Msg.Type* **do**
    case Connect
      $Add(f)$ ▷ Add fog to list
      $Send(Ack, f)$
    case ConAck
      $Send(Ack)$ ▷ Handshake
    case JobRequest
      $v \leftarrow FindOptimal(V, x, x_p, x_{QoS})$ ▷ Using algorithm 1
      $R \leftarrow x$ ▷ Insert job to list
      $TaskExecute(v, Msg_u, x)$ ▷ Execute Job Algorithm 2
      Send(Ack,Msg_u) ▷ Send acknowledgement to user
    case JobAck
      $R.Remove(x)$ ▷ Remove job from list
    case Beacon
      Update V ▷ Update list
end while

TABLE 3. Simulation parameters.

#	Parameter	Value/Description
1	Fog Locations	4
2	Fog Brokers	10-15
3	Dedicated Fog Nodes	10-100
4	Volunteer Nodes	50-200
5	User Nodes	50-400
6	Mobility Models	Linear,circular,Turtle, Gauss Markov
7	Fog Node Capacity	1000 MIPS
8	sendInterval	0.5s
9	Request Size γ	1024
10	Cloud Data-center(s)	1
11	Broker App Name	BaseBrokerApp
12	Fog App Name	ComputeFog
13	User App	mqttApp
14	Volunteer User App	VolunteerApp
15	queueType	DropTailQueue
16	WLAN Bitrate	54Mbps
17	radio.transmitter.power	3.5mW
-	**Component**	**Value/Version**
1	OS	Ubuntu 16.04 64x LTS
2	CPU(s)	Intel $Core^{TM}$ i5-1035G1 1 GHz
3	Core(s)	4
4	Threads	8
5	Memory	8 GB
6	Graphics	Nvidia 2 GB
7	Omnet++	4.6
8	INET	3.2.4

number of nodes. Therefore, with the increase in the number of nodes, the delay increases, and for 2000 network nodes it reaches about 650 seconds which is too high; however, no significant delay has been observed CLI mode. Therefore, CLI mode is recommended for large-scale complex simulation having many devices using heterogeneous mobility models.

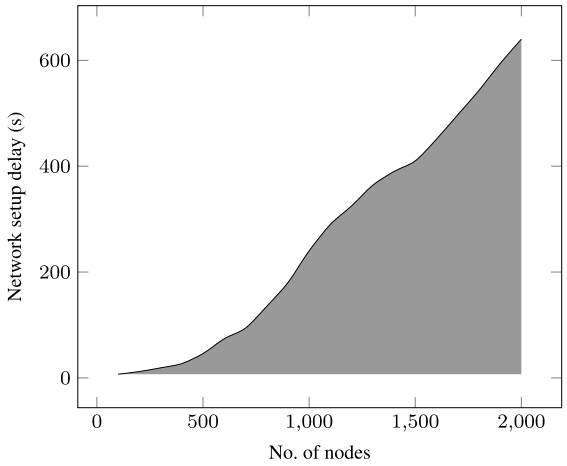


FIGURE 3. One time system booting delay while in GUI mode.

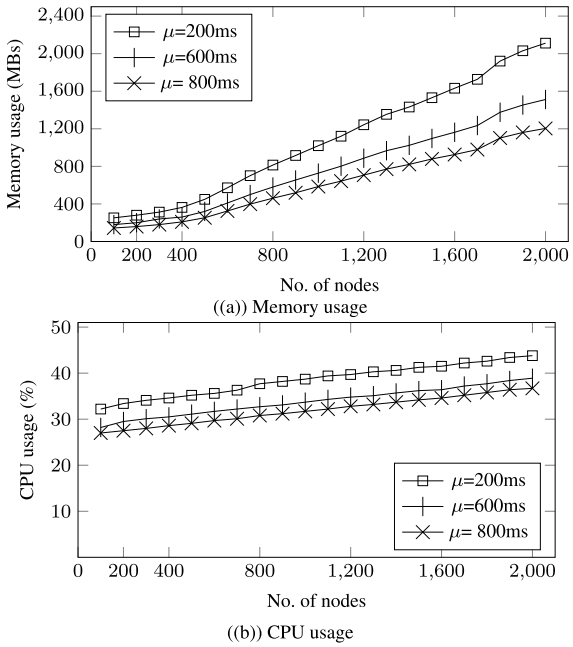


FIGURE 4. Average memory and CPU usage of system according to number of nodes and request rates (μ).

CPU and Memory Usage – In the proposed simulator, each node consists of several sub-modules which include the mobility, application, and energy module. Hence, with the increase in the number of nodes/devices the memory usage and CPU usage increase as well. Figure 4 shows the CPU and memory usage observed with increasing the number of nodes and task generation rate from 200 - 800 ms. Figure 4(b) shows the CPU usage with various task generation rate. As the task rate increases, the system CPU usage also increases. Similarly, memory consumption is also compared with increasing nodes and task generation rate. Figure 4(a) shows that more memory resource usage.

Network Delay – The network delay is a core parameter to understand the quality of service offered by a framework.

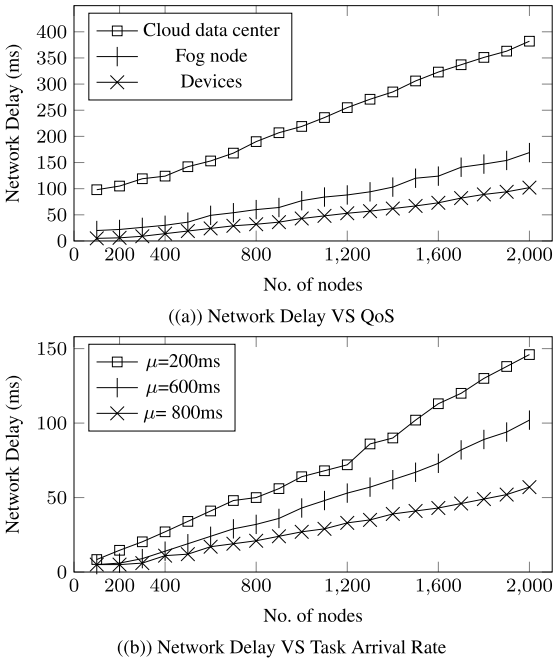


FIGURE 5. Network delay with varying devices and task generation rate.

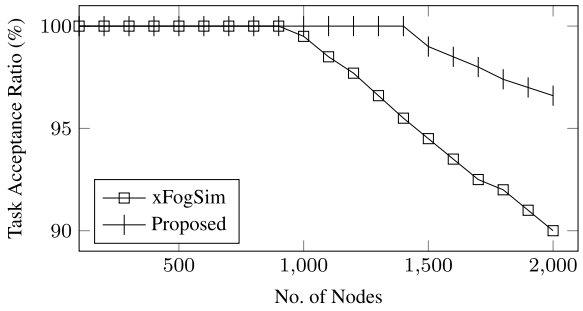


FIGURE 6. A comparison of task acceptance ratio with xFogSim.

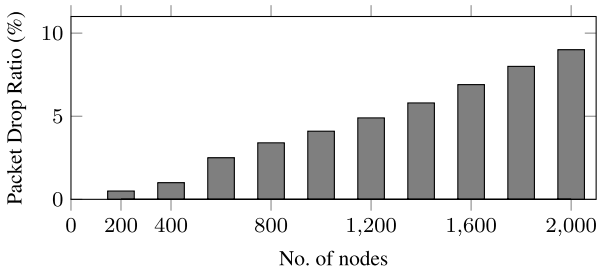


FIGURE 7. Packet drop ratio (%) due to network congestion.

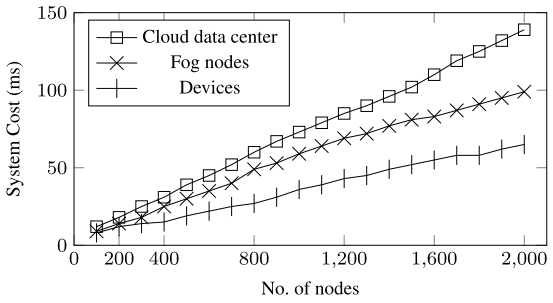


FIGURE 8. The system cost; a combination of queuing cost and job execution cost.

It also depends on the task generation rate. The framework evaluation is performed in terms of quality of service and task generation rate. Figure 5(a) depicts the network delay with varied quality of service. The figure shows the network delay while executing the task at various layers to meet their service demand. The network delay is proportional to the distance between devices and available data rate [49]. The proposed algorithm tries to meet the service requirement through volunteer devices, local execution at fog nodes, or through the help of cloud data centers. However, if no suitable volunteer node is available, the task service demand is met through execution at the fog node. The network delay increases linearly with the increase in the number of nodes because more nodes requesting compute requests which creates long queues at nearby fog locations and causes network traffic congestion. Figure 5(b) shows the network delay with varying task generation rate (200-800ms).

Efficiency(%) – The efficiency is defined as the ratio of executed and total generated tasks. The efficiency of a proposed framework is compared with xFogSim [23] and the results are shown in Figure 6. Task acceptance ratio means the number of tasks a fog/volunteer node accepts. The drop tail queue is implemented at every fog node; thus, after the pre-defined threshold, the fog device stops accepting incoming tasks. Thus, on a higher arrival rate, the xFogsim denying the device resource request due to the limited number of resources available at the fog node. Whereas the proposed framework has more resources due to the usage of volunteer nodes, the task service rate is high.

Packet Drop Ratio (%) – Packet drop is an important measure especially in a wireless environment where the low-level devices can send messages without knowing the channel availability. In our framework, the device can send packets after a 200ms gap to avoid channel congestion; however, due to a large number of devices offloading tasks, a bottleneck is developed at the gateway brokers. Thus, due to the limited buffer size, the gateway drop packets. Figure7 show a packet drop ratio with varying number of nodes. Thus, to resolve such issues, macro levels deployment can facilitate better results as discuss in future directions.

System Cost – The system cost is defined as the sum of communication, queueing delay, and execution time. It is an important measure to gauge the performance of the system to meet the quality of service. Fig. 8 represents the system cost at different levels which utilized to meet the QoS. It is observed that the cost is maximum when tasks are uploaded to cloud data centers because of the communication and queueing delay. Whereas, the cost is minimal when jobs are forwarded to volunteer devices. It is due to the device-to-device communication and limited queue size at volunteer devices. Further, the cost at fog nodes is intermediate as it has long queues due to its centralized regional deployment and intermediate communication delay.

Residual Energy – Fig. 9 shows the average residual energy of volunteer computing devices concerning simulation time. The energy of the devices decreases as

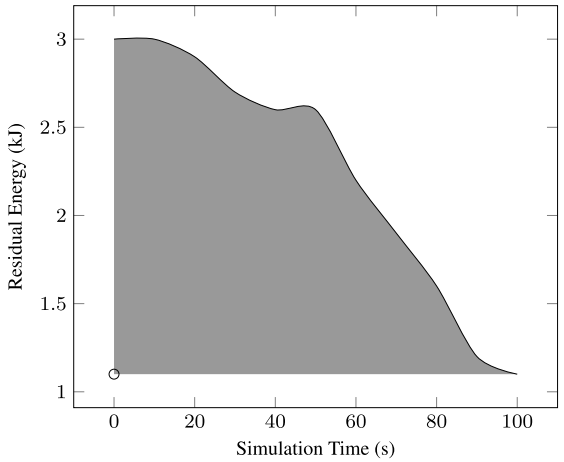


FIGURE 9. The average residual energy of volunteer node with respect to simulation time (s).

simulation time passes as it is consumed for task execution. The proposed framework also provides an energy module to observe energy usage due to resource sharing.

VI. AVAILABILITY

The simulator is developed under GNU General Public License, freely available for academics research purpose. It can be accessed from github at https://github.com/rtqayyum/mFogSim.

VII. CONCLUSION

Fog computing has proven its role for delay-sensitive applications in both domestic and industrial environments. This paper proposed a simulation framework based on a IoT device grid that can act as a fog node to execute the requested tasks. The mobile devices are promoted to fog nodes depending on their energy level and computation capacity. The nodes can move following the pre-defined mobility models. To achieve service quality, an algorithm is proposed which helps in better utilizing the available resources and achieve maximum system throughput. Thus, the framework provides a multi-level resource sharing model. The tasks are allocated based on task deadlines. The evaluation section shows a significant gain in performance by adopting various IoT devices as a work force for the roadside units. However, the proposed work relies on nearby IoT devices; thus, limited availability or access to these devices can effect the performance of the proposed system.

In the future, we are planning to expand this work by incorporating machine learning algorithms to predict device mobility patterns and select the most suitable device for task offloading. Similarly, another way forward is to explore the task dependency issues for the fog computing environment.

REFERENCES

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[2] E. Oughton, Z. Frias, T. Russell, D. Sicker, and D. D. Cleevely, "Towards 5G: Scenario-based assessment of the future supply and demand for mobile telecommunications infrastructure," *Technol. Forecasting Social Change*, vol. 133, pp. 141–155, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0040162517313525
[3] S. K. Biswash, "Node-to-Node communication and mobility management scheme for 5G fog networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2019, pp. 1–6.
[4] A. Morgado, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "A survey of 5G technologies: Regulatory, standardization and industrial perspectives," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 87–97, Apr. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864817302584
[5] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.
[6] J. Byrne, S. Svorobej, A. Gourinovitch, D. M. Elango, P. Liston, P. J. Byrne, and T. Lynn, "RECAP simulator: Simulation of cloud/edge/fog computing scenarios," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2017, pp. 4568–4569.
[7] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.
[8] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "MobFogSim: Simulation of mobility and migration for fog computing," *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102062. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X19301935
[9] S. Forti, A. Pagiaro, and A. Brogi, "Simulating fogdirector application management," *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102021. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X19301522
[10] Q. Wang, "Pfogsim: A simulator for evaluating dynamic and layered fog computing environments," M.S. thesis, Graduate Fac., Auburn Univ., Auburn, AL, USA, May 2019. [Online]. Available: https://etd.auburn.edu/bitstream/handle/10415/6702/Qian%20Master%20Thesis.pdf?isAllowed=y&sequence=2
[11] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for Internet of Things computations," in *Proc. Cloudification Internet Things (CIoT)*, Nov. 2016, pp. 1–6.
[12] *Simplesoft Simpleiotsimulator*. Accessed: Aug. 10, 2018. [Online]. Available: http://www.smplsft.com/SimpleIoTSimulator.html
[13] (Apr. 2020). *Opnet Network Simulator*. [Online]. Available: https://opnetprojects.com/opnet-network-simulator/
[14] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "IFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509
[15] M. Seufert, B. K. Kwam, F. Wamser, and P. Tran-Gia, "Edgenetwork-cloudsim: Placement of service chains in edge clouds using network-cloudsim," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jul. 2017, pp. 1–6.
[16] C. Mechalikh, H. Taktak, and F. Moussa, "PureEdgeSim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2019, pp. 700–707.
[17] M. M. Lopes, W. A. Higashino, M. A. M. Capretz, and L. F. Bittencourt, "MyiFogSim: A simulator for virtual machine migration in fog computing," in *Proc. Companion Proc. the10th Int. Conf. Utility Cloud Comput.*, Dec. 2017, pp. 47–52, doi: 10.1145/3147234.3148101.
[18] M. I. Naas, J. Boukhobza, P. Raipin Parvedy, and L. Lemarchand, "An extension to iFogSim to enable the design of data placement strategies," in *Proc. IEEE 2nd Int. Conf. Fog Edge Comput. (ICFEC)*, May 2018, pp. 1–8.
[19] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 39–44.
[20] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, and V. Vlassov, "SpanEdge: Towards unifying stream processing over central and near-the-edge data centers," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 168–178.
[21] Z. Nikdel, B. Gao, and S. W. Neville, "DockerSim: Full-stack simulation of container-based software-as-a-service (SaaS) cloud deployments and environments," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process. (PACRIM)*, Aug. 2017, pp. 1–6.
[22] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *Proc. IEEE Fog World Congr. (FWC)*, Oct. 2017, pp. 1–6.
[23] A. Malik, T. Qayyum, A. Rahman, M. Khattak, O. Khalid, and S. Khan, "XFogSim: A distributed fog resource management framework for sustainable IoT services," *IEEE Trans. Sustain. Comput.*, early access, Sep. 18, 2020, doi: 10.1109/TSUSC.2020.3025021.

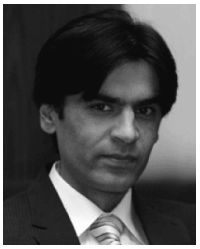
[24] D. Belli, S. Chessa, B. Kantarci, and L. Foschini, "A capacity-aware user recruitment framework for fog-based mobile crowd-sensing platforms," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Dec. 2019, pp. 1–6.
[25] R. J. Tom, S. Sankaranarayanan, V. H. C. de Albuquerque, and J. J. P. C. Rodrigues, "Aggregator based RPL for an IoT-fog based power distribution system with 6LoWPAN," *China Commun.*, vol. 17, no. 1, pp. 104–117, Jan. 2020.
[26] Contiki. (2011). *Contiki OS*. Accessed: Feb. 3, 2020. [Online]. Available: http://www.contiki-os.org/.
[27] N. B. V. and R. M. R. Guddeti, "Heuristic-based IoT application modules placement in the fog-cloud computing environment," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, Dec. 2018, pp. 24–25.
[28] H. Huang, F. Liu, Z. Yang, and Z. Hao, "Automated test case generation based on differential evolution with relationship matrix for iFogSim toolkit," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 5005–5016, Nov. 2018.
[29] K. S. Awaisi, A. Abbas, M. Zareei, H. A. Khattak, M. U. S. Khan, M. Ali, I. Ud Din, and S. Shah, "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159100–159111, 2019.
[30] F. H. Rahman, S. H. S. Newaz, T. W. Au, W. S. Suhaili, and G. M. Lee, "Off-street vehicular fog for catering applications in 5G/B5G: A trust-based task mapping solution and open research issues," *IEEE Access*, vol. 8, pp. 117218–117235, 2020.
[31] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.
[32] B. K. Dar, M. A. Shah, S. U. Islam, C. Maple, S. Mussadiq, and S. Khan, "Delay-aware accident detection and response system using fog computing," *IEEE Access*, vol. 7, pp. 70975–70985, 2019.
[33] K. Deep Singh and S. K. Sood, "5G ready optical fog assisted cyber physical system for IoT applications," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 5, no. 2, pp. 137–144, Jun. 2020.
[34] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing," *IEEE Access*, vol. 7, pp. 115760–115773, 2019.
[35] M. Ammad, M. A. Shah, S. U. Islam, C. Maple, A. A. Alaulamie, J. J. P. C. Rodrigues, S. Mussadiq, and U. Tariq, "A novel fog-based multi-level energy-efficient framework for IoT-enabled smart environments," *IEEE Access*, vol. 8, pp. 150010–150026, 2020.
[36] A. Coutinho, H. Rodrigues, C. Prazeres, and F. Greve, "Scalable fogbed for fog computing emulation," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, May 2018, pp. 00334–00340.
[37] R. Buyya and S. N. Srirama, *Modelling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit*. Hoboken, NJ, USA: Wiley, 2019, pp. 433–465. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119525080.ch17, doi: 10.1002/9781119525080.ch17.
[38] Q. Xu and J. Zhang, "PiFogBed: A fog computing testbed based on raspberry pi," in *Proc. IEEE 38th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Oct. 2019, pp. 1–8.
[39] J. Hasenburg, M. Grambow, E. Grunewald, S. Huk, and D. Bermbach, "MockFog: Emulating fog computing infrastructure in the cloud," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, Jun. 2019, pp. 144–152.
[40] I. Behnke, L. Thamsen, and O. Kao, "Héctor: A framework for testing IoT applications across heterogeneous edge and cloud testbeds," in *Proc. 12th IEEE/ACM Int. Conf. Utility Cloud Comput. Companion*, May 2019, pp. 15–20.
[41] J. Wei, S. Cao, S. Pan, J. Han, L. Yan, and L. Zhang, "SatEdgeSim: A toolkit for modeling and simulation of performance evaluation in satellite edge computing environments," in *Proc. 12th Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2020, pp. 307–313.
[42] N. Gautam, *Analysis of Queues: Methods and Applications*. Boca Raton, FL, USA: CRC Press, 2012.
[43] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.
[44] A. W. Malik, A. Ur Rahman, M. Ali, and M. M. Santos, "Symbiotic robotics network for efficient task offloading in smart industry," *IEEE Trans. Ind. Informat.*, early access, Oct. 19, 2020, doi: 10.1109/TII.2020.3032238.
[45] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
[46] Y. Al-Nidawi, H. Yahya, and A. H. Kemp, "Impact of mobility on the IoT MAC infrastructure: IEEE 802.15.4e TSCH and LLDN platform," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 478–483.
[47] X. Liu, "Sensor deployment of wireless sensor networks based on ant colony optimization with three classes of ant transitions," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1604–1607, Oct. 2012.
[48] A. Ahmad, R. Arshad, S. A. Mahmud, G. M. Khan, and H. S. Al-Raweshidy, "Earliest-deadline-based scheduling to reduce urban traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1510–1526, Aug. 2014.
[49] H. Pucha, Y. Zhang, Z. M. Mao, and Y. C. Hu, "Understanding network delay changes caused by routing events," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2007, pp. 73–84, doi: 10.1145/1254882.1254891.

TARIQ QAYYUM received the M.S.I.T. degree from the National University of Sciences and Technology, Islamabad, Pakistan.

He worked as a Research Assistant with the Pakistan Institute of Engineering and Applied Sciences, under a project Cyber Infrastructure Protection and Malware Analysis (CIPMA) funded by the Higher Education Commission of Pakistan (HEC) and the National Center for Cyber Security (NCCS), Pakistan. His research interests include network security, cloud computing, fog computing, the IoT, and distributed systems.

ZOUHEIR TRABELSI received the Ph.D. degree in computer science from the Tokyo University of Agriculture and Technology, Japan.

He is currently a Professor of Network Security with the College of Information Technology (CIT), United Arab Emirates University (UAEU). Prior joining UAEU, he was a Computer Science Researcher with the Central Research Laboratory, Hitachi, Tokyo, Japan, for four years. His research interests include zero trust architecture, network security, intrusion detection and prevention, firewalls, network covert channels, information security education, and curriculum development.



ASAD WAQAR MALIK (Senior Member, IEEE) received the master's and Ph.D. degrees from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, with a major in parallel and distributed simulation/systems.

He is currently an Assistant Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST-SEECS). He is also a Senior Lecturer with the Department of Information Systems, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. His primary areas of interests include distributed simulation, cloud/fog computing, and the Internet of Things.

KADHIM HAYAWI (Member, IEEE) received the M.Sc. degree in computer science from Dalhousie University, Canada, in 2004, and the Ph.D. degree from the University of Waterloo, Canada, in 2018.

He is currently an Assistant Professor with the College of Technological Innovation, Zayed University, and a member of the Cybersecurity and Digital Forensics research group, where he teaches a wide variety of courses and pursues his research endeavors. He earned several prestigious industry certifications and has over 17 years of experience in academia and industry. His research interests include information security and privacy challenges of emerging technologies, such as the IoT, cloud, social networks, blockchain, zero trust architecture, digital health, and healthcare data analytics.