

Received August 27, 2018, accepted October 6, 2018, date of publication October 23, 2018, date of current version November 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2877696

FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment

TARIQ QAYYUM¹, ASAD WAQAR MALIK¹,
MUAZZAM A. KHAN KHATTAK¹, (Senior Member, IEEE),
OSMAN KHALID², AND SAMEE U. KHAN³

¹Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan

²Department of Computer Science, COMSATS University Islamabad, Islamabad 45550, Pakistan

³Electrical and Computer Engineering Department, North Dakota State University, Fargo, ND 58105, USA

Corresponding author: Asad Waqar Malik (asad.malik@seecs.edu.pk)

ABSTRACT Fog computing is a technology that brings computing and storage resources near to the end user. Being in its infancy, fog computing lacks standardization in terms of architectures and simulation platforms. There are a number of fog simulators available today, among which a few are open-source, whereas rest are commercially available. The existing fog simulators mainly focus on a number of devices that can be simulated. Generally, the existing simulators are more inclined toward sensors' configurations, where sensors generate raw data and fog nodes are used to intelligently process the data before sending to back-end cloud or other nodes. Therefore, these simulators lack network properties and assume reliable and error-free delivery on every service request. Moreover, no simulator allows researchers to incorporate their own fog nodes management algorithms, such as scheduling. In existing work, device handover is also not supported. In this paper, we propose a new fog simulator called FogNetSim++¹ that provides users with detailed configuration options to simulate a large fog network. It enables researchers to incorporate customized mobility models and fog node scheduling algorithms, and manage handover mechanisms. In our evaluation setup, a traffic management system is evaluated to demonstrate the scalability and effectiveness of proposed simulator in terms of CPU and memory usage. We have also benchmarked the network parameters, such as execution delay, packet error rate, handovers, and latency.

INDEX TERMS Cloud, fog, edge network, IoT, mobility models, OMNeT++.

I. INTRODUCTION

Over the years, there is a tremendous paradigm shift in Information and Communication Technology from isolated and limited computing environments to pervasive computing. The advancement in telecommunication and manufacturing industry resulted in development of powerful smart-devices that can ubiquitously join available network at any time. Recent years have seen the evolution of 5G as a future technology. The inclusion of 5G technology not only supports the heterogeneous wireless network connectivity but also provides a wide spectrum to different applications that can take benefit from 5G to improve their performance in terms of e.g., latency, response time, and energy consumption. With advancement in technology, the devices can become a part of a grid and generate significant amount of data that is typically

sent to cloud for processing. According to Gartner² (2017) report, around 8.4 billion devices will be connected to the Internet by the end of 2017. The raw data generated through various sensors require intelligent processing to reduce the bandwidth usage and improve the latency. Therefore, the need to bring resources close to the end user is rising. Fog is one of the emerging edge computing paradigms. The term fog was first introduced by Cisco [1]. As compared to cloud data centers, fog provides a virtualized computing environment deployed closer to the end user [2]. The fog resides between cloud and end-user devices. The cloud and fog provide similar services to end users but the fog is deployed to facilitate specific geographic region. Fog cannot exist standalone rather it augments the cloud computing. It is designed to support delay-sensitive applications, whereas cloud being far away

¹available at <https://fognetsimpp.com>

²<https://www.gartner.com/newsroom/id/3598917>

from the user can exhibit higher latency. The fog provides services to IoT applications from the edge network as well as from the devices such as routers, access points, Road Side Units (RSUs) and other user devices. With the deployment of fog nodes, the reliability, fault tolerance, and scalability of the devices can easily be managed. Thus, it also reduces the bandwidth usage between fog node and backend cloud data centers. The fog is an emerging paradigm but there are several challenges it faces, such as heterogeneous device management, fog nodes architecture, security, device mobility, and privacy. Moreover, interoperability among heterogeneous devices is also a challenging task. Data coming from different devices need to be analyzed and forwarded to other devices for timely decision making. The typical fog paradigm consists of limited fog nodes that provide virtualized computing services. Therefore, to efficiently manage resources requires sophisticated scheduling algorithms. Fig 1 depicts a general scenario where the efficiency is achieved by implementing the fog layer in between the local network and cloud. Similarly, like a cloud, proper management of resources usage and billing is the mandatory requirement of fog paradigm. Fog provides a flexible environment to end devices, but is difficult to manage due to its decentralized paradigm. The fog nodes can be deployed at any place between the device and cloud infrastructure. As fog is designed to support latency-sensitive application, Smart Gateways (SG) are the best location to offer fog computing [3].

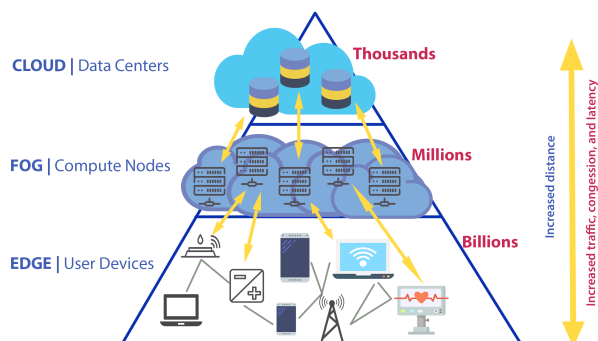


FIGURE 1. Computing architecture pyramid.

Fog computing is based on expensive devices and network equipment. It is important to perform pre-deployment testing of fog platform using simulation tools. However, no such standard simulation tool/framework is available at the moment for the fog computing that makes fog simulation an open research issue as well.

Moreover, the existing fog simulators lack basic features that we explain in the subsequent text. The existing simulators are mainly focused on homogeneous devices. The devices send the data to a centralized node where the data is processed before sending to the cloud. There exist Java-based simulators which do not consider the network properties and thus, only simulate the idealistic environment with no packet drop/error rate, network congestion, and channel collision. The

simulators only provide limited or simple mobility models and lacking features like handover. The simulators are not open source or proprietary and do not support modification. Thus, researchers cannot incorporate their own resource management algorithms for testing and modification. Moreover, energy consumption and cost computation models are also not supported in existing simulators.

Contribution – In this paper, we have proposed a fog simulator named as FogNetSim++. In our proposed FogNetSim++, a user can simulate heterogeneous devices with varying features. It also supports the handover feature to track the source or requested device. Thus, after computation, results can be delivered through different fog nodes deployed at the distinct geographical region. The FogNetSim++ architecture is flexibly designed so that researchers can incorporate their own algorithms by extending the base classes. Previously no such simulators exist that facilitate in such a manner; thus, FogNetSim++ enables users to model, simulate, and evaluate the different realistic fog scenarios. It also provides a facility to create a network environment that allows the static and dynamic nodes in the network and use various fog protocols for communication, such as Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Advanced Message Queuing Protocol (AMQP). The FogNetSim++ includes a range of mobility models, such as StationaryMobility, StaticGridMobility, CircleMobility, LinearMobility, TractorMobility, RectangleMobility, and TractorMobility. The user can extend these models to fulfill their requirements. Moreover, energy module is also incorporated for mobile devices. All the fog nodes are managed through a central broker, thus various resource scheduling algorithms can also be incorporated in the proposed simulator.

The rest of the paper is organized as follows: Section 2 briefly describes the related work. In Section 3, FogNetSim++ system model and design is explained. The experimental setup and simulation details are presented in Section 4. Finally, we conclude this paper in Section 5 and discuss future research directions pertaining to the FogNetSim++ simulator.

II. LITERATURE REVIEW

This section covers details about some of the commonly used simulators, with comparison to the proposed FogNetSim++. Han *et al.* [4] proposed DPWSim, designed for IoT applications. It supports the service-oriented and event-driven model. Sotiriadis *et al.* [5] proposed SimIoT which is designed as an extension of SimIC simulation framework. It provides several communication mechanisms for IoT sensors and cloud data centers. Similarly, a simulation tool EdgeCloudSim is proposed to reduce the barriers in the conventional cloud simulators for simulating the edge computing scenarios. It is designed as an extension to CloudSim [6].

Khan *et al.* [7] proposed a coordination technique to handle large number of IoT devices. The framework is implemented on CloudSim and supports home automation applications.

Zeng *et al.* proposed IoTSim that enables the big data processing in IoT environment. The authors have adopted a MapReduce model and presented a case study to show the effectiveness of proposed simulator. Pflanzner *et al.* [8] presented a mobile based IoT simulator called MobIoTsim. The main objective is to provide a platform to researchers to learn IoT device handling without buying extra sensors and devices. Through MobIoTsim, users can explore the detailed working of IoT systems and help in building initial understanding of a complete IoT connected system.

Fiandrino *et al.* [9] proposed a CrowdSenSim, which is designed to simulate crowd source applications such as smart cities. To evaluate the proposed simulator, authors have simulated a street lighting scenario. However, the simulator can be used for other applications that require data gathering from various diverse sources. The CrowdSenSim is freely available for researchers and users.

SimpleIoTSimulator [10] is commonly used tool to simulate IoT environment with sensors and devices. It also supports number of IoT specific protocols such as MQTT, CoA, etc. However, SimpleIoTSimulator requires 64-bit RedHat Linux environment for successful installation and execution of all the packages. Similarly, IBM offered PaaS enabled IoT simulator termed as IBM Bluemix [11]. It provides a web based interface for quick deployment of cloud applications that can gather data from various sensors and devices. It provides support for hardware devices of ARM, Intel, and Texas Instrument. The data is sent to cloud data center through MQTT protocol. A platform-as-a-service enabled simulator - Parse [12] is launched by Facebook. It supports IoT devices. It provides easy application development environment and also supports various mobile devices. The Google Cloud framework includes Google's IoT solution that provides various Google services [13]. The proposed framework is highly scalable, and allows large number of devices, gathers data and provides a visualization platform. In particular, data from devices are received at Load Balancer and from here it is disseminated to the next layer i.e., AppEngine.

Gupta *et al.* [14] proposed an IoT and fog simulator called iFogSim. It is designed as an extension of CloudSim using Java technology. The authors presented the impact of resource management techniques in terms of latency, congestion, and cost. However, there are few drawbacks: (a) it is based on Java, therefore, core network parameters are not supported, (b) it is not compatible with different versions of java, moreover, the documentation is not extensive to support development. In our proposed simulator, a complete network toolbox is available to simulate delay, latency, congestion, packet drop, thus, it gives the flexibility to simulate diverse networks with variable characteristics.

A Cisco solution based IoT simulation testbed is proposed – called FIT IoT Lab [15]. It also supports open-source solutions. It allows the use of wireless nodes deployed across France. The sensors and gateways are accessed through the remote interface. However, the simulator does not conduct

simulation in a controlled environment, therefore, the simulation cannot be repeated.

WSNet is simulator designed to simulate wireless networks [16]. It is a discrete event simulator. It can be used to simulate IoT network. It already includes modules such as mobility, energy, radio interfaces, and routing protocols. Moreover, it also supports simulation of disasters such as earthquake, and fire.

Conti *et al.* [23] proposed a scheme that help fog computing nodes with battery to store energy that can be used to execute compute intensive tasks. The power storage can help fog node during peak load time even with low power generation. The authors proposed reinforcement based learning technique to minimize the job loss.

Name *et al.* [24] discussed the resource allocation and user mobility. The authors proposed the algorithm to optimize the resource allocation and handle the handover for mobile users in fog environment.

Meng *et al.* [25] studied the delay-constraint model for computation offloading on Cloud or fog computing servers. The main goal is to minimize the energy consumption in terms of computation and communication keeping in consideration the delay constraint of every task. The proposed hybrid approach has reduced the overall energy consumption.

In fog computing, usually the resources are limited compared to cloud; therefore, typical cloud resource allocation algorithms cannot fulfill the rapid changing needs of the user. Ni *et al.* [26] proposed a resource allocation algorithm for fog environment using priced timed petri nets. Priced timed Petri net is the variant of Petri nets that define the system based on price and time. It is used to characterize the dynamic behavior of a system and also help in effectively analyzing the performance in terms of time and cost. The proposed scheme considers price and completion time of a task. Moreover, the algorithm that predicts the time required for a particular task is also proposed in this work.

Fog provides a middle-ware for the applications of IoT where sensors/devices generate significant amount of data [27]. Similarly, the energy management of edge devices is also very crucial. Xu *et al.* [28] proposed a dynamic energy management framework that learn on-the-fly and dynamically manage the workload offloading mechanism.

Ananthanarayanan *et al.* [29] proposed a geographical distributed framework for large-scale video analytics. To support real-time video analytics, gateway nodes are used to fulfill the computing requirement. Ren *et al.* [30] discussed a scalable framework for edge computing. The framework is designed to support smart home applications, the collected data from various sensors are sent to decision making nodes, the communication can be reduced with the installation of nearby computing nodes [31].

A video transmission model is proposed by Xu *et al.* [32] where Mobile Edge Computing (MEC) server is used to control the video caching mechanism. Many well-known content providers, such as Youtube, Netflix, etc., are caching contents at the edge, thus reducing the latency. Chen *et al.* [33]

TABLE 1. Simulator comparison.

Simulators	Prog. Language	Platform	Network Configuration	Open Source	Mobile Nodes	Customize Mobility Models	Scheduling Algorithms	Device Handover	Energy Module
EdgeCloudSim [6]	Java	All	No	Yes	Yes	No	Yes	No	No
MobIoTsim [8]	Java	Linux	No	Yes	Yes	No	No	No	No
SimpleIoTsimulator [10]	Java	Unix	No	No	Yes	No	No	No	No
IBM BlueMax [11]	Java/Python	Cloud	No	No	Yes	No	No	No	No
Google IoT Sim [13]	NA	Cloud	No	No	Yes	No	No	No	No
iFogSim/MyiFogSim [14]	Java	All	No	Yes	Yes	No	Yes	No	No
Cooja [17]	C	Linux	No	Yes	Limited	No	No	No	No
FogTorch [18]	Java	All	No	Yes	No	No	No	No	No
RECAP simulator [19]	N/A	–	Limited	N/A	No	No	No	N/A	N/A
EmuFog [20]	Python	All	Yes	Yes	No	No	No	No	No
Edge-Fog cloud [22]	Python	All	No	Yes	Limited	No	No	No	No
Mobile Fog [23]	N/A	–	No	No	Yes	No	No	No	No
FogNetSim++	C++	All	Yes	Yes	Yes	Yes	Yes	Yes	Yes

proposed a device-to-device crowd sourced framework for MEC using 5G technology. In 5G technology, a device can communicate with other devices. Therefore, a large number of devices can require computing resources at the edge location. Sonmez *et al.* [6] proposed a framework that defines the strategies for better resource sharing and task execution in collaborative and energy-efficient manner.

Tang *et al.* [34] proposed a distributed fog architecture for the integration of different infrastructures, services, and components for smart cities. The authors claimed that with the deployment of fog node, the response time has been reduced significantly. Li *et al.* [35] proposed a cooperative framework to enhance the data collection quality in fiber channel networks. The delay constraints are fulfilled through the concept of offloading at dynamic fog nodes. Meng *et al.* [25] proposed a hybrid offloading framework that can offload data and computation at fog nodes and cloud data centers. The main objective is to reduce the overall energy consumption for computation and communication.

Mukherjee *et al.* [36] discussed the security issues in IoT environment. Similarly, Hu *et al.* [37] discussed the security and privacy of face detection applications while computations are performed at the fog layer. The commonly used protocols for the fog paradigm are publish/subscription based. A content-based publish subscribe scheme is proposed to handle privacy issues [38]. Similarly, the fog architecture is also useful for vehicular applications that require quick computation for real-time decision making. A fog architecture for vehicular computing is proposed by Bonomi *et al* [2].

Most of the simulators discussed above are not flexibly designed to facilitate researchers to incorporate their own algorithms. Moreover, the existing simulators are mainly focused on devices they can support; however, the network protocols and communication stack are not fully considered. To test and analyze the algorithms, there is a need for flexible, extensive, and rapidly deployable fog simulator. The detailed comparison is shown in Table 1. In next section, the details of the proposed simulator are discussed to address the above-mentioned issues.

III. FOGNETSIM++: TOOLKIT FOR MODELING AND SIMULATION

A. SYSTEM MODEL

The proposed system can support P mobile devices (d), M fog nodes and a broker node (B). The mobile devices can communicate with the broker node through the base station (BS). The mobile devices can be of different types, either they are sensing devices which only generate data for further processing at the fog node or they can offload task for computation. Therefore, the mobile device can execute an application which requests for computing capacity at the fog node. Here, we have implemented a traffic model as an M/M/1 queue on mobile device and M/M/c queue on fog nodes. Every mobile device can generate a service request to the fog node through its BS. If the cumulative request rate is less than the accepted rate of fog node, then all the requests will be accepted to process on fog node. Otherwise, the request is further subleased to the neighboring fog nodes. This entire process is managed by the broker node. The broker node continuously monitors the fog nodes, their queue length, and sizes of every request. Based on these parameters, it can sublease the tasks to other fog nodes. Moreover, here we have assumed that offloading task to the back-end cloud data center can further add the delay, however, the option is still available at the broker node which can be utilized in extreme cases where the request rate increases exponentially. Moreover, the requests generated from mobile devices ($d_i, i \in P$) follows a Poisson process model, where (μ_i) is the average request arrival rate.

Every request generated from device (d_i) is computationally intensive, totally independent and can be scheduled on any fog node or cloud data center. Further, there are k homogeneous fog nodes that can be used to execute the received task. In every fog node location, there are q nodes available to provide services. The execution rate is represented as θ_f . The maximum workload of the fog node is represented as ϕ_f . The reason to define the max workload is to evade unnecessary queuing delay. The execution request from a d_i is first received at the B . Thus, according to the Poisson process,

we can write the total arrival rate is as follows:

$$\mu_{total} = \sum_{i=0}^P \mu_i. \tag{1}$$

Further, the request that every fog node can accept is stated as:

$$\varphi = \begin{cases} 1 & \phi_f > \mu_{total} \\ \frac{\phi_f}{\mu_{total}} & \phi_f < \mu_{total} \end{cases}. \tag{2}$$

Therefore, using (1) and (2), we can compute the execution rate at fog nodes as:

$$\vartheta = \mu_{total} \times \varphi = \begin{cases} \mu_{total} & \phi_f \geq \mu_{total} \\ \phi_f & \phi_f < \mu_{total} \end{cases}. \tag{3}$$

Using Erlang’s formula [39] and the queuing theory analysis, we can compute the average waiting time of each request at fog node is as follows.

$$T_{wait} = \frac{\kappa, \frac{\mu_{total}}{\kappa\theta_f}}{\kappa\theta_f - \mu_{total}} + \frac{1}{\theta_f}. \tag{4}$$

The broker node is managing all the fog nodes and if the fog node cannot process any new request, the broker forwards the overloaded request to neighboring fog nodes which are underutilized. Moreover, in extreme case, the task can be transferred to the cloud data center (we are not considering this case here). Therefore, if no fog node is available which can accept the new task, the broker can hold the task in its own queue for time (t). After the expiry of time, it can forward the task to available fog node. Otherwise, after the expiry of time t , the broker can drop the task and request for re-transmission.

Energy Model – FogNetSim++ also provides the energy model. The energy of a given task can be computed using (5). Where E_t represents the energy consumption for task t . It is the combination of device energy and energy consumed at fog node. At device end, the energy is consumed in uploading the task t of size $S(t)$ on data channel of bandwidth B with its transmission power P_{trans} . Whereas, the power consumed in processing task t of size $S(t)$ at fog node (FN) with computing capacity γ_{fn} . P_{idle} represents the idle power. The state diagram of the current energy model is presented in the Fig. 2. The (6) represents the total energy for Q tasks.

$$E_t^{fn} = \frac{S(t)}{B} \cdot P_{trans} + \frac{S(t)}{\gamma_{fn}} \cdot P_{idle}. \tag{5}$$

$$E = \sum_{t=0}^Q E_t^{fn}. \tag{6}$$

Pricing Model – FogNetSim++ also provides different pricing models. The central broker monitors the resource usage. The pricing model available in FogNetSim++ is listed in Table 2. For the pricing model, the broker is responsible for managing the user SLA (service level agreement).

The proposed FogNetSim++ is an extensive simulator designed to support task execution inside the fog. Usually, the resources available at fog level are less compared to

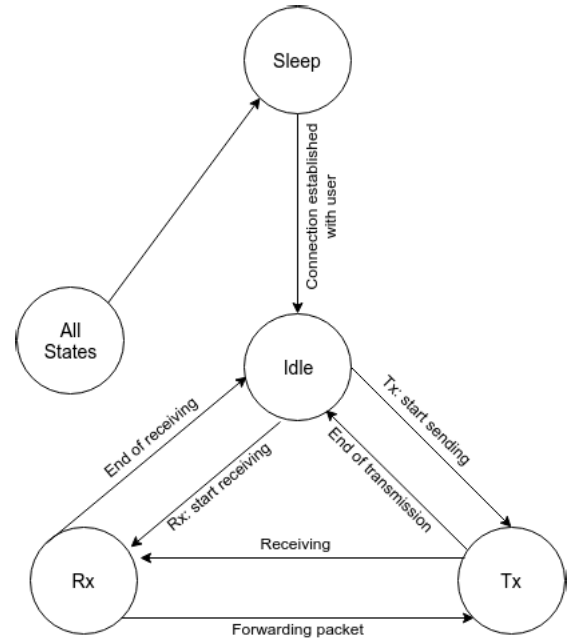


FIGURE 2. State diagram for energy model.

TABLE 2. Pricing models supported in FogNetSim++.

No.	Pricing Model	Features/Description	Price
1.	Pay-as-you-go	Networking, Storage Compute	\$.0004/Mb
2.	Subscription	Monthly Task size 10Gb Monthly Task size 100Gb	\$50/User \$450/User
3.	Pay-for-resources	Storage Compute	\$0.0048/GB \$.0002/Mb
4.	Hybrid Model	A dynamic model changes according to the queue size of broker	\$.0004-\$.0008

the cloud data center. Therefore, efficient utilization of fog resources is highly desirable. Moreover, the proposed framework is designed with a centralized task manager i.e., broker node. The broker is responsible for managing the entire fog resources. Initially, the device contacts broker with its computational requirements. On successful allocation of fog node, the device directly sends the task to the assigned node. Thus, this reduces the overhead at the broker node. Moreover, the proposed work also supports sensing devices, which can only generate data, and other devices can acquire that data through subscription. The rest of the technical details are discussed in the next subsection.

B. IMPLEMENTATION

The FogNetSim++ is developed as an extension of OMNet++. The OMNet++ is commonly used open source tool that provides an extensive library to simulate network characteristics. It provides a number of built-in modules that act like realistic network devices. Fig. 3 shows the design of proposed FogNetSim++. In FogNetSim++, all the available modules of OMNet++ can be easily integrated. The researchers can easily modify the modules of FogNetSim++

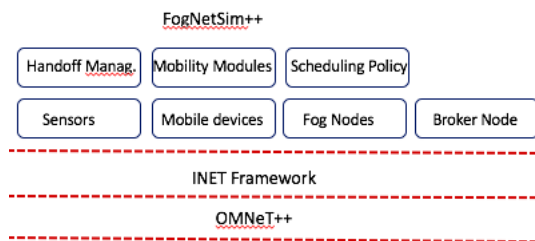


FIGURE 3. FogNetSim++ design.

to simulate their scenarios. The main objective of the development of FogNetSim++ is that the existing frameworks are designed to support different sensors [10], [11]. However, they have ignored the network characteristics such as error rate and data rate that can play a critical role in simulating latency-sensitive applications. Similarly, the existing work does not support mobility or provide simple mobility models [40]. The graphical interface of FogNetSim++ is shown in Fig. 4. Our proposed FogNetSim++ present the complete simulation environment that includes sensors, fog nodes, a broker node and geographically distributed data centers. The broker is responsible for managing the fog nodes, it receives the service request and assigns fog nodes keeping in view the utilization of every fog node. It is a novel framework that supports both mobile and static nodes, devices, and gateways. The modular design of FogNetSim++ is shown in Fig 3. The FogNetSim++ comprises of two main modules i.e., end devices and broker.

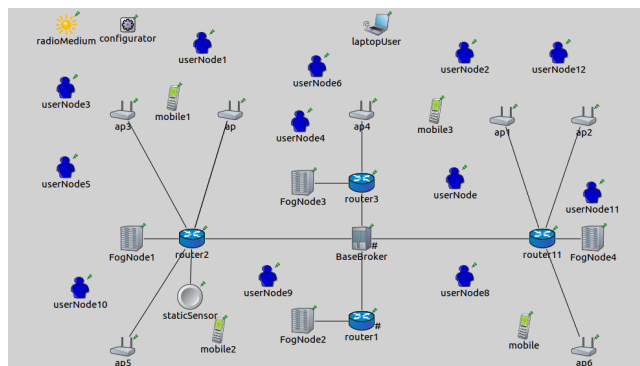


FIGURE 4. FogNetSim++ - A graphical user interface.

The core modules of FogNetSim++ are broker, fog node, and the end devices. The broker is the centralized resource manager that keeps track of all the fog nodes. The fog node provides a computation service. The end device is the actual node/sensor that can move during communication and generate requests for computing nodes. The broker node receives a request from devices to perform the requested computation. The broker node maintains a queue from where the requests are served on first come first serve basis. Thus, the broker is responsible for task scheduling, execution, and handover. The handover module is used in two cases, i). when the node is

moving from one region to another and its requested task is being executed. In that case, broker dynamically transfers the unallocated task to its neighbor fog node; ii). the requested task is transferred to the neighboring node where fog node is available or lightly loaded. The latter is used when directly connected fog node is heavily loaded, and cannot fulfill the request due to large pending tasks, as shown in Fig 5. The broker nodes are also connected to a backbone network which further connects it with the cloud data center. The computed results/data can be sent to the data center for storing and to perform analysis at a later stage. Usually, there are limited fog nodes that provide computation. Every fog node provides services to a large number of devices. In a typical case, it is used to gather data from sensors, and the only meaningful information is sent to the data center. In proposed FogNetSim++, researchers can incorporate their own request scheduling algorithms at the broker node to support the execution of tasks at the fog node. Moreover, the sophisticated handover algorithm can also be incorporated very easily. This can introduce a new research direction in fog simulators. In all the previous simulators, no such scheduling algorithms are available that allow researcher to utilize fog nodes efficiently.

The network module in FogNetSim++ provides complete network physical/wireless link properties such as packet drop, retransmission, bit error rate, and link bandwidth. The FogNetSim++ provides flexibility in network configurations. The user can define network parameters according to their design requirements. Thus, the more realistic network can be simulated. The FogNetSim++ includes a number of communication protocols, that can be used to simulate diverse scenarios. At present, the available protocols are TCP, UDP, FTP, HTTP, MQTT, CoAP, and AMPQ. The HTTP, TCP, and UDP can also be used for communication with the data center. The FogNetSim++ supports heterogeneous devices and fog nodes with a variable number of applications that can execute over each node concurrently. Moreover, the applications can use different communication protocols. Further, it also supports IPv4 and IPv6; however, both cannot be used simultaneously. Moreover, the broker can be used to dispatch the data to the requested devices; therefore, publish, subscribe mechanism is implemented at the broker node. Any device can register with the broker either as a subscriber, publisher or both. Thus, the broker tracks the device location and share updates to all the subscriber devices. This mechanism is added to support the role of various sensors in IoT. For this purpose, MQTT is a publish/subscribe based lightweight messaging protocol³, which is implemented in FogNetSim++ for communication among devices and broker nodes.

1) BROKER MODULE

As discussed above, the broker node is responsible to provide resources on request. In FogNetSim++, we have categorized

³<http://mqtt.org/tag/standard>

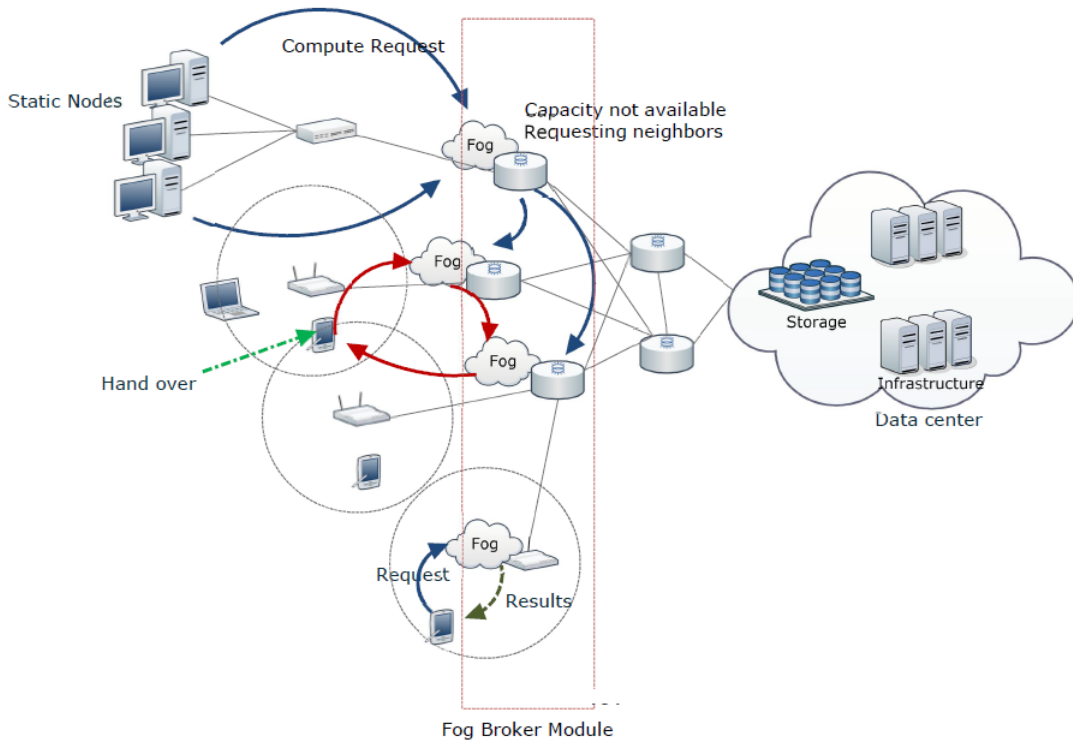


FIGURE 5. FogNetSim++ working model.

the nodes into three types i.e., static, mobility-based, and Wireless Access Based (WAB). The static nodes are the fog computing servers, placed at the gateway to provide computing on request. The mobility-based are the actual mobile nodes that can request computation from fog nodes. The WAB are the access points that can be used to serve the connected nodes. The internal modules of the broker node are shown in Fig 6. The WAB is designed on the inspiration of Cisco Edge⁴ series routers which provide the facility of computing along with the routing. The other features of the broker node are to register/manage publishers, subscribers, schedule fog nodes for requested devices, resource management, optimal utilization of resources among the number of requested devices, manage handovers, provide communication link with data centers, and reliable data delivery. The execution of broker and fog node is listed in Algorithm 1 and Algorithm 2. The simulation parameters of broker node are listed in Table 3.

2) END NODE DEVICES

Another core module of FogNetSim++ is mobile devices that can play an important role in fog computing. Here, we categorize devices as – sensor nodes and user nodes. A sensor nodes act as data generator, and send the generated data to broker or other devices after regular interval of time. Whereas, a user node can generate or receive data. A user

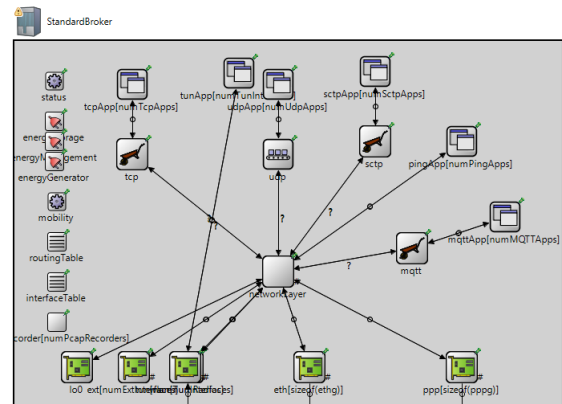


FIGURE 6. Internal modular view of Broker Node.

node can also act as a sensor. Moreover, these nodes are further arranged into two types: (a) wired and (b) wireless nodes. Nodes can be static or mobile, both versions are supported in FogNetSim++. Devices can register to publish data. The data is pushed to a broker node. In the proposed simulator, all the devices are IP enabled, thus can acquire IP addresses through DHCP. The devices support MQTT protocol along with other application protocols such as HTTP, TCP, FTP, SMTP, SNMP, and UDP. Moreover, the selection of a protocol for communication is done at design time. Due to the flexible architecture of FogNetSim++, researchers can easily incorporate other protocols through well-defined classes.

⁴www.cisco.com

Algorithm 1 Algorithm – Broker Node (B)

```

1: List fogNodes[]  $\leftarrow$   $FD_M$ 
2: List devices[]  $\leftarrow$   $d_P$ 
3: Queue taskQueue[]  $\leftarrow$  nil
4: Timer timer  $\leftarrow$  0
5:
6: while ( true ) do
7:   if MessageRecInWaiting then
8:      $Msg \leftarrow$  Message.Received
9:     if Msg.Type == Result then
10:      Forward Msg to  $d_i$ 
11:     else if Msg.Type == FN. $\Phi_i$  then
12:      Update  $FN_i.\Phi_i \leftarrow \Phi_i$ 
13:     else if Msg.Type ==  $d_i$ .position then
14:      Update All  $d_i$ .Position for Handoff
15:     else if Msg.Type == Servicereq then
16:      taskQueue  $\leftarrow$  Msg
17:     end if
18:   end if
19:
20:   if taskQueue  $\neq$  Empty then
21:      $Msg_{req} \leftarrow$  taskQueuepop
22:     if  $FN_i.workload < FN_i.\Phi_i$  then
23:       Forward  $Msg_{req}$  to  $FN_i$ 
24:     else
25:       boolflag  $\leftarrow$  true
26:       for  $i=0 \dots M$  do
27:         if  $FN_i.workload > FN_i.\Phi_i$  then
28:            $Msg_{req} \leftarrow$  taskQueuepop
29:           Forward  $Msg_{req}$  to  $i$ 
30:           flag  $\leftarrow$  true
31:         end if
32:       end for
33:       if flag then
34:         Starttimer  $\leftarrow$   $\Delta_T$ 
35:       end if
36:     end if
37:   end if
38:
39:   if timerexpire then
40:     for  $i=0 \dots M$  do
41:       if  $FN_i.workload > FN_i.\Phi_i$  then
42:          $Msg_{req} \leftarrow$  taskQueuepop
43:         Forward  $Msg_{req}$  to  $i$ 
44:       end if
45:     end for
46:   end if
47: end while

```

The end devices are mainly used to generate or consume data. A device uses MQTT messages for communication with a broker node. The FogNetSim++ supports heterogeneous devices; every device can have a different feature such as data rate, functionality and etc.

Algorithm 2 Algorithm – Fog Node (FN)

```

1: Timer timer  $\leftarrow$  0
2: Queue taskQueue[]  $\leftarrow$  nil
3: while ( true ) do
4:   if MessageRecInWaiting then
5:      $Msg \leftarrow$  Message.Received
6:     taskQueue  $\leftarrow$  Msg
7:   end if
8:   if taskQueue  $\neq$  Empty then
9:      $Msg_{req} \leftarrow$  taskQueuepop
10:    Outcome  $\leftarrow$  Execute $Msg_{req}$ 
11:    Send(Outcome, B)
12:   end if
13:   if timer expired then
14:     Send( $\phi_i$ , B)
15:     timer  $\leftarrow$  reset
16:   end if
17: end while

```

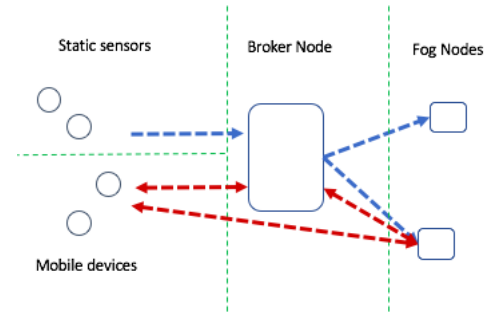


FIGURE 7. Case study - simulated topology.

Mobility Models – Mobility is another very important feature of end devices. In FogNetSim++ wireless and wired devices are supported. The mobility can play an important role, it can create new challenges for researchers in the form of optimum use of resources and resources' handover. Normally mobility models are classified into two categories i.e., entity and group model. In the entity model, the movement of each node is independent of other nodes. The most commonly used entity models are random waypoints, Gauss Markov, and City section. In the group model, the movement of one node is dependent on the movement of other nodes. The most commonly used group models are Column Mobility, Nomadic Community, and Reference Point Group. The FogNetSim++ supports only entity models. The already available models are a random waypoint, Mass mobility, Gauss Markov, Chaing mobility, CircleMobility, LinearMobility and vehicle mobility.

IV. TESTING AND PERFORMANCE EVALUATION

FogNetSim++ offers a comprehensive platform to simulate diverse fog applications. It also helps to understand the basic concept of fog computing. The FogNetSim++ provides a rich network configuration managed through a network

TABLE 3. Simulation parameters – broker.

Parameter	Description
numMQTTApps	The parameter indicates that the number of MQTT applications at each broker, it can be any integer number between 1– n
hasMQTT	a one bit field to indicate that the protocol being used is MQTT
numTcpApps	The parameter indicates that the number of TCP applications concurrently executing on broker node – default(0)
numUdpApps	The parameter indicates that the number of UDP applications concurrently executing on broker node – default(0)
numSctpApps	The parameter indicates that the number of Sctp applications concurrently executing on broker node – default(0)
numPingApps	The parameter indicates that the number of Ping applications concurrently executing on broker node – default(0)

TABLE 4. Machine specification.

Parameters	Value
CPU	4
Core(s) per socket	2
Thread(s) per core	4
CPU MHz	2390
Memory	8 Gb

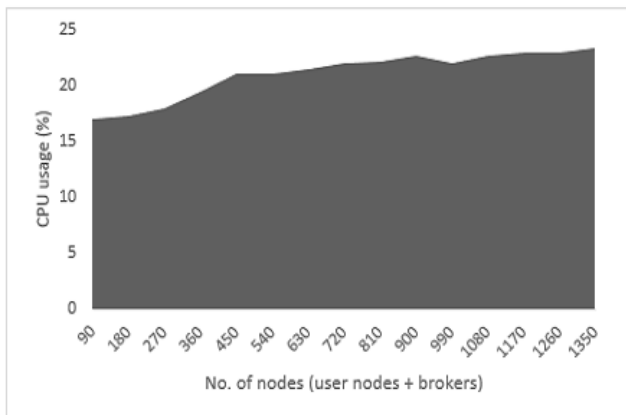


FIGURE 8. FogNetSim++ CPU usage with respect to number of nodes.

module. It allows the simulation of a realistic network environment that opens new challenges for researchers such as resources utilization, data error rate, and handoffs. All the modules are configured through a configuration file with extension *ini*. A user can set values of different parameters, such as, number of brokers, fog nodes, end devices, data rate, channel noise, and mobility models for every individual or group of nodes.

Case Study – To give the basic understanding of the framework and to evaluate the performance of the simulator, a communication network scenario is simulated based on smart traffic management system. The smart traffic management is an ideal case for fog nodes, where a number of static sensors are deployed to gather traffic information. Moreover, pedestrians can also share the surrounding conditions through their smart devices. The pedestrians are mobile, can freely move in any direction. These sensors generate data

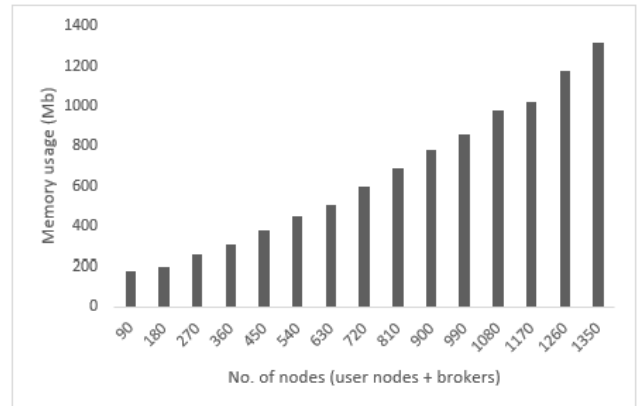


FIGURE 9. FogNetSim++ memory usage with respect to the number of nodes.

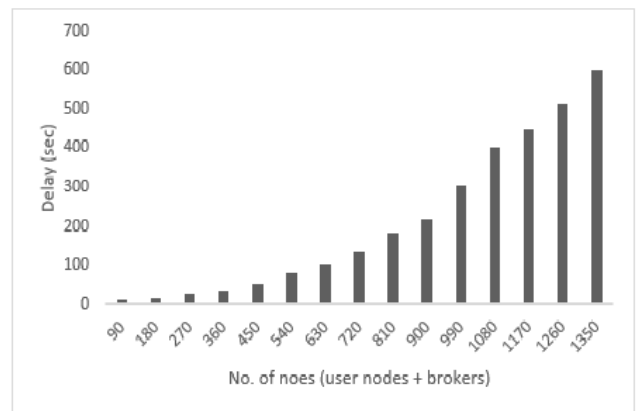


FIGURE 10. Delay in constructing enriched GUI .

at discrete points. The data is processed at the fog node for quick response, to identify any rules violation, such as over speeding, or any unusual situation. Moreover, there are law enforcement agencies which have subscribed to these sensors depending upon their geographical location. The fog nodes are used to identify the violations based on received data and disseminate the results to the subscribers. Apart from this, the law enforcement agencies can also generate a request for computing resources to perform their search or predictions. Here, we have analyzed the network based on the above mentioned scenario using a FogNetSim++. The simulated topology is shown in Fig. 7. The sensor generates data after

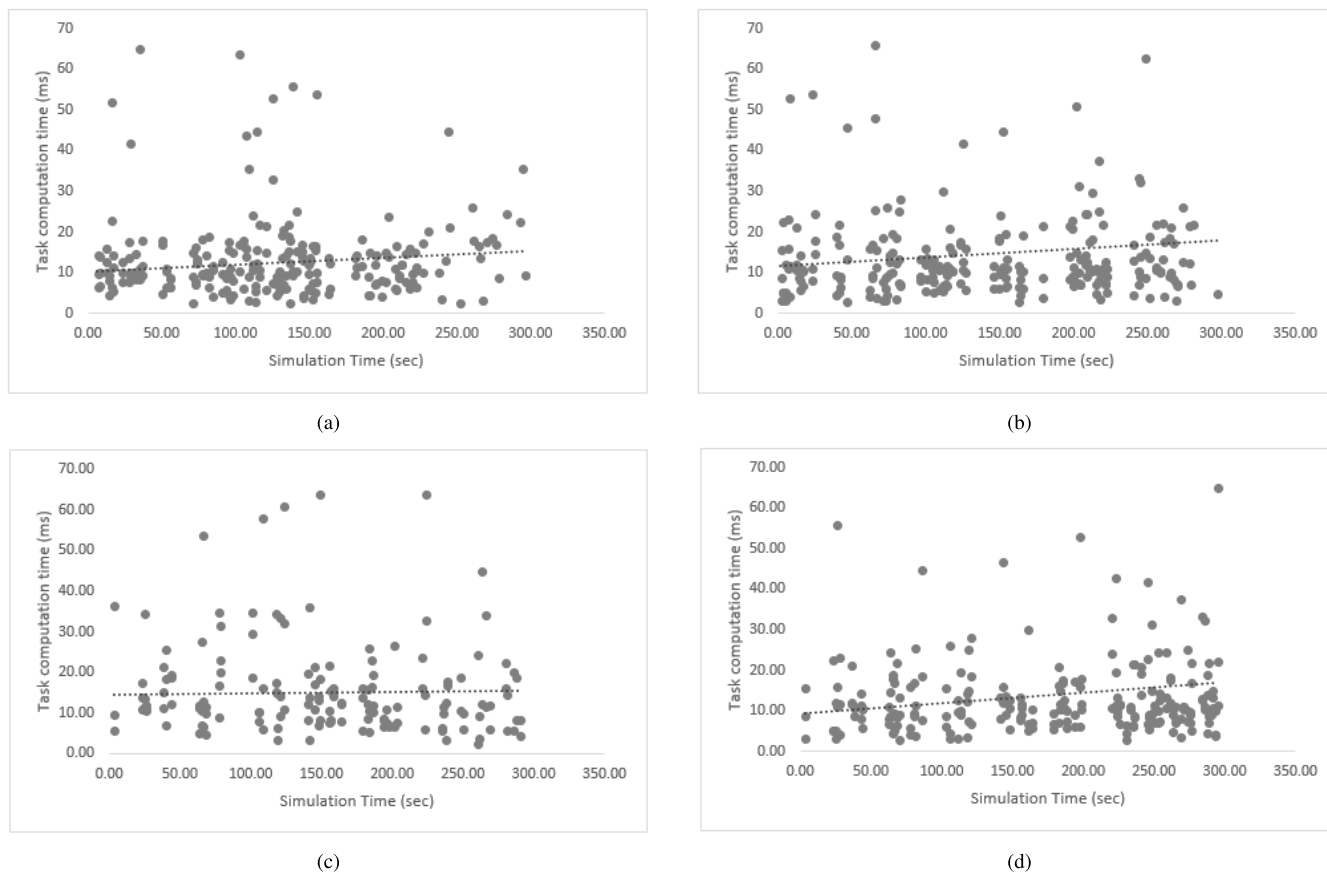


FIGURE 11. Average execution time in different task size categories. (a) Small-sized tasks. (b) Medium-sized tasks. (c) Large-sized tasks. (d) Random-sized tasks.

regular intervals of time. The broker node receives the data and forward to the fog node for processing and results are disseminated to the subscribed devices. Moreover, the devices can send the compute capacity request to the broker node. The broker assigns resources based on the scheduling policy and after execution, sends back the result to the requested device. During the task execution, the broker also tracks the requested or subscribed devices for hand-off operations. The scalability of proposed FogNetSim++ is measured in terms of memory and CPU usage. As the IoT comprises of a large number of devices, therefore, it is important to benchmark the proposed FogNetSim++ in terms of memory and CPU usage. Moreover, the other network parameters that are benchmarked here are the delay, latency, and error rate. The delay is also measured with respect to the execution task. The task is categorized as large (1500 MIPS), medium (900 MIPS), small (200 MIPS), and random (a random number between 200 and 1500 MIPS). Usually, different functionality requires a different amount of computing capacity. For example, in a traffic management system, identifying objects from a snapshot, or predict the movement of group can take a different amount of execution time. The system specifications are mentioned in Table 4. The parameters used in a simulation are listed in Table 5.

TABLE 5. Simulation parameters.

Sr.	Parameters	Value
1	Broker(s)	can support upto 400
2	Wireless sensors	10-400
3	Wireless Mobility	MassMobility
4	AccessPoint Broker(s)	10-400
5	Wireless Access Point(s)	10-400
6	Device(s)	60-880
7	Cloud Data-center(s)	1-4
8	Devices 1-100 Mobility	Circular Mobility
9	Devices 101-300 Mobility	Vehicle Mobility
10	Devices 301-600 Mobility	Mass Mobility
11	Devices 601-880 Mobility	Linear Mobility
12	Fog Nodes 1-1200 mqttApps	1 at each
13	Brokers Topic Name(s)	Sensing data
14	Broker-Broker Link	10 Gbps

The proposed FogNetSim++ is evaluated by varying the number of devices. The Fig. 8 shows that the CPU usage with respect to the total number of modules shows the increasing

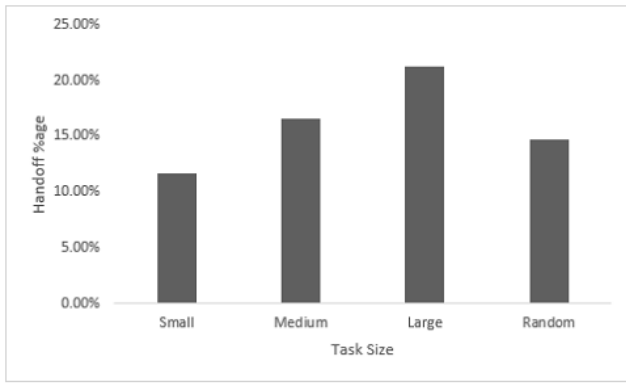


FIGURE 12. Handoff performed w.r.t task size.

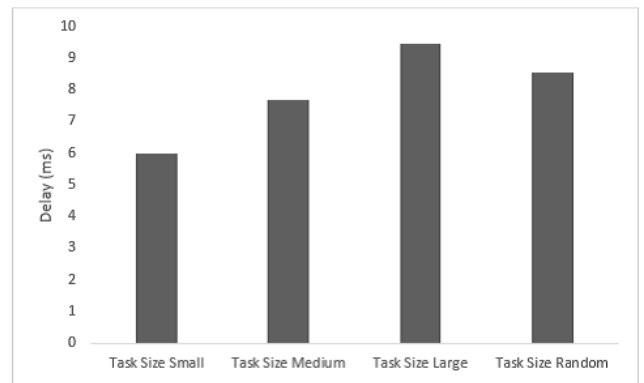


FIGURE 14. Total Delay based on the compute capacity requirement.

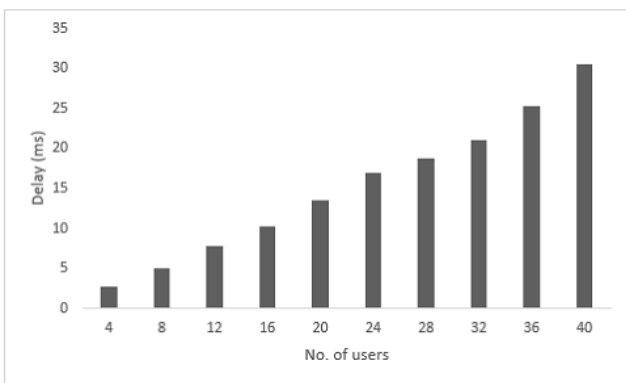


FIGURE 13. Average delay w.r.t users.

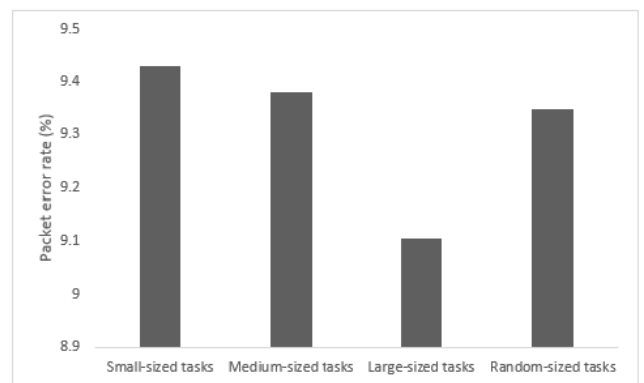


FIGURE 15. Average wireless error rate reported during the execution.

trend; however, for around 1320 nodes, the CPU usage is below 25%. The number of nodes mentioned in the figure includes both devices, fog nodes, and the broker node.

Similarly, with the increase in a number of nodes, the memory consumption also shows an increasing trend. However, for 1320 nodes around only 15% of memory is used (see Fig. 9). Thus, this clearly shows that the proposed simulator can easily support the execution of thousands of nodes on a typical computing machine. This is primarily due to the reason that each node maintains a very less node-specific data. In the simulator design, the common data is shared among all the nodes. The FogNetSim++ is designed on the top of OMNeT++, that provides a rich graphical user interface. Therefore, increasing the number of nodes usually requires a significant amount of time to populate the GUI. So, the delay in construction of GUI interface is also an important parameter to observe. Fig. 10 shows the delay in constructing the graphical user interface (GUI). At 1320 nodes the maximum delay is around 600 seconds. This is due to the rich GUI that allows users to inspect every module and parameters through GUI. However, the proposed simulator can also run in command line mode to avoid such delays.

The Fig. 11 shows the average task computation time for small, medium, large and random sized tasks. The dotted line represents the average time at every simulation interval.

Fig. 12 shows the percentage of hand-offs performed during the evaluation. However, it can be seen that with an increased task length, there are greater chances for a hand-off operation. Moreover, device location and mobility model also played an important role in hand-offs. Thus, the broker node maintains the location of the requested node, so that after the task completion, results can be dispatched accordingly.

The end-to-end delay increases with the increase in the number of user nodes as shown in Fig 13. Here we define end-to-end delay as the time taken by a request from the device to the broker node. In a typical scenario of large number of devices generating requests at regular intervals, it results in more delay as depicted in the figure.

In the experimental setup, the results are gathered based on different task computation requirements. The tasks are categorized as small, medium, large and mix. Fig. 14 shows the average task completion delay in each of the task category. The figure clearly shows that the large task computation requirement takes a significant amount of time in completion. Therefore, small execution tasks can enhance the system performance and reduce the waiting time at fog nodes. However, Fig. 15 shows the average error rate reported during execution. The devices communicate using the wireless channel. However, the rate varies with device placement, congestion of the network, and the mobility model used for a node.

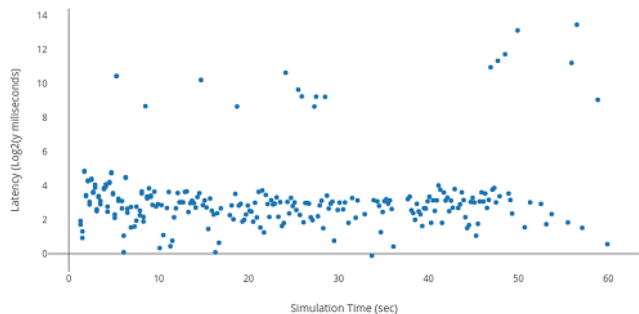


FIGURE 16. Average latency.

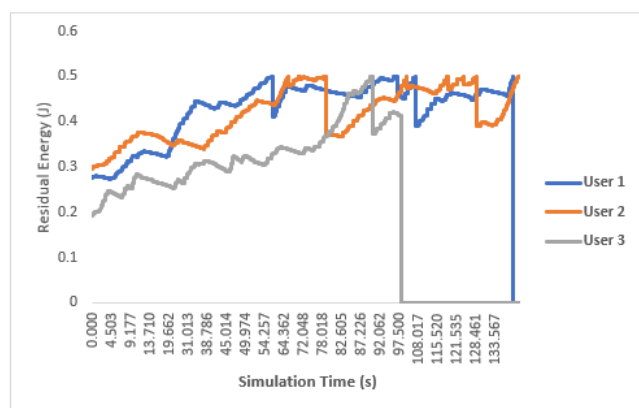


FIGURE 17. Residual Energy Vector over simulation time.

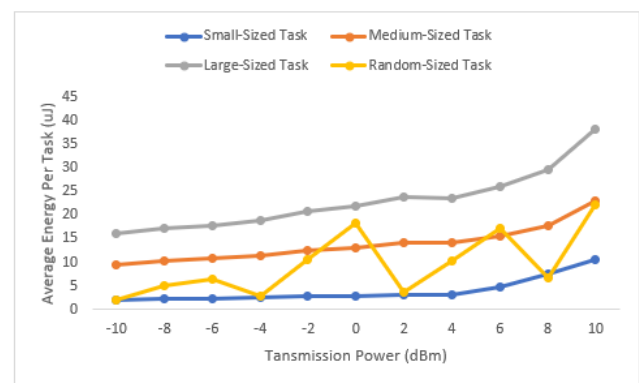


FIGURE 18. Average Energy Consumption vs Transmission power.

The average latency at discrete points is shown in Fig. 16. Here, the x-axis represents the simulation time and y-axis shows the latency. Fig. 17 shows the residual energy of three mobile users. The residual energy is defined as the remaining energy at a mobile node after communication with broker/fog node. The FogNetSim++ provides an extensive way to measure the energy of every mobile node that participated in the simulation. The node no longer becomes the part of the network when its residual energy reaches zero.

Fig. 18 shows the energy consumption with respect to transmission power. In FogNetSim++, the mobile nodes can dynamically change their transmission power

to communicate with broker/fog node. On the other hand, changing the transmission power can directly affect the life of battery operated devices. With large transmission power, the device can last for the shorter period of time.

V. CONCLUSIONS AND FUTURE DIRECTIONS

The inception of fog and edge computing provides new opportunities for delay-sensitive applications. The provision of computing at the network edge, closer to the end user, reduces the traditional network communication delays compared to the cloud-based execution model. In this paper, we have proposed a fog network simulator. In particular, FogNetSim++ provides a framework for researchers to investigate their own resource management techniques, perform resource handover, adopt new algorithms with the complete network stack. All the existing simulators do not completely provide the network characteristics, mobility features, and resource management modules. FogNetSim++ provides the diverse network features and supports a large number of mobility models. Thus, our objective is to facilitate researchers for the rapid development and testing of new resource management algorithms. In this paper, we have benchmarked the FogNetSim++ in terms of network usage and its behavior with respect to increase in traffic. The performance evaluations illustrate the effectiveness of the proposed framework. In future, we would like to extend the FogNetSim++ to include the VM migration among fog nodes. Moreover, interoperability among fog federations is another interesting area that need further exploration.

ACKNOWLEDGMENTS

The work of S. U. Khan is based upon work supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

- [1] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.
- [2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Computing*, 2012, pp. 13–16.
- [3] P. Bellavista, L. Foschini, and D. Scotece, "Converging mobile edge computing, fog computing, and IoT quality requirements," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2017, pp. 313–320.
- [4] S. N. Han *et al.*, "DPWSim: A simulation toolkit for IoT applications using devices profile for Web services," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 544–547.
- [5] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards simulating the Internet of Things," in *Proc. 28th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, May 2014, pp. 444–448.
- [6] 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.
- [7] A. M. Khan, L. Navarro, L. Sharifi, and L. Veiga, "Clouds of small things: Provisioning infrastructure-as-a-service from within community networks," in *Proc. IEEE 9th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2013, pp. 16–21.

- [8] T. Pflanzner, A. Kertész, B. Spinnewyn, and S. Latré, "MobIoTSim: Towards a mobile IoT device simulator," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud Workshops (FiCloudW)*, 2016, pp. 21–27.
- [9] C. Fiandrino et al., "CrowdSenSim: A simulation platform for mobile crowdsensing in realistic urban environments," *IEEE Access*, vol. 5, pp. 3490–3503, 2017.
- [10] *SimpleIoT Simulator*. Accessed: Aug. 10, 2018. [Online]. Available: <http://www.smplsft.com/SimpleIoT Simulator.html>
- [11] *IBM Bluemix Platform*. Accessed: Aug. 10, 2018. [Online]. Available: <https://console.ng.bluemix.net>
- [12] *Parse—The complete Application Stack*. Accessed: Aug. 10, 2018. [Online]. Available: <https://parse.com/products/iot>
- [13] *Google Cloud Platform*. Accessed: Aug. 10, 2018. [Online]. Available: <https://cloud.google.com/solutions/iot/>
- [14] H. Gupta, A. V. 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, 2017.
- [15] C. Adjih et al., "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464.
- [16] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GloTS)*, Jun. 2017, pp. 1–6.
- [17] *Contiki Cooja*. Accessed: Aug. 10, 2018. [Online]. Available: <http://www.contiki-os.org/start.html>
- [18] 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.
- [19] J. Byrne et al., "RECAP simulator: Simulation of cloud/edge/fog computing scenarios," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2017, pp. 4568–4569.
- [20] 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.
- [21] 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.
- [22] K. Hong, D. J. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the Internet of Things," in *Proc. MCC@SIGCOMM*, 2013, pp. 15–20.
- [23] S. Conti, G. Faraci, R. Nicolosi, S. A. Rizzo, and G. Schembra, "Battery management in a green fog-computing node: A reinforcement-learning approach," *IEEE Access*, vol. 5, pp. 21126–21138, 2017.
- [24] H. A. M. Name, F. O. Oladipo, and E. Ariwa, "User mobility and resource scheduling and management in fog computing to support IoT devices," in *Proc. 7th Int. Conf. Innov. Comput. Technol. (INTECH)*, Aug. 2017, pp. 191–196.
- [25] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.
- [26] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed Petri nets," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1216–1228, Oct. 2017.
- [27] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A middleware for mobile edge computing," *IEEE Cloud Comput.*, vol. 4, no. 4, pp. 26–37, Jul. 2017.
- [28] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.
- [29] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.
- [30] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.
- [31] X. Gong, L. Guo, G. Shen, and G. Tian, "Virtual network embedding for collaborative edge computing in optical-wireless networks," *J. Lightw. Technol.*, vol. 35, no. 18, pp. 3980–3990, Sep. 15, 2017.
- [32] X. Xu, J. Liu, and X. Tao, "Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation," *IEEE Access*, vol. 5, pp. 16406–16415, 2017.
- [33] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.
- [34] B. Tang et al., "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2140–2150, Oct. 2017.
- [35] T. Li, Y. Liu, L. Gao, and A. Liu, "A cooperative-based model for smart-sensing tasks in fog computing," *IEEE Access*, vol. 5, pp. 21296–21311, 2017.
- [36] M. Mukherjee et al., "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.
- [37] P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, and X. Yao, "Security and privacy preservation scheme of face identification and resolution framework using fog computing in Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1143–1155, Oct. 2017.
- [38] Q. Wang, D. Chen, N. Zhang, Z. Ding, and Z. Qin, "PCP: A privacy-preserving content-based publish-subscribe scheme with differential privacy in fog computing," *IEEE Access*, vol. 5, pp. 17962–17974, 2017.
- [39] B. Ngo and H. Lee, "Analysis of a pre-emptive priority M/M/c model with two types of customers and restriction," *Electron. Lett.*, vol. 26, no. 15, pp. 1190–1192, Jul. 1990.
- [40] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.



TARIQ QAYYUM received the B.S.C.S. degree from the Islamia University of Bahawalpur, Pakistan. He is currently pursuing the M.S.I.T. degree from the National University of Sciences and Technology, Islamabad, Pakistan. His research interests include cloud computing, fog computing, IoT, and distributed systems.



ASAD WAQAR MALIK received the Ph.D. degree in computer software engineering from NUST, Pakistan. He is currently an Assistant Professor with the School of Electrical Engineering and Computer Science, NUST. His research interests include parallel and distributed simulation, cloud computing, and large-scale networks.



MUAZZAM A. KHAN KHATTAK (SM'15) received the master's degree in mobile ad-hoc networks from IIUI and the Ph.D. degree in computer sciences as sandwich program from IIUI and the University of Missouri Kansas City (UMKC), USA, in 2011. He completed his first Post doc at the University of Ulm, Germany, in 2013, and the second Post doc from the University of Missouri, KC, USA, in 2016. He was with the Networking and Multimedia Lab, UMKC, USA, as a Research Fellow. In 2011, he joined the CS Department, Abdul Wali Khan University Mardan, as an Assistant Professor/Chair. Later, he joined the National University of Sciences and Technology as an Assistant Professor in 2013. He is currently a Tenured Associate Professor/Associate Dean at the Department of Computing, SEECS, National University of Sciences and Technology, Islamabad, Pakistan. His research interests include wireless networks sensor, body area networks, image processing, image compression, image encryption, and data network security.



OSMAN KHALID received the master's degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, and the Ph.D. degree in electrical and computer engineering from North Dakota State University, USA. He is currently an Assistant Professor with the Department of Computer Sciences, COMSATS University Islamabad, Pakistan. His areas of interests include fog computing, disaster response systems, recommender systems, and wireless routing protocols.



SAMEE U. KHAN received the Ph.D. degree from the University of Texas, Arlington, TX, USA, in 2007. He is currently the Lead Program Director (Cluster Lead) for the Computer Systems Research at the National Science Foundation. He is also a Faculty Member at North Dakota State University, Fargo, ND, USA. His research interests include optimization, robustness, and security of computer systems. His work has appeared in over 400 publications. He is an ACM Distinguished Speaker, an IEEE Distinguished Lecturer, and a fellow of the Institution of Engineering and Technology (formerly IEE) and the British Computer Society. He is on the editorial boards of leading journals, such as *ACM Computing Surveys*, *IEEE ACCESS*, *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*, *IET Wireless Sensor Systems*, *IET Cyber-Physical Systems*, and *IEEE IT Professional*.

• • •