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

By
Tariq Qayyum
00000170749

Supervisor
Dr. Asad Waqar Malik
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(October 2018)

Approval

It is certified that the contents and form of the thesis entitled "**FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment**" submitted by **Tariq Qayyum** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Asad Waqar Malik**

Signature: ____________________
Date: ____________________

Committee Member 1: **Dr. Anis-ur-Rehman**

Signature: ____________________
Date: ____________________

Committee Member 2: **Dr. Muazzam Ali Khan**

Signature: ____________________
Date: ____________________

Committee Member 3: **Dr. Arsalan Ahmad**

Signature: ____________________
Date: ____________________

Abstract

Fog computing is an emerging technology that extends the cloud and brings the resources closer to the end devices to achieve better performance in latency-sensitive application. Fog computing still lacks the standardization in terms of simulation environment and architecture. Several fog simulators has been developed and proposed previously. Most of the existing fog simulators ignore core network characteristics like error rate, packet loss, bandwidth etc. 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, 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.

[1]available at https://fognetsimpp.com

Dedication

To my parents,
Dr. Asad Waqar Malik,
Dr. Muazzam Ali Khan,
Dr. Anis-ur-Rehman, and
Dr. Arsalan Ahmad,
Without whom this success would not be possible.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Tariq Qayyum**

Signature: ________________

Acknowledgment

....
Tariq Qayyum

Table of Contents

List of Figures

List of Tables

List of Symbols

Abbreviations

MQTT	Message Queuing Telemetry Transport
ARP	Address Resolution Protocol
CPU	Central Processing Unit
GPU	Graphical Processing Unit
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
MAC	Media Access Control
ONF	Open Networking Foundation
OS	Operating System
RAM	Random Access Memory
RTO	Retransmittion Timeout
RTT	Round Trip Time
SDN	Software Defined Networking
SSH	Secure Shell
TCP	Transmission Control Protocol
VM	Virtual Machine

Nomenclature

σ	Standard Deviation
RTT_{avg}	RTT of Data Plane
RTT_{FE}	RTT of Flow Entry Installation
$RTT_{reinstall}$	RTT of Flow Replacement Policy
RTT_{test}	RTT of Test Pings
T_{hard}	Hard Timeout
T_{sleep}	Sleep Time
T_{step}	Step Size
T_{soft}	Soft Timeout

Chapter 1

Introduction

An incredible paradigm shift is observed in Information and Communication Technology (ICT) from limited and isolated computing environments to the prevalent and passive computing over the years. Due to the progression in the manufacturing and telecommunication industry, powerful smart-devices are developed that are platform independent which can ubiquitously establish connection to the network. In recent years, 5G evolved such that the whole network world looking at it as the future technology. The 5G technology has several features which include the heterogeneously wireless network connectivity and many applications can have benefits from it in term of performance enhancement like response time, latency, and energy consumption. As the technology grows, different devices joining themselves to become the part of a grid and generate huge amount of data to be sent to cloud for different operations like storage, and processing. Around 8.4 billion different devices will be the part of Internet by start of year 2018 Gartner[1] (2017). A significant amount of intelligent processing is required on the raw data that is generated from sensors, to send it to cloud for further processing. This processing is needed to reduce the bandwidth usage, and in achieving better latency. This is the reason that bringing resources closer to the end devices is rising. Among many merging Edge computing paradigms, Fog is a most prominent and efficient when IoT devices are involved.

Cisco [5] first introduced the term Fog. They explicitly described that for is not an alternative to cloud, but it extends the cloud computing closer to the end users (edge) [1]. If compared to the cloud data centers, virtualized computing environment is deployed at edge, closer to end users [1]. Fog is an additional layer between end users and cloud. Both Fog and Cloud provides almost similar services, but Fog has advantage that it provides these services

[1]https://www.gartner.com/newsroom/id/3598917

to facilitate a specific region. The primary purpose of Fog is to improve the latency for delay-sensitive as cloud is far away from the end users and take more time for data to be sent. Many services are provided by the Fog to different IoT applications and other networking devices like Road Side /units, access points, and routers etc. The management of basic network operations like fault tolerance, reliability, and scalability is easy when number of fog nodes increases. Another primary advantage of applying fog node in between cloud and end devices is that, it reduces the bandwidth between cloud and end devices.

Fog computing evolving but it faces several challenges like fog nodes architecture, heterogenous device management, privacy, security, and device mobility. Another challenge is the interoperate-ability between two heterogenous devices. It is necessary to process data coming from several devices before sending it to the cloud or taking any action. Normally fog paradigm consists of a few numbers of fog nodes that provide different services. This is the reason that efficient algorithms are required to process data in efferent and in timely manner. A general scenario is explained in Fig 1, where efficiency is improved by installing a fog layer in between end devices and cloud. Just like in cloud, better management of resources in the data center to achieve maximum efficiency and billing is an important feature that needs to be implemented in the fog. A flexible environment is provided by the fog and with flexibility the challenge of managing the decentralized resources arises. Location is flexible for a fog node as it can be installed anywhere in the network. The main purpose of designing the fog is to improve the performance of latency-sensitive applications, and the nest location for fog is the Smart Gateways [2].

Contribution – In this thesis a fog simulator is proposed which we call FogNetSim++. This toolkit provides the facility to simulate heterogenous devices along with many features. Another important feature of FogNetSim++ is that it provides a very efficient mechanism to perform handover by which the location of a devices can easily be tracked. Thus, the result of the task is returned to the sources by routing through different geographically located devices. FogNetSim++ is designed in such a flexible manner that researchers can easily incorporated and merge their own algorithms. These algorithms include task scheduling algorithms, resource allocation algorithms, and mobility related classes. No such a rich tool exists in the past that provides these features. On the other hand, FogNetSim++ facilitate users to evaluate, simulate fog environment with various realistic network characteristics. Both static and dynamic devices can be simulated using FogNetSim++. It supports several protocols like advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), and Message Queue Telemetry Transport protocol (MQTT). The mobility models in FogNetSim++ includes, StaticGridMobility, CircleMobility, TractorMobility, RectangleMobility, TractorMobility, and StationaryMobility. FogNetSim++ provides energy modules by which user can simulate the energy usage of nodes, and residual energy of the nodes. A central broker manages all fog nodes and FogNetSim++ also have many scheduling algorithms.

1.1 Motivation

The existing Fog computing simulators lacking some of the main features and modules like, mobility while working with wireless nodes and if they provide mobility features they are lacking the safe handover and energy modules as well. The gap in the existing research motivated us to work on the mobility, handover, and energy modules simulation for Fog computing.

1.2 Problem Defination

There is a need of tookit to simulate distributed Fog computing environment by considering the mobility, handover, realistic network characteristics, and energy modules.

1.3 Objectives and Research Goals

The main objectives of the research is are given as

- Implementing modules in the INET, the core module of Omnet++ tool
- Implementation of MQTT protocol for Fog computing IoT devices in omnet++
- Mobility models
- Save Handover
- Consideration and Implementation of realistic network characteristics
- Energy modules

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 covers the overview of Fog computing environment, how it works, and which are the most commonly used protocols. In Chapter 3 the available simulators about Cloud and Fog are discussed. In Chapter 4 the System model and the design of FogNetSim++ is discussed. In Chapter **??** the sample experimental setup is explained and the results are discussed. Finally, we conclude this thesis in Chapter 6 and discussed the possible future enhancements about the FogNetSim++.

Chapter 2

Background Information

This chapter is composed of overview and all the background information about the thesis.

Network equipment and expensive devices combine to achieve fog computing. Simulation provides a pre-deployment testing before the deployment of real resources and expensive devices. No standard toolkit is available in the market to simulate fog computing specifically. However, several other network simulators can be used to simulate a few features of fog computing. This gap in the research motivate researchers to develop an extensive tool that can be used to simulate all possible features of fog.

However, many features have been ignored in the existing fog simulators that we discussed here. Many existing simulators focus on the homogenous devices. A central deice process data before sending this processed to the cloud. Java-based simulators exist that ignore the many important network characteristics like packet drop, or error rate, channel collision, and network congestion. The existing simulators have only limited feature of mobility and they ignore handover completely. Many simulators are not even open source, researchers and students cannot have benefit from them in a way they can with open source simulators. Also, researchers cannot implement their own algorithms for modification and testing. They also ignore cost implementation and energy modules.

The overall Internet devices involved in fog networking are very expensive. It is crucial to test the fog deployment using simulations and tools. However, there are no fog simulation tools available to support deployment and mobility models; thus, this makes it an open research challenge. FogNet-Sim++ is designed for rapid simulation development to test new algorithms for fog environment. The proposed simulator is flexibly designed to simulate three levels i). IoT, where various heterogeneous devices can connect and communicate ii). edge level, where edge servers can be placed to provide services and iii). cloud data center level, connected through a high-speed network. Using proposed simulator, researchers can test their algorithms in terms of efficiency, resource usage, network latency, and efficient allocation of resources.

One such application area is Mobile Edge Computing (MEC) that is specifically focused on mobile applications keeping in view the limitations of mobile devices in terms of storage and processing. Traditionally, devices communicate directly with cloud data centers for services. However, such architecture is not suitable for delay-sensitive applications, e.g., real-time video streaming, or smart healthcare. Therefore, a middleware is added between a mobile device and back-end cloud centers. This middleware layer is termed as network edge in which some of the resources from cloud are made available on the edge nodes. Therefore, MEC models allow users to host/store resources at the network edge that is close to the end devices. The MEC architecture is very promising for delay-sensitive applications as when the resources are deployed near to the end devices, it will take less time for a request/response to travel on the network [3]. For context-aware applications, sometimes it is not desirable to send all the raw information to the cloud. Therefore, edge nodes can be used to intelligently extract the only useful information that should be sent to the cloud data centers.

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 Gartner[1] (2017) report, around 8.4 billion devices will be connected to the internet by the end of 2017. Therefore, the raw data generated through various sensors require intelligent processing to reduce the bandwidth usage and improve the latency. For this purpose the edge computing devices are required that can handle such kind of computation. Many vendors are manufacturing devices for edge computing e.g., Cisco manufactured a series of edge devices Cisco Edge 300 and Cisco Edge 340. These devices provide the facility of computing when installed at the edge of the network. Moreover, any conventional device can also be used for this purpose. Edge computing can also be performed in a distributed manner, as discussed in [4]. The computing paradigm defined by Cisco is the extension to conventional cloud computing model to execute the geo-distributed applications [5]. The Cisco introduced the term Fog Computing (FC) to support latency-sensitive applications [1]. In typical terms, Fog is a cloud close to the end user. Thus, a Fog is an extension of a cloud that provides compute, storage, and networking. Similar to edge computing, the FC utilizes locally connected computing resources to reduce the transmission latency. However,

[1]https://www.gartner.com/newsroom/id/3598917

the cloud data centers are still useful for big data analysis.

Similarly, another cloudlet-based Edge computing is proposed by Satyanarayanan [6]. Cloudlets are decentralized Internet infrastructure that can be used by mobile devices to offload compute intensive tasks. Cloudlet alleviates the response latency and delay compared to cloud deployment. Cloudlet also supports virtualization; services deployed inside virtual machine can easily be migrated to/from cloudlet. Moreover, handoff is also supported in cloudlets [6].

The MEC and Fog are designed to assist devices in terms of performance. The Fog is more flexible compared to MEC [2]. In MEC, the edge locations are often limited and require hardware installation that also includes the setup cost and time. Once, the edge devices are deployed, it is difficult to re-deploy at different location with the dynamic burst of service requirements. Whereas,

To promote Fog and encourage developers to design Fog enabled applications, the OpenFog consortium was held Princeton Univ. in Nov 2015. The well known researchers/developers from Cisco, Microsoft, Dell, Intel and ARM were participated. The generic Fog framework is shared with developers that can support any application such as transportation, agriculture and etc. Thus, enabled the development of latency-sensitive applications.

In computing domain, the paradigm have evolved from parallel to distributed, grid to cloud, and edge to fog computing. The traditional cloud paradigm is based on various features such as on-demand scaling, pay-as-you-go, fault tolerance and etc. It also provide unlimited storage, region-wise replication of data, and security services. Due to the benefits, cloud computing is widely used. There are number of companies that are proving cloud services, few well known companies are Amazon Ec2, Google Cloud, Microsoft Azure and etc. The user can deploy services and store data inside the cloud. The services and data hosted inside the cloud can easily accessed from any device connected through the internet. Before using cloud services, the user typically sign a contract termed as Service Level Agreement (SLA). In SLA, user listed all the required resources and cloud providers are bound to maintain the SLA all the time [7]. In general terms, cloud services are categorized as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Whereas, there are other terms that are very commonly using with cloud are X-as-a-Service (XaaS) (means unit consumed - related to billing), Network-as-a-Service (NaaS), Storage-as-a-Service and etc [8] [9].

Other than the benefits of cloud computing, there are some limitations. One of the limitation is device-to-cloud connectivity that required Internet connection. Especially for large-scale data transfer and latency-sensitive applications, the cloud is definitely not a good option. Some of the latency-sensitive applications are smart grid, and streaming services. Moreover, in cloud, services are often deployed on different physical systems that communicate with each other. The deployment can separate node can further increase the delay due to intra-cloud traffic. Lastly, the cloud data centers are not available in every region, so this further reduces the option to deploy services or store data. Therefore, the concept of edge computing is introduced to minimize the latency and serve the contents from the closed possible location.

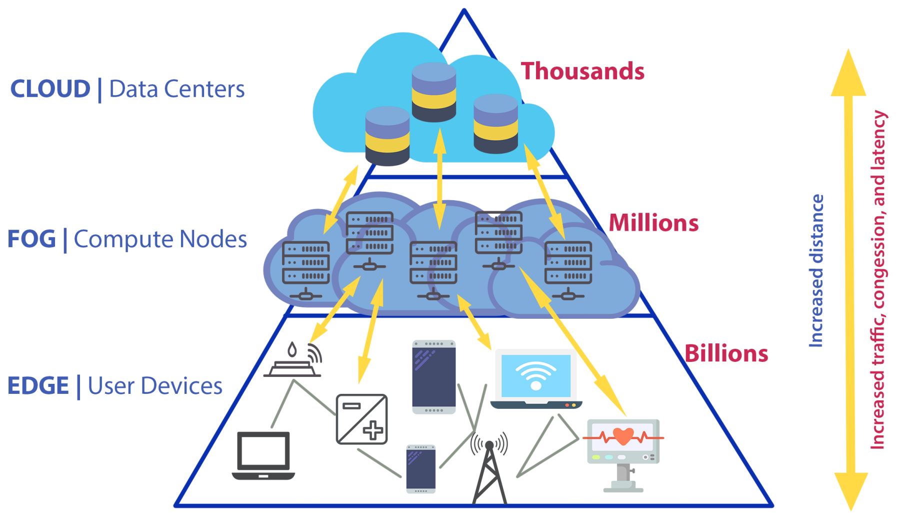

Figure 2.1: Computing Architecture Pyramid

Unlike cloud, Fog cannot be used for big data analysis; moreover, it is scalable compared to the cloud paradigm. However, the edge and fog are very closely related, sometime used interchangeably; the Fog has the capability of device management, data service, and communication that are missing in Edge computing [9]. The key features of Fog and Edge are listed in the table.

Edge computing a mechanism used for the optimization of cloud computing systems. In edge computing the data processing is performed at the edge of the network not at cloud resources. This process gives the benefit of minimizing the communication bandwidth required between data-centers and sensors by performing the computation at the edge of network and send only necessary information to the cloud/data-centers [3]. Sending all information to the cloud has been proved a less intelligent strategy because most

Table 2.1: Difference between Edge, Fog and Cloudlets [?]

Sr.	Attributes	Fog Computing	Edge Computing	Cloudlets
1	Node device	Routers, Switches, Access Points, Gateways	Servers running in base stations	Data Center in a box
2	Node location	Varying between End Devices and Cloud	Radio Network Controller/Macro Base Station	Local/Outdoor installation
3	Software architecture	Fog Abstraction Layer based	Mobile Orchestrator based	Cloudlet Agent based
4	Context awareness	Medium	High	Low
5	Proximity	One or Multiple Hops	One Hop	One Hop
6	Access Mechanisms	Bluetooth, Wi-Fi, Mobile Network	Bluetooth, Wi-Fi, Mobile Network	Bluetooth, Wi-Fi, Mobile Network
7	Inter node Communication	Supported	Partial	Partial
8	Data services	Yes	Yes	Yes
9	Application Hosting	Yes	Limited control	Yes
10	Real-time control	Yes	No	Yes
11	Virtualization	Can be supported	No	Yes

of the times we don't even need to store that much amount of raw data coming from sensors. As the technology growing, scientists are eager to utilize every possible information from almost every electronic device, which might be sensor data coming from a sensor of other information coming from an infrastructure device. If we start throwing that much amount of data to the clouds, the possible problem that can occur are bandwidth resource issues in network, storage and computing problems in the data-centers. The alternative to deal with situation is to compute some computation at the edge of the network instead of sending them blindly to the cloud that can save a sufficient amount of resources at clouds and network as well. The improvement in latency can help in the systems where control information is required in a very low delay like real time systems. For this purpose the edge computing devices are required that can handle such kind of computation. Many vendors are manufacturing devices for edge computing e.g. Cisco manufactured edge series devices Cisco Edge 300 and Cisco Edge 340. These devices provide the facility of computing when installed at the edge of network. Further, any conventional device can also be used for this purpose. Edge computing can also be performed in a distributed environment [4].

The overall Internet devices involved in fog networking are very expensive. It is crucial to test the fog deployment using simulations and tools. However, there are no fog simulation tools available to support deployment and mobility models; thus, this makes it an open research challenge. FogNetSim++ is designed for rapid simulation development to test new algorithms for fog environment. The proposed simulator is flexibly designed to simulate three levels i). IoT, where various heterogeneous devices can connect and communicate ii). edge level, where edge servers can be placed to provide services and iii). cloud data center level, connected through a high-speed network. Using proposed simulator, researchers can test their algorithms in terms of efficiency, resource usage, network latency, and efficient allocation of resources.

Table 2.2: The key features of Fog and Edge - not clear discuss

Features	Edge	Fog
Data services	Yes	Yes
Security	VPN partial point	Data protection, E2E, Hardware & Session level
Application hosting	Yes but limited	Yes
Real-time Control	No	Yes
Availability		
Awareness	Device aware but unaware of the entire domain	Intelligent, aware of the entire domain & device indp.
IoT vertical awareness	No vertical awareness	Supports multiple verticals
IoT vertical integration	No	Yes
Security scope	Limited to devices	End-to-End
Virtualization	Not designed for virtualization	Rich virtualization

Chapter 3

Literature Review

In this section commonly, used simulators are discussed and compared them with FogNetSim++.

3.0.1 Fog – IoT Simulators

DPWSim was proposed by Han et al. [10] to simulate IoT applications. They simulated event-driven, and service-oriented models. In SimIoT [11], Sotiriadis et al. extended the SimIC framework. Many communication mechanisms are provided for cloud data centers and IoT sensors are provided in SmIoT. Another simulation toolkit named EdgeCloudSim which was proposed to minimize the limitations in conventional cloud simulators so that to simulate the edge computing environments. CloudSim was extended to develop EdgeCloudSim [12].

A coordination technique was proposed by Khan et al. [13] to simulate a large number of IoT nodes and also supports things related to home automation. It was proposed as an extension to the CloudSim. IoTSim is proposed especially for big data processing on the data generated by IoT devices. The author explained toolkit by discussing a case study and discussed the results. A mobile based simulator MobIoTSim [14] for IoT devices was proposed to simulate mobile devices. Authors claimed that the main objective of the simulator is to simulate mobile devices without buying the real ones which is also the primary feature of any simulator, In MobIoTSim, users can understand and explore the deep working of connected devices.

In [15], F. Claudio et al. proposed a simulator for crowd source applications like smart cities and named it as CrowdSenSim. A street lightening environment was simulated to evaluate the performance and working of simulator. It can also be used at the places where data is collected in big amount and need processing and it is available free for the researchers to work on it as well ass they can enhance the features if they like to.

IoT sensors and devices are commonly simulated by SimpleIoTSimulator [16]. This tool supports several IoT related protocols like CoAP, and MQTT etc. the limitation observed in SimpleIoTSimulator is that it only support RedHat 64 bit. A user can install the simulator environment only using RedHad distribution. A PaaS enabled simulator for IoT devices is proposed by IBM and is known as IBM Bluemix [17]. A web-based environment is used to quickly install and simulate cloud applications and data is gathered from IoT sensors and devices using these applications. With IBM Bluemix hardware devices can be simulated from Intel, Texas Instrument, and ARM. MQTT protocol is used to send data to cloud. Another platform-as-a-service (PaaS) simulator Parse [18] is proposed by Facebook that supports IoT sensors and devices. A very easy environment is provided for application deployment. It supports several mobile devices. Google is also in the race and the proposed simulator is Google Cloud Platform where many devices can be virtualized and gather data from them. They named it Google AppEngine.

A fog-based simulator iFogSim was proposed by Harshit et al. [20]. It is developed in Java technology as an extension to CloudSim. Here scientists discussed the resource management techniques' impact in term of cost, congestion, and latency. It has a few limitations as well. First limitation is that it is java based and core network characteristics are not supported or ignored. Also, it is not even compatible with different java versions. And the documentation is not rich enough to understand and simulate environment from scratch using this tool. On the other hand, in our proposed simulator all tool like congestion, delay, packet drop, latency and many more are available and it allows a user to flexibility simulate and test Fog computing environment by varying these network characteristics.

Cisco has emerged as a big organization that provided a variety of network devices. A toolkit which supports Cisco based devices is proposed and is known as FIT [21]. It is an open source solution. Using FIT simulation is not performed in a controlled environment and therefore simulation cannot be repeated.

To simulate wireless networks, WSNet [22]– a discrete event simulation model is proposed to simulate IoT devices. IoT networks can also be simulated with it. It contains modules like energy, routing protocols, radio interfaces, and mobility. Disaster situations like fire, and earthquake can also be simulated with it.

3.0.2 Cloud Simulators

CloudSim is the most commonly used cloud simulator, designed in Java at University of Melbourne, Australia. It provides basic classes to define virtual machines, users, data centers, computational resources, and applications policies. Thus, facilitate with the generalized framework to cloud computing services [?] [?]. To overcome the deficiency of CloudSim, number of simulators have been designed as an extension to CloudSim. NetworkCloudSim is an extension of Cloudsim, designed to simulate applications like workflows, high performance applications, cloud data center and etc. This framework provides a structure to develop cloud data centers, and the simulate different policies. It can be used to simulate cloud data centers networking applications that involves communication among processes [?].

EMUSIM is another extension of CloudSim, designed to visualize the behavior services on cloud platforms [?]. It used open source software stack and has limitations in terms of scalability due to hardware and cannot manage large workloads. Cloud Analyst [?] provides the optimal resource scheduling among users based on geographical location. It provides a rich map based simulation framework for monitoring, load balancing, deployment of data centers, data flows and cloud cluster monitoring. The key features include its flexible configuration, high degree of control, and simulation of data center virtual machines, resource allocation policies, and internal communication[14].

GreenCloud is build using NS2 simulation framework. It is a collaborative project between University of Luxembourg and North Dakota State University. It is designed for packet level simulation and latest version also support virtual machine (VM) migration and consolidation techniques. iCanCloud is build on top of OMNeT++, designed for large cloud simulations. It also compute the cost of using compute resources. It provides support for Amazon EC2 and hypervisor that can be used to compare different policies.

GroudSim developed using Java, designed to support cloud and grid computing. It allow researchers to execute complex simulation scenarios including background workload generation and cost calculation. CloudNetSim++ is designed on the top of OMNeT++ and utilized the features of INET framework. It provides a comprehensive framework that allow users to define their own VM migration policies, and analyze usage cost.

A scheme was proposed by Canti et al. [30] to store energy in fog nodes so that complex operations can be performed. This scheme is helpful because power storage is a crucial part while we are working on IoT devices. IoT devices have a very low amount of residual energy and on the other hand radio uses energy in a very vast amount in sending and receiving data packets.

Table 3.1: Simulator Comparison

Simulators	**Prog. Language**	**Platform**	**Network Configuration**	**Open Source**	**Mobile Nodes**	**Customize Mobility Models**	**Scheduling Algorithms**	**Device Handover**	**Energy Module**
MobIoTSim [14]	Java	Linux	No	Yes	Yes	No	No	No	No
SimpleIoTSimulator [16]	Java	Unix	No	No	Yes	No	No	No	No
IBM BlueMax [17]	Java/Python	Cloud	No	No	Yes	No	No	No	No
Google IoT Sim [19]	NA	Cloud	No	No	Yes	No	No	No	No
iFogSim/MyiFogSim [20]	Java	All	No	Yes	Yes	No	Yes	No	No
Cooja [23]	C	Linux	No	Yes	Limited	No	No	No	No
FogTorch [24]	Java	All	No	Yes	No	No	No	No	No
RECAP simulator [25]	N/A	–	Limited	N/A	No	No	No	N/A	N/A
EmuFog [26]	Python	All	Yes	Yes	No	No	No	No	No
EdgeCloudSim [27]	Java	All	No	Yes	Yes	No	Yes	No	No
Edge-Fog cloud [28]	Python	All	No	Yes	Limited	No	No	No	No
Mobile Fog [29]	N/A	–	No	No	Yes	No	No	No	No
FogNetSim++	C++	All	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Haruna et al. [31] proposed a technique to efficiently allocate the resource in fog and it also supports handover in fog nodes. Handover is very important while working with wireless nodes to make sure no data loss.

Fog is deployed as the middle layer in between cloud and source data i.e. sensors and IoT devices that generate massive amount of data and send this data to fog for processing. The energy management becomes very crucial at this stage, hence and dynamic energy management technique was proposed by Jie Cu et al. [35] which dynamically manage energy and learn during the simulation to manage overall job mechanism.

Delay-constraint model is studied by Meng et al. [32] for computation task offloading to the fog and cloud servers. The primary purpose of the research to reduce the energy consumption while performing computation or during communication. Results showed that the overall energy consumption is reduced.

Compare to cloud, fog nodes contain only limited amount of resources, and hence typical algorithms used for task scheduling at clouds are not good enough for fog nodes because of rapid changes in user requirements. Lina Ni and team [33] worked hard on the problem and came up with a resource allocation algorithm for fog computing. Time and price were used to define the system. The actions are taken placed dynamically depending on the variables' current situation.

A geographically distributed framework was proposed by Ananthanarayanan et al. [36] for video analytics at large scale. This framework supports real time video analytics, and it uses gateway nodes to provide streamline performance. Ju Ren et al. [37] worked on the edge computing and formulated a scalable framework for performance evaluation of edge computing. This framework provide facility to smart home appliances and IoT devices, where data is collected from a number of different sensors, and is sent to nearby computing nodes. This computing node acts as the fog node in the network and hence the communication and number of messages sent can be reduced by pulling the intelligence closer to the devices generating data.

Bo Tang et al [41] contributed in the fog computing by introducing a distributed architecture for different servers, and infrastructures, and they claimed that they obtained very improved results. Li Ting et al. [42] worked on the fiber channel to improve the quality of data collection with their proposed cooperative framework. They reduced the delay by introducing the fog relay nodes. A framework for hybrid offloading data and computation tasks to the fog nodes is proposed in [43]. With this framework less energy is consumed in task offloading because it used adaptive offloading algorithms.

Xiaodong Xu et al. [39] proposed a video transmission model where a number of servers are used for caching of video data so that to prove delay less transmission of video streaming. Several well-known companies are using the technologies like NetFlix, and Youtube etc. They use this technique to cache data at the edge of the network and provide delay less service. A crowd sourced framework that works device-to-device is proposed by Chen et al. [40] for edge computing and especially for mobile using 5G technology. A device can communicate efficiently with another device in 5G technology. So, a computing is required for a very large number of devices at the edge. Another task sharing and resource allocation strategy is proposed by Sonmez et al [12].

Security is an important factor while working with IoT devices. Many researchers worked on the security aspects of the IoT devices. Among them, Mithun Mukherjee and his team proposed a security framework for IoT devices. PengFei Hu [45] proposed a framework using which security and privacy related complex computation can be performed at the fog node. They used this technique face detection at fog. Face detection needs a complex computation that we can perform at cloud and not at the devices because of less energy at devices, and it is performed at cloud, the latency is a very big overhead. So, here fog layer helps in reducing the unnecessary delay by performing such computations at fog nodes. In vehicular network, timely response is required because human life is at risk. Hence introducing fog layer at in vehicular networks help in reducing delay [1].

We discussed many simulators above but most of them don't facilitate the researchers to introduce and incorporate their own algorithms for task offloading, task scheduling, and mobility. Also, most of them are not open sourced and not available freely in the market. Many of them ignore basic network characteristics like, packet loss, bandwidth, and delay etc. Only a limited number of mobility models are present in some simulators and most of them ignore the mobility altogether. A table is constructed to compare detailed feature in Table 3.1.

Chapter 4

Methodology

The main objective of this research is to establish a new simulating toolkit, where researchers can simulate distributed Fog Computing environments. It provides a number of additional features that previous tools were lacking.

4.1 FogNetSim++: Toolkit for Modeling and Simulation

4.1.1 System Model

FogNetSim++ can support M fog nodes, P mobile nodes (d), and B broker nodes. A base station (BS)is used between mobile nodes and broker nodes. There are many types of moble devices like they can generate and send sensor data or they can upload task to broker for computation (asking for resources). So, mobile devices have the tendency to request resources from broker, and broker ask other nearby fog nodes to facilitate the mobile nodes. M/M/1 queue and M/M/c queue is used at mobile and fog node respectively for the sample traffic scenario. A mobile node ask the nearby broker for resources and offload task for computation, the broker communicate with nearby fog node and if the task size is not compatible or less resources are available at this fog node, the broker asks other nearby fog node for resources and further lease. The broker node keeps track of every task request and the current status of resources at every fog node. If broker could not find resource for an incoming task, it will add the task request to its queue, and retry later. This way, the delay increases. Hence after time t the request will be dropped and ask for re transmission. The task generation ($d_i, i \in P$) at mobile node follows the Poisson process with average req rate (μ_i).

A very computationally extensive request is generated from device (d_i). This request is independent of the schedule, it can be schedule at any fog device. The task is executed at any of available k number of homogeneous fog devices. The θ_f represents the execution rate and ϕ_f represents the max resource and load at a fog device. The maximum load at fog node is defined so that unnecessary delay can be ignored. B received the request from d_i and according to Poisson process:

$$\mu_{total} = \sum_{i=0}^{P} \mu_i. \tag{4.1}$$

A node can accept the request as:

$$\varphi = \begin{Bmatrix} 1 & \phi_f > \mu_{total} \\ \frac{\phi_f}{\mu_{total}} & \phi_f < \mu_{total} \end{Bmatrix}. \tag{4.2}$$

Therefore, using (4.1) and (4.2), the execution rate can be compared at fog device as:

$$\vartheta = \mu_{total} \times \varphi = \begin{Bmatrix} \mu total & \phi_f \geqslant \mu_{total} \\ \phi_f & \phi_f < \mu_{total} \end{Bmatrix}. \tag{4.3}$$

Using queuing theory analysis and Erlang's formula [47], the avg waiting time about every request can also be computed as.

$$T_{wait} = \frac{\kappa, \frac{\mu_{total}}{\kappa\theta_f}}{\kappa\theta_f - \mu total} + \frac{1}{\theta_f}. \tag{4.4}$$

The broker module have all the responsibility to perform and execute the tasks. If broker node doesn't have enough resources it forward the task to the neighboring fog node. If the neighboring doesn't have enough resource the task is offloaded to cloud. In case of no resources available it the cloud, the task is placed in queue of fog node for time t and after this time t retried to offload the task or drop the task, and request re-transmission.

The proposed toolkit, FogNetSim++ is designed to support the execution of task at the fog nodes at the edge of network. It is observed that less resources are available at the fog layer when we compare it to the cloud. I cloud we have a huge amount of processing power and resources but at fog these resources are always in limited amount. This is the reason efficient usage of these resources is required at fog layer. Hence, FogNetSim++ proved a number of algorithms that share these resources such that to have maximum efficiently and usability of all available resources. Also, researchers can incorporate their own task scheduling algorithms to test and compare performance. User nodes send tasks to the fog nodes and fog node check if it has the enough processing power or computation resources. If resources are available at the fog node, it starts execution of the task. Otherwise, fog node ask nearby fog nodes to execute task also keeps the record of task so that to return response and result of respective user node. FogNetSim++ uses publish/subscribed based model to reserve resources and communication. sensor devices are also supported in FogNetSim++

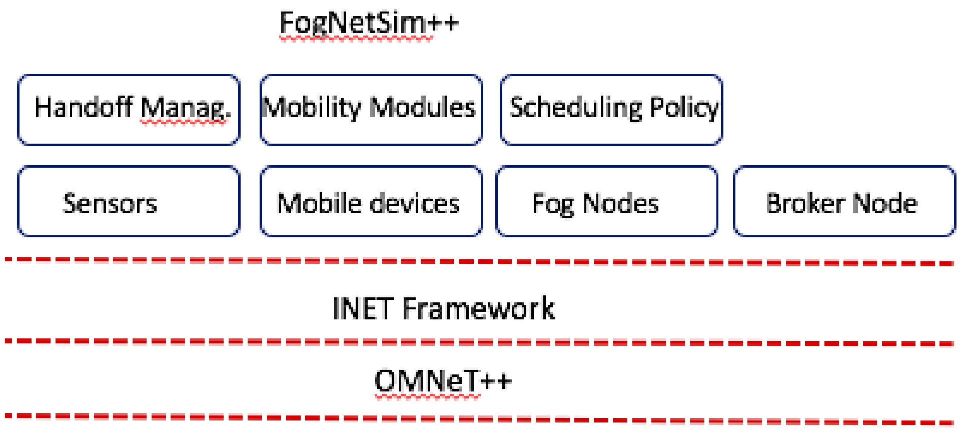

Figure 4.1: FogNetSim++ Design

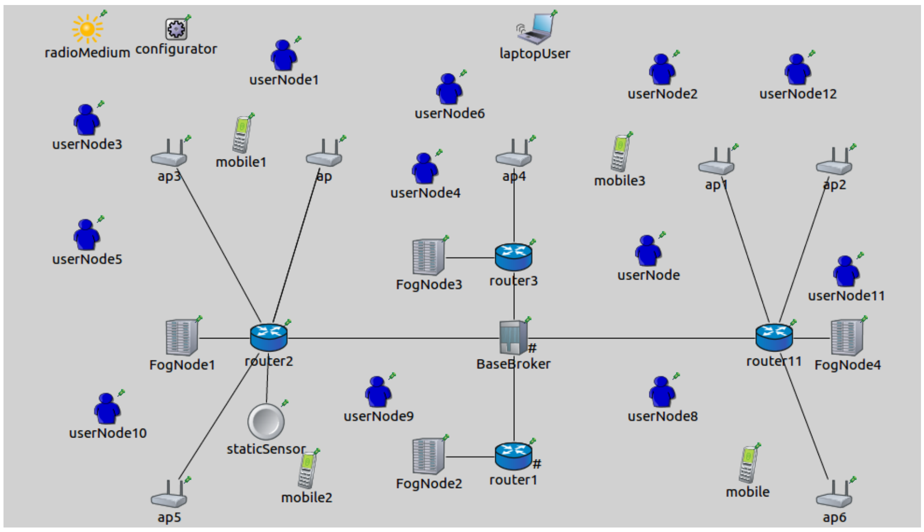

Figure 4.2: FogNetSim++ - A Graphical User Interface

Algorithm 4.1 Algorithm – Broker Node (B)

```
List fogNodes[] ← $FD_M$
List devices[] ← $d_P$
Queue taskQueue[] ← $nill$
Timer timer ← 0

while ( $true$ ) do
    if $MessageRecInWaiting$ then
        $Msg \leftarrow Message.Received$
        if $Msg.Type == Result$ then
            Forward $Msg$ to $d_i$
        else if $Msg.Type == FN.\Phi_i$ then
            Update $FN_i.\Phi_i \leftarrow \Phi_i$
        else if $Msg.Type == d_i.position$ then
            Update All $d_i.Position$ for $Handoff$
        else if $Msg.Type == Service_{req}$ then
            $taskQueue \leftarrow Msg$
        end if
    end if

    if taskQueue $! = Empty$ then
        $Msg_{req} \leftarrow taskQueue_{pop}$
        if $FN_i.workload < FN_i.\Phi_i$ then
            Forward $Msg_{req}$ to $FN_i$
        else
            $boolflag \leftarrow true$
            for i=0 .. M do
                if $FN_i.workload > FN_i.\Phi_i$ then
                    $Msg_{req} \leftarrow taskQueue_{pop}$
                    Forward $Msg_{req}$ to $i$
                    $flag \leftarrow true$
                end if
            end for
            if flag then
                $Starttimer \leftarrow \Delta_T$
            end if
        end if
    end if

    if $timerexpire$ then
        for i=0 .. M do
            if $FN_i.workload > FN_i.\Phi_i$ then
                $Msg_{req} \leftarrow taskQueue_{pop}$
                Forward $Msg_{req}$ to $i$
            end if
        end for
    end if
end while
```

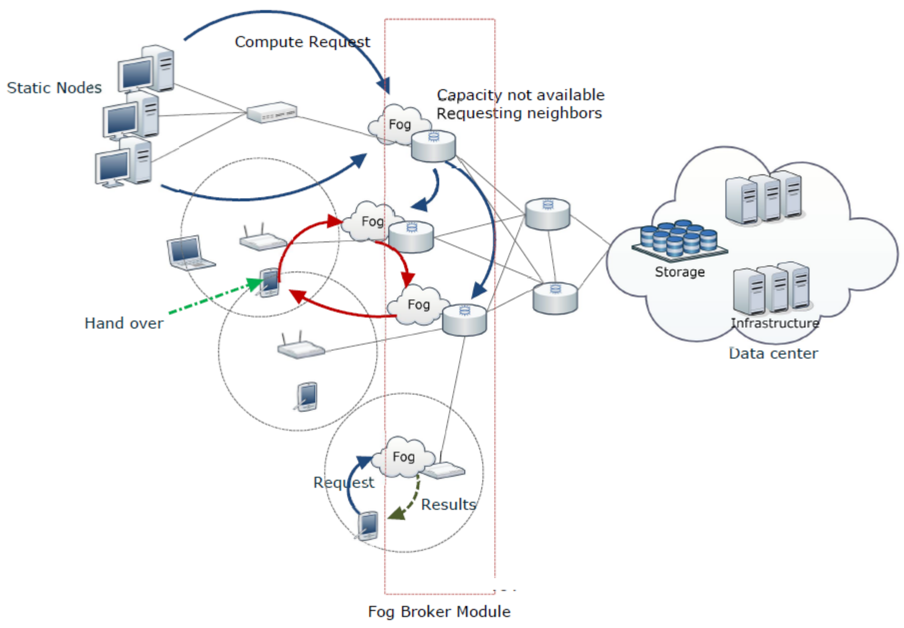

Figure 4.3: FogNetSim++ Working Model

4.1.2 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 4.1. For the pricing model, the broker is responsible for managing the user SLA (service level agreement).

4.1.3 Energy Modeling

We proposed an energy model to estimate the energy consumption of wireless nodes. Generally a sensor node consists of three main parts; a transceiver, a micro-controller and power supply. FognetSim++ provides uses the existing power supply modules in Inet framework like ICcEnergyGenerator and IEpEnergyGenerator. A wireless node has two parts software part and hardware part. In software part the running application at the node which generates tasks and in hardware part two energy consumers as micro-controller and transceiver. The sample values for energy consumption are listed in TABLE **??**. The state diagram of the current energy model is presented in the

Algorithm 4.2 Algorithm – Fog Node (FN)

```
1: Timer timer ← 0
2: Queue taskQueue[] ← $nill$
3: while ( $true$ ) do
4:     if $MessageRecInWaiting$ then
5:         $Msg \leftarrow Message.Received$
6:         $taskQueue \leftarrow Msg$
7:     end if
8:     if $taskQueue! = Empty$ then
9:         $Msg_{req} \leftarrow taskQueue_{pop}$
10:        $Outcome \leftarrow ExecuteMsg_{req}$
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
```

Table 4.1: 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

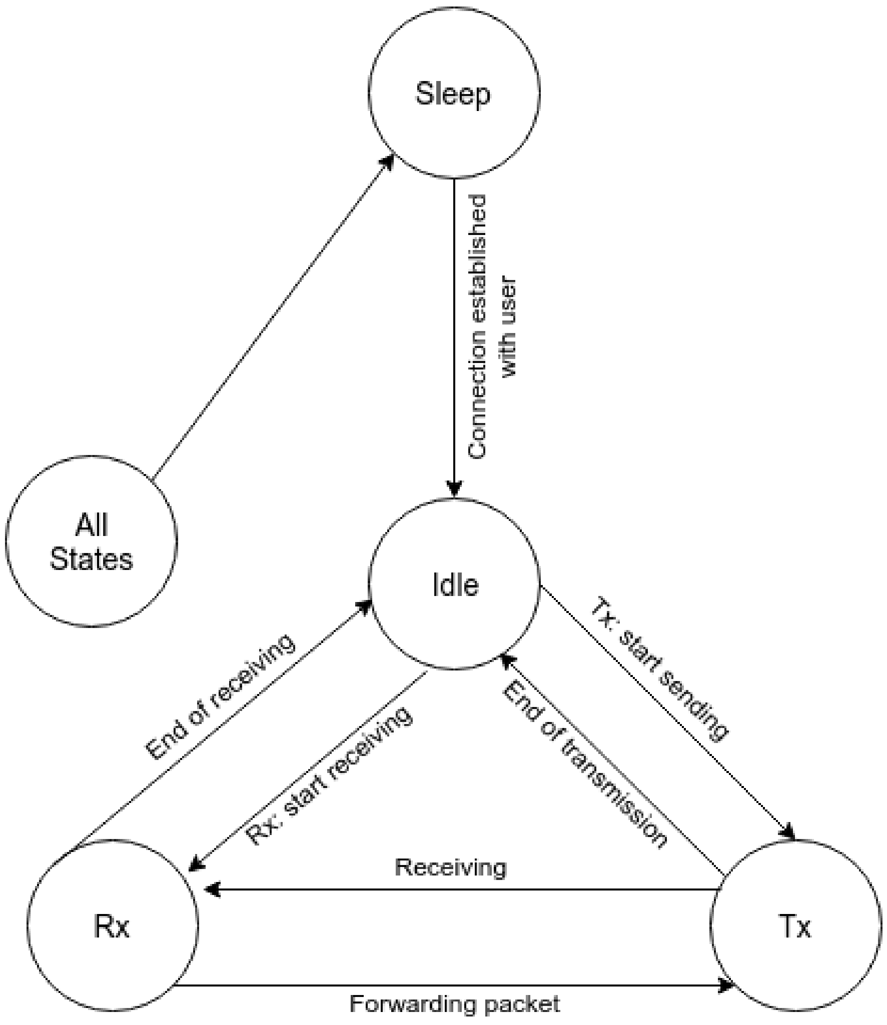

Figure 4.4: State diagram for energy model

Fig. 4.4. The energy consumption is calculated as:

$$E_t^{fn} = \frac{S(t)}{B} \cdot P_{trans} + \frac{S(t)}{\gamma_{fn}} \cdot P_{idle} \tag{4.5}$$

In eq. 4.5 the energy consumption E_t^{fn} is given for task t while computing at fog node fn. This energy is the combination of two things, first the power consumption at user node in uploading the task t of size $S(t)$ on data channel with bandwidth B and transmission power P_{trans}. Second, the power consumption at Fog Node (FN) in processing the task t of size $S(t)$ with fog node's computing power γ_{fn} and the idle power P_{idle}.

$$E = \sum_{t=0}^{Q} E_t^{fn} \tag{4.6}$$

Eq. 4.6 represents the total energy for Q number of tasks.

Table 4.2: Sample measurements of sensor nodes energy model's calibration in FogNetSim++

Measurement of radio power	
Sleep	60 μA
Idle	1.38 mA
Rx	9.6 mA
Tx (-18 dBm)	8.8 mA
Tx (-13 dBm)	9.8 mA
Tx (-10 dBm)	10.4 mA
Tx (-6 dBm)	11.3 mA
Tx (-2 dBm)	15.6 mA
Tx (0 dBm)	17.0 mA
Tx (+3 dBm)	20.2 mA
Tx (+4 dBm)	22.5 mA
Tx (+5 dBm)	26.9 mA

4.1.4 Implementation

The proposed toolkit is developed as an extension to *OMNeT++*. Omnet++ is an open sourced tool to simulate discrete event based network simulations. It already has a number of modules developed by the contribution of community. FogNetSim++ is also an open source simulator so that it can be available freely to everyone and everyone can enhance it by incorporating their own algorithms. Fig. 4.1 depicts the design of FogNetSim++.

The existing simulators are not able to support all kind of sensor devices. Hence, this is the motivation for designing FogNetSim++, which support all kind of sensors and IoT devices [16] [17]. Also, they ignore basic network characteristics like delay, packet loss etc. The existing work also doesn't support mobility models [6]. Fig. 4.2 shows the geographical representation of the toolkit. FogNetSim++ is a complete solution for the simulation of network environment using sensor nodes, broker nodes, and cloud data centers. The Broker nodes manages the requests and resources. FogNetSim++

is novel toolkit that provide the facility to simulate both static and mobile nodes. Fig 4.1 depicts the modular diagram of FogNetSim++. The Toolkit is composed of two main modules, user devices and brokers.

The mobility can introduce new challenges resulting from of device/node hand over. The proposed framework provides a number of mobility models. Moreover, the framework also supports network protocols that include UDP, TCP, and MQTT. A working example of the use of these protocols is included in the proposed framework, so researchers can customize as per their requirements.

4.1.5 Implementation – FogNetSim++

The Broker is the core module in FogNetSim++, it handles the user devices as well as fog nodes. User nodes establish a connection with broker using MQTT protocol, broker is managing a table with all active user nodes. User send task to the broker with publish message. Fog nodes also establish connection with the broker, and keep updating the broker about their queue, waiting time, and computation power. When the task arrives at broker, broker calculate the best fog node at which this task will be executed using the task scheduling algorithms present in FogNetSim++. If User Fog node doesn't have enough resources, the broker asks other fog node. If this node also don't have enough resources to execute the task, then task is sent to cloud for execution or in case of issue with cloud the task is placed in the queue of fog node. The task is picked from from the queue when time arrives or the queuing time of the task expires. Fig 4.3 shoes the load and internal structure. When the task is computed, the fog node send results to the broker, and then broker forward this result to the relevant user. In this way, no task or data is loss. FogNetSim++ assure the delivery of network packets with the assured delivery network protocol MQTT. In FogNetSim++, researchers can implement their own algorithms to schedule tasks in better way and they can enhance the mobility by adding new mobility models. Handover is another important and crucial stage while working with wireless nodes. Nodes keep moving while sending request for a task computation and when results arrives about that task, node has moved from its location. To find that node in the network and return the result about its task is referred as handover. FogNetSim++ performs handover in very accurate manner so that to avoid any data loss.

FogNetSim++ provides complete luxury of playing with physical network characteristics like, packer loss, bit error, and bandwidth etc. Users can change these network parameters very easily and test the network simulation environment. This toolkit also provides a number of network protocols like MQTT, HTTP, FTP, USD, TCP, and AMQP. Further, researchers can enhance the tool by adding new protocols which is very easy. FogNetSim++ provides the facility to simulate heterogeneous devices. Sometimes, sensors are fixed at different locations and sometimes they keep moving a particular mobility pattern. Hence, both mobile and static devices are supported by the FogNetSi++. There is a limitation in FogNetSim++ which says that both IPv6 and IPv4 cannot be used at the same time but these versions can be used in FogNetSim++.

The broker module uses publish, subscribed method to dispatch data to the user devices and fog devices. devices can connect to broker in two possible ways, either user device, or as a fog node. If node is connected as fog node, it means this node is proposing its computation and storage resources to the network and other devices can use these resources through broker. Broker will decide which fog node will be used as computing node. The MQTT protocol[1], is implemented in the FogNetSim++, which was not present in the previous versions of Omnet++.

A simple communication between broker and user device is presented in Fig 5.1. The devices can register with broker node through *Register(...)* function call. The call includes arguments such as broker *id*. The broker node receives the updates through a function call *Update(....)*. The updates are pushed to the subscribers through *Reflect(....)* function call. The broker also sends an acknowledgment to the publisher device/node. Any node can act as a subscriber/publisher or both. The broker maintains key terms/topics. A node can also request for computing power available at the broker. This type of communication is handled through TCP/UDP protocols. Internal execution of MQTT at broker node is shown in Fig 4.5.

MQTT is a publish/subscriber based lightweight messaging protocol, first created by Arcom/IBM in 1998. Now it is an ISO standard (ISO/IEC20922)[2]. MQTT is implemented in FogNetSim++ for communication among devices and broker nodes.

Broker Module

As discussed above, broker node is responsible to provide resources on request. In FogNetSim++ we have categorized broker node into three types i.e. static, mobility-based, and Wireless access based.The static nodes are the computing servers, placed at the gateway to provide computing on request.

[1]http://mqtt.org/tag/standard
[2]http://mqtt.org/tag/standard

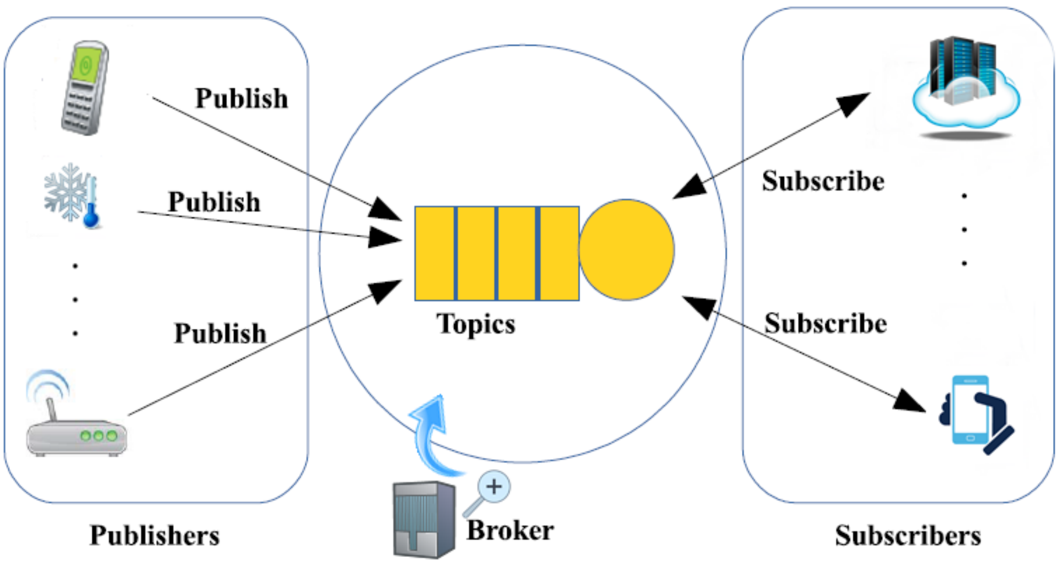

Figure 4.5: FogNetSim++ - A basic view of architecture

The mobility-based are the actual mobile nodes that can offer computation to its neighboring nodes. The Wireless Access Based (WAB) is the access points that can be used to serve the connected nodes. The internal modules of broker node is shown in Fig 4.6. The WAB is designed on the inspiration of Cisco Edge[3] series routers which provide the facility of computing along with the routing. A brief list of broker features is listed below:

- Manage a list of publishers and subscribers
- The publisher/subscribers topics are provided to broker through configuration file
- Register publishers, subscribers and disseminate updates
- Provide computing capacity on request
- Optimally utilize resources among number of requested nodes
- If required, perform resource handover with neighboring brokers
- Communicate with data centers
- Provide interface to incorporate new algorithms for resource scheduling and computing

[3]www.cisco.com

- Reliable data delivery
- Run at the edge, intermediate and smart gateway nodes

As explained above, the most important node is the broker which facilitate the user nodes to provide resources as request arrives. The user nodes are categorized into two types, one is static nodes that are connected to network with Ethernet cables and the second is wireless nodes. Similarly, the broker module is also categorized into two types, one is wired brokers in the network and the other one is wireless that can be wireless nodes, or the wireless access points. Fig.Fig 4.6 shows the internal structure of broker module. As the Cisco[4] has already introduced the routers that can perform computations at themselves. FogNetSim++ provides the facility to users to simulate network routers as broker nodes. Algorithm 4.1 and Algorithm 4.2 are deployed at broker and fog nodes respectively. The sample simulataion parameters about the environment is given in Table 4.3

Table 4.3: 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)

In FogNetSim++, brokerApp is an abstract class designed to work on IoT protocols. It provides a skeleton class for fog node implementation. By default it supports all types of brokers packed inside FogNetSim. The implementation manages lists of subscribers along with their other required data structures to support above mentioned functionality.

End Node Devices

Another core module of FogNetSim++ is end devices. Under end devices, we have two type – sensor nodes and user node. The sensor nodes act as a data generator; whereas, the user node can generate or receive data. The user node can also act as a sensor. Further, these nodes are further categorized into two – wired and wireless nodes. The nodes can be static or mobile, both versions are supported. The salient features are given as:

- The sensor class register to publish data

[4]www.cisco.com

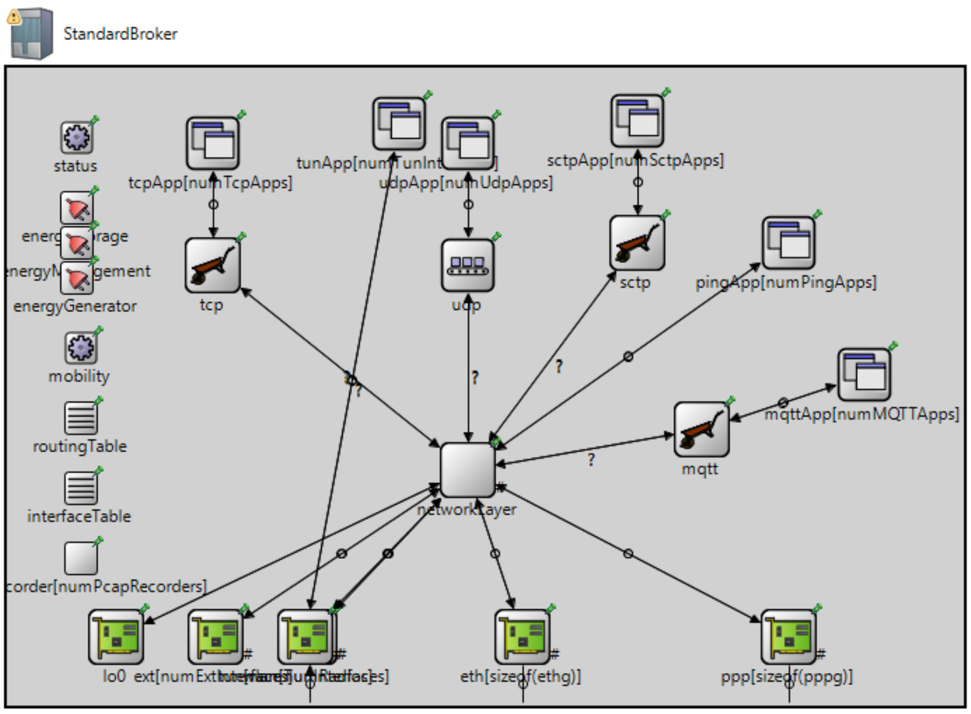

Figure 4.6: Internal modular view of Broker Node

- The objects inherited from user node class can register and subscribe for data
- The data is pushed to the broker node
- Node acquire IP address through DHCP
- All nodes support MQTT protocol along with other application protocols such as HTTP, TCP, FTP, SMTP, SNMP and UDP. Further, researchers can write their own protocols for these nodes by extending the interface class

Following parameters are used to set the initial values:

- numMQTTApps - parameter is used to specify the number of MQTT applications at each node
- destBroker – parameter the destination address of broker node, it can IP address or name of the Broker or can be provided dynamically

End Node Implementation – End node devices is a skeleton module designed to act as a sensor/user node that is mainly used to generate or consume data. It uses MQTT messages for communication with broker node. It provides a framework for researchers to incorporate their own application algorithms to generate data or consumption. The basic parameters required by this module are:

- string localAddress – acquire through DHCP
- int localPort – default(2498)
- double startTime
- string topics=default("") a list of topics, comma separated
- int connectPort – default(2498)
- int connectAddress – dynamically populate

Mobility Models – Mobility is another very important feature of end node devices. In FogNetSim wireless, and wired nodes/devices are supported. The mobility can play an important role, it can create a new challenges for researchers in the form of optimum use of resources and resources hand over. In OMNeT++, an open source module (i.e., INET) provides different type of mobility models. Normally mobility models are classified into two categories, one type is called "entity model" where movement of each node is independent of other nodes; whereas, the second category is called a "group model". In this model, movement of one node is dependent on the movement of other nodes. The most commonly used entity and group models are listed here:

- Random Waypoint – entity model
- Random Walk – entity model
- Guass-Markov – entity model
- Random Direction – entity model
- City Section – entity model
- Column Mobility – group model
- Pursue Mobility – group model
- Nomadic Community – group model
- Reference Point Group – group model

New models can also be integrated into FogNetSim++ but the integration is available for entity models only. The already available mobility models in FogNetSim++ are:

- Random Waypoint model – Introduces the pause times between variations in speed and destination.
- Mass Mobility – Introduces a mass point with momentum and inertia.
- Deterministic Motions models– The mobility of fixed point nodes as well as moving nodes around linear and rectangular paths.
- Chiang Mobility – Where probabilistic transition matrix is used to alter the state of motion.
- Gauss-Markov – Where a turning parameter is used to change the amount of randomness in the pattern.

Other then above listed models, the other available models are: StationaryMobility, StaticGridMobility, CircleMobility, LinearMobility, TractorMobility, RectangleMobility, TractorMobility, RandomWPMobility, GaussMarkovMobility, MassMobility, ConstSpeedMobility, ChiangMobility, Ns2Mobility, BonnMotionMobility, and ANSimMobility.

Chapter 5

Results & Discussion

5.1 Testing and Performance Evaluation

FogNetSim++ offers a comprehensive platform to simulate diverse fog applications. It also help to understand the basic concept of fog computing. FogNetSim++, provides a rich network configurations managed through a network module. It allows the simulation over realistic network environment that opens new challenges for researchers such as resources utilization, and resources handoff.

All the modules are configured through *ini* file. The user can set the values of different parameters such as the number of brokers, user nodes, applications at each node, link data rate, channel noise, and mobility models for every individual or group of nodes. To give the basic understanding of the framework and to evaluate the performance of a simulator, network scenario is simulated where a variable number of user nodes mobile and static are placed, they are generating messages to the broker nodes. The broker node executes the requested tasks in FIFO order and sends the result back to the node. The performance is measured in terms of memory and CPU usage. As the IoT simulator comprises of a large number of devices, therefore, it is important to benchmark the proposed FogNetSim++. Moreover, the network contains a number of standardBrokers, wirelessBrokers, accesspointBrokers, and a cloud data-centers. The system specifications are mentioned in Table 5.1. The parameters used in a simulation is listed in Table 5.2.

5.2 Case Study

FogNetSim++ and its working can be understand by a network scenario. In this sample network scenario, the traffic model is simulated where vehicles

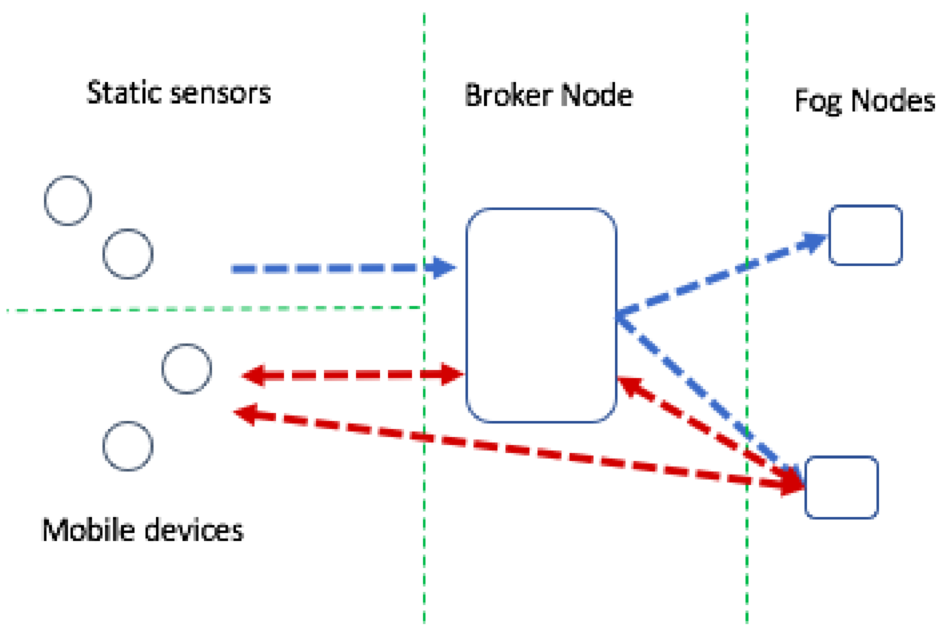

Figure 5.1: Case study - simulated topology

have send data, and there are some fix points where sensors are installed to generate the data. Pedestrians also taking part in the network, they have mobile phones and users can move freely in the map and generate data. This generated data is sent to broker and fog nodes for further processing and the respective action is taken place according to the results of this generated data. Law enforcement agencies can use this system to identify faces and vehicles at a particular part of the city or map. They have ask fog nodes to process data for rapid action based on the results. This is the best application of fog in vehicular and security point of view. The fog nodes are distributed geographically on the map so that to enhance the performance. The network is analyzed according to the above mentioned network situation. The architecture of the above network is shown in Fig. 5.1.

The sensor devices generates data and send it to the broker. The broker then receives and forward this data to the other devices based on the publish, subscribed model. When a broker receives a computation request, it runs its task scheduling algorithms and assign task to the best fog node for computation. The fog nodes continuously keep updating broker about their resources. The broker also keeping a track on the devices to assure safe handover. The result is then returned to the subscribers (user devices) and action is taken placed according to the result. FogNetSim++ is salable is it is measured in terms of CPU usage and memory. As the FogNetSim++ supports a large number of devices and sensors so the tool is bench-marked in terms of CPU usage and memory usage. The delay, latency, error rate, and task computation time measured. The simulation used four types of task sizes. The small, medium, large and random with 200, 900, 1500 MIPS, and a random number in rage of 200-1500 MIPS. The simulation is executed for a fixed amount of time. The fog nodes have different amount of resources that can be initialed in .ini file. The simulation results show that the value of these parameters, latency, delay, and error rate is different for different size of tasks. Table 5.1 shows the specification of system as well as simulation parameters.

Table 5.1: Machine Specification

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

The above simulation environment is tested by varying the number of nodes and it is observed that with the increase in number of nodes the CPU also increases as referenced in the The Fig. 5.2. However, it is also observed that for around 1300 nodes the total CPU usage is less than 25% which is very cost effective.

Similar behavior is observed with the memory usage. It also increases with the increase in number of nodes as shown in Fig. 5.3. Also, results show that overall memory usage for around 1300 nodes is 15% which is very affordable. The memory usage depends on the number of nodes but not in very extensive sense. As graph shows that it is always less than 15% even when the number of nodes are around 1300. The reason about this less memory consumption is that most of the nodes usage shared resources and hence the overall performance is improved.

The FogNetSim++ is implemented and developed as a top layer of Omnet++. The Omnet++ has a limitation of delayed construction of enriched graphical interface. The FogNetSim++ is evaluated by increasing the number of nodes and it is observed that with the increase in number of nodes

Table 5.2: 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 delay in constructing the GUI also increase. The Fig. 5.4 depicts that delay for 1300 nodes is around 600 seconds which a very big number. But this is not a big issue because this is only one time delay, as the simulation is loaded completely, there is no delay, and the simulator work in a streamline environment.

The average task computation time is plotted in Fig. 5.5, 5.6, 5.7, 5.8 for small, medium, large, and random-sized tasks respectively. As the results are taken by varying the task size and are compared. Average task computation time is represented by the dotted lines

As the tested simulation environment includes the wireless sensors with mobility. So the handover percentage is also evaluated and the results proved that with the larger task size, the handover percentage also increase. The Fig. 5.9 depicts the results about handover. This is because the task size is large, it needs more time in execution at fog but the user node is moving

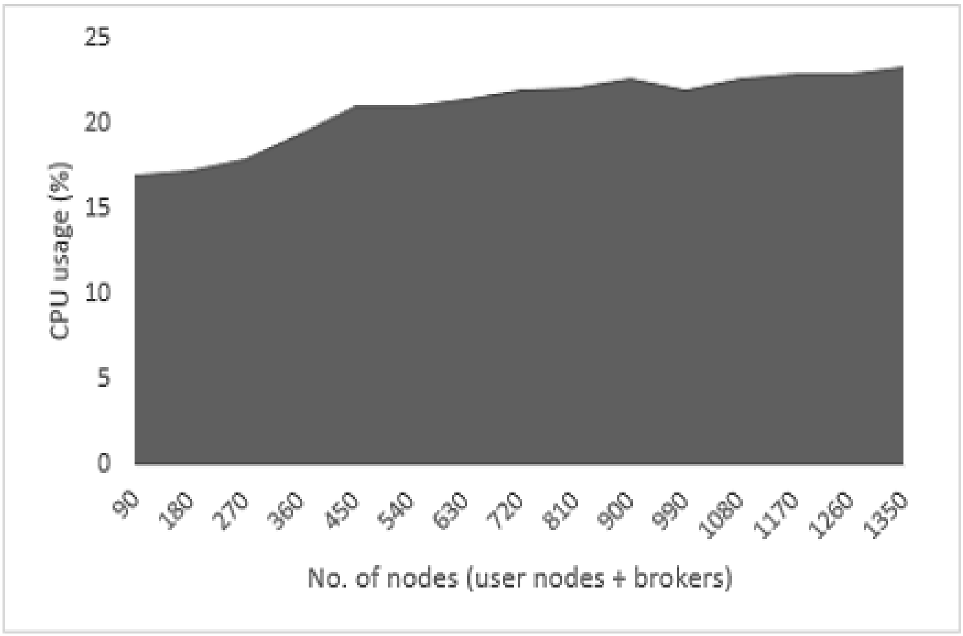

Figure 5.2: FogNetSim++ CPU usage with respect to number of nodes

continuously because of the mobility model installed at the node, and it passes through a number of access points' range and hence perform handover. Finally it receives the result from a closer fog node.

Fig 5.10 shows that the end-to-end delay also increase with the increase in number of nodes. This is because the in a network with large number of devices asking for task execution or resources, the congestion is observed as well as a queue is built at broker node which causes the increase in the values of end-to-end delay.

As discussed above that in simulation the task size is divided into four categories, small, medium, large and random task size. The task execution time is observed and evaluated which shoes that task execution time is directly proportional to the size of task as shown in Fig. 5.11. Hence, the usage of small-sized tasks can enhance the system performance.

However, Fig. 5.12 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.

The average Latency of the system is represented in the Fig. 5.13 where latency is showed at y-axis and the simulation time at x-axis. The energy is also simulated in for the case study scenario. The residual energy is the type of energy a node has at time t. The Fig. 5.14 explains the residual energy of

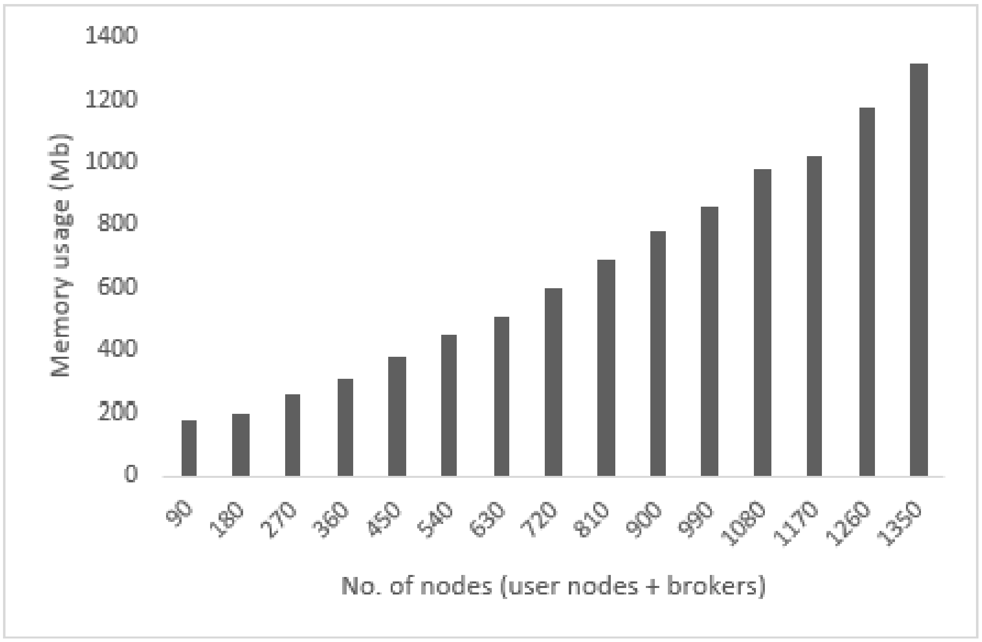

Figure 5.3: FogNetSim++ memory usage with respect to the number of nodes

nodes in FogNetSim++ at any time. As the simulation proceed the residual energy keeps decreasing and a time comes when the residual energy of node drops to zero and the node gets dead.

The consumption of energy also depends upon the transmission power of the wireless. More the transmission power more the energy is consumed. This behavior is represented in the Fig. 5.15. The results are obtained by changing the transmission power and results are plot in the Fig. 5.15. IoT devices have limited amount of energy so the energy can be saved by decreasing the transmission power. Yet it evolves a new problem of range but it can coped by installing more nodes in the network.

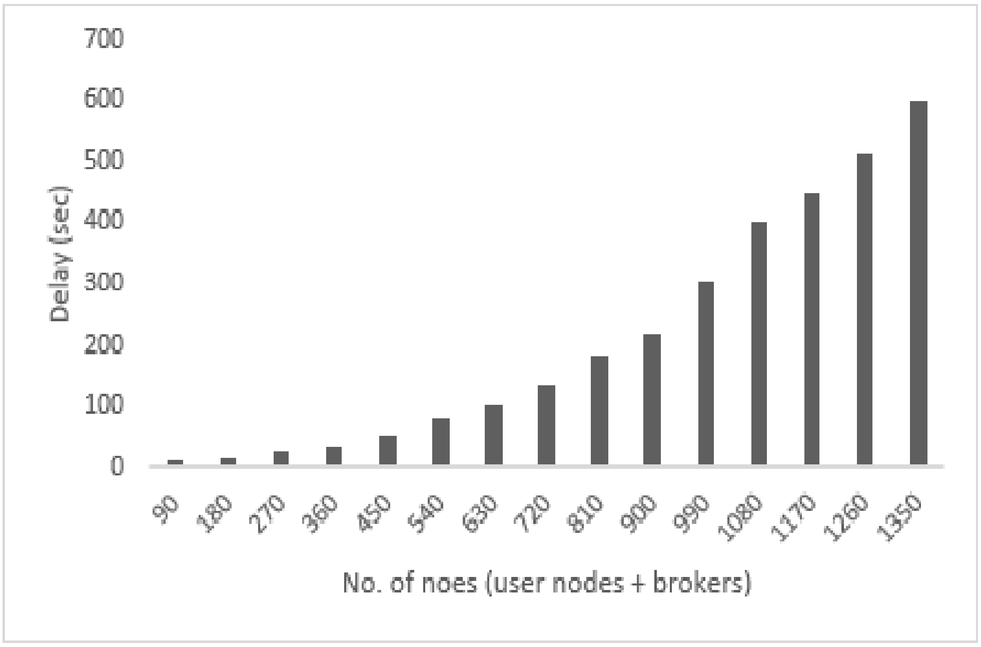

Figure 5.4: Delay in constructing enriched GUI

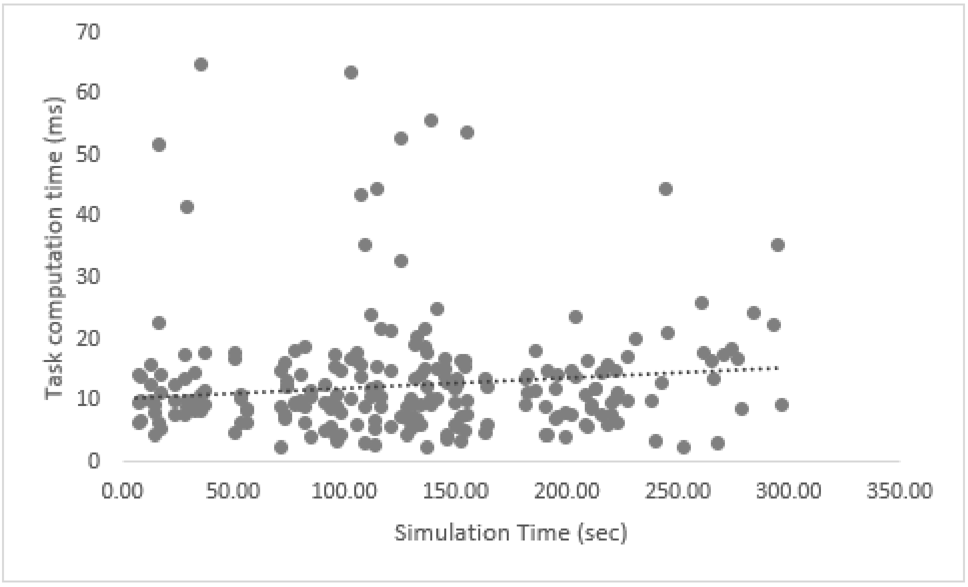

Figure 5.5: Average Task computation time for small-sized tasks

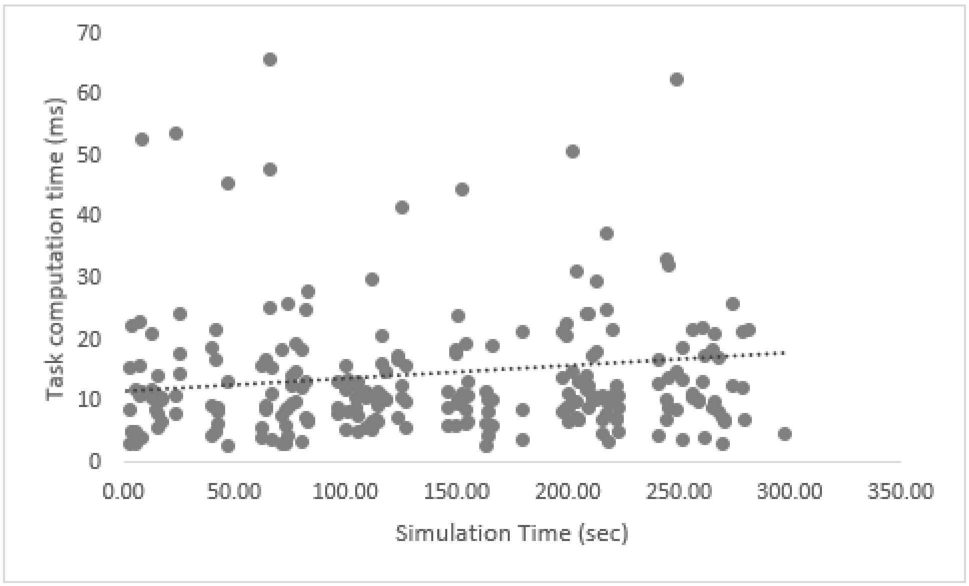

Figure 5.6: Average Task computation time for medium-sized tasks

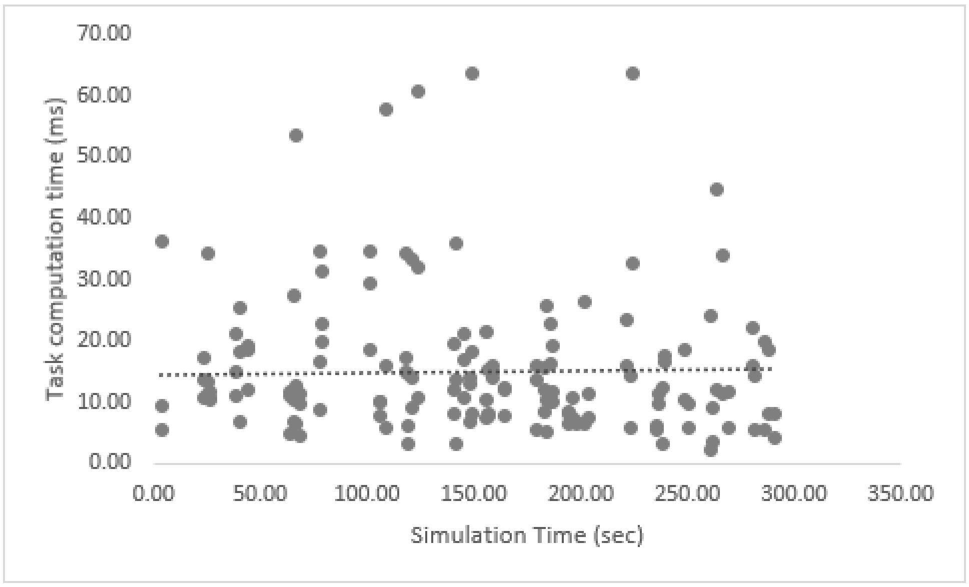

Figure 5.7: Average Task computation time for large-sized tasks

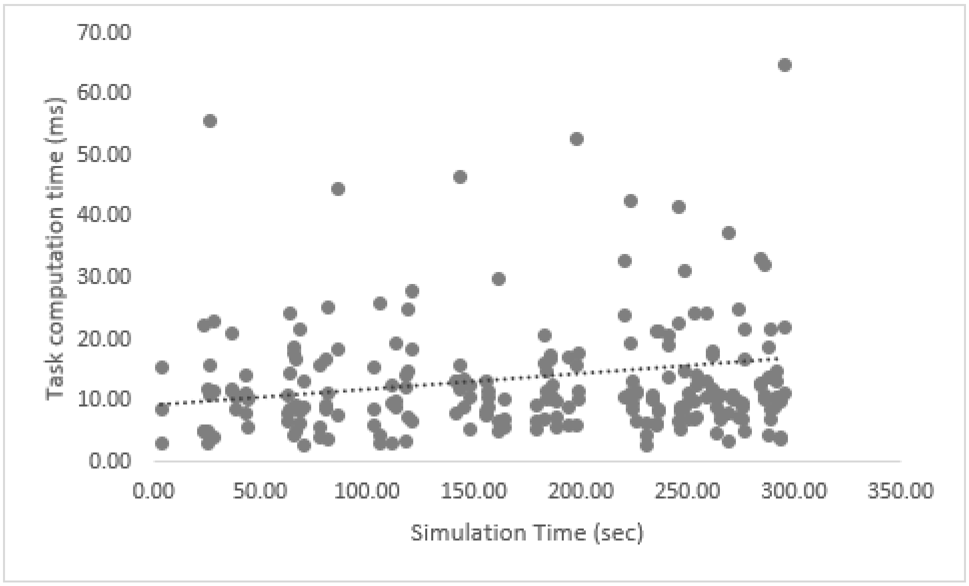

Figure 5.8: Average Task computation time for random-sized tasks

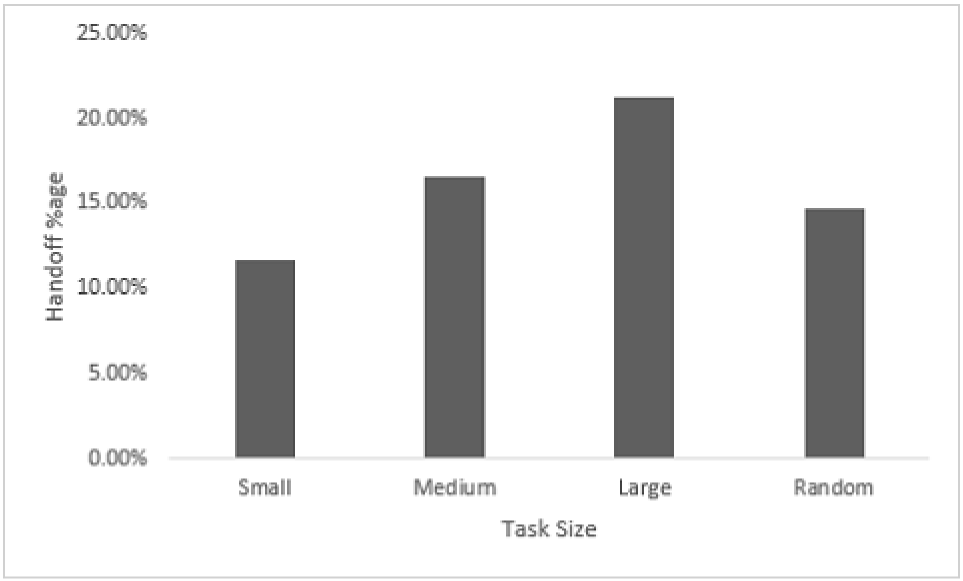

Figure 5.9: Handoff performed w.r.t task size

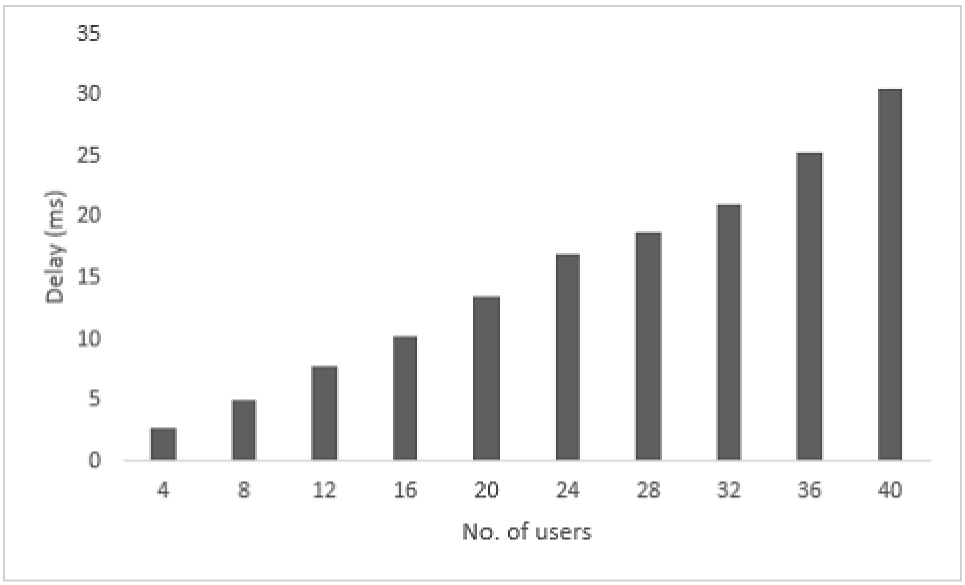

Figure 5.10: Average delay w.r.t task size

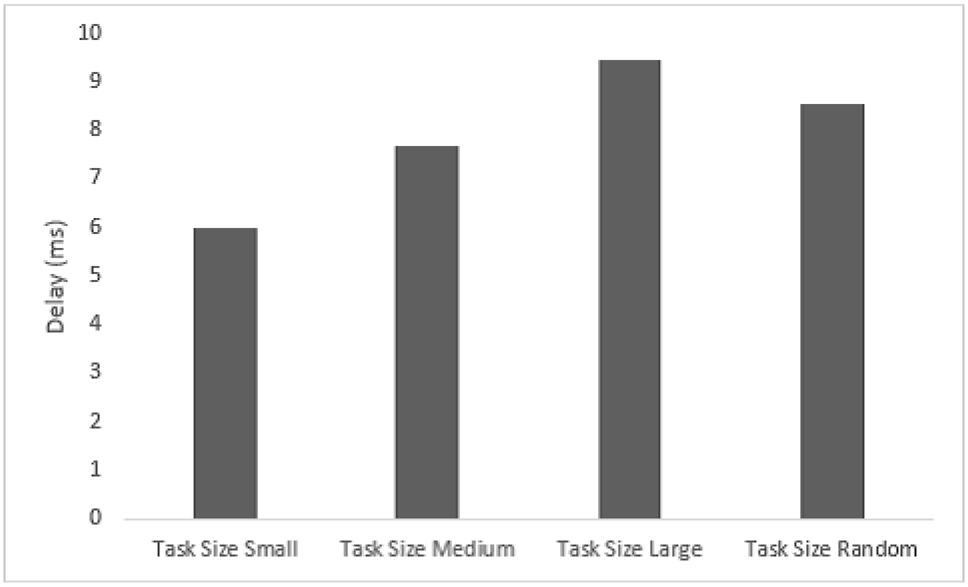

Figure 5.11: Total Delay based on the compute capacity requirement

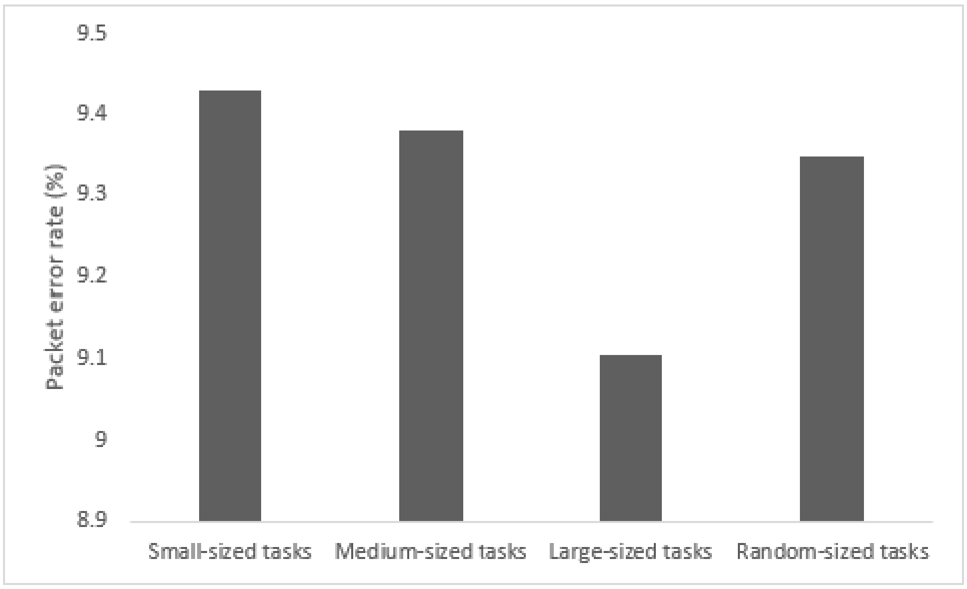

Figure 5.12: Average wireless error rate reported during the execution

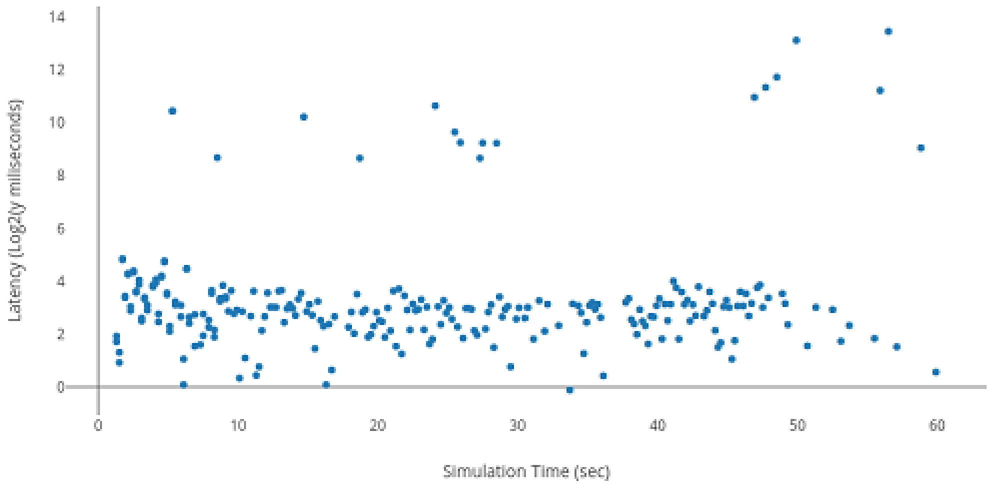

Figure 5.13: Average latency

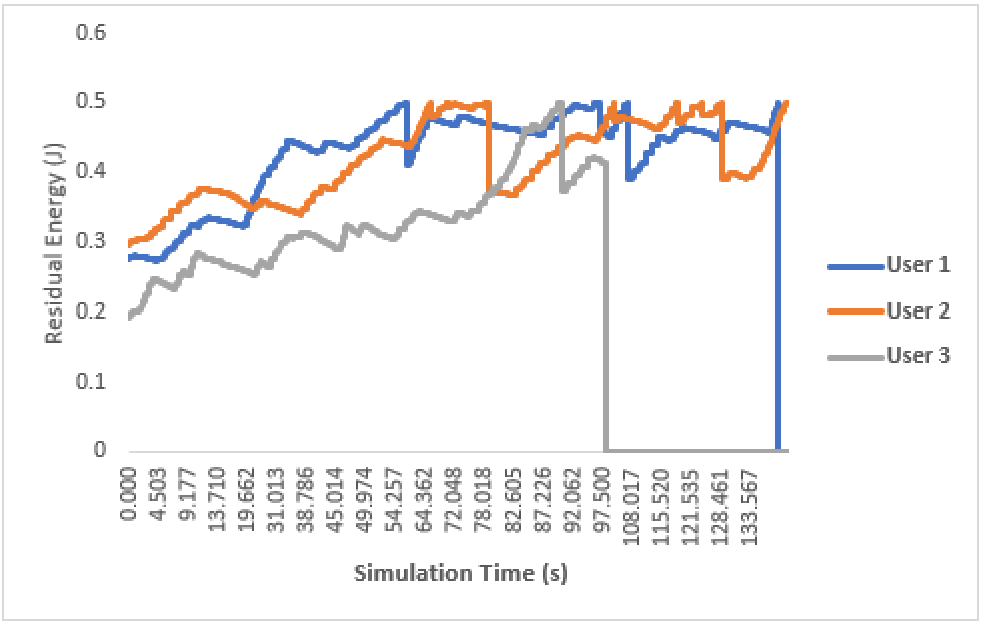

Figure 5.14: Residual Energy Vector over simulation time

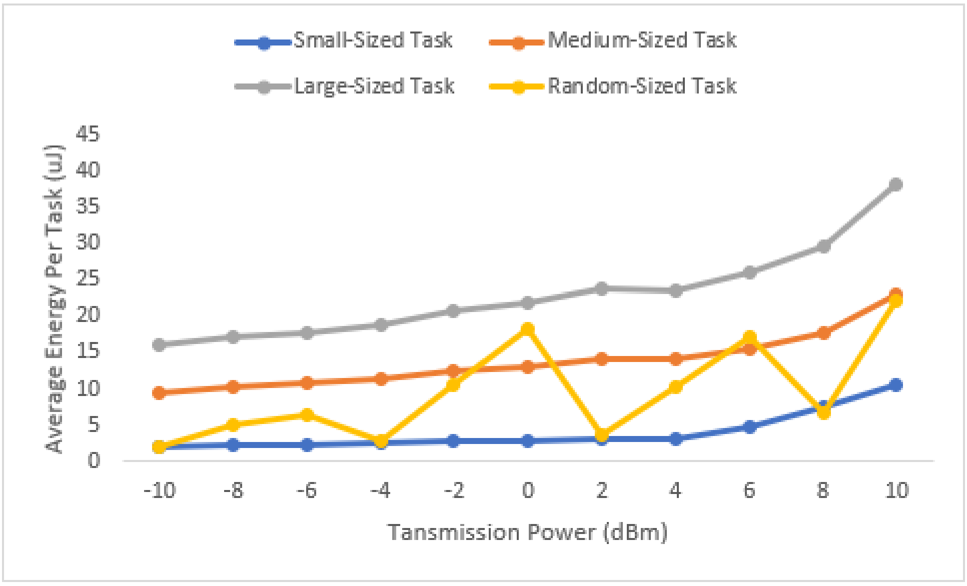

Figure 5.15: Average Energy Consumption vs Transmission power

Chapter 6

Conclusion & Future Work

Lastly, this chapter concludes the presented research work. In which, section **??** covers the future directions and some other research challenges that need to be addressed and section **??** presents the conclusion of this research work.

6.1 Future Work

FogNetSim++ can be enhanced by providing an extensive framework which helps the broker nodes to communicate with each other to share the resources more effectively in the network. Further, researchers can add their own algorithms about scheduling and resource allocation. FogNetSim++ is open to extend the mobility models. New protocols can also be introduced and can easily be added in the Toolkit. Researchers can work on the idea of using both IPv4 and IPv6 together in a simulation.

6.2 Conclusion

The introduction of edge and fog technoloes helped the delay-sensitive applications. The Fog computing is the extension to cloud. In proposed simulator, researchers can simulate distributed fog computing environments. Prior research has a number of gaps where FogNetSim++ helps. FogNetSim++ considers core network characteristics like delay, packet loss, error rate, and bandwidth which old simulators were lacking. The problem of safe handover is also solved in FogNetSim++. FogNetSim++ support a number of mobility models as well as several network protocols. A number of scheduling algorithms has been added in proposed toolkit. Researchers can also add new scheduling algorithms very easy. The energy module and a pricing model is also proposed in FogNetSim++. A sample network scenario is discussed and the results are discussed. The performance of the simulator is evaluated with CPU, memory, and drawing enriched GUI. The FogNetSim++ is benchmarked at different platforms and results shows that simulator is working very fine and well as compare to existing toolkit fro distributed fog computing.

Bibliography

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing.* ACM, 2012, pp. 13–16.

[2] P. Bellavista, L. Foschini, and D. Scotece, “Converging mobile edge computing, fog computing, and iot quality requirements,” in *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on.* IEEE, 2017, pp. 313–320.

[3] J. Pan and J. McElhannon, “Future edge cloud and edge computing for internet of things applications,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, Feb 2018.

[4] M. M. Gaber, J. B. Gomes, and F. Stahl, “Pocket data mining,” *Big Data on Small Devices. Series: Studies in Big Data*, 2014.

[5] W. Shi and S. Dustdar, “The promise of edge computing,” *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[6] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE pervasive Computing*, 2009.

[7] H. Goudarzi and M. Pedram, “Hierarchical sla-driven resource management for peak power-aware and energy-efficient operation of a cloud datacenter,” *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 222–236, April 2016.

[8] A. Singh, S. Sharma, S. R. Kumar, and S. A. Yadav, “Overview of paas and saas and its application in cloud computing,” in *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Feb 2016, pp. 172–176.

[9] M. Guzek, P. Bouvry, and E. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing [review article]," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 53–67, May 2015.

[10] S. N. Han, G. M. Lee, N. Crespi, N. Van Luong, K. Heo, M. Brut, and P. Gatellier, "Dpwsim: A simulation toolkit for iot applications using devices profile for web services," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on.* IEEE, 2014, pp. 544–547.

[11] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, "Towards simulating the internet of things," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on.* IEEE, 2014, pp. 444–448.

[12] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on.* IEEE, 2017, pp. 39–44.

[13] A. M. Khan, L. Navarro, L. Sharifi, and L. Veiga, "Clouds of small things: Provisioning infrastructure-as-a-service from within community networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on.* IEEE, 2013, pp. 16–21.

[14] T. Pflanzner, A. Kertész, B. Spinnewyn, and S. Latré, "Mobiotsim: towards a mobile iot device simulator," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW).* IEEE, 2016, pp. 21–27.

[15] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano, "Crowdsensim: a simulation platform for mobile crowdsensing in realistic urban environments," *IEEE Access*, vol. 5, pp. 3490–3503, 2017.

[16] "Simplesoft simpleiotsimulator," http://www.smplsft.com/SimpleIoTSimulator.html, accessed: 2018-08-10.

[17] "Ibm bluemix platform," https://console.ng.bluemix.net, accessed: 2018-08-10.

[18] "Parse," https://parse.com/products/iot, accessed: 2018-08-10.

[19] "Google cloud platform," https://cloud.google.com/solutions/iot/, accessed: 2018-08-10.

[20] 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," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[21] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "Fit iot-lab: A large scale open experimental iot testbed," in *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on.* IEEE, 2015, pp. 459–464.

[22] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Global Internet of Things Summit (GIoTS), 2017.* IEEE, 2017, pp. 1–6.

[23] "Contiki cooja," http://www.contiki-os.org/start.html, accessed: 2018-08-10.

[24] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Oct 2017.

[25] 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 *2017 Winter Simulation Conference (WSC)*, Dec 2017, pp. 4568–4569.

[26] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *2017 IEEE Fog World Congress (FWC)*, Oct 2017, pp. 1–6.

[27] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 39–44.

[28] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things (CIoT)*, Nov 2016, pp. 1–6.

[29] 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 *MCC@SIGCOMM*, 2013.

[30] 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. 21 126–21 138, 2017.

[31] 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 *Innovative Computing Technology (INTECH), 2017 Seventh International Conference on.* IEEE, 2017, pp. 191–196.

[32] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21 355–21 367, 2017.

[33] 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 of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[34] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A middleware for mobile edge computing," *IEEE Cloud Computing*, vol. 4, no. 4, pp. 26–37, 2017.

[35] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[36] 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.

[37] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017.

[38] X. Gong, L. Guo, G. Shen, and G. Tian, "Virtual network embedding for collaborative edge computing in optical-wireless networks," *Journal of Lightwave Technology*, vol. 35, no. 18, pp. 3980–3990, 2017.

[39] 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. 16 406–16 415, 2017.

[40] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, “Exploiting massive d2d collaboration for energy-efficient mobile edge computing,” *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

[41] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, “Incorporating intelligence in fog computing for big data analysis in smart cities,” *IEEE Transactions on Industrial informatics*, vol. 13, no. 5, pp. 2140–2150, 2017.

[42] 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. 21 296–21 311, 2017.

[43] X. Meng, W. Wang, and Z. Zhang, “Delay-constrained hybrid computation offloading with cloud and fog computing,” *IEEE Access*, vol. 5, pp. 21 355–21 367, 2017.

[44] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, “Security and privacy in fog computing: Challenges,” *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[45] 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 of Things Journal*, vol. 4, no. 5, pp. 1143–1155, 2017.

[46] 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. 17 962–17 974, 2017.

[47] B. Ngo and H. Lee, “Analysis of a pre-emptive priority m/m/c model with two types of customers and restriction,” *Electronics Letters*, vol. 26, no. 15, pp. 1190–1192, July 1990.