Birla Institute of Technology & Science Pilani

2022

Report

on

Fog Computing Based IoT Networks

Submitted In Fulfillment of the Course

Study Oriented Project – CS F266

Submitted by

SIDHARTH VARGHESE
2019A7PS1133P

Under the supervision and guidance of

Dr L Rajya Lakshmi
Faculty, Department of CSIS
Birla Institute of Technology & Science, PILANI, 333031
Rajasthan

Acknowledgement

I would like to acknowledge the guidance provided by my supervisor and guide- Dr L Rajya Lakshmi, without whom this project would not have been possible. I have learnt so much throughout the course of the semester. The guidance provided by her has been the key to the success of this project, and I would like to express my immense gratitude towards her.

Sincerely,

Sidharth Varghese
2019A7PS1133P

Table of Contents

Introduction and Motivation

Fog and Cloud computing paradigms have emerged as a backbone of modern economy and utilize Internet to provide on demand services to users. Both of these domains have captured significant attention of industries and academia. But because of high time delay, cloud computing is not a good option for applications requiring real-time response. Technological developments like edge computing, fog computing, IoT and Big Data have gained importance due to their robustness and ability to provide diverse response characteristics based on target application Fog is an emergent architecture for storage, computation and networking that distributes these services closer to end users along the *"cloud-to-thing continuum".*

It is especially useful when interfaced with IoT devices, and has several use-cases. It is utilized along with the cloud to provide the best experience for the users. In order to implement Fog systems, it is important to understand it completely first. This can be done by simulating the system on the computer. There are several softwares that can be used, among which the mostpopular is iFogSim on Java. Through the course of this project, I have worked on how we can incorporate Fog Computing in the Healthcare Industry. The software I have worked on is iFogsim. I have used both fog-based, and cloud-based architecture for the simulations in order to judge which would be more effective, and have presented my results and conclusion.

Background Information

What is IoT?

IoT is a self-configuring and adaptive complex system made out of networks of sensors and smart objects. These objects collect and transfer data over a wireless network without human involvement. The main purpose of an IoT systemis to interconnect everyday devices to make them intelligent and interactive with humans. Any object can be part of the IoT system, as long as it has its own IP address, and can connect and send or receive data through a network. An IoT device is made up of 3 things: A measure (Also called a sensor), an identity, and an actuator. Some of the technology in our day to day lives that incorporate IoT include smart fridges, cleaning robots etc.

What is Cloud Computing?

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Large clouds often have functions distributed over multiple locations, each location being a data center. Cloud computing relies on sharing of resources to achieve coherence and typically using a "pay-as-you-go" model which can help in reducing capital expenses but may also lead to unexpected operating expenses for unaware users.

The term ‘cloud computing’ also refers to the technology that makes cloud work. This includes some form of virtualized IT infrastructure—servers, operating system software, networking, and other infrastructure that’s abstracted, using special software, so that it can be pooled and divided irrespective of physical hardware boundaries.

What is Fog Computing?

Fog computing is a decentralized computing infrastructure where data, computation, and storage is done somewhere between the data source and the cloud.
An intuitive way of defining fog computing can be- a scenario where a huge number of heterogeneous universal and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxedenvironment.

It complements the cloud in the sense that it is mainly for short-term analytics that doesn’t require resource heavy analysis. It brings the advantages and powerof the cloud closer to where data is created and acted upon. It is more scalable and gives a better big-picture view of the network as multiple data points feed data into it.

Fog Computing Architecture

A typical fog computing model would primarily include 3 layers- The Cloud Layer, The Fog Layer and The Terminal Layer.

Cloud layer:

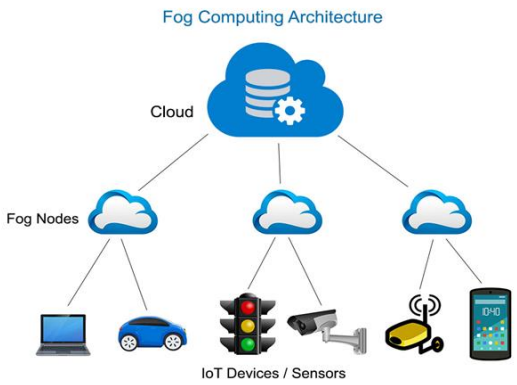
This is the topmost layer. Physical data center nodes are placed here. Each nodehas

CPU(s), main memory and network bandwidth and is used to satisfy user requests for resources. Clouds are connected to Wide Area Networks (WANs) and provide economic benefits, elastic services, and data-intensive analysis for end-users. However, clouds suffer from high latency and are incapable of supporting context-aware computing for IoT applications.

Fog layer:

This is the next layer, between the end devices and the cloud. It is a collection ofprocessing devices, gateways, and networked devices (routers and switches).
Fog resources are interconnected and used to deliver a number of services tousers such as computing, storage and network services.

Diagrammatically, it can be represented as:



Issues of IoT with Cloud Computing

Modern day IoT devices suffer from several issues that cannot be adequatelysolved by current cloud computing models.

Some of these include:

1. *Lack of Computational Resources and Bandwidth Constraints*

   Most IoT devices are severely lacking in computational capabilities, since the

bulk of the device is made up of sensors and actuators. However, if allthese devices are to interact directly with the cloud, it would lead to heavynetwork congestion and costs. Additionally, since each device collects massive amounts of data, sending all this to the cloud for processing is often impractical.

2. *Requirement of Uninterrupted Services*

   Cloud services cannot provide uninterrupted services to devices that have intermittent network connectivity (devices that only have satellite communication channels to connect to the cloud). This could have a massive impact in certain situations, like an emergency at an oil rig.

3. *Security Issues*

   Every day, the number of connected devices increases. Hence, a growing challenge is how security credentials of these devices are to be managed and kept up to date. Clearly, we cannot connect every device to the cloud. For example, in a smart city, there are millions of interconnected devices that keep sending data. A key issue is to ensure that this data is both sent securely, and processed quickly. The trustworthiness of the device can be proved cryptographically, but as mentioned before, an individual IoT devicewill not have the computational capacity to solve these cryptographic functions. Additionally, anytime a security flaw is encountered, the response is very disruptive, often shutting down the entire network, disregarding the severity of the flaw. This clearly is very inefficient, especially in manufacturing plants or smart grids.

4. *Low delay/latency requirements in Industrial IoT*

   Many industrial control systems, healthcare applications, and vehicular applications often require end-to-end latencies between the sensor and thecontrol node to be within a few milliseconds. These requirements fall far outside what mainstream cloud services can achieve.

From the points mentioned above, it is reasonable to say that cloud computing is not preferred for certain IoT applications. Incorporating Fog Computing can be an effective

in these situations.

Features of Fog Computing

It is clear from above that the advantages of Fog Computing arise due to the fact that storage, computation, and communication of data occur close to the endpoints.

Some of the advantages of Fog Computing are as follows:

1. Privacy

   Fog computing can be used to control the extent of privacy. Any sensitive data of the user can be analyzed locally instead of sending them to a centralized cloud infrastructure. Through this way the team of IT will be able to track and control the respective device. Furthermore, if any subset of data needs to be analyzed it can be sent to the cloud.

2. Productivity

   If customer needs to make the machine function according to the way they want, they can utilize fog applications. These fog applications can be easily made by the developers with the right set of tools. After the development has taken place it can be deployed whenever they want.

3. Security

   Fog computing has the capability to connect multiple devices to the same network. Because of this the operations take place at various end points in a complex distributed environment rather than a centralized location. This makes it easier to identify potential threats before it effects the whole network.

4. Bandwidth

   The bandwidth required for transmitting data can be expensive depending upon the resources. Due to the fact that the selected data can be processed locally instead

of sending it to the cloud, there are very less number of bandwidth requirements. This bandwidth savings will be specially beneficial when increasing the number of IoT devices.

5. Latency

Another benefit of processing selected data locally is the latency savings. The data can be processed at the nearest data source geographically closer to the user. This can produce instant responses especially for the time sensitive services

Even though the implementation of fog computing can be very advantageous drawbacks do exist of using a fog-based system.

Some of the main drawbacks to consider are:

1. Complexity

Due to its complexity, the concept of Fog computing can be difficult to understand. There are many devices located at different locations storing and analyzing their own set of data. This could add more complexity to the network. In addition to that there are more sophisticated fog nodes present in a fog infrastructure.

2. Security

As mentioned earlier there are numerous devices and different fog nodes be present in a fog computing architecture. There are chances for these fog nodes to be in a less secure environment. Hackers can easily impose fake IP address in them gaining access to the respective fog node. Or else they increase the risk of corrupted files infiltrating the main data stream infecting both the device and the company. This makes them vulnerable to Man-in-the-middle attacks.

3. Authentication

Service offered by a fog computing is of large scale. The fog computing is comprised of end users, internet service providers and cloud providers. This can often rise trust and authentication issues in the fog.

4. Maintenance

Unlike cloud architecture, where maintenance is made seamless, it is not so in fog. Since controllers and storages are distributed across various locations in the network it needs more maintenance. The fog architecture is decentralized for processing.

5. Power Consumption

The number of fog nodes present in a fog environment is directly proportional to the energy consumption of them. Which means that these fog nodes require high amount of energy for them to function. As there are more fog nodes in a fog infrastructure there are more power consumption as well. Most companies often try to lower their cost using these fog nodes.

Fog Computing v/s Cloud Computing

As mentioned throughout the report fog computing isn't a like-for-like replacement for cloud computing. It is used in addition to the cloud, and stands out in three distinct ways: It can carry out a substantial amount of data storage, computing and control functions, and communication and networking at or near the end user, asopposed to performing these functions in remote data locations.

Control Functions include applications for end users and their devices, functions for controlling end-user systems such as smart grids, and services for supporting cloud-based applications, such as collecting and preprocessing data to be sent tothe cloud.

It helps in improving the performance and scalability of device to device(D2D) networks, control of radio access networks (RANs), and integrate local ad-hocnetworks with the infrastructure networks.

Fog computing and Cloud computing are interdependent, and heavily reliant oneach other. Cloud services can manage the fog, while fog can act as a proxy server, both for the cloud to deliver cloud services to endpoints, and for the endpoints to interact with the cloud.

Software for Simulating Fog Topologies

Although the concepts pertaining to fog computing are straight forward and intuitive, implementing a Fog Computing system is not a piece of cake. There are several variables that go into the design of the system, such as deciding the number of fog nodes in order to achieve the minimum latency and bandwidth used, selecting the sensors and actuators that will be a part of the system, deciding thestorage capacity and how powerful the nodes and the cloud will be, understanding the complexity arising from the large

number of participating objects and their interactions .In order to overcome these issues, researchers typically employ models and simulation tools to approximate the actual fog system. This way, if there were any miscalculations, it can be easily rectified.

Simulating on a software allows us to mimic the operation of real systems, and gives us the freedom to modify the inputs, model a number of characteristics or support the design of new systems. With reference to Fog Computing, simulators should support three crucial features: storage, computation and communication.

Although there is a wide range of simulation tools for cloud computing, they cannot be used as-is for studies in the field of fog computing; they have thus

been adapted to meet the new needs. At the same time, novel simulation toolshave been proposed and developed, specifically for the fog. Some of these tools include FogTorch, Edge-Fog, FogBus, iFogSim, etc.

The most popular however, is iFogSim, and this is the simulator used in the project.

iFogSim

The metric focused in iFogSim are energy consumption, operationalcosts, and network congestion. The architecture of iFogSim provides physical, and logical components. Physical components include fog devices, actuators, and sensors, while the logical components represent processing modules and their interaction as a directed graph.

The user can draw physical elements, define their characteristics and build theirtopology using a user-friendly GUI. Alternatively, the user can define topologies programmatically using Java APIs.

However, it has a few serious drawbacks:

(i) It focuses primarily on resource management, ignoring other important fog computing facets like infrastructure and cost.

(ii) It supports only tree-like topologies, and does not considerQoS (Quality of Service) requirements.

Fog Computing in the HealthCare Industry

The requirement for medical assistance has been increasing drastically especially after the emergence of Covid-19 a couple years ago. With a lot of patients coming in and a lesser ratio of doctors present, there is a need to manage patient data in an organized manner and notify the doctor as and when required. Throughout the report, I will be focusing on how IoT complemented with Fog proves to be a major boon in the healthcare industry, particularly healthcare monitoring - by reducing doctor-patient overload and ensuring that in case of emergencies, ambulances are dispatched to the patients with minimal delays.

Over the past decade, IoT has significantly improved daily tasks in the healthcare industry like heartrate monitoring, surgeries etc. It plays a major role in healthcare as IoT devices can be tagged with sensors and used for real-time remote monitoring of patients.

Wireless sensors in the form of wireless wearable accessories or devices are attached to a patient such that this information can be used for the monitoring process. The wireless sensor networks generate a huge amount of data that have been collected from all the devices connected to the network may be useful as well as redundant. All these unprecedented amounts of data can overwhelm the data storage systems and the data analysis applications.

In a typical IoT-Cloud model, the IoT layer would generate a bulk of data that can be refined using the cloud. However, the main issue arises for instantaneous monitoring, when significant delays are caused when the patient data recorded by the sensors is to be transferred to the cloud layer. In a typical healthcare setting, we have emergency response systems that require real-time operations in which efficiency and time play a major part. This may suffer due the delays caused by the cloud and hence transfer of such immense amounts of data back and forth is not an efficient option not only due to latency issues but also due to security.

With Fog computing, we bring the resources closer to the users thus decreasing the latency and thereby increasing the safety measure. Getting quicker results implies fast actions for critical heart patients.

Fog Architecture for Monitoring of Patients

The fog architecture that I have simulated is a typical 3-layered architecture comprising a sensor layer, fog layer and a cloud layer. Additionally, there is a proxy server between the fog layer and the cloud layer which provides a gateway between the patient data and the internet.

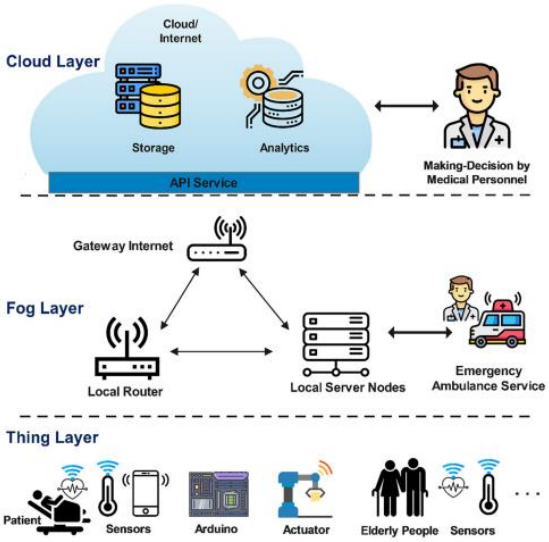
The Tracking Device/Sensor Layer: Patients (present in hospital or at home) can wear a strap on the upper arm which tracks and provides live updates on information such as blood pressure, heart rate, temperature etc over the Internet. All of the data are then sent to individual fog nodes over a network. The patient can also enter data into his or her smart phone, and these data will then be made available for processing.

The Fog Layer: In this layer, data pertaining to each individual in the network is extracted and analysed from the respective IoT tracking devices in the fog nodes. Fog nodes here refer to the different hospitals in a city. In addition to the classic fog layer, one can also include a layer containing super fog nodes which in this case would represent a group of hospitals in a city or state.

Now in case any of the patient-related values cross a certain threshold, a red-alert or high priority message (indicated by an alarm/LED from the sensor layer) can be sent over in real time to the respective hospital. Ambulances will be linked to the fog layer in order to quickly respond to an abnormal increase in a patient's health data.

The Cloud Layer: This is the top-most layer that manages the various actions that are to be performed by the health monitoring system. A component of the monitoring app runs on the sensors which enables the sensors to collect data and send it to the fog layer. In instances where a large amount of generalised data needs to be relayed over the network, the cloud layer is used. It can be used to prepare and execute activities that the fog layer fails to do. The patient data which is unlikely to be updated on a real time basis is stored over the cloud which can be accessed in the future whenever need be.

This can be better explained through a diagram:



Code Explanation

Fog nodes, sensors and actuators are dynamically allocated a list depending on user input.

I have created two variables-

(i) numOfHospitals- indicates the number of fog nodes, in our case the number of hospitals in a city

(ii) numOfPatientsPerHospital- indicates the number of patients per hospital that are transmitting live real-time data via a sensor.

The number of fog nodes or hospitals that I would be taking is 10 which remains constant throughout.

I would be taking a different number of patients per hospital ranging from 20 patients per hospital all the way up to 100 to illustrate the latency and amount of network used as the number of patients increase.

To show how vital the fog layer is in the healthcare monitoring model, I have taken two cases-

(i) Without a fog layer or initializing the Boolean cloud variable to FALSE

(ii) With a fog layer or initializing the Boolean cloud variable to TRUE

Since the sensors attached to the patient would generate continuous real-time data, we fix an instance of the module 'patient_data' to each patient.

Similarly, the fog node checks if the real-time data crosses a certain threshold and an instance of the module 'check_for_emergency' is fixed to each fog device.

After this the various devices in the fog architecture- comprising cloud, proxy server, individual fog nodes and sensors- are created.

Each device had 8 different parameters- CPU length in million instructions per second, RAM in megabytes, Uplink Bandwidth and Downlink Bandwidth in megabytes, Level, Rate per Mips, Busy Power in watts, Idle Power in watts

In iFogSim, these parameters have a fixed set of values:

	CPU Length	RAM	Uplink Bandwidth	Downlink Bandwidth	Level	Rate Per Mips	Busy Power	Idle Power
Cloud	44800	40000	100	10000	0	0.01	1648	1332
Proxy	2800	4000	10000	10000	1	0	107.339	83.43
Fog	2800	4000	10000	10000	2	0	107.339	83.43
Device	500	1000	10000	10000	3	0	87.53	82.44

Observations and Results from Test Cases:

Fog computing is implemented in the healthcare industry in hopes of reducing both latency and network usage so that data is transmitted with shorter delays and use up lesser bandwidth.

As mentioned earlier, I have taken the number of hospitals or fog nodes as 10. The number of patients in a hospital increase by 20 in each case.

Number of Patients per Hospital	Latency of Fog in ms	Latency of Cloud in ms
20	10.24	13.98
40	12.51	19.33
60	14.86	73.23
80	17.23	968.43
100	19.12	3942.68

Figure 1

rom figure-1, we can conclude that the 3-layer fog model easily surpasses the cloud model in terms of latency as the number of patients per hospital approach 60.

Number of Patients per Hospital	Fog Network Usage(kB)	Cloud Network Usage(kB)
20	5545	57823
40	10045	96349
60	14545	133751
80	19045	149982
100	23545	161354

Figure 2

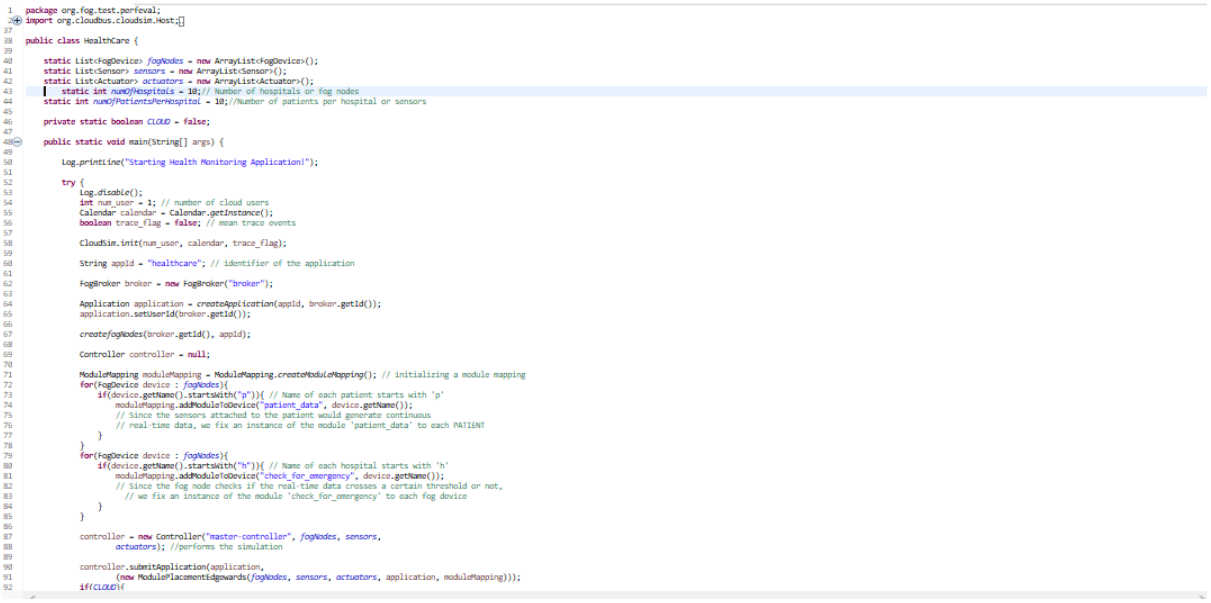
Similarly, from figure-2 we can conclude that the fog model once again beats the cloud model by a factor of almost 10 in terms of network usage.

Conclusion

Fog Computing as a concept is still in its early stages. Most of the results achieved from using fog computing work in theory, but in a practical world there are many more factors to consider. Healthcare monitoring as a service is a huge project. In this paper I have only focused on the healthcare aspects for heart patients by proposing a possible fog model that reduces latency and network usage. As shown earlier above, if built properly, the fog computing architecture can prove to be a major boon to healthcare (especially emergencies) by significantly reducing time overhead. The research area is a very promising one and when implemented will prove to be a very useful technology.

Screenshots of Code

The code given below was implemented on iFogSim on the Eclipse workspace. Link to my GitHub which contains code- https://github.com/sid050101/FogComputingHealthcare



```
package org.fog.test.perfeval;
import org.cloudbus.cloudsim.Host;

public class HealthCare {

    static List<FogDevice> fogNodes = new ArrayList<FogDevice>();
    static List<Sensor> sensors = new ArrayList<Sensor>();
    static List<Actuator> actuators = new ArrayList<Actuator>();
    static int numOfHospitals = 10;// Number of hospitals or fog nodes
    static int numOfPatientsPerHospital = 10;//Number of patients per hospital or sensors

    private static boolean CLOUD = false;

    public static void main(String[] args) {

        Log.printLine("Starting Health Monitoring Application!");

        try {
            Log.disable();
            int num_user = 1; // number of cloud users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false; // mean trace events

            CloudSim.init(num_user, calendar, trace_flag);

            String appId = "healthcare"; // identifier of the application

            FogBroker broker = new FogBroker("broker");

            Application application = createApplication(appId, broker.getId());
            application.setUserId(broker.getId());

            createFogNodes(broker.getId(), appId);

            Controller controller = null;

            ModuleMapping moduleMapping = ModuleMapping.createModuleMapping(); // initializing a module mapping
            for(FogDevice device : fogNodes){
                if(device.getName().startsWith("p")){ // Name of each patient starts with 'p'
                    moduleMapping.addModuleToDevice("patient_data", device.getName());
                    // Since the sensors attached to the patient would generate continuous
                    // real-time data, we fix an instance of the module 'patient_data' to each PATIENT
                }
            }
            for(FogDevice device : fogNodes){
                if(device.getName().startsWith("h")){ // Name of each hospital starts with 'h'
                    moduleMapping.addModuleToDevice("check_for_emergency", device.getName());
                    // Since the fog node checks if the real-time data crosses a certain threshold or not,
                     // we fix an instance of the module 'check_for_emergency' to each fog device
                }
            }

            controller = new Controller("master-controller", fogNodes, sensors,
                    actuators); //performs the simulation

            controller.submitApplication(application,
                    (new ModulePlacementEdgewards(fogNodes, sensors, actuators, application, moduleMapping)));
            if(CLOUD){
```

```
                    (new ModulePlacementEdgewards(fogNodes, sensors, actuators, application, moduleMapping)));
            if(CLOUD){
                // if the mode of deployment is cloud-based
                moduleMapping.addModuleToDevice("patient_data", "cloud"); // placing all instances of "patient_data module" in the Cloud
                moduleMapping.addModuleToDevice("check_for_emergency", "cloud"); // placing all instances of "check_for_emergency" module in the Cloud
            }

            controller = new Controller("master-controller", fogNodes, sensors, actuators);

            controller.submitApplication(application,(CLOUD)?(new ModulePlacementMapping(fogNodes, application, moduleMapping))
                            :(new ModulePlacementEdgewards(fogNodes, sensors, actuators, application, moduleMapping)));

            TimeKeeper.getInstance().setSimulationStartTime(Calendar.getInstance().getTimeInMillis());

            CloudSim.startSimulation();

            CloudSim.stopSimulation();

            }
        catch (Exception e) {
            e.printStackTrace();
            Log.printLine("An error occurred!");
        }
    }

    /**
     * Creates the fog devices in the physical topology of the simulation.
     * @param userId
     * @param appId
     */
    private static void createFogNodes(int userId, String appId) {
        FogDevice cloud = createFogDevice("cloud", 44800, 40000, 100, 10000, 0, 0.01, 1648, 1332);
        cloud.setParentId(-1);
        fogNodes.add(cloud);
        FogDevice proxy = createFogDevice("proxy_server", 2800, 4000, 10000, 10000, 1, 0.0, 107.339, 83.4333);
        proxy.setParentId(cloud.getId());
        proxy.setUplinkLatency(100); // connection latency b/w proxy server and cloud
        fogNodes.add(proxy);
        for(int i=0;i<numOfHospitals;i++){
            addHospital(i+"", userId, appId, proxy.getId());
        }
    }

    private static FogDevice addHospital(String id, int userId, String appId, int parentId){
        FogDevice router = createFogDevice("d-"+id, 2800, 4000, 10000, 10000, 2, 0.0, 107.339, 83.4333);
        fogNodes.add(router);
        router.setUplinkLatency(2); // latency of connection between router and proxy server is 2 ms
        for(int i=0;i<numOfPatientsPerHospital;i++){
            String mobileId = id+"-"+i;
            FogDevice Patient = addPatient(mobileId, userId, appId, router.getId()); // adding a smart Patient to the physical topology. Smart Patients have been modeled as fog devices as well.
            Patient.setUplinkLatency(2); // latency of connection between Patient and router is 2 ms
            fogNodes.add(Patient);
        }
        router.setParentId(parentId);
        return router;
    }

    private static FogDevice addPatient(String id, int userId, String appId, int parentId){
```

```
    private static FogDevice addPatient(String id, int userId, String appId, int parentId){
        FogDevice Patient = createFogDevice("m-"+id, 500, 1000, 10000, 10000, 3, 0, 87.53, 82.44);
        Patient.setParentId(parentId);
        Sensor sensor = new Sensor("s-"+id, "PATIENT", userId, appId, new DeterministicDistribution(5)); // inter-transmission time of Patient (sensor) follows a deterministic distribution
        sensors.add(sensor);
        Actuator ptz = new Actuator("ptz-"+id, userId, appId, "PTZ_CONTROL");
        actuators.add(ptz);
        sensor.setGatewayDeviceId(Patient.getId());
        sensor.setLatency(1.0);  // latency of connection between each Patient (sensor) and the parent Smart Patient is 1 ms
        ptz.setGatewayDeviceId(Patient.getId());
        ptz.setLatency(1.0);  // latency of connection between PTZ Control and the parent Smart Patient is 1 ms
        return Patient;
    }

    /**
     * Creates a vanilla fog device
     * @param nodeName name of the device to be used in simulation
     * @param mips MIPS
     * @param ram RAM
     * @param upBw uplink bandwidth
     * @param downBw downlink bandwidth
     * @param level hierarchy level of the device
     * @param ratePerMips cost rate per MIPS used
     * @param busyPower
     * @param idlePower
     * @return
     */
    private static FogDevice createFogDevice(String nodeName, long mips,
            int ram, long upBw, long downBw, int level, double ratePerMips, double busyPower, double idlePower) {

        List<Pe> peList = new ArrayList<Pe>();

        // 3. Create PEs and add these into a list.
        peList.add(new Pe(0, new PeProvisionerOverbooking(mips))); // need to store Pe id and MIPS Rating

        int hostId = FogUtils.generateEntityId();
        long storage = 1000000; // host storage
        int bw = 10000;

        PowerHost host = new PowerHost(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerOverbooking(bw),
                storage,
                peList,
                new StreamOperatorScheduler(peList),
                new FogLinearPowerModel(busyPower, idlePower)
            );

        List<Host> hostList = new ArrayList<Host>();
        hostList.add(host);

        String arch = "x86"; // system architecture
        String os = "Linux"; // operating system
        String vmm = "Xen";
        double time_zone = 10.0; // time zone this resource located
        double cost = 3.0; // the cost of using processing in this resource
```

```
        double cost = 3.0; // the cost of using processing in this resource
        double costPerMem = 0.05; // the cost of using memory in this resource
        double costPerStorage = 0.001; // the cost of using storage in this
                                       // resource
        double costPerBw = 0.0; // the cost of using bw in this resource
        LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not adding SAN
                                            // devices by now

        FogDeviceCharacteristics characteristics = new FogDeviceCharacteristics(
                arch, os, vmm, host, time_zone, cost, costPerMem,
                costPerStorage, costPerBw);

        FogDevice fogdevice = null;
        try {
            fogdevice = new FogDevice(nodeName, characteristics,
                    new AppModuleAllocationPolicy(hostList), storageList, 10, upBw, downBw, 0, ratePerMips);
        } catch (Exception e) {
            e.printStackTrace();
        }

        fogdevice.setLevel(level);
        return fogdevice;
    }

    /**
     * Function to create the Intelligent Surveillance application in the DDF model.
     * @param appId unique identifier of the application
     * @param userId identifier of the user of the application
     * @return
     */
    @SuppressWarnings({"serial" })
    private static Application createApplication(String appId, int userId){

        Application application = Application.createApplication(appId, userId);
        /*
         * Adding modules (vertices) to the application model (directed graph)
         */
        application.addAppModule("patient_data", 10);
        application.addAppModule("check_for_emergency", 10);
        application.addAppEdge("PATIENT", "patient_data", 1000, 500, "PATIENT", Tuple.UP, AppEdge.SENSOR); // adding edge from PATIENT to take_picture module
        application.addAppEdge("patient_data", "check_for_emergency", 1000, 500, "slots",Tuple.UP, AppEdge.MODULE);

        application.addAppEdge("check_for_emergency", "PTZ_CONTROL", 100, 28, 100, "PTZ_PARAMS", Tuple.DOWN, AppEdge.ACTUATOR);
        application.addTupleMapping("patient_data", "PATIENT", "slots",
        new FractionalSelectivity(1.0));
        application.addTupleMapping("check_for_emergency", "slots",
        "PTZ_PARAMS", new FractionalSelectivity(1.0));
        final AppLoop loop = new AppLoop(new ArrayList<String>()
        {{add("PATIENT");
        add("patient_data");add("check_for_emergency");
        add("PTZ_CONTROL");}});
        List<AppLoop> loops = new ArrayList<AppLoop>(){{add(loop);}};
        application.setLoops(loops);
        return application;
    }


}
```

References

1. 1 Qadri YA, Nauman A, Zikria YB, Vasilakos AV, Kim SW (2020) The future of healthcare internet of things: a survey of emerging technologies. IEEE Commun Surv Tutor 22(2):1121–1167. https://doi.org/10.1109/COMST.2020.2973314

2. Buke A, Gaoli F, Yongcai W, Lei S, Zhiqi Y (2015) Healthcare algorithms by wearable inertial sensors: a survey. China Commun 12(4):1–12. https://doi.org/10.1109/CC.2015.7114054

3. Kumar A, Krishnamurthi R, Nayyar A, Sharma K, Grover V, Hossain E (2020) A novel smart healthcare design, simulation, and implementation using healthcare 4.0 processes. IEEE Access 8:118433–118471. https://doi.org/10.1109/

ACCESS.2020.3004790

4. Constant, N., Borthakur, D., Abtahi, M., Dubey, H., & Mankodiya, K. (2017). FogAssisted wIoT: A Smart Fog Gateway for End-toEnd Analytics in Wearable Internet ofThings. ArXiv, abs/1701.08680.

5. N. Patii and B. Iyer, "Health monitoring and tracking system for soldiers using Internet of Things(IoT)," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 2017, pp. 1347-1352, doi:10.1109/CCAA.2017.8230007.

6. Xie, Qingsong & Wang, Guoxing & Peng, Zhengchun & Lian, Yong. (2018). Machine Learning Methods for Real-Time Blood Pressure Measurement Based on Photoplethysmography. 1-5. 10.1109/ICDSP.2018.8631690.

7. X.-Q. Pham and E.-N. Huh, “Towards task scheduling in a cloud-fog computing system,” in 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–4, Kanazawa, Japan, October 2016.

8. T. Benson, “Why general practitioners use computers and hospital doctors do not—Part 1: incentives,” BMJ, vol. 325, no. 7372, pp. 1086–1089, 2002.

9. H. Gupta, A. Dastjerdi, S. Ghosh, and A. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments,” 2016, http://arxiv.org/abs/1606.02007

10. Sukhpal Singh Gill, Rajesh Chand Arya, Gurpreet Singh Wander, Rajkumar Buyya, Fog-based smart healthcare as a Big Data and cloud service for heart patients using IoT, in: International Conference on Intelligent Data Communication Technologies and Internet of Things, Springer, Cham, 2018, pp. 1376–1383.

11. S. He, B. Cheng, H. Wang, Y. Huang, J. Chen, Proactive personalized services through fog-cloud computing in large-scale IoT-based healthcare application, China Commun. 14 (11) (2017) 1–16.

12. Shreshth Tuli, Redowan Mahmud, Shikhar Tuli, Rajkumar Buyya, FogBus: A blockchain-based lightweight framework for edge and fog computing, J. Syst. Softw. 154 (2019) 22–36.

13. El-Sayed H et al (2018) Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing

environment. IEEE Access 6:1706–1717. https://doi.org/10.1109/ACCESS.2017.2780087

14. Wang H et al (2020) Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. IEEE Commun Surv Tutor 22(4):2349–2377. https://doi.org/10.1109/COMST.2020.3020854