CSE3005: Fundamentals of Fog and Edge Computing

J-Component Final Report

Prof. Umadevi K S

Slot – G1

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Title: IMPLEMENTING FOG-BASED MODELS FOR IOT APPLICATIONS

Team Members

19BCT0221 - PARTH MAHESHWARI – parth.maheshwari2019@vitstudent.ac.in
19BCT0204 – PRAMIT GUPTA – pramit.gupta2019@vitstudent.ac.in
19BCT0217 – AGRIM NAGRANI – agrim.nagrani2019@vitstudent.ac.in

TABLE OF CONTENTS

Abstract:

The Internet of Things (IoT) is a new paradigm that has transformed traditional lifestyles into high-tech ones. Smart cities, smart homes, pollution management, energy conservation, smart transportation, and smart industries are examples of IoT-driven developments. Many important research studies and investigations have been conducted in order to improve technology via IoT. However, in order to realise the full potential of IoT, a number of obstacles and issues must be addressed. Some of these includes performance related issues, including reduced bandwidth, higher latency, and security related vulnerabilities. This is where Fog and Edge Computing can play an important role.

In the coming future, fog computing will rule the industry of Internet of Things. Fog's dominance will be fuelled by an ability to collect data closer to the source (the user device). The capacity to process data locally is more critical than in the past. As the Internet of Things grows in popularity, more and more devices are being added to the network. For data transmission and reception, each device is wirelessly connected. IOT is known for the big data analysis that its applications come with. The 5 v's in big data analytics namely: volume, velocity, variety, veracity, and value can be handled in a better fashion with the help of Fog and Edge Computing. Fog will significantly reduce energy consumption, minimizes time complexity and memory requirements, and maximizes this data's utility and performance. This in turn helps us to reduce the latency and boost the bandwidth in communication perspective. In this report we plan on checking the feasibility/deployment of Fog models on various IOT applications using 'iFogSim' and measure important metrics including energy consumption, latency and Total network usage.

Introduction:

What is Fog and Edge Computing?

The term 'fog' refers to a cloud that is close to the ground, exactly as fog concentrates around the network's edge. Fog computing is a decentralised computing system in which data, storage, processing, and applications are distributed between the data source and the cloud or near the extreme nodes. Fog computing or edge computing, brings the cloud's benefits and power closer to where data is created and acted upon. Because both involve moving intelligence and processing closer to where the data is created, the words fog computing and edge computing are often used interchangeably.

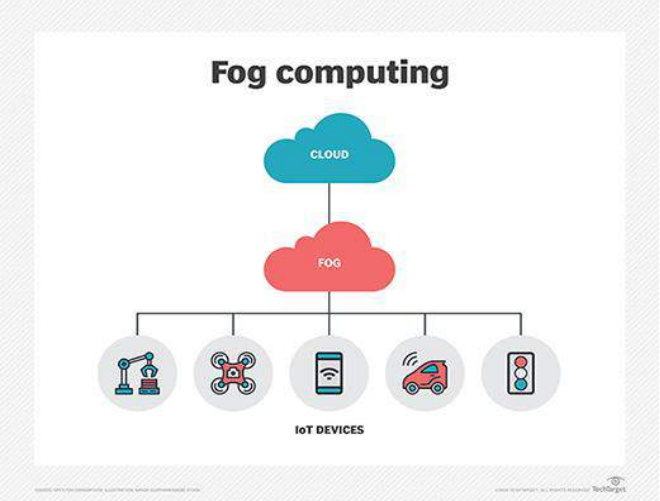


Fig: Fog Computing in IOT.

Fog computing is a type of decentralised computing infrastructure in which data, compute, storage, and other applications are distributed between the data source and the cloud. Cloud computing refers to providing these services at a much distant location. However, the processing capacity of Cloud resources are very powerful. Note that edge computing is a distributed information technology (IT) architecture in which client data is processed as close to the original source as possible at the network's perimeter.

Cloud and Fog Computing:
a Comparison Chart

	CLOUD	FOG
Architecture	Centralized	Distributed
Communication with devices	From a distance	Directly from the edge
Data processing	Far from the source of information	Close to the source of information
Computing capabilities	Higher	Lower
Number of nodes	Few	Very large
Analysis	Long-term	Short-term
Latency	High	Low
Connectivity	Internet	Various protocols and standards
Security	Lower	Higher

Fig: Cloud vs Fog Computing.

Pros and Cons of Fog Architecture?

Some of the parameters that we expect to optimize while using this technology are latency reduction, improved response time, better compliance, increased security, greater data privacy, reduced cost of bandwidth, overall increase in speed and efficiency, greater up time of critical systems and enhanced services for remote locations. Although the above-mentioned factors are some of the critical parameters/features of IOT applications. Note that although it enhances performance factors in applications, it also has some cons. Some of them includes, increased design complexity, physical security issues and decentralisation of the overall system. With the increase in IOT technology, we plan on checking the feasibility of the Fog architecture in various IOT applications, including Smart Parking, Smart Waste Management, Smart Mining and UAV based sensing services. These are some of the most trending topics that have been under a lot of research in last 10 years.

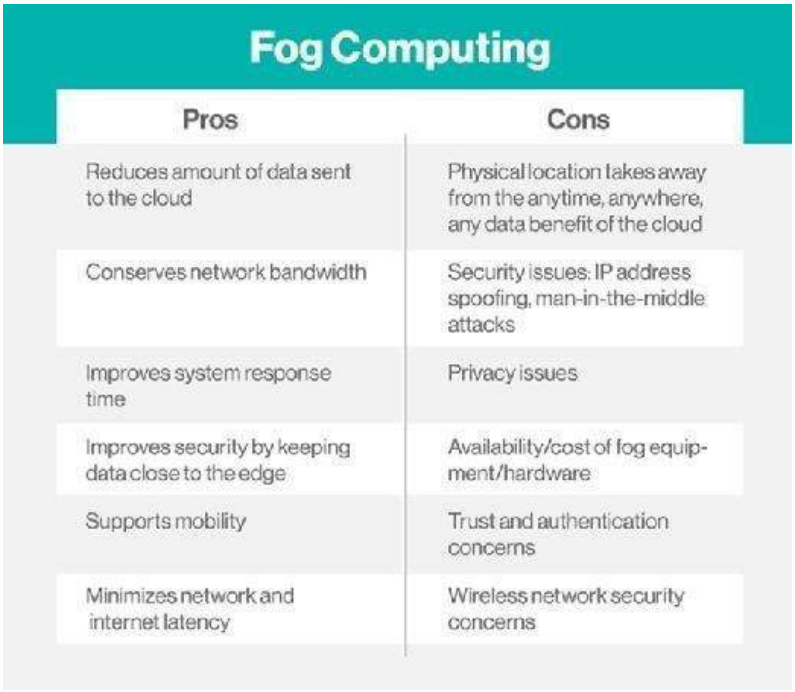


Fig: Pros and Cons of Fog Architecture.

iFogSim Simulator?

We plan on using ‘iFogSim’ as our simulation tool for analysing our assumption and our proposed work. ‘iFogSim’ is a Java based open-source simulation tool for simulating fog computing scenarios. ‘iFogSim’ enables the modelling and simulation of fog computing to evaluate resource management and scheduling policies across edge and cloud resources under different scenarios. This is a high-performance open-source fog computing, edge computing, and IoT toolkit for modelling and simulating edge computing, Internet of Things,

and fog computing networks. iFogSim incorporates resource management approaches that can be tailored to the research field. CloudSim, a widely used framework for simulating cloud-based settings and resource management, is used in conjunction with iFogSim. iFogSim is used by the CloudSim layer to handle events between the fog computing components.

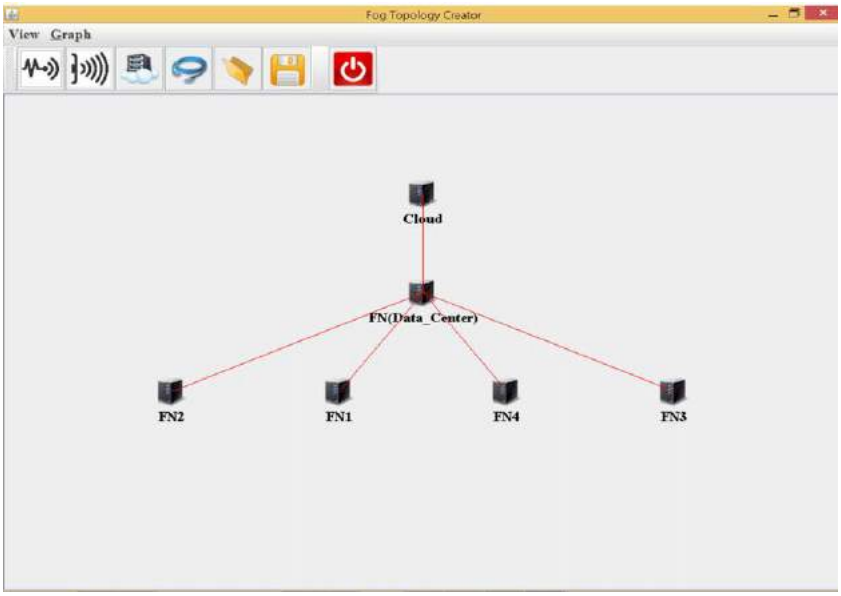


Fig: iFogSim Simulator.

Dividing IOT implementations:

IOT (Internet of Things) is a vast field and it's derivative applications is applicable in different kinds of environment which include Smart Homes, Smart City, Self-Driven Cars, IoT Retail Shops, Farming, Smart Parking, Industrial Internet, Traffic Management, Water and Waste Management. Note that each of the above application has it's own unique functionality, and architecture to tackle the challenges associated with it. Fog/Edge can play a vital role for enhancing the performance of these applications. Deployment of fog nodes in a certain fashion can be helpful for reducing latency and help providing better device integration.Thus, it is important for us analyse the required architecture/model for implementing fog/edge computing in IoT applications

PROPERTIES	MODEL-1	MODEL-2	MODEL-3	MODEL-4
Linerality	Linear	Non-Linear	Non-Linear	Linear
Heterogenous Nodes	Yes	No	Yes	Yes/No
Modules	Multiple	Master+Multiple	Master+Mulitiple+Associated	Mulitple
Cloud/External Storage	Not Avaliable	Avaliable	Avaliable	Avaliable
Complexity	Minimum	Medium	Very High	High

Table: Comparison of various models.

Literature Review:

Review: 1

Title: Smart parking systems: comprehensive review based on various aspects.

Authors: AbrarFahima, MehediHasanbMuhtasim and AlamChowdhurya.

Summary: Parking allocation has become a serious issue in modern cities, prompting the development of several smart parking systems (SPS). The goal of this research is to provide a thorough examination, comparison, and analysis of SPSs in terms of technological approach, sensors used, networking technologies, user interface, computational methodologies, and service provided. Furthermore, the work addresses a research gap by demonstrating the appropriateness of SPSs in diverse environmental circumstances and highlighting their benefits and drawbacks. Researchers, designers, and policymakers would be able to determine the best-suited SPS and understand current trends in the sector by comparing numerous elements of SPSs.

Review: 2

Title: IoT based Smart Mine Monitoring System

Authors: C. Shobana Nageswari1, C.G. Sangeetha2 and V.B. Yogambigai

Summary: Mining is one of the most hazardous professions on the planet. Underground miners in some nations lack safety and social protection, and in the event of injury, they may be left to fend for themselves. There are also negative societal consequences, such as

displacement and lost income. Among all industries, the mining industry has the greatest rate of occupational mortality. Rock falls, fires, explosions, methane intoxication, and electrocution are all common causes of occupational mortality. There are numerous case studies on underground mines, with one recent case study in China revealing that underground mining is the world's deadliest business. We have developed a superior communication technology that must be used for an intelligent sensing and warning system to overcome all of these tragedies.

Review: 3

Title: A Literature Survey of Unmanned Aerial Vehicle Usage for Civil Applications

Authors: Mithra Sivakumar1, Naga Malleswari

Summary: The technology of unmanned vehicles/systems (UVs/USs) has surged in recent years. Unmanned vehicles are used in the air, on the ground, and in the water. On/in the sea, or on/in the ground. Unmanned vehicles, rather than human systems, play a larger role in several civil application fields such as remote sensing, surveillance, precision agriculture, and rescue operations. Unmanned vehicles outperform manned vehicles. In terms of mission safety and operational costs, manned systems are superior to unmanned systems. In the civil sector, unmanned aerial vehicles (UAVs) are widely used. Low maintenance costs, ease of deployment, hovering capabilities, and high mobility make them ideal for infrastructure. The unmanned aerial vehicles (UAVs) can collect photos more quickly and accurately than satellite photography, allowing for speedier review. Research is also included. UAVs with artificial intelligence: trends, major civil concerns, and future insights (smart AI). In addition, this paper. This article compares and contrasts the features of numerous drone models and simulators. Precision agriculture, according to the literature review, is one of the smart UAV's civic applications Unmanned aerial vehicles help in weed detection, agricultural management, and pest control. Researchers will be able to use drones to identify plant diseases and other difficulties in the future, opening the way for future drone applications.

Review: 4

Title: Smart garbage monitoring and clearance system using internet of things

Authors: T. Senthilkumaran, Mahantesh Mathapati

Summary: The rise in population has resulted in a significant deterioration in the quality of hygiene in the waste management system. The spillover of rubbish in civic places pollutes the environment in the surrounding areas. It has the potential to exacerbate a variety of

serious ailments in the surrounding area. The impacted area's appraisal will be humiliated as a result of this. It takes a "smartness-based waste management system" to eliminate or mitigate garbage and preserve cleanliness. This study proposes an IoT-based smart waste clean management system that uses sensor systems to monitor the waste level in the dustbins. As soon as it was recognised, the system was changed to concern permitted via GSM/GPRS. Microcontroller was employed as an interface between the sensor system and the GSM/GPRS system in this system.Microcontroller was employed as an interface between the sensor system and the GSM/GPRS system in this system. An android application is being built to monitor and combine the needed information, which is related to the varied levels of garbage in various areas. As a result, the environment has become greener, and support for Swachh Bharat for cleanliness has grown.

Review: 5

Title: Edge and fog computing for IoT: A survey on current research activities & future directions

Authors: Ali Cherif Moussa, Hassine Moungla.

Summary: The Internet of Items (IoT) is a network that permits communication between devices, things, and any digital assets that transmit and receive data without requiring human contact. The key feature of IoT is the massive amount of data generated by end-user devices that must be processed in the cloud in a short amount of time. The existing cloud computing approach is inefficient for analysing massive amounts of data in a short amount of time while still meeting the needs of consumers. The cloud will take a long time to analyse such a large amount of data, lowering the quality of service (QoS) and significantly impacting IoT applications and overall network performance. To address these issues, a new architecture known as edge computing - which allows for the decentralisation of data - has been developed.The Internet of Things (IoT) is a network that allows devices, things, and other digital assets to communicate with one another without requiring human interaction. The large volume of data created by end-user devices that must be handled in the cloud in a short amount of time is a crucial element of IoT. The current cloud computing model is inefficient for analysing large amounts of data in a short period of time while yet meeting consumer demands. The cloud will take a long time to process such a vast amount of data, decreasing QoS and having a severe impact on IoT applications and overall network performance. And thus fog and edge computing can be of great importance in this respect.

Proposed Method:

1. **Smart Parking System:**

The number of automobiles on the road has expanded dramatically as a result of the expanding population in cities, as personal vehicles is one of the most important modes of transportation. In densely populated locations, parking spaces have become a significant problem. People waste significant amount of time looking for an available car parking place, resulting in CO2 emissions, wasted time, and wasted gasoline. Parking issues have become more prevalent. In recent years, it has gotten a lot of attention, and several studies have been done on it. IoT-based automobile parking solutions have been presented by many researchers as a remedy. We tend to present a Fog-based architecture for such a large-scale implementation of Smart Parking System. Some of the components that we tend to include in the system are Cameras, LED Display, Fog Nodes, and Cloud.

The smart cameras are installed in the parking lanes, and they are responsible for taking the pictures of the parking lanes. An image processing technique has been constructed on the fog node to find vacant parking places. The parking slots information on the LED is refreshed after the vacant parking slots are detected. The data is temporarily stored and processed in the fog node before being transferred to the cloud server for the ultimate processing and storage functionalities. When the vehicle gets at the parking gate, the driver promptly locates a vacant automobile parking space with the help of the LED display and parks the vehicle in accordance with the empty space.

Model 1: This a linear, basic architectural model which involves a sensor sensing the environment and then sending the data for processing and when the data gets processed the results/tasks are send to the actuators

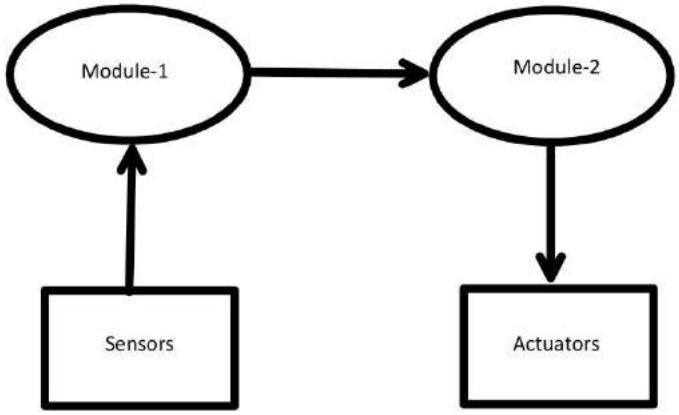


Fig: Model for Smart Parking.

2. Smart Waste Management System:

Waste management is an important issue that needs to have a concern in every country including India. As reported India generates 62 million tons of waste each year. About 43 million tons (70%) are collected of which about 12 million tons are treated, and 31 million tons are dumped in landfill sites. Waste management in India at this time, is still limited and manually, the officer will clean up at a specified time according to the schedule, this is very ineffective because the trash can have been fully before the garbage collection schedule, the delay of garbage collection will cause the garbage on the trash can overflow and smell. Waste volume produced by inefficient waste management would cause insects, bacteria and viruses multiply rapidly that can infect humans. The existing system has the limitations as time intense, trucks go and empty the containers, even they're empty. the price is high with insanitary setting. Even the dangerous odour causes the unhealthy setting. So planned model talks regarding the way to create use of the recent advancements in technology to form our place clean and tidy. Thus, nodes like embedded systems and ultrasonic sensors are placed at the top of the smart bins, in order to effectively share the status of the bin. The model that we are discussing here is for garbage monitoring system for a metropolitan city.

We plan on installing these ultrasonic sensors HC-05 on these bins, which would be responsible for collecting the information regarding the height of garbage in the bins. Thus, these sensors are responsible for sending this information for displaying, processing, and storing on the cloud. However, we plan on introducing a fog module in between the two of them. Note that we plan on creating separate

modules for all the necessary different kinds of users/departments, such as recycling unit, healthcare department, and the cleaning authority to disseminate the waste collection information among the relevant collection databases.

Model 2: This is a non-linear model which involves sending the processed information to higher order for propagation/storage devices.

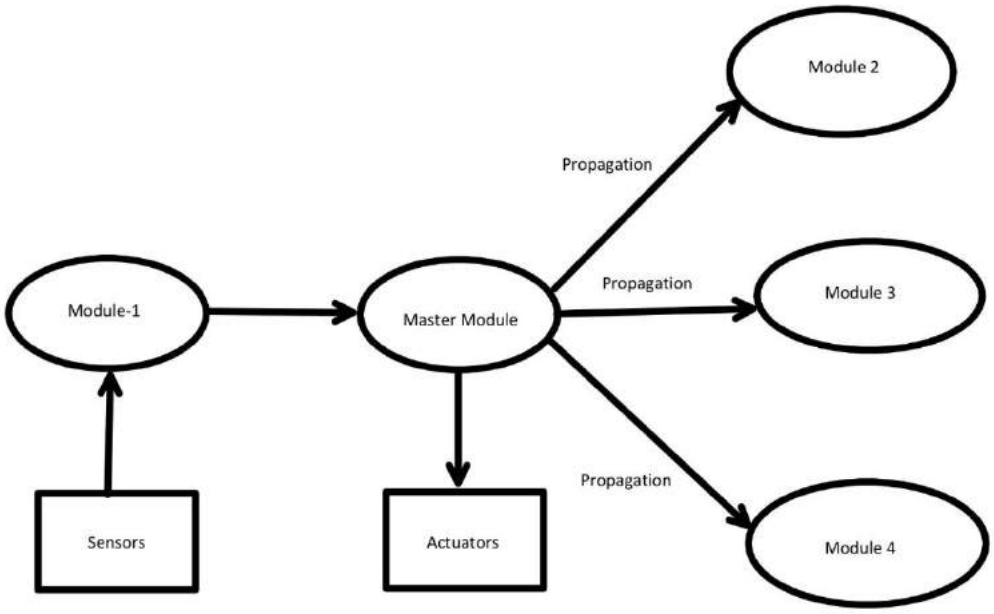


Fig: Model for Garbage Monitoring System.

3. Smart Mining:

Mining is one of the most extensive and difficult job. It is of great industrial significance, that what makes it necessary, commercial, and dangerous. It is one of the most essential and well-known businesses that necessitates a significant amount of data analysis. Despite its importance, the mining business is fraught with dangers. chemical reactions, hazardous gas emissions, asphyxia, and rock sliding are among the potential hazards to mining personnel's life. Respiratory disorders including pneumoconiosis, asbestosis, and silicosis are caused by inhaling small particles from enormous amounts of dust generated by mining activities like blasting and drilling. Mining results in biodiversity loss, significant use of water resources, erosion, sinkholes, deforestation, dammed rivers and ponded waters, wastewater disposal

issues, acid mine drainage, and contamination of soil, ground, and surface water all over the world, all of which can cause health problems in local populations/area. Also, let's not forget what negative impacts, mining operations cause to the workers/employees at the pit site. As a result, in order to tackle such hazardous situations, IOT can play an important role at the architecture point of view.

We plan on using many embedded systems and sensors including heterogeneous sensors to detect gases, chemicals, and surrounding data and to inform users. We plan on using chemical sensors, gas sensors and other surrounding sensors for getting the important details about the mining site. With these heterogenous sensors, having different functionalities each, it becomes difficult to manage the communication channels. Thus, fog computing can be of great use here, where pre-processing of these large and variety of data can help along with reduction of latency and better network bandwidth.

Model 3: This is a multi-sensor, non-linear architecture. It involves having multiple sub-modules over a supreme module to tackle multiple processing of the data received by the sensors.

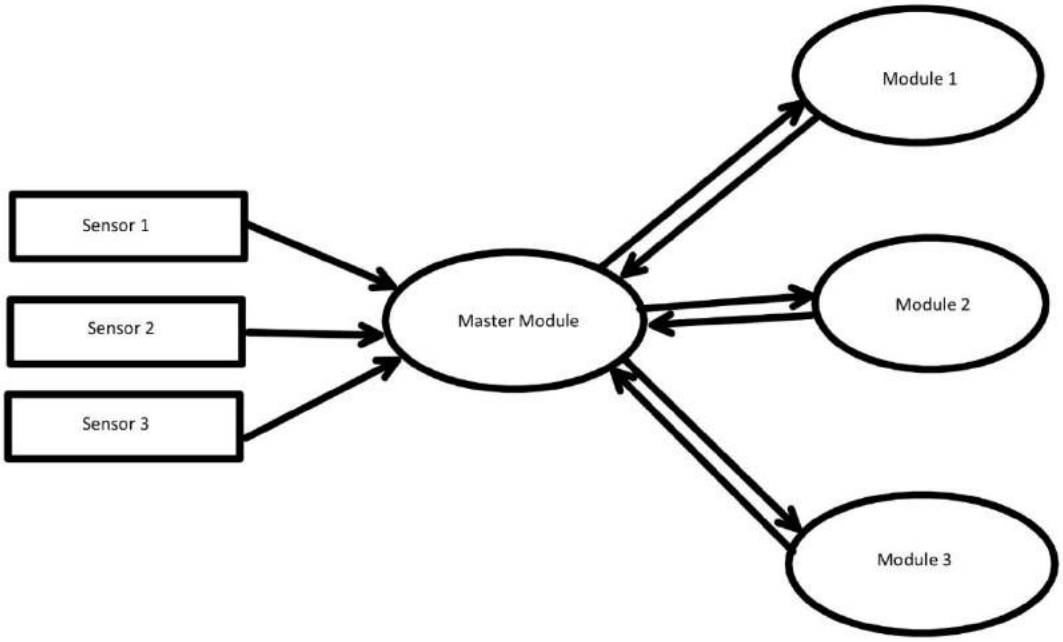


Fig: Smart Mining System.

4. UAV Sensing – Disaster Management:

UAVs refer to unnamed aerial vehicle. It is used for observation and tactical planning. This technology can now be used to aid crew members in emergency situations. UAVs are categorised according to their altitude range, endurance, and weight, and can be used for both military and commercial purposes. UAVs can be of very great help to conduct effective surveys in disaster-prone or physically inaccessible places, as well as quickly assess and control the damage caused by landslides, floods, and earthquakes or any other calamity to enable relief efforts. Nowadays, UAVs are also used for photography and videography purposes. It is important to note that we need a to transfer, process and store these data as fast as possible, whereas most of these nodes/devices are energy, processing and storage constrained. Thus, Fog and Edge computing can be of great significance as an intermediatory processing module between these vehicles and the cloud.

Here note that for these heterogenous/mobile nodes we need different modules for both the sensors and actuators. The data generated by the client is then processed and stored in accordance with the functionality. The fog nodes not only help to reduce the latency for these performance related tasks, but also can act as a temporary storage unit. Thus, improving the overall optimization operation.

Model 4: It is a linear model in which the processed data is further send to cloud for further use/storage.

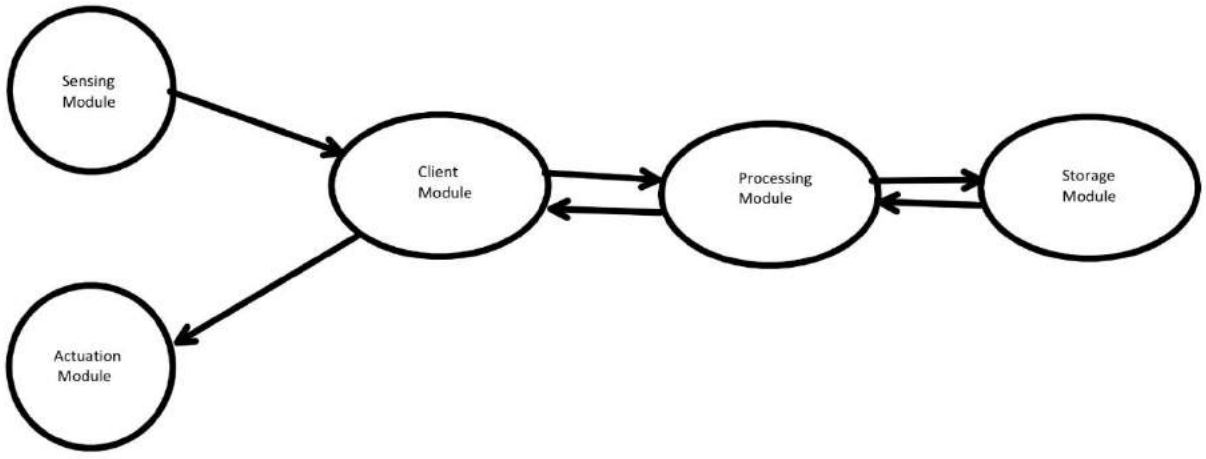


Fig: Sensing-as-a-Service.

Simulation/Code:

1.Smart Car Parking System:

Source Code:

```
package org.fog.test.perfeval;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.sdn.overbooking.BwProvisionerOverbooking;
import org.cloudbus.cloudsim.sdn.overbooking.PeProvisionerOverbooking;
import org.fog.application.AppEdge;
import org.fog.application.AppLoop;
import org.fog.application.Application;
import org.fog.application.selectivity.FractionalSelectivity;
import org.fog.entities.Actuator;
import org.fog.entities.FogBroker;
import org.fog.entities.FogDevice;
import org.fog.entities.FogDeviceCharacteristics;
import org.fog.entities.Sensor;
import org.fog.entities.Tuple;
import org.fog.placement.Controller;
import org.fog.placement.ModuleMapping;
import org.fog.placement.ModulePlacementEdgewards;
import org.fog.placement.ModulePlacementMapping;
import org.fog.policy.AppModuleAllocationPolicy;
import org.fog.scheduler.StreamOperatorScheduler;
import org.fog.utils.FogLinearPowerModel;
import org.fog.utils.FogUtils;
import org.fog.utils.TimeKeeper;
import org.fog.utils.distribution.DeterministicDistribution;
public class SmartMining {
	//Create the list of fog devices
	static List<FogDevice> fogDevices = new ArrayList<FogDevice>();
	//Create the list of sensors
	static List<Sensor> sensors = new ArrayList<Sensor>();
	//Create the list of actuators
	static List<Actuator> actuators = new ArrayList<Actuator>();
	//Define the number of fog nodes will be deployed
	static int numOfFogDevices = 2;
	//Define the number of gas sensors with each fog nodes
	static int numOfGasSensorsPerArea=10;
	//Define the number of chemical sensors with each fog nodes
	static int numOfChSensorsPerArea=10;
	//Define the number of surrounding sensors with each fog nodes
	static int numOfSrSensorsPerArea=10;
	//We are using the fog nodes to perform the operations.
	//cloud is set to false
	private static boolean CLOUD = true;
	public static void main(String[] args) {
```

```
		Log.printLine("Smart Mining Industry System...");
		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 = "mins"; // identifier of the application
			FogBroker broker = new FogBroker("broker");
			Application application = createApplication(appId, broker.getId());
			application.setUserId(broker.getId());
			createFogDevices(broker.getId(), appId);
			Controller controller = null;
			// initializing a module mapping
			ModuleMapping moduleMapping = ModuleMapping.createModuleMapping();
//			for(FogDevice device : fogDevices){
//				if(device.getName().startsWith("a")){
//					moduleMapping.addModuleToDevice("master-module",
device.getName());
//				}
//			}
			for(FogDevice device : fogDevices){
				if(device.getName().startsWith("g")){
					moduleMapping.addModuleToDevice("gasinfo-module",
device.getName());
				}
			}
			for(FogDevice device : fogDevices){
				if(device.getName().startsWith("c")){
					moduleMapping.addModuleToDevice("chinfo-module",
device.getName());
				}
			}
			for(FogDevice device : fogDevices){
				if(device.getName().startsWith("s")){
					moduleMapping.addModuleToDevice("srinfo-module",
device.getName());
				}
			}
			// if the mode of deployment is cloud-based
			if(CLOUD){
				// placing all instances of master-module in Cloud
				moduleMapping.addModuleToDevice("master-module", "cloud");
				moduleMapping.addModuleToDevice("gasinfo-module", "cloud");
				moduleMapping.addModuleToDevice("chinfo-module", "cloud");
				moduleMapping.addModuleToDevice("srinfo-module", "cloud");
			}

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

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


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

			CloudSim.startSimulation();
```

```
                CloudSim.stopSimulation();

                Log.printLine("mining industry simulation finished!");
            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }

        /**
         * Creates the fog devices in the physical topology of the simulation.
         * @param userId
         * @param appId
         */
        private static void createFogDevices(int userId, String appId) {
            FogDevice cloud = createFogDevice("cloud", 44800, 40000, 100, 10000, 0, 0.01, 16*103,
16*83.25);
            cloud.setParentId(-1);
            fogDevices.add(cloud);
            FogDevice router = createFogDevice("proxy-server", 7000, 4000, 10000, 10000, 1, 0.0,
107.339, 83.4333);
            router.setParentId(cloud.getId());
            router.setUplinkLatency(100);
            fogDevices.add(router);
            for(int i=0;i<numOfFogDevices;i++){
                addFogNode(i+"", userId, appId, router.getId());
            }
        }

        private static FogDevice addFogNode(String id, int userId, String appId, int parentId){
            FogDevice fognode = createFogDevice("a-"+id, 5000, 4000, 10000, 10000, 3, 0.0, 107.339,
83.4333);
            fogDevices.add( fognode);
            fognode.setUplinkLatency(2);
            for(int i=0;i<numOfGasSensorsPerArea;i++){
                String mobileId = id+"-"+i;
                FogDevice gasSensor = addGasSensors(mobileId, userId, appId, fognode.getId());
                gasSensor.setUplinkLatency(2);
                fogDevices.add(gasSensor);
            }
            for(int i=0;i<numOfChSensorsPerArea;i++){
                String mobileId = id+"-"+i;
                FogDevice chSensor = addChSensors(mobileId, userId, appId, fognode.getId());
                chSensor.setUplinkLatency(2);
                fogDevices.add(chSensor);
            }
            for(int i=0;i<numOfSrSensorsPerArea;i++){
                String mobileId = id+"-"+i;
                FogDevice srSensor = addSrSensors(mobileId, userId, appId, fognode.getId());
                srSensor.setUplinkLatency(2);
                fogDevices.add(srSensor);
            }
            fognode.setParentId(parentId);
            return fognode;
        }

        private static FogDevice  addGasSensors(String id, int userId, String appId, int parentId){
            FogDevice gasSensor = createFogDevice("g-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
```

```
                gasSensor.setParentId(parentId);
                Sensor sensor = new Sensor("s-"+id, "GAS", userId, appId, new
DeterministicDistribution(5));
                sensors.add(sensor);
                Actuator ptz = new Actuator("act-"+id, userId, appId, "ACT_CONTROL");
                actuators.add(ptz);
                sensor.setGatewayDeviceId(gasSensor.getId());
                sensor.setLatency(40.0);
                ptz.setGatewayDeviceId(parentId);
                ptz.setLatency(1.0);
                return gasSensor;
        }

        private static FogDevice addChSensors(String id, int userId, String appId, int parentId){
                FogDevice chSensor = createFogDevice("c-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
                chSensor.setParentId(parentId);
                Sensor sensor = new Sensor("sch-"+id, "CH", userId, appId, new
DeterministicDistribution(5));
                sensors.add(sensor);
                Actuator ptzch = new Actuator("actch-"+id, userId, appId, "ACT_CONTROLCH");
                actuators.add(ptzch);
                sensor.setGatewayDeviceId(chSensor.getId());
                sensor.setLatency(40.0);
                ptzch.setGatewayDeviceId(parentId);
                ptzch.setLatency(1.0);
                return chSensor;
        }

        private static FogDevice addSrSensors(String id, int userId, String appId, int parentId){
                FogDevice srSensor = createFogDevice("s-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
                srSensor.setParentId(parentId);
                Sensor sensor = new Sensor("ssr-"+id, "SR", userId, appId, new
DeterministicDistribution(5));
                sensors.add(sensor);
                Actuator ptzch = new Actuator("actsr-"+id, userId, appId, "ACT_CONTROLSR");
                actuators.add(ptzch);
                sensor.setGatewayDeviceId(srSensor.getId());
                sensor.setLatency(40.0);
                ptzch.setGatewayDeviceId(parentId);
                ptzch.setLatency(1.0);
                return srSensor;
        }

        /**
         * 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 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);
```

```
		application.addAppModule("gasinfo-module", 10);
		application.addAppModule("master-module", 10);
		application.addAppModule("chinfo-module", 10);
		application.addAppModule("srinfo-module", 10);

		application.addAppEdge("GAS", "master-module",1000, 2000, "GAS", Tuple.UP,
AppEdge.SENSOR);
		application.addAppEdge("CH", "chinfo-module", 1000, 2000, "CH", Tuple.UP,
AppEdge.SENSOR);
		application.addAppEdge("SR", "srinfo-module", 1000, 2000, "SR", Tuple.UP,
AppEdge.SENSOR);
		application.addAppEdge("master-module", "gasinfo-module", 1000, 2000, "gasTask",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "chinfo-module", 1000, 2000, "chTask",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "srinfo-module", 1000, 2000, "srTask", Tuple.UP,
AppEdge.MODULE);
		//Response
		application.addAppEdge("gasinfo-module", "master-module", 1000, 2000, "gasResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("chinfo-module", "master-module", 1000, 2000, "chResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("srinfo-module", "master-module", 1000, 2000,"srResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "ACT_CONTROL", 100, 28, 100,
"ACT_PARAMS", Tuple.UP, AppEdge.ACTUATOR);

		application.addTupleMapping("master-module", "GAS", "gasTask", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "CH", "chTask", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "SR", "srTas", new
FractionalSelectivity(1.0));
		application.addTupleMapping("gasinfo-module", "gasTask", "gasResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("chinfo-module", "chTask", "chResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("srinfo-module", "srTask", "srResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "gasResponse", "ACT_PARAMS", new
FractionalSelectivity(1.0));

		final AppLoop loop1 = new AppLoop(new ArrayList<String>(){
			{
				add("GAS");
				add("master-module");
				add("gasinfo-module");
				add("gasTask");
				add("gasResponse");
				add("ACT_PARAMS");
			}
		});
		final AppLoop loop2 = new AppLoop(new ArrayList<String>(){
			{
				add("CH");
				add("master-module");
				add("chinfo-module");
				add("chTask");
				add("chResponse");
			}
```

```
        });
        final AppLoop loop3 = new AppLoop(new ArrayList<String>(){
                {
                        add("SR");
                        add("master-module");
                        add("srinfo-module");
                        add("srTask");
                        add("srResponse");
                }
        });
        List<AppLoop> loops = new ArrayList<AppLoop>(){
                {
                        add(loop1);
                        add(loop2);
                        add(loop3);
                }
        };
        application.setLoops(loops);
        return application;
}
}
```

2.Smart Garbage Monitoring System:

Source Code:

```
package org.fog.test.perfeval;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;

import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
```

```
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.sdn.overbooking.BwProvisionerOverbooking;
import org.cloudbus.cloudsim.sdn.overbooking.PeProvisionerOverbooking;
import org.fog.application.AppEdge;
import org.fog.application.AppLoop;
import org.fog.application.Application;
import org.fog.application.selectivity.FractionalSelectivity;
import org.fog.entities.Actuator;
import org.fog.entities.FogBroker;
import org.fog.entities.FogDevice;
import org.fog.entities.FogDeviceCharacteristics;
import org.fog.entities.Sensor;
import org.fog.entities.Tuple;
import org.fog.placement.Controller;
import org.fog.placement.ModuleMapping;
import org.fog.placement.ModulePlacementEdgewards;
import org.fog.placement.ModulePlacementMapping;
import org.fog.policy.AppModuleAllocationPolicy;
import org.fog.scheduler.StreamOperatorScheduler;
import org.fog.utils.FogLinearPowerModel;
import org.fog.utils.FogUtils;
import org.fog.utils.TimeKeeper;
import org.fog.utils.distribution.DeterministicDistribution;
public class GarbageManagement {
	//Create the list of fog devices
	static List<FogDevice> fogDevices = new ArrayList<FogDevice>();
	//Create the list of sensors
	static List<Sensor> sensors = new ArrayList<Sensor>();
	//Create the list of actuators
	static List<Actuator> actuators = new ArrayList<Actuator>();
	//Define the number of areas
	static int numOfTotalAreas = 10;
	//Define the number of waste bins with each fog nodes
	static int numOfBinsPerArea=1;
	//We are using the fog nodes to perform the operations.
	//cloud is set to false
	private static boolean CLOUD = false;
	public static void main(String[] args) {
		Log.printLine("Waste Management system...");
		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 = "swms"; // identifier of the application
                FogBroker broker = new FogBroker("broker");
                Application application = createApplication(appId, broker.getId());
                application.setUserId(broker.getId());
                createFogDevices(broker.getId(), appId);
                Controller controller = null;
                ModuleMapping moduleMapping =
ModuleMapping.createModuleMapping();

                for(FogDevice device : fogDevices){
                        if(device.getName().startsWith("b")){
                                // names of all Smart Bins start with ’b’
                                moduleMapping.addModuleToDevice("waste-info-
module", device.getName());
                                // mapping waste information module on waste bins
                        }
                }

                for(FogDevice device : fogDevices){
                        if(device.getName().startsWith("a")){
                                // names of all fog devices start with ’a’
                                // mapping master-module on area devices.
                                moduleMapping.addModuleToDevice("master-
module", device.getName());
                                // mapping health-module on area devices
                                moduleMapping.addModuleToDevice("health-
module", device.getName());
                                // mapping recycle-module on area devices.
                                moduleMapping.addModuleToDevice("recycle-
module", device.getName());
                                // mapping municipal-module on area devices.
                                moduleMapping.addModuleToDevice("municipal-
module", device.getName());
                        }
                }
```

```
                        if(CLOUD){ // if the mode of deployment is cloud-based
                                // placing all instances of master-module in the Cloud
                                moduleMapping.addModuleToDevice("master-module",
"cloud");
                                // placing all instances of health-module in the Cloud
                                moduleMapping.addModuleToDevice("health-module",
"cloud");
                                //placing all instances of recycle-module in the Cloud
                                moduleMapping.addModuleToDevice("recycle-module",
"cloud");
                                // placing all instances of municipal-module in the Cloud
                                moduleMapping.addModuleToDevice("municipal-module",
"cloud");
                        }

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

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


        TimeKeeper.getInstance().setSimulationStartTime(Calendar.getInstance().getTimeIn
Millis());

                        CloudSim.startSimulation();

                        CloudSim.stopSimulation();

                        Log.printLine("waste management simulation finished!");
                } catch (Exception e) {
                        e.printStackTrace();
                        Log.printLine("Unwanted errors happen");
                }
        }

        /**
         * Creates the fog devices in the physical topology of the simulation.
```

```
	 * @param userId
	 * @param appId
	 */
	private static void createFogDevices(int userId, String appId) {
		FogDevice cloud = createFogDevice("cloud", 44800, 40000, 100, 10000, 0,
0.01, 16*103, 16*83.25);
		cloud.setParentId(-1);
		fogDevices.add(cloud);
		FogDevice router = createFogDevice("proxy-server", 7000, 4000, 10000,
10000, 1, 0.0, 107.339, 83.4333);
		router.setParentId(cloud.getId());
		// latency of connection between proxy server and cloud is 100 ms
		router.setUplinkLatency(100.0);
		fogDevices.add(router);
		for(int i=0;i<numOfTotalAreas;i++){
			addArea(i+"", userId, appId, router.getId());
		}
	}

	private static FogDevice addArea(String id, int userId, String appId, int parentId){
		FogDevice area_fognode = createFogDevice("a-"+id, 5000, 4000, 10000,
10000, 3, 0.0, 107.339, 83.4333);
		fogDevices.add( area_fognode);
		area_fognode.setUplinkLatency(1.0);
		for(int i=0;i<numOfBinsPerArea;i++){
			String mobileId = id+"-"+i;
			FogDevice bin = addBin(mobileId, userId, appId,
area_fognode.getId());
			bin.setUplinkLatency(2.0);
			fogDevices.add(bin);
		}
		//assigning x coordinate value to the fog node
		area_fognode.setxCoordinate(getCoordinatevalue(10));
		//assigning y coordinate value to the fog node
		area_fognode.setyCoordinate(getCoordinatevalue(10));
		area_fognode.setParentId(parentId);
		return area_fognode;
	}

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

```
                FogDevice bin = createFogDevice("b-"+id, 5000, 1000, 10000, 10000, 4, 0,
87.53, 82.44);
                bin.setParentId(parentId);
                Sensor sensor = new Sensor("s-"+id, "BIN", userId, appId, new
DeterministicDistribution(getCoordinatevalue(5)));
                sensors.add(sensor);
                Actuator ptz = new Actuator("act-"+id, userId, appId, "ACT_CONTROL");
                actuators.add(ptz);
                sensor.setGatewayDeviceId(bin.getId());
                sensor.setLatency(1.0);
                ptz.setGatewayDeviceId(parentId);
                ptz.setLatency(1.0);
                //assigning x coordinate value to the smart bin
                bin.setxCoordinate(getCoordinatevalue(10));
                //assigning y coordinate value to the smart bin
                bin.setyCoordinate(getCoordinatevalue(10));
                return bin;
        }

        /**
         * 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 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);
		application.addAppModule("waste-info-module", 10);
		application.addAppModule("master-module", 10);
		application.addAppModule("recycle-module", 10);
		application.addAppModule("health-module", 10);
		application.addAppModule("municipal-module", 10);

		application.addAppEdge("BIN", "waste-info-module",1000, 2000, "BIN",
Tuple.UP, AppEdge.SENSOR);
		application.addAppEdge("waste-info-module", "master-module", 1000, 2000,
"Task1", Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "municipal-module", 1000, 2000,
"Task2", Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "recycle-module", 1000, 2000,
"Task3", Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "health-module", 1000, 2000,
"Task4", Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "ACT_CONTROL", 100, 28, 100,
"ACT_PARAMS", Tuple.UP, AppEdge.ACTUATOR);

		application.addTupleMapping("waste-info-module", "BIN", "Task1", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "BIN", "Task2", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "BIN", "Task3", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "BIN", "Task4", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "BIN", "ACT_CONTROL",
new FractionalSelectivity(1.0));
```

```
                final AppLoop loop1 = new AppLoop(new ArrayList<String>(){
                        {
                                add("BIN");
                                add("waste-info-module");
                                add("master-module");
                                add("municipal-module");
                                add("recycle-module");
                                add("health-module");
                                add("ACT_CONTROL");
                        }
                });
                List<AppLoop> loops = new ArrayList<AppLoop>(){
                        {
                                add(loop1);
                        }
                };
                application.setLoops(loops);
                return application;
        }
        private static double getCoordinatevalue(double min)
        {
                Random rn=new Random();
                return rn.nextDouble()+min;
        }
}
```

3.Smart Mining System:

Source Code:

```
package org.fog.test.perfeval;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
```

```
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.sdn.overbooking.BwProvisionerOverbooking;
import org.cloudbus.cloudsim.sdn.overbooking.PeProvisionerOverbooking;
import org.fog.application.AppEdge;
import org.fog.application.AppLoop;
import org.fog.application.Application;
import org.fog.application.selectivity.FractionalSelectivity;
import org.fog.entities.Actuator;
import org.fog.entities.FogBroker;
import org.fog.entities.FogDevice;
import org.fog.entities.FogDeviceCharacteristics;
import org.fog.entities.Sensor;
import org.fog.entities.Tuple;
import org.fog.placement.Controller;
import org.fog.placement.ModuleMapping;
import org.fog.placement.ModulePlacementEdgewards;
import org.fog.placement.ModulePlacementMapping;
import org.fog.policy.AppModuleAllocationPolicy;
import org.fog.scheduler.StreamOperatorScheduler;
import org.fog.utils.FogLinearPowerModel;
import org.fog.utils.FogUtils;
import org.fog.utils.TimeKeeper;
import org.fog.utils.distribution.DeterministicDistribution;
public class SmartMining {
	//Create the list of fog devices
	static List<FogDevice> fogDevices = new ArrayList<FogDevice>();
	//Create the list of sensors
	static List<Sensor> sensors = new ArrayList<Sensor>();
	//Create the list of actuators
	static List<Actuator> actuators = new ArrayList<Actuator>();
	//Define the number of fog nodes will be deployed
	static int numOfFogDevices = 2;
	//Define the number of gas sensors with each fog nodes
	static int numOfGasSensorsPerArea=10;
	//Define the number of chemical sensors with each fog nodes
	static int numOfChSensorsPerArea=10;
	//Define the number of surrounding sensors with each fog nodes
	static int numOfSrSensorsPerArea=10;
	//We are using the fog nodes to perform the operations.
	//cloud is set to false
	private static boolean CLOUD = true;
	public static void main(String[] args) {
		Log.printLine("Smart Mining Industry System...");
		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 = "mins"; // identifier of the application
			FogBroker broker = new FogBroker("broker");
			Application application = createApplication(appId, broker.getId());
			application.setUserId(broker.getId());
			createFogDevices(broker.getId(), appId);
			Controller controller = null;
			// initializing a module mapping
			ModuleMapping moduleMapping = ModuleMapping.createModuleMapping();
//			for(FogDevice device : fogDevices){
//				if(device.getName().startsWith("a")){
```

```
//                                      moduleMapping.addModuleToDevice("master-module",
device.getName());
//                              }
//                      }
                        for(FogDevice device : fogDevices){
                                if(device.getName().startsWith("g")){
                                        moduleMapping.addModuleToDevice("gasinfo-module",
device.getName());
                                }
                        }
                        for(FogDevice device : fogDevices){
                                if(device.getName().startsWith("c")){
                                        moduleMapping.addModuleToDevice("chinfo-module",
device.getName());
                                }
                        }
                        for(FogDevice device : fogDevices){
                                if(device.getName().startsWith("s")){
                                        moduleMapping.addModuleToDevice("srinfo-module",
device.getName());
                                }
                        }
                        // if the mode of deployment is cloud-based
                        if(CLOUD){
                                // placing all instances of master-module in Cloud
                                moduleMapping.addModuleToDevice("master-module", "cloud");
                                moduleMapping.addModuleToDevice("gasinfo-module", "cloud");
                                moduleMapping.addModuleToDevice("chinfo-module", "cloud");
                                moduleMapping.addModuleToDevice("srinfo-module", "cloud");
                        }

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

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


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

                        CloudSim.startSimulation();

                        CloudSim.stopSimulation();

                        Log.printLine("mining industry simulation finished!");
                } catch (Exception e) {
                        e.printStackTrace();
                        Log.printLine("Unwanted errors happen");
                }
        }

        /**
         * Creates the fog devices in the physical topology of the simulation.
         * @param userId
         * @param appId
         */
        private static void createFogDevices(int userId, String appId) {
```

```
		FogDevice cloud = createFogDevice("cloud", 44800, 40000, 100, 10000, 0, 0.01, 16*103,
16*83.25);
		cloud.setParentId(-1);
		fogDevices.add(cloud);
		FogDevice router = createFogDevice("proxy-server", 7000, 4000, 10000, 10000, 1, 0.0,
107.339, 83.4333);
		router.setParentId(cloud.getId());
		router.setUplinkLatency(100);
		fogDevices.add(router);
		for(int i=0;i<numOfFogDevices;i++){
			addFogNode(i+"", userId, appId, router.getId());
		}
	}

	private static FogDevice addFogNode(String id, int userId, String appId, int parentId){
		FogDevice fognode = createFogDevice("a-"+id, 5000, 4000, 10000, 10000, 3, 0.0, 107.339,
83.4333);
		fogDevices.add( fognode);
		fognode.setUplinkLatency(2);
		for(int i=0;i<numOfGasSensorsPerArea;i++){
			String mobileId = id+"-"+i;
			FogDevice gasSensor = addGasSensors(mobileId, userId, appId, fognode.getId());
			gasSensor.setUplinkLatency(2);
			fogDevices.add(gasSensor);
		}
		for(int i=0;i<numOfChSensorsPerArea;i++){
			String mobileId = id+"-"+i;
			FogDevice chSensor = addChSensors(mobileId, userId, appId, fognode.getId());
			chSensor.setUplinkLatency(2);
			fogDevices.add(chSensor);
		}
		for(int i=0;i<numOfSrSensorsPerArea;i++){
			String mobileId = id+"-"+i;
			FogDevice srSensor = addSrSensors(mobileId, userId, appId, fognode.getId());
			srSensor.setUplinkLatency(2);
			fogDevices.add(srSensor);
		}
		fognode.setParentId(parentId);
		return fognode;
	}

	private static FogDevice  addGasSensors(String id, int userId, String appId, int parentId){
		FogDevice gasSensor = createFogDevice("g-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
		gasSensor.setParentId(parentId);
		Sensor sensor = new Sensor("s-"+id, "GAS", userId, appId, new
DeterministicDistribution(5));
		sensors.add(sensor);
		Actuator ptz = new Actuator("act-"+id, userId, appId, "ACT_CONTROL");
		actuators.add(ptz);
		sensor.setGatewayDeviceId(gasSensor.getId());
		sensor.setLatency(40.0);
		ptz.setGatewayDeviceId(parentId);
		ptz.setLatency(1.0);
		return gasSensor;
	}

	private static FogDevice addChSensors(String id, int userId, String appId, int parentId){
		FogDevice chSensor = createFogDevice("c-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
```

```
		chSensor.setParentId(parentId);
		Sensor sensor = new Sensor("sch-"+id, "CH", userId, appId, new
DeterministicDistribution(5));
		sensors.add(sensor);
		Actuator ptzch = new Actuator("actch-"+id, userId, appId, "ACT_CONTROLCH");
		actuators.add(ptzch);
		sensor.setGatewayDeviceId(chSensor.getId());
		sensor.setLatency(40.0);
		ptzch.setGatewayDeviceId(parentId);
		ptzch.setLatency(1.0);
		return chSensor;
	}

	private static FogDevice addSrSensors(String id, int userId, String appId, int parentId){
		FogDevice srSensor = createFogDevice("s-"+id, 5000, 1000, 10000, 10000, 4, 0, 87.53,
82.44);
		srSensor.setParentId(parentId);
		Sensor sensor = new Sensor("ssr-"+id, "SR", userId, appId, new
DeterministicDistribution(5));
		sensors.add(sensor);
		Actuator ptzch = new Actuator("actsr-"+id, userId, appId, "ACT_CONTROLSR");
		actuators.add(ptzch);
		sensor.setGatewayDeviceId(srSensor.getId());
		sensor.setLatency(40.0);
		ptzch.setGatewayDeviceId(parentId);
		ptzch.setLatency(1.0);
		return srSensor;
	}

	/**
	 * 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 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);
                application.addAppModule("gasinfo-module", 10);
                application.addAppModule("master-module", 10);
                application.addAppModule("chinfo-module", 10);
                application.addAppModule("srinfo-module", 10);

                application.addAppEdge("GAS", "master-module",1000, 2000, "GAS", Tuple.UP,
AppEdge.SENSOR);
                application.addAppEdge("CH", "chinfo-module", 1000, 2000, "CH", Tuple.UP,
AppEdge.SENSOR);
                application.addAppEdge("SR", "srinfo-module", 1000, 2000, "SR", Tuple.UP,
AppEdge.SENSOR);
                application.addAppEdge("master-module", "gasinfo-module", 1000, 2000, "gasTask",
Tuple.UP, AppEdge.MODULE);
                application.addAppEdge("master-module", "chinfo-module", 1000, 2000, "chTask",
Tuple.UP, AppEdge.MODULE);
                application.addAppEdge("master-module", "srinfo-module", 1000, 2000, "srTask", Tuple.UP,
AppEdge.MODULE);
```

```
		//Response
		application.addAppEdge("gasinfo-module", "master-module", 1000, 2000, "gasResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("chinfo-module", "master-module", 1000, 2000, "chResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("srinfo-module", "master-module", 1000, 2000,"srResponse",
Tuple.UP, AppEdge.MODULE);
		application.addAppEdge("master-module", "ACT_CONTROL", 100, 28, 100,
"ACT_PARAMS", Tuple.UP, AppEdge.ACTUATOR);

		application.addTupleMapping("master-module", "GAS", "gasTask", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "CH", "chTask", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "SR", "srTas", new
FractionalSelectivity(1.0));
		application.addTupleMapping("gasinfo-module", "gasTask", "gasResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("chinfo-module", "chTask", "chResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("srinfo-module", "srTask", "srResponse", new
FractionalSelectivity(1.0));
		application.addTupleMapping("master-module", "gasResponse", "ACT_PARAMS", new
FractionalSelectivity(1.0));

		final AppLoop loop1 = new AppLoop(new ArrayList<String>(){
			{
				add("GAS");
				add("master-module");
				add("gasinfo-module");
				add("gasTask");
				add("gasResponse");
				add("ACT_PARAMS");
			}
		});
		final AppLoop loop2 = new AppLoop(new ArrayList<String>(){
			{
				add("CH");
				add("master-module");
				add("chinfo-module");
				add("chTask");
				add("chResponse");
			}
		});
		final AppLoop loop3 = new AppLoop(new ArrayList<String>(){
			{
				add("SR");
				add("master-module");
				add("srinfo-module");
				add("srTask");
				add("srResponse");
			}
		});
		List<AppLoop> loops = new ArrayList<AppLoop>(){
			{
				add(loop1);
				add(loop2);
				add(loop3);
			}
		};
```

```
                application.setLoops(loops);
                return application;
        }
}
```

4.UAV Sensing:

Source Code:

```
package org.fog.test.perfeval;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.sdn.overbooking.BwProvisionerOverbooking;
import org.cloudbus.cloudsim.sdn.overbooking.PeProvisionerOverbooking;
import org.fog.application.AppEdge;
import org.fog.application.AppLoop;
import org.fog.application.Application;
import org.fog.application.selectivity.FractionalSelectivity;
import org.fog.entities.Actuator;
import org.fog.entities.FogBroker;
import org.fog.entities.FogDevice;
import org.fog.entities.FogDeviceCharacteristics;
import org.fog.entities.Sensor;
import org.fog.entities.Tuple;
import org.fog.placement.Controller;
import org.fog.placement.ModuleMapping;
import org.fog.placement.ModulePlacementEdgewards;
import org.fog.placement.ModulePlacementMapping;
import org.fog.policy.AppModuleAllocationPolicy;
import org.fog.scheduler.StreamOperatorScheduler;
```

```
import org.fog.utils.FogLinearPowerModel;
import org.fog.utils.FogUtils;
import org.fog.utils.TimeKeeper;
import org.fog.utils.distribution.DeterministicDistribution;

public class UAVBasedSensing {
	static List<FogDevice> fogDevices = new ArrayList<FogDevice>();
	static List<Sensor> sensors = new ArrayList<Sensor>();
	static List<Actuator> actuators = new ArrayList<Actuator>();
	static int numOfAreas = 1;
	static int numOfDronesPerArea = 4;

	private static boolean CLOUD = true;

	public static void main(String[] args) {

		Log.printLine("Starting UAV Based Sensing as a Service...");

		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 = "ubsas"; // identifier of the application

			FogBroker broker = new FogBroker("broker");

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

			createFogDevices(broker.getId(), appId);

			Controller controller = null;

			ModuleMapping moduleMapping =
ModuleMapping.createModuleMapping(); // initializing a module mapping
			for(FogDevice device : fogDevices){
```

```
                        if(device.getName().startsWith("m")){ // names of all Smart
Drones start with 'm'

        moduleMapping.addModuleToDevice("motion_detector", device.getName());  //
fixing 1 instance of the Motion Detector module to each Smart Drone
                        }
                }
                moduleMapping.addModuleToDevice("user_interface", "cloud"); //
fixing instances of User Interface module in the Cloud
                if(CLOUD){
                        // if the mode of deployment is cloud-based
                        moduleMapping.addModuleToDevice("object_detector",
"cloud"); // placing all instances of Object Detector module in the Cloud
                        moduleMapping.addModuleToDevice("object_tracker",
"cloud"); // placing all instances of Object Tracker module in the Cloud
                }

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

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


        TimeKeeper.getInstance().setSimulationStartTime(Calendar.getInstance().getTimeIn
Millis());

                CloudSim.startSimulation();

                CloudSim.stopSimulation();

                Log.printLine("UAV Based Sensing as a Service finished!");
        } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
        }
    }
```

```
	/**
	 * Creates the fog devices in the physical topology of the simulation.
	 * @param userId
	 * @param appId
	 */
	private static void createFogDevices(int userId, String appId) {
		FogDevice cloud = createFogDevice("cloud", 44800, 40000, 100, 10000, 0,
0.01, 16*103, 16*83.25);
		cloud.setParentId(-1);
		fogDevices.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); // latency of connection between proxy server
and cloud is 100 ms
		fogDevices.add(proxy);
		for(int i=0;i<numOfAreas;i++){
			addArea(i+"", userId, appId, proxy.getId());
		}
	}

	private static FogDevice addArea(String id, int userId, String appId, int parentId){
		FogDevice router = createFogDevice("d-"+id, 2800, 4000, 10000, 10000, 1,
0.0, 107.339, 83.4333);
		fogDevices.add(router);
		router.setUplinkLatency(2); // latency of connection between router and proxy
server is 2 ms
		for(int i=0;i<numOfDronesPerArea;i++){
			String mobileId = id+"-"+i;
			FogDevice Drone = addDrone(mobileId, userId, appId, router.getId());
// adding a smart Drone to the physical topology. Smart Drones have been modeled as fog
devices as well.
			Drone.setUplinkLatency(2); // latency of connection between Drone
and router is 2 ms
			fogDevices.add(Drone);
		}
		router.setParentId(parentId);
		return router;
	}
```

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

	/**
	 * 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 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("object_detector", 10);
		application.addAppModule("motion_detector", 10);
		application.addAppModule("object_tracker", 10);
		application.addAppModule("user_interface", 10);

		/*
		 * Connecting the application modules (vertices) in the application model (directed graph) with edges
		 */
		application.addAppEdge("Drone", "motion_detector", 1000, 20000, "Drone", Tuple.UP, AppEdge.SENSOR); // adding edge from Drone (sensor) to Motion Detector module carrying tuples of type Drone
		application.addAppEdge("motion_detector", "object_detector", 2000, 2000, "MOTION_VIDEO_STREAM", Tuple.UP, AppEdge.MODULE); // adding edge from Motion Detector to Object Detector module carrying tuples of type MOTION_VIDEO_STREAM
		application.addAppEdge("object_detector", "user_interface", 500, 2000, "DETECTED_OBJECT", Tuple.UP, AppEdge.MODULE); // adding edge from Object Detector to User Interface module carrying tuples of type DETECTED_OBJECT
```

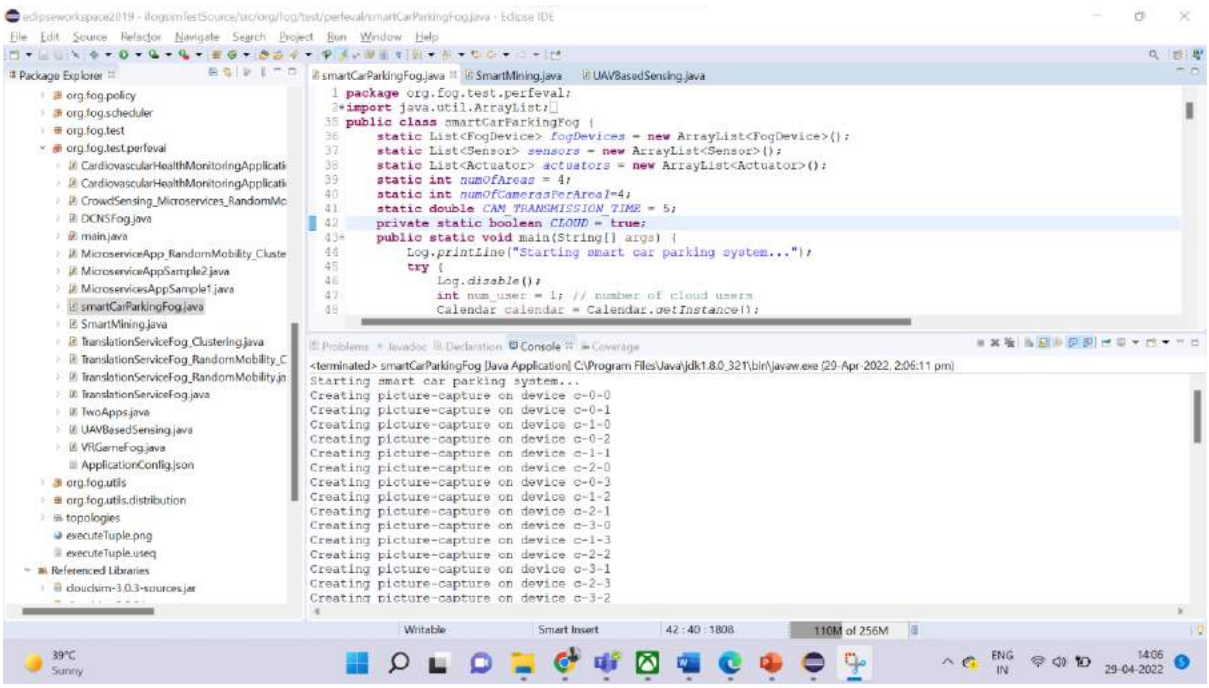
```
		application.addAppEdge("object_detector", "object_tracker", 1000, 100, "OBJECT_LOCATION", Tuple.UP, AppEdge.MODULE); // adding edge from Object Detector to Object Tracker module carrying tuples of type OBJECT_LOCATION
		application.addAppEdge("object_tracker", "PTZ_CONTROL", 100, 28, 100, "PTZ_PARAMS", Tuple.DOWN, AppEdge.ACTUATOR); // adding edge from Object Tracker to PTZ CONTROL (actuator) carrying tuples of type PTZ_PARAMS

		/*
		 * Defining the input-output relationships (represented by selectivity) of the application modules.
		 */
		application.addTupleMapping("motion_detector", "Drone", "MOTION_VIDEO_STREAM", new FractionalSelectivity(1.0)); // 1.0 tuples of type MOTION_VIDEO_STREAM are emitted by Motion Detector module per incoming tuple of type Drone
		application.addTupleMapping("object_detector", "MOTION_VIDEO_STREAM", "OBJECT_LOCATION", new FractionalSelectivity(1.0)); // 1.0 tuples of type OBJECT_LOCATION are emitted by Object Detector module per incoming tuple of type MOTION_VIDEO_STREAM
		application.addTupleMapping("object_detector", "MOTION_VIDEO_STREAM", "DETECTED_OBJECT", new FractionalSelectivity(0.05)); // 0.05 tuples of type MOTION_VIDEO_STREAM are emitted by Object Detector module per incoming tuple of type MOTION_VIDEO_STREAM

		/*
		 * Defining application loops (maybe incomplete loops) to monitor the latency of.
		 * Here, we add two loops for monitoring : Motion Detector -> Object Detector -> Object Tracker and Object Tracker -> PTZ Control
		 */
		final AppLoop loop1 = new AppLoop(new ArrayList<String>(){{add("motion_detector");add("object_detector");add("object_tracker");}});
		final AppLoop loop2 = new AppLoop(new ArrayList<String>(){{add("object_tracker");add("PTZ_CONTROL");}});
		List<AppLoop> loops = new ArrayList<AppLoop>(){{add(loop1);add(loop2);}};

		application.setLoops(loops);
		return application;
	}
```

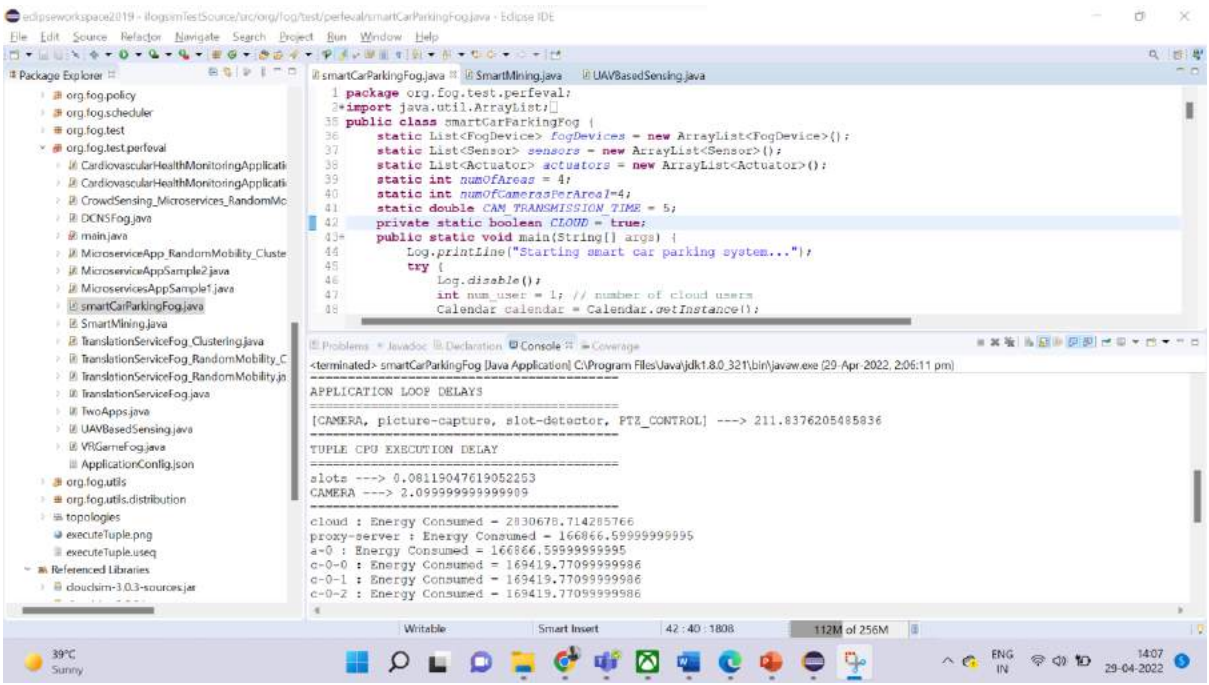
}

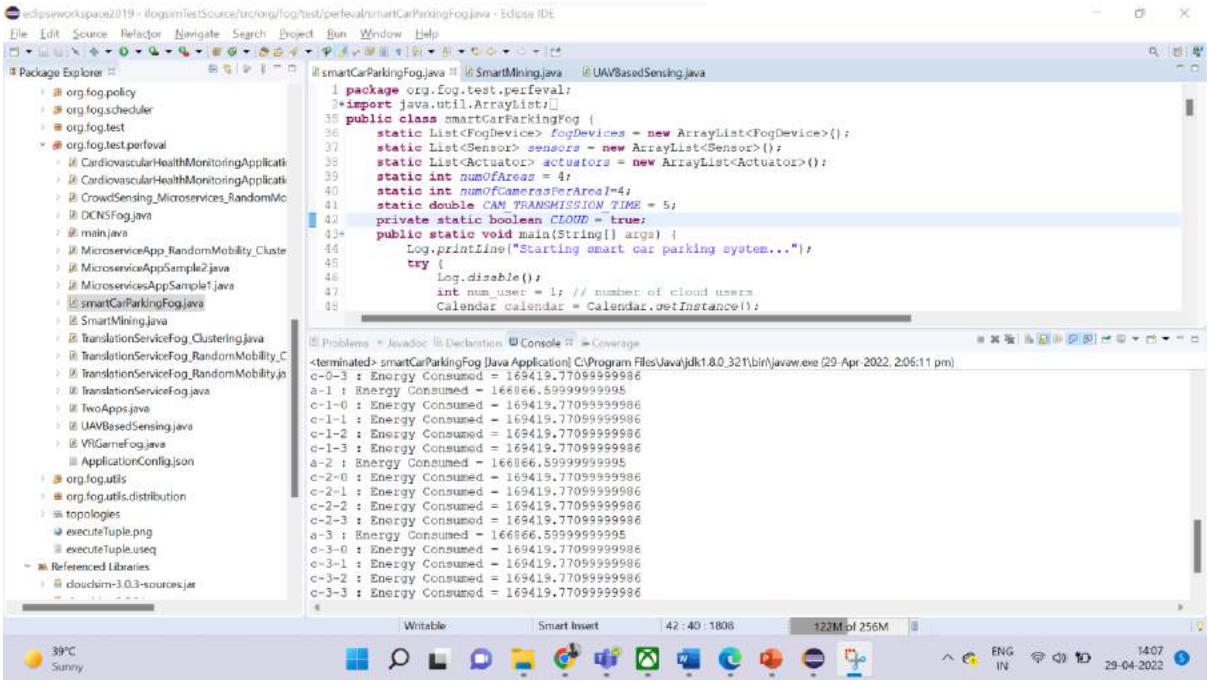
Results and Discussion:

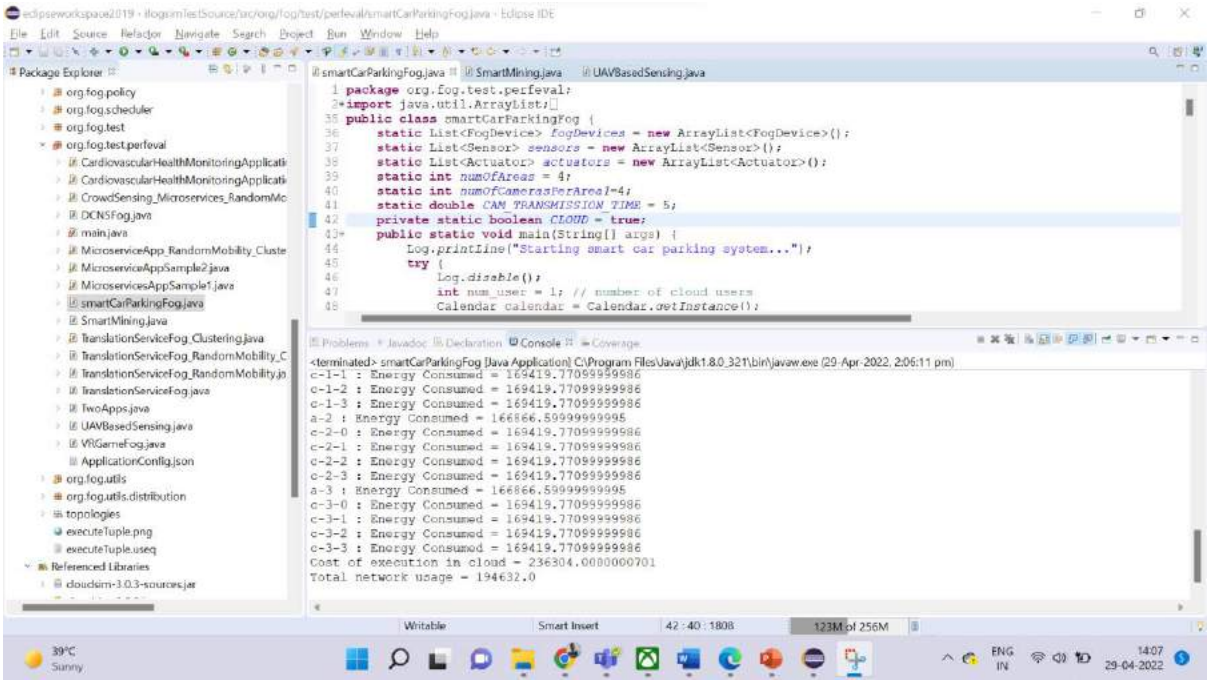
1. Smart Parking System:

Simulation for Cloud:

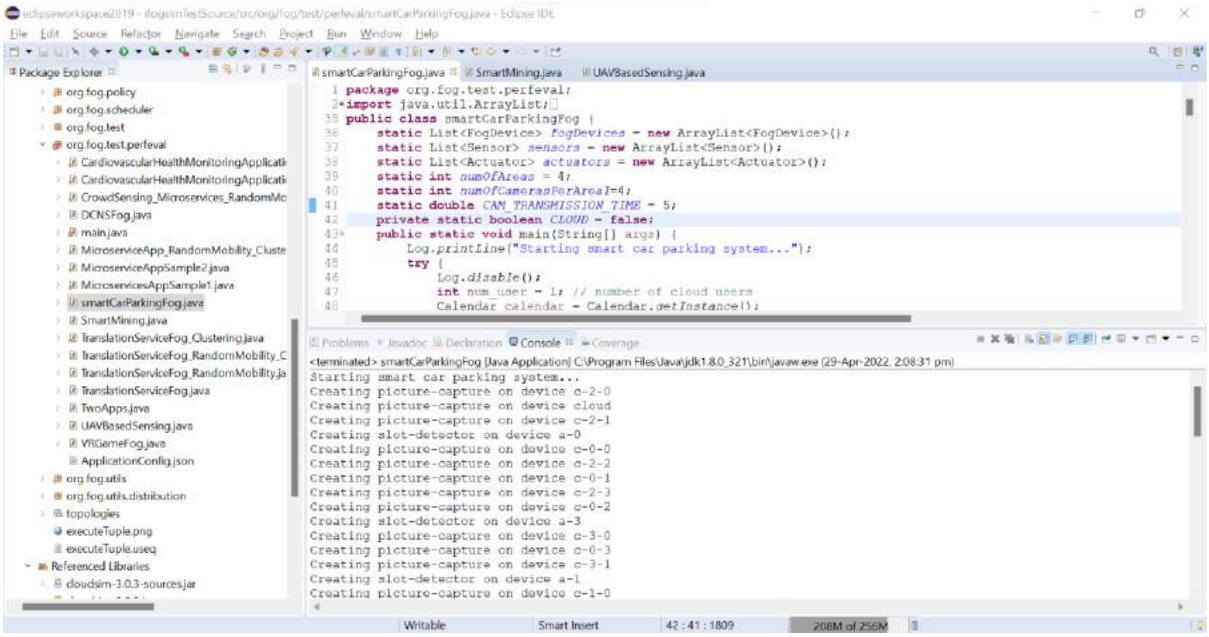


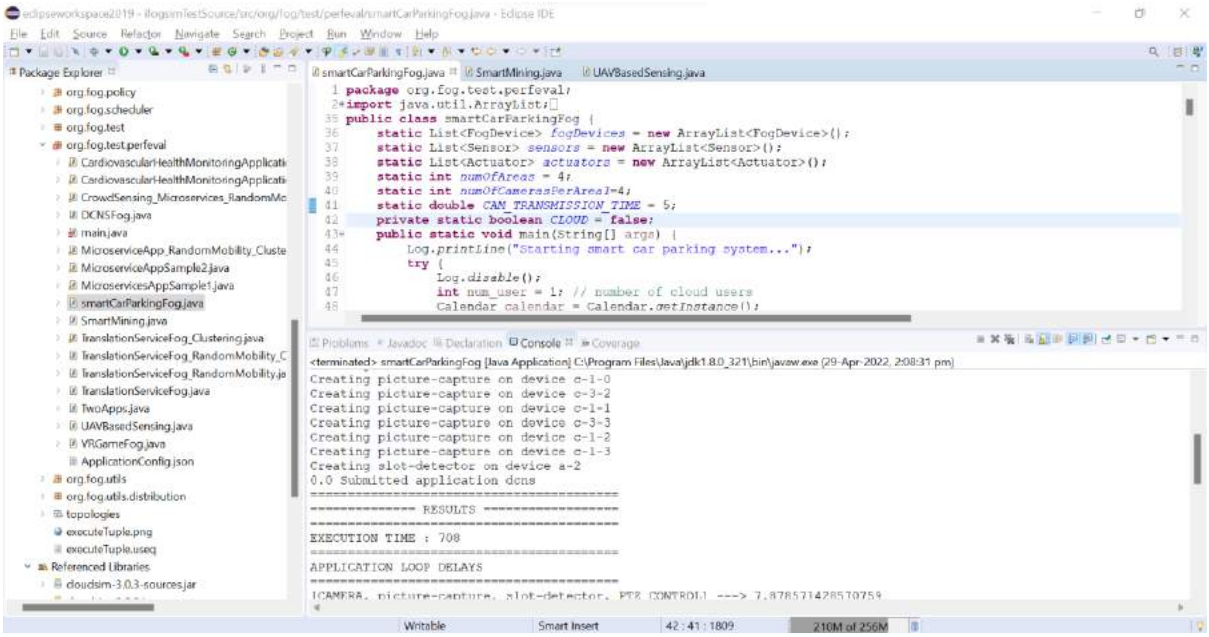


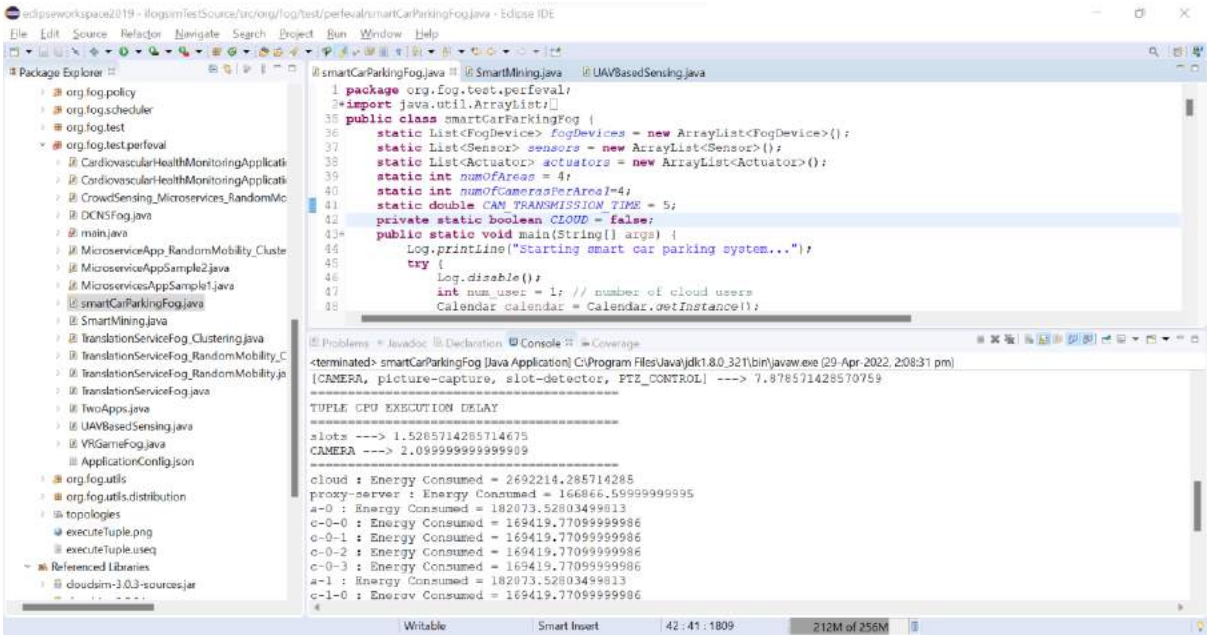


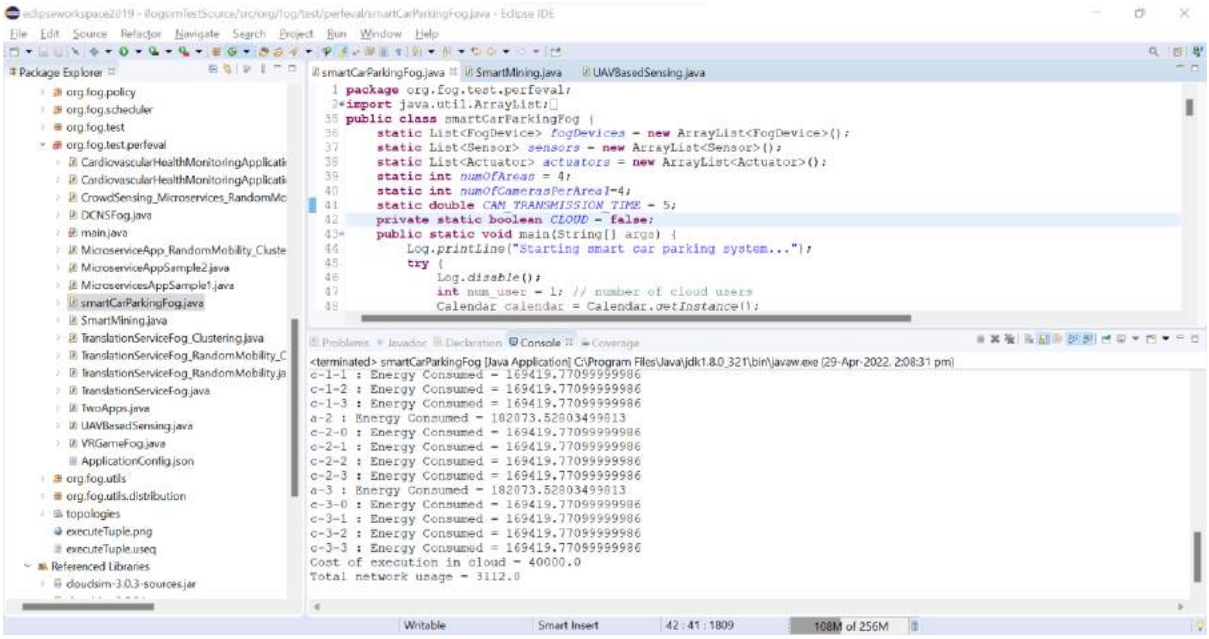


For Fog Computing:


File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
Problems Javadoc Declaration Console Coverage
<terminated> smartCarParkingFog [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:08:31 pm)
Starting smart car parking system...
Creating picture-capture on device c-2-0
Creating picture-capture on device cloud
Creating picture-capture on device c-2-1
Creating slot-detector on device a-0
Creating picture-capture on device c-0-0
Creating picture-capture on device c-2-2
Creating picture-capture on device c-0-1
Creating picture-capture on device c-2-3
Creating picture-capture on device c-0-2
Creating slot-detector on device a-3
Creating picture-capture on device c-3-0
Creating picture-capture on device c-0-3
Creating picture-capture on device c-3-1
Creating slot-detector on device a-1
Creating picture-capture on device c-1-0
Writable
Smart Insert
42 : 41 : 1809
208M of 256M

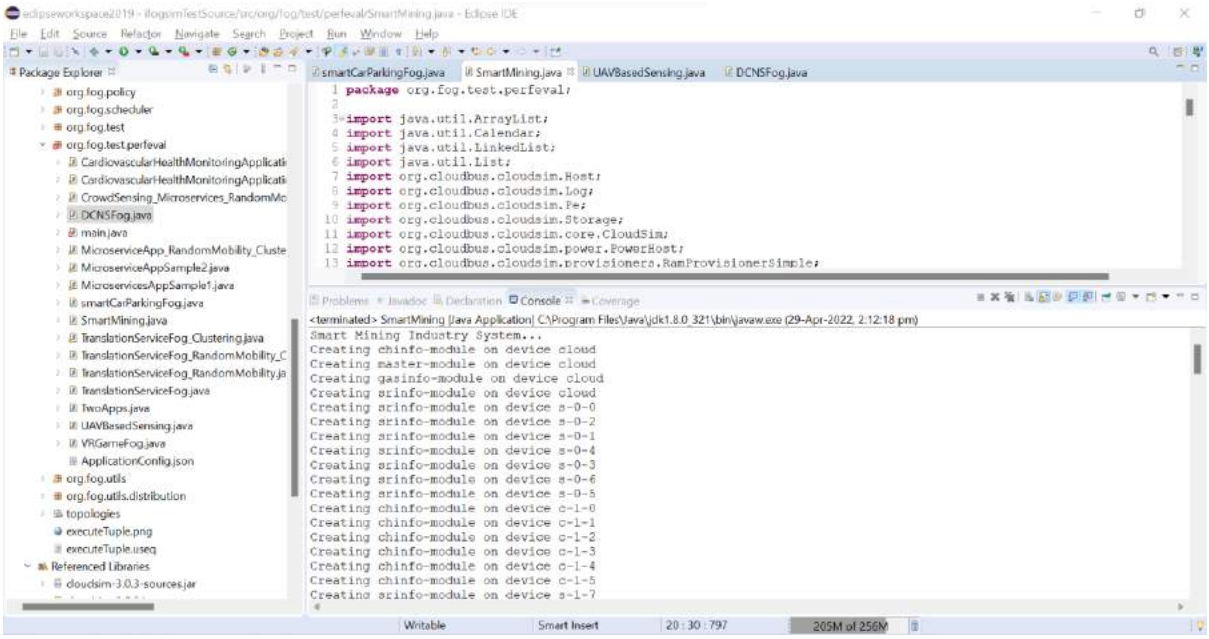

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
Problems Javadoc Declaration Console Coverage
<terminated> smartCarParkingFog [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:08:31 pm)
Creating picture-capture on device c-1-0
Creating picture-capture on device c-3-2
Creating picture-capture on device c-1-1
Creating picture-capture on device c-3-3
Creating picture-capture on device c-1-2
Creating picture-capture on device c-1-3
Creating slot-detector on device a-2
0.0 Submitted application dcns
========== RESULTS ==========
EXECUTION TIME : 708
APPLICATION LOOP DELAYS
Writable
Smart Insert
42 : 41 : 1809
210M of 256M

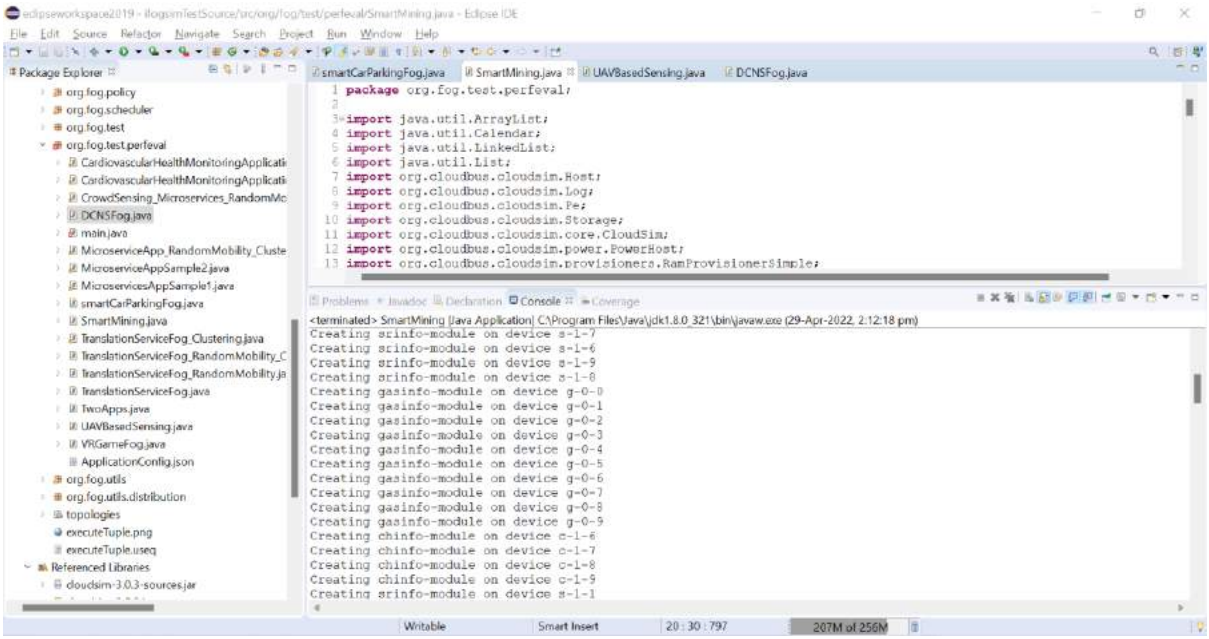


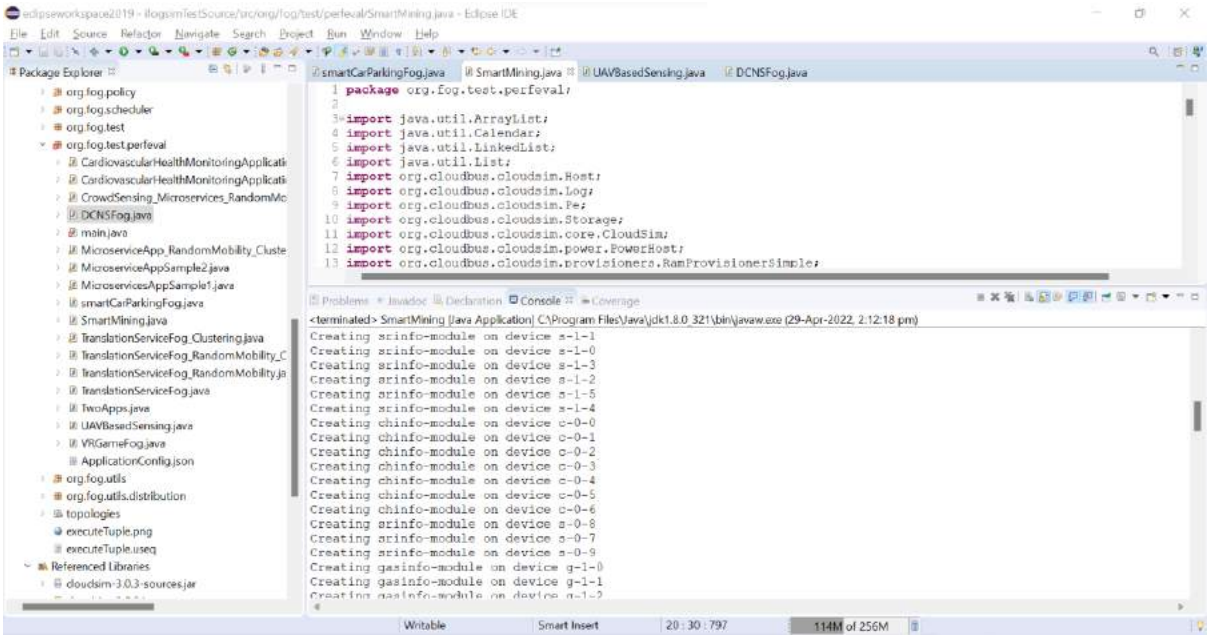


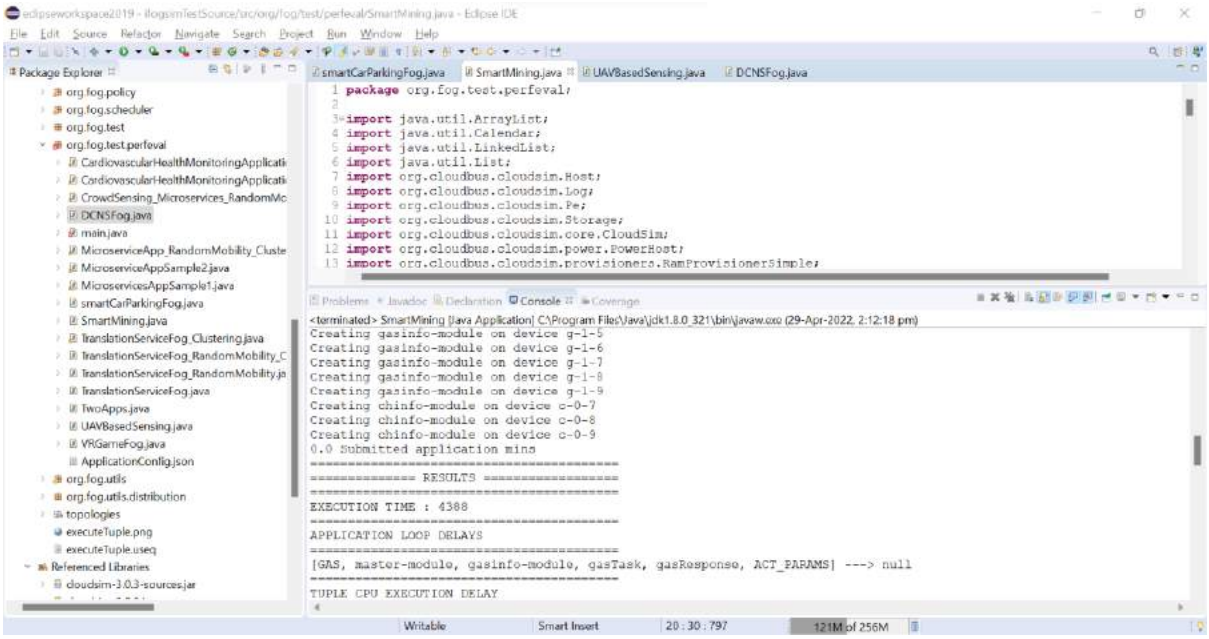
2. Smart Mining:

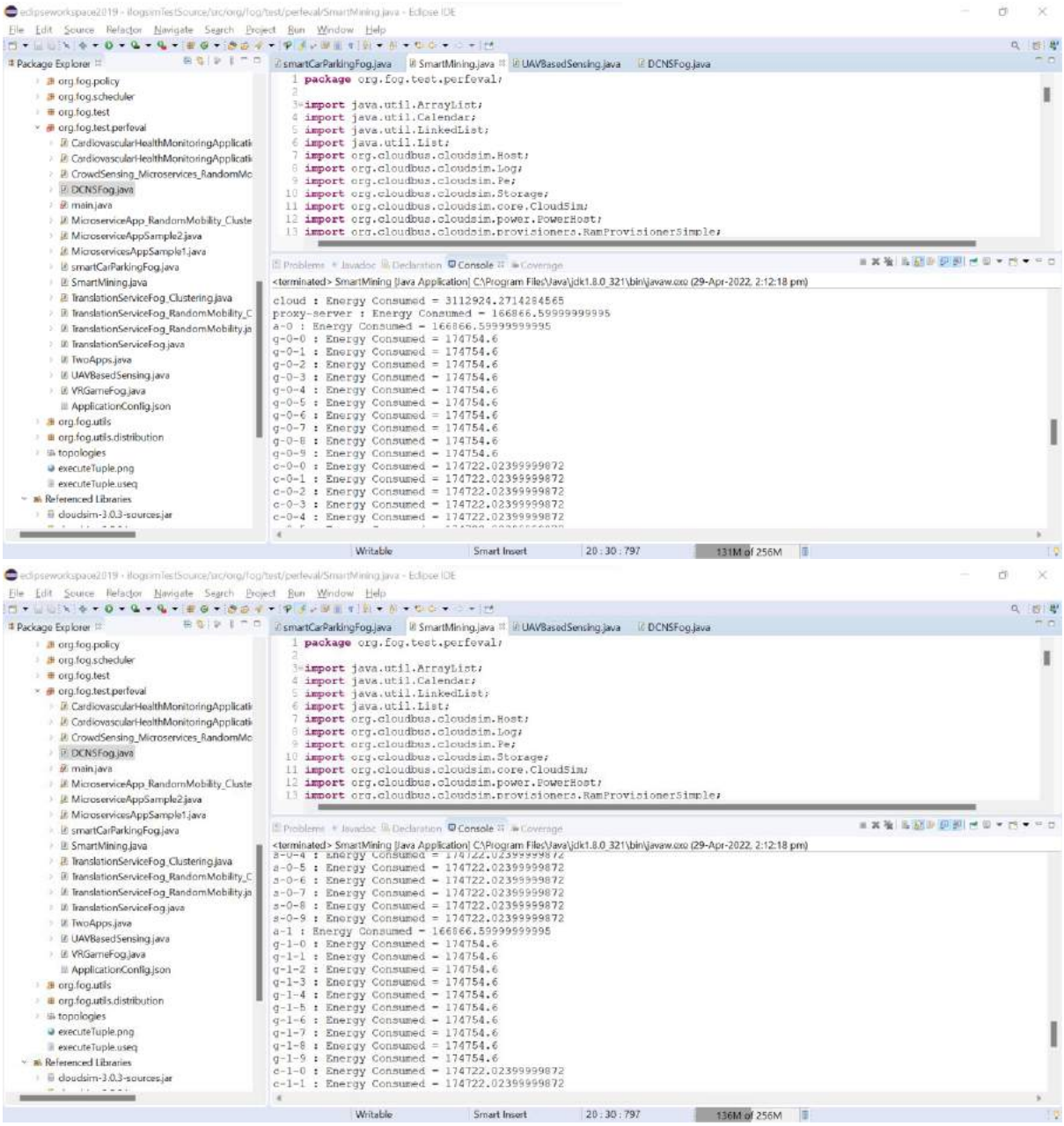
Simulation for Cloud:

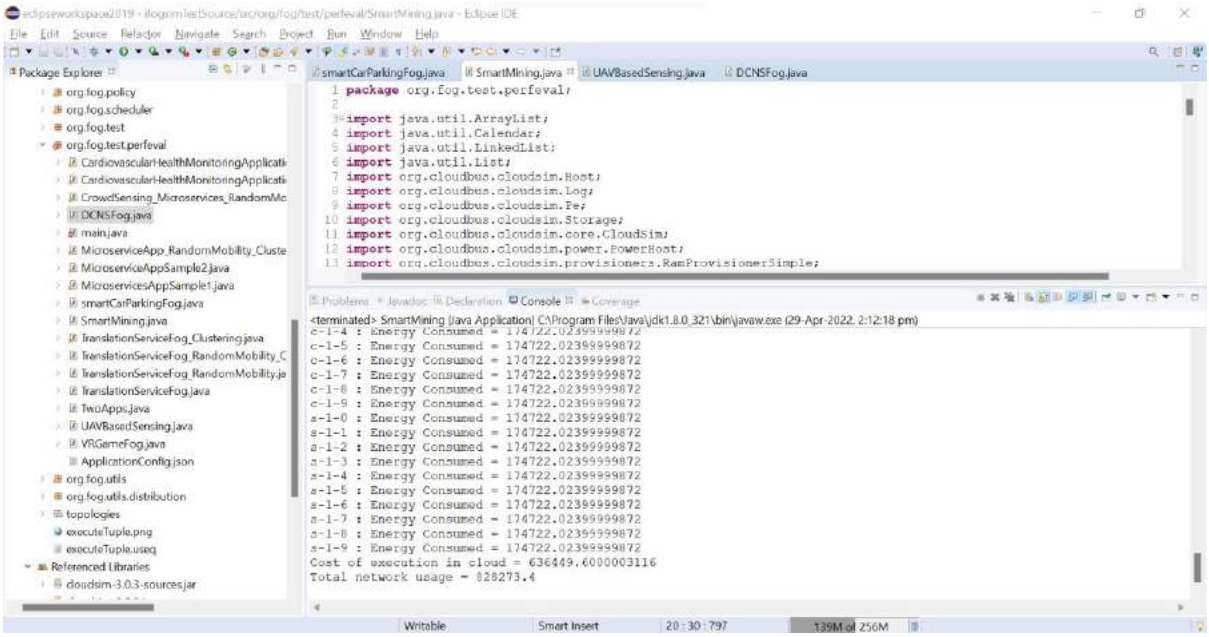

eclipseworkspace2019 - iFogSimTestSource/src/org/fog/test/perfeval/SmartMining.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
2
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
Smart Mining Industry System...
Creating chinfo-module on device cloud
Creating master-module on device cloud
Creating gasinfo-module on device cloud
Creating srinfo-module on device cloud
Creating srinfo-module on device s-0-0
Creating srinfo-module on device s-0-2
Creating srinfo-module on device s-0-1
Creating srinfo-module on device s-0-4
Creating srinfo-module on device s-0-3
Creating srinfo-module on device s-0-6
Creating srinfo-module on device s-0-5
Creating chinfo-module on device c-1-0
Creating chinfo-module on device c-1-1
Creating chinfo-module on device c-1-2
Creating chinfo-module on device c-1-3
Creating chinfo-module on device c-1-4
Creating chinfo-module on device c-1-5
Creating srinfo-module on device s-1-7
Writable
Smart Insert
20 : 30 : 797
205M of 256M


eclipseworkspace2019 - iFogSimTestSource/src/org/fog/test/perfeval/SmartMining.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
2
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
Creating srinfo-module on device s-1-7
Creating srinfo-module on device s-1-6
Creating srinfo-module on device s-1-9
Creating srinfo-module on device s-1-8
Creating gasinfo-module on device g-0-0
Creating gasinfo-module on device g-0-1
Creating gasinfo-module on device g-0-2
Creating gasinfo-module on device g-0-3
Creating gasinfo-module on device g-0-4
Creating gasinfo-module on device g-0-5
Creating gasinfo-module on device g-0-6
Creating gasinfo-module on device g-0-7
Creating gasinfo-module on device g-0-8
Creating gasinfo-module on device g-0-9
Creating chinfo-module on device c-1-6
Creating chinfo-module on device c-1-7
Creating chinfo-module on device c-1-8
Creating chinfo-module on device c-1-9
Creating srinfo-module on device s-1-1
Writable
Smart Insert
20 : 30 : 797
207M of 256M

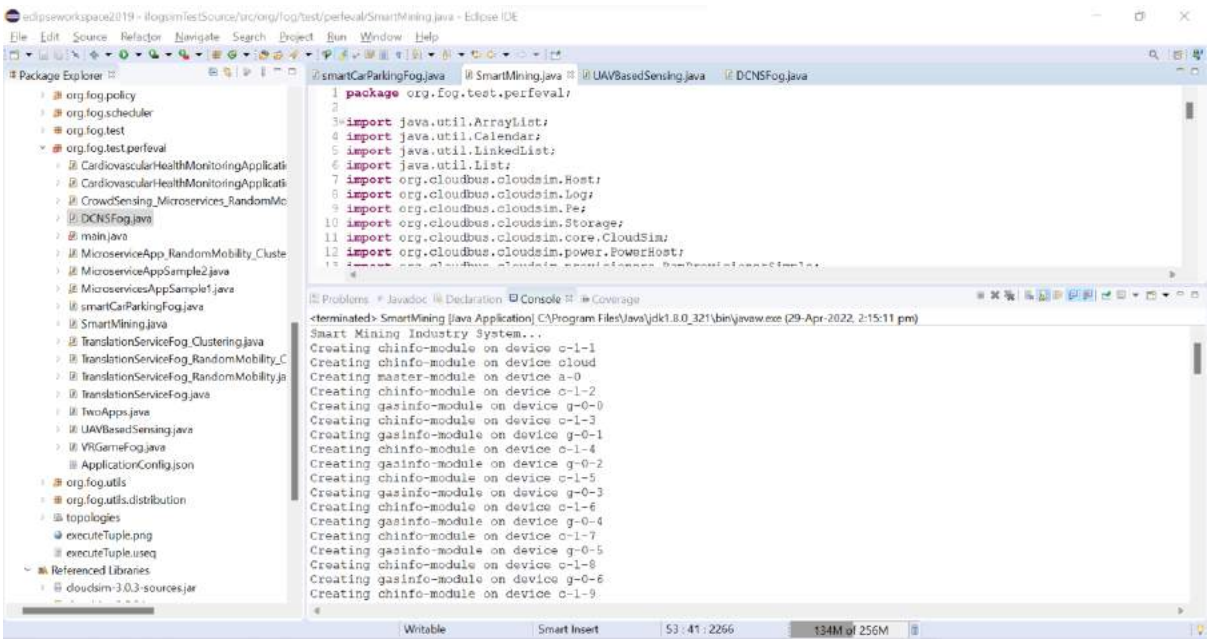

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
Creating srinfo-module on device s-1-1
Creating srinfo-module on device s-1-0
Creating srinfo-module on device s-1-3
Creating srinfo-module on device s-1-2
Creating srinfo-module on device s-1-5
Creating srinfo-module on device s-1-4
Creating chinfo-module on device c-0-0
Creating chinfo-module on device c-0-1
Creating chinfo-module on device c-0-2
Creating chinfo-module on device c-0-3
Creating chinfo-module on device c-0-4
Creating chinfo-module on device c-0-5
Creating chinfo-module on device c-0-6
Creating srinfo-module on device s-0-8
Creating srinfo-module on device s-0-7
Creating srinfo-module on device s-0-9
Creating gasinfo-module on device g-1-0
Creating gasinfo-module on device g-1-1
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
Writable
Smart Insert
20 : 30 : 797
114M of 256M

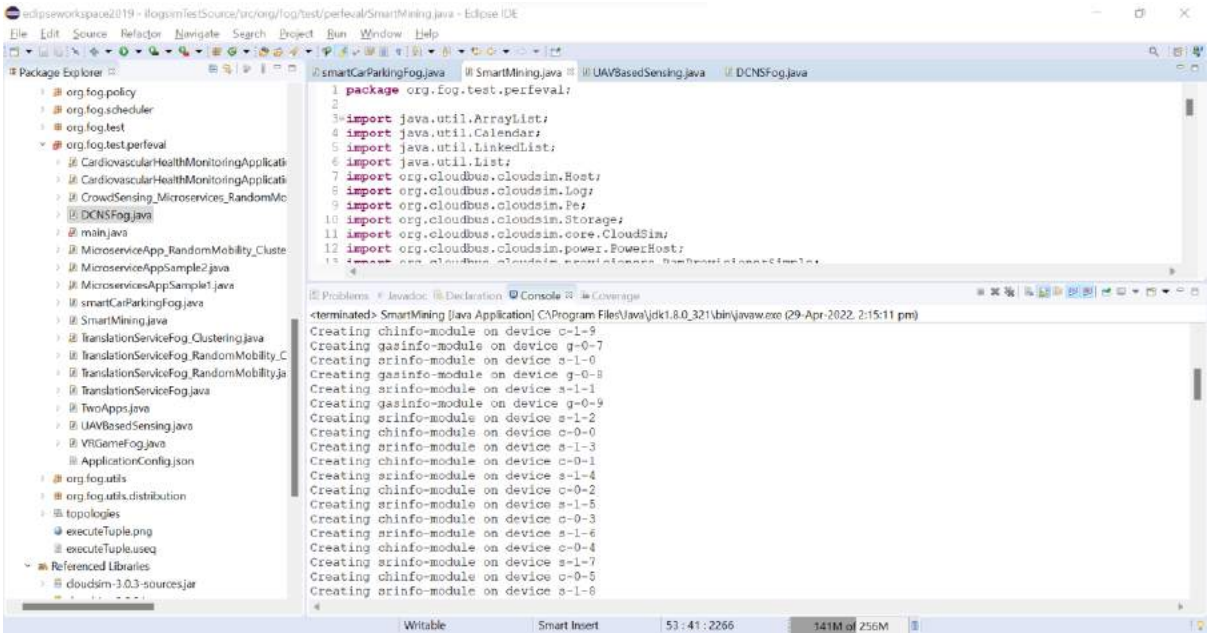

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
Creating gasinfo-module on device g-1-5
Creating gasinfo-module on device g-1-6
Creating gasinfo-module on device g-1-7
Creating gasinfo-module on device g-1-8
Creating gasinfo-module on device g-1-9
Creating chinfo-module on device c-0-7
Creating chinfo-module on device c-0-8
Creating chinfo-module on device c-0-9
0.0 Submitted application mins
========== RESULTS ==========
EXECUTION TIME : 4388
APPLICATION LOOP DELAYS
[GAS, master-module, gasinfo-module, gasTask, gasResponse, ACT_PARAMS] ---> null
TUPLE CPU EXECUTION DELAY
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
Writable
Smart Insert
20 : 30 : 797
121M of 256M

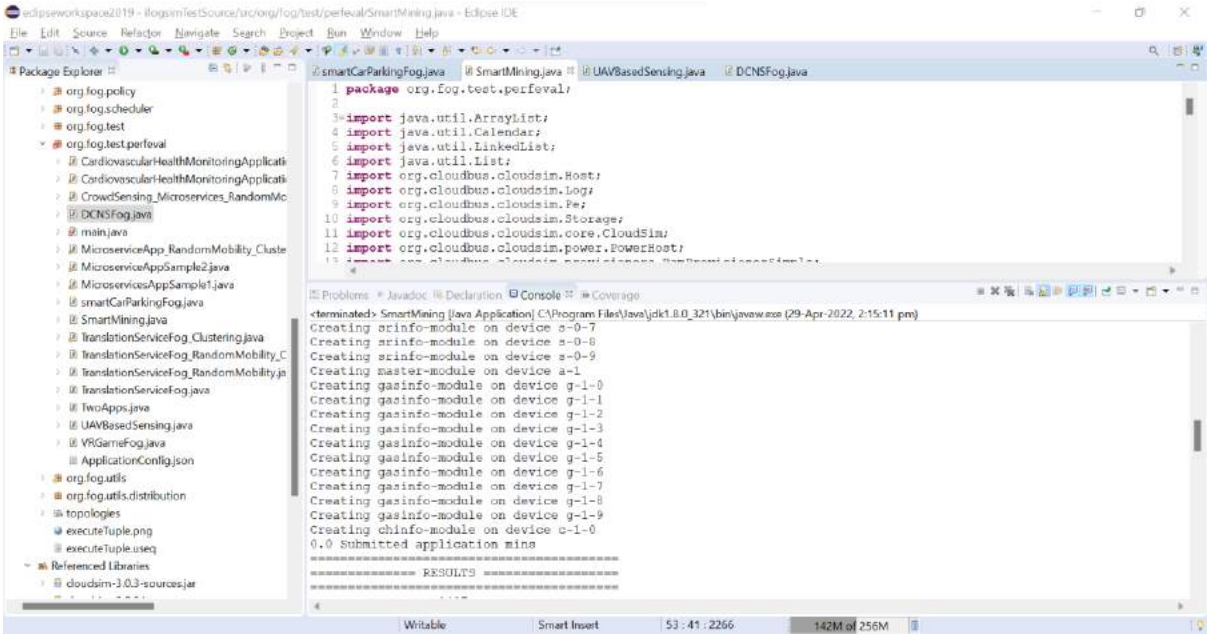

eclipse-workspace2019 - iFogSimTestSource/src/org/fog/test/perfeval/SmartMining.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
2
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
cloud : Energy Consumed = 3112924.2714284565
proxy-server : Energy Consumed = 166866.59999999995
a-0 : Energy Consumed = 166866.59999999995
g-0-0 : Energy Consumed = 174754.6
g-0-1 : Energy Consumed = 174754.6
g-0-2 : Energy Consumed = 174754.6
g-0-3 : Energy Consumed = 174754.6
g-0-4 : Energy Consumed = 174754.6
g-0-5 : Energy Consumed = 174754.6
g-0-6 : Energy Consumed = 174754.6
g-0-7 : Energy Consumed = 174754.6
g-0-8 : Energy Consumed = 174754.6
g-0-9 : Energy Consumed = 174754.6
c-0-0 : Energy Consumed = 174722.02399999872
c-0-1 : Energy Consumed = 174722.02399999872
c-0-2 : Energy Consumed = 174722.02399999872
c-0-3 : Energy Consumed = 174722.02399999872
c-0-4 : Energy Consumed = 174722.02399999872
Writable
Smart Insert
20 : 30 : 797
131M of 256M
eclipse-workspace2019 - iFogSimTestSource/src/org/fog/test/perfeval/SmartMining.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
2
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
13 import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:12:18 pm)
a-0-4 : Energy Consumed = 174722.02399999872
a-0-5 : Energy Consumed = 174722.02399999872
a-0-6 : Energy Consumed = 174722.02399999872
a-0-7 : Energy Consumed = 174722.02399999872
a-0-8 : Energy Consumed = 174722.02399999872
a-0-9 : Energy Consumed = 174722.02399999872
a-1 : Energy Consumed = 166866.59999999995
g-1-0 : Energy Consumed = 174754.6
g-1-1 : Energy Consumed = 174754.6
g-1-2 : Energy Consumed = 174754.6
g-1-3 : Energy Consumed = 174754.6
g-1-4 : Energy Consumed = 174754.6
g-1-5 : Energy Consumed = 174754.6
g-1-6 : Energy Consumed = 174754.6
g-1-7 : Energy Consumed = 174754.6
g-1-8 : Energy Consumed = 174754.6
g-1-9 : Energy Consumed = 174754.6
c-1-0 : Energy Consumed = 174722.02399999872
c-1-1 : Energy Consumed = 174722.02399999872
Writable
Smart Insert
20 : 30 : 797
136M of 256M

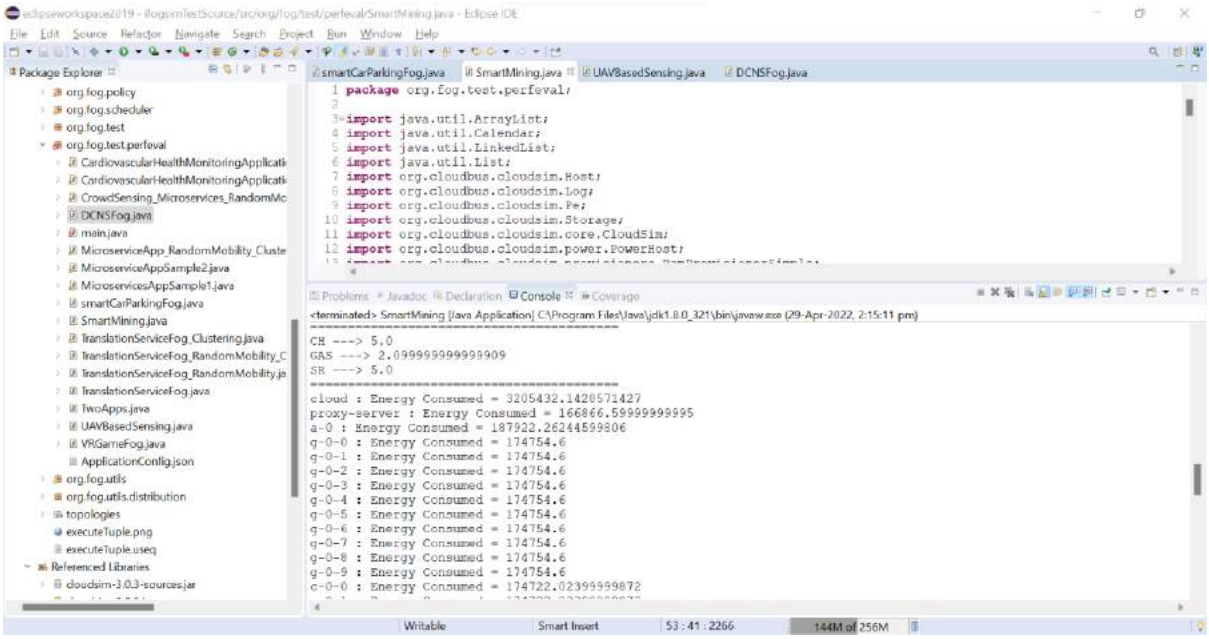


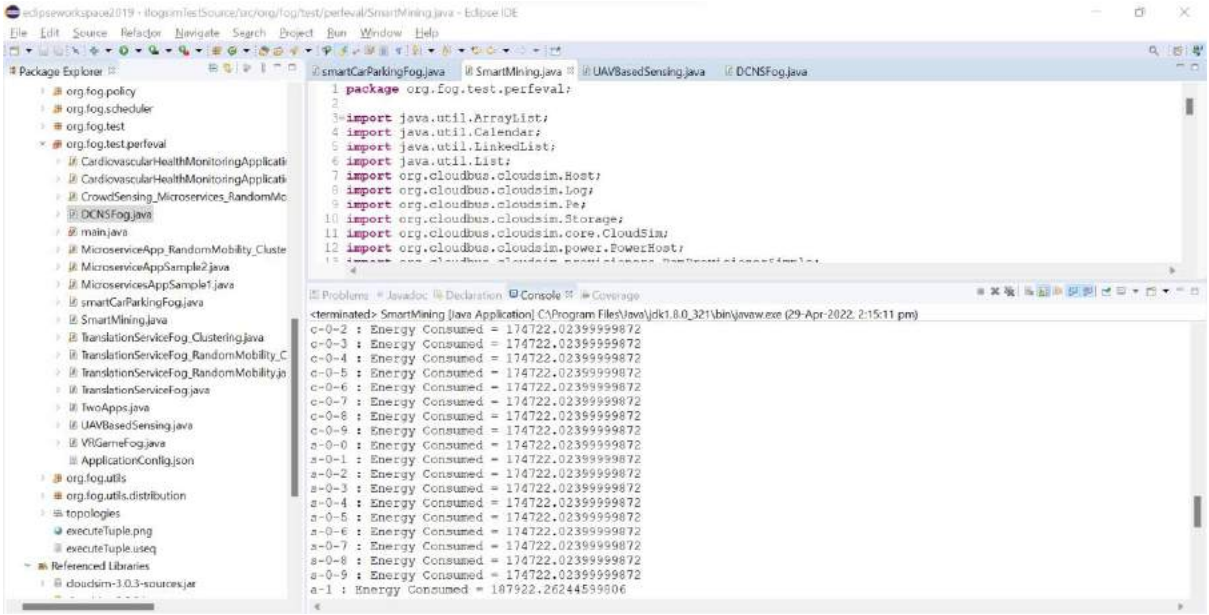
For Fog Computing:

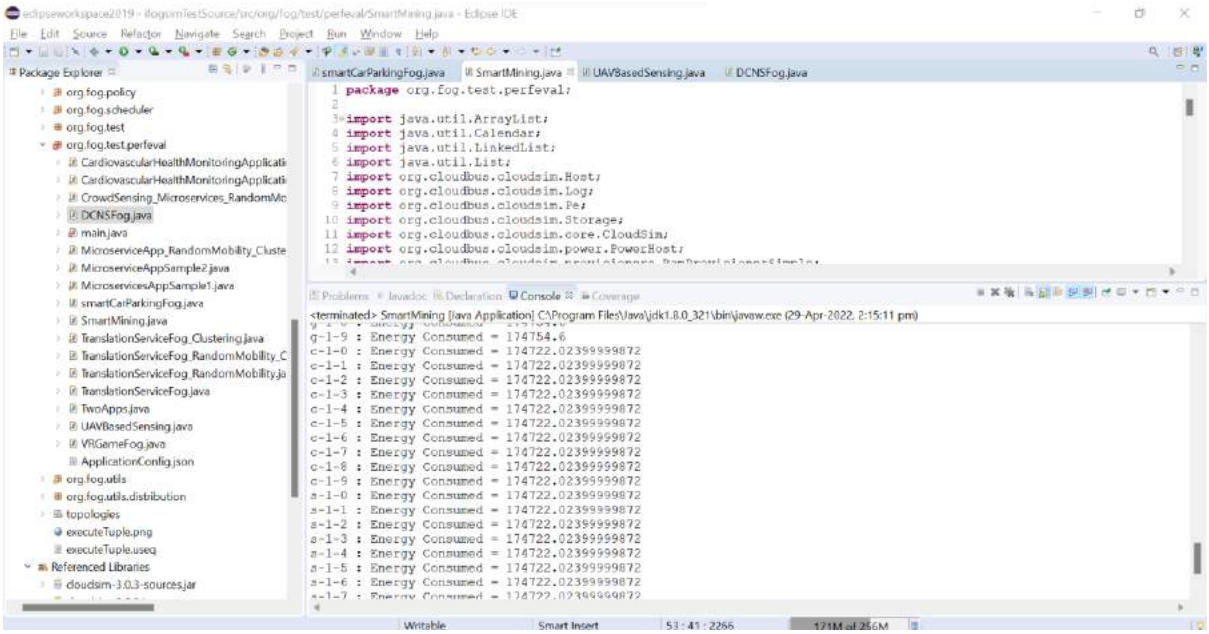


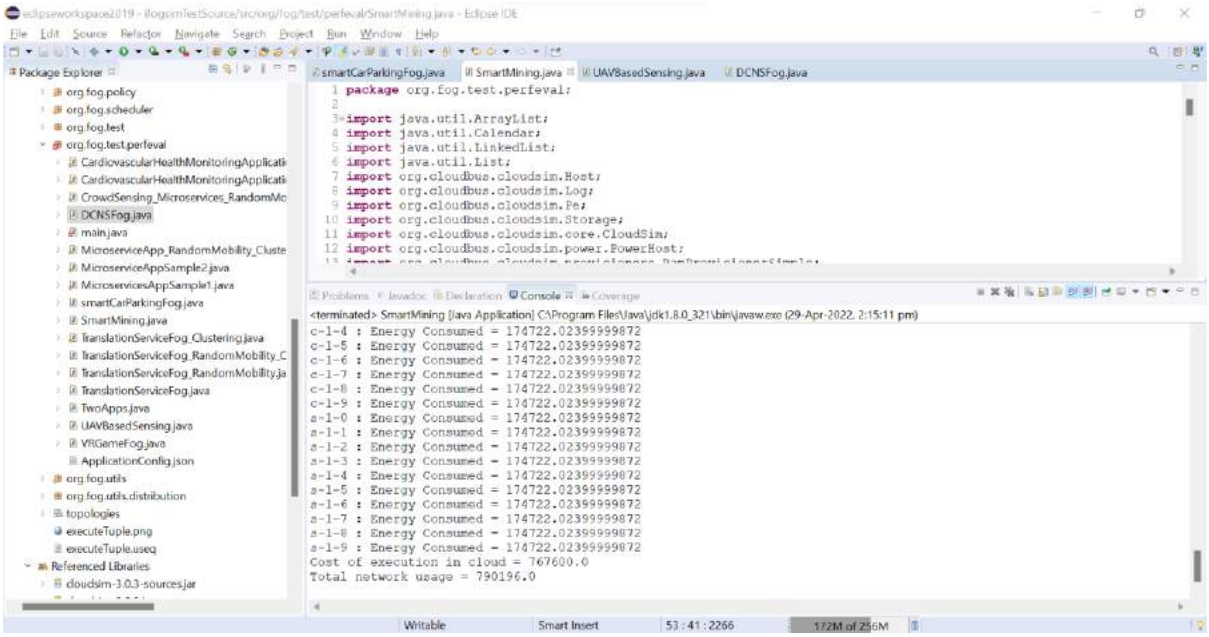

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
DCNSFog.java
main.java
smartCarParkingFog.java
SmartMining.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:15:11 pm)
Creating chinfo-module on device c-1-9
Creating gasinfo-module on device g-0-7
Creating srinfo-module on device s-1-0
Creating gasinfo-module on device g-0-8
Creating srinfo-module on device s-1-1
Creating gasinfo-module on device g-0-9
Creating srinfo-module on device s-1-2
Creating chinfo-module on device c-0-0
Creating srinfo-module on device s-1-3
Creating chinfo-module on device c-0-1
Creating srinfo-module on device s-1-4
Creating chinfo-module on device c-0-2
Creating srinfo-module on device s-1-5
Creating chinfo-module on device c-0-3
Creating srinfo-module on device s-1-6
Creating chinfo-module on device c-0-4
Creating srinfo-module on device s-1-7
Creating chinfo-module on device c-0-5
Creating srinfo-module on device s-1-8
Writable
Smart Insert
53 : 41 : 2266
141M of 256M


File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
DCNSFog.java
main.java
smartCarParkingFog.java
SmartMining.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
1 package org.fog.test.perfeval;
3 import java.util.ArrayList;
4 import java.util.Calendar;
5 import java.util.LinkedList;
6 import java.util.List;
7 import org.cloudbus.cloudsim.Host;
8 import org.cloudbus.cloudsim.Log;
9 import org.cloudbus.cloudsim.Pe;
10 import org.cloudbus.cloudsim.Storage;
11 import org.cloudbus.cloudsim.core.CloudSim;
12 import org.cloudbus.cloudsim.power.PowerHost;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:15:11 pm)
Creating srinfo-module on device s-0-7
Creating srinfo-module on device s-0-8
Creating srinfo-module on device s-0-9
Creating master-module on device a-1
Creating gasinfo-module on device g-1-0
Creating gasinfo-module on device g-1-1
Creating gasinfo-module on device g-1-2
Creating gasinfo-module on device g-1-3
Creating gasinfo-module on device g-1-4
Creating gasinfo-module on device g-1-5
Creating gasinfo-module on device g-1-6
Creating gasinfo-module on device g-1-7
Creating gasinfo-module on device g-1-8
Creating gasinfo-module on device g-1-9
Creating chinfo-module on device c-1-0
0.0 Submitted application mins
========== RESULTS ==========
Writable
Smart Insert
53 : 41 : 2266
142M of 256M


File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
package org.fog.test.perfeval;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.PowerHost;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:15:11 pm)
CH ---> 5.0
GAS ---> 2.099999999999909
SR ---> 5.0
cloud : Energy Consumed = 3205432.1420571427
proxy-server : Energy Consumed = 166866.59999999995
a-0 : Energy Consumed = 187922.26244599806
g-0-0 : Energy Consumed = 174754.6
g-0-1 : Energy Consumed = 174754.6
g-0-2 : Energy Consumed = 174754.6
g-0-3 : Energy Consumed = 174754.6
g-0-4 : Energy Consumed = 174754.6
g-0-5 : Energy Consumed = 174754.6
g-0-6 : Energy Consumed = 174754.6
g-0-7 : Energy Consumed = 174754.6
g-0-8 : Energy Consumed = 174754.6
g-0-9 : Energy Consumed = 174754.6
c-0-0 : Energy Consumed = 174722.02399999872
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar
Writable
Smart Insert
53 : 41 : 2266
144M of 256M

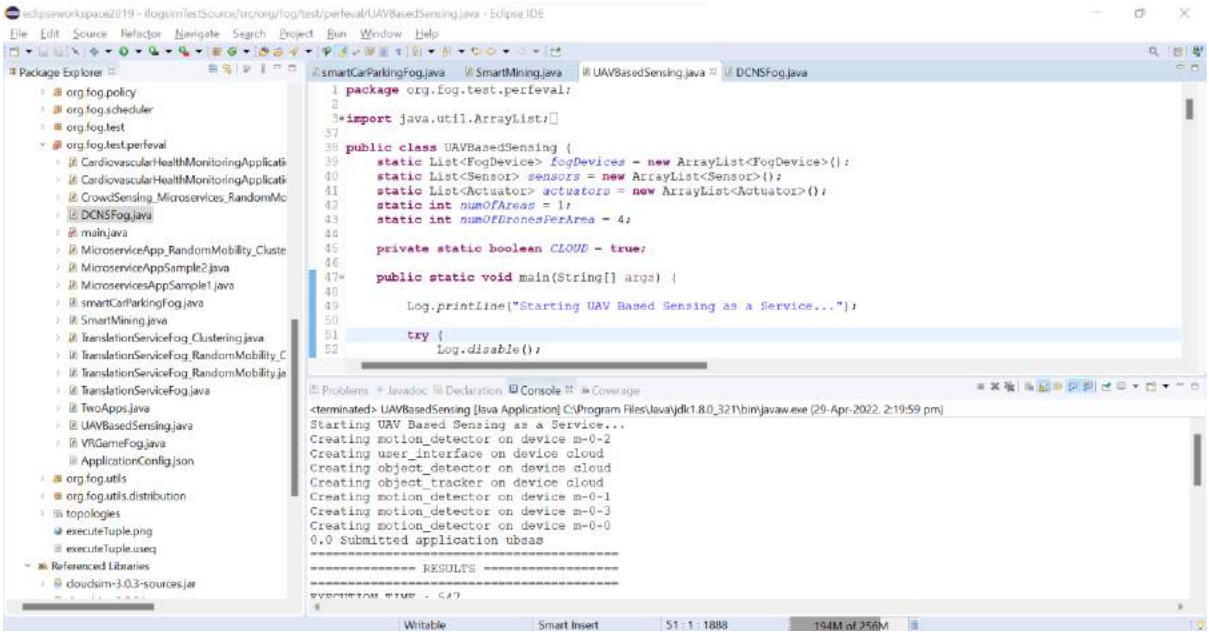

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
smartCarParkingFog.java
SmartMining.java
UAVBasedSensing.java
DCNSFog.java
package org.fog.test.perfeval;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.PowerHost;
Problems Javadoc Declaration Console Coverage
<terminated> SmartMining [Java Application] C:\Program Files\Java\jdk1.8.0_321\bin\javaw.exe (29-Apr-2022, 2:15:11 pm)
c-0-2 : Energy Consumed = 174722.02399999872
c-0-3 : Energy Consumed = 174722.02399999872
c-0-4 : Energy Consumed = 174722.02399999872
c-0-5 : Energy Consumed = 174722.02399999872
c-0-6 : Energy Consumed = 174722.02399999872
c-0-7 : Energy Consumed = 174722.02399999872
c-0-8 : Energy Consumed = 174722.02399999872
c-0-9 : Energy Consumed = 174722.02399999872
a-0-0 : Energy Consumed = 174722.02399999872
a-0-1 : Energy Consumed = 174722.02399999872
a-0-2 : Energy Consumed = 174722.02399999872
a-0-3 : Energy Consumed = 174722.02399999872
a-0-4 : Energy Consumed = 174722.02399999872
a-0-5 : Energy Consumed = 174722.02399999872
a-0-6 : Energy Consumed = 174722.02399999872
a-0-7 : Energy Consumed = 174722.02399999872
a-0-8 : Energy Consumed = 174722.02399999872
a-0-9 : Energy Consumed = 174722.02399999872
a-1 : Energy Consumed = 187922.26244599806
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
CardiovascularHealthMonitoringApplicati
CardiovascularHealthMonitoringApplicati
CrowdSensing_Microservices_RandomMc
DCNSFog.java
main.java
MicroserviceApp_RandomMobility_Cluste
MicroserviceAppSample2.java
MicroservicesAppSample1.java
smartCarParkingFog.java
SmartMining.java
TranslationServiceFog_Clustering.java
TranslationServiceFog_RandomMobility_C
TranslationServiceFog_RandomMobility.ja
TranslationServiceFog.java
TwoApps.java
UAVBasedSensing.java
VRGameFog.java
ApplicationConfig.json
org.fog.utils
org.fog.utils.distribution
topologies
executeTuple.png
executeTuple.useq
Referenced Libraries
cloudsim-3.0.3-sources.jar

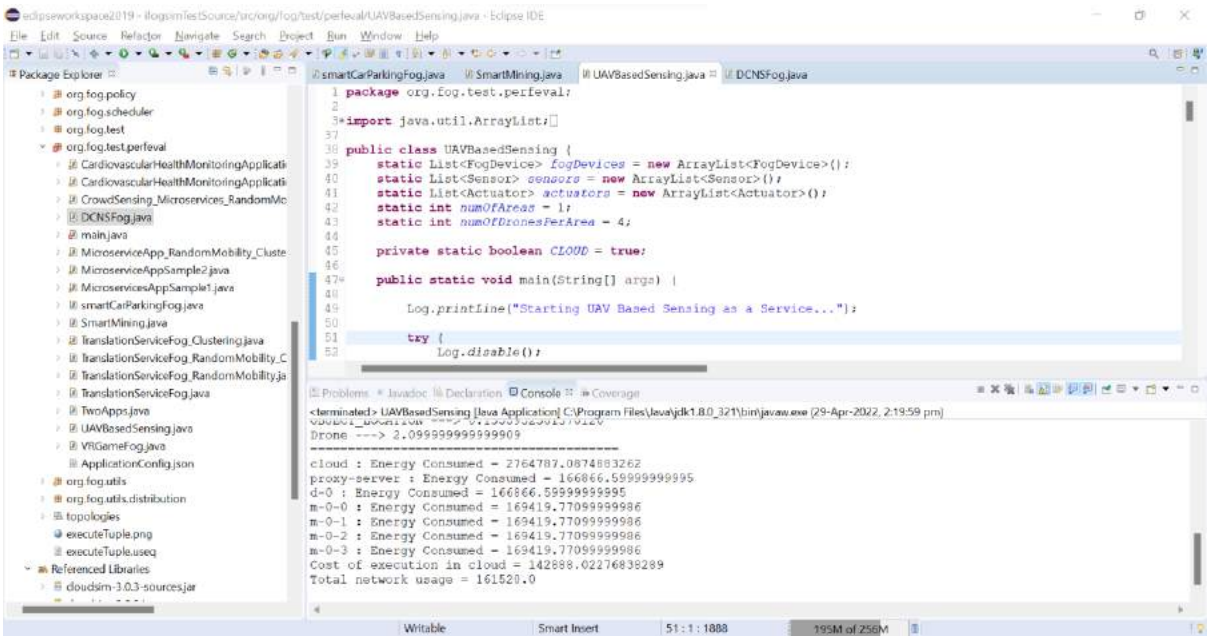




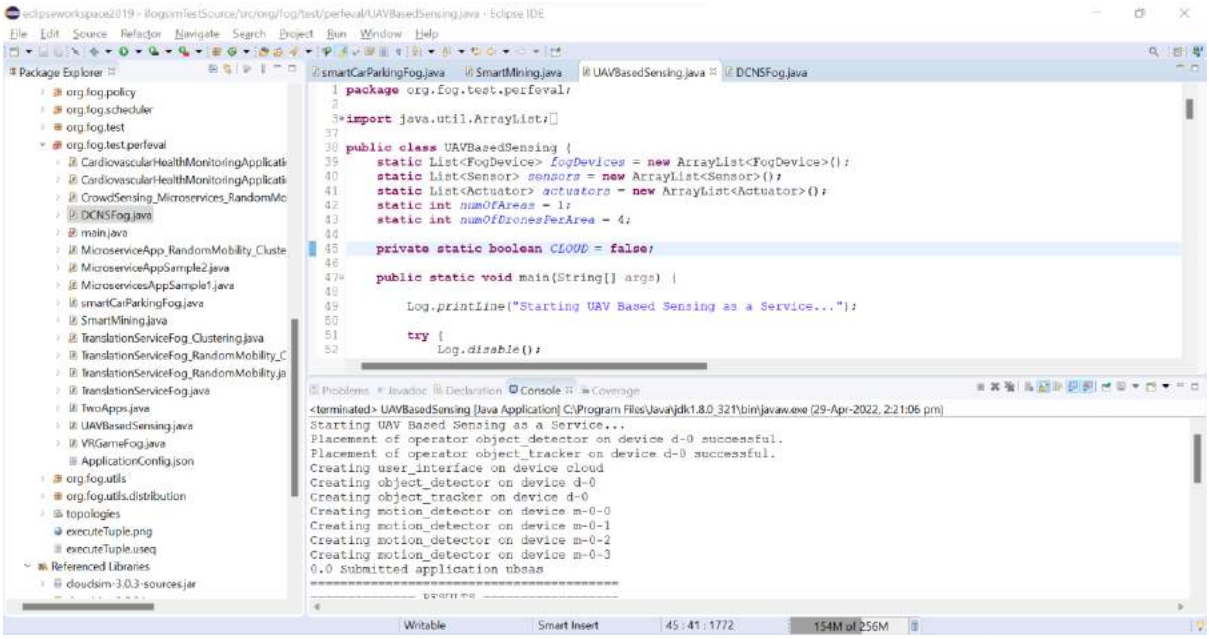
3. UAV based Sensing:

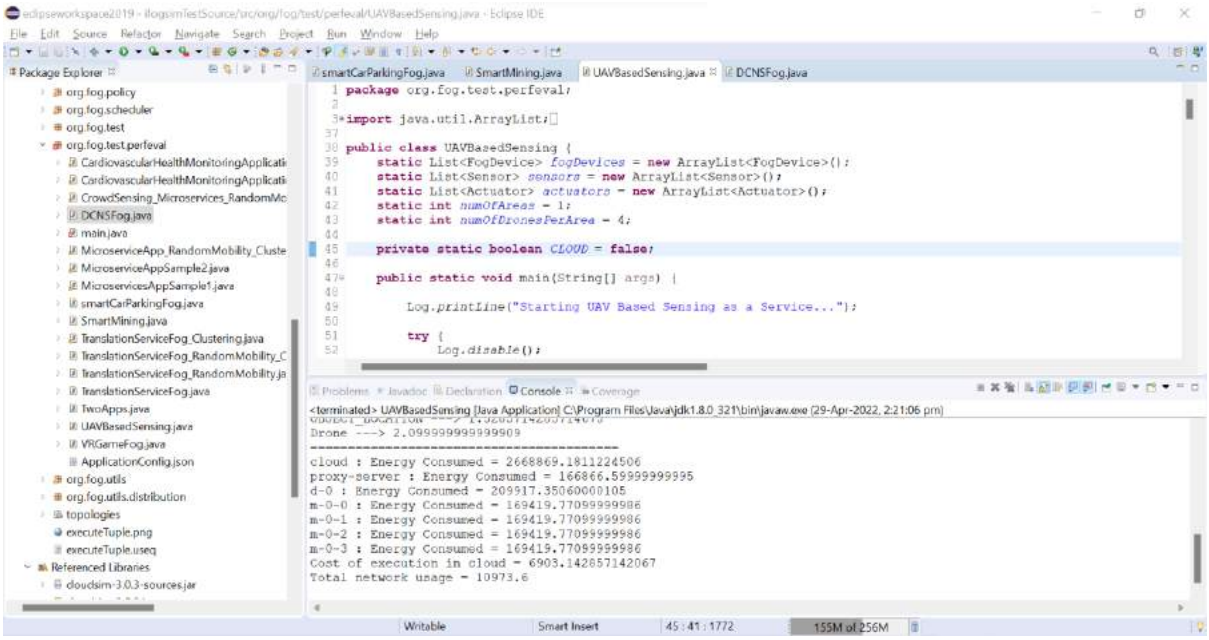
Simulation for Cloud:





For Fog and Edge Computing:





Discussion:

Analysis of Latency:

One thing common that we have noticed in the results of all the above experiments is that the latency is exponentially reduced when we have shifted from the cloud architecture to the fog one. For an instance you can see that the latency for the overall communication in cloud architecture for smart parking system was about 200ms which reduced to about 7ms when shifted to the fog architecture.

In contexts and situations demanding great real-time performance and optimization, latency must be minimised. Fog computing has the advantage of avoiding frequent cloud access and performing computations at the network's edge, resulting in a speedy response back to the client device and lowering latency. The data from the sensors and embedded systems are delivered to the fog nodes, which are located at the network's edge near the nodes, for processing. The reason why latency is so drastically reduced is because the one fog node is dedicated to a single area of network, and thus there is sufficient computational capacity to process the data of that area explicitly and locally before it is transferred to the cloud.

Analysis of Network Usage:

Another thing to notice is that the network consumed in cloud architecture is much greater than that in fog architecture. This is the result of increased bandwidth and better network utilization that fog helps us to achieve. For an instance the network used in fog is about 3112 units whereas in cloud it jumps to 194632 for smart parking system.

When traffic on a cloud server increases, only cloud resources are utilised. Increasing network utilisation arises from increased traffic on the cloud server. As a result of the increased traffic, the data rates on the network fall. For globally spread servers, one fog node is devoted to one geographical area to handle that area's requests. As a result, network consumption falls in that situation, but the transmission rate for the remaining traffic increases.

Conclusion:

With the advancement in new technologies like IOT and data science, applications and functionalities have improved a lot to put humankind on ease. High demanding extensive jobs have started to improve a lot in terms of performance and other important metrics. There is no doubt that IOT is going to be one of the most important and biggest discovery in past a decade. With the advancement in IOT and big data analysis, a new concept called Fog and Edge computing can be of great significance. Fog/Edge computing refers to processioned storing data near the node where the data is being generated. Fog provides us with multiple advantages including low latency, reduced volume of data, high responsiveness, better security, decent speed, multiple data sources and devices integration, mobility support, location awareness, decentralized and distributed architecture. In the above paper we tried to analyse and explore the role that Fog and Edge computing can play for common IOT applications, problems and functionalities and it would be safe to say that Fog and Edge computing is indeed one of the best architectural models for IOT implementation.

References:

1. AbrarFahima, MehediHasanbMuhtasim and AlamChowdhurya. Smart parking systems: comprehensive review based on various aspects.
2. C. Shobana Nageswari1, C.G. Sangeetha2 and V.B. Yogambigai. IoT based Smart Mine Monitoring System.
3. Mithra Sivakumar1, Naga Malleswari. A Literature Survey of Unmanned Aerial Vehicle Usage for Civil Applications.
4. T. Senthilkumaran, Mahantesh Mathapati. Smart garbage monitoring and clearance system using internet of things.
5. Ali Cherif Moussa, Hassine Moungla. Edge and fog computing for IoT: A survey on current research activities & future directions.
6.https://my.ece.msstate.edu/faculty/skhan/pub/A_K_2021_BC.pdf
7.https://www.researchgate.net/publication/328895359_Towards_Smart_Parking_Based_on_Fog_Computing
8.https://www.researchgate.net/publication/336977279_Towards_a_Fog_Enabled_Efficient_Car_Parking_Architecture