IoTSim-Osmosis User Manual

Table of Contents

1 Explanation of IoTSim-Osmosis

1.1 What is IoTSim-Osmosis?

Osmotic computing paradigm sets out the principles and algorithms for simplifying the deployment of Internet of Things (IoT) applications in integrated edge-cloud environments. Osmotic Computing focuses on strategies and mechanisms to extend the IoT capabilities by defining, designing, and implementing a modern computing model (IoT, edge, cloud, and SD-WAN).

IoTSim-Osmosis is a simulation framework that supports the testing and validation of osmotic computing applications. In particular, it enables a unified modelling and simulation of complex IoT applications over heterogeneous edge-cloud SDN-aware environments. IoTSim-Osmosis is capable of capturing the key functions, characteristics, and behaviors of osmotic paradigm. A wide range of osmosis applications can be simulated and evaluated in IoTSim-Osmosis.

For further details of IoTSim-Osmosis, please refer to our paper entitled "IoTSim-Osmosis: A Framework for Modelling & Simulating IoT Applications over an Edge-Cloud Continuum".

1.2 Unique Features of IoTSim-Osmosis

IoTSim-Osmosis is developed to allow such hybrid infrastructures to be simulated. The dynamic management and performance metrics of IoT-oriented services across edge and cloud datacentres that communicate via SDWAN are easily achieved. In particular, IoTSim-Osmosis is capable of modeling and simulating:

- Osmotic applications running between edge and cloud
- The behaviors and features of osmotic applications running in dynamic SDN and SD-WAN networks;
- Dynamic routing mechanisms based on graph theory to enable any type of network topology to be seamlessly simulated;
- Several policies for SDN, SD-WAN, and MEL, VM for multilevel optimization**.**

2 Getting Started

2.1 Lifecycle of IoTSim-Osmosis

The overall architecture of osmotic computing in IoTSim-Osmosis is divided into four main layers: input, management, osmotic orchestrator, and infrastructure. IoTSim-Osmosis requires two input files to start running. First, it requires end-to-end configuration file, which includes a detailed requirement of every infrastructure element. For example, it requires an attributes of IoT device (e.g., name, bandwidth, battery capacity). When IoTSim-Osmosis finishes building the required infrastructures, it would require an IoT-MEL graph workload file. The workload contains a journey description of every IoT transaction. The transaction represents a single unit of logic for each IoT generated data (refer to IoTSim-Osmosis paper for more information). The transaction also contains several MEL and network operations. Each transaction can have different performance, which can be used to evaluate the performance of a given osmotic application.

2.2 System and Software Requirements

- Operating System: Windows, Linux or Mac OS.
- CPU: 1-GHz processor or equivalent (Minimum).
- RAM: 2GB (Minimum).
- Java Platform: JDK version 11+ (recommended)
- Any IDE for Java programming language such as Eclipse or NetBeans

2.3 Download IoTSim-Osmosis

IoTSim-Osmosis can be downloaded from https://github.com/kalwasel/IoTSim-Osmosis

2.4 Directory Structure of IoTSim-Osmosis

The structure of IoTSim-Osmosis framework is defined as follows:

- IoTSim-Osmosis/
- examples/ -- Contains examples of osmotic applications
- sources/ -- Contains the source code of IoTSim-Osmosis
- inputFiles/ -- Contains the required files to be submitted to IoTSim-Osmosis
- outputFiles/ -- Contains all the output results of IoTSim-Osmosis

2.5 Main Packages of IoTSim-Osmosis

IoTSim-Osmosis is mainly developed using the following package list:

1. org.cloudbus.cloudsim.edge.core.edge
2. org.cloudbus.cloudsim.edge.iot
3. org.cloudbus.cloudsim.edge.iot.network
4. org.cloudbus.cloudsim.edge.iot.protocol
5. org.cloudbus.cloudsim.sdn
6. org.cloudbus.cloudsim.sdwan
7. org.cloudbus.osmosis.core
8. org.cloudbus.osmosis.core.polocies

Package 1, 2, 3, and 4 contains classes that models the behaviors and characteristics of IoT and edge datacenters. Package 5 and 6 contains classes that models the behaviors and characteristics of SDN and SD-WAN networks. Package 7 contains classes that models the behaviors and characteristics of osmotic applications. Package 8 contains classes that models a list of osmotic policies, such as SDN routing policy. Figure 2.1 shows the packages and their classes in detail.

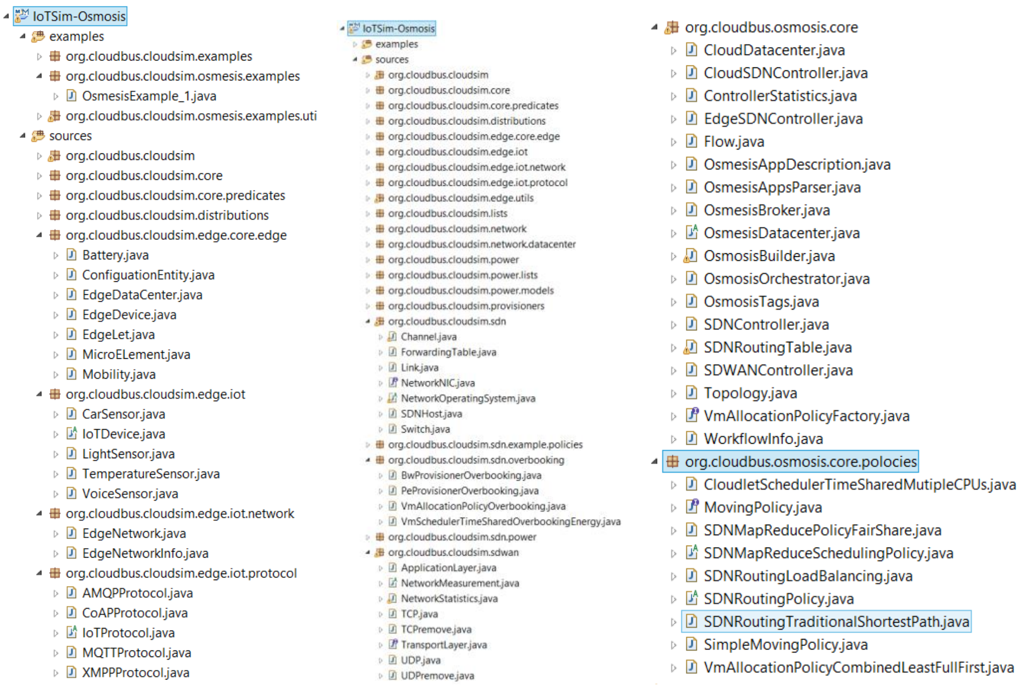


Figure 2.1

2.6 Setup IoTSim-Osmosis

Prior to use IoTSim-Osmosis, you need to import and configure the project properly. Here, we use Eclipse to illustrate how to setup the IoTSim-Osmosis project. The project is based on Maven. The main steps are given as follows:

Step 1:

- Install Eclipse from https://www.eclipse.org/downloads/
- Install Maven on Eclipse, follow the steps given in https://www.eclipse.org/m2e/

Step 2: Import IoTSim-Osmosis as a Maven project by Opening Eclipse -> selecting File -> and selecting import (see Figure 2.2)

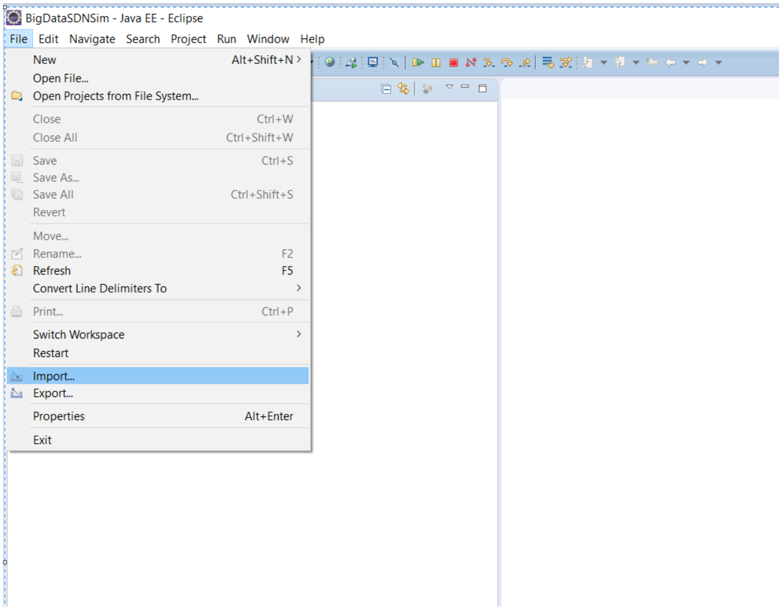


Figure 2.2

Step 3: Select Maven -> select Existing Maven Projects (see Figure 2.3)

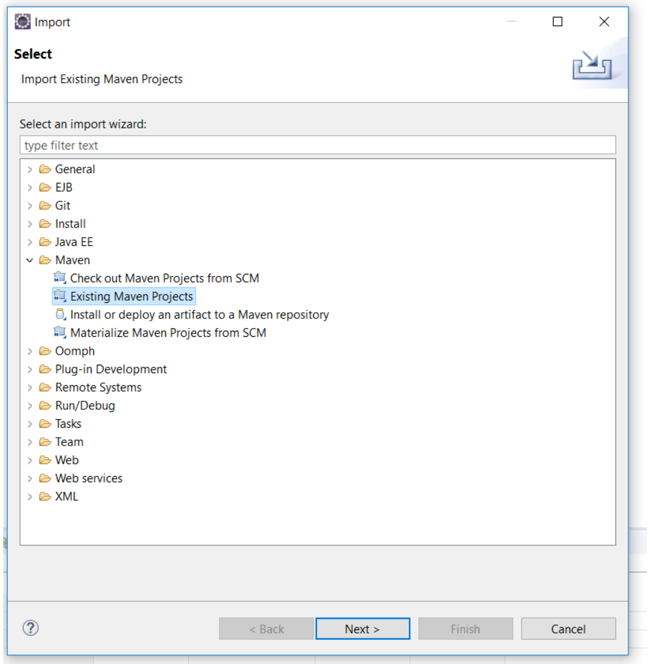


Figure 2.3

Step 4: Select the folder corresponding to IoTSim-Osmosis project. Next, click on Finish (see Figure 2.4)

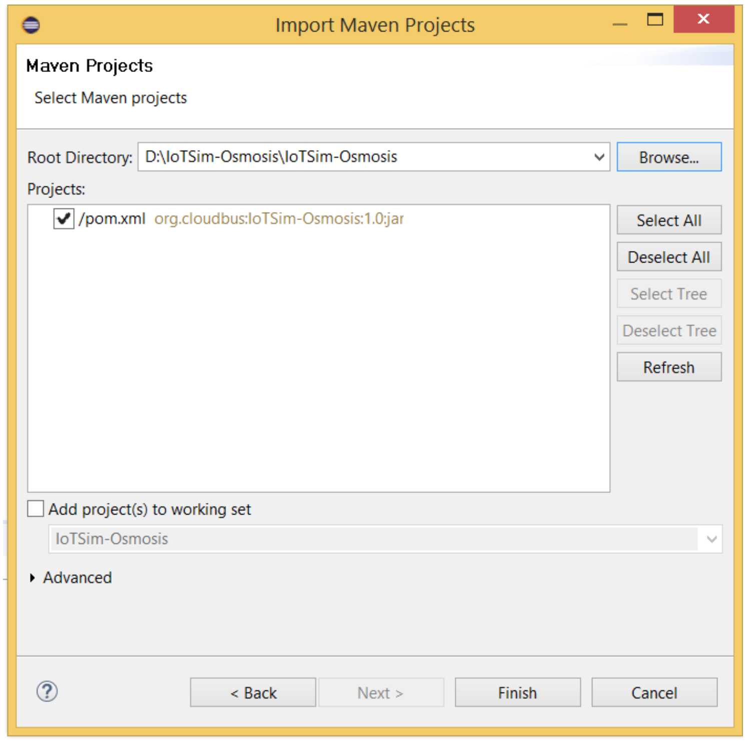


Figure 2.4

Step 5: Right click on IoTSim-Osmosis project and click on Update Project under Maven option (see Figure 2.5)

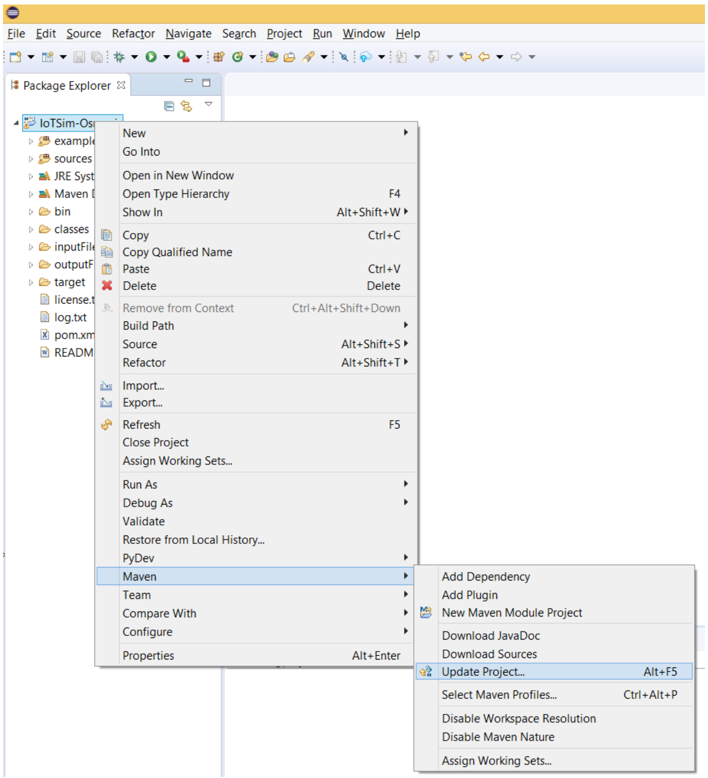


Figure 2.5

Step 6: Right click on IoTSim-Osmosis project and click on Maven install that found under Run As option (see Figure 2.6)

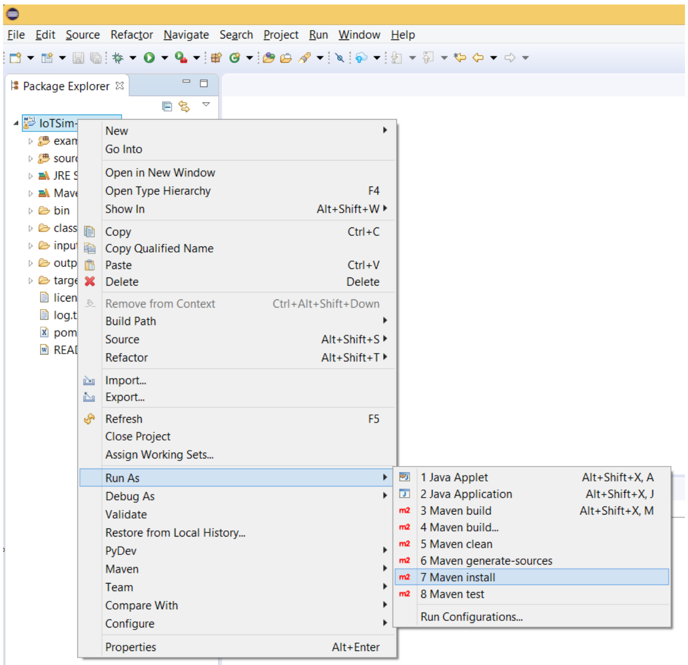


Figure 2.6

When Maven successfully builds IoTSim-Osmosis in your Eclipse, you will see “BUILD SUCCESS” as shown in Figure 2.7. At this point, you have successfully built and configured IoTSim-Osmosis.

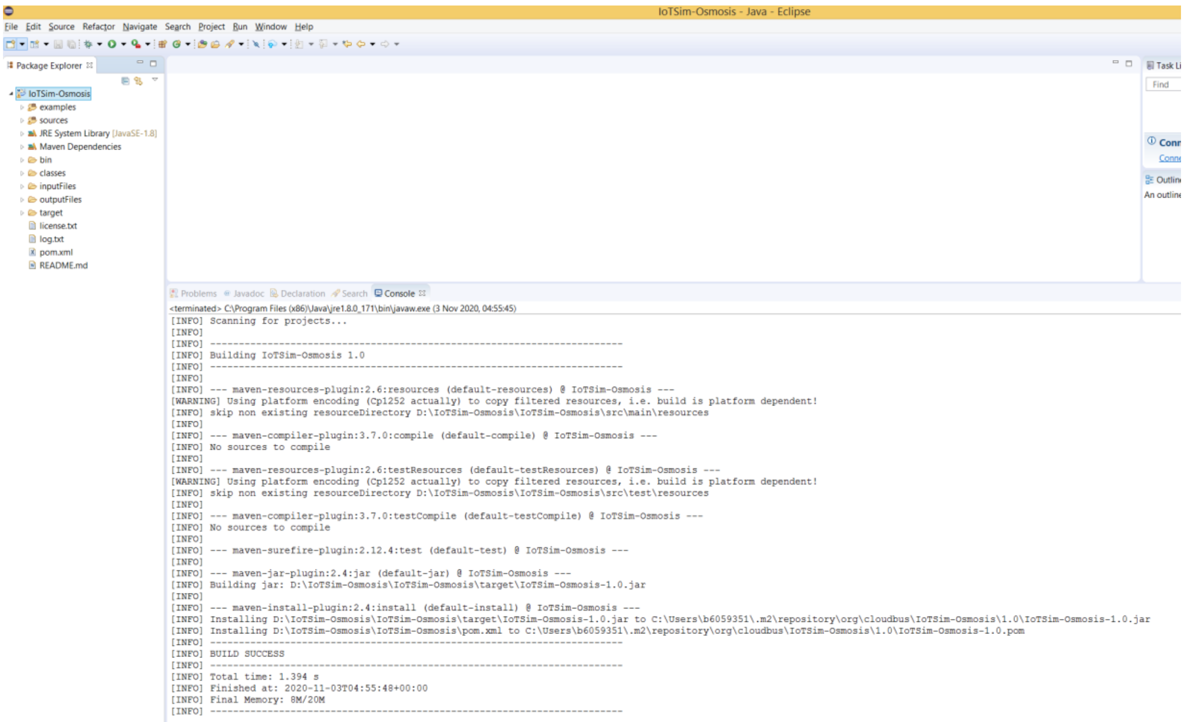


Figure 2.7

Step 7: IoTSim-Osmosis uses Lombok library to configure its entities. Open Maven Dependencies directory, right click on Lombok*.jar file, Run As Java file and then follow the instructions (see Figure 8). Refer to YouTube Lombok tutorial if you encounter problems.

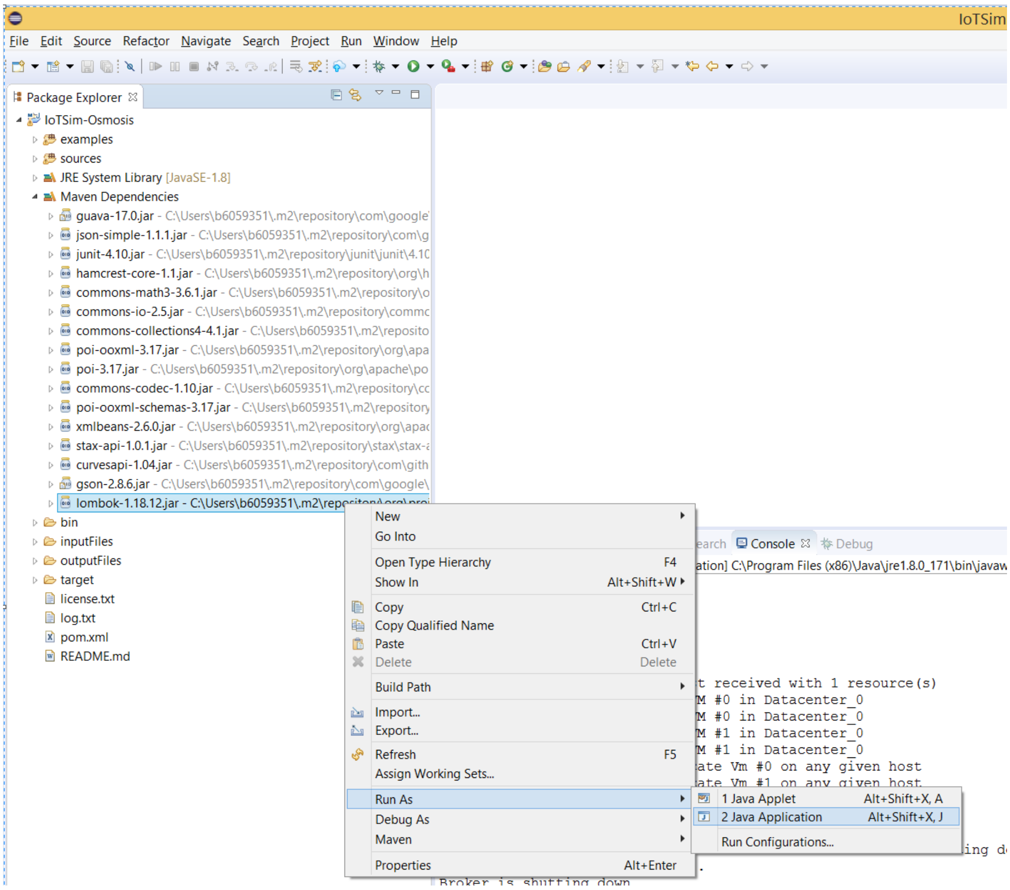


Figure 8

3 Simulation configuration

Before starting the actual simulation, you have to configure the infrastructure of every data center (cloud and edge) and SD-WAN network that connects distributed datacenters. The infrastructure of every data center can be easily configured using a configuration file named **Example1_configuration** in the *inputFiles* folder. The parameters of the configuration file is illustrated in Table 3.1, which is defined in a JSON format. These parameters are read during initialization, which configures the environment of IoTSim-Osmosis accordingly. A snapshot of the confirmation file is given in Figure 3.1.

Table 3.1 User-defined of physical configuration (IoT, edge, cloud, and SD-WAN)

	Entity	Parameter	Description
Edge	Information	Name	Datacenter's name
		Type	Datacenter's type (e.g. edge, cloud)
		vmAllocationPolicy	Policy to allocate MELs among edge devices (hosts)
	Characteristics	The parameters of datacenters' characteristics can be left with no change. Refer to a CloudSim project if you are interested to tune the parameters (e.g. cost of network bandwidth, etc.). For IoTSim-Osmosis, such parameters are not important.	
	Hosts	name	Edge device's name
		pes	Number of CPUs

		ramSize	Size of host's memory
		bwSize	Speed of NIC interface
		storage	Size of host's storage
		mips	Million Instructions Per Second (MIPS) represents the speed of host's CPU
	MELEntities	name	Name of MEL
		bw	Speed of MEL network
		mips	Million Instructions Per Second (MIPS) represents the speed of MEL's CPU
		ram	Size of MEL's memory
		pesNumber	Number of CPUs
		cloudletSchedulerClassName	Policy to execute tasks in MELs
	Controllers (SDN)	name	Name of SDN controller
		trafficPolicy	Policy to control network traffic among applications
		routingPolicy	Policy to find routes from sources to destinations
	switches	name	Name of switch
		type	Type of switch (gateway, core, etc.)
		controller	A corresponding SDN controller to communicate with
		iops	Speed of I/O
	links	source	A source element (e.g. MEL)
		destination	A destination element (e.g. VM)
		bw	Speed of a link
	ioTDevices	name	Name of an IoT device
		bw	Speed of IoT's NIC
		max_battery_capacity	Battery capacity
		battery_sensing_rate	Battery consumption for sensing
		battery_sending_rate	Battery consumption for sending
		ioTClassName	Type of IoT device (e.g. temperature sensor)
		mobilityEntity	Mobility information
		communicationProtocol	IoT protocol (e.g. XMPP)

<table>
<tr><td></td><td></td><td>networkType</td><td>Type of network connection (e.g. WiFi)</td></tr>
<tr><td rowspan="5">Cloud</td><td>hosts</td><td colspan="2" rowspan="5">Cloud datacenters have similar parameters’ definitions as edge datacenters (e.g. hosts, switches)</td></tr>
<tr><td>VMs</td></tr>
<tr><td>controllers</td></tr>
<tr><td>switches</td></tr>
<tr><td>links</td></tr>
<tr><td rowspan="3">SDWAN</td><td>controllers</td><td colspan="2" rowspan="3">SDWAN network has similar parameters’ definitions as datacenter networks (e.g. controllers, links)</td></tr>
<tr><td>switches</td></tr>
<tr><td>links</td></tr>
</table>


```
{
  "edgeDatacenter":[
   {
      "name":"Edge_1",
      "type":"edge",
      "schedulingInterval":1.0,
      "vmAllocationPolicy":{
         "className":"VmAllocationPolicyCombinedLeastFullFirst"
      },
      "characteristics":{
         "costPerMem":0.05,
         "cost":1.0,
         "os":"Linux",
         "costPerSec":0.0,
         "vmm":"Xen",
         "timeZone":10.0,
         "costPerBw":0.0,
         "costPerStorage":0.001,
         "architecture":"x86"
      },
      "hosts":[
         {
            "name":"edgeDevice_1",
            "pes" : 4,
            "ramSize":10000,
            "bwSize":100,
            "storage":10000,
            "mips":10000
         },
         {
            "name":"edgeDevice_2",
            "pes" : 4,
            "ramSize":10000,
            "bwSize":100,
            "storage":10000,
            "mips":10000
         }
         ],
      "MELEntities":[
      {
         "name":"MEL_1",
         "bw":100,
         "mips":250,
         "ram":10000,
         "pesNumber":1,
         "vmm":"xxx",
         "cloudletSchedulerClassName":"org.cloudbus.cloudsim.CloudletSchedulerTimeShared"
      },
```

```
"ioTDevices":[
      {
        "name":"temperature_1",
        "bw":100,
        "max_battery_capacity":100.0,
        "battery_sensing_rate":0.001,
        "battery_sending_rate":0.001,
        "ioTClassName":"org.cloudbus.cloudsim.edge.iot.TemperatureSensor",
         "mobilityEntity":{
            "movable":false,
            "location":{
               "x":0.0,
               "y":0.0,
               "z":0.0
            }
         },
         "networkModelEntity":{
            "communicationProtocol":"xmpp",
            "networkType":"wifi"
         }
      },
      {
        "name":"temperature_2",
        "bw":100,
        "max_battery_capacity":100.0,
        "battery_sensing_rate":0.001,
        "battery_sending_rate":0.001,
        "ioTClassName":"org.cloudbus.cloudsim.edge.iot.TemperatureSensor",
         "mobilityEntity":{
            "movable":false,
            "location":{
               "x":0.0,
               "y":0.0,
               "z":0.0
            }
         },
         "networkModelEntity":{
            "communicationProtocol":"xmpp",
            "networkType":"wifi"
         }
      },
```


Figure 3.1

4 Simulation examples

Before starting the actual simulation, the processing and transmission logic of every osmotic application must be provided. The current version of IoTSim-Osmosis has one application example for illustration. The logic of any osmotic applications can be easily configured using a CSV file named **Example1_Worload** in the *inputFiles* folder. The description of every parameter is given in Table 4.1. A snapshot of the file is given in Figure 4.1.

Table 4.1 The parameter description of MapReduce applications

Parameter	**Description**
OsmesisApp	App name (e.g. App_1)
ID	App ID
DataRate_Sec	Data generation rate (e.g. every one second)
StopDataGeneration_Sec	Time to stop generating data
IoTDevice	The name of IoT device (e.g. temperature_1). Note that IoT devices’ name must be defined in the infrastructure file (Example1_configuration)

IoTDeviceOutputData_Mb	The data size in Mb an IoT device generates according to the data generation rate (e.g. sending 100 Mb every 2 seconds)
MELName	MEL destination that receives data from an IoT device
OsmesisEdgelet_MI	Size of a task in million instructions (MI), which is executed in a MEL
MELOutputData_Mb	Every MEL generates new data in Mb (e.g. filtering, sorting, calculating) and sends the data to a VM in cloud datacentre via SD-WAN network layer
VmName	VM's name that receives data from a MEL
OsmesisCloudlet_MI	Finally, VM processes received data in a form of MI

A	B	C	D	E	F	G	H	I	J	K
OsmesisApp	ID	DataRate_Sec	StopDataGeneration_Sec	IoTDevice	IoTDeviceOutputData_Mb	MELName	OsmesisEdgelet_MI	MELOutputData_Mb	VmName	OsmesisCloudlet_MI
App_1	1	1.2	300	temperature_1	90	MEL_1	250	70	VM_1	200

Figure 4.1 Example1_Worload

The example can be found in org.cloudbus.cloudsim.osmesis.examples package, as shown in Figure 4.2.

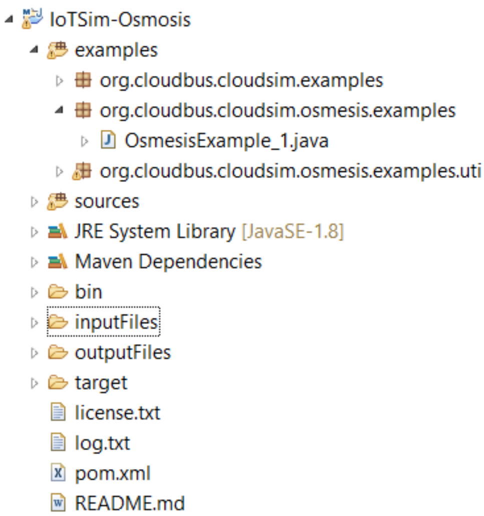


Figure 4.2

4.1 Policies

To run any osmotic application, you have to provide, select, or propose several policies. Figure 4.3 shows the list of policies already developed in IoTSim-Osmosis or used from different simulation tools (e.g. BigDataSDNSim). The policy selection can be configured either in the infrastructure file (Example1_configuration) or in the OsmesisExample_1.java. The following describes the purpose of every policy:

1. *VM and MEL placement:* It determines how VMs and MELs are placed on a given edge and cloud hosts (e.g. VmMELAllocationPolicyCombinedLeastFullFirst).
2. *VM-CPU scheduling:* It determines how tasks submitted to VMs or MELs are scheduled (e.g. CloudletSchedulerTimeSharedMutipleCPUs)
3. *Routing:* It is used to determine routes among MEL and/or VMs in edge, cloud and SD-WAN networks (e.g. SDNRoutingTraditionalShortestPath).
4. *Traffic:* It is used to control the sharing of resources among osmotic applications in given networks (e.g. edge, cloud, SD-WAN). An example is SDNTrafficPolicyFairShare.
5. *Moving:* It is used to determine the movement policy of given IoT devices. Current version of IoTSim-Osmosis does not require this policy as it considers IoT devices to be in a fixed location.

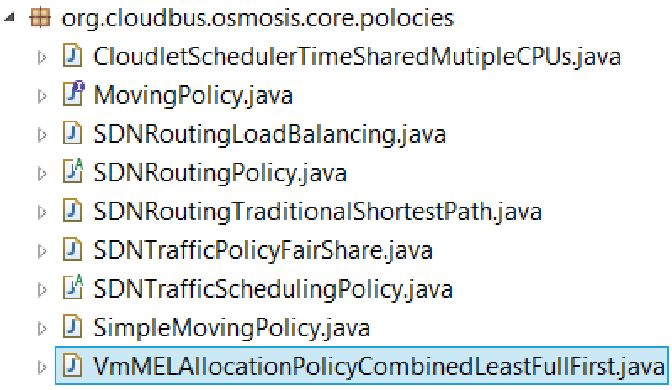


Figure 4.3

4.2 Output results of IoTSim-Osmosis

Once IoTSim-Osmosis finishes running, it would produce results, which are stored in a text file named result.txt located in outputFiles folder. The results contain a lot of information. At the end of the result file, the results are structured as follows:

- Results of each IoT transaction
- Results of every osmotic application
- Battery consumption of IoT devices
- Power consumption of edge, cloud, and SD-WAN infrastructures
- Total power consumption

4.3 Running an example (OsmesisExample_1)

Step 1: Select OsmesisExample_1.java -> click on the small down arrow next to the play button and select Run Configurations (see Figure 4.4).

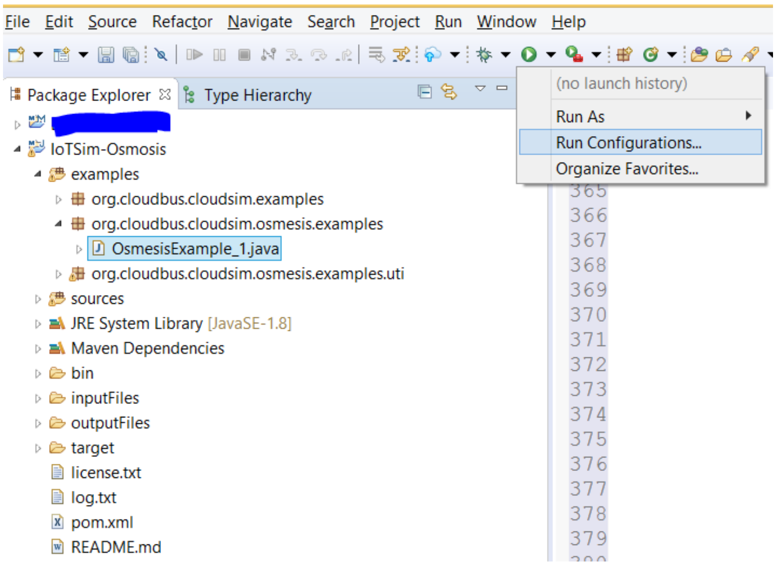


Figure 4.4

Step 2: Double click on Java Application and Eclipse will create the first example automatically (see Figure 4.5).

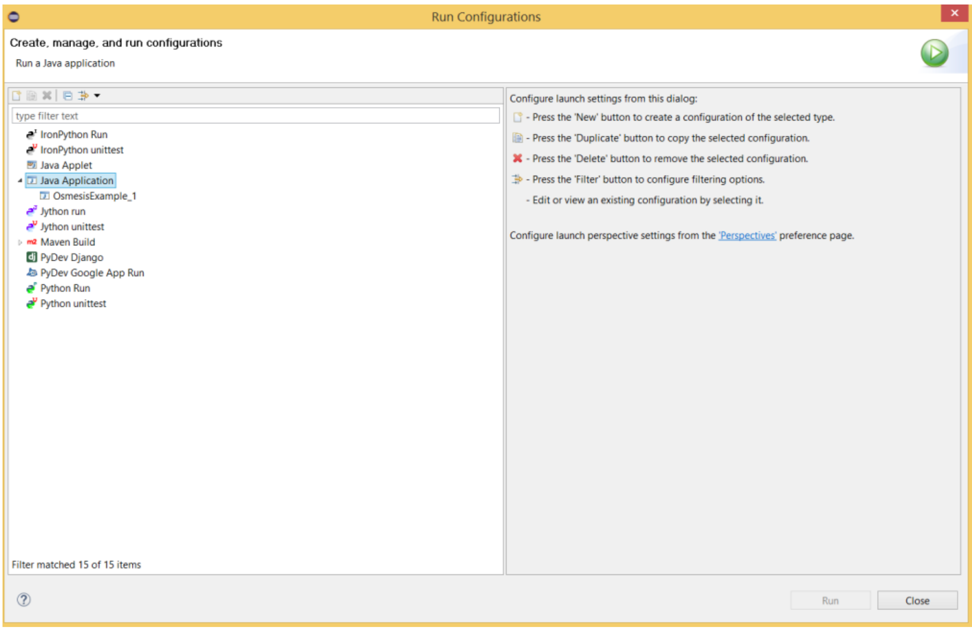


Figure 4.5

Step 2: Click on Common -> check mark Output File -> click workspace -> select IoTSim-Osmosis folder -> select outFiles folder -> select result.txt -> click Ok -> click Run (see Figure 4.6). This step will run the first example and store all the outputs on the result.txt file.

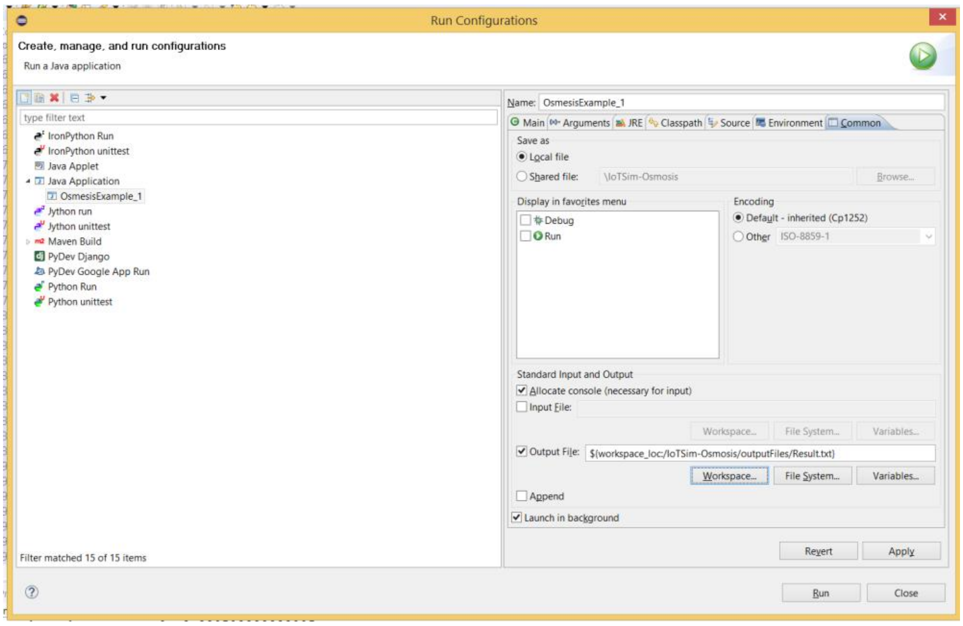
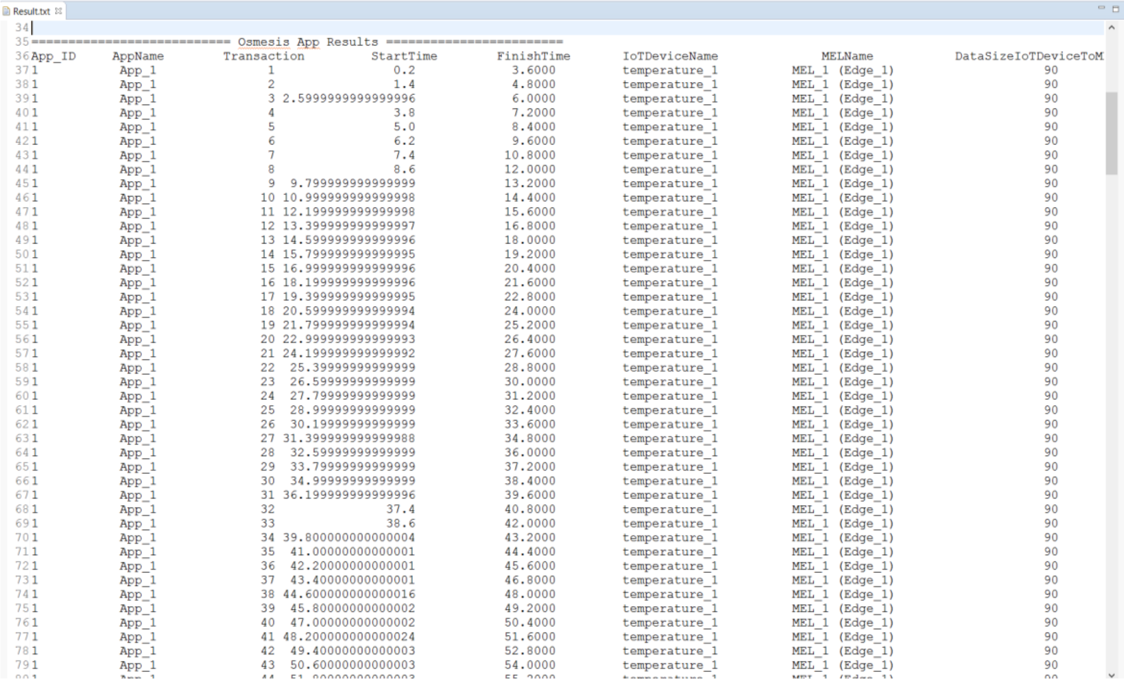


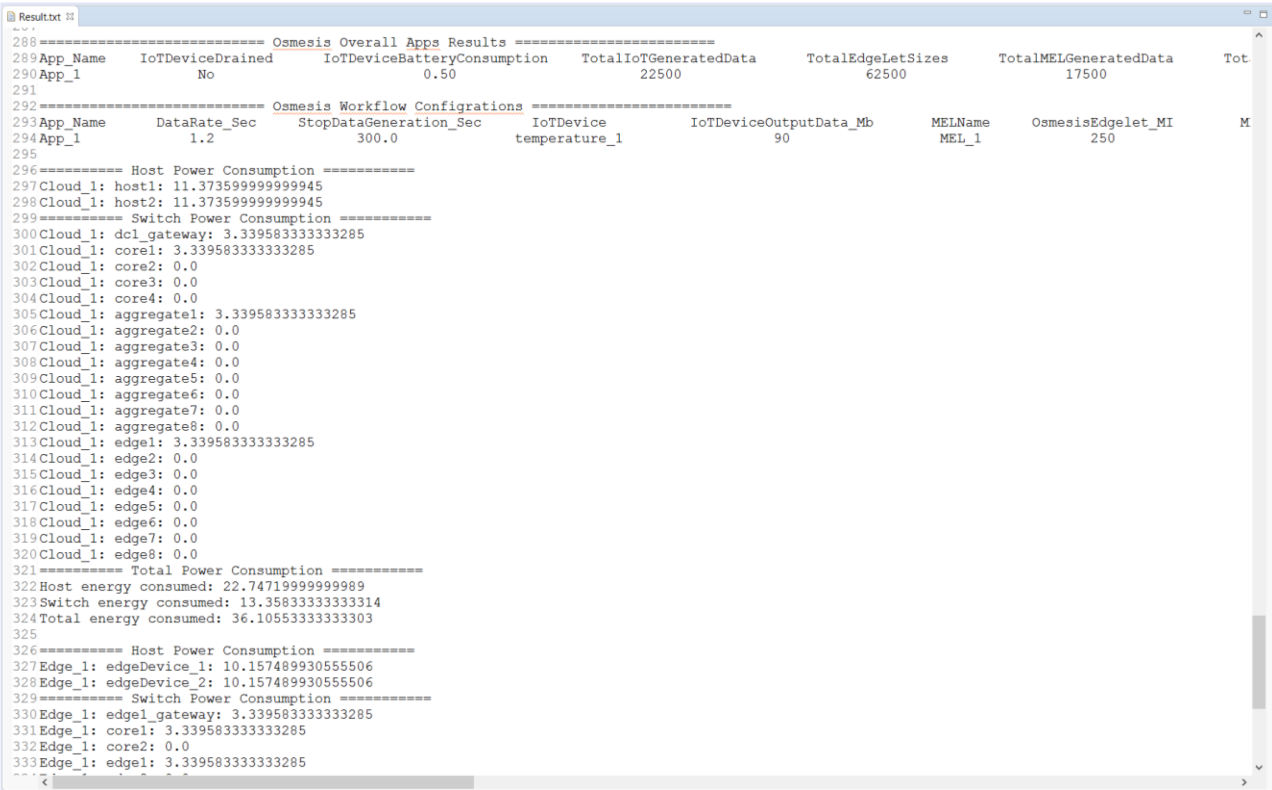
Figure 4.6

Figure 4.7 and Figure 4.8 show a sample of the result obtained by running the example.



```
========================== Osmesis App Results ========================
App_ID    AppName    Transaction    StartTime    FinishTime    IoTDeviceName    MELName    DataSizeIoTDeviceToM
1    App_1    1    0.2    3.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    2    1.4    4.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    3 2.5999999999999996    6.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    4    3.8    7.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    5    5.0    8.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    6    6.2    9.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    7    7.4    10.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    8    8.6    12.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    9  9.799999999999999    13.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    10 10.999999999999998    14.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    11 12.199999999999998    15.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    12 13.399999999999997    16.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    13 14.599999999999996    18.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    14 15.799999999999995    19.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    15 16.999999999999996    20.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    16 18.199999999999996    21.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    17 19.399999999999995    22.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    18 20.599999999999994    24.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    19 21.799999999999994    25.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    20 22.999999999999993    26.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    21 24.199999999999992    27.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    22  25.39999999999999    28.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    23  26.59999999999999    30.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    24  27.79999999999999    31.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    25  28.99999999999999    32.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    26  30.19999999999999    33.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    27 31.399999999999988    34.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    28  32.59999999999999    36.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    29  33.79999999999999    37.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    30  34.99999999999999    38.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    31 36.199999999999996    39.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    32    37.4    40.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    33    38.6    42.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    34 39.800000000000004    43.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    35  41.00000000000001    44.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    36  42.20000000000001    45.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    37  43.40000000000001    46.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    38 44.600000000000016    48.0000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    39  45.80000000000002    49.2000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    40  47.00000000000002    50.4000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    41 48.200000000000024    51.6000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    42  49.40000000000003    52.8000    temperature_1    MEL_1 (Edge_1)    90
1    App_1    43  50.60000000000003    54.0000    temperature_1    MEL_1 (Edge_1)    90
```

Figure 4.7



```
=========================== Osmesis Overall Apps Results ========================
App_Name    IoTDeviceDrained    IoTDeviceBatteryConsumption    TotalIoTGeneratedData    TotalEdgeLetSizes    TotalMELGeneratedData    Tot
App_1    No    0.50    22500    62500    17500

=========================== Osmesis Workflow Configrations ========================
App_Name    DataRate_Sec    StopDataGeneration_Sec    IoTDevice    IoTDeviceOutputData_Mb    MELName    OsmesisEdgelet_MI    M
App_1    1.2    300.0    temperature_1    90    MEL_1    250

========== Host Power Consumption ===========
Cloud_1: host1: 11.373599999999945
Cloud_1: host2: 11.373599999999945
========== Switch Power Consumption ===========
Cloud_1: dc1_gateway: 3.339583333333285
Cloud_1: core1: 3.339583333333285
Cloud_1: core2: 0.0
Cloud_1: core3: 0.0
Cloud_1: core4: 0.0
Cloud_1: aggregate1: 3.339583333333285
Cloud_1: aggregate2: 0.0
Cloud_1: aggregate3: 0.0
Cloud_1: aggregate4: 0.0
Cloud_1: aggregate5: 0.0
Cloud_1: aggregate6: 0.0
Cloud_1: aggregate7: 0.0
Cloud_1: aggregate8: 0.0
Cloud_1: edge1: 3.339583333333285
Cloud_1: edge2: 0.0
Cloud_1: edge3: 0.0
Cloud_1: edge4: 0.0
Cloud_1: edge5: 0.0
Cloud_1: edge6: 0.0
Cloud_1: edge7: 0.0
Cloud_1: edge8: 0.0
========== Total Power Consumption ===========
Host energy consumed: 22.74719999999989
Switch energy consumed: 13.35833333333314
Total energy consumed: 36.10553333333303

========== Host Power Consumption ===========
Edge_1: edgeDevice_1: 10.157489930555506
Edge_1: edgeDevice_2: 10.157489930555506
========== Switch Power Consumption ===========
Edge_1: edge1_gateway: 3.339583333333285
Edge_1: core1: 3.339583333333285
Edge_1: core2: 0.0
Edge_1: edge1: 3.339583333333285
```

Figure 4.8

5 Contact

Please feel free to contact me if you need any further information at kalwasel@gmail.com
This manual is written by Khaled Alwasel under the supervision of Prof. Rajiv Ranjan.