IoTSim-Edge User Manual

Contents

1 What is IoTSim-Edge

Edge computing is new paradigm that offers processing and storage of IoT data near the generation point. It provide multiple advantages for response intensive applications. Also it reduces the network bandwidth by filtering and processing the raw data and sending the data to cloud only if required for further processing and storage.

Simulating and modeling a realistic IoT scenario is very challenging due to various reasons such as (i) Variety of IoT devices need to be combined with edge device and cloud to satisfy the requirements of an application; (ii) Modeling networking graph between diverse type of IoT and edge computing device in an abstract manner can be very challenging; (iii) Modeling data and control flow dependencies between IoT and Edge layers to support diverse data analysis work-flow structure is non-trivial; (iv) capacity planning across edge computing layer is challenging as it depends on various configuration parameters including data volume, data velocity, upstream/downstream network bandwidth, to name a few; (v) The communication between IoT and edge device is very different from cloud datacenter communication, which are generally based on wired and/or wireless protocol. The connectivity between IoT and edge computing layers, as we discuss later in the paper, can be diverse. Hence, they are very difficult to model in an abstract way while not loosing the expressiveness, i.e. lower level details related to protocol latency, impact of protocol on battery discharge rate of underlying IoT device, etc; (vi) Mobility is another important parameter of IoT devices as sensors embedded to many physical devices are moving. Since the range of edge device is limited, the movement of sensor may leads to handoff. Also, the data sent to an edge device for processing may not be in the current range of IoT device. Thus for receiving the processed data, an edge to edge communication is required. Modeling the mobility and handoff for large number of IoT devices with varying velocity is very challenging; (vii) Dynamicity of IoT environment leads to addition and removal of IoT and edge devices very frequently. This may be caused by numerous factors e.g. device failure, network link failure. Modeling the scalability of IoT devices with heterogeneous features at a fast rate is very challenging; and (viii) Since IoT environment is an emerging area, new applications might be developed in future. It is very important for a simulator to allow users to customize and extend the framework based on their requirements. Making a general simulator that allows easy customization is very challenging.

To simulate the features of IoT and edge environment that resolves all the above challenges, we propose IoTSim-Edge simulator. IoTSim-Edge simulator models the distribution and processing of streaming data generated by IoT devices in Edge computing environment. Our proposed simulator captures the behaviour of heterogeneous IoT and edge computing infrastructure and allows user to test their proposed infrastructure, applications and algorithms in an easy and configurable manner.

For technical detail about IoTSim-Edge, please refer to our paper entitled “IoTSim–Edge: A simulation framework for modeling the behaviour of IoT and edge computing environments”.

2 Unique Features

IoTSim-Edge is able to model the following scenarios:

- New IoT application graph modeling abstraction that allows practitioners to define the data analytic operations and their mapping to different parts of the infrastructure (e.g. IoT and edge).

- Abstraction that supports modeling of heterogeneous IoT protocols along with their energy consumption profile. It allows practitioners to define the configuration of edge and IoT devices along with the specific protocols they support for networking.

- Abstraction that supports modeling of mobile IoT devices. It also captures the effect of handoff caused by the movement of IoT devices. To maintain a consistent communication, IoTSim–Edge supports a cooperative edge-to-edge communication that allows the transfer the processed data of the respective IoT device by one edge via another edge.

3 Getting Started

3.1 System and Software Requirements

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

3.2 Download IoTSim-Edge

The simulation toolkit (IoTSim-Edge) can be downloaded from https://github.com/DNJha/IoTSIM.

3.3 Directory Structure of IoTSim-Edge

The structure of IoTSim-Edge package is as follows:

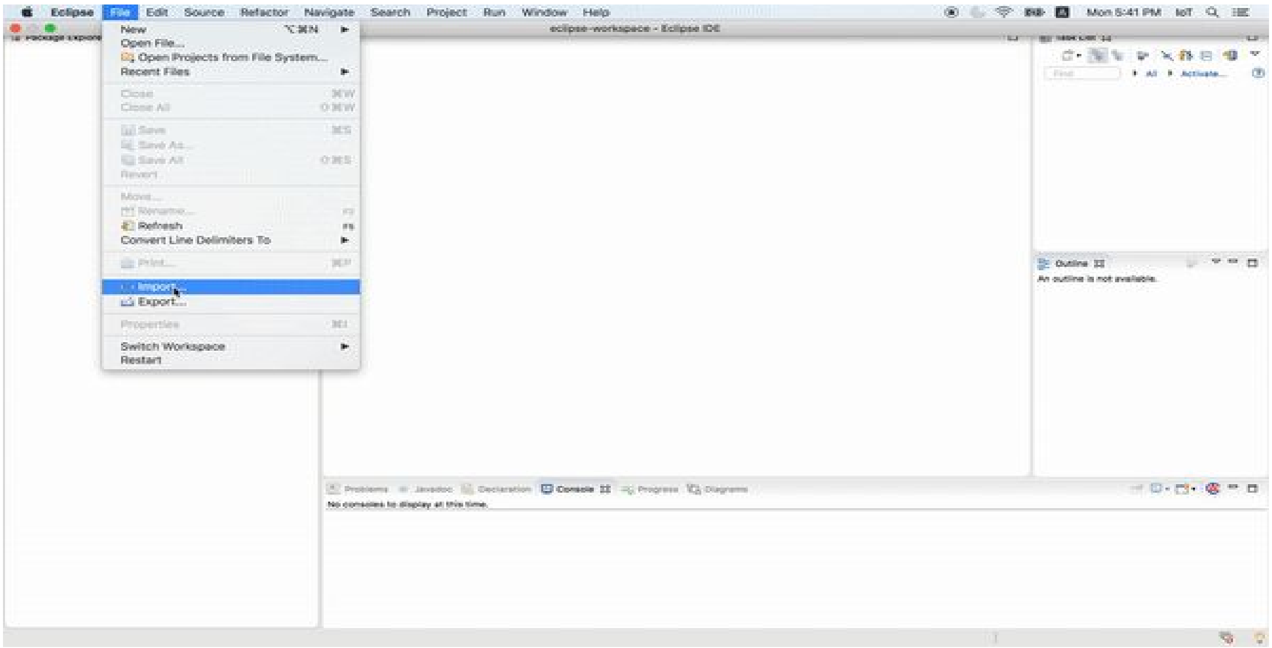
- IoTSim-Edge/
  - docs/ -- The API documentation of IoTSim-Stream
  - examples/ -- Some examples of stream graph applications
  - jars/ -- The jar archives of IoTSim-Stream
  - sources/ -- The source code of IoTSim-Stream

3.4 Setup IoTSim-Edge

Prior to use and work with IoTSim-Edge, one need to import and configure the project in their chosen IDE. Here, we use Eclipse to illustrate how to setup IoTSim-Edge project. The project is a maven project. The main steps are given below:

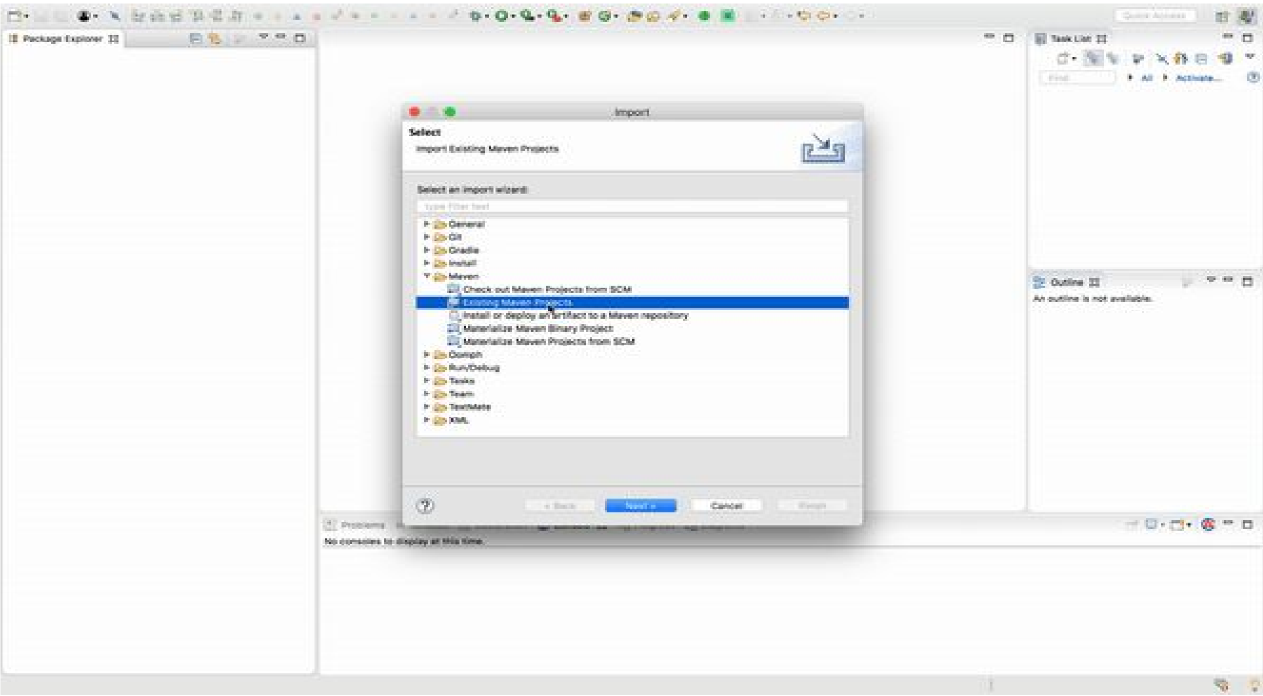
Step 1

Open Eclipse IDE -> File -> import -> Maven -> Existing Maven Projects.



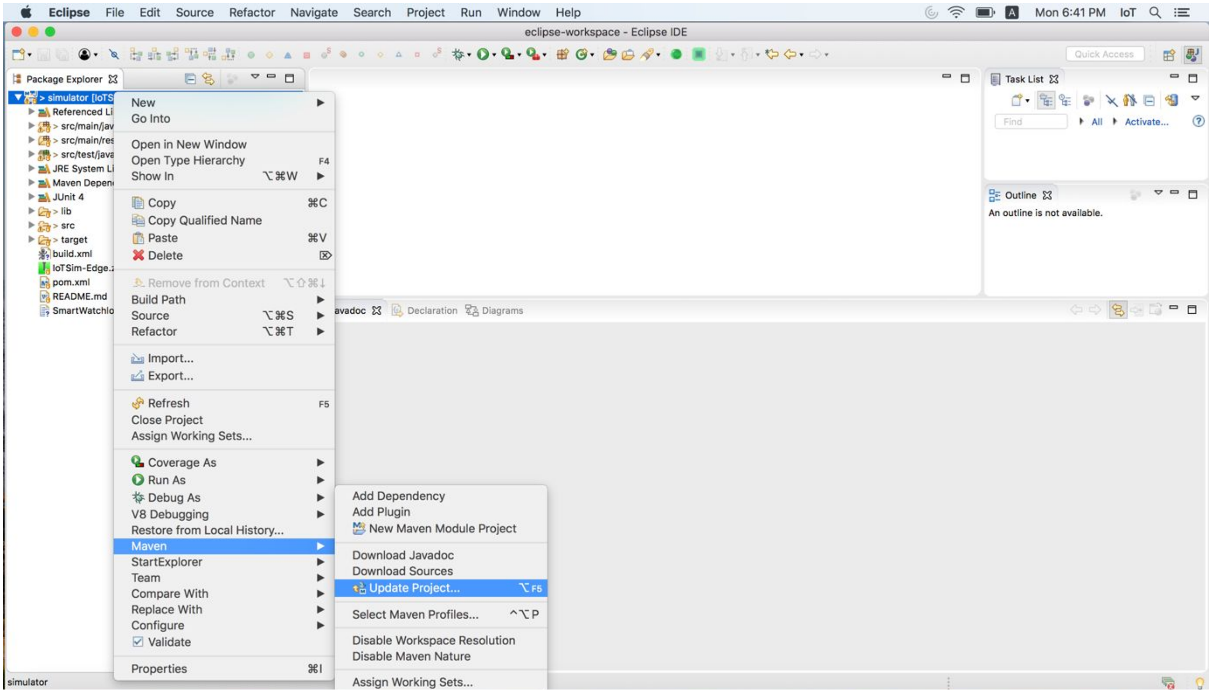
Step 2

Select the folder corresponding to IoTSim- Edge, then click on Open. After that click on Select All then click on Finish.



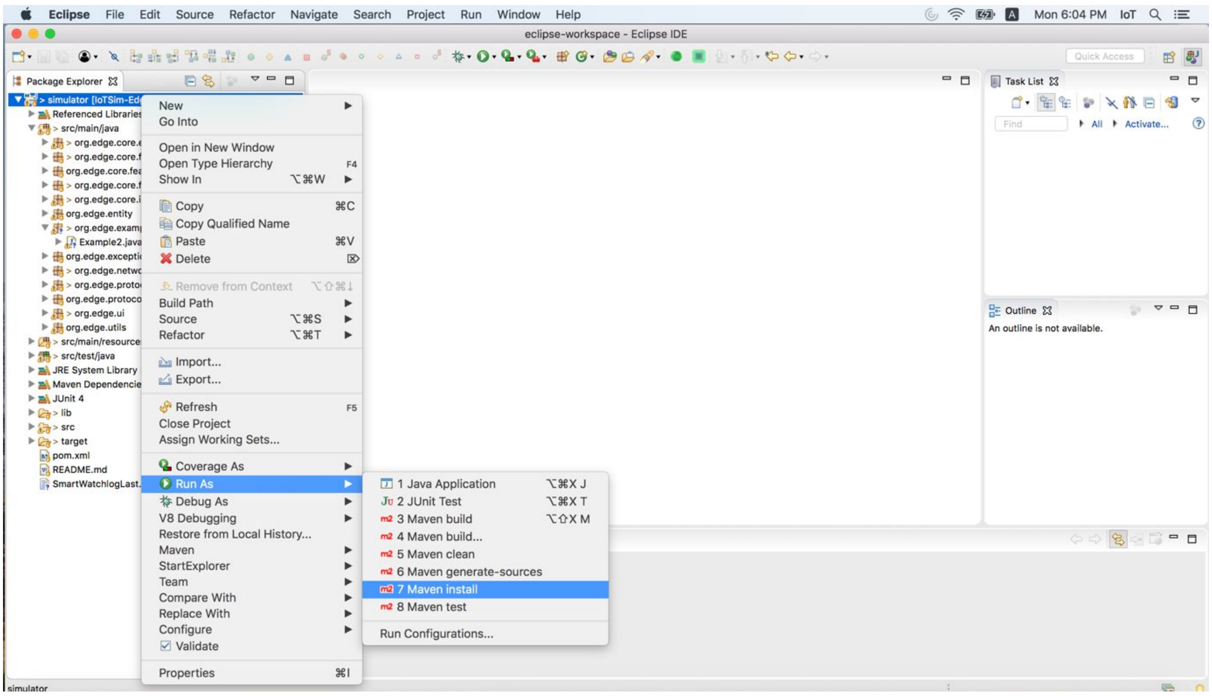
Step 3

Right click on IoTSim-Edge project and click on Update Project that found under Maven.

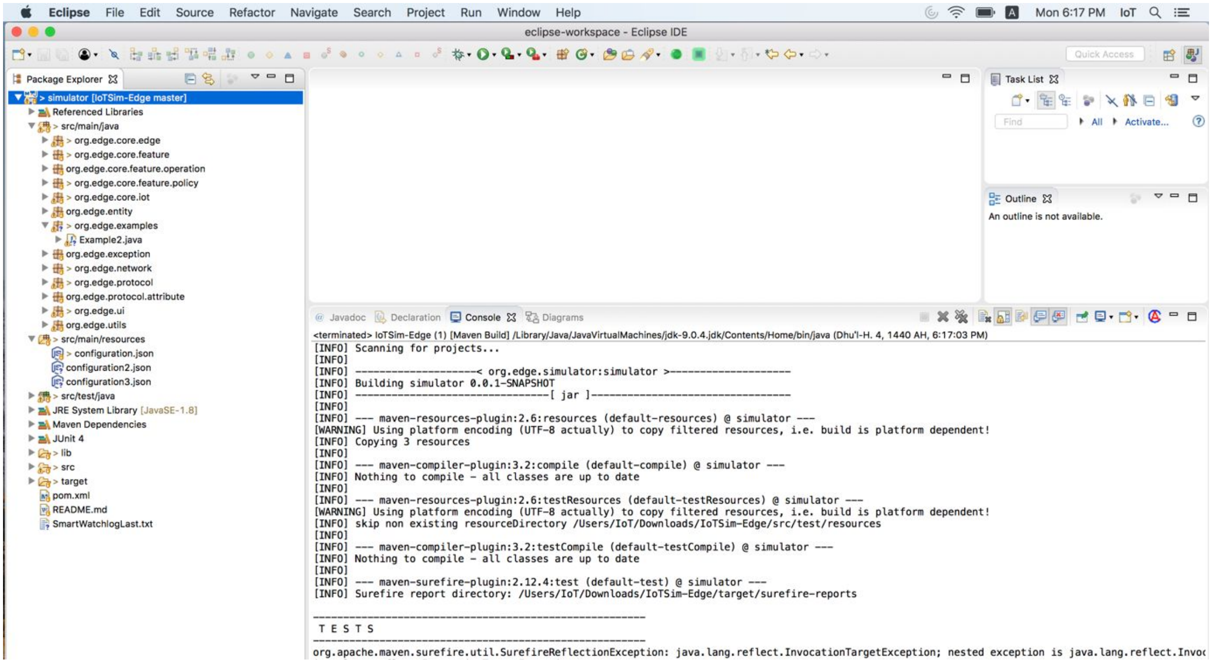


Step 4

Right click on IoTSim-Edge project and click on Maven install that found under Run As.



Now, the installing project main artefact is started.



When this process being completed, you will see “BUILD SUCCESS” as shown in the below screenshot. At this point, you successfully built and configured IoTSim-Edge.

3.5 Simulation Configuration

Before starting the actual simulation, one has to configure the application by setting up the parameters using a configuration file. To allow user to define their system configuration in an easy way, the configuration file is given in json format (configuration.json). These parameters are read by the main program during initialization which configures the environment accordingly. The main parameters defined in the configuration file is discussed in the table given below.

Table 1: User-defined Simulation Parameters Configuration

Entity	Parameter	Description
IoTDevicesEntity	numberofEntity	Number of IoT devices to be configured
	iotType	Type of IoT devices
	name	Name of the IoT device
	data_frequency	Rate of data generation
	dataGenerationTime	Time for data generation
	complexityOfDataPackage	Complexity of generated data
	networkType	Type of network used by the IoT device
	communicationProtocol	Communication protocol used by the IoT device
	max_battery_capacity	Maximum battery capacity of the IoT device
	battery_drainage_rate	Rate of battery drainage
	processingAbility	Whether able to perform processing or not
	movable	Metric to check the mobility
	Location(x,y,z)	Current location of the IoT device
	velocity	Velocity of the IoT device

MELEntitity	vmid	Id of the VM required
	mips	MIPS rate of the VM required
	size	Storage size
	ram	RAM size of the VM required
	bw	Bandwidth capacity of the VM required
	pesNumber	Processing element number of the VM required
	vmm	Virtual machine manager type required
	type	Type of operation performed by the MEL
	datasizeShrinkFactor	Data shrinking due to the operation performed
	MELToplology uplinkIds	Ids of parent MELs connected with the defined MEL
	MELToplology downlinkIds	Ids of child MELs connected with the defined MEL
Broker	name	Name of the broker
edgeDatacenter	edgeType	Type of Edge device
	name	Name of edge device
	architecture	Architecture of VMs supported by edge device
	os	OS supported by VMs of edge devices
	vmm	Virtual machine manager at the datacenter
	ramSize	RAM size supported by edge device
	bwSize	Bandwidth supported by edge device
	storage	Storage provided by edge device
	peEntities mips	Processing elements supported by edge devices
	vmScheduler	VM scheduler for the edge datacenter
	movable	Metric to check the mobility of edge devices
	location (x,y,z)	Current location of the edge device
	velocity	Velocity of the edge device
	signalRange	Signal range of the edge device
	networkType	Network type supported by the edge device
	communicationProtocolSupported	Communication protocols supported by the edge device
	maxIoTDevice_capacity	Max IoT device handled by the edge device
	battery	Metric to check whether edge device is battery operated or not
	max_battery_capacity	Maximum battery capacity of the edge device
	current_battery_capacity	Current battery capacity of the edge device
	battery_drainage_rate	Battery drainage rate of the edge device
	iotDeviceclassNameSupported	Type of IoT devices supported by the edge device

A snapshot of the IoT device entity taken from the configuration file is given below. It shows different metrics and their set values for our example. Other details can be found from the configuration file.

```
"ioTDeviceEntities": [
        {
                "mobilityEntity": {
                        "movable": false,
                        "location": {
                                "x": 0.0,
                                "y": 0.0,
                                "z": 0.0
                        }
                },
                "assignmentId": 1,
                "ioTClassName": "org.edge.core.iot.TemperatureSensor",
                "iotType": "environmental",
                "name": "temperature",
                "data_frequency": 1.0,
                "dataGenerationTime": 1.0,
                "complexityOfDataPackage": 1,
                "networkModelEntity": {
                        "networkType": "wifi",
                        "communicationProtocol": "xmpp"
                },
                "max_battery_capacity": 100.0,
                "battery_drainage_rate": 1.0,
                "processingAbility": 1.0,
                "numberofEntity": 5
        },
```

4 Simulation examples

4.1 Example 1: Healthcare system

This example mimics the behaviour of the communication and energy consumption of smart IoT and Micro elements. In this example, we considered two edge devices in which one edge device has an embedded IoT device. Where the IoT device generates the data based on the defined data rate. In addition, we considered edge devices are also battery powered, based on the processing happening on the edge device. We create many scenarios for preforming more operations (shrinking: Shrinking factor represents the processing happened on the edge device) on the ML near to the IoT. The configuration for this scenario is saved in a json file named (configuration.json) and the corresponding java file for this example is (Example1.java). The configuration of experiment is shown in Table 1, where one IoT device is connected to MEL (id 1) then MEL (id 1) send the data after shrinking it to his down link MEL (id 2). We changed the shrinking factor on MEL (id 1) using the previous json file to study how does this affect the battery for the edge device that hosts the MEL (id 1).

Table 1 Configuration file for Case 1.

IoT device		MEL		Edge device 1	
Location	0, 0, 0	id	1	Type	Raspberry Pi
IoT type	healthcare	MIPS	10000	Location	100, 0, 0
Movable	false	RAM	10000	Movable	false
Data frequency	1	BW	10000	Signal range	100
Data generation time	1	Shrinking factor	'variable'	Max IoT device capacity	10000
Network protocol	bluetooth	Network protocol	bluetooth	Max battery capacity	20000 units
IoT Protocol	COAP	Uplink	-	Battery drainage rate for processing	0.1
Max battery capacity	300	Downlink	2	Battery drainage rate for transfer	0.6

4.1.1 Example 1: configuration file.

4.1.1.1 IoT device configuration.

The configuration file for this example is ([configuration.json]). In this file, we defined the network protocol and IoT protocol between the IoT and edge device as the following snippet shows:

```
        "assignmentId": 1,
        "ioTClassName": "org.edge.core.iot.TemperatureSensor",
        "iotType": "health",
        "name": "smartWatch",
        "data_frequency": 1.0,
        "dataGenerationTime": 1.0,
        "complexityOfDataPackage": 1,
        "networkModelEntity": {
            "networkType": "bluetooth",
            "communicationProtocol": "coap"
        },
        "max_battery_capacity": 300.0,
        "battery_drainage_rate": 1.0,
        "processingAbility": 1.0,
        "numberofEntity": 1
    }
```

4.1.1.2 Micro Element configuration.

The configuration for the first micro element where we define and change its shrinking factor and connected micro element(s) (as down link), can be shown in the following snippet which is taken from ([configuration.json]):

```
"MELEntities": [
    {
        "vmid": 1,
        "mips": 10000,
        "size": 2048,
        "ram": 10000,
        "bw": 10000,
        "pesNumber": 1,
        "vmm": "xxx",
        "cloudletSchedulerClassName": "org.cloudbus.cloudsim.CloudletSchedulerTimeShared",
        "type": "filtering",
        "datasizeShrinkFactor": 0.2,
        "edgeOperationClass": "org.edge.core.feature.operation.FilterOperation",
        "MELTopology": {
            "downLinkIds": [2]
        }
```

The configuration for the second micro element that is connected to the first micro element is:

```
{
    "vmid": 2,
    "mips": 10000,
    "size": 2048,
    "ram": 10000,
    "bw": 10000,
    "pesNumber": 1,
    "vmm": "xxx",
    "cloudletSchedulerClassName": "org.cloudbus.cloudsim.CloudletSchedulerTimeShared",
    "type": "filtering",
    "datasizeShrinkFactor": 0.2,
    "edgeOperationClass": "org.edge.core.feature.operation.FilterOperation",
    "MELTopology": {
        "upLinkId": 1,
        "downLinkIds": []
    }
}
```

4.1.1.3 *IoT and Micro Element connection.*

It is imperative to setup the corresponding IoT devices for each MEL to run the simulation properly. In this example, the IoT with id 1 is connected to MEL with id 1 and it can be done as following:

```
"connections": [
    {
        "vmId": 1,
        "assigmentIoTId": 1
    }
```

4.1.2 Example 1: running the example.

After preparing the configuration file which is named (configuration.json) and can be found in the simulation folder (src/main/resources/configuration.json). Now, you can simply start the simulation by clicking Run File “Example1.java”. During the simulation, you will see the detailed execution of given stream graph application in output toolbar/pane, where IoTSim-Edge logs each event. At the end of simulation, you will see the summary of workflow execution like the below.

========== OUTPUT ==========

Edgelet ID	MicroELement ID	Execution Time	Start Time	Finish Time	Length	Size
000	1	3.00	4.10	7.10	1000.0	30.0
001	1	3.00	7.10	10.10	1000.0	30.0
002	2	3.00	7.10	10.10	1000.0	30.0
003	1	3.00	10.10	13.10	1000.0	30.0
004	2	3.00	10.10	13.10	1000.0	30.0
005	1	3.00	13.10	16.10	1000.0	30.0
006	2	3.00	13.10	16.10	1000.0	30.0
007	1	3.00	16.10	19.10	1000.0	30.0
008	2	3.00	16.10	19.10	1000.0	30.0

4.2 Example 2: Smart Building

In this example, we simulate multiple IoT devices sends their data to one edge device following one specific communication protocol. Features like latency directly depends on the packet size and data rate. We simulate two different communication protocols (COAP & XMPP) to study the latency and energy consumption for IoT devices.

4.2.1.1 IoT device Configuration

For this example, we create two configuration files (configuration2A.json & configuration2B.json). In those files, we defined the network protocol and IoT protocol between the IoT and edge device as the following snippet shows:


```
    "name": "building",
    "data_frequency": 1.0,
    "dataGenerationTime": 1.0,
    "complexityOfDataPackage": 1,
    "networkModelEntity": {
        "networkType": "bluetooth",
        "communicationProtocol": "coap"
    },
    "max_battery_capacity": 100.0,
    "battery_drainage_rate": 1.0,
    "processingAbility": 1.0,
    "numberofEntity": 50
}
```

```
    "dataGenerationTime": 1.0,
    "complexityOfDataPackage": 1,
    "networkModelEntity": {
        "networkType": "bluetooth",
        "communicationProtocol": "xmpp"
    },
    "max_battery_capacity": 100.0,
    "battery_drainage_rate": 1.0,
    "processingAbility": 1.0,
    "numberofEntity": 50
  }
],
```


Figure 1 JSON configuration files for COAP and XMPP protocol

4.2.2 Example 2: running the example.

After preparing the configuration files which are named (configuration2A.json & configuration2B.json) and can be found in the simulation folder (src/main/resources/configuration2A.json or configuration2B.json). Now, you can simply start the simulation by clicking Run File “Example2A.java” or “Example2B.java”. During the simulation, you will see the detailed execution of given stream graph application in output toolbar/pane, where IoTSim-Edge logs each event. At the end of simulation, you will see the summary of workflow execution like the below.

```
152.1: default: Destroying VM #1
default is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
default is shutting down...
edgeDatacenter1 is shutting down...
Simulation completed.
========== OUTPUT ==========
Edgelet ID    MicroELement ID    Execution Time    Start Time    Finish Time    Length    Size
    000           1                 7.64              4.10          11.74        1000.0     30.0
    001           1                 7.64              4.10          11.74        1000.0     30.0
    002           1                 7.64              4.10          11.74        1000.0     30.0
    003           1                 7.64              4.10          11.74        1000.0     30.0
```

Figure 2 Logs for Example2A.java

4.3 Example 3: Capacity planning for Roadside Units (RSUs)

4.3.1 Example 1: configuration file.

4.3.1.1 IoT device configuration.

The configuration file for this example is (configuration3.json). In this file, we defined the network protocol and IoT protocol between the IoT and edge device. Additionally, three important setting must be specified:1) Are IoT devices movable or no? 2) The starting location for the IoT devices. 3) The velocity for the IoT devices. The following snippet shows these three settings:


```
"ioTDeviceEntities": [
    {
        "mobilityEntity": {
            "movable": true,
            "location": {
                "x": 0.0,
                "y": 0.0,
                "z": 0.0
            },
        "volecity": 0.5,
        "range":{
```

4.3.1.2 *Micro Element and Edge configuration.*

The configuration for the micro elements and Edge devices where we define its configurations such as: 1) the location to place our two edge devices 2) the signal range for each edge device, can be shown in the following snippet which is taken from (configuration3.json).

```
"edgeType": "RASPBERRY_PI",
"geo_location": {
    "movable": false,
    "volecity": 0.0,
    "signalRange": 25.0,
    "location": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
    }
}
```

4.3.2 Example 3: running the example.

After preparing the configuration file which is named (configuration3.json) and can be found in the simulation folder (src/main/resources/configuration3.json). Now, you can simply start the simulation by clicking Run File "Example3.java". During the simulation, you will see the detailed execution of given stream graph application in output toolbar/pane, where IoTSim-Edge logs each event. At the end of simulation, you will see the summary of workflow execution like the below.

```
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
default is shutting down...
edgeDatacenter1 is shutting down...
Simulation completed.
========== OUTPUT ==========
Edgelet ID    MicroELement ID    Execution Time    Start Time    Finish Time    Length    Size
    000         1             39.41             4.10         43.51        1000.0      30.0
    001         1             39.41             4.10         43.51        1000.0      30.0
    002         1             39.41             4.10         43.51        1000.0      30.0
    003         1             39.41             4.10         43.51        1000.0      30.0
    004         1             39.41             4.10         43.51        1000.0      30.0
    005         1             39.41             4.10         43.51        1000.0      30.0
    006         1             39.41             4.10         43.51        1000.0      30.0
    007         1             39.41             4.10         43.51        1000.0      30.0
    008         1             39.41             4.10         43.51        1000.0      30.0
```