# Subject : Edge and Fog Computing
# Experiment No. 6

**Title**: Edge computing architecture

**Aim:** Design and deploy an edge computing architecture using edge simulators such as Mobius / EdgeCloudSim.

**Theory:** EdgeCloudSim provides a modular architecture to provide support for a variety of crucial functionalities such as network modeling specific to WLAN and WAN, device mobility model, realistic and tunable load generator. As depicted in Figure 2, the current EdgeCloudSim version has five main modules available: Core Simulation, Networking, Load Generator, Mobility and Edge Orchestrator. To ease fast prototyping efforts, each module contains a default implementation that can be easily extended.

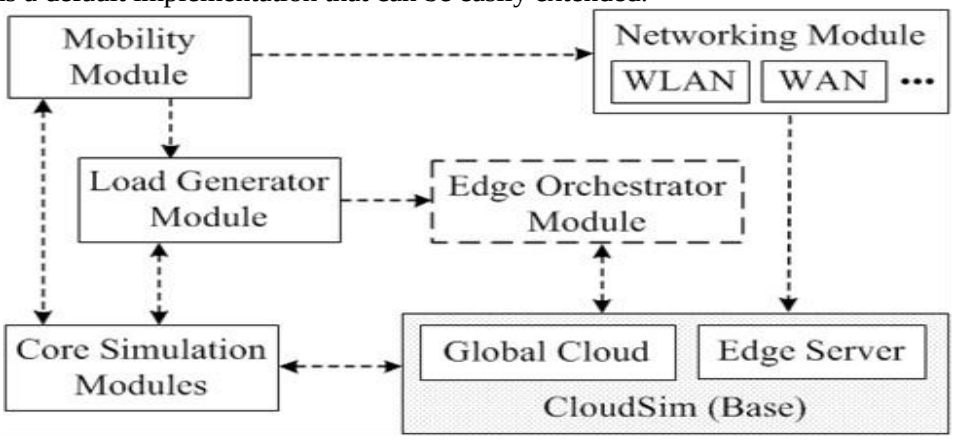
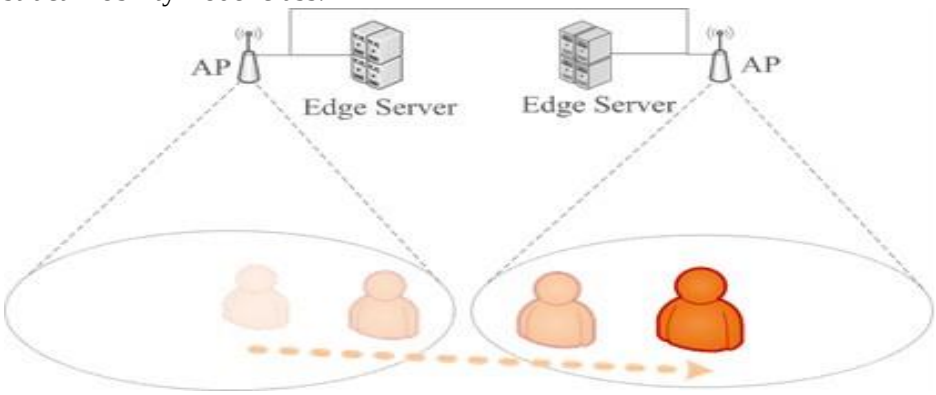



Figure 1: Relationship between EdgeCloudSim modules.

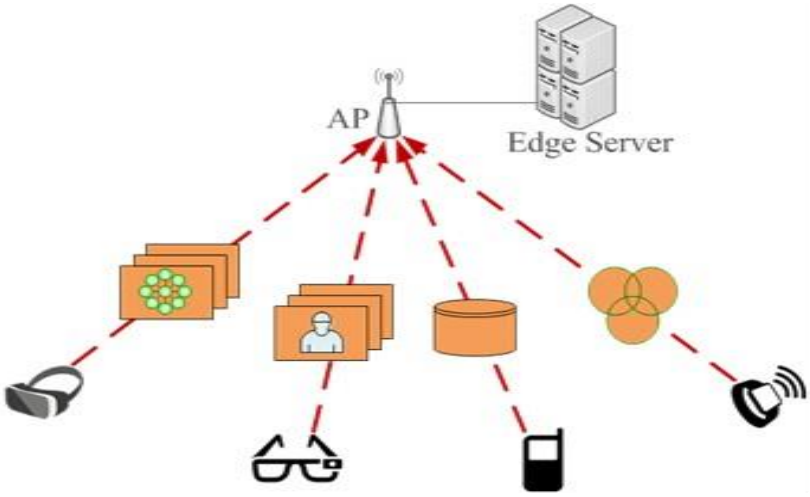
**Mobility Module**: The mobility module manages the location of edge devices and clients. Since CloudSim focuses on the conventional cloud computing principles, the mobility is not considered in the framework. In our design, each mobile device has x and y coordinates which are updated according to the dynamically managed hash table. By default, we provide a nomadic mobility model, but different mobility models can be implemented by extending abstract MobilityModel class.



# Subject : Edge and Fog Computing
# Experiment No. 6

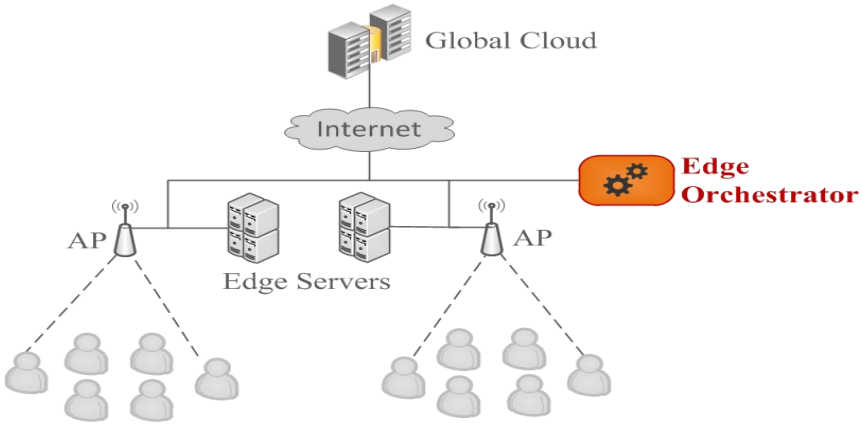
**Load Generator Module**

The load generator module is responsible for generating tasks for the given configuration. By default, the tasks are generated according to a Poisson distribution via active/idle task generation pattern. If other task generation patterns are required, abstract Load Generator Model class should be extended.




Edge Orchestrator Module

The edge orchestrator module is the decision maker of the system. It uses the information collected from the other modules to decide how and where to handle incoming client requests. In the first version, we simply use a probabilistic approach to decide where to handle incoming tasks, but more realistic edge orchestrator can be added by extending abstract EdgeOrchestrator class.




**Procedure:**

# Subject : Edge and Fog Computing
# Experiment No. 6

**Core Simulation Module**

The core simulation module is responsible for loading and running the Edge Computing scenarios from the configuration files. In addition, it offers a logging mechanism to save the simulation results into the files. The results are saved in comma-separated value (CSV) data format by default, but it can be changed to any format.

**Ease of Use**

At the beginning of our study, we observed that too many parameters are used in the simulations and managing these

```
SimSettings SS = SimSettings.getInstance();

//Load settings from configuration file
SS.initialize(configFile, edgeDevicesFile, applicationsFile);

//Enable statistics logging
if(SS.getFileLoggingEnabled())
    SimLogger.enableFileLog();

//Initialize the CloudSim library
CloudSim.init(num_user, calendar, trace_flag, 0.01);

//Generate EdgeCloudsim Scenario Factory
ScenarioFactory sampleFactory = new SampleScenarioFactory(
        numOfClient, simTime, orchestratorPolicy, simScenario);

//Generate EdgeCloudSim Simulation Manager
SimManager manager = new SimManager(sampleFactory, numOfClient, simScenario);

//Start simulation
manager.startSimulation();
```

parameters programmatically is difficult. As a solution, we propose to use configuration files to manage the parameters. EdgeCloudSim reads parameters dynamically from the following files: **config.properties:**

**Simulation settings are managed in configuration file applications.xml: Application properties are stored in xml file. edge_devices.xml: Edge devices (datacenters, hosts, VMs etc.)** are defined in xml file

# Subject : Edge and Fog Computing
# Experiment No. 6

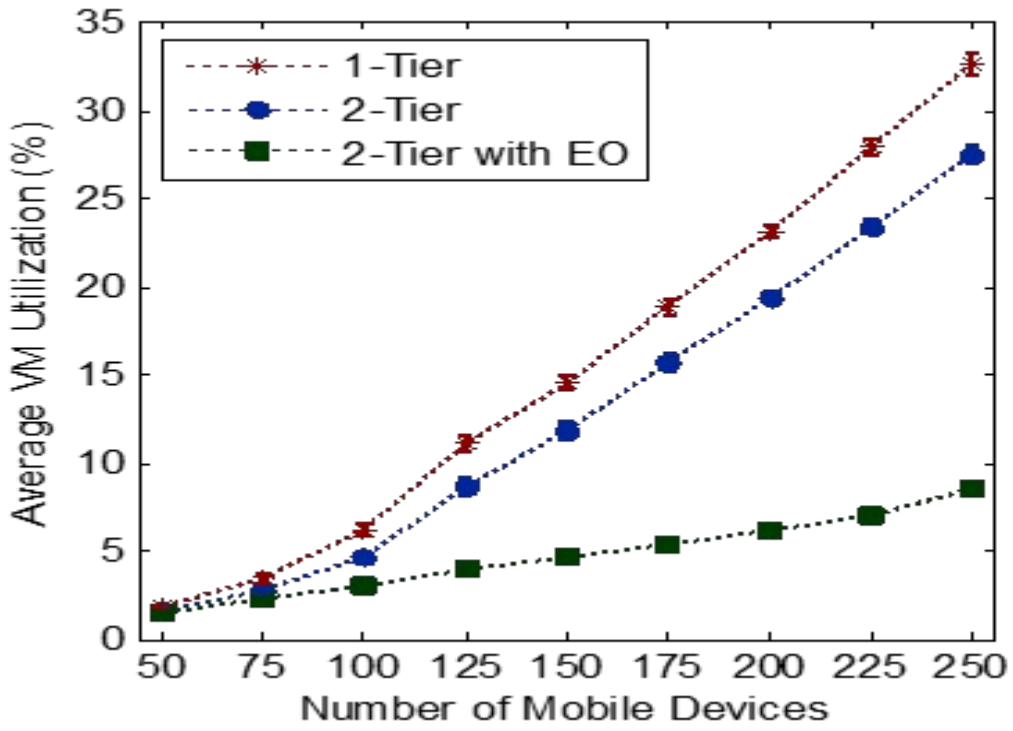
**Compilation and Running**

To compile sample application, compile.sh script which is located in scripts/sample_application folder an be used. You can rewrite similar script for your own application by modifying the arguments of javac command in way to declare the java file which includes your main method. Please note that this script can run on Linux based systems, including Mac OS. You can also use your favorite IDE (eclipse, netbeans etc.) to compile your project. In order to run multiple sample_application scenarios in parallel, you can use run_scenarios.sh script which is located in scripts/sample_application folder. To run your own application, modify the java command in runner.sh script in a way to declare the java class which includes your main method. The details of using this script is explained in this wiki page.

You can also monitor each process via the output files located under scripts/sample_application/output/date folder. For example: ./run_scenarios.sh 8 10 tail -f output/date/ite_1.log

**Analyzing the Results**

At the end of each iteration, simulation results will be compressed in the output/date/ite_n.tgz files. When you extract these tgz files, you will see lots of log file in csv format. You can find matlab files which can plot graphics by using these files under scripts/sample_application/matlab folder. You can also write other scripts (e.g. python scripts) with the same manner of matlab plotter files.



Subject : Edge and Fog Computing
Experiment No. 6

## Conclusion:

In this experiment, we successfully designed and deployed an edge computing architecture using the EdgeCloudSim simulator. The modular structure of EdgeCloudSim allowed us to simulate various aspects of edge computing, including mobility, load generation, and orchestration of edge resources. Through this simulation, we demonstrated how edge computing brings computational resources closer to the user, improving latency and performance compared to traditional cloud models. The practical use of configuration files, automated scripts, and analysis of results provided insights into the efficiency and scalability of edge computing for real-time applications.