Modelling and Simulation of Edge Computing Environments

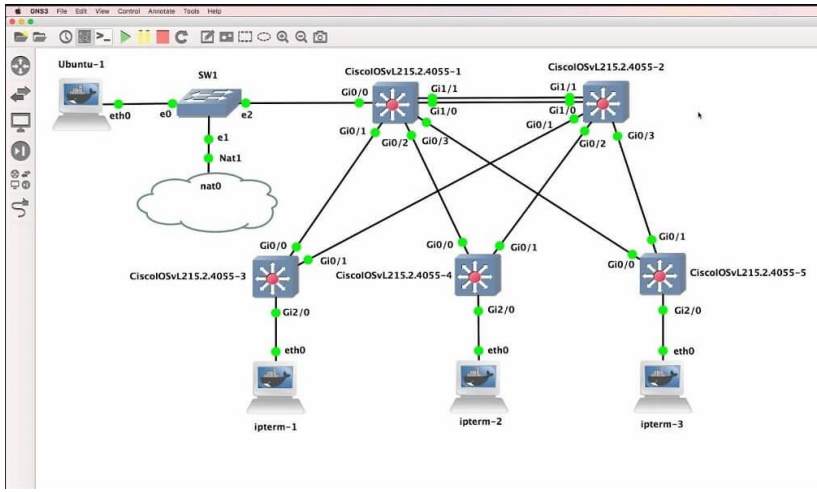
Cagatay Sonmez

October 2025

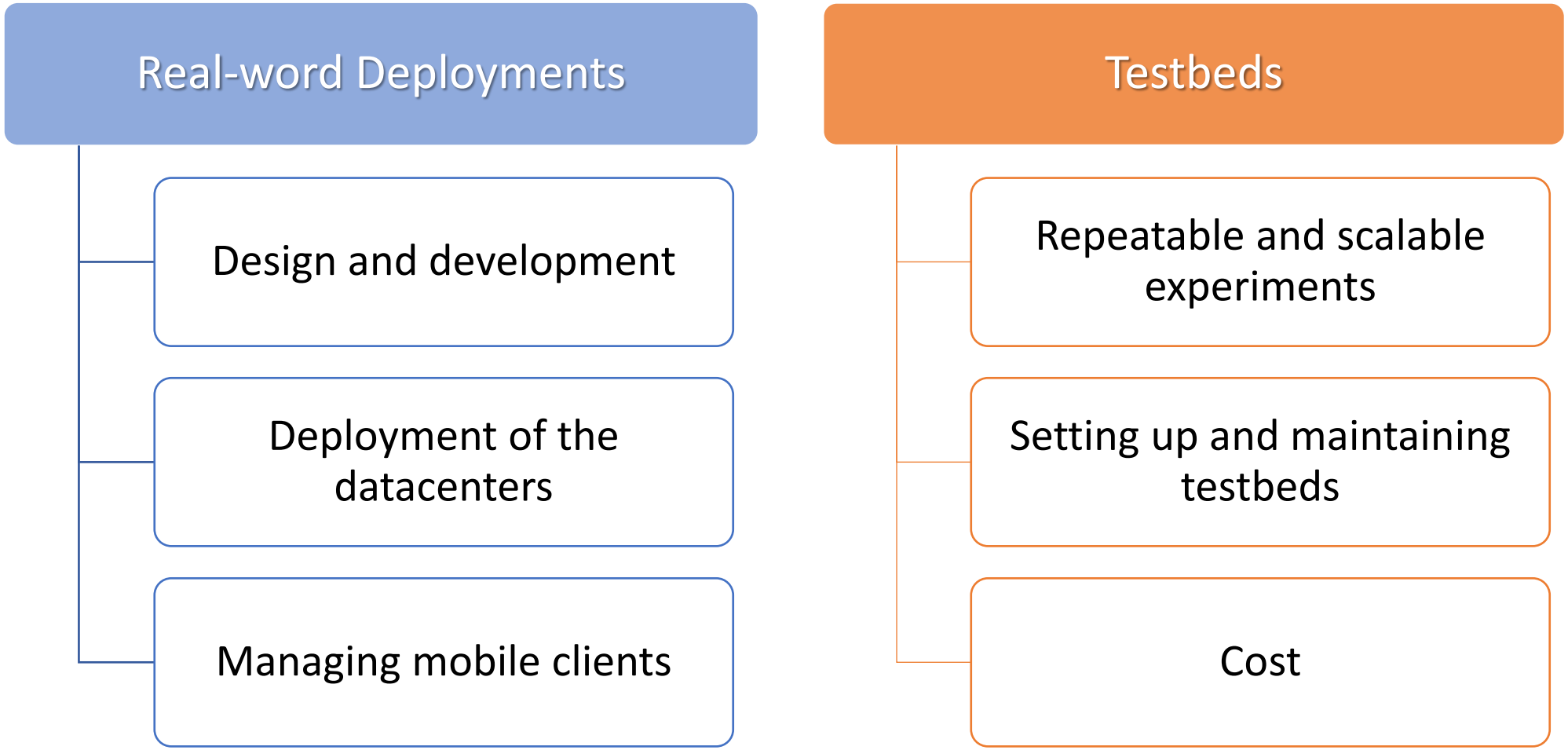
The Primary Sources Used in This Document

- Monika Gill, Dinesh Singh, "**A comprehensive study of simulation frameworks and research directions in fog computing**," *Computer Science Review*, Volume 40, 2021.
- Andras Markus, Attila Kertesz, "**A survey and taxonomy of simulation environments modelling fog computing**", *Simulation Modelling Practice and Theory*, Volume 101, 2020.
- Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "**CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms**", *Softw. Pract. Exper.*, Vol. 41, No. 1, pp. 23–50, Jan. 2011.
- Sonmez, Cagatay, Atay Ozgovde, and Cem Ersoy. "**EdgeCloudSim: An environment for performance evaluation of edge computing systems**" *Transactions on Emerging Telecommunications Technologies* 29, no. 11 (2018): e3493.

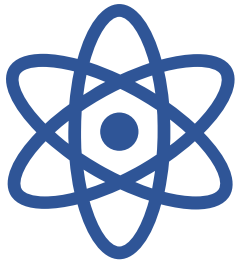
Reality, Testbeds, Simulators



Motivation of Using Simulators

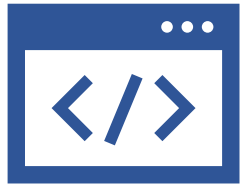


Methods of Performance Evaluation



Emulators

- Duplicates all the software, hardware, and operating systems of a real device
- Emulators are more reliable and suitable for debugging
- However, emulators usually runs slow



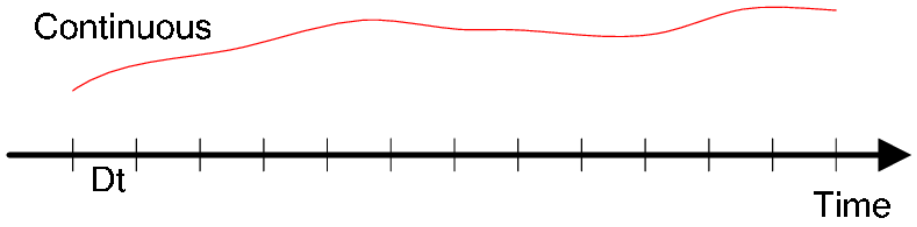
Simulators

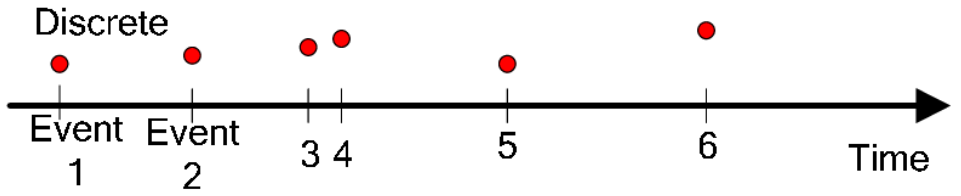
- Creates an environment that mimics the behavior of a real device
- Suitable for rapid prototyping and testing of an idea that has not yet been developed
- However, simulators can sometimes oversimplify the real solution

Hybrid Models

- In some cases, simulators and emulators are used together. Here, the emulators are used to model the focused area more realistically
- In some studies, the network topology is determined with a simulation study, and then the performance of the relevant topology is evaluated through emulators such as maxinet and mininet

Continuous-Time and Discrete-Event Models





Continuous-time simulation

- Tracks system state continuously
- More granular and more accurate but require more computational resources
- Continuous-time simulation is typically applied to natural sciences phenomena like biological, chemical, and environmental processes

Discrete-event simulation (DES)

- Emphasizes events and state changes
- Uses time skipping; it jumps between changes in the system
- Often relies on queueing theory
- The majority of network and cloud simulators are DES-based.

Image source: Helal, Magdy. A hybrid system dynamics-discrete event simulation approach to simulating the manufacturing enterprise. University of Central Florida, 2008.

CloudSim Simulator

- CloudSim [1] is the most used cloud simulator for typical cloud computing scenarios.
- CloudSim is open-source, Java based solution, and it is built on SimJava [2].
- It is mainly designed for modeling Infrastructure as a Service (IaaS) cloud computing environments.
- In CloudSim, users define tasks by creating so-called cloudlets.
- Cloudlets (tasks) are processed by virtual machines running on cloud resources.
- CloudSim also contains a power model, but it is only restricted to CPU energy consumption.

CloudSim Simulator cont.

- CloudSim supports both modeling of the cloud components such as datacenters, hosts, and VMs and the resource provisioning policies such as CPU, storage, memory, and bandwidth utilization models.
- Using CloudSim for Edge/Fog scenarios brings some difficulties.
- Some aspects of the VM utilization model are unrealistic.
  - There is no limitation while assigning tasks to VMs.
- It does not provide realistic network delay model.
  - It uses fixed static variables as network delay.
- It does not provide anything related to user mobility.

 There are many extensions of CloudSim, focusing on mostly insufficient features.

Extensions of CloudSim

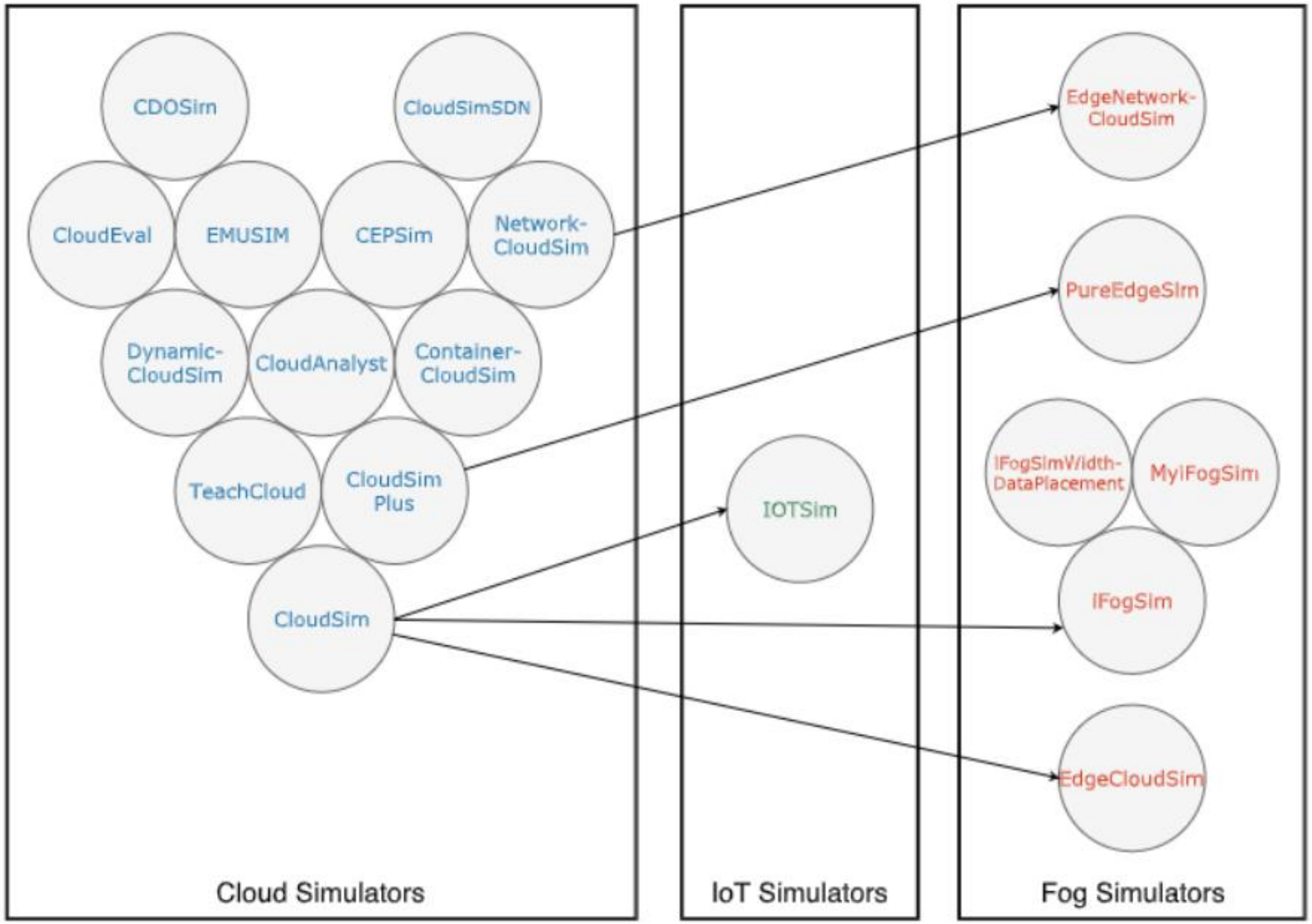


Image source: Andras Markus, Attila Kertesz, "A survey and taxonomy of simulation environments modelling fog computing", *Simulation Modelling Practice and Theory*, Volume 101, 2020.

CloudSim Core Entities

Datacenter: Encapsulates a set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations.

Host: A host is associated to a datacenter. Host executes actions related to management of VMs.

VM: A virtual machine models which is managed and hosted by a Cloud host component.

Cloudlet: This term is used as a task in CloudSim. Cloudlet simply represents a task.

DatacenterBroker: Represents a broker (user) and has two mechanisms for submitting VM provisioning requests to data centers and submitting tasks to VMs.

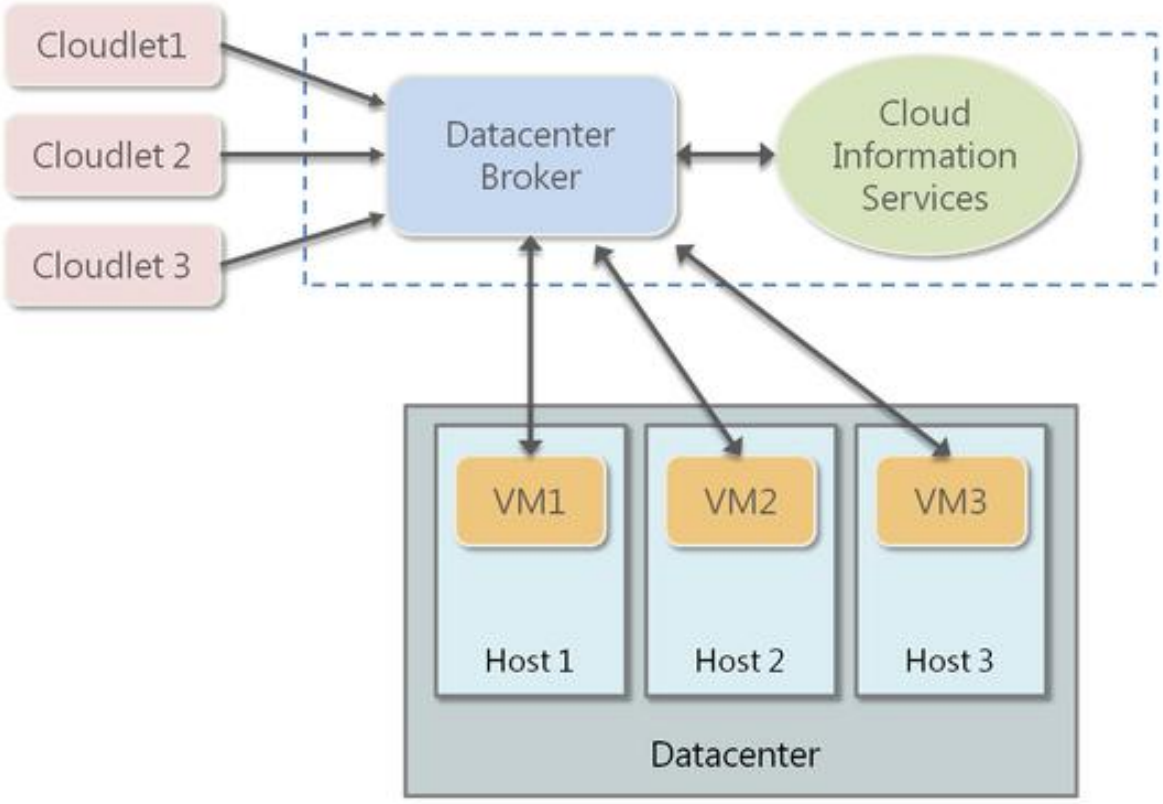


Image source: https://medium.com/ingkwan/getting-started-with-cloudsim-631e7f6b85d6

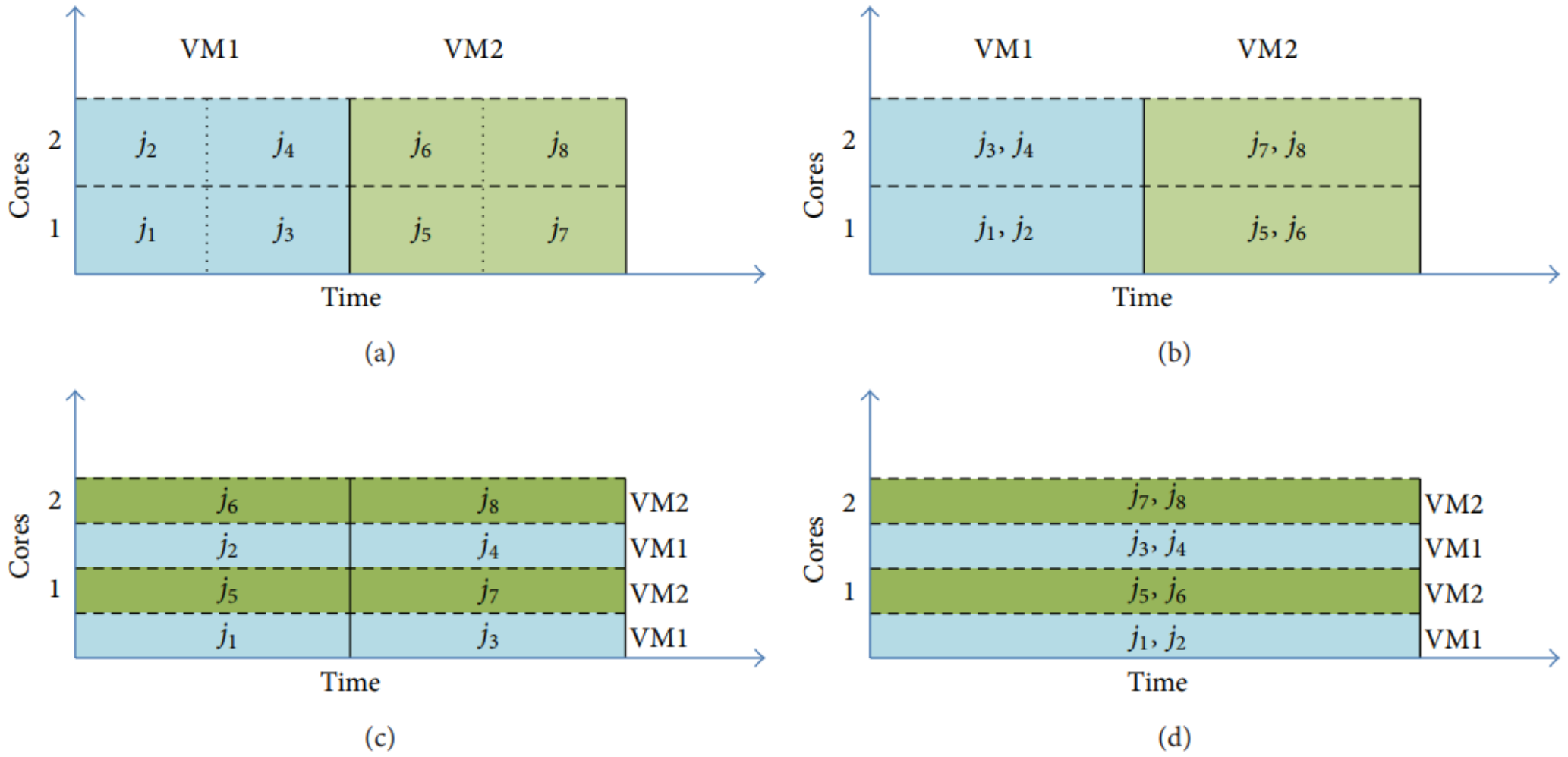
CloudSim VM and Task Provisioning Policies

- **VMScheduler**
  - *VmSchedulerSpaceShared*: Only one VM can run on a host's core at a time.
  - *VmSchedulerTimeShared*: Dynamically distribute the capacity of a core among.
- **CloudletScheduler**
  - *CloudletSchedulerSpaceShared*: Only one task can run on a VM's core at a time.
  - *CloudletSchedulerTimeShared*: All tasks can be assigned to the same VM, and they are dynamically context switched during their life cycle.

★ **Space Shared** -> queued and executed sequentially

★ **Time Shared** -> run simultaneously by context-switching

Time-shared and space-shared policy for VMs and jobs

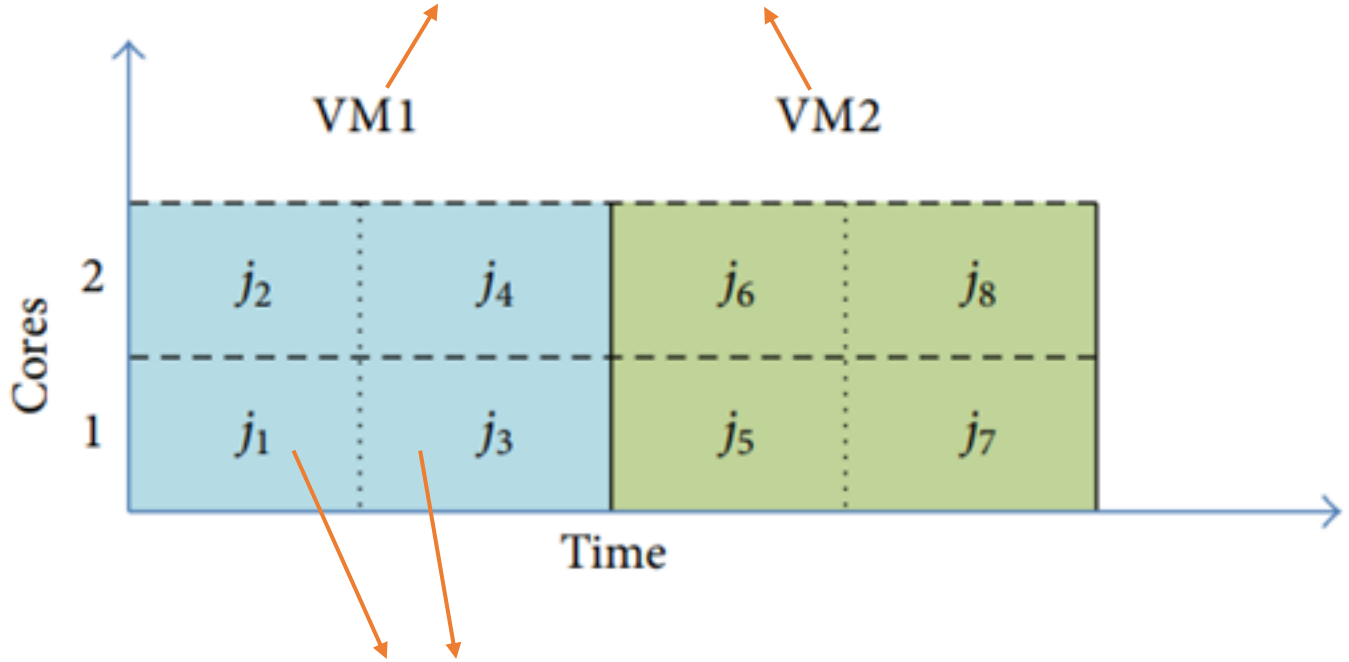


(a) Space-share for VMs and jobs, (b) space-share for VMs and time-share for jobs, (c) time-share for VMs and space-share for jobs, and (d) time-share for VMs and jobs.

Space-Shared Schedular for VMs and Tasks

- Jobs (tasks) j1, j2, j3, and j4 to be hosted in VM1, whereas j5, j6, j7, and j8 to be hosted in VM2

As each VM requires **two** cores, only **one** VM can run at a time.
(VM2 must wait for VM1 to finish)

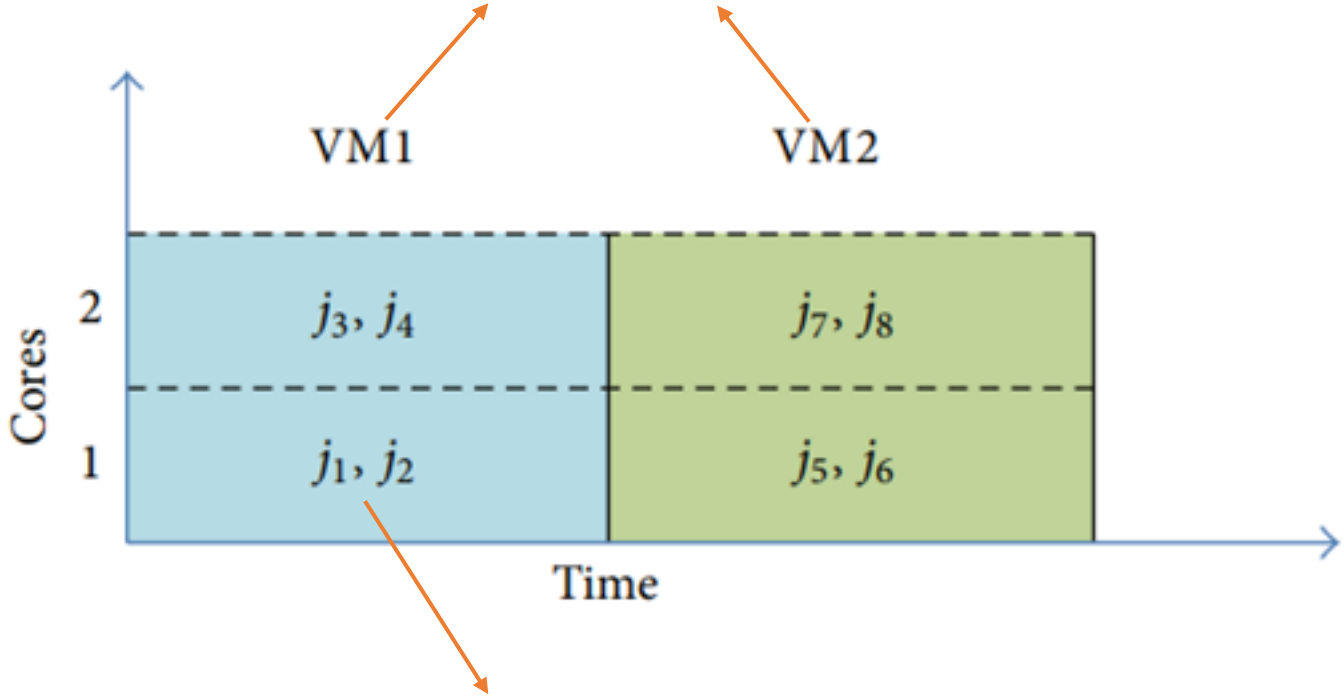


Each job demands only **one** core, hence j1 and j2 can run simultaneously.
During this period, the j3 and j4 wait in the execution queue.

Space-Shared for VMs & Time-Shared for Tasks

- Jobs (tasks) j1, j2, j3, and j4 to be hosted in VM1, whereas j5, j6, j7, and j8 to be hosted in VM2

As each VM requires **two** cores, only **one** VM can run at a time.



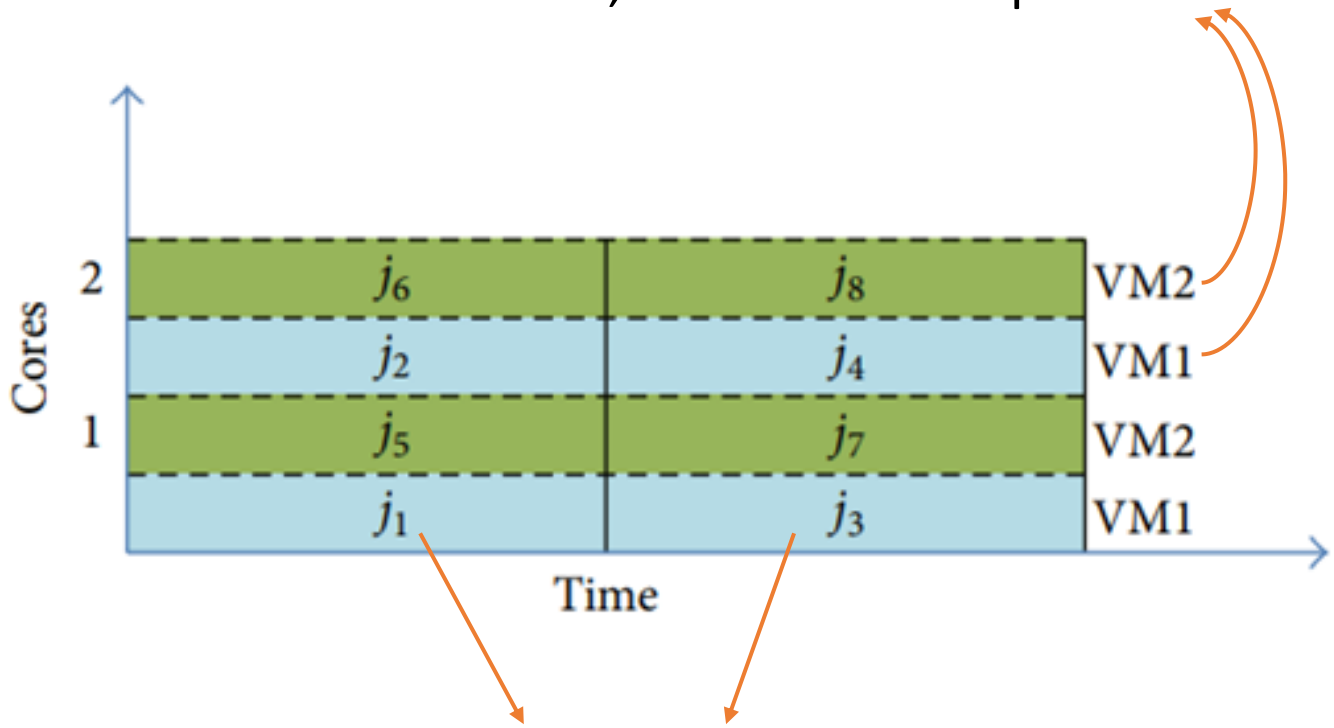
⚠ CloudSim simplifies context switching by extending each task's runtime to reflect shared resource usage, not by executing them in discrete time slices.

All tasks assigned to VM1 run simultaneously and they are dynamically context switched during their life cycle.

Time-Shared for VMs & Space-Shared for Tasks

- Jobs (tasks) j1, j2, j3, and j4 to be hosted in VM1, whereas j5, j6, j7, and j8 to be hosted in VM2

Each VM uses 2 cores, both can run in parallel.



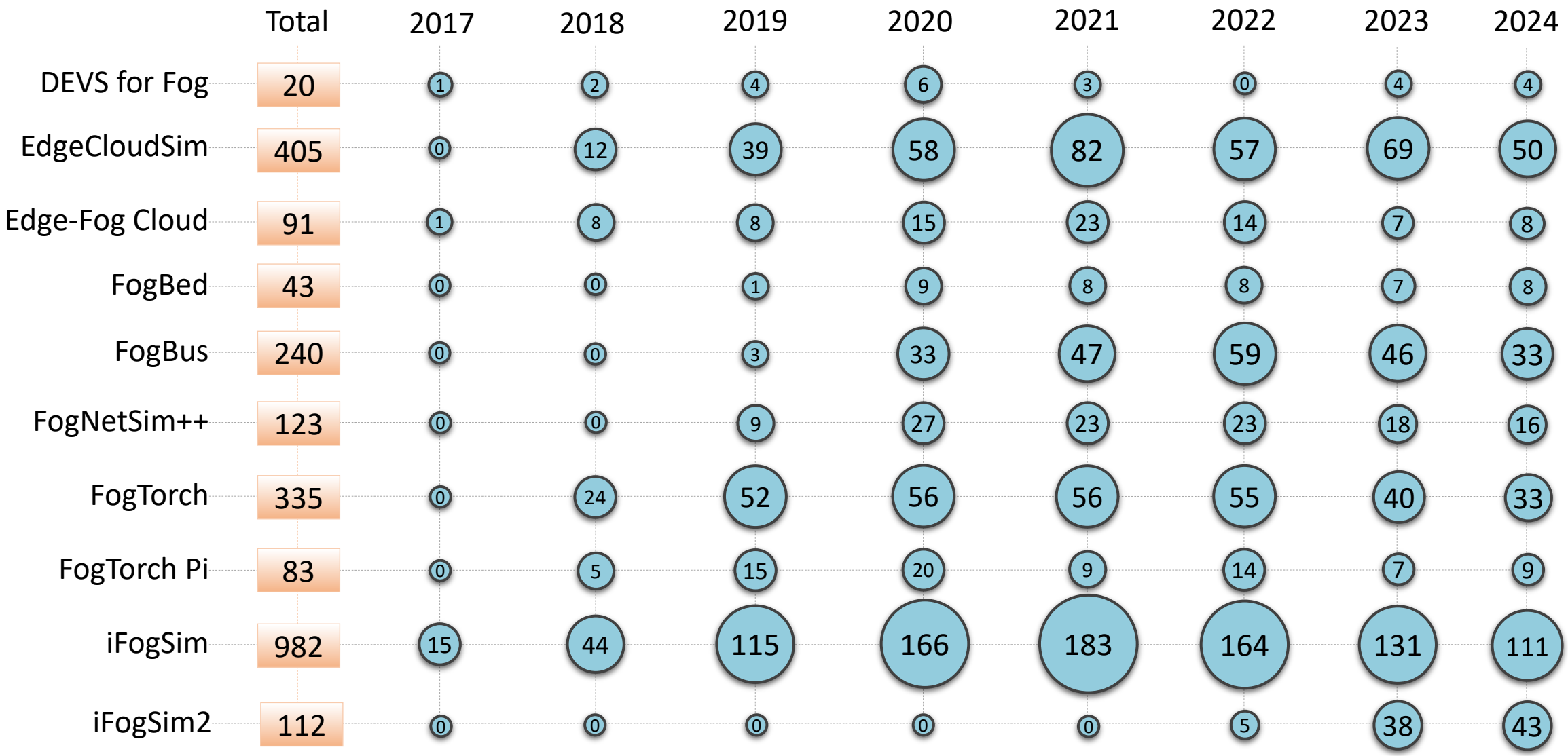
Each job requires only **one** core and **cannot** run simultaneously.
Each job should **wait** for the preceding jobs in the **queue** to finish executing.

Characteristic Features of Edge/Fog Simulators

Simulator	Lang.	Type	Core simulator	Mo-bility	Scala-bility	Energy + Cost	Network Modelling	Infrastructure entities	Pub. year
iFogSim [3]	Java	Event driven	CloudSim	✗	✗	E + C	Link bandwidth, delay, network usage	Sensors, Actuators, Fog devices and Datacenters	2017
FogTorch [4]	Java	NA	NA	✗	✗	✗	Latency, bandwidth	Things, Fog and Cloud	2017
EdgeCloudSim [5]	Java	Event driven	CloudSim	✓	✗	✗	WAN link model, WLAN link model	Mobile client, Edge server, Cloud	2017
FogBus [6]	Java	NA	NA	–	✓	E	Latency, network usage	IoT devices, Fog gw nodes, Fog compute nodes, Cloud datacenter	2018
FogNetSim++ [7]	C++	Network driven	OMNET++	✓	✓	E+C	Packet drop, retransmission, link bandwidth, bit error rate	Mobile end node devices, Fog nodes, Broker nodes, Base stations	2018
Edge-Fog Cloud [8]	Python	NA	NA	✗	✓	C	Network cost	Edge, Fog layer and Datastore	2016
FogBed [9]	Python	Emulator	MiniNet & Docker	✗	✗	–	Service latency	Virtual cloud/fog/edge instance, Virtual switches and links	2018
EmuFog [10]	Java	Emulator	MaxiNet	✗	✓	C	Latency	Fog nodes, Network devices(routers)	2017
DEVS for Fog [11]	C++	Event driven	NA	–	–	–	–	User, Broker, Fog, Cloud	2017

Data source: Monika Gill, Dinesh Singh, "A comprehensive study of simulation frameworks and research directions in fog computing," *Computer Science Review*, Volume 40, 2021.

Citations of Simulators Distributed Over the Years



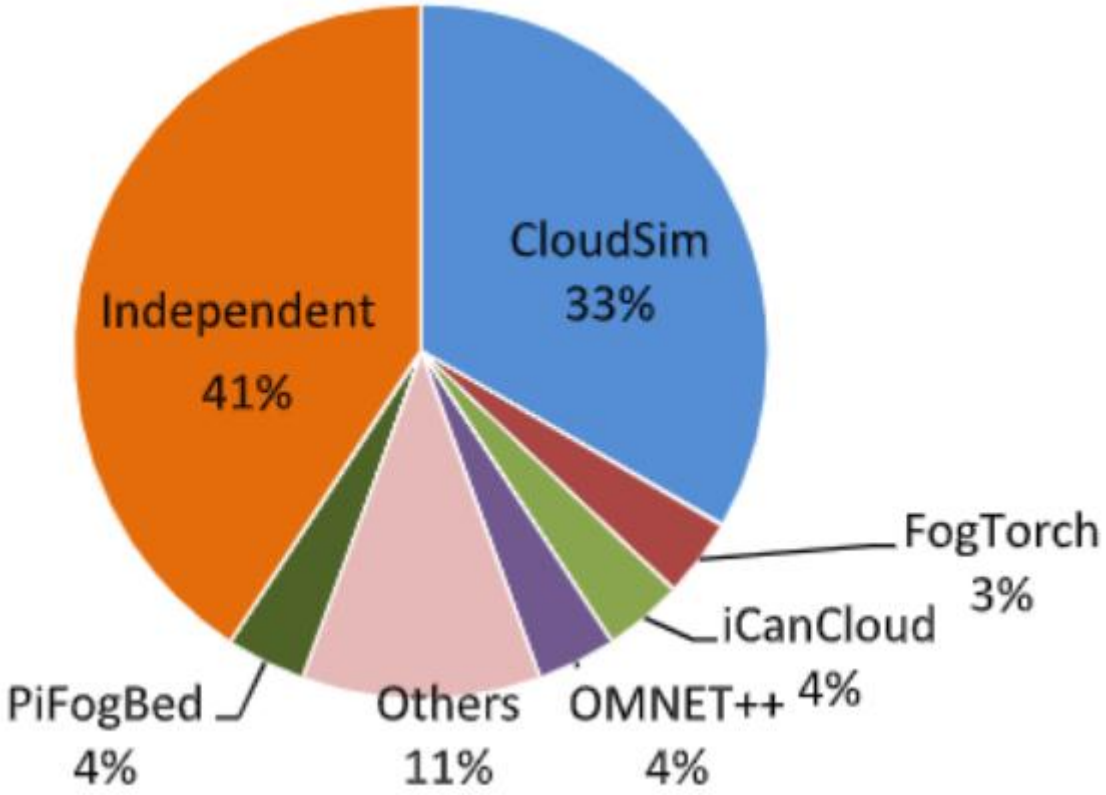
Data Source: Web of Science (the *total value includes 2025, as of November 10th*)

Open Repositories of Fog/Edge Simulators

Simulator	Source Code	Last Commit	Current Release	# of commits	# of contrib.	License
iFogSim	https://github.com/Cloudslab/iFogSim1	21 Sep. 2016	v2.0	71	2	–
iFogSim2	https://github.com/Cloudslab/iFogSim	1 Apr. 2025	v2.0.0	66	1	–
FogTorch	https://github.com/di-unipi-socc/FogTorch	22 Dec. 2016	NRP	28	1	MIT
EdgeCloudSim	https://github.com/CagataySonmez/EdgeCloudSim	2 Nov. 2020	v 4.0	42	3	GPL-3.0
FogBus	https://github.com/shreshthtuli/FogBus	30 Mar. 2019	v 2.0	199	2	GPL-2.0
FogNetSim++	https://github.com/rtqayyum/fognetsimpp	5 Dec. 2018	NRP	9	1	GPL-3.0
Edge-Fog Cloud	https://github.com/nitindermohan/EdgeFogSimulator	17 Oct. 2016	NRP	9	1	GPL-3.0
FogBed	https://github.com/fogbed/fogbed	11 Nov. 2018	NRP	2105	35	Mininet 2.3.0d1
EmuFog	https://github.com/emufog/emufog	28 Sep. 2020	v 2.0	259	2	MIT
DEVS for Fog	https://www.csit.carleton.ca/~msthilaire/FogDEVS/	23 Sep. 2016	NRP	–	–	–

As of October 2025 (README updates are ignored), NRP: No Releases Published, Blank (–): Not Available.

Existing Edge/Fog Simulators & Programming Lang. Used



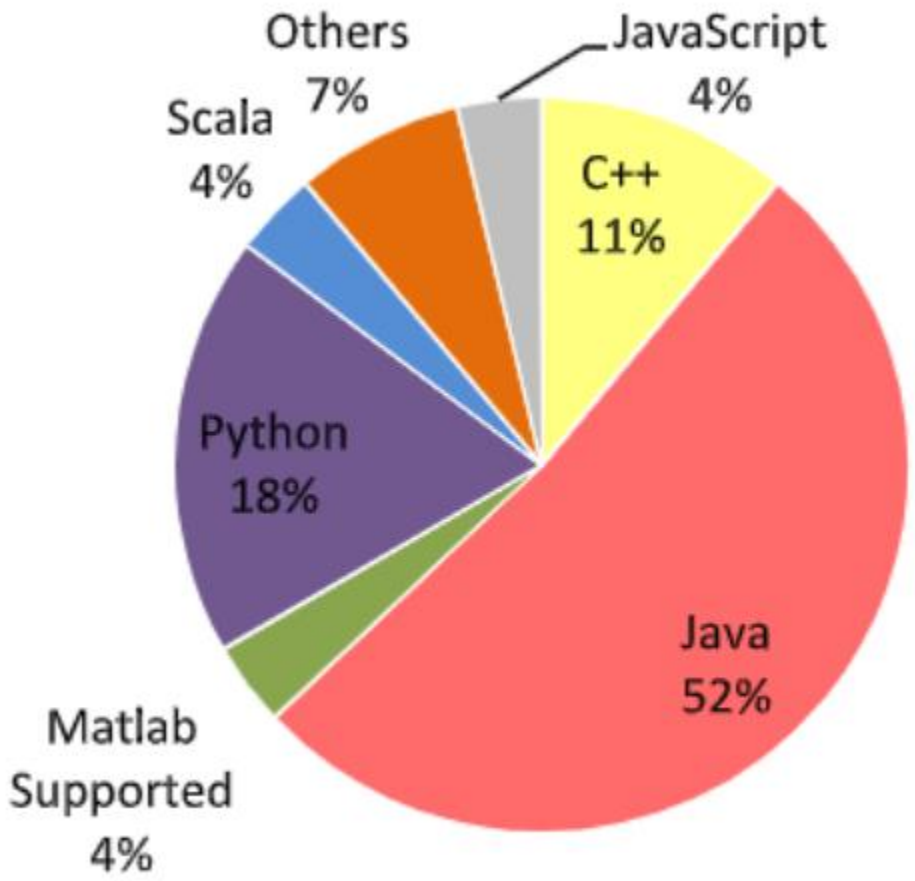


Image source: Monika Gill, Dinesh Singh, "A comprehensive study of simulation frameworks and research directions in fog computing," *Computer Science Review*, Volume 40, 2021.

EdgeCloudSim

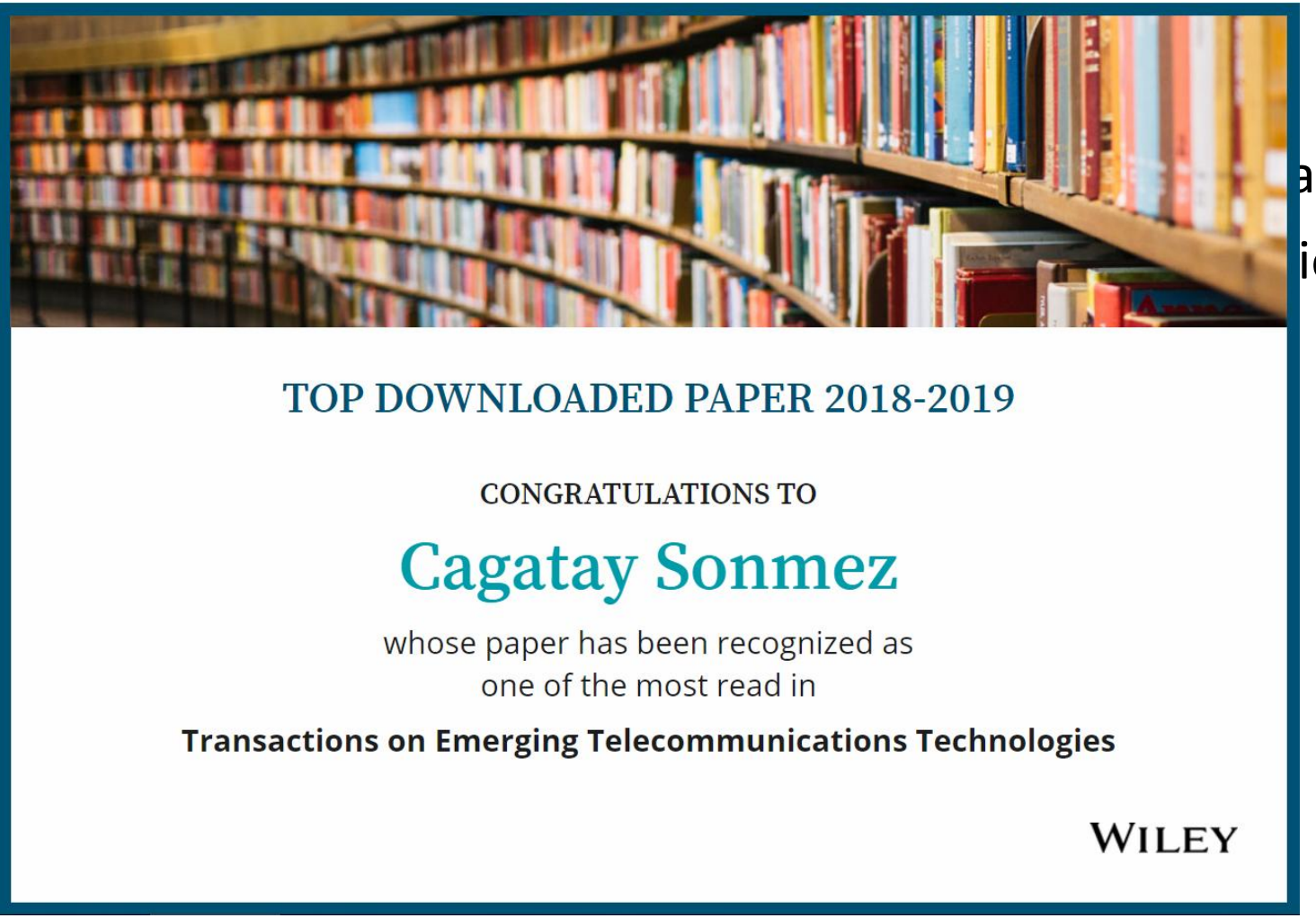
C. Sonmez, A. Ozgovde and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, Vol. 29, No. 11, p. e3493, 2018

What is EdgeCloudSim

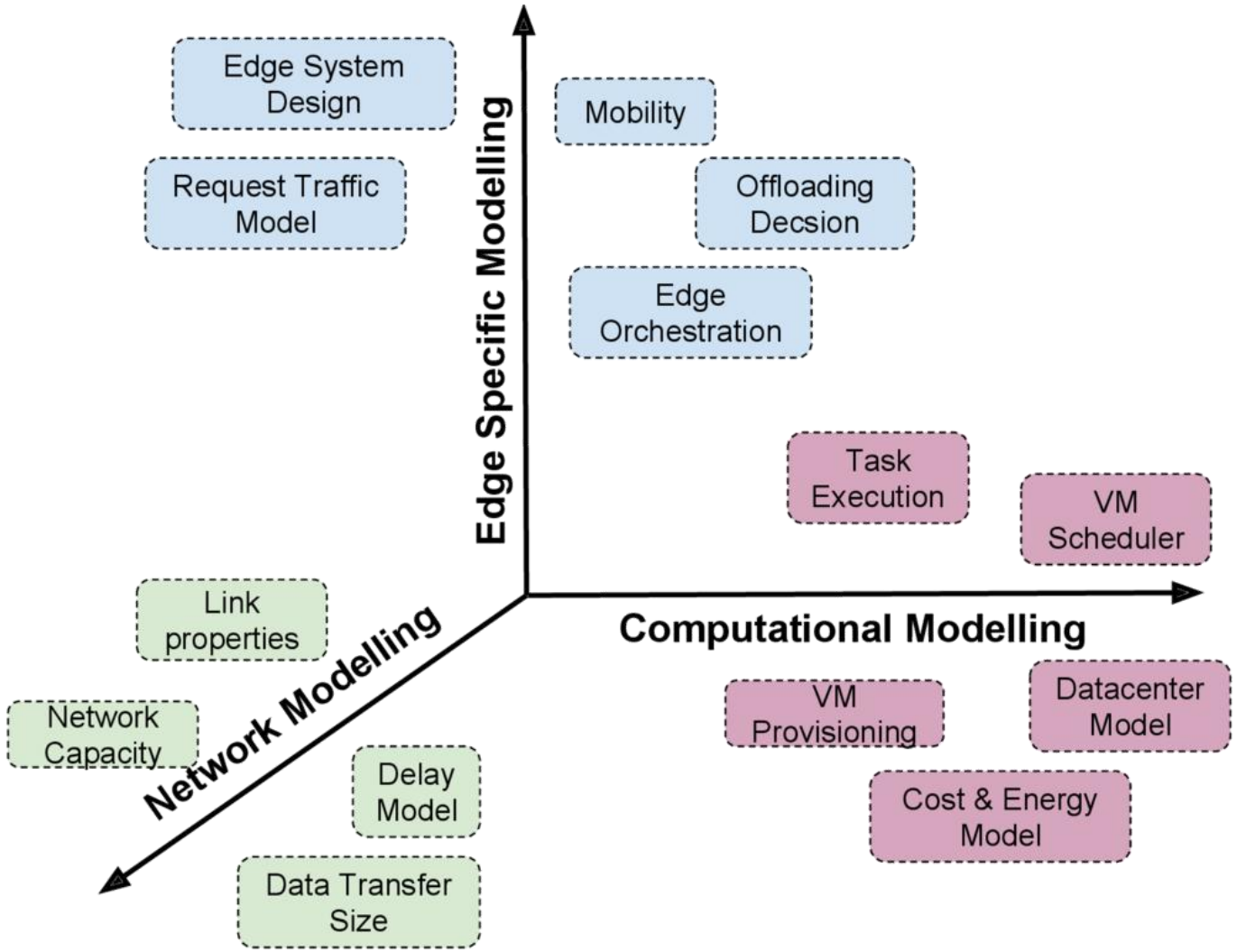
- EdgeCloudSim is a new simulator
- Provides a simulation environment specific to edge computing scenarios
- EdgeCloudSim is based on CloudSim but adds some additional functionalities
- Extensible and easy-to-use
- Publicly available on GitHub
  - https://github.com/CagataySonmez/EdgeCloudSim
- Has high reputation; as of October 2025
  - 736 citations based on Google Scholar
  - A discussion forum [12] with more than 200 members
  - More than 26K views on YouTube channel [13]

What is EdgeCloudSim

- EdgeCloudSi
- Provides a si arios
- EdgeCloudSi ionalities
- Extensible a
- Publicly avai
  - https://gi
- Has high rep
  - 736 citati
  - A discuss
  - More tha

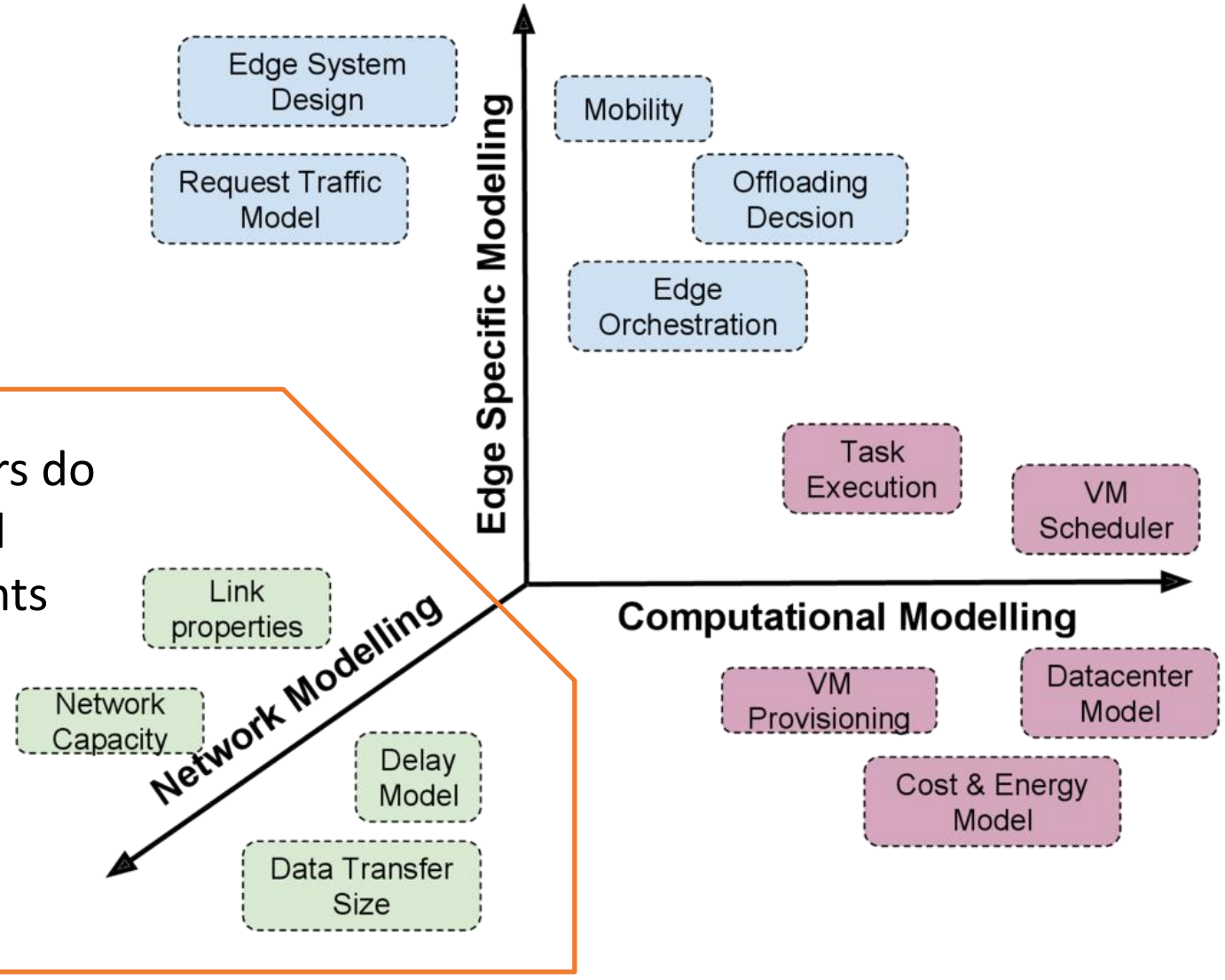


Motivation of Developing a Simulator



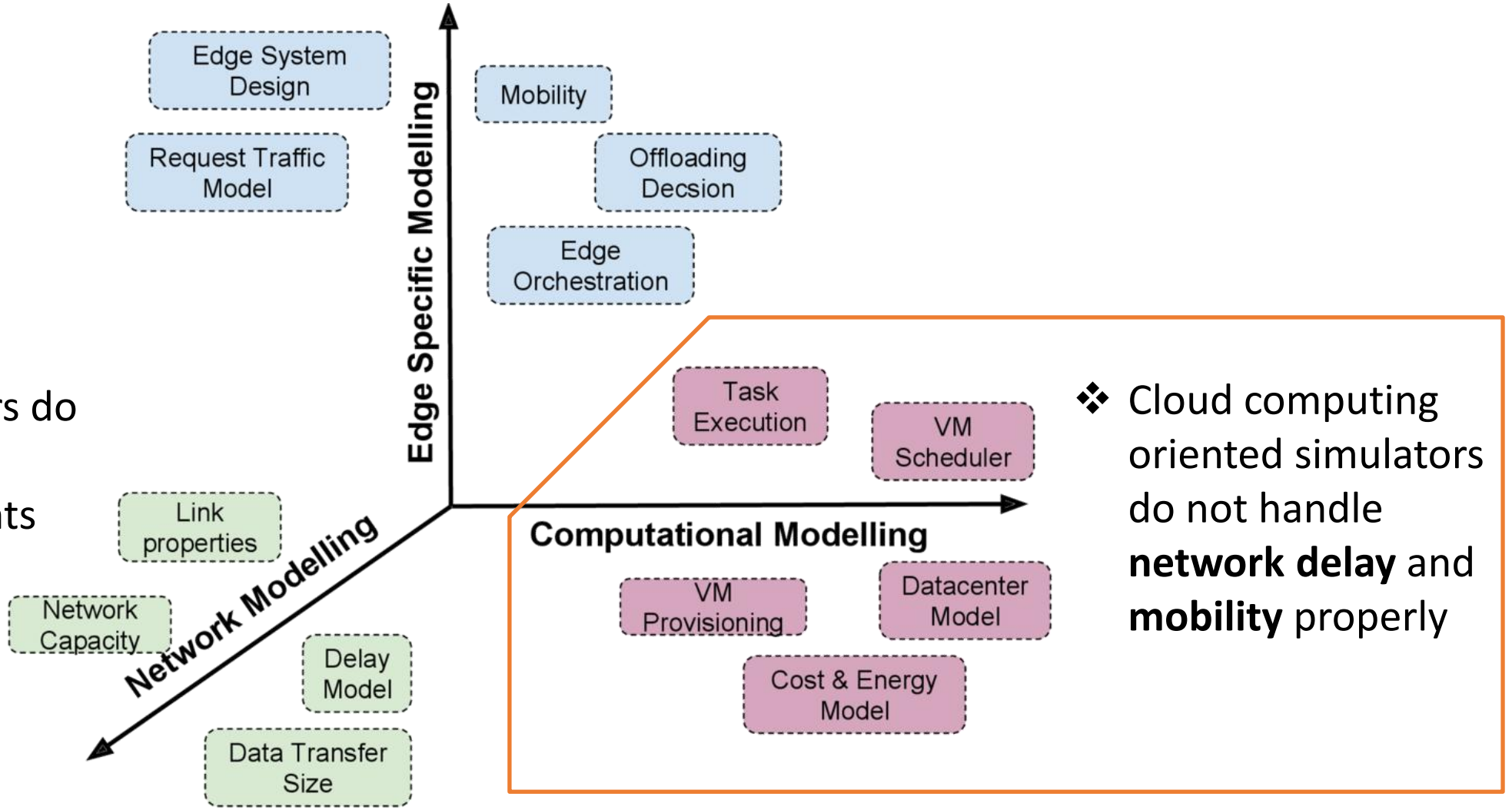
Motivation of Developing a Simulator

- Network simulators do not consider cloud computing elements
  - Datacenters
  - Hosts
  - VMs
  - Brokers



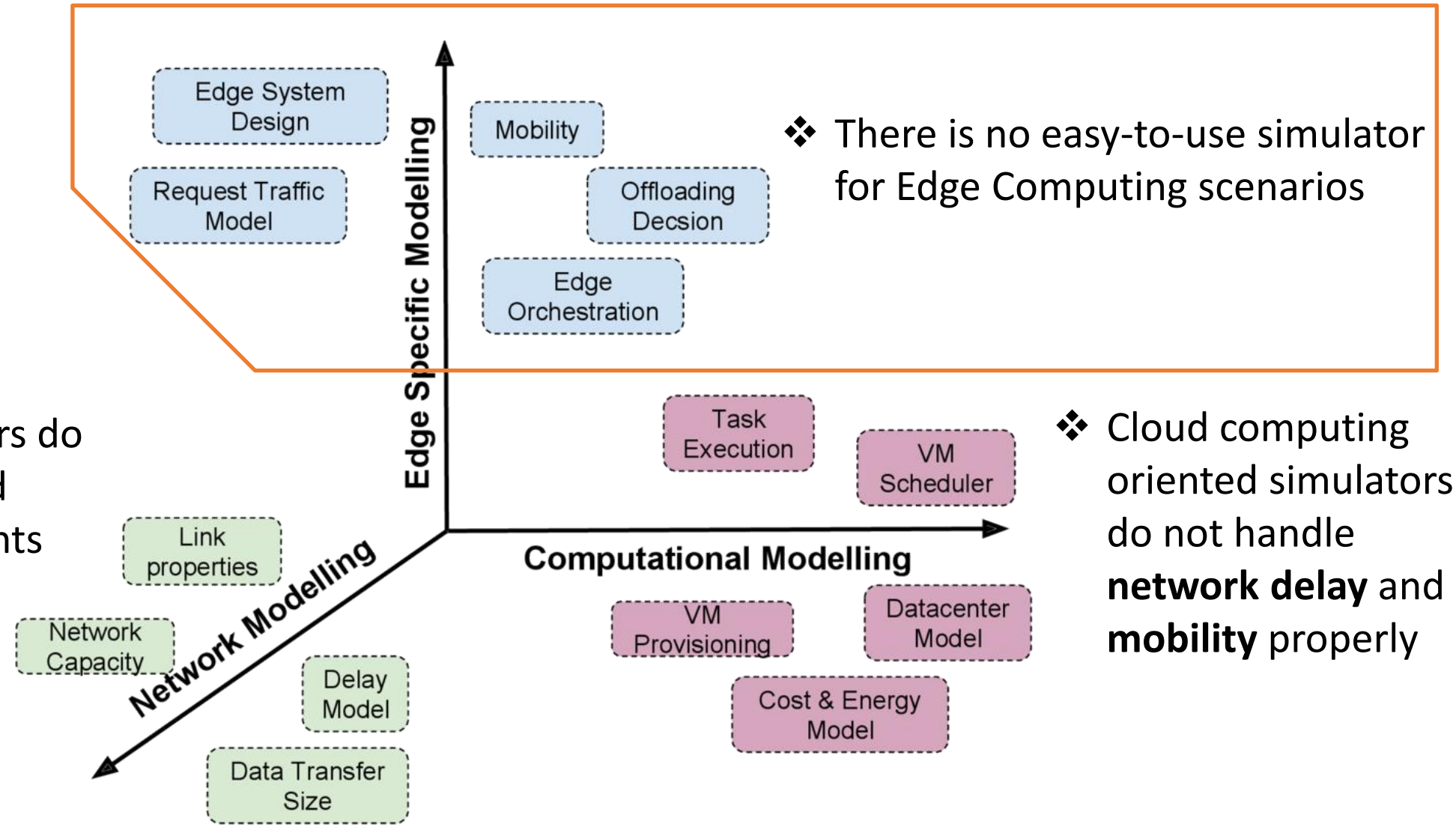
Motivation of Developing a Simulator

- Network simulators do not consider cloud computing elements
  - Datacenters
  - Hosts
  - VMs
  - Brokers



- Cloud computing oriented simulators do not handle **network delay** and **mobility** properly

Motivation of Developing a Simulator

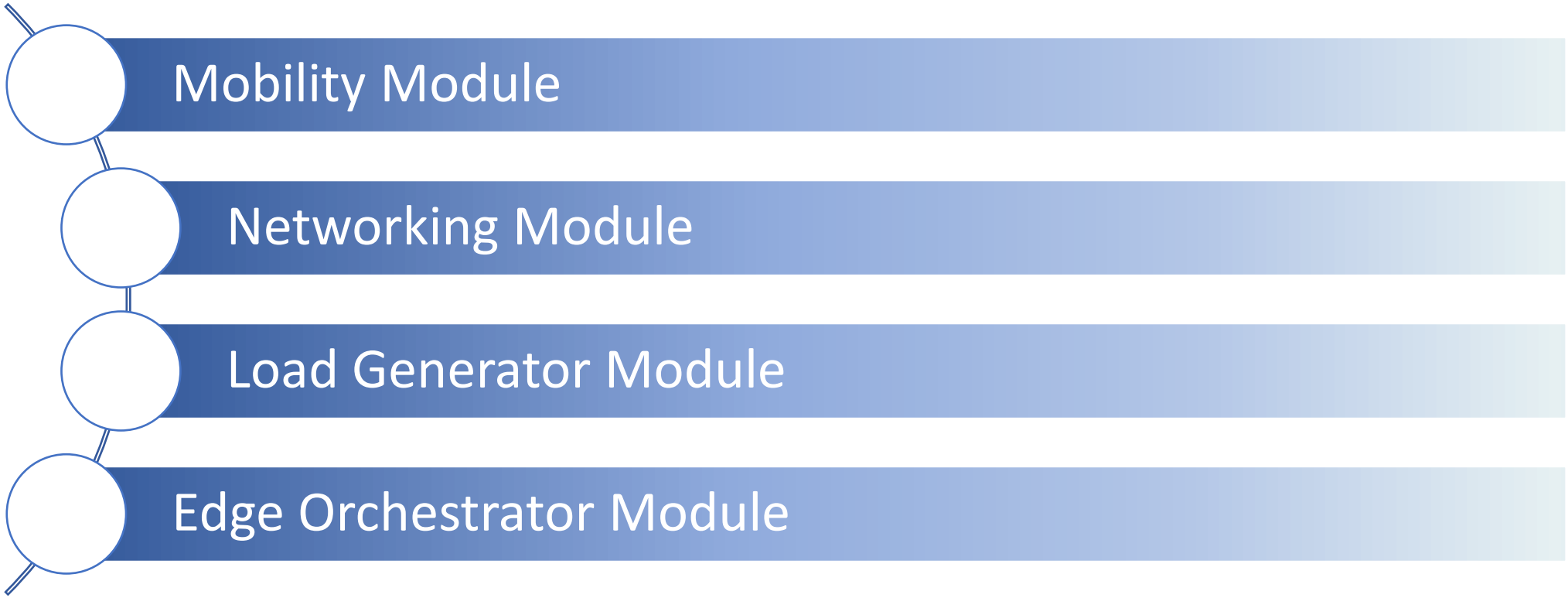


- There is no easy-to-use simulator for Edge Computing scenarios

- Network simulators do not consider cloud computing elements
  - Datacenters
  - Hosts
  - VMs
  - Brokers

- Cloud computing oriented simulators do not handle **network delay** and **mobility** properly

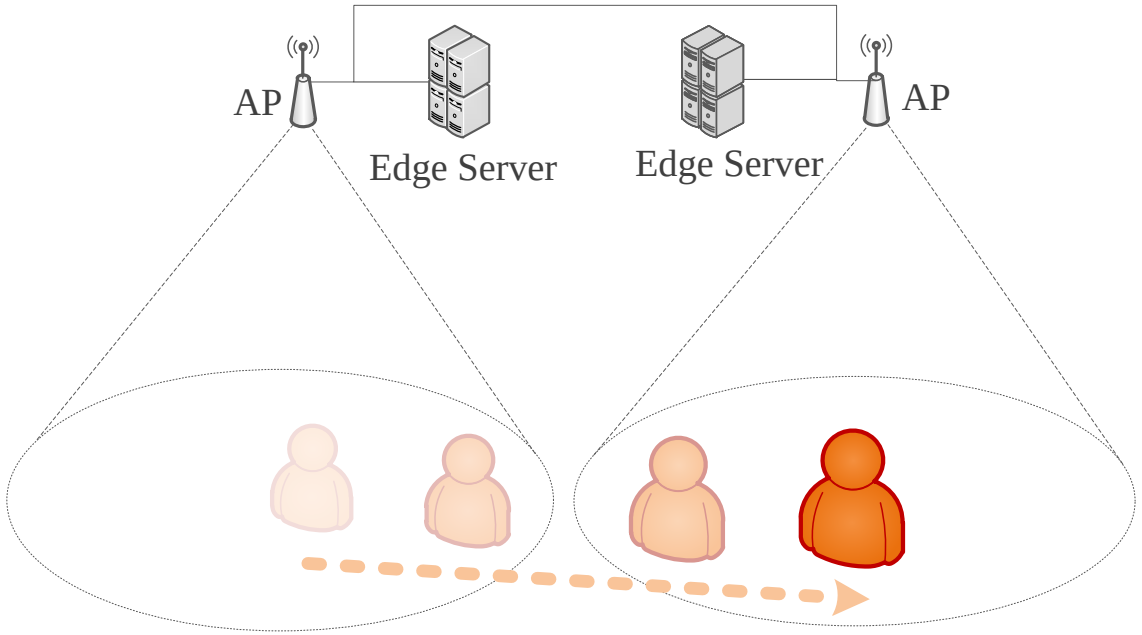
EdgeCloudSim Core Modules



EdgeCloudSim Core Modules cont.

Mobility Module

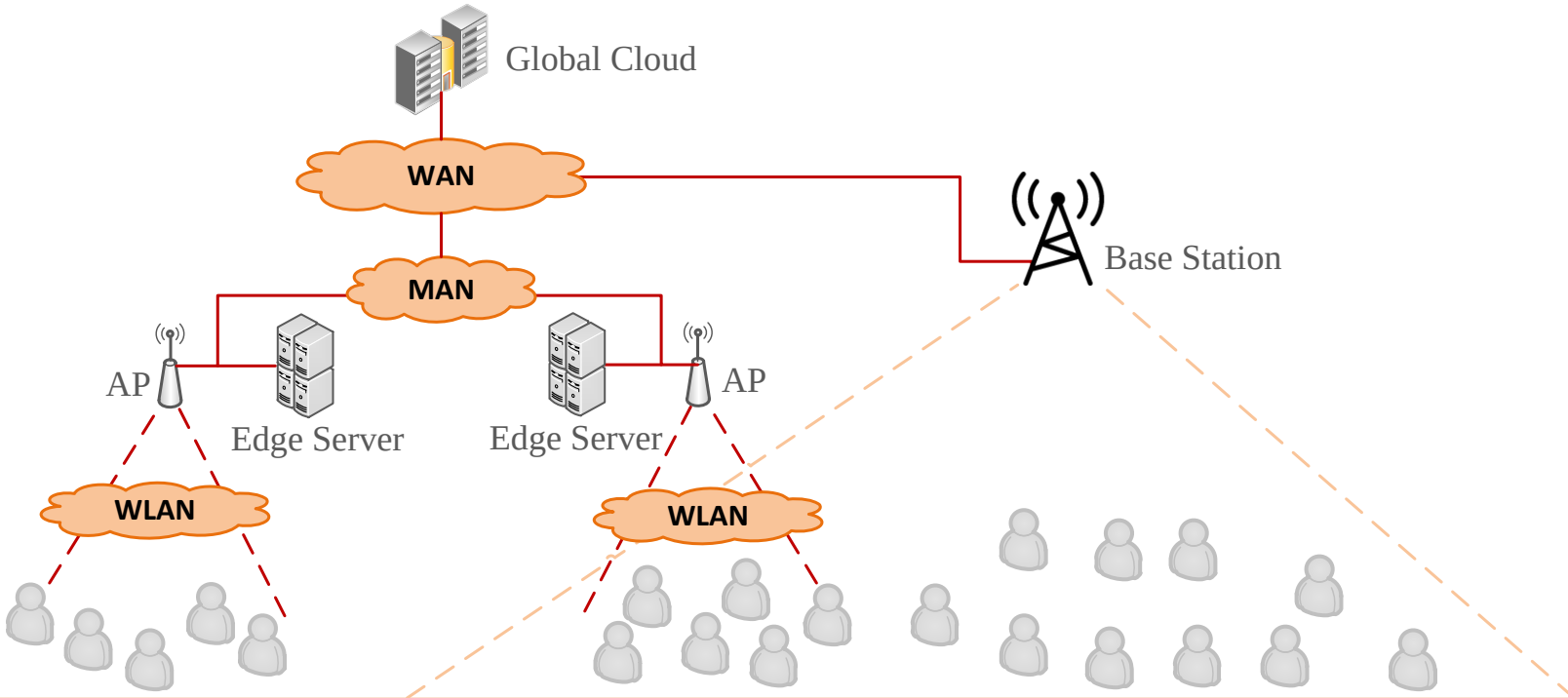
- Manages the location of edge devices and clients



EdgeCloudSim Core Modules cont.

Networking Module

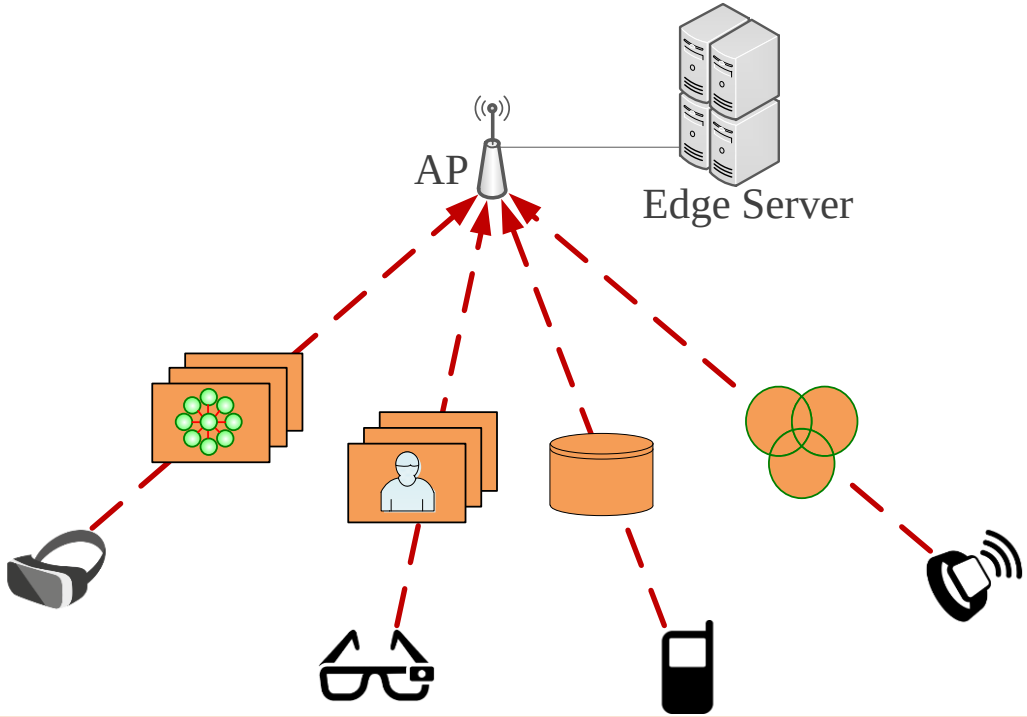
- Adds link delays between the network components



EdgeCloudSim Core Modules cont.

Load Generator Module

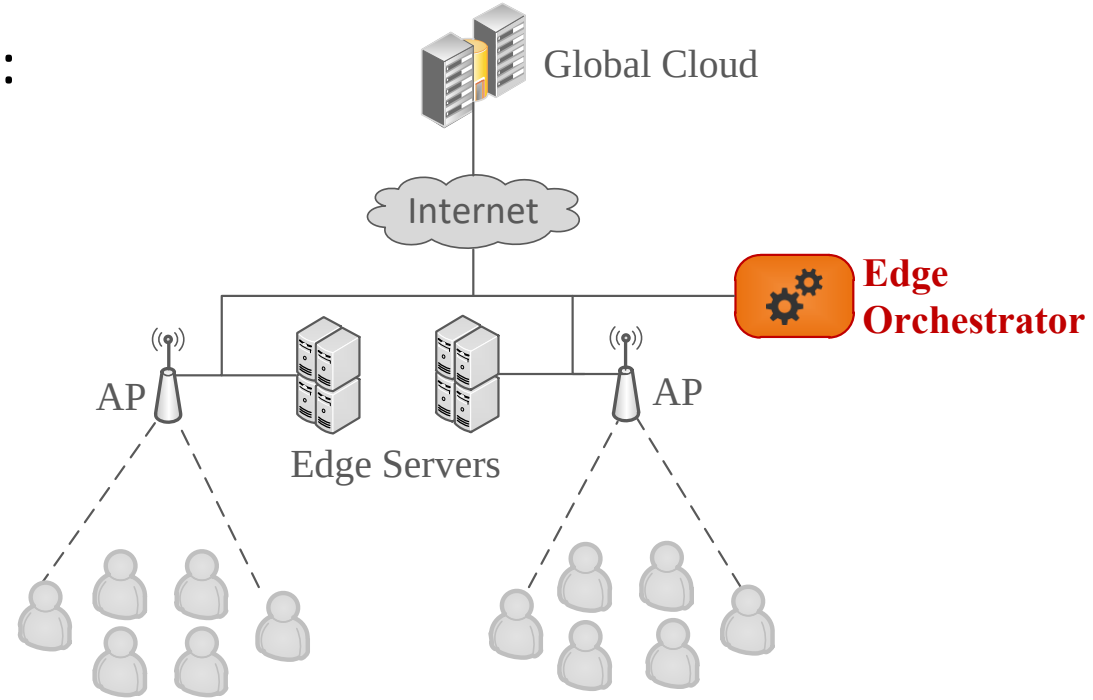
- Generates tasks based on the simulated scenario



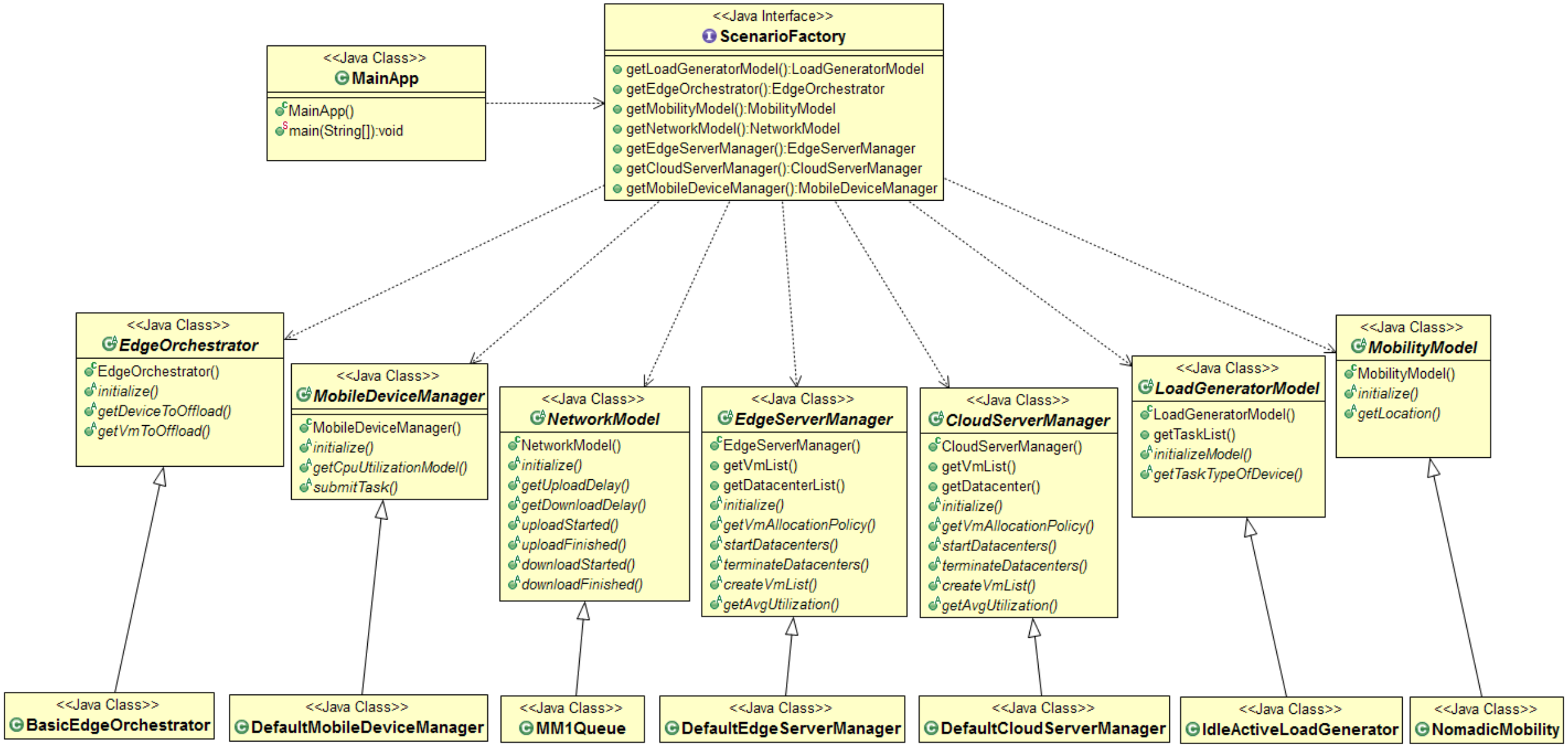
EdgeCloudSim Core Modules cont.

Edge Orchestrator Module

- The edge orchestrator can be considered as the central nervous system
- It makes critical decisions, such as:
  - Resource provisioning
  - Scales up/down the servers
  - Generates/terminates VMs
  - Migrates tasks
  - Coordinates services



Extensibility



Sample Factory Class

```
public class VehicularScenarioFactory implements ScenarioFactory {
	...
	@Override
	public LoadGeneratorModel getLoadGeneratorModel() {
		return new VehicularLoadGenerator(numOfMobileDevice, simulationTime, simScenario);
	}
	@Override
	public EdgeOrchestrator getEdgeOrchestrator() {
		return new VehicularEdgeOrchestrator(numOfMobileDevice, orchestratorPolicy, simScenario);
	}
	@Override
	public MobilityModel getMobilityModel() {
		return new VehicularMobilityModel(numOfMobileDevice,simulationTime);
	}
	@Override
	public NetworkModel getNetworkModel() {
		return new VehicularNetworkModel(numOfMobileDevice, simScenario, orchestratorPolicy);
	}
	@Override
	public EdgeServerManager getEdgeServerManager() {
		return new VehicularEdgeServerManager();
	}
	@Override
	public CloudServerManager getCloudServerManager() {
		return new DefaultCloudServerManager();
	}
	@Override
	public MobileDeviceManager getMobileDeviceManager() throws Exception {
		return new VehicularMobileDeviceManager();
	}
	@Override
	public MobileServerManager getMobileServerManager() {
		return new VehicularMobileServerManager(numOfMobileDevice);
	}
```

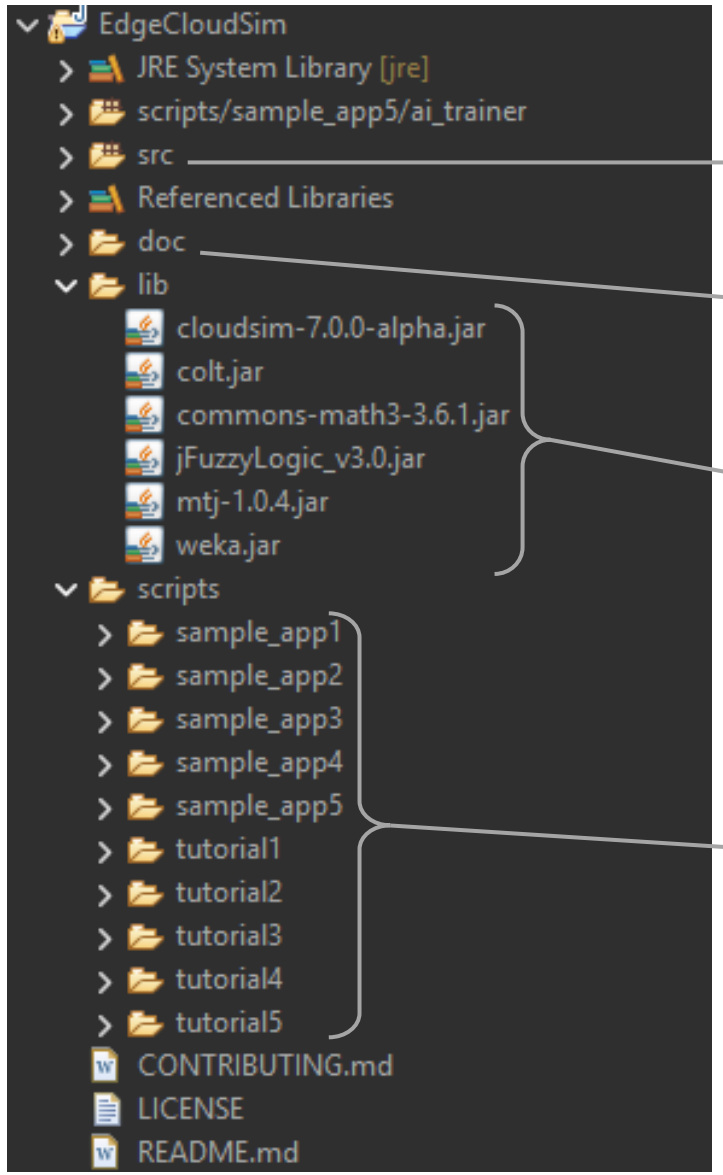
Extending Network Model…

```
public class VehicularNetworkModel extends NetworkModel {
        public static double maxWlanDelay = 0;
        public static double maxWanDelay = 0;
        public static double maxGsmDelay = 0;

        private class MMPPWrapper {
                private double currentPoissonMean;
                private double currentTaskSize;

                //record last values used for successful packet transmission
                private double lastPoissonMean;
                private double lastTaskSize;

                //record last n task statistics during MM1_QUEUE_MODEL_UPDATE
                private double numOfTasks;
                private double totalTaskSize;

                public MMPPWrapper() {
                        currentPoissonMean = 0;
                        currentTaskSize = 0;

                        lastPoissonMean = 0;
                        lastTaskSize = 0;

                        numOfTasks = 0;
                        totalTaskSize = 0;
                }
        ...
```

Downloading EdgeCloudSim

- EdgeCloudSim is publicly available in GitHub.
- You should clone EdgeCloudSim repository to start development.
  - `$git clone https://github.com/CagataySonmez/EdgeCloudSim.git`
- After cloning the repository, you can use your favorite IDE such as Eclipse and NetBeans.
- Please check EdgeCloudSim wiki page for more.
  - Using command line
    - https://github.com/CagataySonmez/EdgeCloudSim/wiki/How-to-compile-EdgeCloudSim-application
  - Using Eclipse IDE
    - https://github.com/CagataySonmez/EdgeCloudSim/wiki/EdgeCloudSim-in-Eclipse:-step-by-step-installation-&-running-sample-application
  - Using Netbeans IDE
    - https://github.com/CagataySonmez/EdgeCloudSim/wiki/EdgeCloudSim-in-NetBeans:-step-by-step-installation-&-running-sample-application

EdgeCloudSim Folder Hierarchy



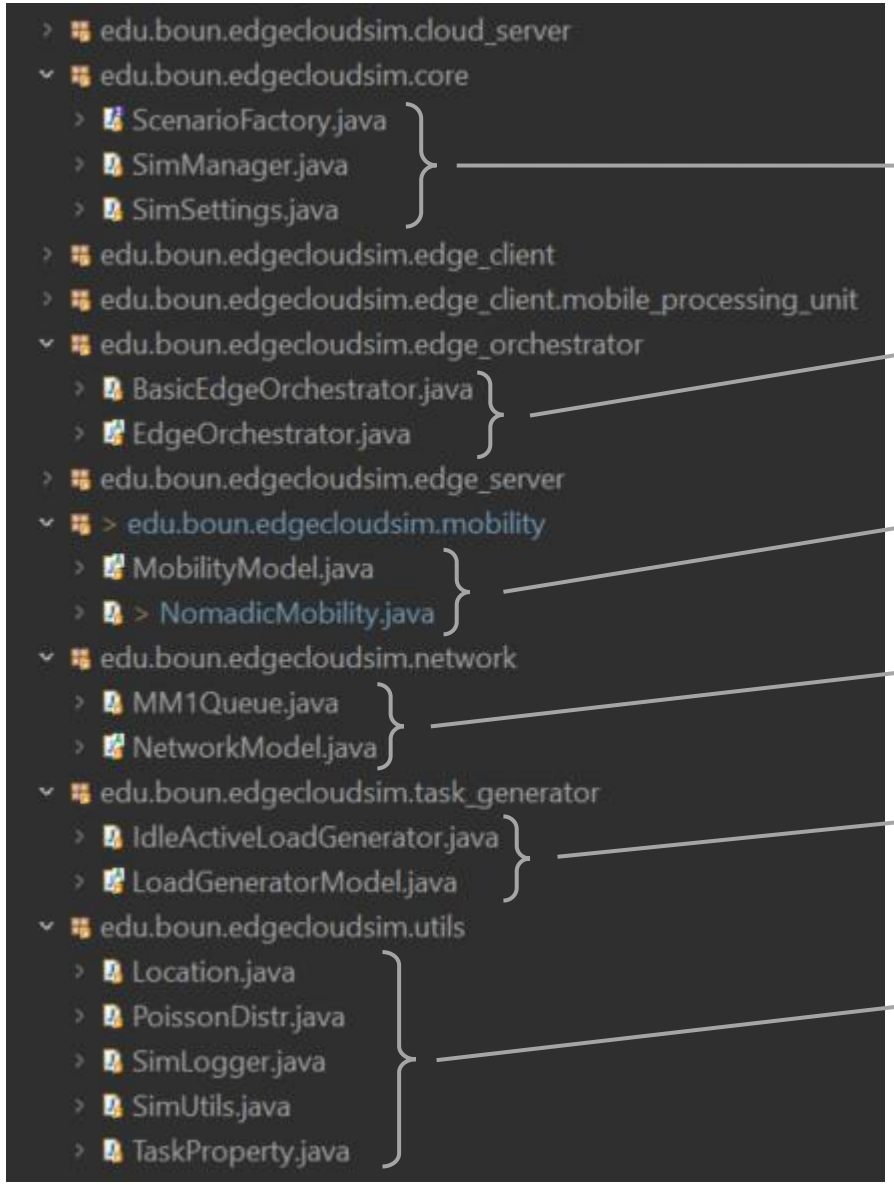
Source folder containing java source codes

Documents folder mostly used in GitHub readme files

Java libraries used by EdgeCloudSim and sample applications

Configuration files and Matlab (plotter) scripts for each sample application is stored under scripts folder

EdgeCloudSim Core Classes



core package of EdgeCloudSim and related classes

Abstract edge orchestrator class and its default implementation

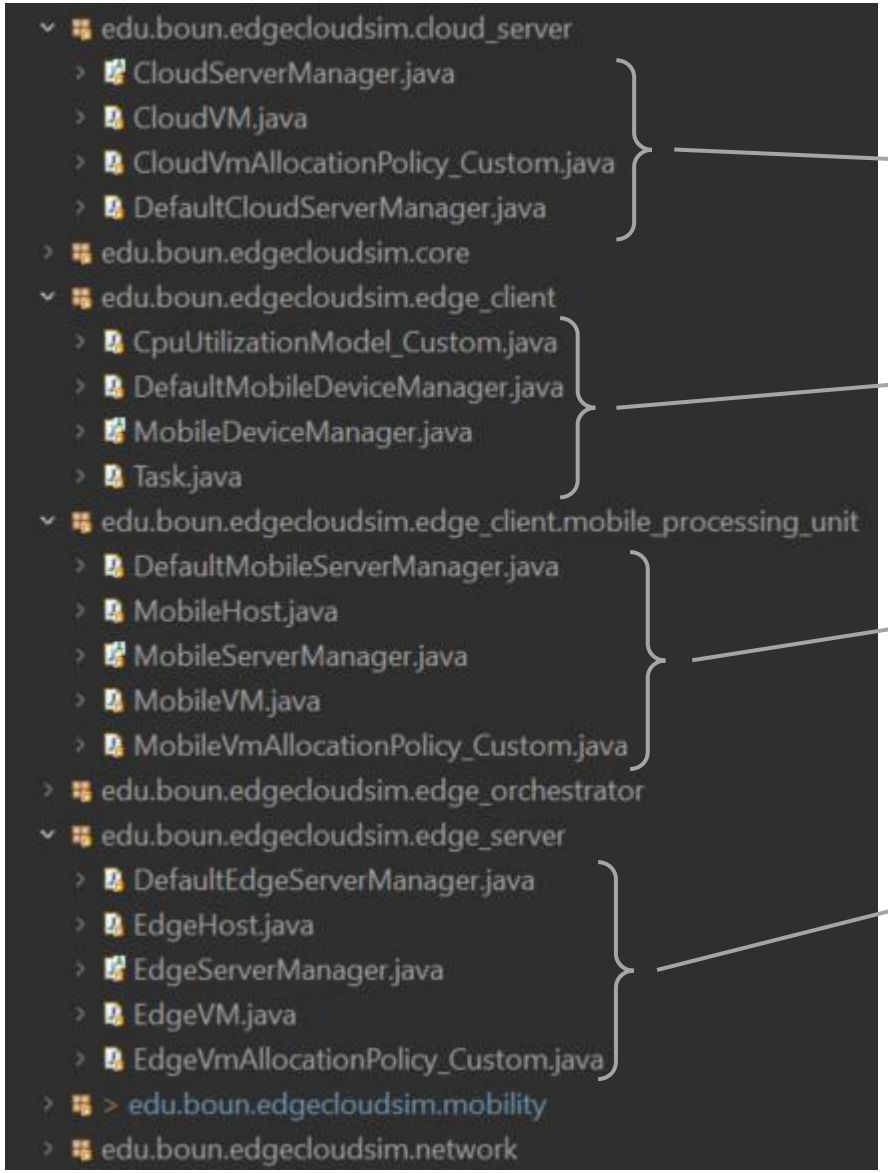
Abstract mobility model class and its default implementation

Abstract network model class and its default implementation

Abstract load generator class and its default implementation

utils package of EdgeCloudSim and related classes

EdgeCloudSim Computational Classes



cloud_server package includes classes for the cloud datacenter and associated hosts and VMs.

edge_client package includes classes that represents a broker (user).

mobile_processing_unit package includes classes for the mobile datacenter and associated hosts and VMs.

edge_server package includes classes for the edge datacenter and associated hosts and VMs.

Important Classes: SimSettings

```java
package edu.boun.edgecloudsim.core;

import java.io.File;

public class SimSettings {
    private static SimSettings instance = null;
    private Document edgeDevicesDoc = null;

    public static final double CLIENT_ACTIVITY_START_TIME = 10;

    //predifined IDs for the components.
    public static final int CLOUD_DATACENTER_ID = 1000;
    public static final int MOBILE_DATACENTER_ID = 1001;
    public static final int EDGE_ORCHESTRATOR_ID = 1002;
    public static final int GENERIC_EDGE_DEVICE_ID = 1003;

    //delimiter for output file.
    public static final String DELIMITER = ";";

    private double SIMULATION_TIME; //minutes unit in properties file
    private double WARM_UP_PERIOD; //minutes unit in properties file
    private double INTERVAL_TO_GET_VM_LOAD_LOG; //minutes unit in proper
    private double INTERVAL_TO_GET_LOCATION_LOG; //minutes unit in prope
    private double INTERVAL_TO_GET_AP_DELAY_LOG; //minutes unit in prope
    private boolean FILE_LOG_ENABLED; //boolean to check file logging op
    private boolean DEEP_FILE_LOG_ENABLED; //boolean to check deep file

    private int MIN_NUM_OF_MOBILE_DEVICES;
    private int MAX_NUM_OF_MOBILE_DEVICES;
    private int MOBILE_DEVICE_COUNTER_SIZE;
    private int WLAN_RANGE;

    private int NUM_OF_EDGE_DATACENTERS;
    private int NUM_OF_EDGE_HOSTS;
    private int NUM_OF_EDGE_VMS;
    private int NUM_OF_PLACE_TYPES;
```

The SimSettings class is located under core package, and responsible for storing all simulation settings by reading the values in the configuration files.

SimSettings class is not designed to be extended via the factory pattern; if you need to store different simulation settings for your simulation scenario, you will need to modify this core class!

config.properties file

```
#default config file
simulation_time=33
warm_up_period=3
vm_load_check_interval=0.1
location_check_interval=0.1
file_log_enabled=true
deep_file_log_enabled=false
min_number_of_mobile_devices=200
max_number_of_mobile_devices=2000
mobile_device_counter_size=200
wan_propagation_delay=0.1
lan_internal_delay=0.005
wlan_bandwidth=0
wan_bandwidth=0
gsm_bandwidth=0
...
#each mobile device has one host which serves one VM
#all the host runs on a single datacenter due to the
#out of memory (oom) issue
core_for_mobile_vm=1
mips_for_mobile_vm=4000
ram_for_mobile_vm=2000
storage_for_mobile_vm=32000
#use ',' for multiple values
orchestrator_policies=ONLY_EDGE,ONLY_MOBILE,HYBRID
#use ',' for multiple values
simulation_scenarios=MOBILE_PROCESSING_SCENARIO
...
```

Simulation and warm-up time

Values to use for file logging

Number of mobile devices to be used in the simulation

Values to be used in the network model

CPU, memory and storage specifications for mobile devices

Orchestrator policies and scenarios

★ Any variables related to simulation settings are kept in this file.

Important Classes: MobilityModel

```java
package edu.boun.edgecloudsim.mobility;

import edu.boun.edgecloudsim.utils.Location;

public abstract class MobilityModel {
    protected int numberOfMobileDevices;
    protected double simulationTime;

    public MobilityModel(int _numberOfMobileDevices, double _simulationTime){
        numberOfMobileDevices=_numberOfMobileDevices;
        simulationTime=_simulationTime;
    };

    /*
     * Default Constructor: Creates an empty MobilityModel
     */
    public MobilityModel() {
    }

    /*
     * calculate location of the devices according to related mobility model
     */
    public abstract void initialize();

    /*
     * returns location of a device at a certain time
     */
    public abstract Location getLocation(int deviceId, double time);
}
```

The MobilityModel class is located under mobility package, and responsible for providing location of the devices at a specific time.

You can implement a custom mobility model to handle your business logic. Be sure to use memory- and CPU-optimized approaches, such as utilizing a TreeMap, etc.

Extending Mobility Model…

```java
public class VehicularMobilityModel extends MobilityModel {
	private final double SPEED_FOR_PLACES[] = {20, 40, 60}; //km per hour

	private int lengthOfSegment;
	private double totalTimeForLoop; //seconds
	private int[] locationTypes;

	//prepare following arrays to decrease computation on getLocation() function
	//NOTE: if the number of clients is high, keeping following values in RAM
	//	may be expensive. In that case sacrifice computational resources!
	private int[] initialLocationIndexArray;
	private int[] initialPositionArray; //in meters unit
	private double[] timeToDriveLocationArray;//in seconds unit
	private double[] timeToReachNextLocationArray; //in seconds unit

	public VehicularMobilityModel(int _numberOfMobileDevices, double _simulationTime) {
		super(_numberOfMobileDevices, _simulationTime);
	}

	@Override
	public void initialize() {
		//Find total length of the road
		Document doc = SimSettings.getInstance().getEdgeDevicesDocument();
		NodeList datacenterList = doc.getElementsByTagName("datacenter");

	...
```

applications.xml file

```
<?xml version="1.0"?>
<applications>
  <application name="AUGMENTED_REALITY">
    <usage_percentage>30</usage_percentage>
    <prob_cloud_selection>20</prob_cloud_selection>
    <poisson_interarrival>2</poisson_interarrival>
    <delay_sensitivity>0</delay_sensitivity>
    <active_period>40</active_period>
    <idle_period>20</idle_period>
    <data_upload>1500</data_upload>
    <data_download>250</data_download>
    <task_length>12000</task_length>
    <required_core>1</required_core>
    <vm_utilization_on_edge>8</vm_utilization_on_edge>
    <vm_utilization_on_cloud>0.8</vm_utilization_on_cloud>
    <vm_utilization_on_mobile>20</vm_utilization_on_mobile>
  </application>
  <application name="HEALTH_APP">
    <usage_percentage>20</usage_percentage>
```

- How much this app is used by mobile devices
- Possibility of offloading to the cloud (for some scenarios)
- Interarrival time for task generation (Poisson process)
- How sensitive the app is to latency (for some scenarios)
- Active period for Active/Idle task generation model
- Idle period for Active/Idle task generation model
- Average size of data loaded by the task
- Average size of data downloaded after the task execution
- Average task length in millions of instructions (MI)
- How much CPU core is used by tasks created by this app
- CPU utilization ratio of related tasks on compute nodes

★ Applications create tasks to be offloaded with the properties specified in this file.

Important Classes: LoadGeneratorModel

```java
package edu.boun.edgecloudsim.task_generator;

import java.util.List;

public abstract class LoadGeneratorModel {
    protected List<TaskProperty> taskList;

    ...

    /*
     * each task has a virtual start time
     * it will be used while generating task
     */
    public List<TaskProperty> getTaskList() {
        return taskList;
    }

    /*
     * fill task list according to related task generation model
     */
    public abstract void initializeModel();

    /*
     * returns the task type (index) that the mobile device uses
     */
    public abstract int getTaskTypeOfDevice(int deviceId);
}
```

The LoadGeneratorModel is responsible for generating a task list that includes task properties, such as the task start time, etc.

Most applications use the basic IdleActiveLoadGeneratorModel, but you can implement your own task generation model by preparing a custom task list.

edge_devices.xml file

```
<?xml version="1.0"?>
<edge_devices>
        <datacenter arch="x86" os="Linux" vmm="Xen">
                <costPerBw>0.1</costPerBw>
                <costPerSec>3.0</costPerSec>
                <costPerMem>0.05</costPerMem>
                <costPerStorage>0.1</costPerStorage>
                <location>
                        <x_pos>1</x_pos>
                        <y_pos>1</y_pos>
                        <wlan_id>0</wlan_id>
                        <attractiveness>0</attractiveness>
                </location>
                <hosts>
                        <host>
                                <core>16</core>
                                <mips>80000</mips>
                                <ram>16000</ram>
                                <storage>400000</storage>
                                <VMs>
                                        <VM vmm="Xen">
                                                <core>2</core>
                                                <mips>10000</mips>
                                                <ram>2000</ram>
                                                <storage>50000</storage>
                                        </VM>
                                        <VM vmm="Xen">
                                                <core>2</core>
                                                <mips>10000</mips>
                                                <ram>2000</ram>
                                                <storage>50000</storage>
```

Bandwidth, CPU, memory and storage cost values for this datacenter (based on CloudSim specifications)

x, y position of the datacenter (will be important for your mobility model)

Each WLAN should have a unique id

Attractiveness level of this location (for some scenarios)

CPU, memory and storage specifications for the corresponding host

CPU, memory and storage specifications for the corresponding host

★ Edge servers are created with properties specified in this file

Important Classes: *ServerManager

```java
package edu.boun.edgecloudsim.cloud_server;

import java.util.ArrayList;

public abstract class CloudServerManager {
    protected Datacenter localDatacenter;
    protected List<List<CloudVM>> vmList;
```

```java
package edu.boun.edgecloudsim.edge_server;

import java.util.ArrayList;

public abstract class EdgeServerManager {
    protected List<Datacenter> localDatacenters;
    protected List<List<EdgeVM>> vmList;
```

```java
package edu.boun.edgecloudsim.edge_client.mobile_processing_unit;

import java.util.ArrayList;

public abstract class MobileServerManager {
    protected Datacenter localDatacenter;
    protected List<List<MobileVM>> vmList;
```

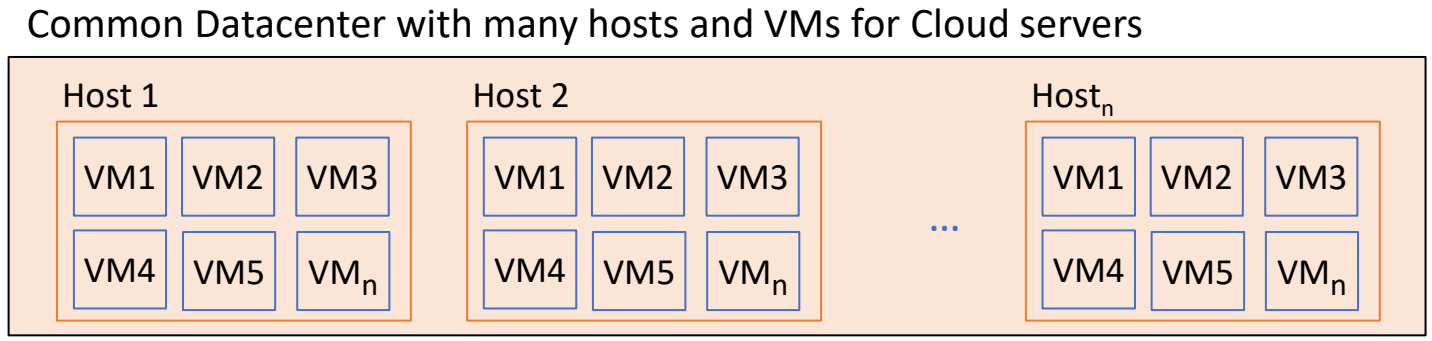
These classes are responsible for creating computational resources provided by EdgeCloudSim.

Each class's default implementation uses the configuration values related to the corresponding server (processing unit) on the cloud, edge, or mobile device.

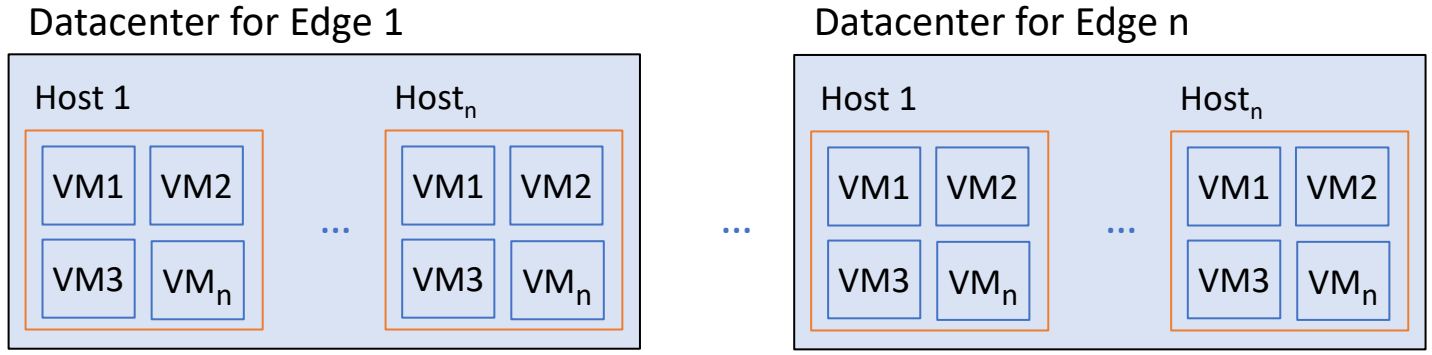
The name MobileServerManager may not seem intuitive. It represents the processing unit of the mobile device, and this name was chosen to follow a common convention.

Datacenter, Host and VM Hierarchy in EdgeCloudSim

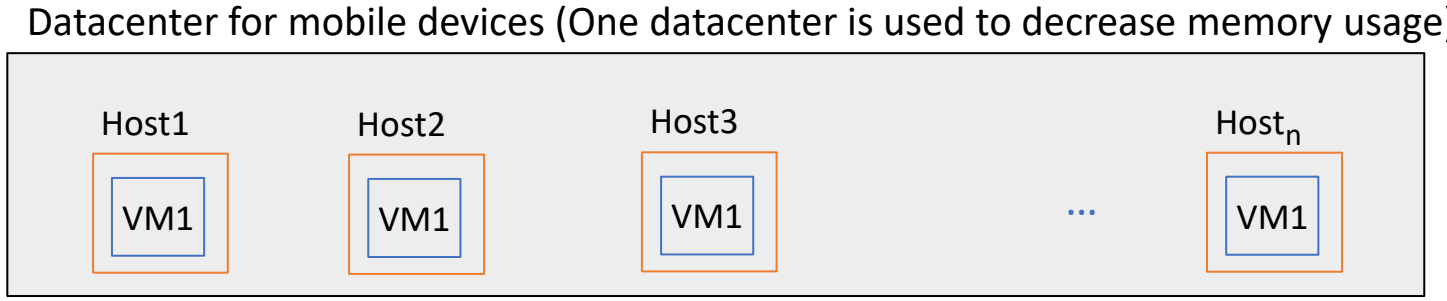
Cloud Layer



Edge Layer



End User Layer



Using Conf. Files

```
String configFile = "";
String outputFolder = "";
String edgeDevicesFile = "";
String applicationsFile = "";
if (args.length == 5){
        configFile = args[0];
        edgeDevicesFile = args[1];
        applicationsFile = args[2];
        outputFolder = args[3];
        iterationNumber = Integer.parseInt(args[4]);
}
else{
        SimLogger.printLine("Simulation setting file, output folder and iterati
        configFile = "scripts/sample_app1/config/default_config.properties";
        applicationsFile = "scripts/sample_app1/config/applications.xml";
        edgeDevicesFile = "scripts/sample_app1/config/edge_devices.xml";
        outputFolder = "sim_results/ite" + iterationNumber;
}

//load settings from configuration file
SimSettings SS = SimSettings.getInstance();
if(SS.initialize(configFile, edgeDevicesFile, applicationsFile) == false){
        SimLogger.printLine("cannot initialize simulation settings!");
        System.exit(0);
}
```

Configuration files are mostly provided via command line arguments (especially when running through terminal).

Sometimes developers can use static file path for convenience (especially when running the simulations through IDEs during development).

SimSettings is initialized with the configuration files.

Important Classes: ScenarioFactory

```java
public interface ScenarioFactory {
    /**
     * provides abstract Load Generator Model
     */
    public LoadGeneratorModel getLoadGeneratorModel();

    /**
     * provides abstract Edge Orchestrator
     */
    public EdgeOrchestrator getEdgeOrchestrator();

    /**
     * provides abstract Mobility Model
     */
    public MobilityModel getMobilityModel();

    /**
     * provides abstract Network Model
     */
    public NetworkModel getNetworkModel();

    /**
     * provides abstract Edge Server Model
     */
    public EdgeServerManager getEdgeServerManager();

    /**
     * provides abstract Cloud Server Model
     */
    public CloudServerManager getCloudServerManager();

    /**
     * provides abstract Mobile Server Model
     */
    public MobileServerManager getMobileServerManager();

    /**
     * provides abstract Mobile Device Manager Model
     */
    public MobileDeviceManager getMobileDeviceManager() throws Exception;
}
```

The ScenarioFactory class provides extensibility. You can provide versions of most of the important classes with different behaviors to implement your simulation scenario.

You have to provide a concrete implementation of ScenarioFactory class to create the SimManager class and run the simulation in your java main method.

```java
// Generate EdgeCloudsim Scenario Factory
ScenarioFactory sampleFactory = new SampleScenarioFactory(j,

// Generate EdgeCloudSim Simulation Manager
SimManager manager = new SimManager(sampleFactory, j, simSce

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

Sample code snippet from Java main method

Important Classes: SimManager

```
/*
 * Title:        EdgeCloudSim - Simulation Manager
 *
 * Description:
 * SimManager is an singleton class providing many abstract classes such as
 * Network Model, Mobility Model, Edge Orchestrator to other modules
 * Critical simulation related information would be gathered via this class
 *
 * Licence:      GPL - http://www.gnu.org/copyleft/gpl.html
 * Copyright (c) 2017, Bogazici University, Istanbul, Turkey
 */

package edu.boun.edgecloudsim.core;
import java.io.IOException;
public class SimManager extends SimEntity {

    private static SimManager instance = null;

    public SimManager(ScenarioFactory _scenarioFactory, int _numOfMobileDevice,
        super("SimManager");

        SimLogger.print("Creating tasks...");
        loadGeneratorModel = scenarioFactory.getLoadGeneratorModel();
        loadGeneratorModel.initializeModel();
        SimLogger.printLine("Done, ");

        SimLogger.print("Creating device locations...");
        mobilityModel = scenarioFactory.getMobilityModel();
        mobilityModel.initialize();
        SimLogger.printLine("Done.");

        //Generate network model
        networkModel = scenarioFactory.getNetworkModel();
```

SimManager provides many abstract classes such as Network Model, Mobility Model, Edge Orchestrator to other modules.

You need to extend SimEntity to create custom events!

This class was intended to be a singleton; however, due to specific requirements for its recreation, it could not fully maintain singleton properties.

★ This class handles important events such as task creation, simulation termination etc.

Important Classes: EdgeOrchestrator

```java
package edu.boun.edgecloudsim.edge_orchestrator;

import org.cloudbus.cloudsim.Vm;

public abstract class EdgeOrchestrator extends SimEntity{
    protected String policy;
    protected String simScenario;

    public EdgeOrchestrator(String _policy, String _simScenario){
        super("EdgeOrchestrator");
        policy = _policy;
        simScenario = _simScenario;
    }

    /*
     * Default Constructor: Creates an empty EdgeOrchestrator
     */
    public EdgeOrchestrator() {
            super("EdgeOrchestrator");
    }

    /*
     * initialize edge orchestrator if needed
     */
    public abstract void initialize();

    /*
     * decides where to offload
     */
    public abstract int getDeviceToOffload(Task task);

    /*
     * returns proper VM from the edge orchestrator point of view
     */
    public abstract Vm getVmToOffload(Task task, int deviceId);
}
```

The edge orchestrator can be considered as the central nervous system. Make all critical decisions here in this class.

You need to extend SimEntity to create custom events!

This generic function is mostly used to decide which task should be sent to which datacenter.

This generic function is mostly used to decide which VM the task should be assigned to.

You can add other helper functions in your concrete implementation of this class to handle your business logic!

Important Classes: MobileDeviceManager

```java
package edu.boun.edgecloudsim.edge_client;

import org.cloudbus.cloudsim.DatacenterBroker;

public abstract class MobileDeviceManager  extends DatacenterBroker {

    public MobileDeviceManager() throws Exception {
        super("Global_Broker");
    }

    /*
     * initialize mobile device manager if needed
     */
    public abstract void initialize();

    /*
     * provides abstract CPU Utilization Model
     */
    public abstract UtilizationModel getCpuUtilizationModel();

    public abstract void submitTask(TaskProperty edgeTask);
}
```

MobileDeviceManager extends CloudSim's DatacenterBroker class which represents a broker (user).

CloudSim does not provide realistic VM CPU utilization model, so that this class is responsible for providing a CPU utilization model based on the simulation scenario.

It is basically responsible for submitting VM provisioning requests to data centers (submitting tasks to VMs).

★ For convenience, this class often handles (simulates) the tasks transition between the entities.

Important Classes: SimLogger

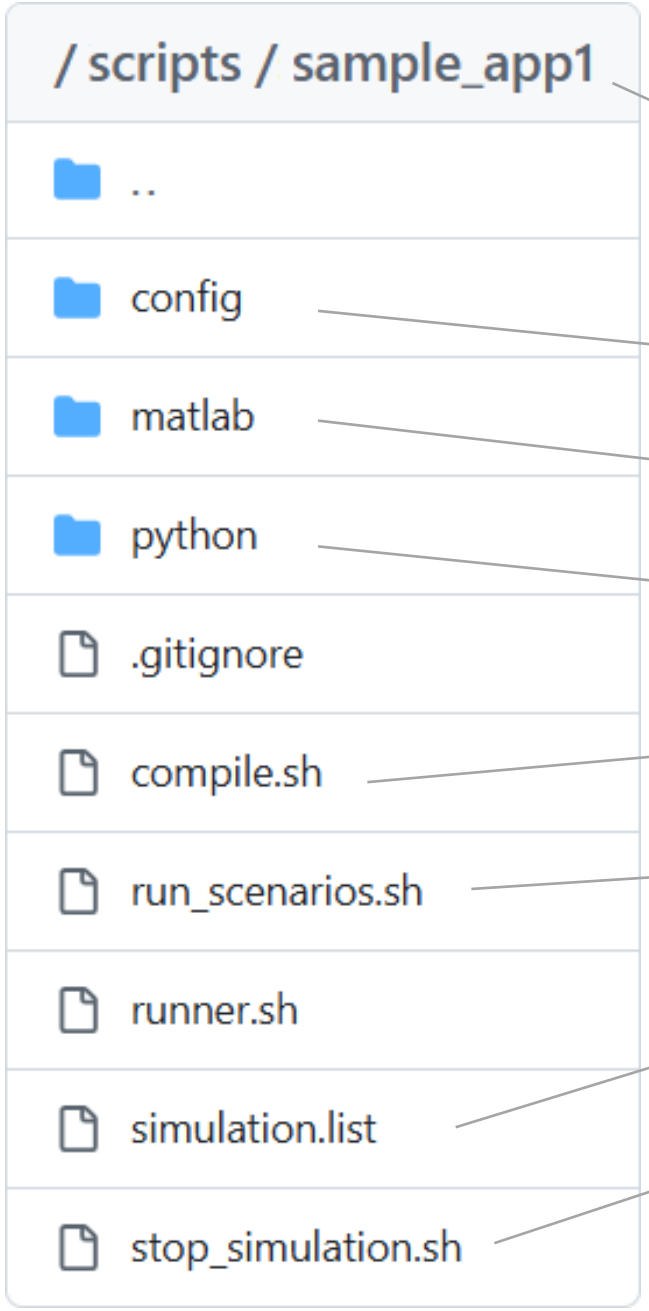
```java
package edu.boun.edgecloudsim.utils;

import java.io.BufferedWriter;

public class SimLogger {
    public static enum TASK_STATUS {
        CREATED, UPLOADING, PROCESSING, DOWNLOADING, COMLETED,
        REJECTED_DUE_TO_VM_CAPACITY, REJECTED_DUE_TO_BANDWIDTH,
        UNFINISHED_DUE_TO_BANDWIDTH, UNFINISHED_DUE_TO_MOBILITY,
        REJECTED_DUE_TO_WLAN_COVERAGE
    }

    public static enum NETWORK_ERRORS {
        LAN_ERROR, MAN_ERROR, WAN_ERROR, GSM_ERROR, NONE
    }

    private long startTime;
    private long endTime;
    private static boolean fileLogEnabled;
    private static boolean printLogEnabled;
    private String filePrefix;
    private String outputFolder;
    private Map<Integer, LogItem> taskMap;
    private LinkedList<VmLoadLogItem> vmLoadList;
    private LinkedList<ApDelayLogItem> apDelayList;

    private static SimLogger singleton = new SimLogger();

    private int numOfAppTypes;

    private File successFile = null, failFile = null;
    private FileWriter successFW = null, failFW = null;
    private BufferedWriter successBW = null, failBW = null;
```

The SimSettings class is located under util package, and responsible for saving simulation results to files.

The first version of SimLogger was logging events to files as they happened, but this approach requires a lot of I/O. Therefore, it was updated to collect the results of the events in data structures and write them to the files at the end of the simulation!

★ SimLogger class is not designed to be extended via the factory pattern; if you need to log different simulation results, you will need to modify this core class!

Helper Scripts



Each application has utilities in its scripts folder to run simulations and plot results easily

Configuration files located in config folder

MATLAB scripts to plot graphs

Python scripts to plot graphs

Script to compile sample app

Script to run sample app

Used to run multiple simulations with different config files

Script to stop all simulation processes

★ Scripts are tested & verified on Linux based operating systems including Mac OS.

Running Applications on Linux Terminal

1. **Open Terminal:** Open your terminal and navigate to

```
cd EdgeCloudSim/scripts/<sample_app>
```

2. **Compile the Code:** Run the compile script

```
./compile.sh
```

3. **Run Scenarios:** Execute the simulation with

```
./run_scenarios.sh <num_parallel_processes> <num_iterations>
```

4. **View the Results:** Simulation outputs will be automatically generated under

```
EdgeCloudSim/scripts/<sample_app>/output/<date>/default_config/
```

5. **Visualize the Results:** Use the provided Python or MATLAB scripts to plot and analyze your results.

More info: https://github.com/CagataySonmez/EdgeCloudSim/wiki/Running-sample-application-on-the-Linux-terminal

Running Applications on Your IDE

```
// Parse command line arguments:
// Expected: 0:config 1:edge_devices 2:applications 3:output_folder 4:iteration
// If not provided, fall back to defaults for quick local tests.
if (args.length == EXPECTED_NUM_OF_ARGS){
    configFile = args[0];
    edgeDevicesFile = args[1];
    applicationsFile = args[2];
    outputFolder = args[3];
    iterationStart = Integer.parseInt(args[4]);
    iterationEnd = iterationStart;
}
else{
    // Inform user that defaults are used (common in IDE debugging)
    SimLogger.printLine("Simulation setting file, output folder and iteration number are not
    configFile = "scripts/" + APPLICATION_FOLDER + "/config/default_config.properties";
    applicationsFile = "scripts/" + APPLICATION_FOLDER + "/config/applications.xml";
    edgeDevicesFile = "scripts/" + APPLICATION_FOLDER + "/config/edge_devices.xml";

    // !! IMPORTANT NOTICE !!
    // For those who are using IDE (eclipse etc) can modify
    // -> iteration value to run a specific iteration
    // -> iteration Start/End value to run multiple iterations at a time
    //    in this case start shall be less than or equal to end value
    int iteration = 1;
    iterationStart = iteration;
    iterationEnd = iteration;
}
```

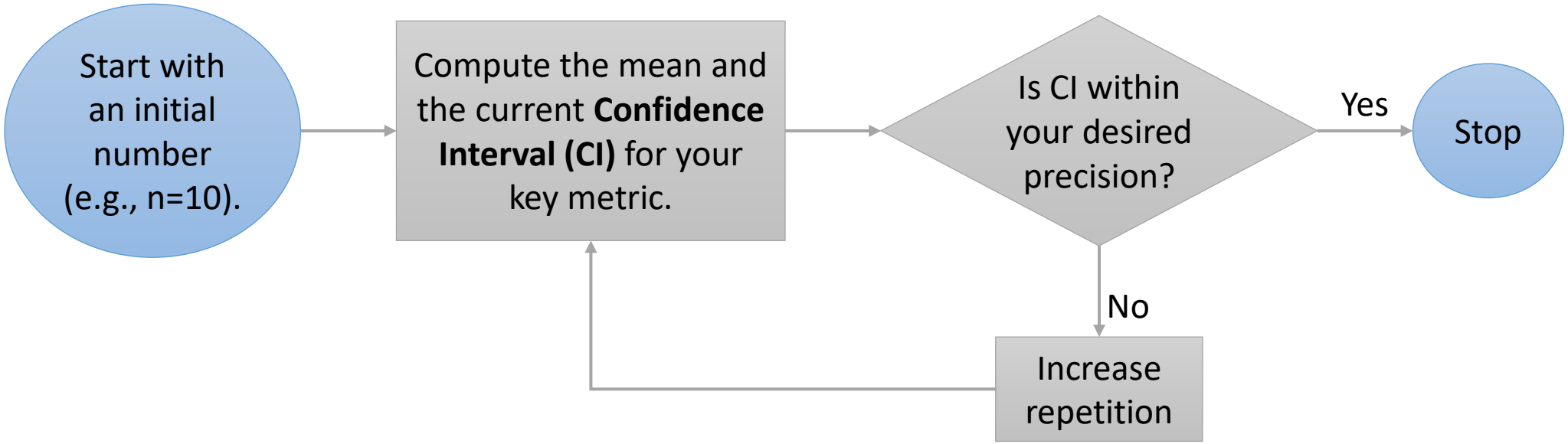
★ It is recommended to run the scripts via the Linux terminal for proper execution. However, for fast prototyping, running them directly through an IDE can be more convenient.

Users who prefer this approach can execute the main method without providing any arguments, but they must ensure that the values displayed in the figures are correctly set beforehand.

More info: https://github.com/CagataySonmez/EdgeCloudSim/wiki/EdgeCloudSim-in-Eclipse:-step-by-step-installation-&-running-sample-application

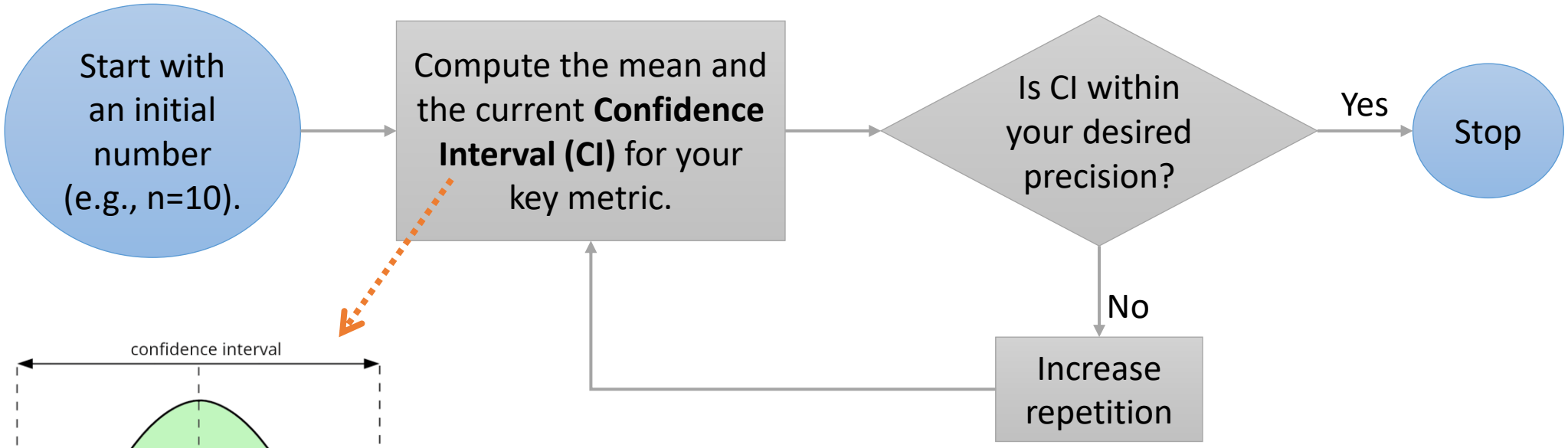
How Many Repetitions?

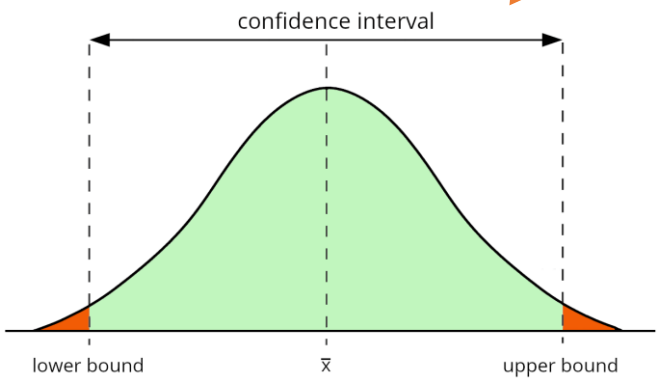
- ✓ The required number of repetitions depends on your results and your research goals.
- ✓ The most important factor is the **variance**!
- ✓ You should run the simulation until the desired Confidence Interval (CI) is achieved.



How Many Repetitions?

- ✓ The required number of repetitions depends on your results and your research goals.
- ✓ The most important factor is the **variance**!
- ✓ You should run the simulation until the desired Confidence Interval (CI) is achieved.





$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}}$$

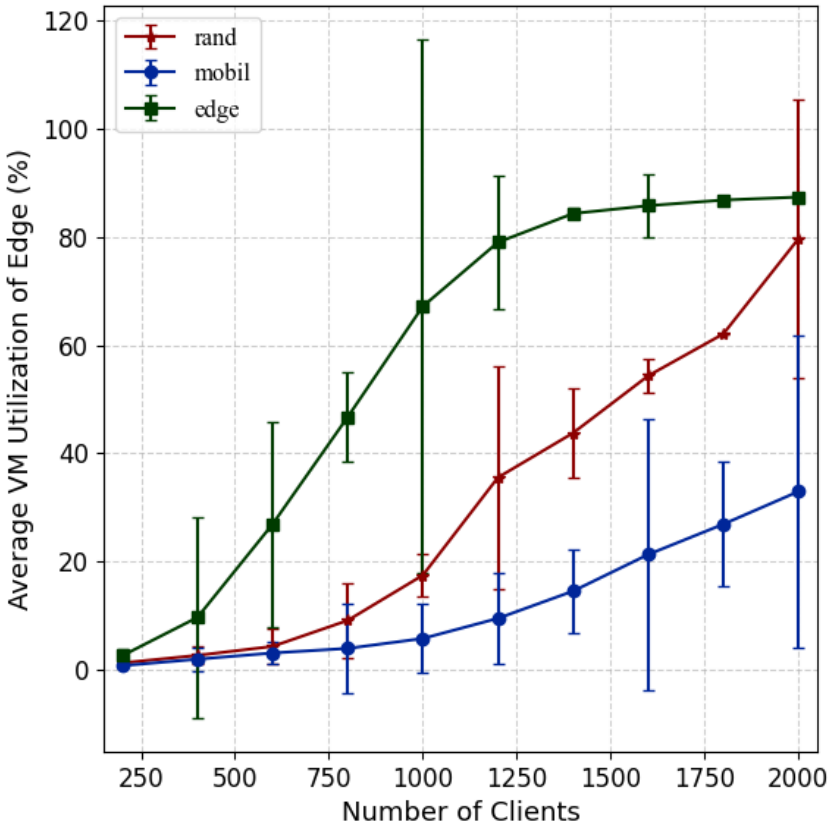
$\bar{x}$ = mean s = std deviation

z = confidence level n = sample size

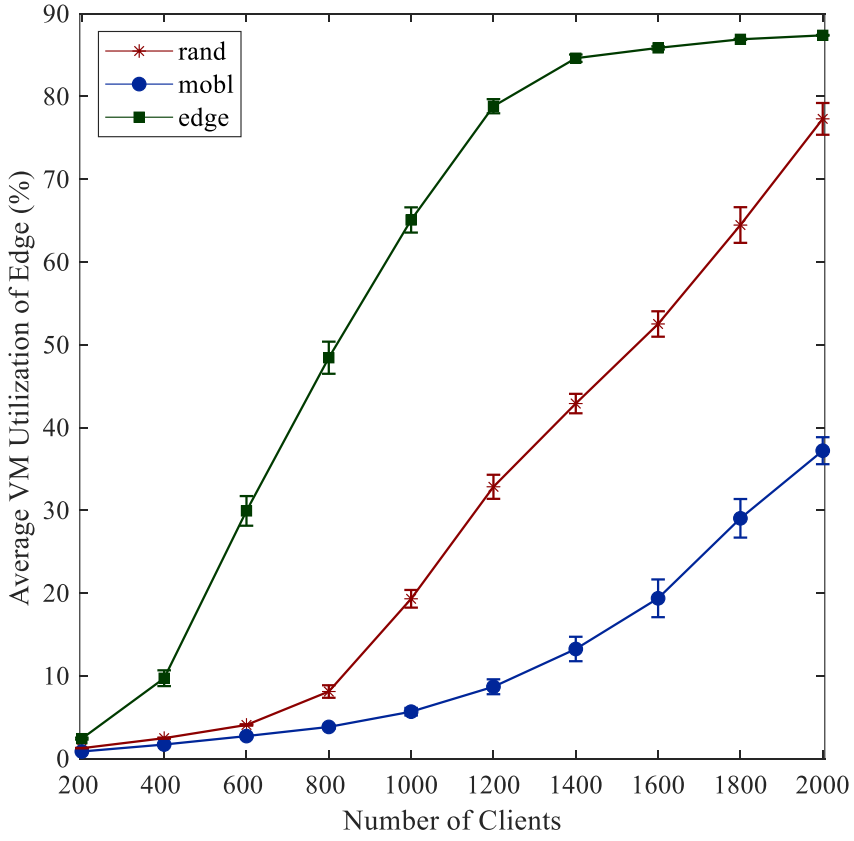
2 vs 10 Repetitions

- Performing only a few repetitions leads to a wider Confidence Interval, decreasing the accuracy and reliability of the results.

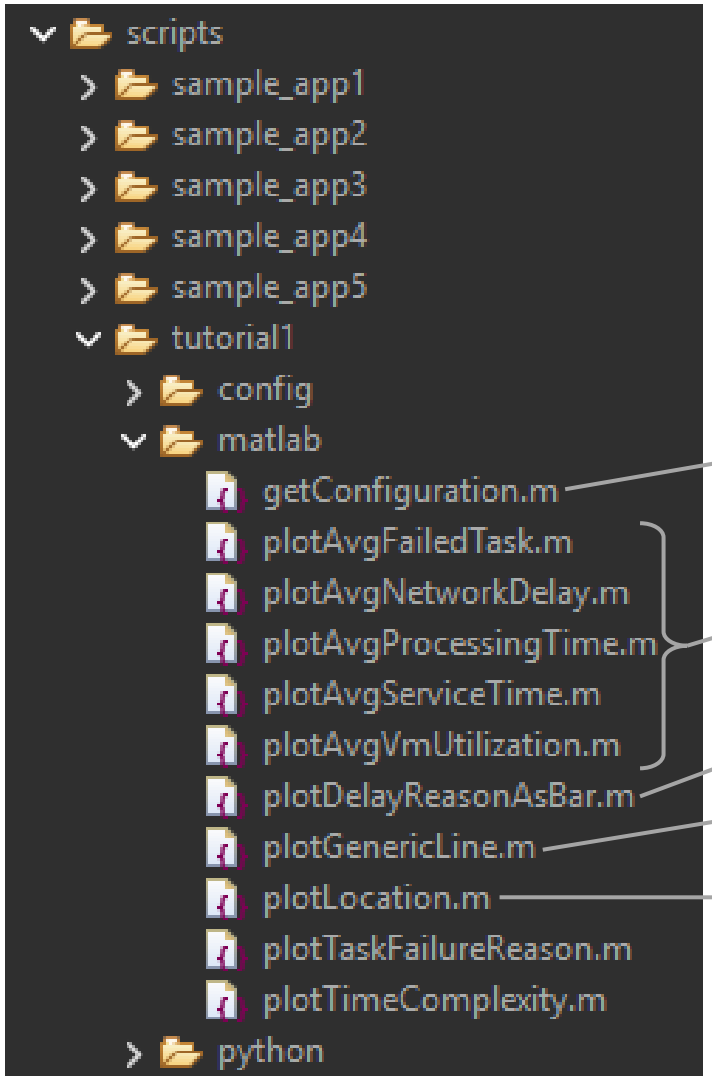
2 Repetitions (Low Reliability)



10 Repetitions (Higher Reliability)



Plotting Simulation Results with MATLAB



★ First of all, configure getConfiguration.m file based on your simulation scenario and preferences!

Configuration file for the other helper scripts

Helper scripts to plot graphics using generic line plotter

Plots bar graphs for delay types

Generic line plotter that generates most of the graphics

Plots a heat-map like graphics to analyze location of the clients

MATLAB - Config File (getConfiguration.m)

```
function [ret_val] = getConfiguration(argType)
    if(argType == 1)
        ret_val = '..\..\..\sim_results\scenario3';
    elseif(argType == 2)
        ret_val = 15; %simulation time (in minutes)
    elseif(argType == 3)
        ret_val = 10; %Number of iterations
    elseif(argType == 4)
        ret_val = 1; %x tick interval for number of mobile devices
    elseif(argType == 5)
        ret_val = {'RANDOM','NETWORK_BASED','UTILIZATION_BASED'};
    elseif(argType == 6)
        ret_val = {'rand','nw','util'};
    elseif(argType == 7)
        ret_val=[6 3 15 15]; %position of figure
    elseif(argType == 8)
        ret_val = [13 12 12]; %font size for x/y label, legend and x/y
    elseif(argType == 9)
        ret_val = 'Number of Clients'; %Common text for x axis
    elseif(argType == 10)
        ret_val = 200; %min number of mobile device
    elseif(argType == 11)
        ret_val = 200; %step size of mobile device count
    elseif(argType == 12)
        ret_val = 2000; %max number of mobile device
    elseif(argType == 17)
        ret_val = 0; %return 1 if you want to add 10^n text at x axis
    elseif(argType == 18)
        ret_val = 0; %return 1 if you want to save figure as pdf
    elseif(argType == 19)
        ret_val = 1; %return 1 if you want to plot errors
    elseif(argType == 20)
        ret_val= 1; %return 1 if graph is plotted colorful
    elseif(argType == 21)
```

Folder path where simulation results are saved

Number of iterations

Scenario names used in the simulations

Corresponding legend texts in figures

Position, size and font size of graphs

Common x axis label

Number of clients used in the simulation

Option to save graph in pdf format

Option to plot graphs with 95% confidence interval error bars

Option to plot graph in color

MATLAB - Sample Plotter Script

5: Calculate percentage value based on all results

3: Label for y axis

6: Position of the legend

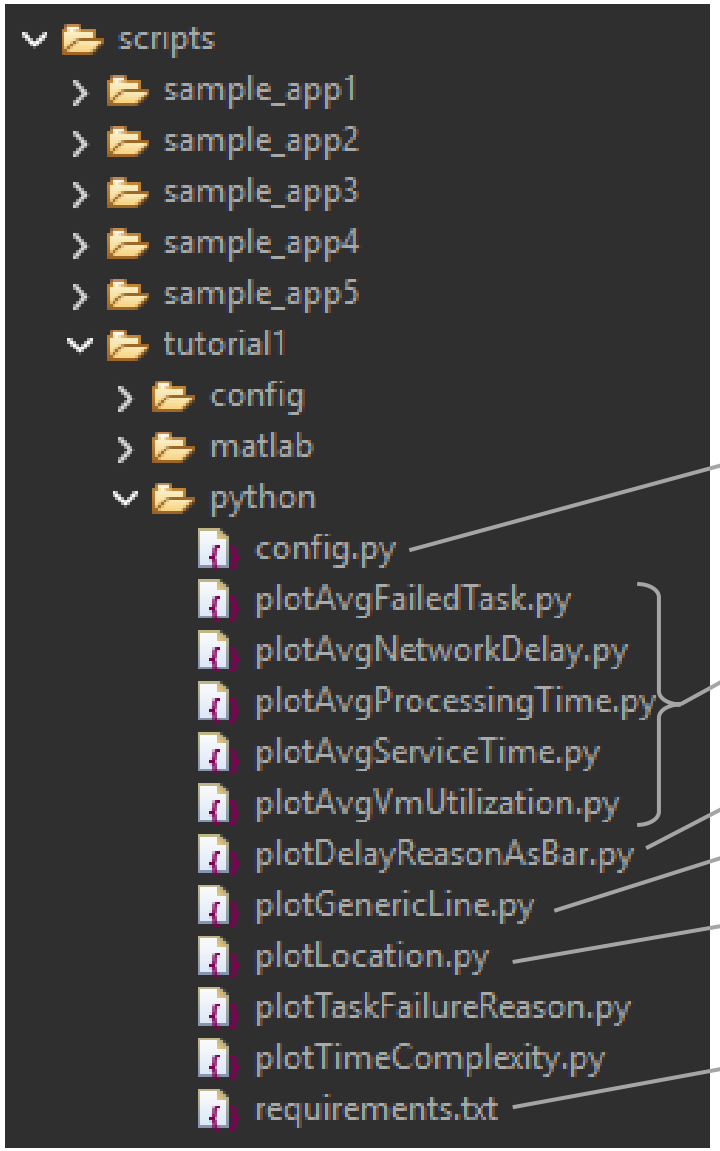
```
function [] = plotAvgFailedTask()

    plotGenericLine(1, 2, 'Failed Tasks (%)', 'ALL_APPS', 'percentage_of_all', 'NorthWest');

    plotGenericLine(1, 2, {'Failed Tasks for';'Augmented Reality App (%)'}, 'AUGMENTED_REALITY', 'percentage_of_all', 'NorthWest');


end
```

1: Line number of the result

2: Column number of the result

4: Result of a specific application

Plotting Simulation Results with Python



★ Python implementation of the MATLAB graph plotter scripts, developed to reproduce the same figures and analyses in a more flexible, open-source environment.

Configuration file for the other helper scripts

Helper scripts to plot graphics using generic line plotter

Plots bar graphs for delay types

Generic line plotter that generates most of the graphics

Plots a heat-map like graphics to analyze location of the clients

List of packages or libraries needed to run Python scripts

Python - Config File (config.py)

```python
config = {
    'folder_path': '../../../sim_results/scenario4',
    'num_iterations': 2,
    'x_tick_interval': 1,
    'scenario_types': ['RANDOM_CAPACITY', 'EQUAL_CAPACITY', 'TRAFFIC_HEURISTIC'],
    'legends': ['rand', 'equal', 'traffic'],
    'figure_position': [6, 3, 15, 15],  # [left, bottom, width, height] in centimeters
    'font_sizes': [13, 12, 12],  # [xy_label, legend, xy_axis_ticks]
    'x_axis_label': 'Number of Vehicles',
    'min_devices': 1000,
    'step_devices': 100,
    'max_devices': 2000,
    'use_scientific_notation_x_axis': False, # For future use
    'save_figure_as_pdf': True,
    'plot_confidence_interval': False,
    'use_color': True,
    # Colors for plots
    'colors': [
        [0.55, 0, 0],       # Color for first line
        [0, 0.15, 0.6],     # Color for second line
        [0, 0.23, 0],       # Color for third line
        [0.6, 0, 0.6],      # Color for fourth line
        [0.08, 0.08, 0.08]  # Color for fifth line
    ],
    # Line styles and markers for colorless plots
    'bw_markers': ['-k*', '-ko', '-ks', '-kv', '-kp'],
    # Line styles and markers for colorful plots
    'color_markers': ['-k*', '-ko', '-ks', '-kv', '-kp']
}
```

- Folder path where simulation results are saved
- Number of iterations
- Scenario names used in the simulations
- Corresponding legend texts in figures
- Position, size and font size of graphs
- Common x axis label
- Number of clients used in the simulation
- Option to save graph in pdf format
- Option to use 95% confidence interval error bars
- Option to plot graph in color
- Line/marker styles for colorful/colorless graphs

Python - Sample Plotter Script

5: Calculate percentage value based on all results

3: Label for y axis

6: Position of the legend

```
from plotGenericLine import plot_generic_line

if __name__ == '__main__':
    plot_generic_line(1, 2, 'Failed Tasks (%)', 'ALL_APPS', 'percentage_of_all', 'upper left')
    plot_generic_line(1, 2, 'Failed Tasks for\nAugmented Reality App (%)', 'AUGMENTED_REALITY', 'percentage_of_all', 'upper left')
```

1: Line number of the result

2: Column number of the result

4: Result of a specific application

Live Demo Session

Case Study 1 – VM Scheduling

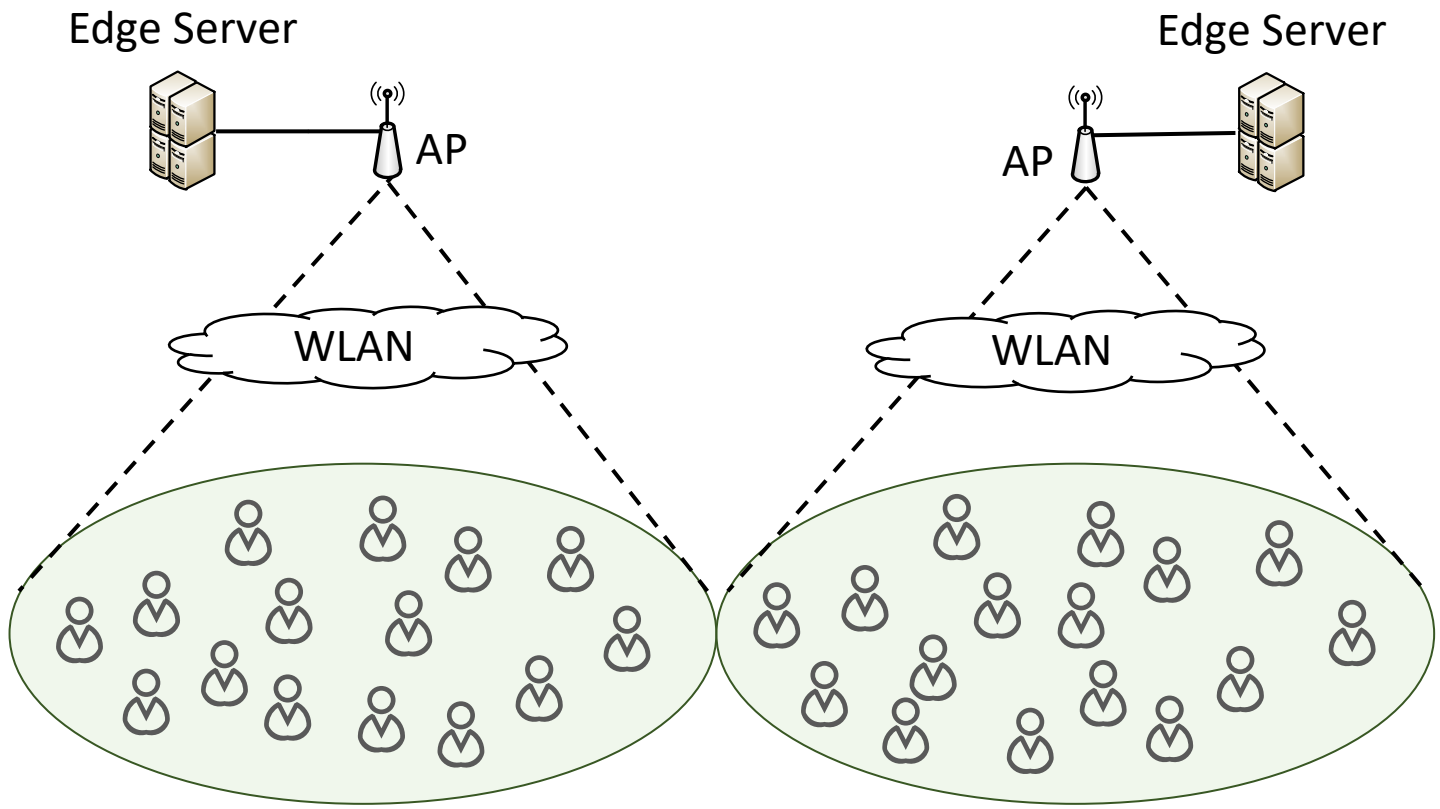
Performance Evaluation of Different VM Allocation Policies

Find the source code below:

https://github.com/CagataySonmez/EdgeCloudSim/tree/master/src/edu/boun/edgecloudsim/applications/tutorial1

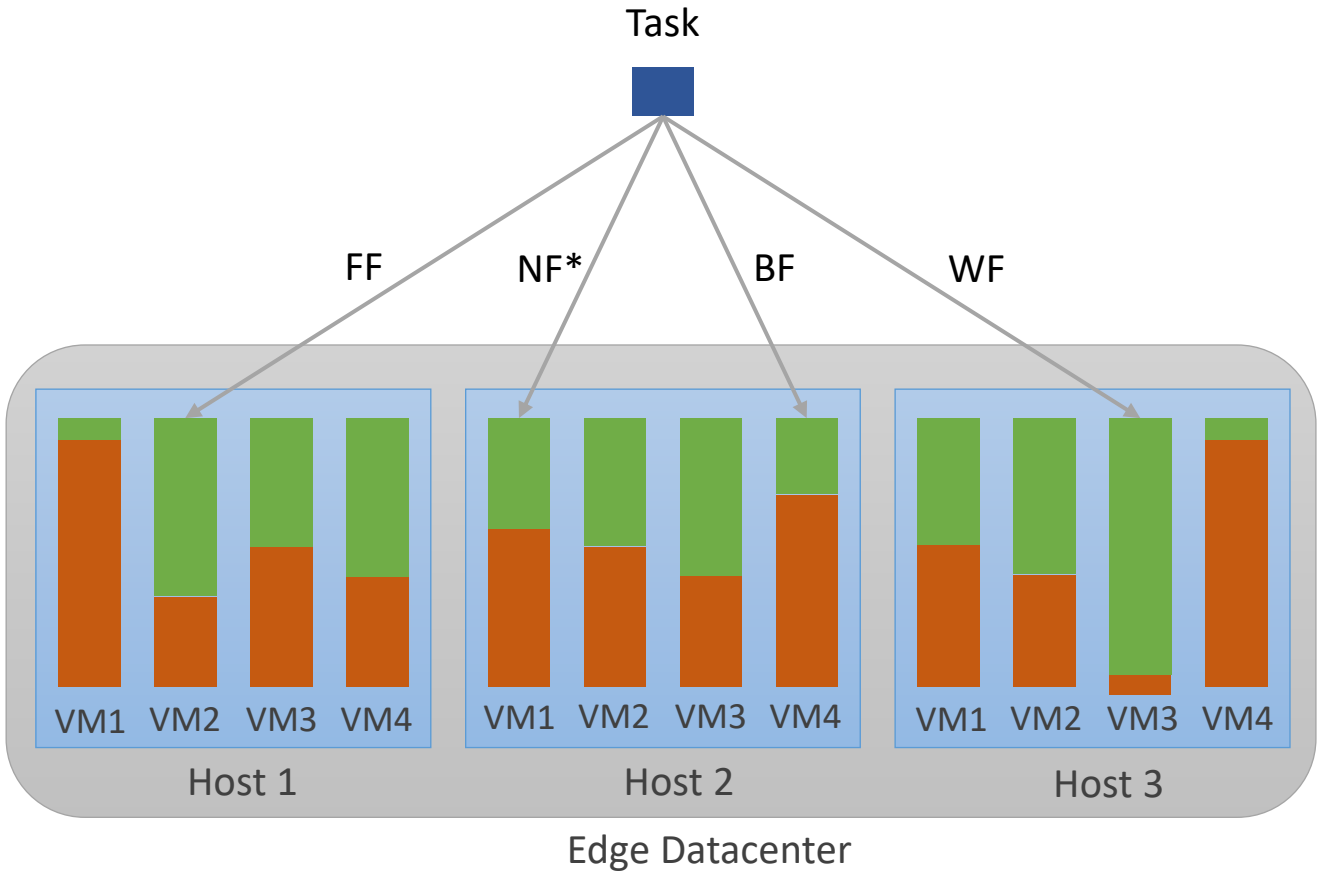
Simulation Scenario

- Mobile devices can only offload task to the edge servers connected to the serving access point
- Edge servers operates a variable number of VMs
- This scenario compares different VM provisioning algorithms



Competitor VM Provisioning Algorithms

- **Random (RND)**: A random VM is selected
- **First-Fit (FF)**: First available VM is selected
- **Next-Fit (NF)**: The hosts are visited in order and the first suitable VM is selected.
- **Best-Fit (BF)**: The VM with the highest CPU utilization is selected
- **Worst-Fit (WF)**: The VM with the least CPU utilization is selected



* Assuming the previous selection of the NF algorithm was one of the VM in Host1.

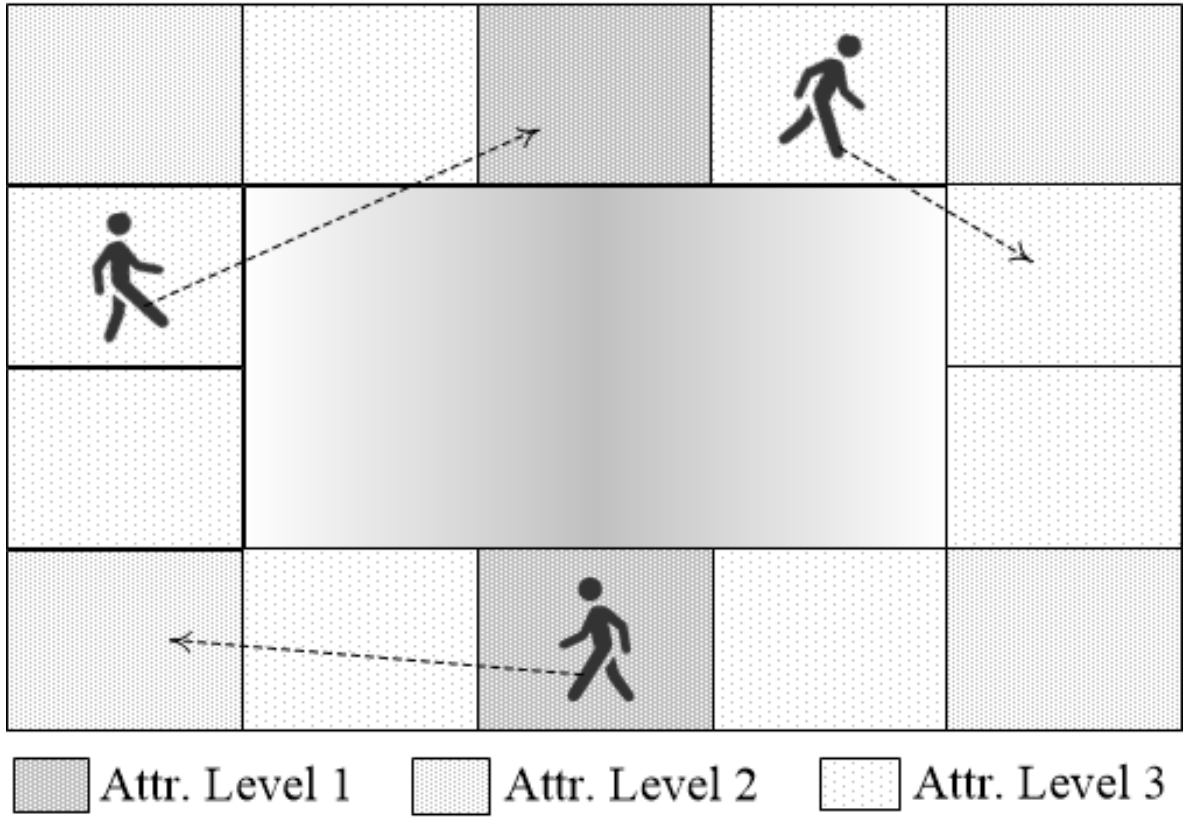
Applications Used in This Simulation

Parameter	Aug. Reality	Health	Infotainment
Usage Percentage (%)	30	20	50
Task Interarrival (sec)	2	3	7
Active/Idle Period (sec)	40/20	45/90	30/45
VM Utilization on Edge Server (%)	6	2	3.6
Task Length (GI)	15	3	9
Upload/Download Data (KB)	1500/50	50/1250	250/1000

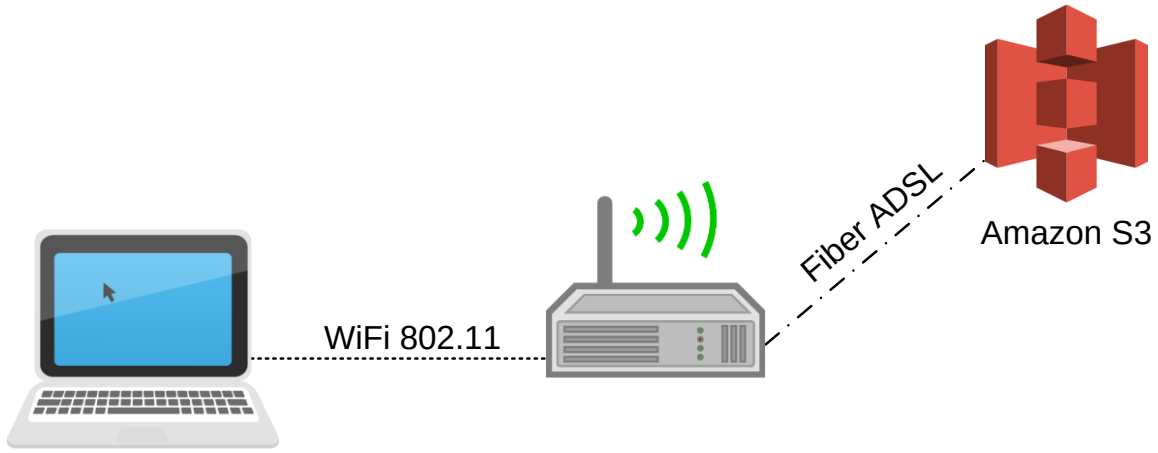
★ Clients are utilizing an application that generates task according to a Poisson process.

Nomadic Mobility Model

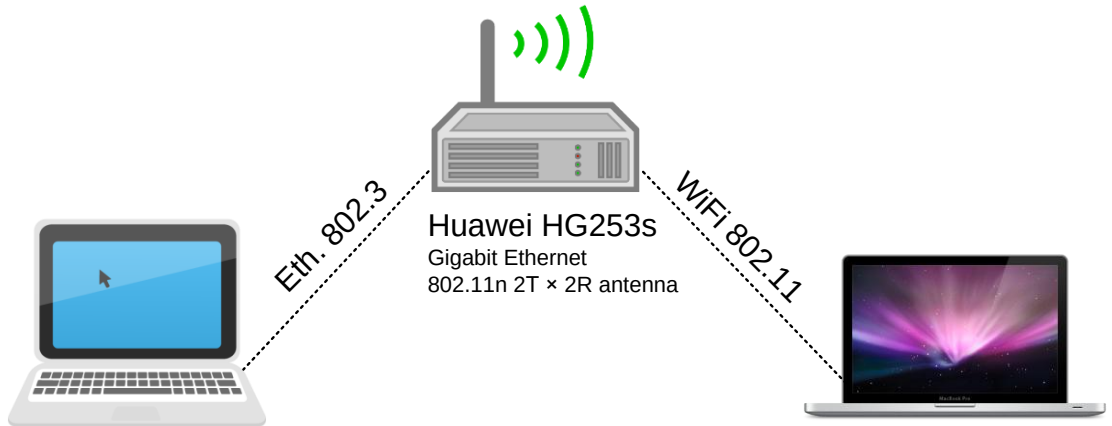
- There is no real time walking pattern, user location is updated at random time intervals
- Probability of selecting a new location is same for all locations
- We use variable locations with different attractiveness levels in simulations
- The attractiveness level determines how much time the user will spend (dwell time) in the corresponding place



Empirical WAN/WLAN Model



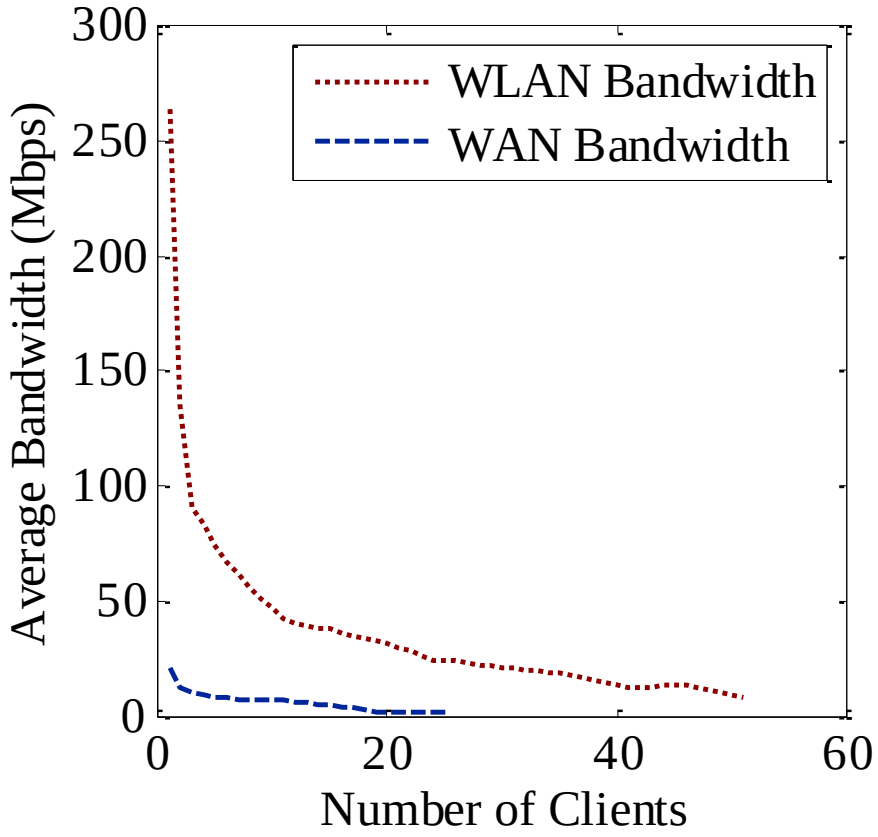
WAN Bandwidth Analysis Setup



Lenovo T550
Intel Core i7-5600U (2.6 Ghz)
Gigabit Ethernet
16 GB RAM

MacBook Pro Late 2011
Intel Core i7-2675QM (2.2 Ghz)
802.11n WiFi
8 GB RAM

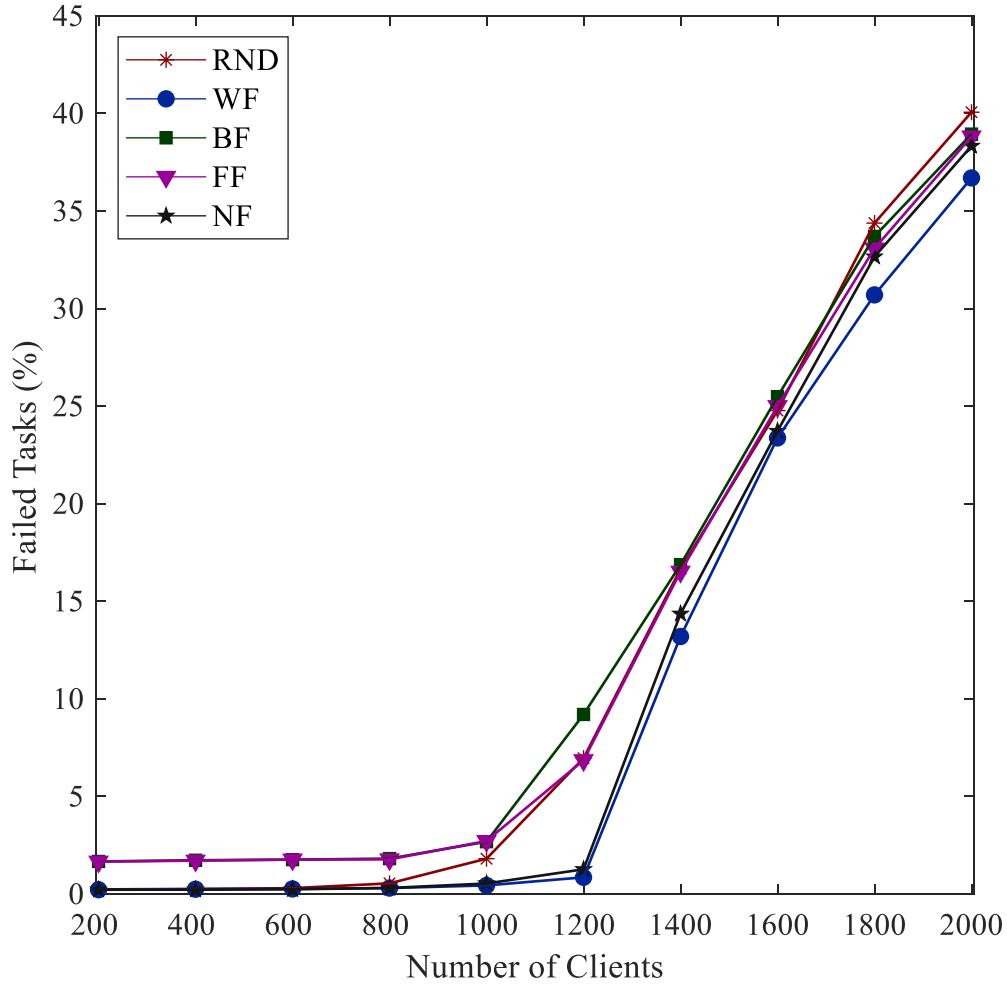
WLAN Bandwidth Analysis Setup

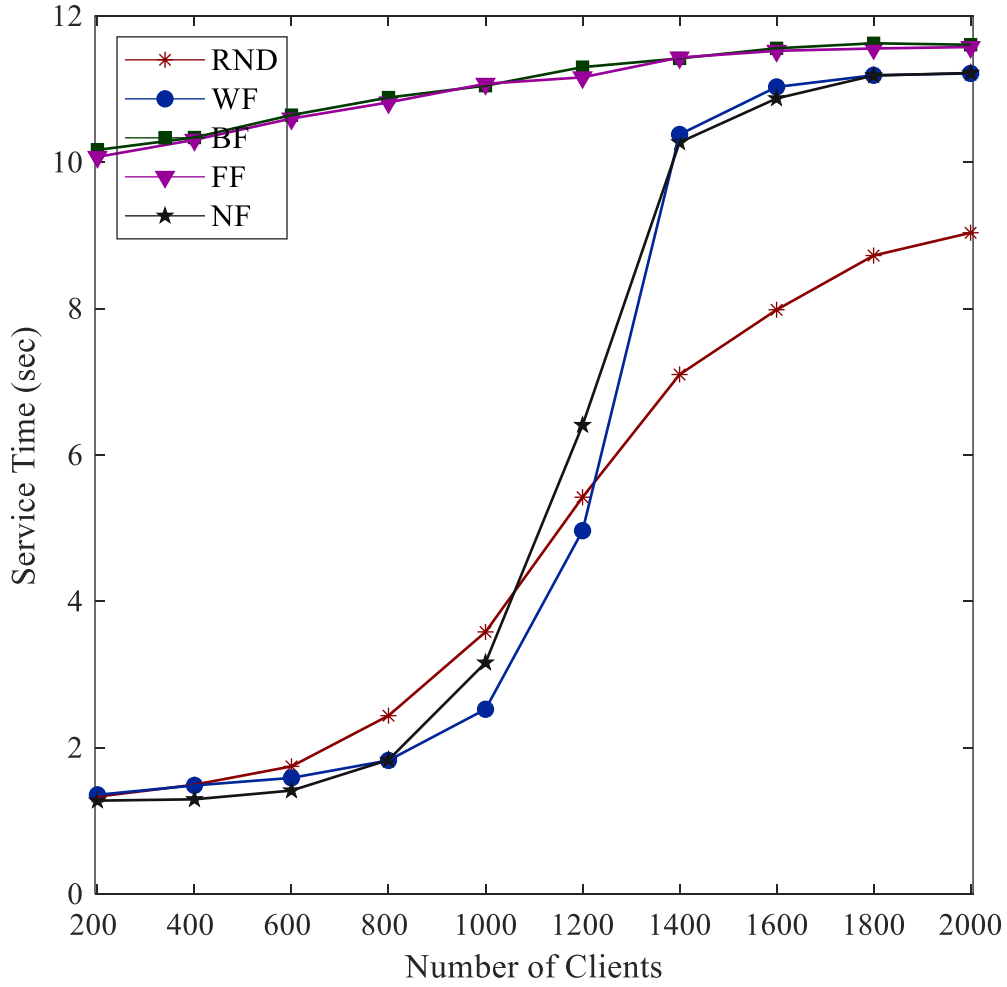


Simulation Parameters

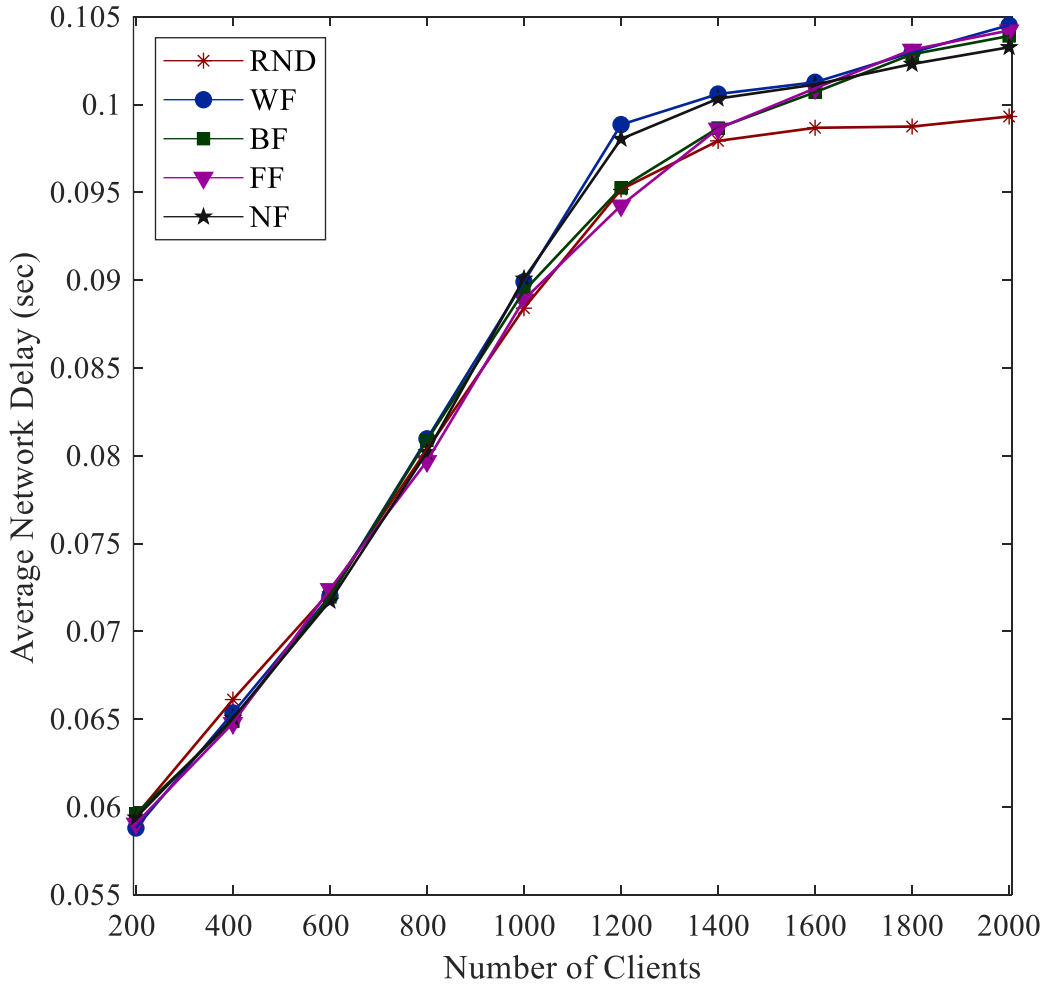
Parameter	Value
Simulation Time/Warm-up Period	30/5 minutes
Number of Repetitions	10
WLAN Delay Model	Empirical
MAN Delay	Fixed (10 ms)
Number of VMs per Edge Host	8
Number of Cores per Edge VM	2
VM Processor Speed per Edge CPU	10 GIPS
Mobility Model	Nomadic Mobility
Number of Locations for Type 1/2/3 places	2/4/8
Mean waiting (dwell) time in Type 1/2/3 places	10/10/10 minutes

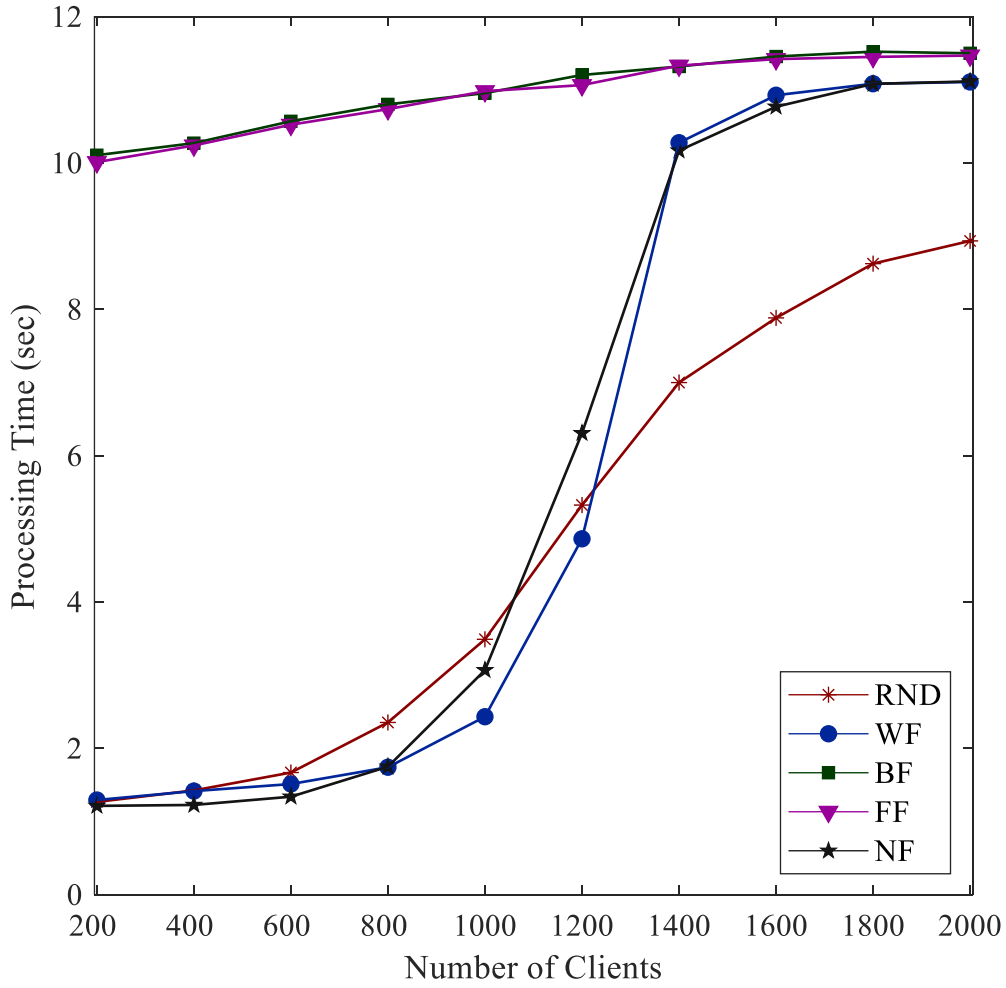
Important Simulation Results





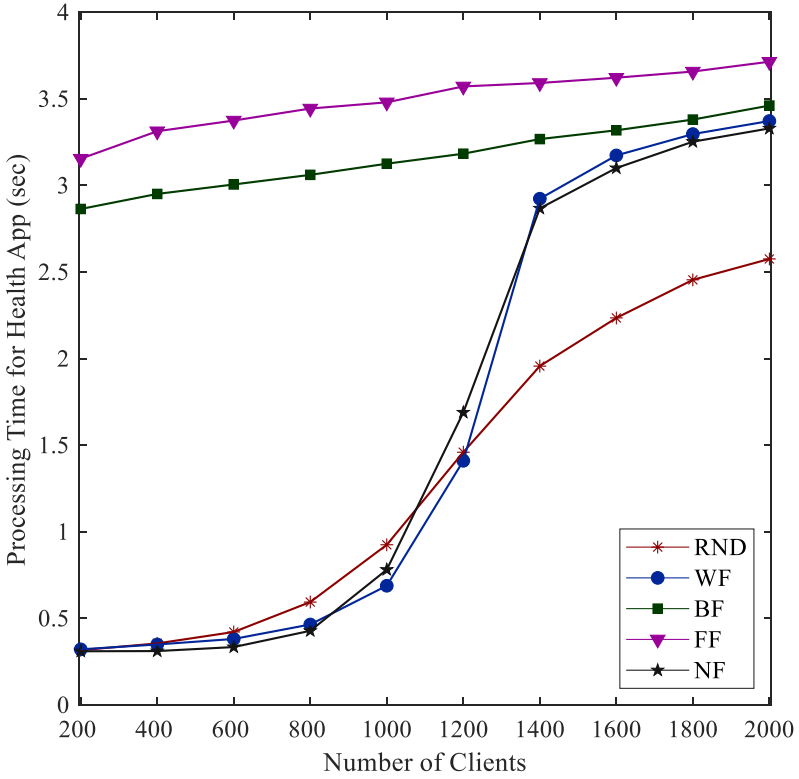
Service Time Analysis



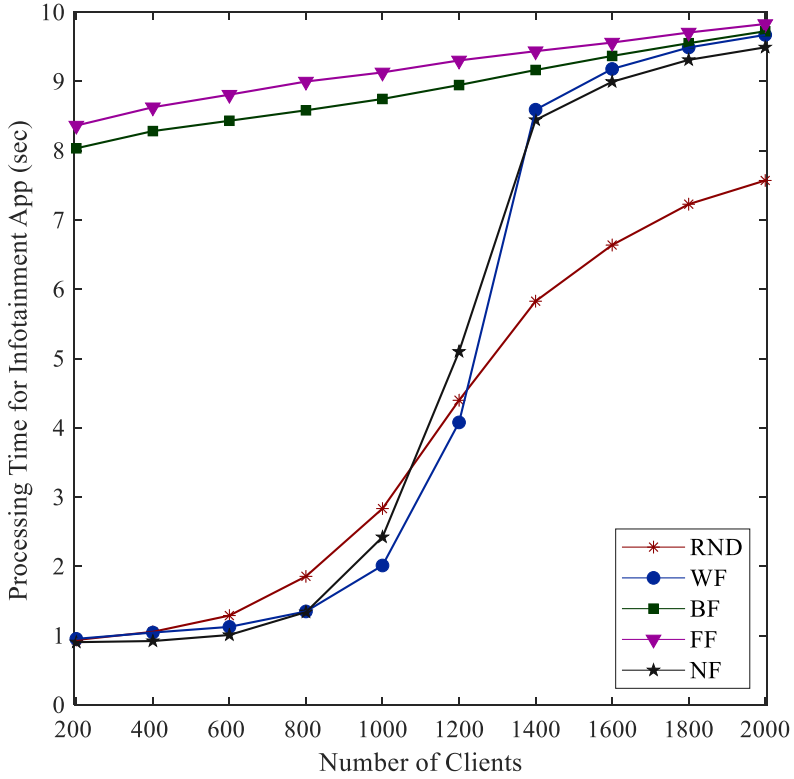


Processing time dominates the average service time!

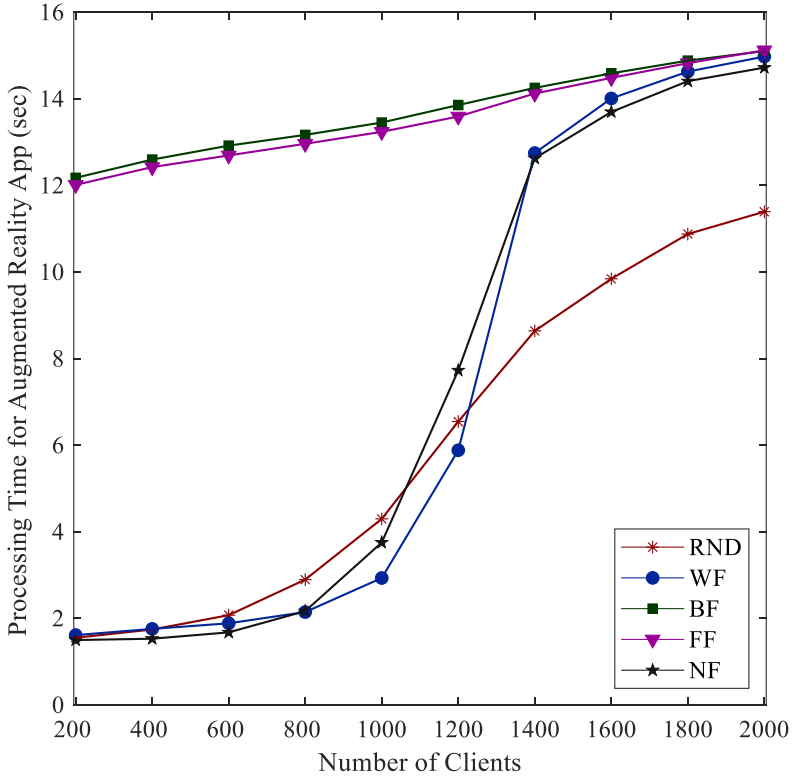
Processing Time per Task Types



Task Size: 3 GI



Task Size: 9 GI



Task Size: 15 GI

Case Study 2 – What to Offload

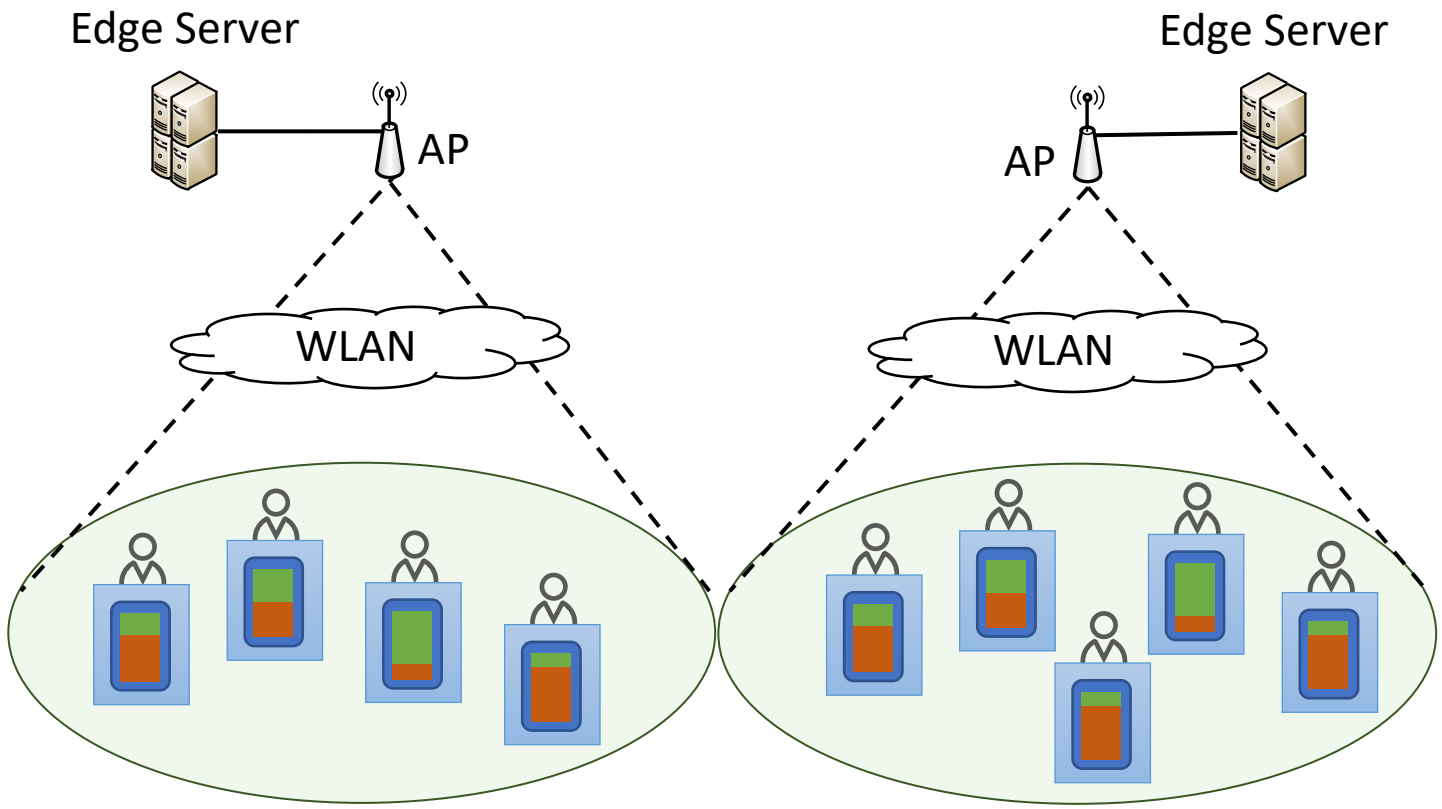
Performance Evaluation of Different Approaches that Decide Granularity of Task Offloading

Find the source code below:

https://github.com/CagataySonmez/EdgeCloudSim/tree/master/src/edu/boun/edgecloudsim/applications/tutorial2

Simulation Scenario

- Mobile devices can operate tasks locally or offload them to the edge servers connected to the serving access point
- Edge servers operates a variable number of VMs
- Worst-fit VM provisioning (least loaded first) algorithm is used



Competitor Task Offloading Algorithms

- **Random**: A random VM is selected
- **Mobile Device Utilization Heuristic**

> If average mobile device CPU utilization < 75 execute task locally
> Otherwise, offload to edge server

- **Edge Utilization Heuristic**

> If average edge server CPU utilization < 90 offload task to edge server
> Otherwise, execute task locally

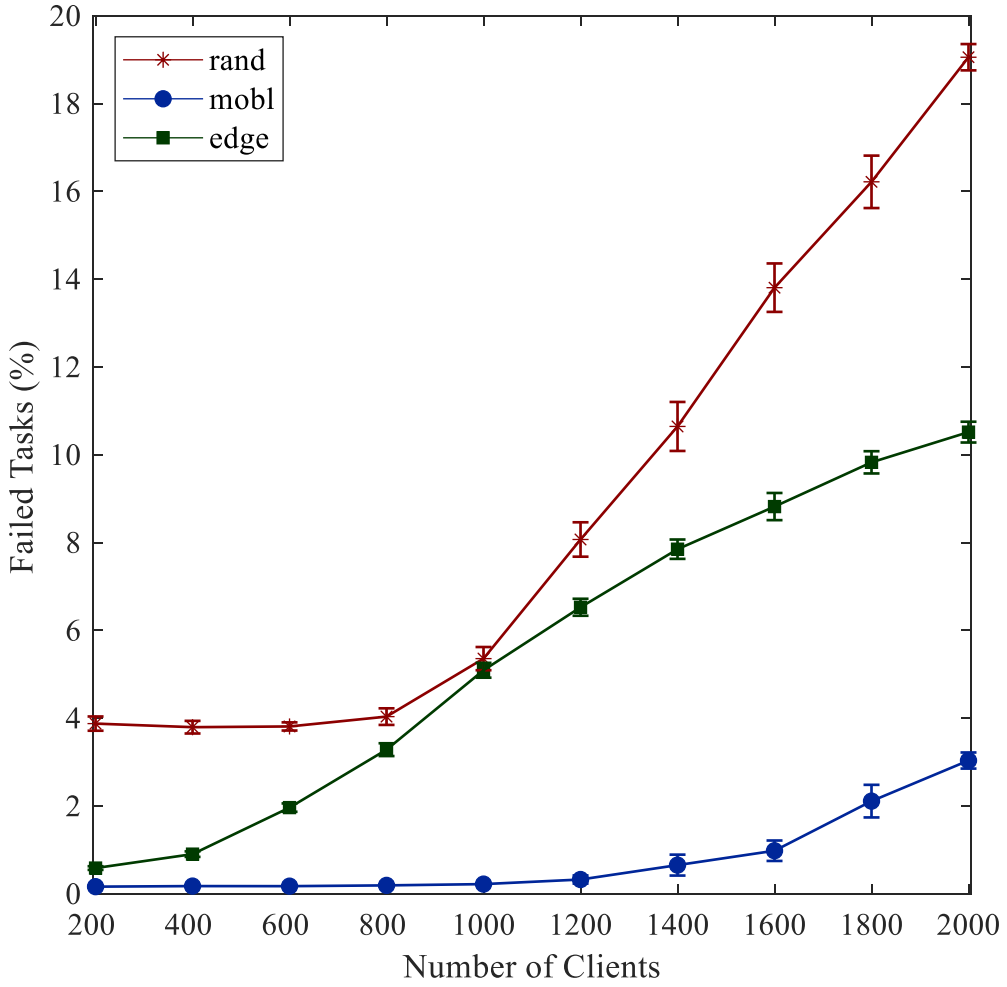
Applications Used in This Simulation

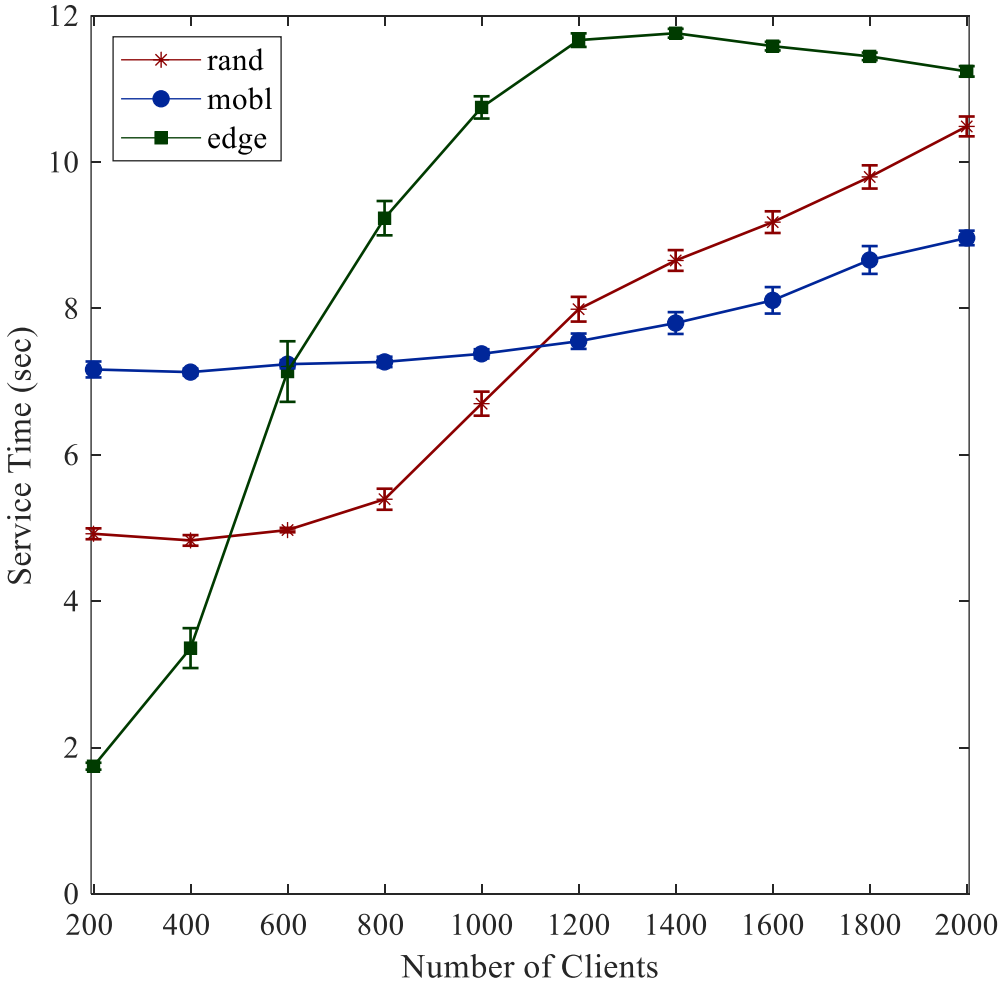
Parameter	Aug. Reality	Health	Infotainment
Usage Percentage (%)	30	30	40
Task Interarrival (sec)	2	3	5
Active/Idle Period (sec)	40/20	60/30	30/30
VM Utilization on Edge/Client (%)	5/20	2/8	4/16
Task Length (GI)	20	8	16
Upload/Download Data (KB)	3000/1000	900/500	2000/4000

Simulation Parameters

Parameter	Value
Simulation Time/Warm-up Period	15/1 minutes
Number of Repetitions	10
WLAN Delay Model	Empirical
MAN Delay	Fixed (5 ms)
Number of VMs per Edge/Mobile Host	8/1
Number of Cores per Edge/Mobile VM	2/1
VM Processor Speed per Edge/Mobile CPU	10/4 GIPS
Mobility Model	Nomadic Mobility
Number of locations for Type 1/2/3 places	2/4/8
Mean waiting (dwell) time in Type 1/2/3 places	10/6.6/3.3 minutes

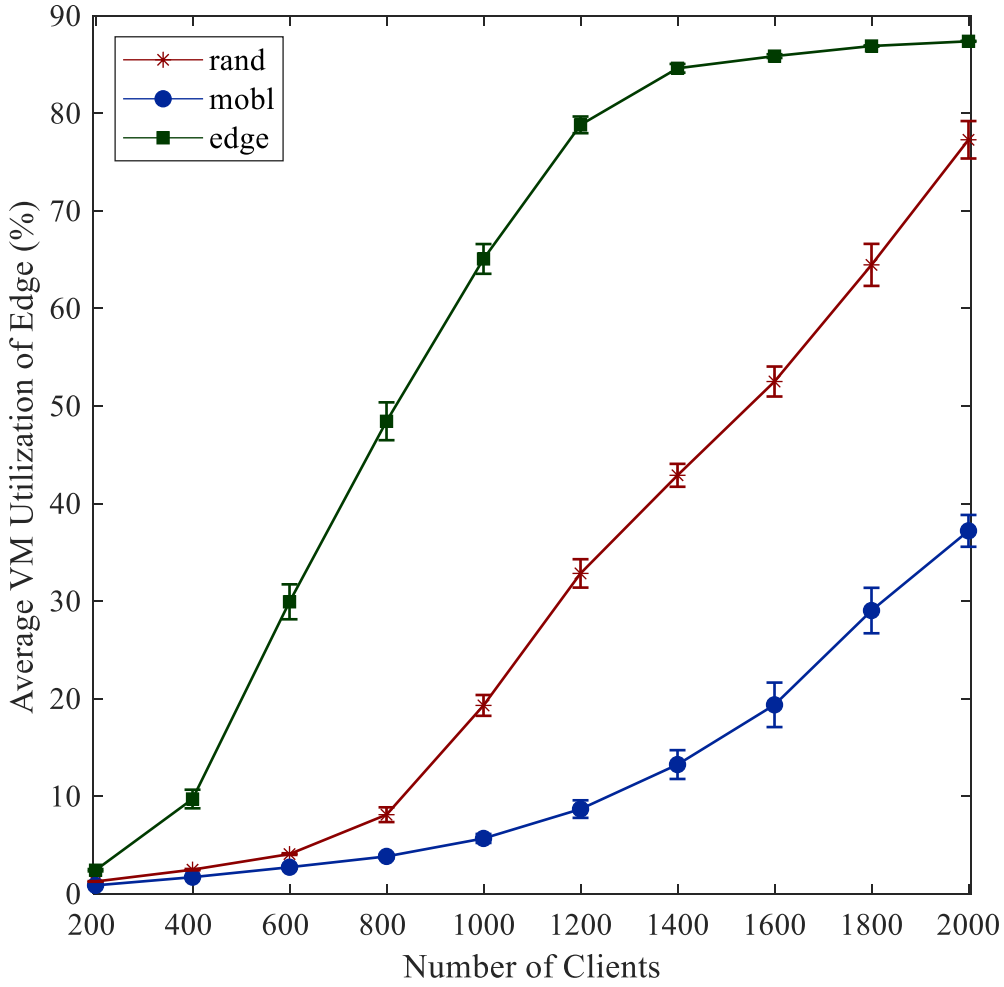
Important Simulation Results

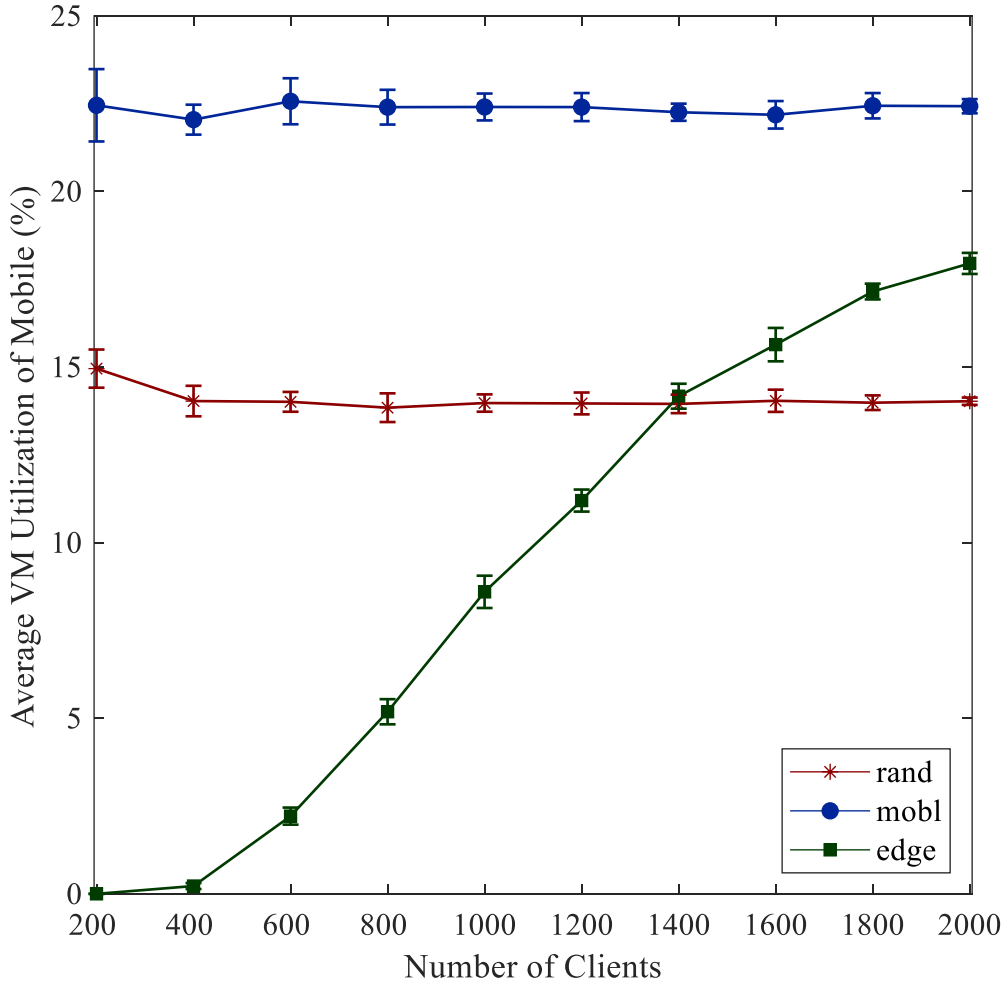




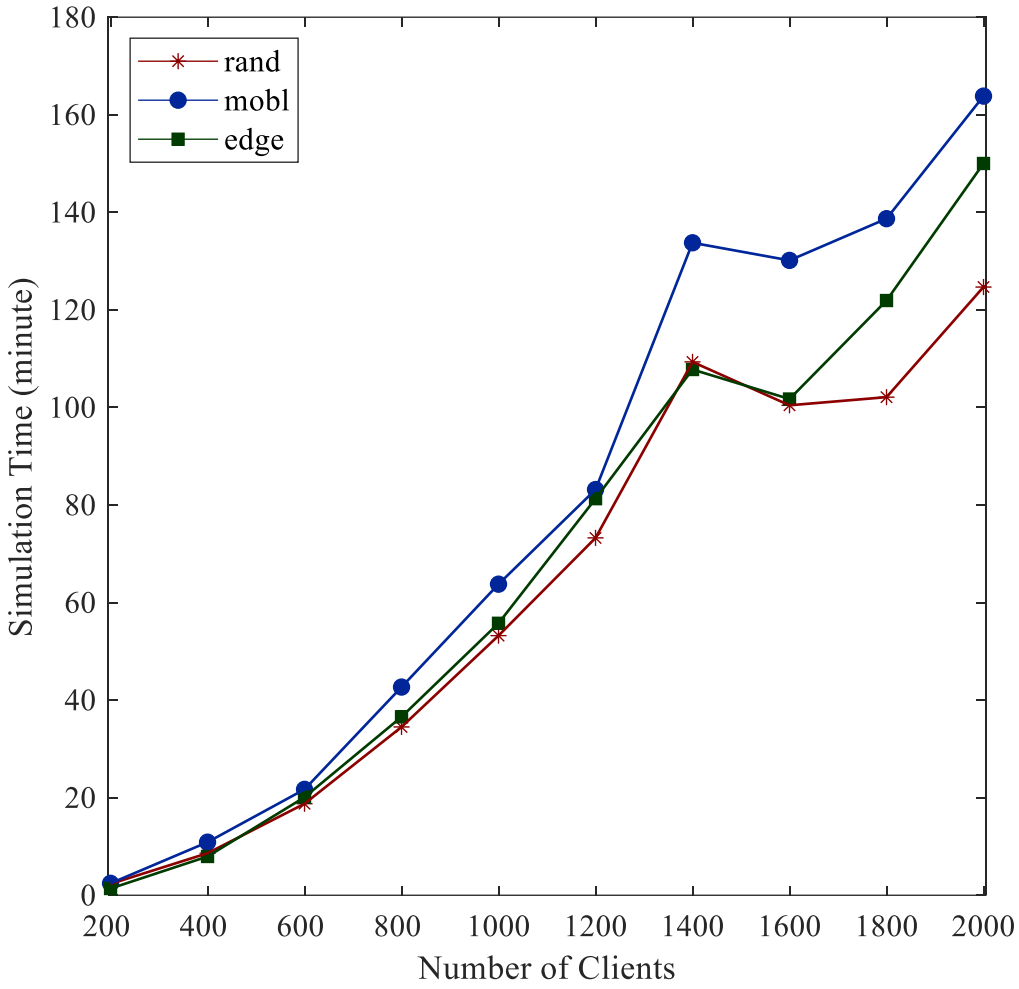
Graph is plotted with 95% confidence interval error bars.

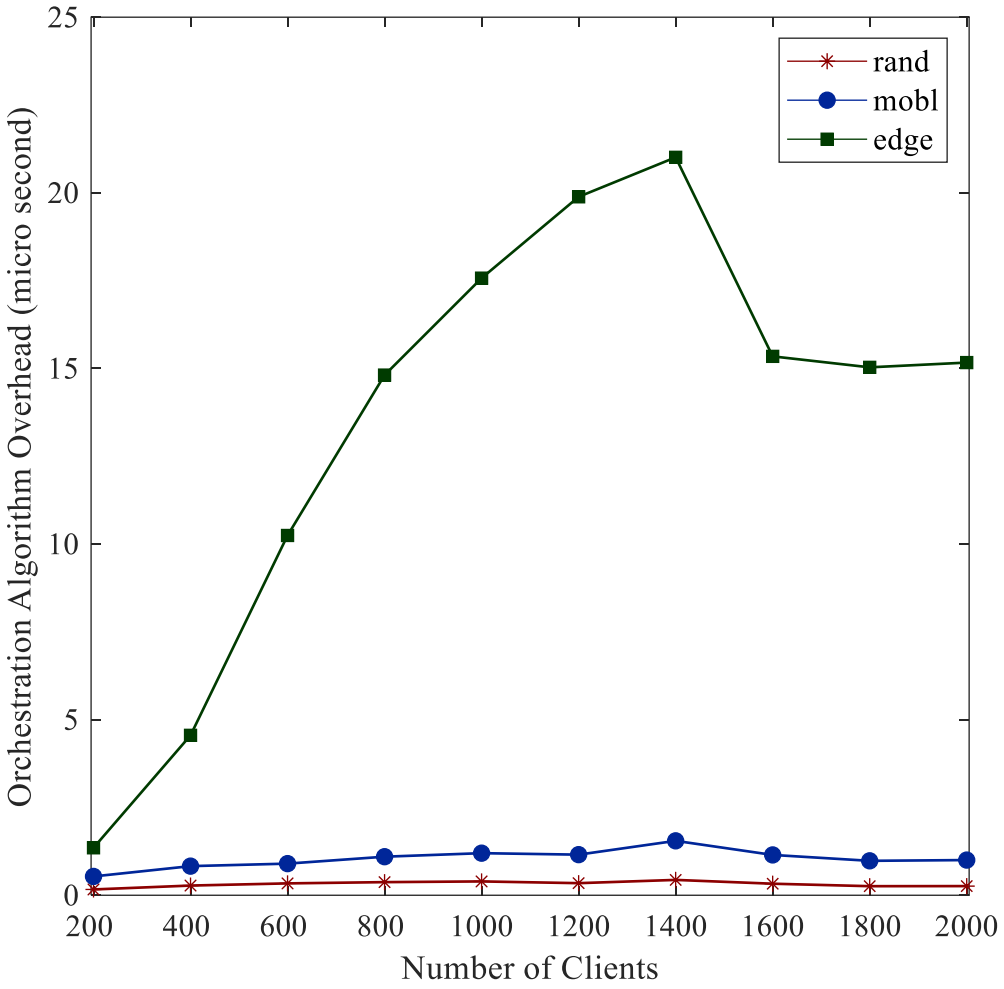
VM Utilization Analysis





Complexity Analysis





These results are highly dependent to host machine load while running the simulation!

Case Study 3 – Where to Offload

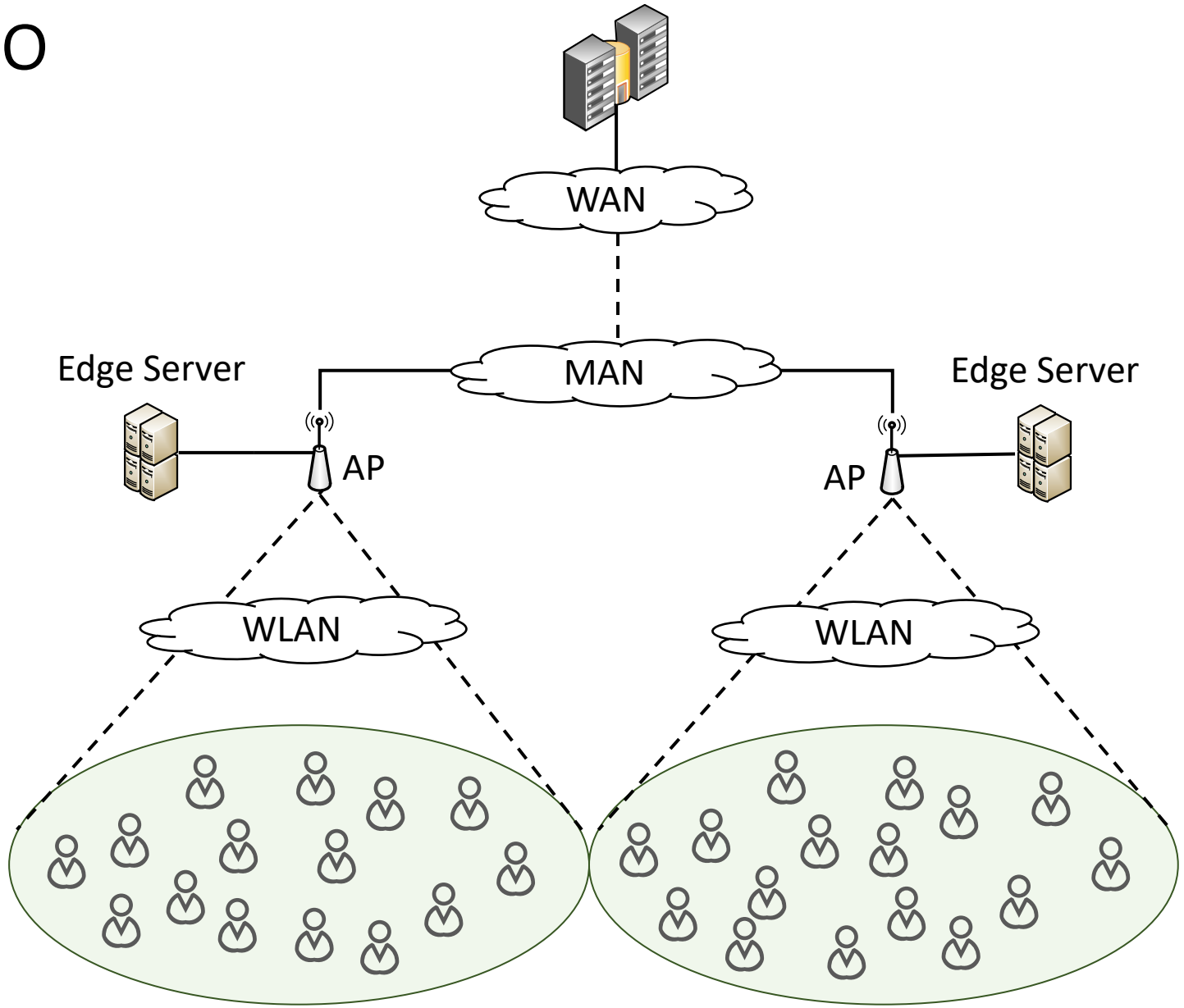
Performance Evaluation of Different Workload Orchestration Policies

Find the source code below:

https://github.com/CagataySonmez/EdgeCloudSim/tree/master/src/edu/boun/edgecloudsim/applications/tutorial3

Simulation Scenario

- Mobile can offload tasks to edge or cloud servers
- Worst-fit VM provisioning (least loaded first) algorithm is used
- If the task is sent to another edge server outside the connected network, it is transmitted via MAN
- WLAN and WAN delays are modeled independently, so the WLAN is not affected if task is sent to remote server



Competitor Edge Orchestration Algorithms

- **Random**: A random server is selected to offload task
- **Edge Server Utilization Heuristic**

> If average edge servers CPU utilization > 75, offload task to cloud server
> Otherwise, offload task to edge servers

- **Network Utilization Heuristic**

> If WAN bandwidth > 5 Mbps, offload task to cloud server
> Otherwise, offload task to edge servers

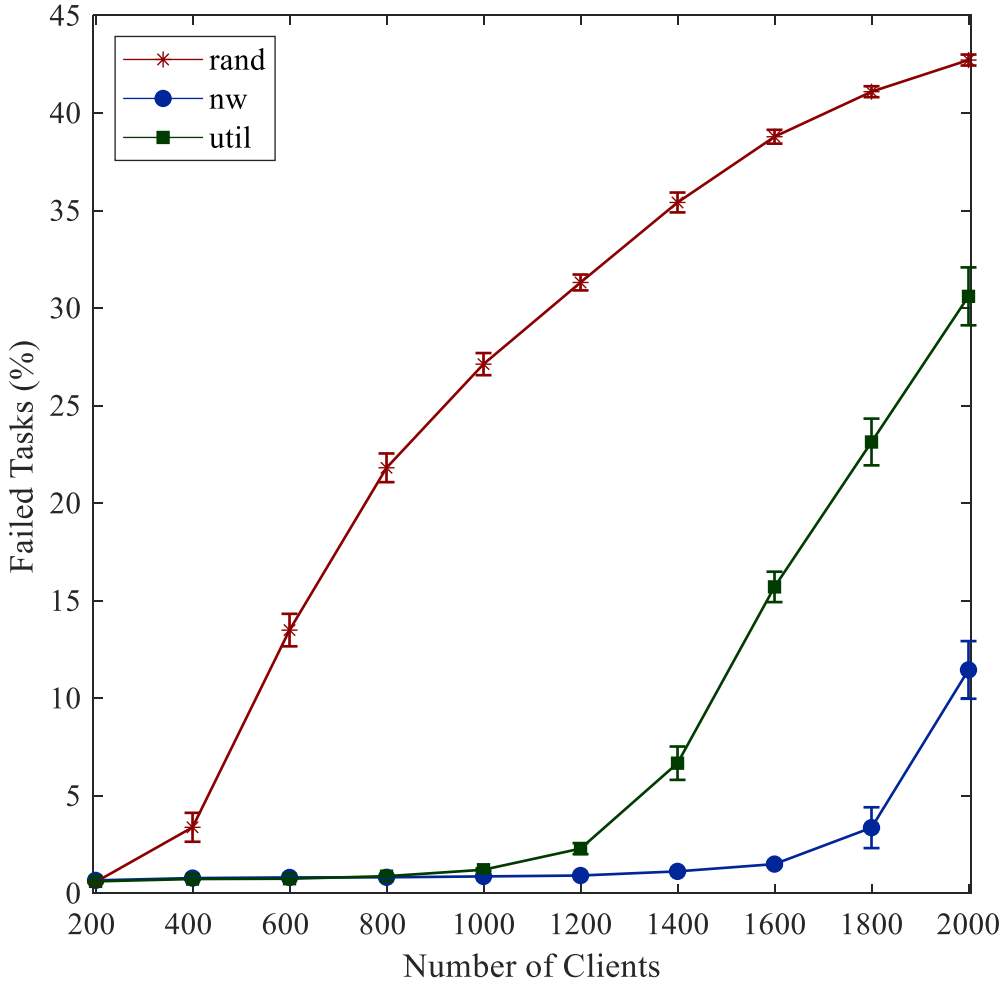
Applications Used in This Simulation

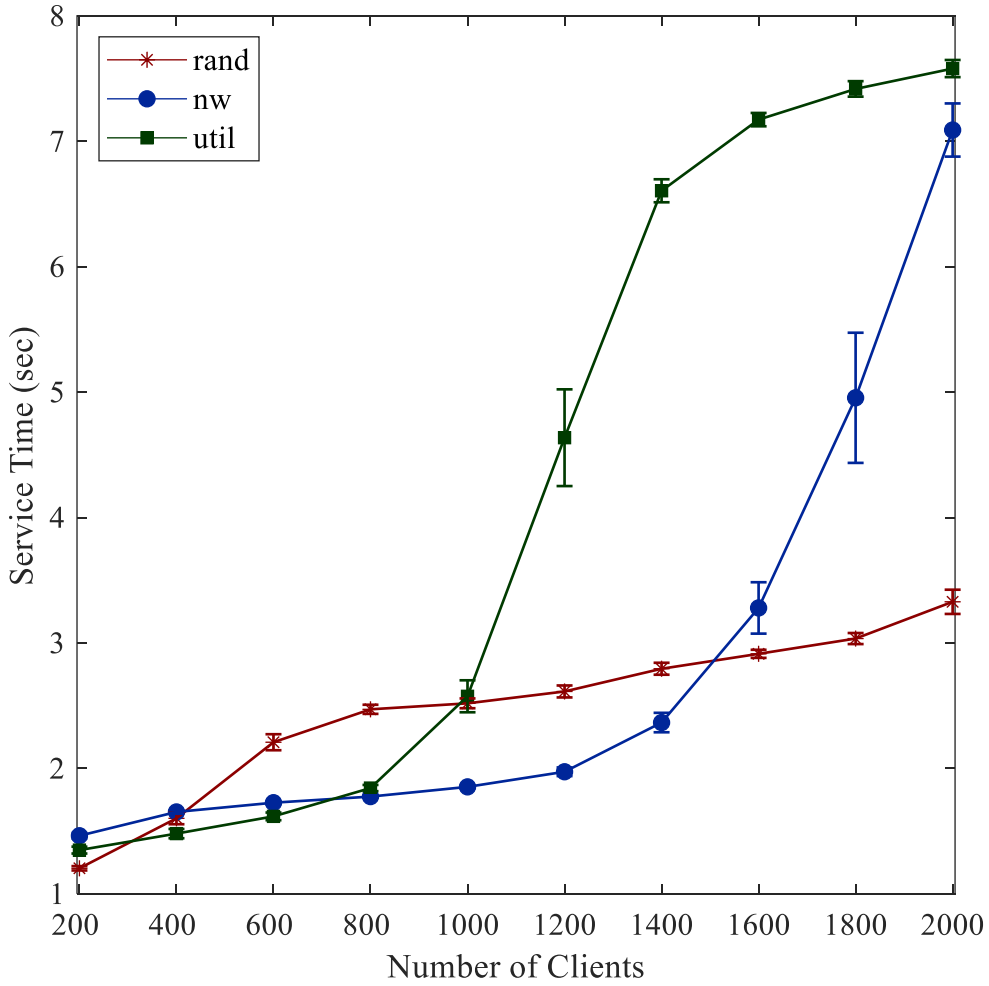
Parameter	Aug. Reality	Health	Infotainment
Usage Percentage (%)	30	20	50
Task Interarrival (sec)	2	3	7
Active/Idle Period (sec)	40/20	45/90	30/45
VM Utilization on Edge/Cloud (%)	6/0.6	2/0.2	3.6/0.36
Task Length (GI)	15	3	9
Upload/Download Data (KB)	1500/50	50/1250	250/1000

Simulation Parameters

Parameter	Value
Simulation Time/Warm-up Period	30/5 minutes
Number of Repetitions	10
WAN/WLAN Delay Model	Empirical
MAN Delay	Fixed (5 ms)
Number of VMs per Edge/Cloud Host	8/4
Number of Cores per Edge/Cloud VM	2/4
VM Processor Speed per Edge/Cloud CPU	10/100 GIPS
Mobility Model	Nomadic Mobility
Number of Locations for Type 1/2/3 Places	2/4/8
Mean waiting time in Type 1/2/3 Places	8/5/2 minutes

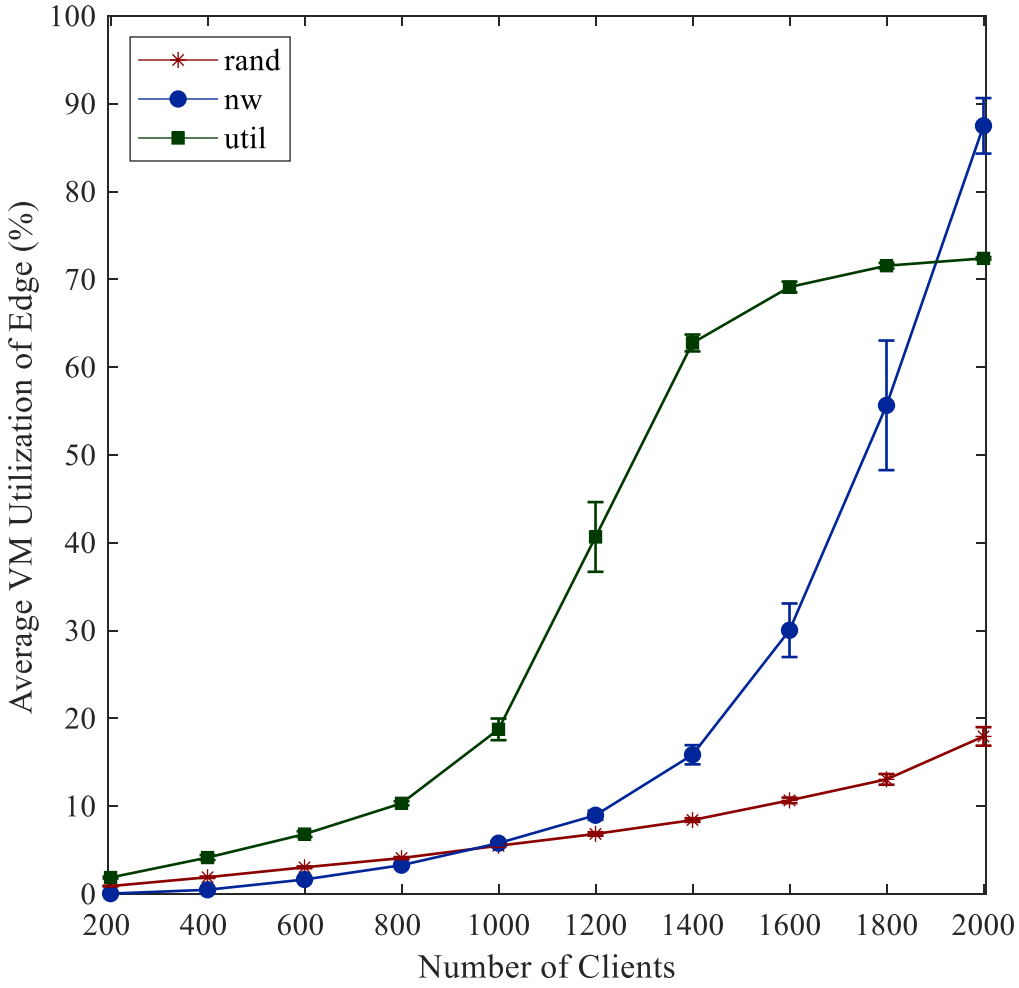
Important Simulation Results

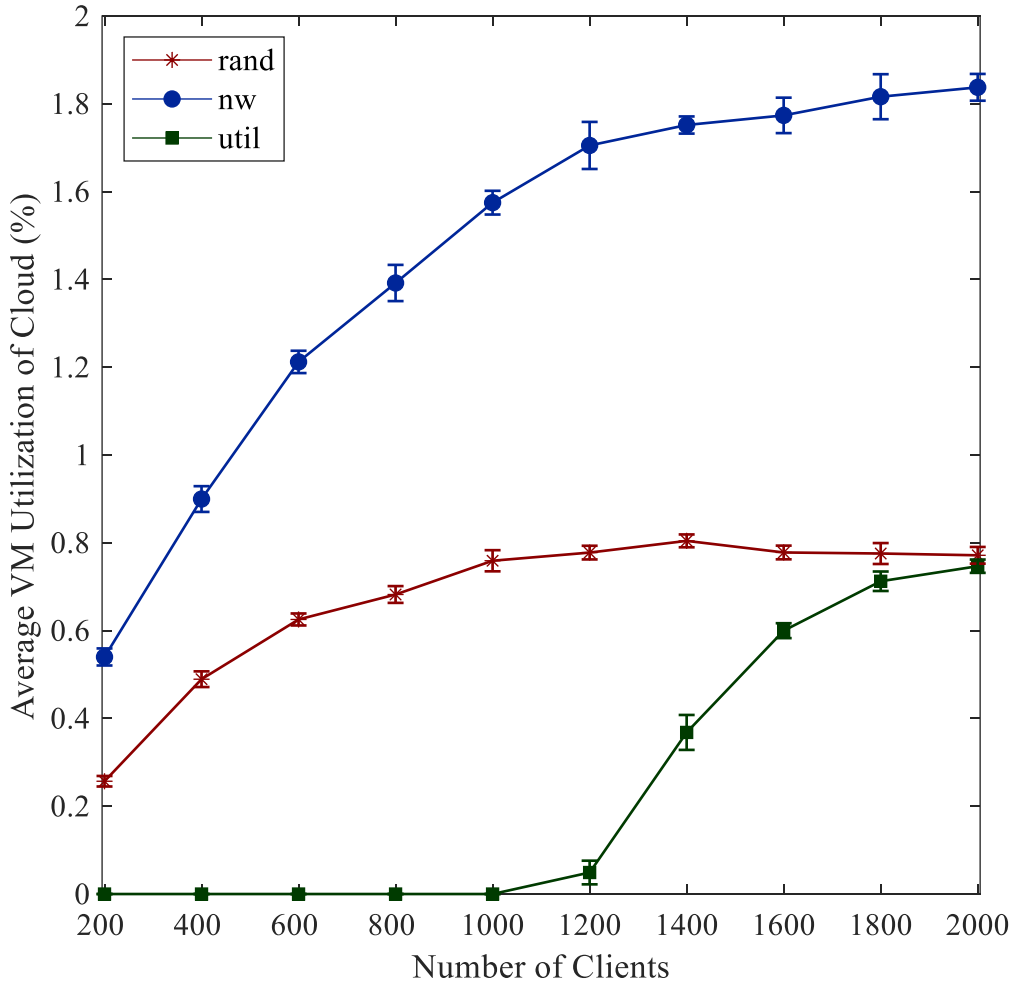




- Graph is plotted with 95% confidence interval error bars.

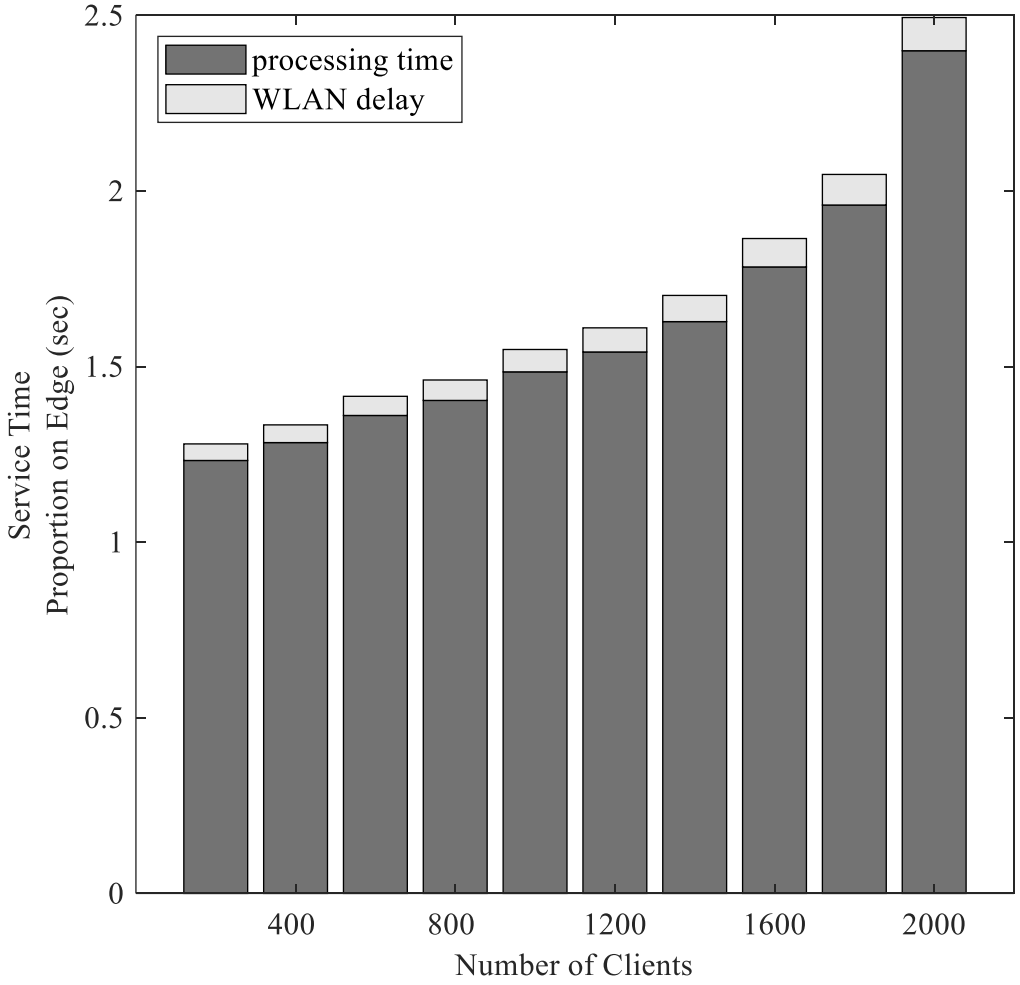
VM Utilization Analysis

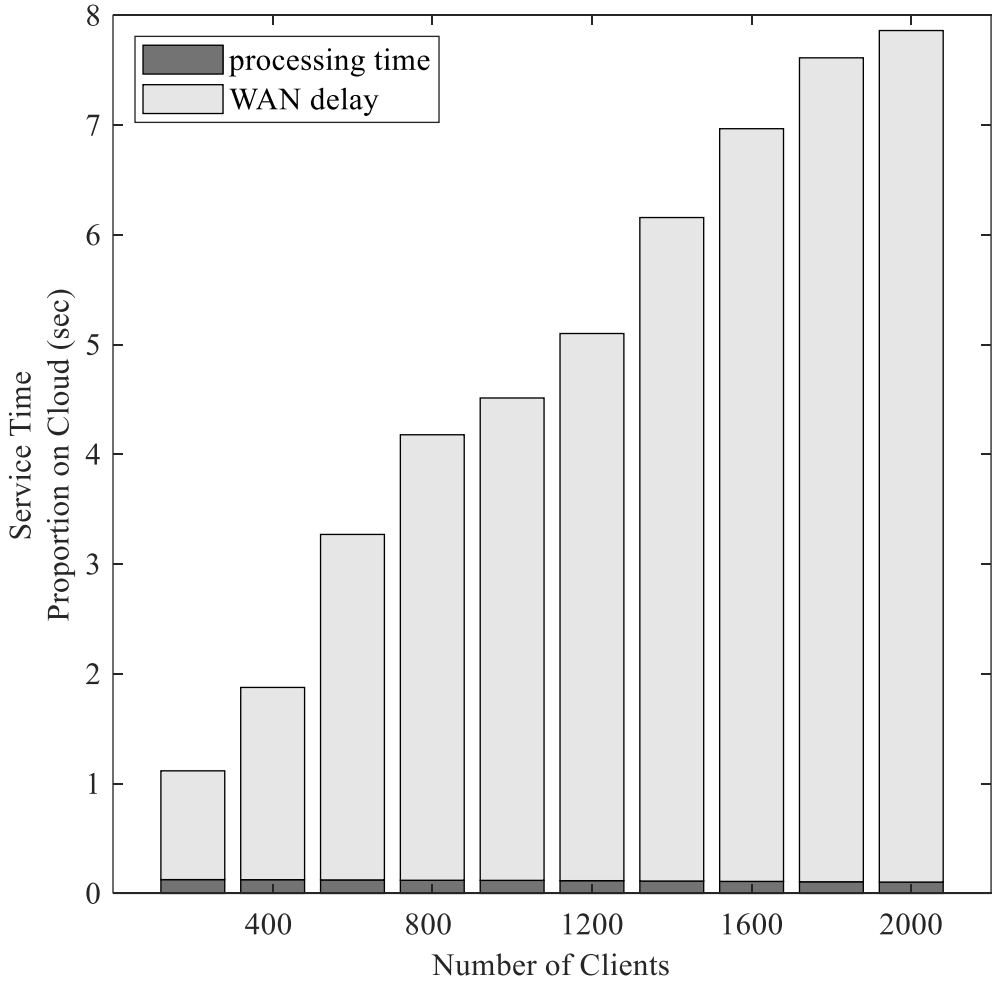




The cloud is mostly considered to be an endless pool of resources.

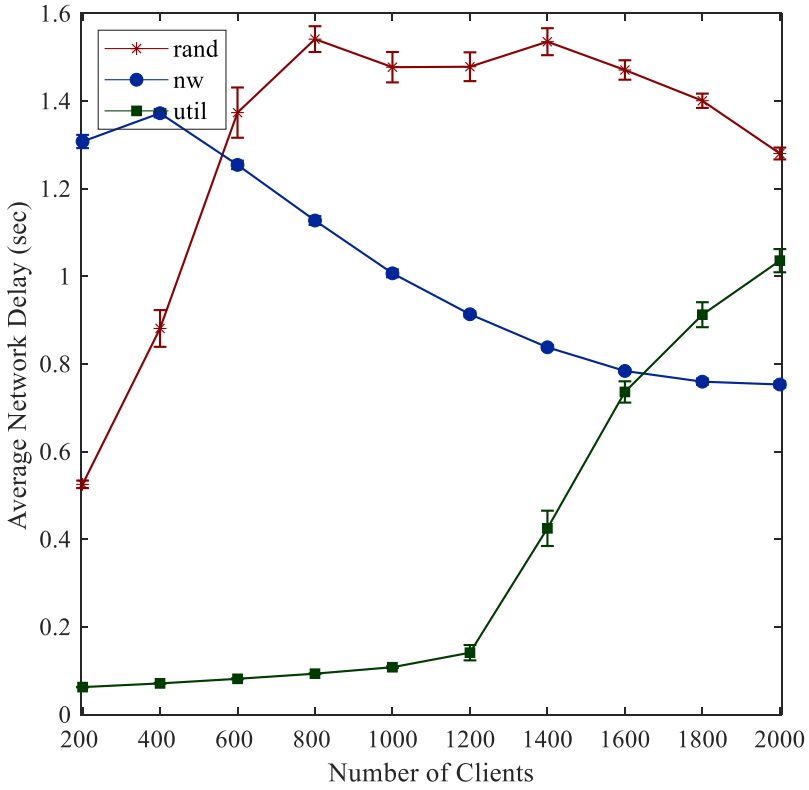
Service Time Analysis

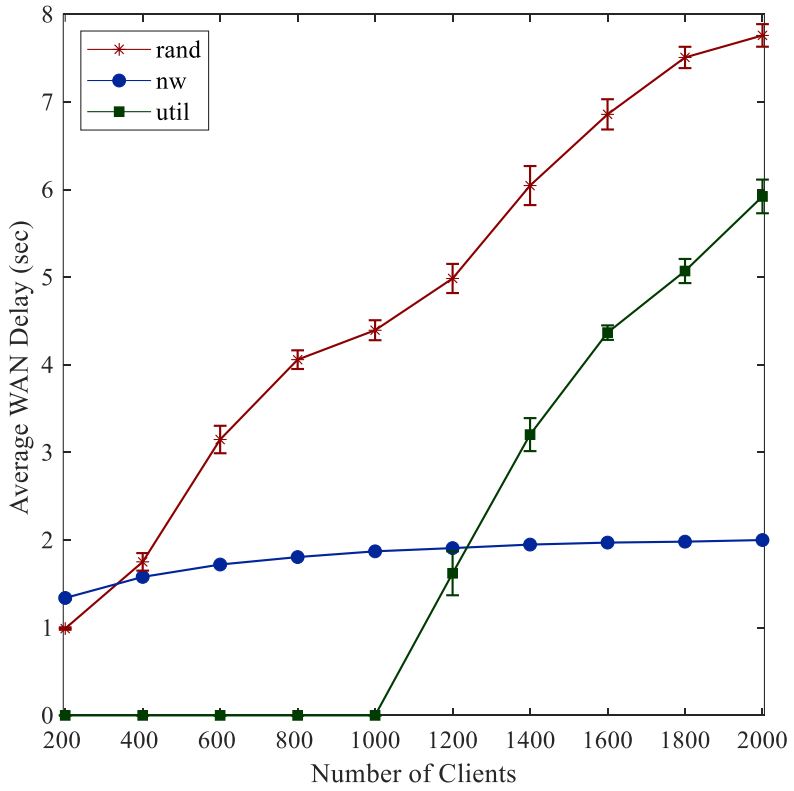


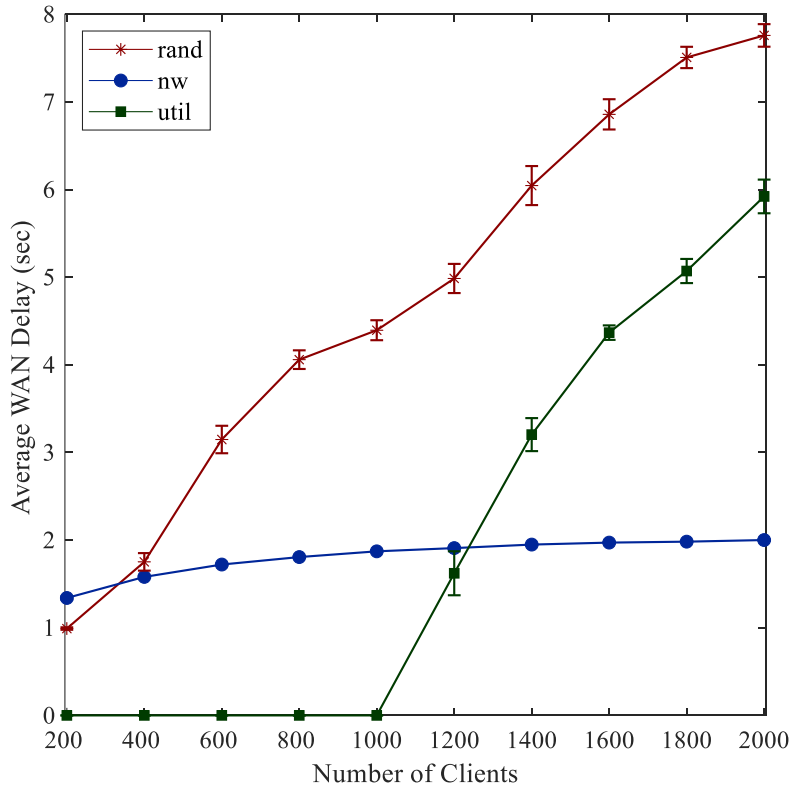


★ Processing time is a bottleneck for edge devices, while WAN delay is a bottleneck for cloud servers.

Network Delay Analysis







- WAN delay dominates the average network delay.

Case Study 4 – Server Capacity Planning

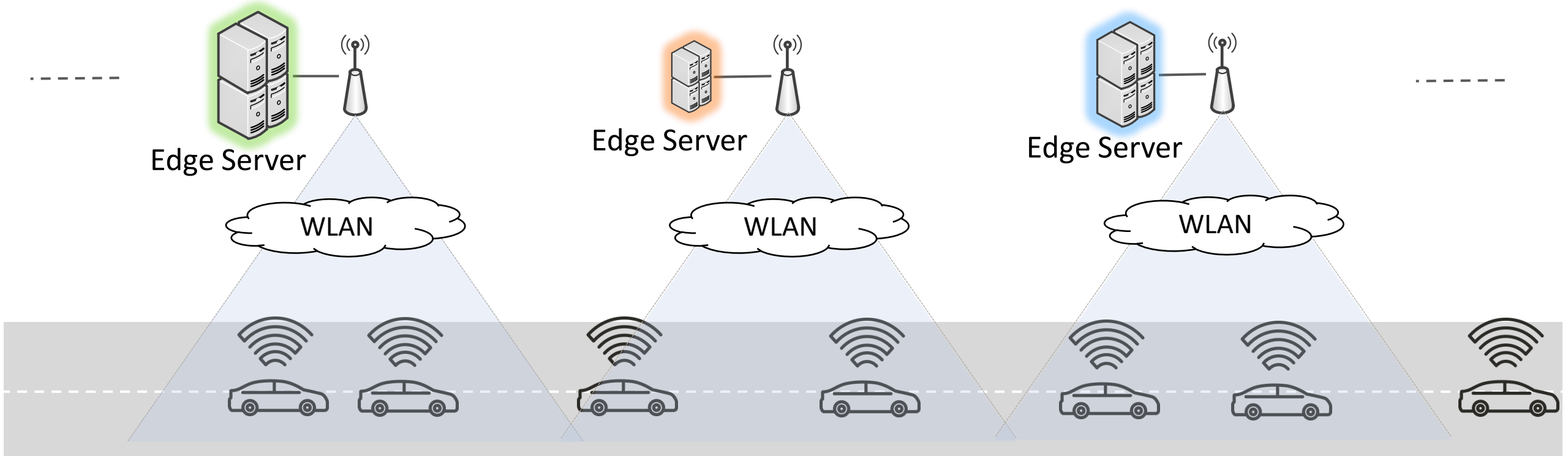
Performance Evaluation of Different Capacity Planning Approaches

Find the source code below:

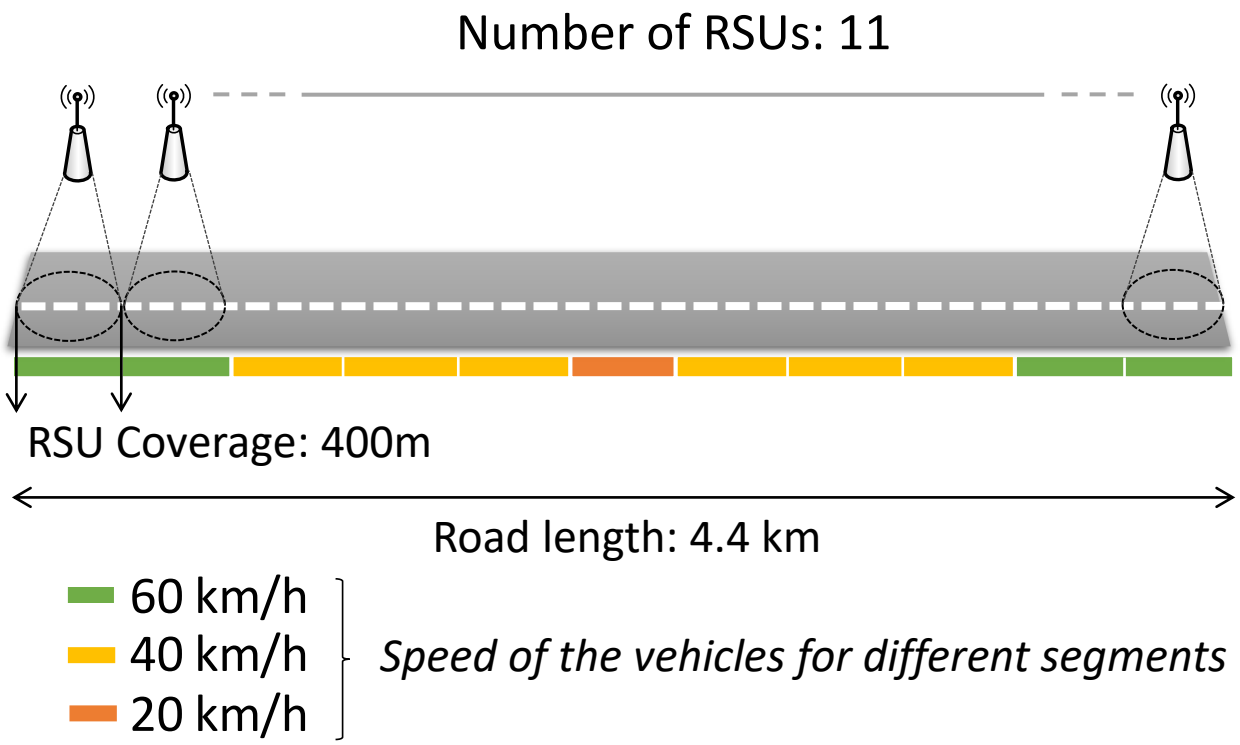
https://github.com/CagataySonmez/EdgeCloudSim/tree/master/src/edu/boun/edgecloudsim/applications/tutorial4

Simulation Scenario

- Vehicles can only offload task to the edge servers connected to the serving access point
- Edge servers run host machines of varying capacity
- This scenario compares different edge server capacity planning algorithms



Vehicular Mobility Model



- A **smart highway** environment is simulated
- 1000 to 2000 vehicles traveling on a circular road
- Dynamic velocity values based on the vehicle position is used

Competitor Capacity Planning Algorithms

- RANDOM CAPACITY
  - Total capacity is randomly assigned (allocated) to hosts
  - Total capacity is 220 GIPS
- EQUAL CAPACITY
  - Total capacity is equally distributed to the hosts
  - 20 GIPS computing capacity for all hosts
- TRAFFIC DENSITY HEURISTIC
  - Total capacity is distributed to the hosts in proportion to the intensity of the traffic
  - 44 GIPS computing capacity for the hosts in high density areas
  - 20 GIPS computing capacity for the hosts in medium density areas
  - 14 GIPS computing capacity for the hosts in low density areas

Applications Used in This Simulations

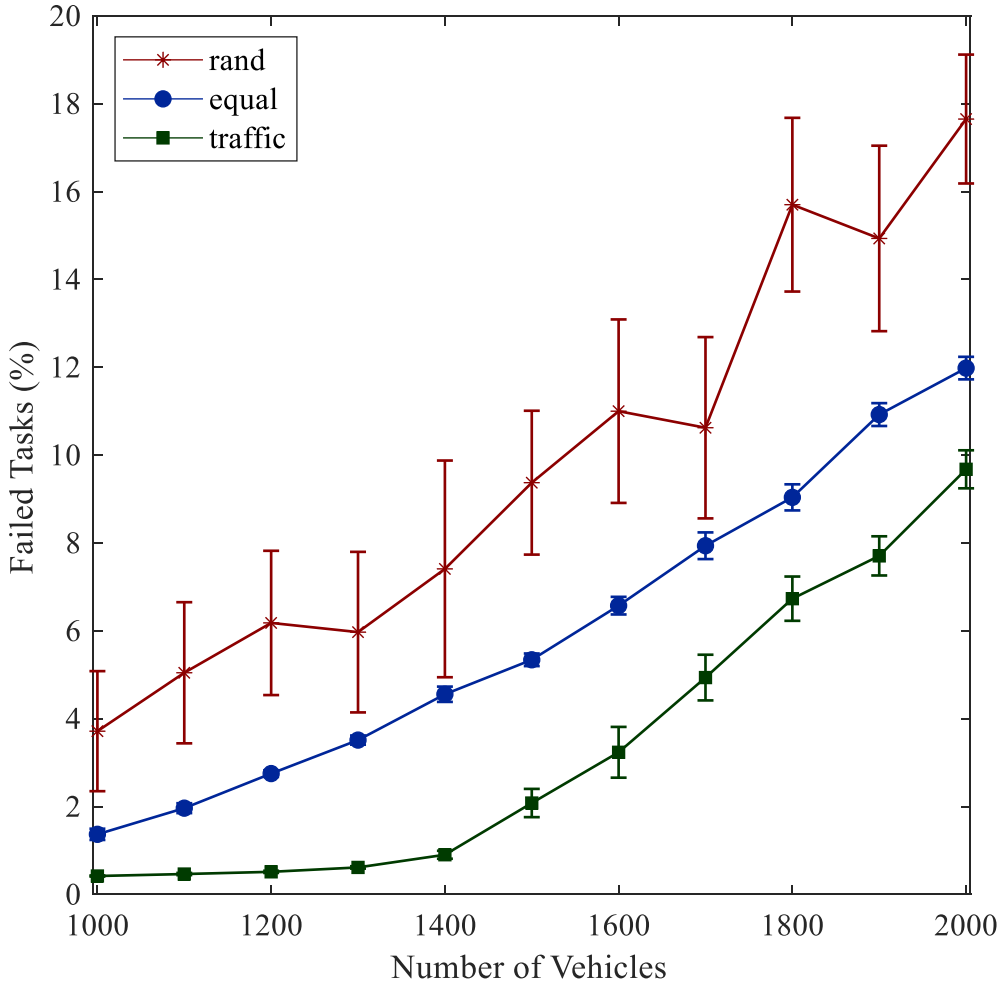
Parameter	Navigation App	Danger Assessment	Infotainment App
Usage Ratio (%)	50	25	25
Task Interarrival Time (sec)	3	5	15
Active/Idle Period (min)	always/0	always/0	always/0
Upload/Download Data (KB)	350/350	500/350	350/500
Task Length (MI)	600	1000	1600
Min - Max Edge VM Utilization(%)*	[1.3 - 4.2]	[2.2 - 7.1]	[3.4 - 11.4]

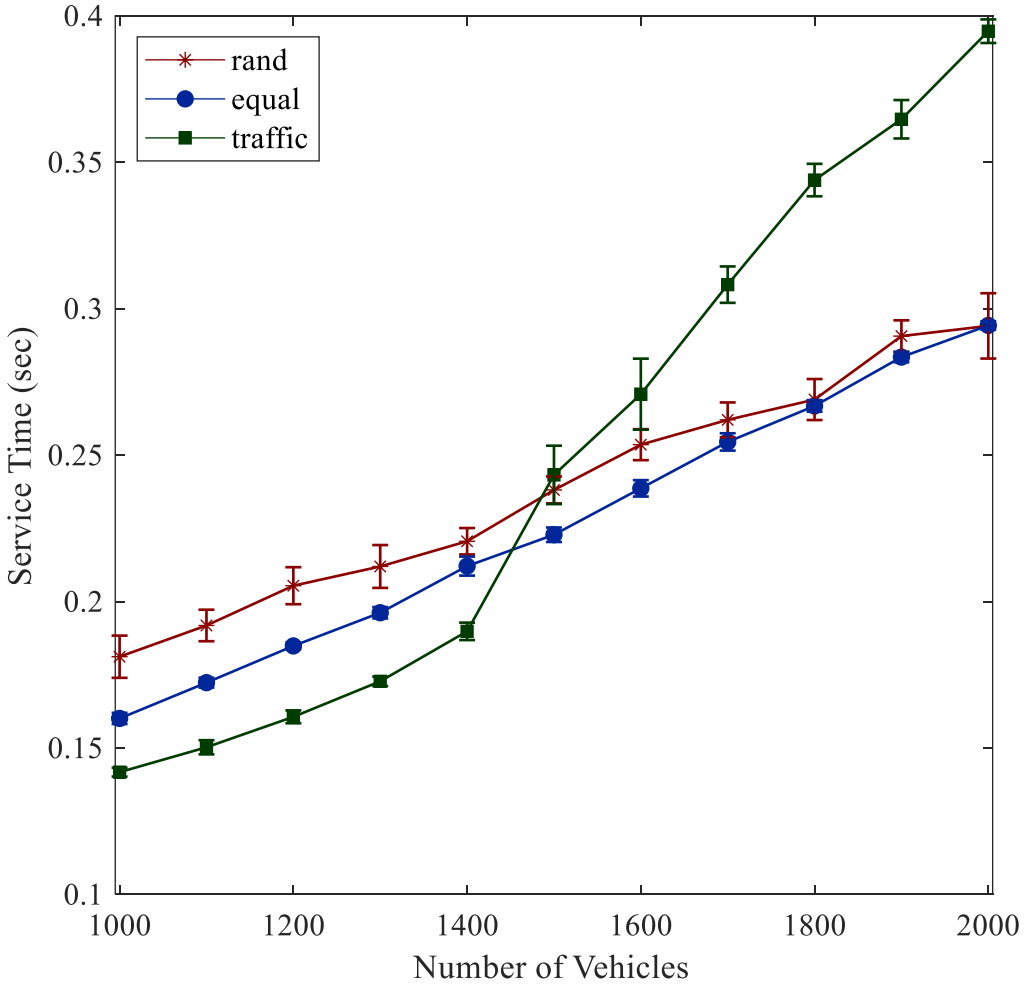
* This scenario uses a dynamic CPU utilization model as follows: *100 * (Task Lenght / VM CPU Speed)*

Simulation Parameters

Parameter	Value
Simulation Time/Warm-up Period	15/1 minutes
Number of Repetitions	10
WLAN Delay Model	Empirical
MAN Delay	Fixed (10 ms)
Number of VMs per Edge Host	2
Number of Cores per Edge VM	2
VM Processor Speed per Edge CPU	10-44 GIPS
Mobility Model	Vehicular Mobility
Number of locations for Type 1/2/3 Places	1/4/6
Speed of Vehicles in Type 1/2/3 Places	20/40/60 km/hour

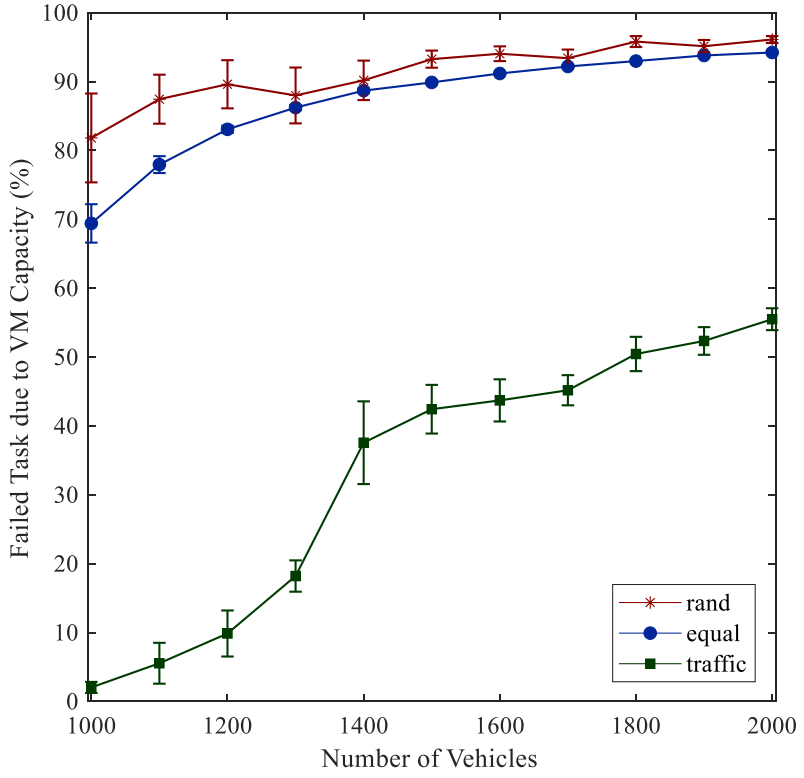
Important Simulation Results

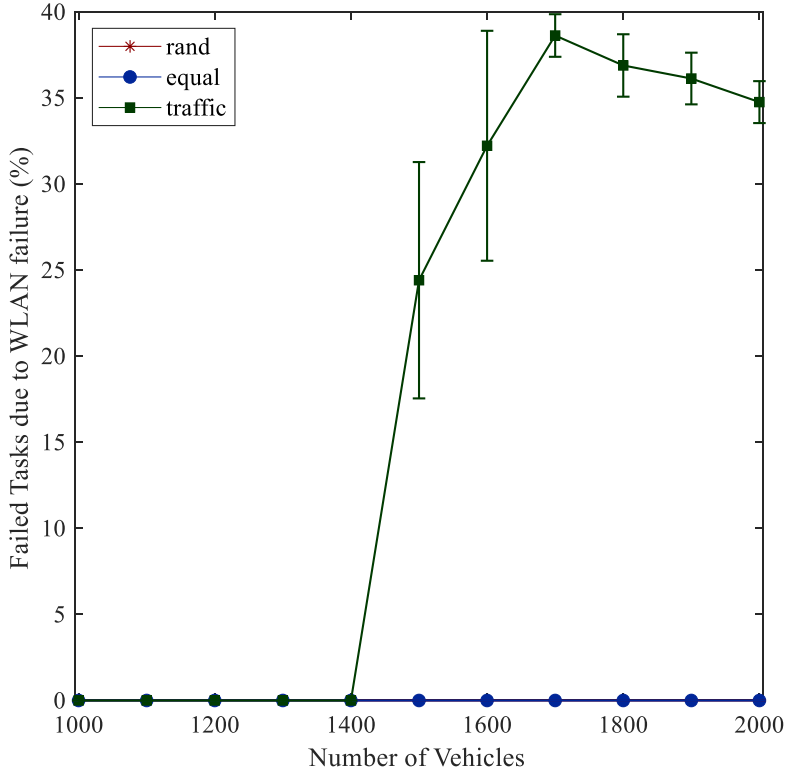


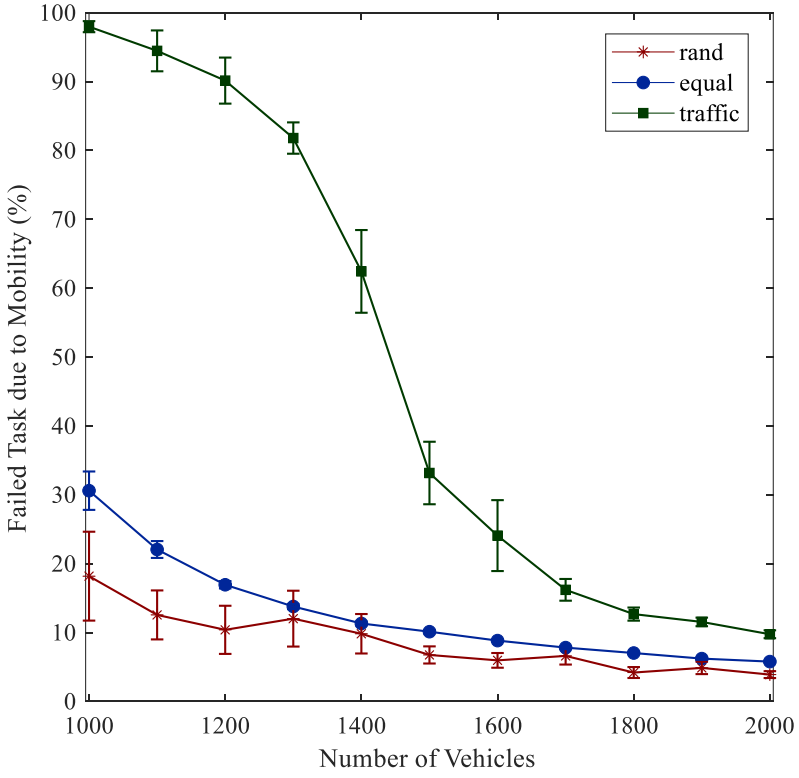


★ Graph is plotted with 95% confidence interval error bars.

Task Failure Reason Analysis

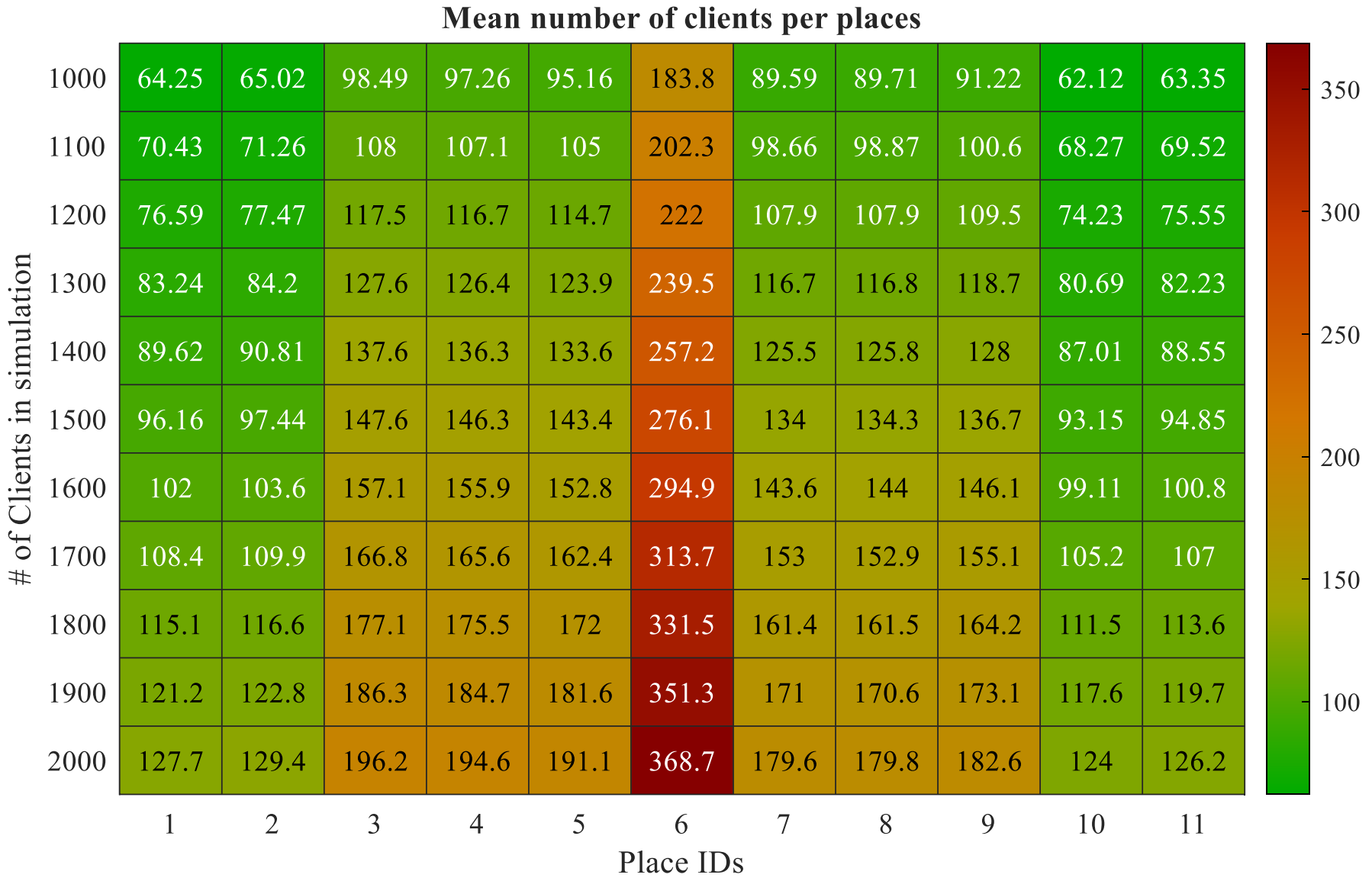




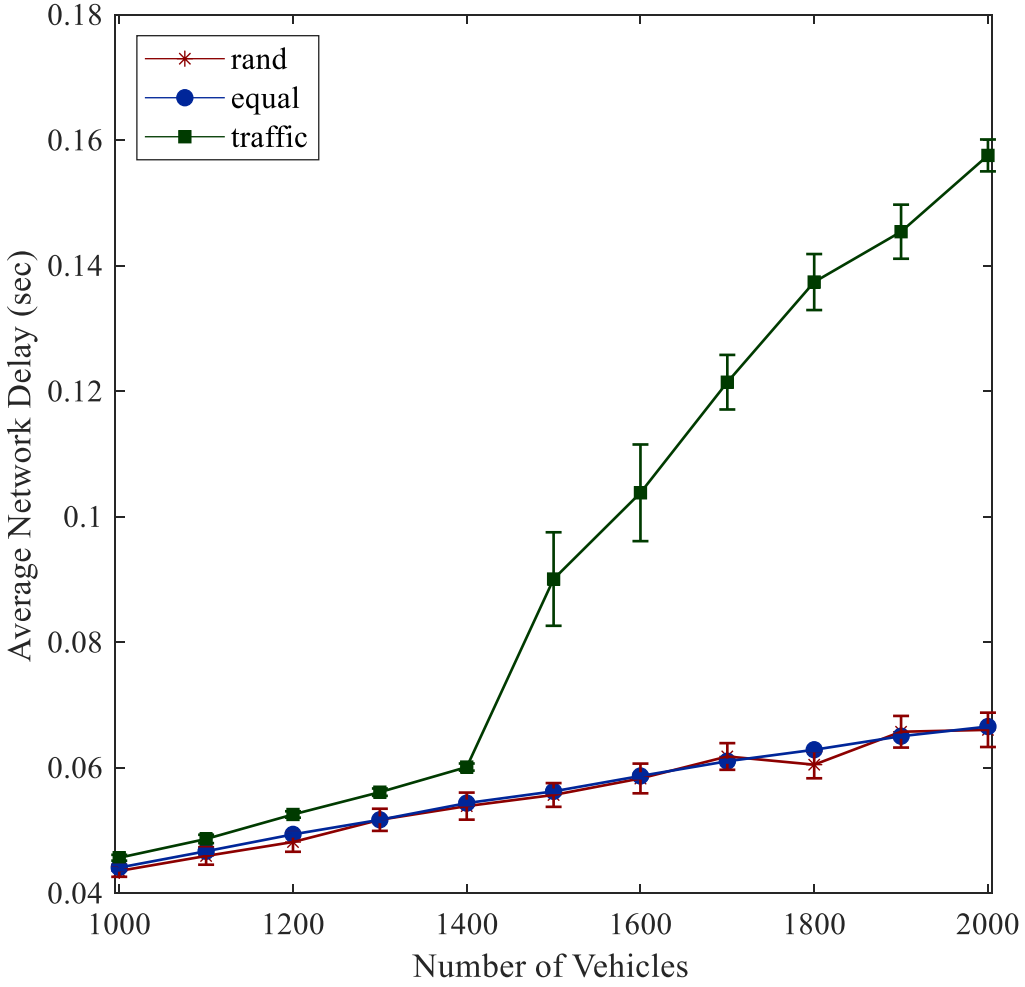


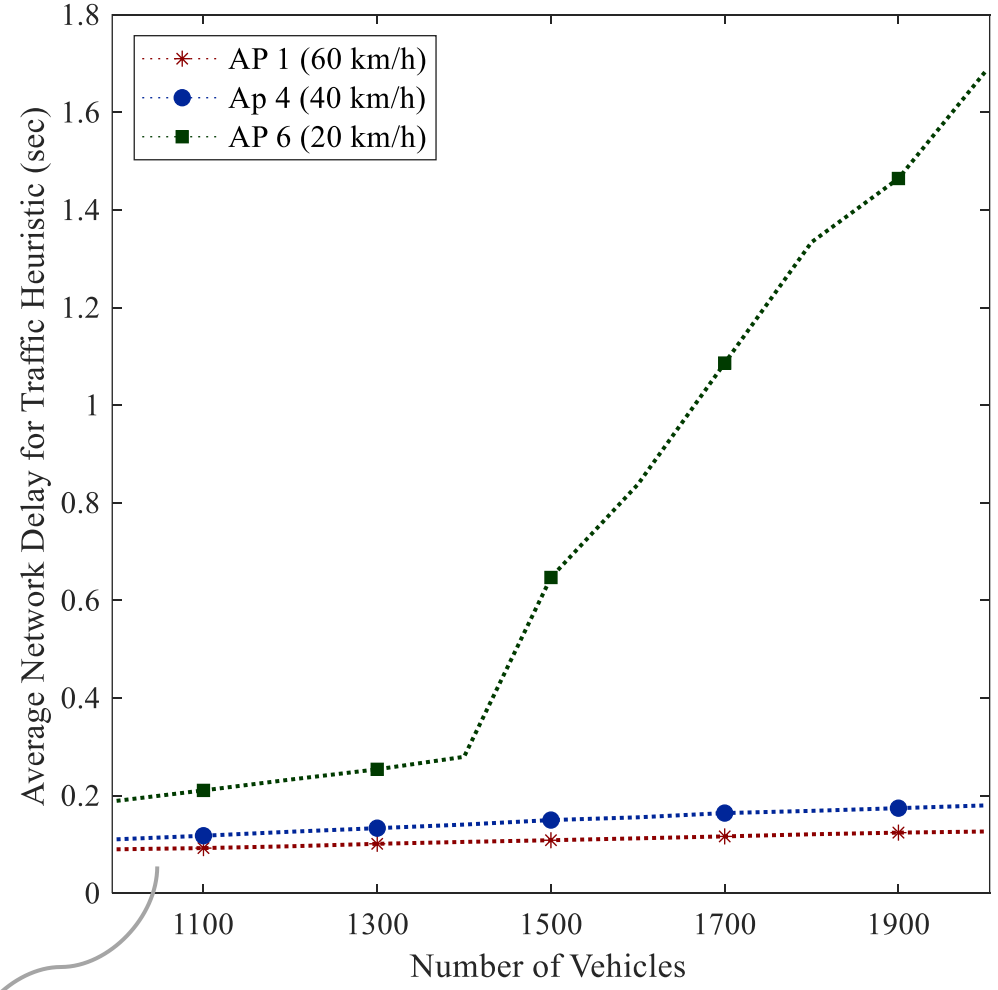
Failed tasks due to mobility dominates the failure reasons!

Mobility Model Verification



Network Delay Analysis





★ Network delay values of different access points.

Case Study 5 – Network & Server Capacity P.

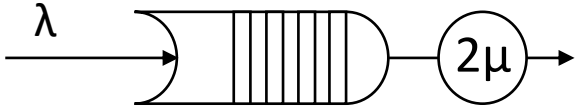
Performance Evaluation of Different Capacity Planning Approaches

Find the source code below:

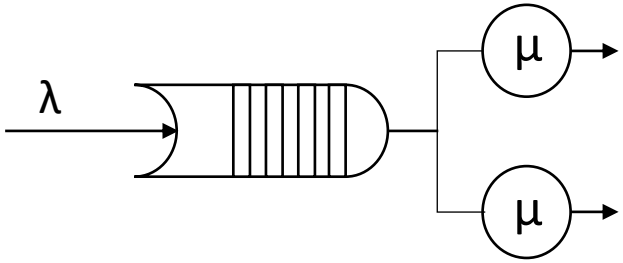
https://github.com/CagataySonmez/EdgeCloudSim/tree/master/src/edu/boun/edgecloudsim/applications/tutorial5

Which One Provides the Best Network Delay?

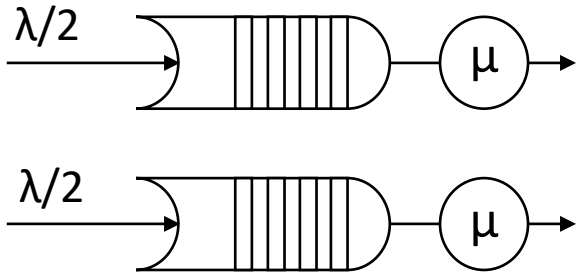
Case 1



Case 2

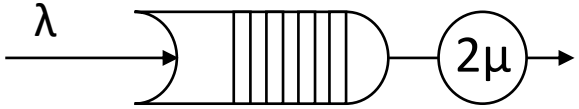


Case 3



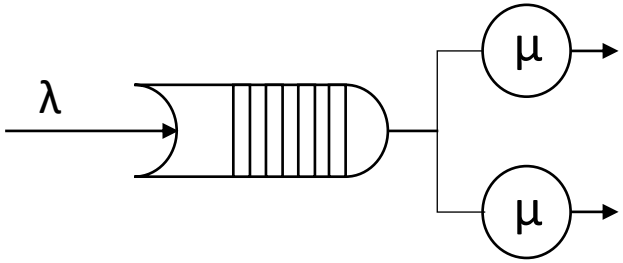
Which One Provides the Best Network Delay?

Case 1



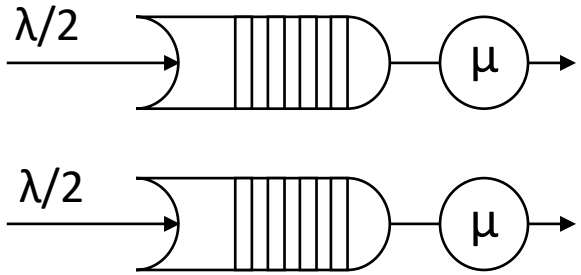
M/M/1 Queue

Case 2



M/M/2 Queue

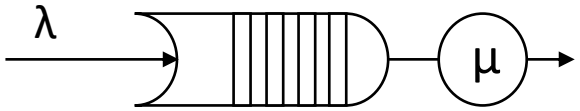
Case 3



Parallel M/M/1 Queues

Case 1: M/M/1 Queue

- Arrivals occur at rate λ according to a Poisson process
- Service times have an exponential distribution with rate parameter μ
- A **single** server serving with first-come first-served discipline



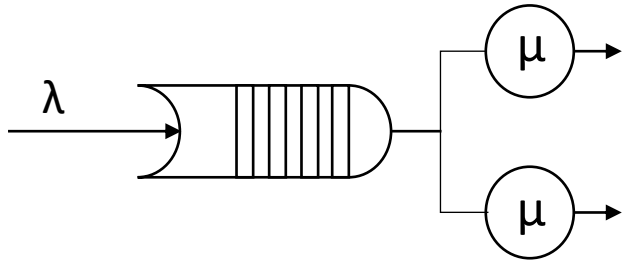
- Response time $E(T) = \frac{1}{\mu - \lambda}$, $\mu > \lambda$

capacity/packet length (p/s)

rate of the traffic (p/s)

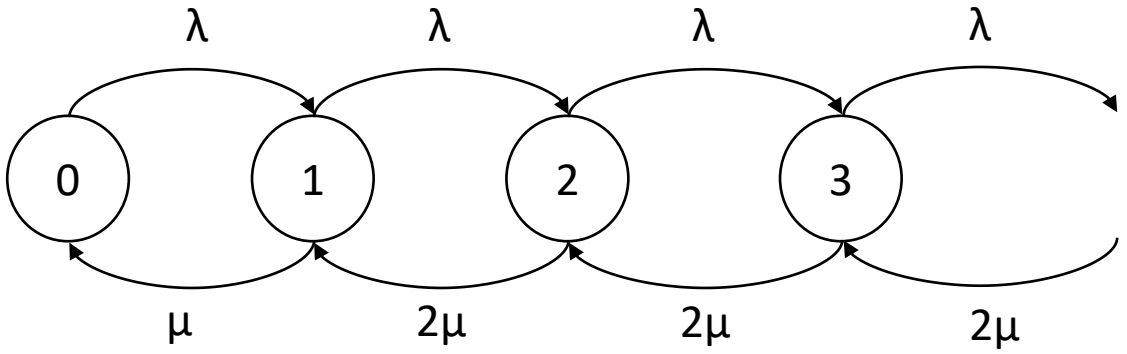
Case 2: M/M/2 Queue

- Arrivals occur at rate λ according to a Poisson process
- Service times have an exponential distribution with rate parameter μ
- A single queue with **multiple** servers



- Response time can be calculated with birth-death process model

Case 2: M/M/2 Queue cont.



$$P_0 \lambda = P_1 \mu$$

$$P_1 \lambda + P_1 \mu = P_0 \lambda + P_2 2\mu$$

$$P_2 \lambda + P_2 2\mu = P_1 \lambda + P_3 2\mu$$

$P_n \lambda = P_{n+1} 2\mu$ for $n \geq 1$

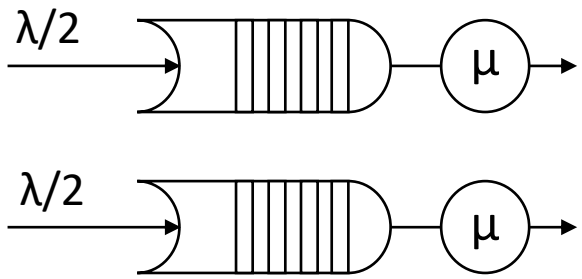
$\Sigma P_n = 1$

(The sum of all probabilities must equal 1)

After some math →

$$E(T) = \frac{4\mu}{(2\mu - \lambda)(2\mu + \lambda)}$$

Case 3: Two Parallel M/M/1 Queues

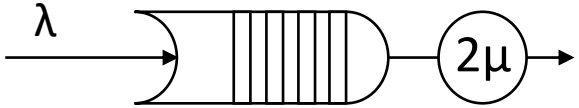


$$E(T) = \frac{1}{\mu - \lambda/2} \times 2$$

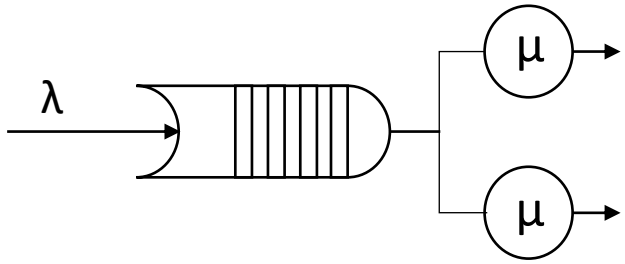
$$E(T) = \frac{4}{2\mu - \lambda}$$

Expected Network Delay for All Cases

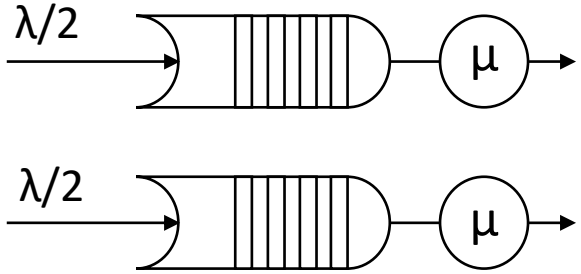
Case 1



Case 2

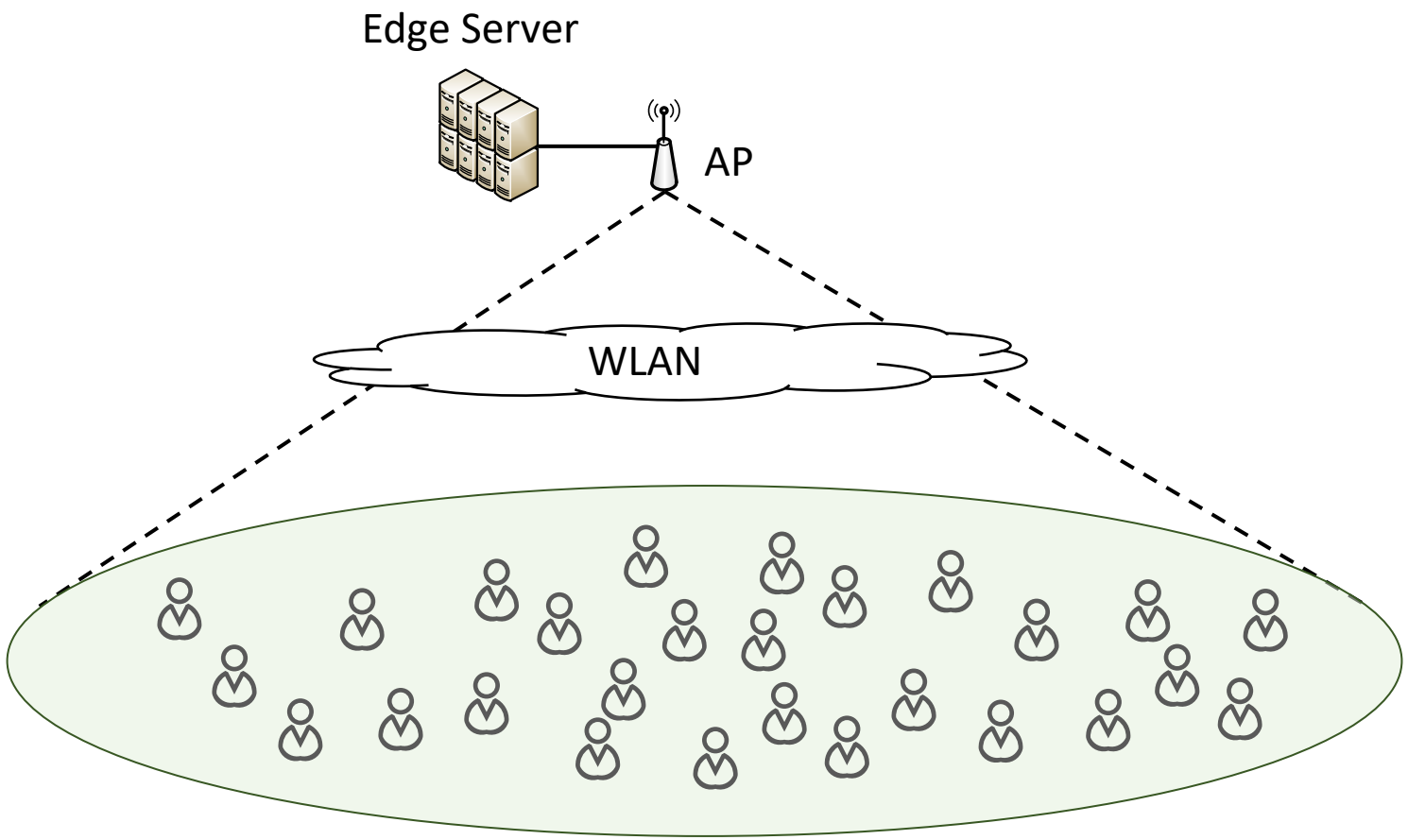


Case 3

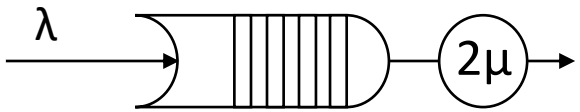


$$E(T) = \frac{1}{2\mu - \lambda} < E(T) = \frac{4\mu}{(2\mu - \lambda)(2\mu + \lambda)} < E(T) = \frac{4}{2\mu - \lambda}$$

Implementation of Case1

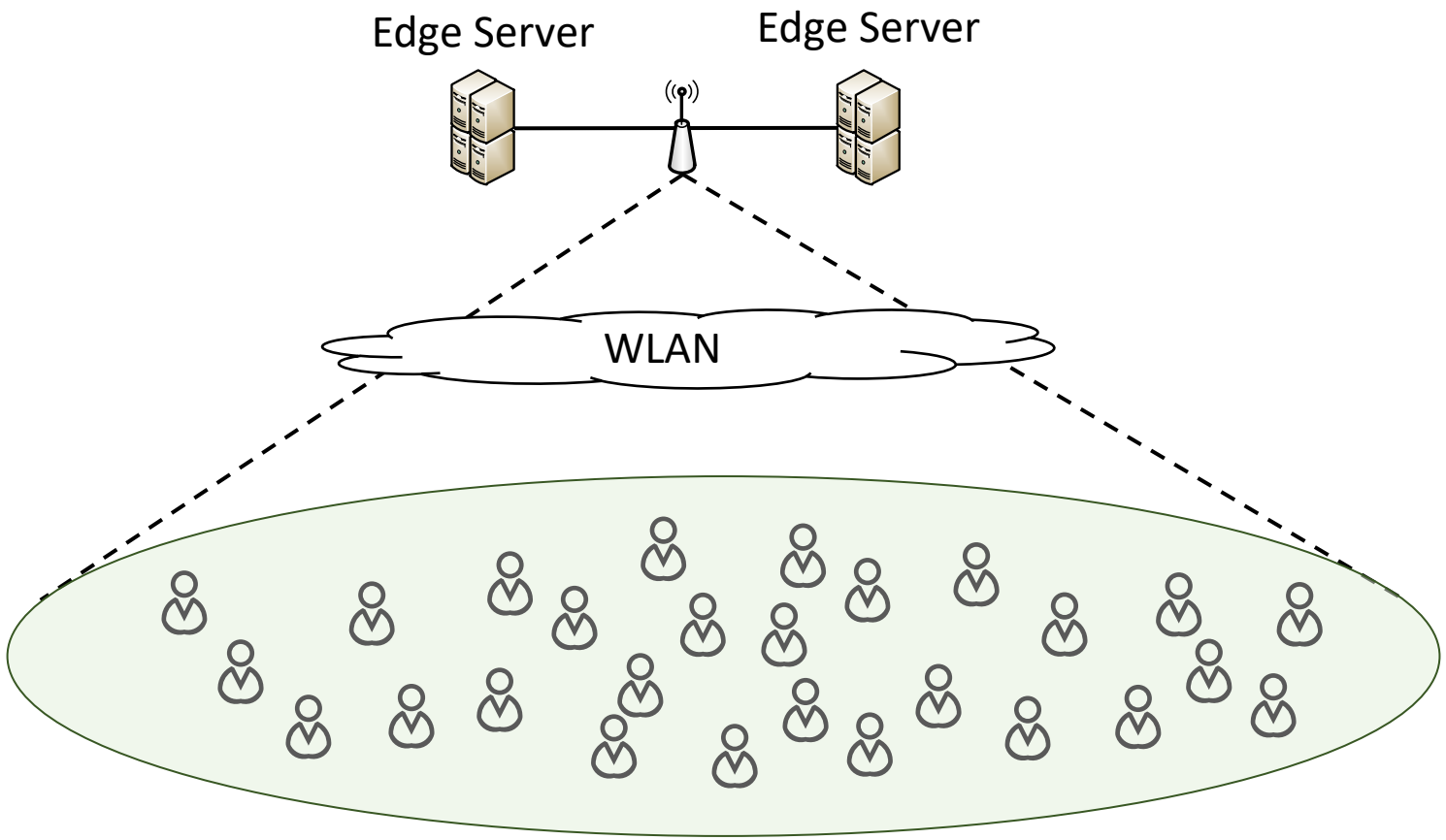


M/M/1 Queue

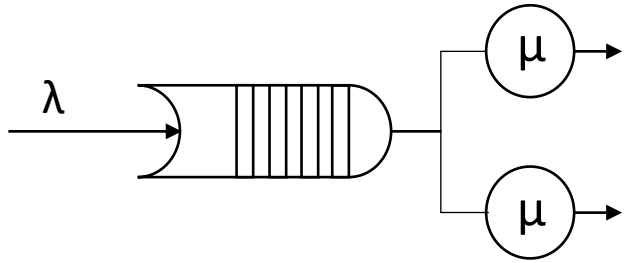


Parameter	Value
WLAN Bandwidth	100 Mbps
Capacity of Edge VM	20 GIPS
# of VMs per Edge Server	1
# of Cores per VM	1

Implementation of Case2



M/M/2 Queue

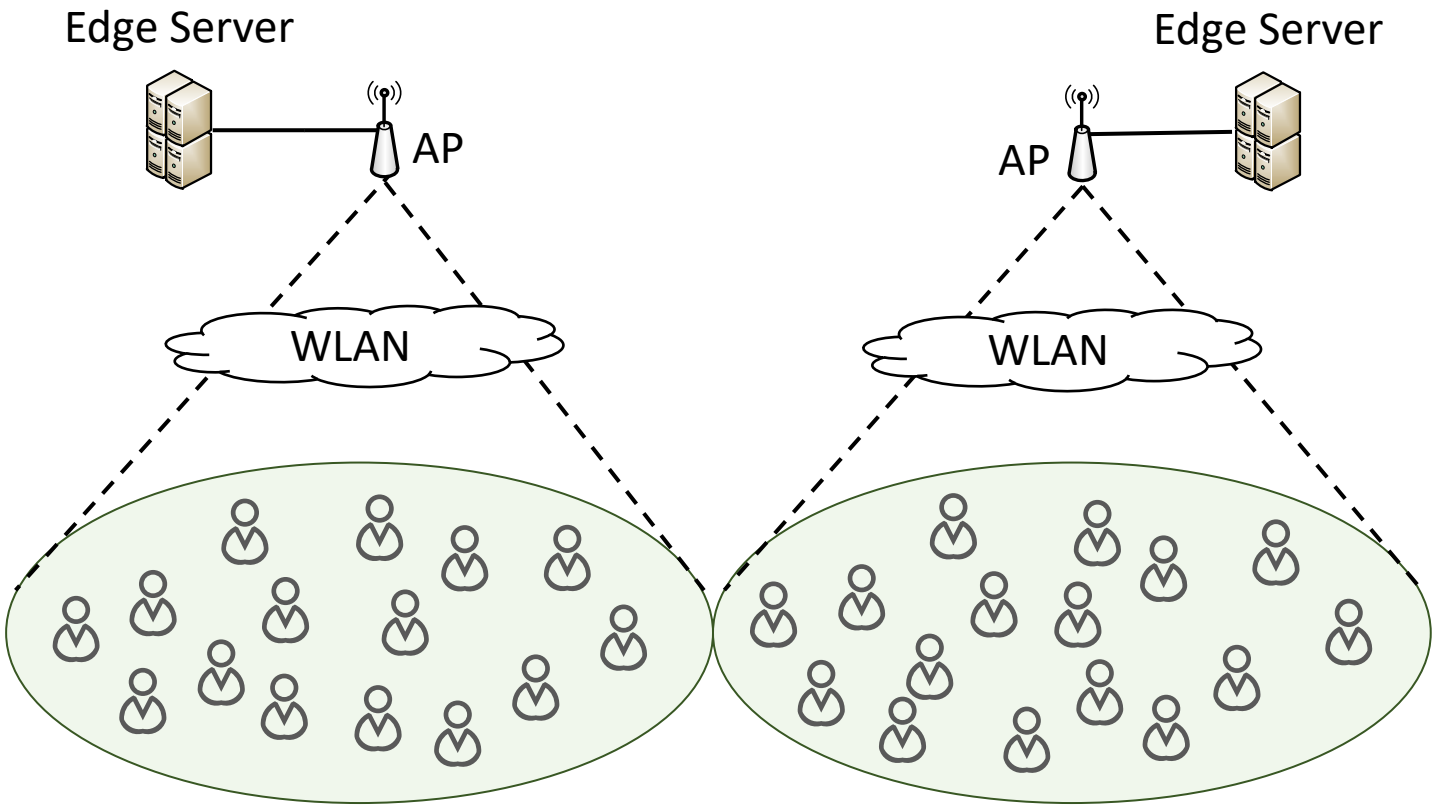


Parameter	Value
WLAN Bandwidth *	50 Mbps
Capacity of Edge VMs **	10 GIPS
# of VMs per Edge Server	1
# of Cores per VM	1

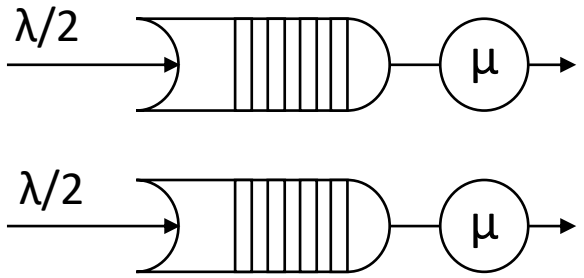
* Assume a 100 Mbps network is shared by 2 servers.

** Edge servers are simulated on 1 host with 2 VMs.

Implementation of Case3



Parallel M/M/1 Queues



Parameter	Value
WLAN Bandwidth	50 Mbps
Capacity of Edge VMs	10 GIPS
# of VMs per Edge Server	1
# of Cores per VM	1

Application Used in Simulations

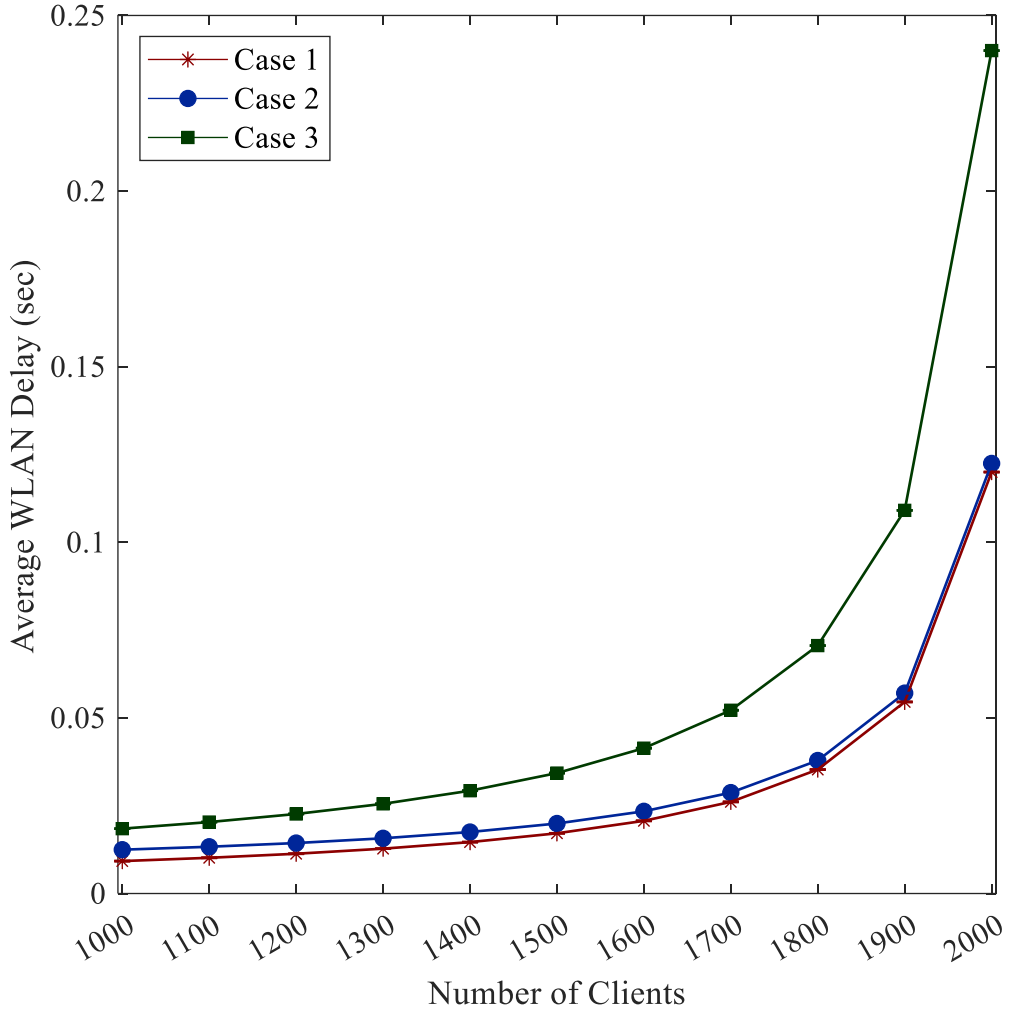
Parameter	Sample App
Task Interarrival (sec)	5
Active/Idle Period (sec)	30/1
VM Utilization on Edge/Cloud (%)	3
Task Length (MI)	125
Upload Data Size (KB)	30
Download Data Size (KB)	30

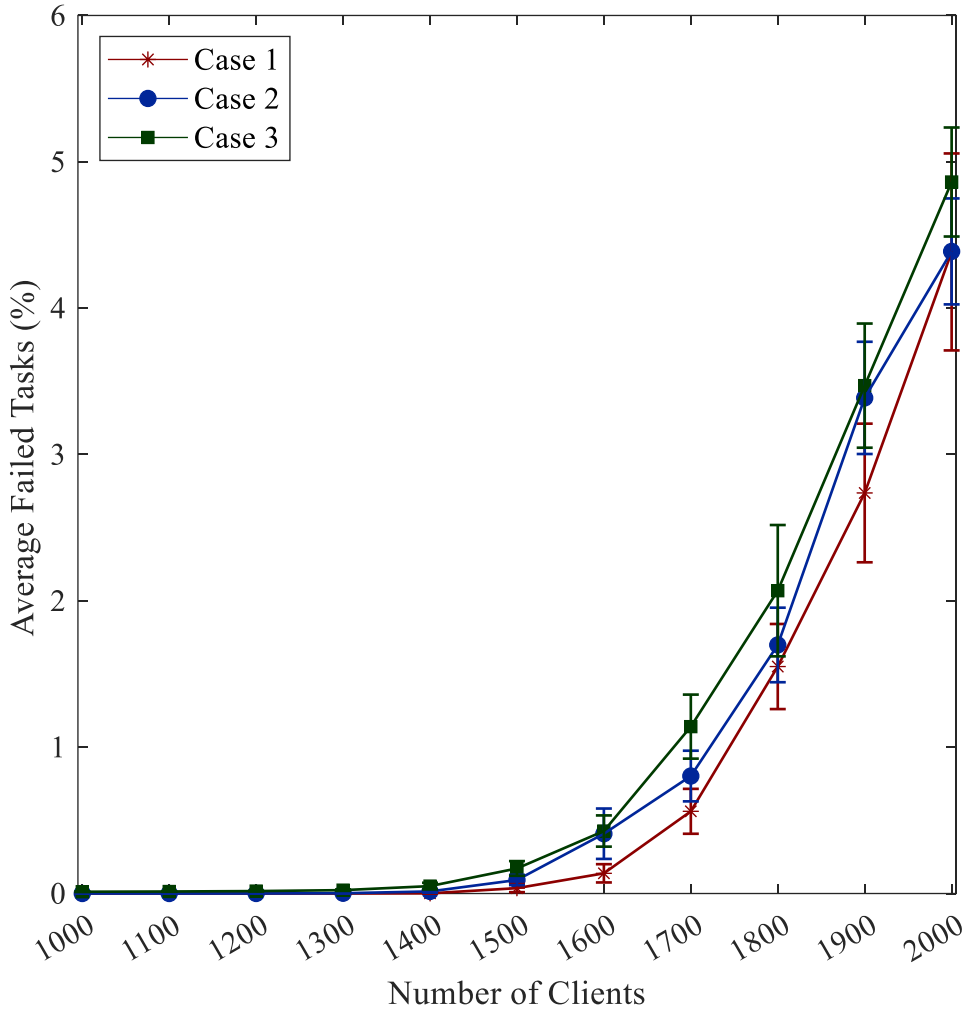
Simulation Parameters

Parameter	Value
Simulation Time	30 minutes
Warm-up Period	5 minutes
Number of Repetitions	10
Mobility Model	Nomadic Mobility
Number of Mobile Clients	1000 to 2000
Length of the Simulated Area*	6 KM

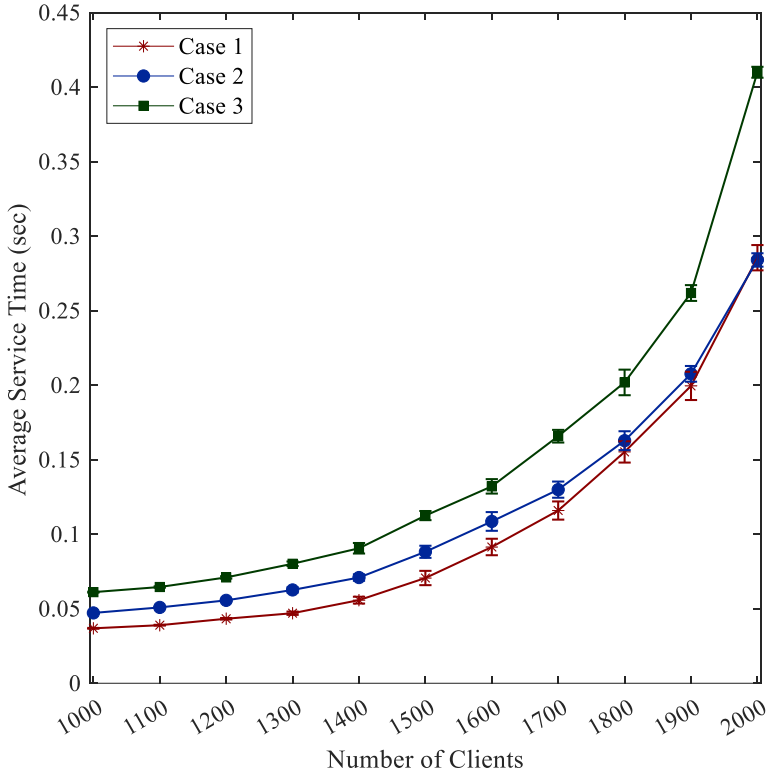
* Simulated area is a symbolic value; in the end, we have 1 or 2 places depending on the scenario.

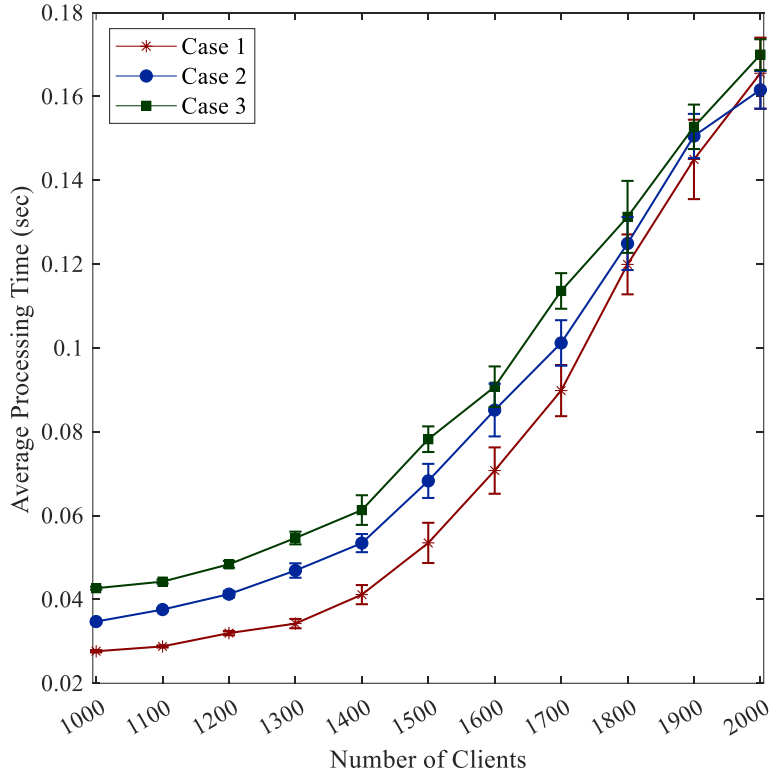
Average WLAN Delay & Success Rate

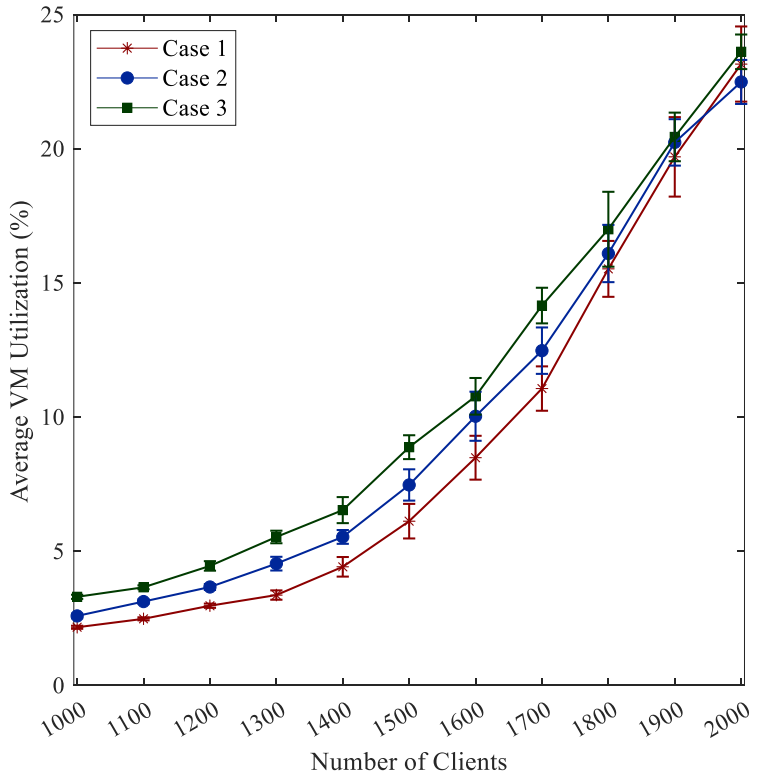




Edge Server Side Statistics







★ CloudSim's task scheduler algorithm provides results similar to our M/M/1 & M/M/2 queue models.

References

[1] Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms”, *Softw. Pract. Exper.*, Vol. 41, No. 1, pp. 23–50, Jan. 2011.

[2] Howell, Fred, and Ross McNab. "SimJava: A discrete event simulation library for java." *Simulation Series* 30 (1998): 51-56.

[3] Gupta, Harshit, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments." *Software: Practice and Experience* 47, no. 9 (2017): 1275-1296.

[4] Brogi, Antonio, and Stefano Forti. "QoS-aware deployment of IoT applications through the fog." *IEEE Internet of Things Journal 4*, no. 5 (2017): 1185-1192.

[5] Sonmez, Cagatay, Atay Ozgovde, and Cem Ersoy. "Edgecloudsim: An environment for performance evaluation of edge computing systems." *Transactions on Emerging Telecommunications Technologies* 29, no. 11 (2018): e3493.

[6] Tuli, Shreshth, Redowan Mahmud, Shikhar Tuli, and Rajkumar Buyya. "Fogbus: A blockchain-based lightweight framework for edge and fog computing." *Journal of Systems and Software* 154 (2019): 22-36.

[7] Qayyum, Tariq, Asad Waqar Malik, Muazzam A. Khan Khattak, Osman Khalid, and Samee U. Khan. "FogNetSim++: A toolkit for modeling and simulation of distributed fog environment." *IEEE Access* 6 (2018): 63570-63583.

[8] N. Mohan and J. Kangasharju, "Edge-Fog cloud: A distributed cloud for Internet of Things computations," 2016 *Cloudification of the Internet of Things (CIoT)*, 2016, pp. 1-6.

[9] A. Coutinho, F. Greve, C. Prazeres and J. Cardoso, "Fogbed: A Rapid-Prototyping Emulation Environment for Fog Computing," 2018 *IEEE International Conference on Communications (ICC)*, 2018, pp. 1-7.

[10] R. Mayer, L. Graser, H. Gupta, E. Saurez and U. Ramachandran, "EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures," 2017 *IEEE Fog World Congress (FWC)*, 2017, pp. 1-6.

[11] M. Etemad, M. Aazam and M. St-Hilaire, "Using DEVS for modeling and simulating a Fog Computing environment," 2017 *International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 849-854.

[12] "EdgeCloudSim Discussion Forum", https://groups.google.com/u/1/g/edgecloudsim, accessed in October 2025.

[13] "EdgeCloudSim YouTube Channel", https://www.youtube.com/channel/UC2gnXTWHHN6h4bk1D5gpcIA, accessed in October 2025.

Thank You