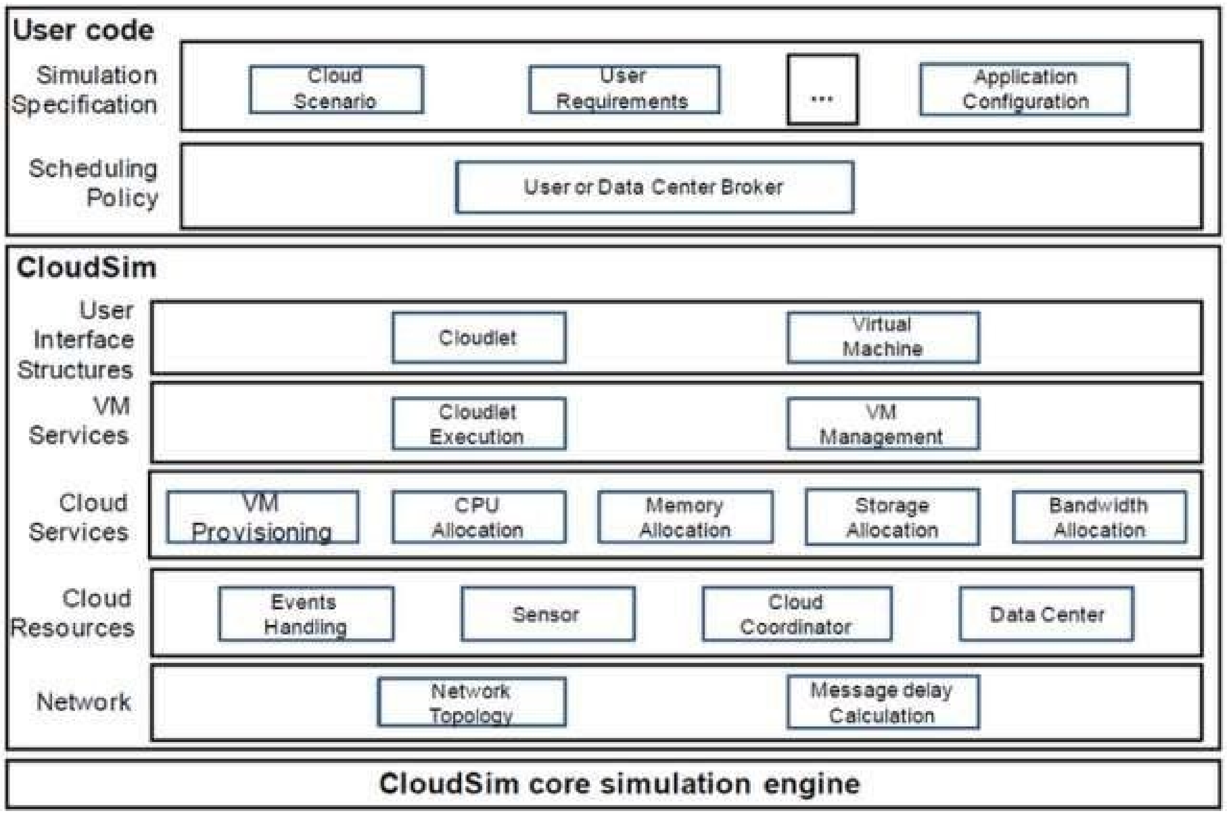
IotSim-Edge

I. Introduction to Cloudsim

CloudSim is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment, allowing for easy experimentation and design of algorithms in what would otherwise be a costly process

It is a library for the simulation of cloud scenarios. It provides essential classes for describing data centres, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning.

The CloudSim Core simulation engine provides support for modeling and simulation of virtualized Cloud-based data center environments including queuing and processing of events, creation of cloud system entities (like data center, host, virtual machines, brokers, services, etc.) communication between components and management of the simulation clock.



II. Classes & how they work

1. ***Cloudlet*** - Cloudlet is one of the most important models which defines the specifications for a simulation engine corresponding to the real-life candidate application to be considered for moving to a Cloud-based system. We can also define this cloudlet as a single process or task being executed on the cloud-based system which is to be simulated through a Cloudsim simulation engine.
2. ***CloudletScheduler*** **-** This is responsible for the implementation of different policies that determine the share of processing power among Cloudlets in a VM. Allows for experimentation to find the optimal policy for an application.
3. ***Datacenter*** - A set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations. It deals with processing of VM queries (i.e., handling of VMs) instead of processing Cloudlet-related queries.
4. ***Datacenter Broker / Cloud Broker*** **-** DatacentreBroker represents a broker acting on behalf of a user. A broker, which is responsible for mediating negotiations between SaaS and Cloud providers and such negotiations are driven by QoS requirements. It hides VM management, as VM creation, submission of cloudlets to this VMs and destruction of VMs.
5. ***Host*** - Host is a physical resource such as a computer or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (if it is a multi-core machine), an allocation of policy for provisioning the compute, memory and bandwidth to the VMs.
6. ***Virtual Machine*** **-** Every VM component has access to a component that stores the following characteristics related to a VM (i.e.) accessible memory, processor, storage size, and the VM's internal provisioning policy that is extended from an abstract class called the CloudletScheduler. VMAllocation Policy is an abstract class that represents a provisioning policy to be utilized by VM Monitor for mapping VMs to hosts. The primary role is to select the best fit host in a data center that meets the memory, storage, and availability requirement for VM deployment mapping.

III. Introduction to IotSim-Edge

Advances in IoT have a transformative impact on society and the environment through a multitude of application areas including smart homes, smart agriculture, manufacturing and healthcare. To achieve this, an ever increasing number of heterogeneous IoT devices are continuously being networked to support real-time monitoring and actuation across different domains. Traditionally, the enormous amount of data, generally known as the big data, is sent to the cloud by IoT devices for further processing and analysis. However, the centralised processing in cloud is not suitable for numerous IoT applications due to the following reasons:

(i) Some applications require close coupling between request and response

(ii) Delay incurred by the centralised cloud-based deployment is unacceptable for many latency-sensitive applications

(iii) There is a higher chance of network failure and data loss

(iv) Sending all the data to cloud may drain the battery of the IoT device at a faster rate

The introduction of edge computing addresses these issues by providing the computational capacity in a near proximity to the data generating devices. Smart edge devices such as Smart phone, Raspberry Pi, UDOO board, etc. supports local processing and storage of data on a widespread but smaller scale. However, the constituent devices in edge computing are heterogeneous as each one may have specific architecture and follows particular protocols for communication.

Evaluating the framework in a real environment gives the best performance behaviour, however, it is not always possible as most of the test frameworks are in development. Even if the infrastructure is available, it is very complex to perform the experiment as setting it up requires knowledge of all the associated IoT and edge devices which is not intuitive. Running multiple experiments to test a desired framework requires reconfiguration of multiple devices and changes to required parameters quickly becomes untenable due to the volume of changes needed. Additionally, performing the experiment in the real environment is very expensive due to the set up and maintenance cost incurred. Since, the real environment is dynamic, it is very hard to reproduce the same result for different iterations which may lead to misinterpretation of the evaluation. All these challenges hinders the use of real environment for benchmarking the edge computing environments. To overcome this issue, another feasible alternative is the use of **simulators**. The simulators offer a window of opportunity for evaluating the proposed hypothesis (frameworks and policies) in a simple, controlled and repeatable environment. A simulation environment must mimic the key complexity and heterogeneity of real networks and support multiple scenarios that affect real IoT deployments.

IV. Architecture of IotSim-Edge

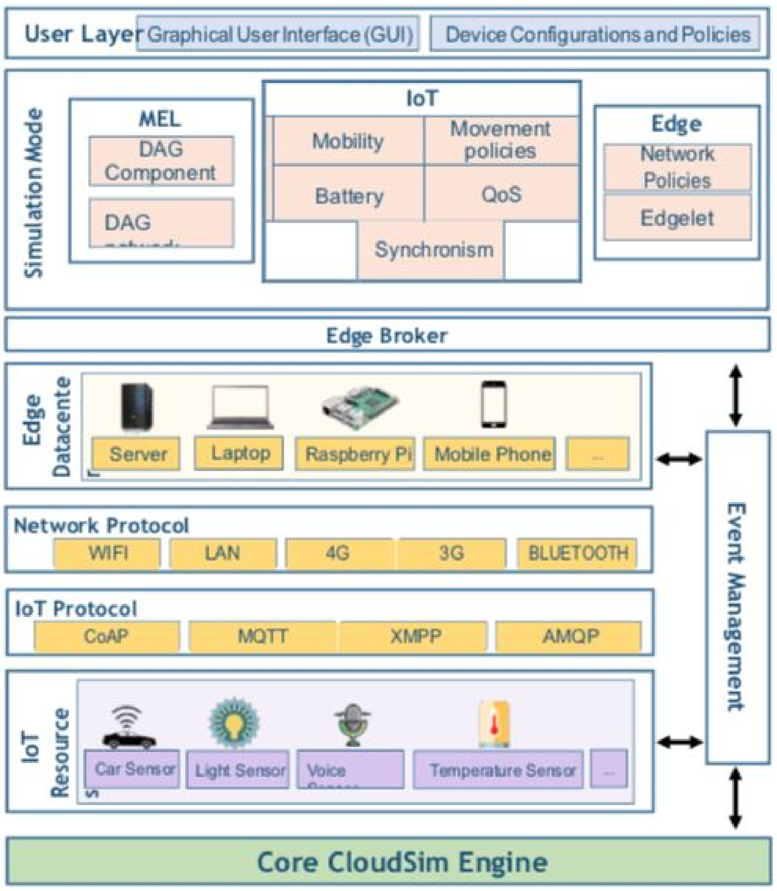


Fig. 2: The architecture of IoTSim-Edge simulator.

IoTSim-Edge is built on the top of CloudSim simulation tool. CloudSim provides the underlying mechanisms for handling communication among subscribed components (e.g. broker, edge datacenter, IoT resources) using an event management system. The core components of CloudSim are extended to represent the edge infrastructure in line with edge's features and characteristics. IoT resources layer contains different types of IoT devices (e.g. car sensor, motion sensor) where each one has their own features and behaviours along with performing different operations of sensing and actuation. Sensors in the IoT device seamlessly generates data while actuators are responsible for generating the response. Traditionally, cloud resources are used for processing IoT data. But in Edge-IoT approach, sensor data is processed in the edge datacenter for faster processing time. Edge datacenter consists of heterogeneous processing devices such as smartphone, laptop, Raspberry Pi and single server machine. User can provide users input through a graphical user interface (GUI) by mentioning different device configuration and policies. Edge-IoT management layer consists of several components such as Edgelet, policies, mobility, battery, synchronism, QoS, network protocols, communication protocols, transport protocols, and security protocols.

For the implementation of IoTSim-Edge, existing classes of CloudSim are extended as shown below as well as numerous new classes are defined in order to model realistic IoT and edge environments. Any entity that extends SimEntity class can seamlessly send and receive events to other entities when required through the event management engine.

For modelling an edge infrastructure, new classes are designed and implemented as shown and explained below. The main classes are EdgeDataCenter, EdgeBroker, EdgeDatacenterCharacteristics, EdgeDevice, MicroELement, and EdgeLet.

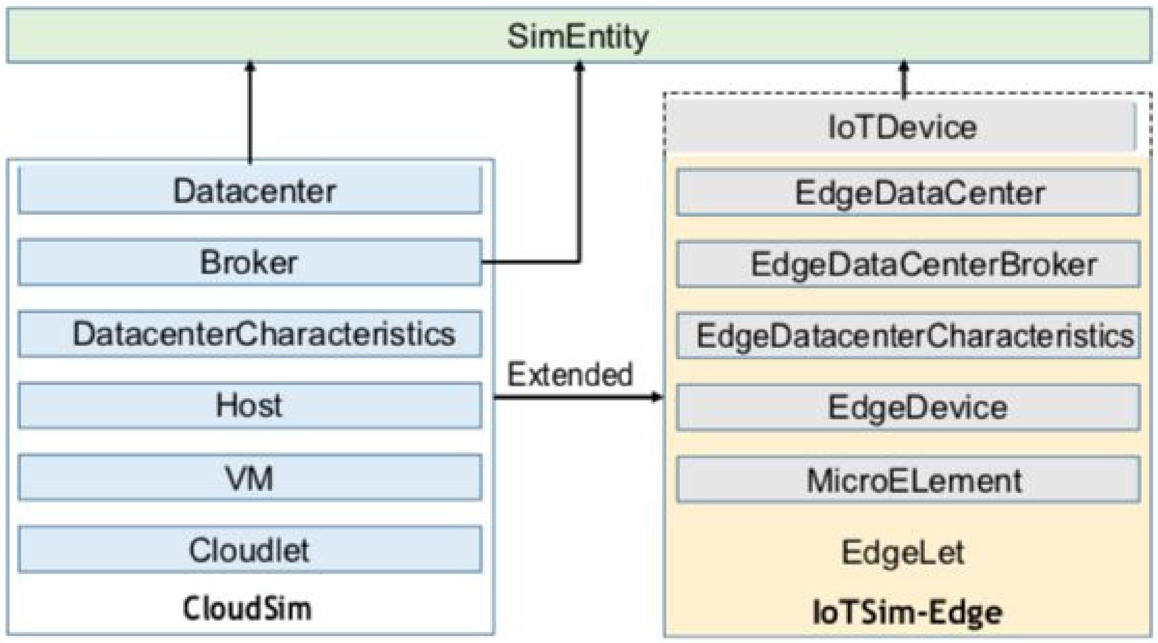


Fig. 3: Used classes of ClousSim in IoTSim-Edge plus classes use event management system

V. How IotSim-Edge builds on Cloudsim

For modeling an edge infrastructure, numerous new classes are designed and implemented, utilizing the infrastructure of Cloudsim. A few of them are listed below:

1. ***EdgeLet*** - This class models the IoT generated data and MEL processing data. Once an IoT device establishes connection with its respective edge device(s), it generates IoT sensed data as required in a form of EdgeLet.
2. ***EdgeDataCentre*** - It intercepts all incoming events and performs different operations based on the payload of the event, such as resource provisioning and submitting EdgeLet requests to respective MEL(s).
3. ***EdgeBroker*** - This class is a users' proxy, in which it generates users' requests in accordance with their prescribed requirements. Its duties include requesting IOT devices to send data to their respective edge devices and receiving processing results from MEL.
4. ***EdgeDevice*** - It hosts several MELs and facilitates the procedure of CPU sharing mechanism via a given CPU sharing policy.
5. ***IoTDevice*** - This class models the core characteristics of IoT devices. Any new required type of IoT device can easily extend the IoTDevice Class and implement the new features. This class uses IOTProtocol and NetworkProtocol classes to categorize any IOT device.
6. ***MEL*** - This class represents one component of the IoT application graph. It represents the main processing requirement of the application component.

VI. Extensions to Cloudsim to solve challenges in IoT environment

There are many challenges involved in simulating an IoT environment. Some of them involve:

- Variety of heterogeneous devices interacting with each other through multiple protocols.
- Modeling data and control-flow dependencies between IoT and edge layers.
- Capacity planning is challenging as it depends on various configuration parameters such as mobility, velocity, battery etc.
- Dynamicity of IoT environment leads to addition and removal of IoT and edge devices very frequently.

IoTSim-Edge tackles these challenges in novel ways. It uses a graph modeling abstraction with MicroELements (MELs), allowing us to model various kinds of applications. It has a high level of abstraction allowing us to combine heterogeneous IoT devices and all their configuration parameters (battery capacity, mobility etc.) for real-world applications.

In essence, it provides a generic model that can be adapted for the specifics of any application using a number of IoT devices which in turn use a number of configurations varying in protocol, battery capacity, mobility etc.