Challenges IoTSim Edge solves

Cloud computing is a model for on-demand access to a shared pool of configurable resources (e.g. compute, networks, servers, storage, applications, services, and software). Cloud computing platforms are well suited for hosting IoT applications as they offer an elastic hardware resources (e.g. CPU, Storage, and Network) that can be scaled on-demand for handling large quantities of data from IoT applications with uncertain volume, variety, velocity, and query types.

These two technologies are inherently and increasingly getting entwined with each other. Because of this, evaluation and analysis of IoT applications in a real cloud computing environment can be a challenge for several reasons:

- It is not cost-effective to procure or rent a large scale datacenter resource pool that will accurately reflect realistic application deployment and let practitioners experiment with dynamic hardware resource and big data processing framework configurations, and changing data volume, velocity, and variety.
- Frequently changing experiment configurations in a large-scale real test bed involves a lot of manual configuration, making the performance analysis itself time-consuming. As a result, the reproduction of results becomes extremely difficult.
- The real experiments on such large-scale distributed platforms are sometimes impossible due to multiple test runs in different conditions.
- It is almost impractical to set up a very large cluster consisting of hundreds or thousands of nodes to test the scalability of the system.

An obvious solution to the aforementioned problems is to use a simulator supporting IoT application processing. A simulator not only allows us to measure scalability of computing resources for IoT applications efficiently, but also enables us to determine the effects of various variables like datacenter configuration, the numerous types of IoT devices and their communication protocols etc.

However, there are quite a few challenges in developing an IoT simulator as well due to the heterogeneous nature of the environment. Some challenges and the manner in which IotSim-Edge solves them are:

Challenge #1:
Variety of IoT devices need to be combined with edge device and cloud to satisfy the requirements of an application

A typical IoT application depends on a variety of devices, each of which different in any number of ways. The datacenters at the Edge, which can be a Raspberry Pi or a smartphone to anything in between, increases the complexity of any feasible simulator.

IoTSim has a layered architecture, one which builds over CloudSim's Core engine. The IoT Datacenter is extended from the CloudSim Datacenter to include functionalities of storing, processing and analysing data for IoT services. 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.

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. The whole issue is abstracted and can be configured easily for the purposes of the simulation. Thus, IoTSim Edge is capable of supporting a variety of devices and edge devices to satisfy the requirements of an application.

Challenge #2:
Modeling networking graph between diverse type of IoT and edge computing device in an abstract manner can be very challenging

NetworkProtocol class presents the modeling of network protocols (e.g. WiFi, 4G LTE). Implementing such models in the IoTSim-Edge framework is required to properly evaluate the performance of IoT-Edge applications. Each network type is designated with its relative network speed (e.g. 200 Mbps for WiFi, 150 Mbps for 4G LTE). By modeling the transmission rate, transmission time can be taken into account the EdgeLet size.

Due to the distributed nature of IoT, there are a large number of devices present, connected in necessarily complicated ways. A suitable manner of representation is required to model any non-trivial application.

IoTSim solves this issue by abstracting the connections and making them configurable through the use of a JSON file, which specifies how the devices are connected in a simple and human-readable manner.

Challenge #3:
Modeling data and control flow dependencies between IoT and edge layers to support diverse data analysis work-flow structure is non-trivial.

Two types of IoT nodes could be modelled and configured, i.e. sensors and actuator nodes. Sensing nodes will collect the information of surroundings through sensors and send the information for processing and storage. Actuators will be activated based on the analysis of the data.
The communication layer is responsible for data transfer to/from IoT devices, edge devices and cloud. Employing combinations of these various node types enables the modelling of all topologies and architectures of IoT systems.

IoT nodes are configured with a power source that could be a battery, USB charging point or continuous power supply. Battery consumption is tracked during the simulation for real deployments. Nodes are associated with different types of connections (e.g. 3G, Bluetooth, WiFi) and tracked signal strength.

Possible data of each sensor are stored in a JSON file, and the sensors are configured to read data from their files and submit a reading at each time interval. Data could be selected sequentially, randomly, or randomly within a specific range according to the hypothetical scenarios.

Challenge #4:
Capacity planning across edge computing layer is challenging as it depends on various configuration parameters including data volume, data velocity, upstream/downstream network etc.

To solve this issue, IoTSim utilizes the features of one of the most popular cloud simulators, Cloudsim. Similar to how Cloudsim allows for configuration of parameters such as network latency, data volume and velocity. Cloudsim implements two schedulers for resource allocation: VM scheduler allocates VMs to host while the cloudlet scheduler allocates different tasks to VMs. IoTSim's EdgeDataCenter class is responsible for establishing connection between edge and IoT devices based on the given IoT protocol (e.g. CoAP) along with performing edge *resource provisioning*, scheduling policies, and monitoring edge processing. IoTSim builds on the same to provide flexible capacity planning across the edge computing layer.

Challenge #5:
The communication between IoT and edge devices is very different from cloud datacenter communication, which are generally based on wired and/or wireless protocol.

The EdgeBroker class acts on behalf of users in terms of establishing connection with edge and IoT devices, negotiating with resources, submitting IoT and edge requests, and receiving results. This class is a users' proxy, in which it generates users' requests in accordance to their prescribed requirements. It has a range of duties to perform, such as submitting edge and MEL provisioning requests to edge datacenter, requesting IoT devices to generate and send data to their respective edge devices, and receiving final processing results from MEL.

The connectivity between IoT and edge nodes is often diverse in nature. Connection types include Wi-Fi, 3G, Bluetooth, LoRa, Zigbee, short and

long-range radio. Each connection is detailed with the signal type, capacity, power model and traffic management protocol. Parameters of any connection type could be obtained from models developed in the literature. For an IoT service, tracking the signal type for possible signal loss scenarios is important when evaluating systems design.

Challenge #6:
Mobility is another important parameter of IoT devices as sensors embedded to many physical devices are moving.

Mobility of the IoT devices is a problem unique to the IoT environment. Edge devices tend to have a small range where communication is possible and in scenarios where the IoT devices are moving e.g. FitBit or Car sensors, there must be a way to model signal handoff between Edge devices. This is further complicated by the heterogeneous natures of the Edge devices.

IoTSim allows us to configure whether a sensor is movable or not, and if so, what its velocity would be as well as the range of its movement. The movement policy of IoT devices is directed by the MovingPolicy class; it can be extended with new moving policies according to users' requirements (e.g. velocity and location of cars' sensors).

A mobile sensor extends the sensor node with functionalities of changing location and altitude. Location and altitude are configured using x, y and z parameters respectively. These could be read from a JSONfile, changed randomly or systematically following a specific pattern.

Challenge #7:
Dynamicity of IoT environment leads to addition and removal of IoT and edge devices very frequently. Modeling the scalability of IoT devices with heterogeneous features at a fast rate is very challenging

When compared to OMNet++, TOSSIM, iCanCloud, GreenCloud, CloudSim and many other simulators, IoTSim Edge solves almost all the problems that other simulators do not. And it would be best if it were developed over CloudSim (as it's the most stable option) to give IoTSim Edge the upper hand of modelling scalability at a fast rate.
An IoT environment is a very dynamic environment, with devices entering and exiting the network multiple times. This may be caused by numerous factors e.g. device failure, network link failure. And again, the heterogeneous nature of the IoT devices adds a layer of complexity to modeling this situation.

Challenge #8:
The simulator must allow users to customize and extend the framework based on their requirements.

Since IoT is an emerging environment, it is very important that a simulator allows the users to customize and extend the framework as required. IoTSim builds on Cloudsim in a clean and intuitive manner, allowing for easy modification in case of custom applications.

For example, in our demo application, we modified the Temperature sensor to generate a random number along with every EdgeLet it sent. Meanwhile the EdgeDataCenter is similarly modified to examine the EdgeLet and take actions depending on the value, which in this case, was logging of extreme values.