Experiment 6
Implement Identity management in cloud environment using OMNET++

Aim: To Implement Identity management in cloud environment using OMNET++.

Software Required: OMNeT++ 6.0.1

Operating System Required: Windows OS

Algorithm:

To build a "network" that consists of two nodes, where one of the nodes will create a packet and the two nodes will keep passing the same packet back and forth.

STEP 1: Start the OMNeT++ IDE by typing omnetpp in your terminal.

In the IDE, choose *New -> OMNeT++ Project* from the menu.

STEP 2: A wizard dialog will appear. Enter 'tictoc' as project name, choose *Empty project* when asked about the initial content of the project, then click *Finish*. An empty project will be created.

The project will hold all files that belong to our simulation

STEP 3: Adding the NED file

To add the file to the project,

Switch into *Source* mode, and enter the following:

```
#include <omnetpp.h>

using namespace omnetpp;

namespace fifo {

/**
 * Generates messages or jobs; see NED file for more info.
 */
class Source : public cSimpleModule
{
  private:
    cMessage *sendMessageEvent = nullptr;
```

```
  public:
    virtual ~Source();

  protected:
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};

Define_Module(Source);

Source::~Source()
{
    cancelAndDelete(sendMessageEvent);
}

void Source::initialize()
{
    sendMessageEvent = new cMessage("sendMessageEvent");
    scheduleAt(simTime(), sendMessageEvent);
}

void Source::handleMessage(cMessage *msg)
{
    ASSERT(msg == sendMessageEvent);

    cMessage *job = new cMessage("job");
    send(job, "out");

    scheduleAt(simTime()+par("interarrivalTime").doubleValue(), sendMessageEvent);
}

}; //namespace
```

STEP 4: Adding the NED file

To implement the functionality of the Txc1 simple module in C++.

STEP 5: Adding the omnetpp.ini file:
To be able to run the simulation, we need to create an omnetpp.ini file. omnetpp.ini tells the simulation program which network you want to simulate. We are now done with creating the model, and ready to compile and run it.

```
[Fifo1]
description = "low job arrival rate"
network = SingleQueue
**.source.interarrivalTime = exponential(0.2s)
**.fifo.serviceTime = 0.1s

[Fifo2]
description = "high job arrival rate"
network = SingleQueue
**.source.interarrivalTime = exponential(0.1s)
**.fifo.serviceTime = 0.1s
```

Explanation

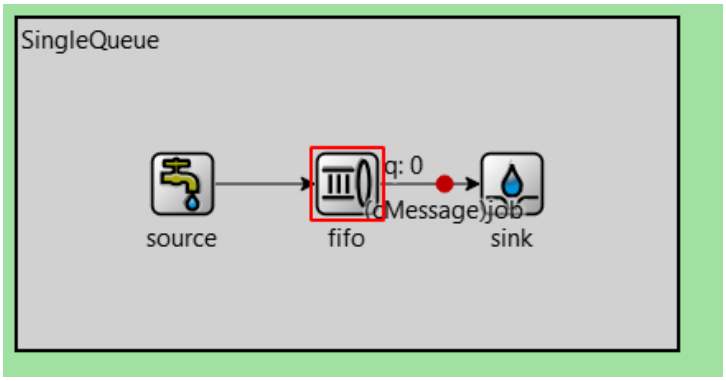
FIFO

In computing and in systems theory, FIFO is an acronym for first in, first out (the first in is the first out), a method for organizing the manipulation of a data structure (often, specifically a data buffer) where the oldest (first) entry, or "head" of the queue, is processed first. Such processing is analogous to servicing people in a queue area on a first-come, first-served (FCFS) basis, i.e. in the same sequence in which they arrive at the queue's tail.

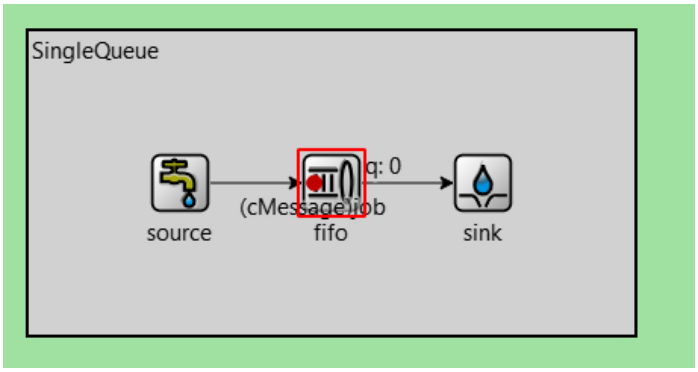
FCFS is also the jargon term for the FIFO operating system scheduling algorithm, which gives every process central processing unit (CPU) time in the order in which it is demanded.[1] FIFO's opposite is LIFO, last-in-first-out, where the youngest entry or "top of the stack" is processed first.[2] A priority queue is neither FIFO or LIFO but may adopt similar behaviour temporarily or by default. Queueing theory encompasses these methods for processing data structures, as well as interactions between strict-FIFO queues.

Output:

FIFO- Low Job Arrival Rate



FIFO- High Job Arrival Rate



Result:

Thus, the formation of Identity management in Cloud Environment using OMNET++ was implemented and executed successfully.