Experiment 3
Implement secure efficient data routing in IoT using OMNET++

Aim: To Implement secure efficient data routing in IoT 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.

Let's call the nodes tic and toc.

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: Routing.cc

```
#ifdef _MSC_VER
#pragma warning(disable:4786)
#endif

#include <map>
#include <omnetpp.h>
#include "Packet_m.h"

using namespace omnetpp;
```

```
class Routing : public cSimpleModule
{
  private:
    int myAddress;

    typedef std::map<int, int> RoutingTable;  // destaddr -> gateindex
    RoutingTable rtable;

    simsignal_t dropSignal;
    simsignal_t outputIfSignal;

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

Define_Module(Routing);

void Routing::initialize()
{
    myAddress = getParentModule()->par("address");

    dropSignal = registerSignal("drop");
    outputIfSignal = registerSignal("outputIf");

    cTopology *topo = new cTopology("topo");

    std::vector<std::string> nedTypes;
    nedTypes.push_back(getParentModule()->getNedTypeName());
    topo->extractByNedTypeName(nedTypes);
    EV << "cTopology found " << topo->getNumNodes() << " nodes\n";

    cTopology::Node *thisNode = topo->getNodeFor(getParentModule());

    // find and store next hops
    for (int i = 0; i < topo->getNumNodes(); i++) {
        if (topo->getNode(i) == thisNode)
            continue;  // skip ourselves
        topo->calculateUnweightedSingleShortestPathsTo(topo-
>getNode(i));

        if (thisNode->getNumPaths() == 0)
            continue;  // not connected

        cGate *parentModuleGate = thisNode->getPath(0)->getLocalGate();
        int gateIndex = parentModuleGate->getIndex();
        int address = topo->getNode(i)->getModule()->par("address");
        rtable[address] = gateIndex;
        EV << "  towards address " << address << " gateIndex is " <<
gateIndex << endl;
    }
    delete topo;
}

void Routing::handleMessage(cMessage *msg)
{
```

```
    Packet *pk = check_and_cast<Packet *>(msg);
    int destAddr = pk->getDestAddr();

    if (destAddr == myAddress) {
        EV << "local delivery of packet " << pk->getName() << endl;
        send(pk, "localOut");
        emit(outputIfSignal, -1);  // -1: local
        return;
    }

    RoutingTable::iterator it = rtable.find(destAddr);
    if (it == rtable.end()) {
        EV << "address " << destAddr << " unreachable, discarding packet
" << pk->getName() << endl;
        emit(dropSignal, (intval_t)pk->getByteLength());
        delete pk;
        return;
    }

    int outGateIndex = (*it).second;
    EV << "forwarding packet " << pk->getName() << " on gate index " <<
outGateIndex << endl;
    pk->setHopCount(pk->getHopCount()+1);
    emit(outputIfSignal, outGateIndex);

    send(pk, "out", outGateIndex);
}
```

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.

```
[RandomGraph]
network = networks.RandomGraph
**.destAddresses = "0 2"

[Mesh]
network = networks.Mesh
**.destAddresses = "0 18 52"
```

Explanation:

Random Routing

Random routing is also a simple and robust technique or routing and also has some improvements over flooding − It results in a controlled traffic load.

The outgoing link is picked in a random fashion after eliminating the incoming link. If the likely possibilities of choosing any link are the same, it amounts to picking the outgoing link in a round-robin style. This is to designate probabilities to every outgoing link and to choose a link based on those probabilities. The probabilities can be determined based on the data rate.

Where,

Pi=probability of choosing link i

Ri = data rate of i^{th} link.

Random routing do not need any information about the network, and in the case of the random routing, the actual route may not be either the minimum-hop or the least-cost route.

Mesh Routing

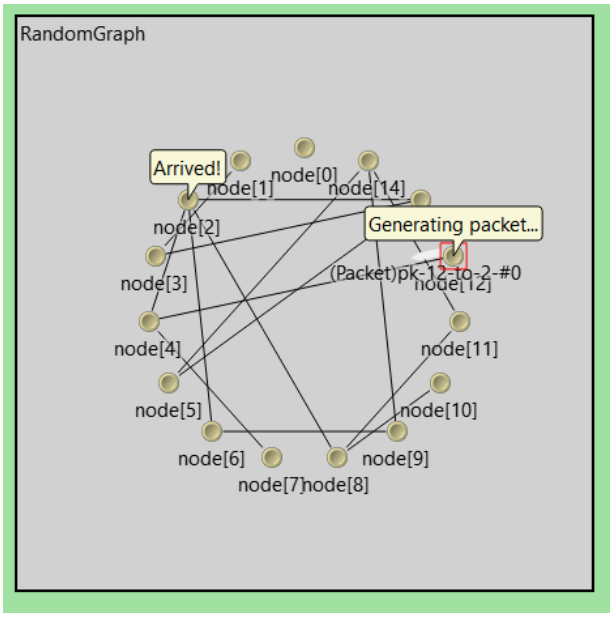
A mesh network is defined as a type of local area network (LAN) topology, where multiple devices or nodes are connected in a non-hierarchical manner, so that they can coexist, cooperate, and provide comprehensive network coverage to a wider area than what is possible by a single router.

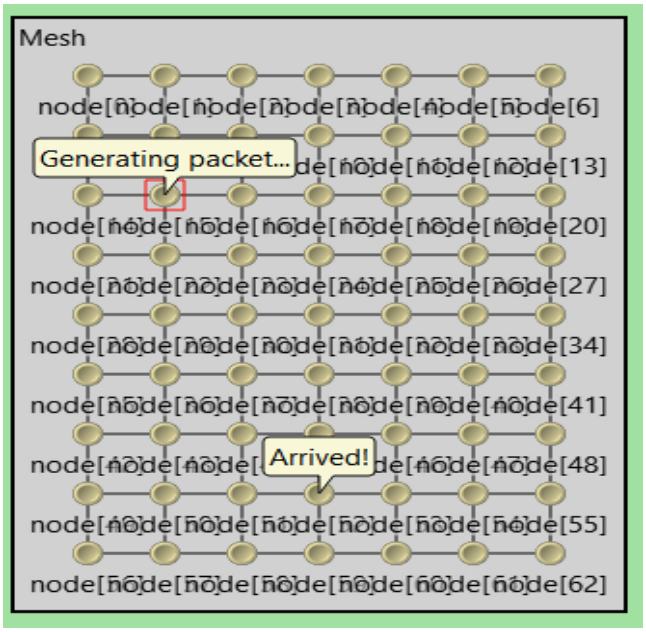
A mesh network is a wireless system consisting of multiple computers connected by a network. Each computer in the network sends its own signals and relays information from other computers. Every node on the mesh network is connected to another node via a dedicated link. This connection allows information to travel from node to node without delays or failures. Mesh networks are also called “self-configuring” networks because a new node automatically becomes part of the network’s existing structure.

A mesh network allows for additional coverage around your home by using several smaller routers. The central node plugs into a modem from your internet provider, and other devices can connect to it. A mesh network allows for the

management of each device through a mobile application. This app will also allow you to prioritize devices in the mesh network, monitor data speeds, and manage network issues. You can use mobile apps to manage your network from anywhere and anytime and control the network from your smartphone.

Output:





Result:

Thus, the formation of secure efficient data routing in IoT using OMNET++ was implemented and executed successfully.