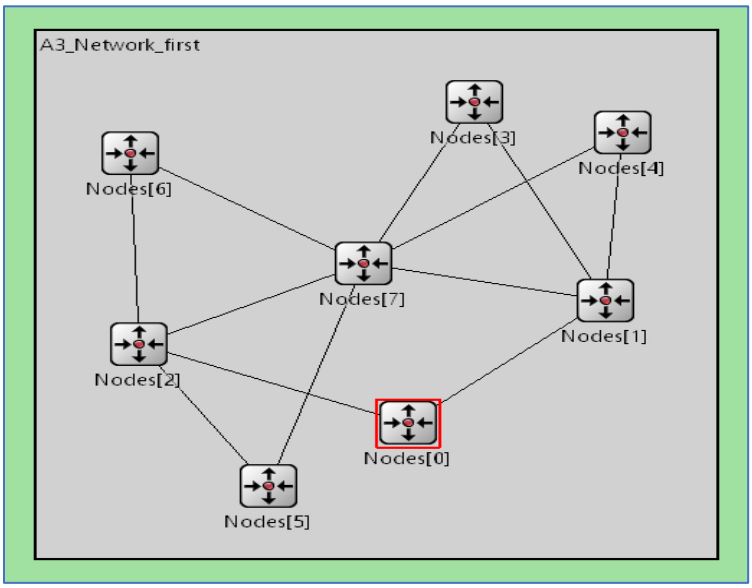
“Random Network Simulation Using OMNeT++”

The network has 8 nodes and which are connected in the following manner



For Complete Project

GitHub

.NED File(Network Description):

```
simple Node
{
    parameters:
        @display("i=block/routing");
    gates:
        inout port[];  // declare two way connections
}


network A3_Network_first
{
    //we can customize the channel based on our requirements
    types:
        channel Channel extends ned.DelayChannel {
            delay = 100ms;
        }
    submodules:
        Nodes[8]: Node;
    //creating a random network connection
    connections:
        Nodes[0].port++ <--> Channel <--> Nodes[1].port++;
        Nodes[0].port++ <--> Channel <--> Nodes[2].port++;
        Nodes[1].port++ <--> Channel <--> Nodes[3].port++;
        Nodes[1].port++ <--> Channel <--> Nodes[4].port++;
        Nodes[1].port++ <--> Channel <--> Nodes[7].port++;
        Nodes[3].port++ <--> Channel <--> Nodes[7].port++;
        Nodes[4].port++ <--> Channel <--> Nodes[7].port++;
        Nodes[2].port++ <--> Channel <--> Nodes[7].port++;
        Nodes[2].port++ <--> Channel <--> Nodes[5].port++;
        Nodes[2].port++ <--> Channel <--> Nodes[6].port++;
```

```
        Nodes[5].port++ <--> Channel <--> Nodes[7].port++;
        Nodes[6].port++ <--> Channel <--> Nodes[7].port++;
}
```

Message Definition:

```
message A3_First_MSG
{
    int source;
    int destination;
    int hopCount = 0;
}
```

.CC File (Functionality of network):

```
#include <stdio.h>
#include <string.h>
#include <omnetpp.h>
#include "a3firstmsg_m.h"

using namespace omnetpp;

class Node : public cSimpleModule
{
  private:
    long numSent;
    long numReceived;
    cHistogram hopCountStats;
    cOutVector hopCountVector;

  protected:
    virtual A3_First_MSG *generateMessage();
    virtual void forwardMessage(A3_First_MSG *msg);
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;

    // The finish() function is called by OMNeT++ at the end of the
simulation:
    virtual void finish() override;
};

Define_Module(Node);

void Node::initialize()
{
    // Initialize variables
    numSent = 0;
    numReceived = 0;
    WATCH(numSent);
    WATCH(numReceived);
```

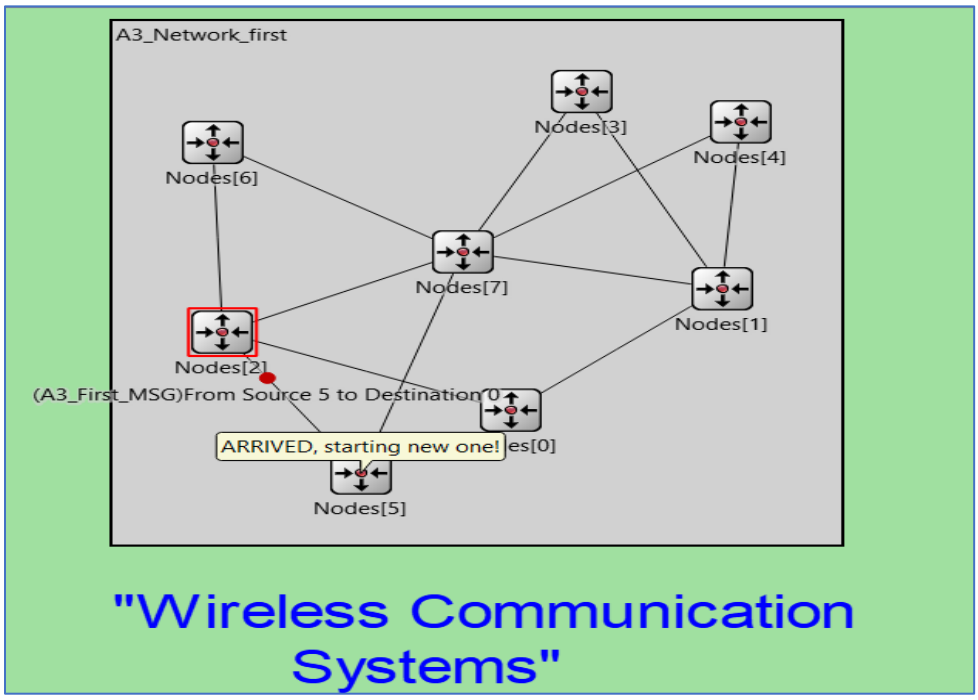
```
    hopCountStats.setName("hopCountStats");
    hopCountVector.setName("HopCount");

    // Module 0 sends the first message
    if (getIndex() == 0) {
        // Add static text to the canvas
        cTextFigure *text = new cTextFigure("StaticText");
       // The text to display
        text->setText("\"Wireless Communication \n \t Systems\" ");
        text->setPosition(cFigure::Point(0,400));     // Position on the
canvas
        text->setColor(cFigure::BLUE);                    // Text color
        text->setFont(cFigure::Font("Arial", 20));        // Font style
and size
        getParentModule()->getCanvas()->addFigure(text);

        // Boot the process scheduling the initial message as a self-
message.
        A3_First_MSG *msg = generateMessage();
        scheduleAt(0.0, msg);
    }
}

void Node::handleMessage(cMessage *msg)
{
    A3_First_MSG *ttmsg = check_and_cast<A3_First_MSG *>(msg);

    if (ttmsg->getDestination() == getIndex()) {
        // Message arrived
        int hopcount = ttmsg->getHopCount();
        EV << "Message " << ttmsg << " arrived after " << hopcount << "
hops.\n";
        bubble("ARRIVED, starting new one!");

        // update statistics.
        numReceived++;
        hopCountVector.record(hopcount);
        hopCountStats.collect(hopcount);

        delete ttmsg;

        // Generate another one.
        EV << "Generating another message: ";
        A3_First_MSG *newmsg = generateMessage();
        EV << newmsg << endl;
        forwardMessage(newmsg);
        numSent++;
    }
    else {
        // We need to forward the message.
        forwardMessage(ttmsg);
    }
```

```
}

A3_First_MSG *Node::generateMessage()
{
    // Produce source and destination addresses.
    int src = getIndex();
    int n = getVectorSize();
    int dest = intuniform(0, n-2);
    if (dest >= src)
        dest++;

    char msgname[20];
    sprintf(msgname, "From Source %d to Destination %d", src, dest);

    // Create message object and set source and destination field.
    A3_First_MSG *msg = new A3_First_MSG(msgname);
    msg->setSource(src);
    msg->setDestination(dest);
    return msg;
}

void Node::forwardMessage(A3_First_MSG *msg)
{
    // Increment hop count.
    msg->setHopCount(msg->getHopCount()+1);

    // Same routing as before: random gate.
    int n = gateSize("port");
    int k = intuniform(0, n-1);

    EV << "Forwarding message " << msg << " on port[" << k << "]\n";
    send(msg, "port$o", k);
}

void Node::finish()
{
    // This function is called by OMNeT++ at the end of the simulation.
    EV << "Sent:     " << numSent << endl;
    EV << "Received: " << numReceived << endl;
    EV << "Hop count, max:    " << hopCountStats.getMax() << endl;

    recordScalar("#sent", numSent);
    recordScalar("#received", numReceived);

    hopCountStats.recordAs("hop count");
}
```

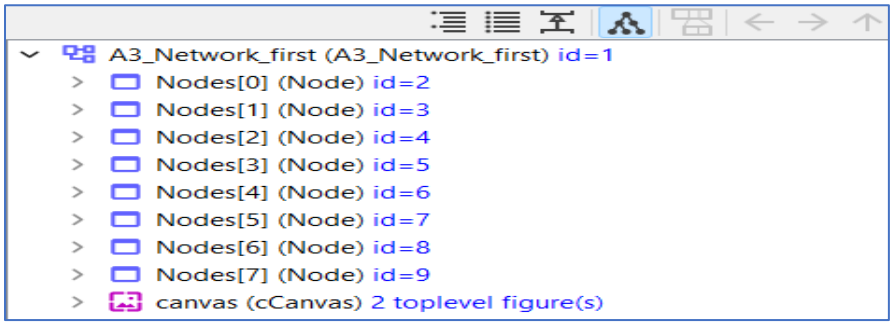
.ini File:

```
[A3_Network_first]
network = A3_Network_first
record-eventlog = true
```

Simulation:



Children View:



Message Logs:

```
INFO: Forwarding message (A3_First_MSG)From Source 0 to Destination 3  on port[1]
** Event #20  t=1.9  A3_Network_first.Nodes[7] (Node, id=9)  on From Source 0 to Destination 3 (A3_First_MSG, id=2)
INFO: Forwarding message (A3_First_MSG)From Source 0 to Destination 3  on port[0]
** Event #21  t=2  A3_Network_first.Nodes[1] (Node, id=3)  on From Source 0 to Destination 3 (A3_First_MSG, id=2)
INFO: Forwarding message (A3_First_MSG)From Source 0 to Destination 3  on port[1]
** Event #22  t=2.1  A3_Network_first.Nodes[3] (Node, id=5)  on From Source 0 to Destination 3 (A3_First_MSG, id=2)
INFO: Message (A3_First_MSG)From Source 0 to Destination 3  arrived after 5 hops.
INFO: Generating another message: (A3_First_MSG)From Source 3 to Destination 6 (new msg)
INFO: Forwarding message (A3_First_MSG)From Source 3 to Destination 6 (new msg) on port[1]
** Event #23  t=2.2  A3_Network_first.Nodes[7] (Node, id=9)  on From Source 3 to Destination 6 (A3_First_MSG, id=3)
INFO: Forwarding message (A3_First_MSG)From Source 3 to Destination 6  on port[5]
** Event #24  t=2.3  A3_Network_first.Nodes[6] (Node, id=8)  on From Source 3 to Destination 6 (A3_First_MSG, id=3)
INFO: Message (A3_First_MSG)From Source 3 to Destination 6  arrived after 2 hops.
INFO: Generating another message: (A3_First_MSG)From Source 6 to Destination 0 (new msg)
INFO: Forwarding message (A3_First_MSG)From Source 6 to Destination 0 (new msg) on port[1]
** Event #25  t=2.4  A3_Network_first.Nodes[7] (Node, id=9)  on From Source 6 to Destination 0 (A3_First_MSG, id=4)
INFO: Forwarding message (A3_First_MSG)From Source 6 to Destination 0  on port[4]
** Event #26  t=2.5  A3_Network_first.Nodes[5] (Node, id=7)  on From Source 6 to Destination 0 (A3_First_MSG, id=4)
INFO: Forwarding message (A3_First_MSG)From Source 6 to Destination 0  on port[1]
** Event #27  t=2.6  A3_Network_first.Nodes[7] (Node, id=9)  on From Source 6 to Destination 0 (A3_First_MSG, id=4)
INFO: Forwarding message (A3_First_MSG)From Source 6 to Destination 0  on port[0]
```

The simulation was run for 355 events which resulted in the following data, this data was extracted from Result Analysis Tool

Tabular form:

	Module	Count	Mean	StdDev	Variance	Max
0	Nodes[0]	6	14.000000	14.546477	211.600000	38
1	Nodes[1]	3	4.333333	4.932883	24.333333	10
2	Nodes[2]	4	9.250000	6.849574	46.916667	19
3	Nodes[3]	4	13.500000	6.027714	36.333333	19
4	Nodes[4]	5	15.200000	14.754660	217.700000	37
5	Nodes[5]	2	11.000000	4.242641	18.000000	14
6	Nodes[6]	3	13.000000	10.535654	111.000000	23
7	Nodes[7]	2	6.000000	7.071068	50.000000	11

Visualization:

This visualization is obtained with the help of python