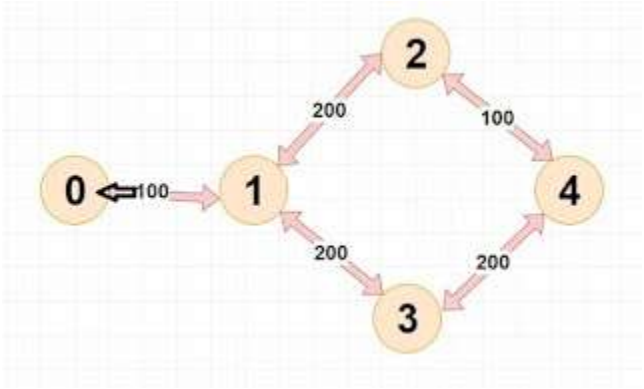
The LNM Institute of Information Technology, Jaipur

Computer Networks Lab

Lab Assignment 4 Solution

Tasks 1: Network Design

1. Create a network design (name "forwardingNetwork") for the given network topology



Tasks 2: Behavioural Design

1. The Source node sends packets to the destination node through the above network. (source and destination information are defined in omnettpp.ini)

2. Each node in the network will either accept the packet (if the packet belongs to the same node) or forward the packet with the help of forwarding table implemented on each node (as defined in the above diagram).

   (Hint: use map (http://www.cplusplus.com/reference/map/map/at/) data-structure to define the forwarding table)

3. Display Delay of the packet at the destination node.

Hint:

1. Use array for declaring gates

2. Use "connections allowunconnected" in forwardingNetwork.ned

3. Implement forwarding table using a map data structure.

4. Node.h:

a. Include “#include <map>”

b. “using namespace std;”

c. Declare map: “map<int, int> LUT”

```
simple Node

{

    parameters:

        int address;

        int source_address;

        int dest_address;



    gates:

        input gIn[3];

        output gOut[3];

}



network ForwardingNetwork

{

    parameters:

        int source_address;

        int dest_address;



        @display("bgb=400,200");

    submodules:
```

```
        N[5]: Node {
            address = index;
            source_address=source_address;
            dest_address=dest_address;
        }

    connections allowunconnected:
        N[0].gIn[0] <-- {  delay = 100ms; } <-- N[1].gOut[0];
        N[0].gOut[0] --> {  delay = 100ms; } --> N[1].gIn[0];

        N[1].gIn[1] <-- {  delay = 200ms; } <-- N[2].gOut[0];
        N[1].gOut[1] --> {  delay = 200ms; } --> N[2].gIn[0];

        N[2].gIn[1] <-- {  delay = 100ms; } <-- N[4].gOut[0];
        N[2].gOut[1] --> {  delay = 100ms; } --> N[4].gIn[0];

        N[1].gIn[2] <-- {  delay = 200ms; } <-- N[3].gOut[0];
        N[1].gOut[2] --> {  delay = 200ms; } --> N[3].gIn[0];

        N[3].gIn[1] <-- {  delay = 200ms; } <-- N[4].gOut[1];
        N[3].gOut[1] --> {  delay = 200ms; } --> N[4].gIn[1];
}

packet N_PDU {
    int Address;
```

```
    int Source;
    int Dest;
    simtime_t Start;
}


#ifndef __FOURTH_CN_LAB_NODE_H_
#define __FOURTH_CN_LAB_NODE_H_


#include <omnetpp.h>
#include <map>
#include <N_PDU_m.h>
using namespace omnetpp;
using namespace std;
class Node : public cSimpleModule
{
  protected:
        int address;
        int source_address;
        int dest_address;
        cGate* in;
        cGate* out;
        map<int,int> LUT;
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};
```

```
#endif

#include "node.h"

Define_Module(Node);

void Node::initialize()
{

        dest_address = par("dest_address");
        source_address = par("source_address");
        address = par("address");
        if (address==0){
            LUT = {{1,0},{2,0},{3,0},{4,0}};
        }
        else if (address==1){
            LUT = {{0,0},{2,1},{3,2},{4,1}};
        }
        else if (address==2){
            LUT = {{0,0},{1,0},{3,1},{4,1}};
             }
        else if (address==3){
            LUT = {{0,0},{1,0},{2,1},{4,1}};
        }
```

```
        else if (address==4){
            LUT = {{0,0},{1,0},{2,0},{3,1}};
        }

        if(address==source_address){
            cMessage* event = new cMessage();
            scheduleAt(0, event);
        }
}

void Node::handleMessage(cMessage *msg)
{
    // TODO - Generated method body
    if (msg->isSelfMessage()){
            N_PDU* data = new N_PDU();
            data->setStart(simTime());
            send(data, "gOut", LUT.at(dest_address));
        }
        else{
            if(address==dest_address){
                N_PDU* data = check_and_cast<N_PDU*>(msg);
                EV<<"Delay"<<data->getArrivalTime() - data->getStart();
            }
            else
            {
```

```
                N_PDU* data = check_and_cast<N_PDU*>(msg);

                send(data, "gOut", LUT.at(dest_address));

            }

        }

}
```