

LAB 04A

1. Giới hạn số lần trao đổi tin nhắn từ một host:

lab4a.ned

```
simple SimpleHost{
    parameters:
        int limit;
    gates:
        input in;
        output out;
}

network Network{
    submodules:
        host[2]: SimpleHost;
    connections:
        host[0].out --> {delay = 1000ms; } --> host[1].in;
        host[1].out --> {delay = 1000ms; } --> host[0].in;
}
```

SimpleHost.cc

```
#include <omnetpp.h>
#include <string.h>

using namespace omnetpp;

class SimpleHost: public cSimpleModule {
private:
    int counter = 0;
    int limit = 0;
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(SimpleHost);

void SimpleHost::initialize() {
    limit = par("limit").intValue();
    WATCH(counter);
    if (strcmp(getFullName(), "host[0]") == 0) {
        cMessage *msg = new cMessage("hello");
        send(msg, "out");
    }
}

void SimpleHost::handleMessage(cMessage *msg) {
    counter++;
    if (counter < limit) {
        send(msg, "out");
    }
    else {
        delete msg;
    }
}
```

omnet.ini

```
[General]
network = Network
Network.host[*].limit = 5
```

kết quả

The screenshot displays the OMNeT++ simulation environment. On the left, the 'Network (Network) id=1' is expanded, showing two hosts: 'host[0] (SimpleHost) id=2' and 'host[1] (SimpleHost) id=3'. Both hosts have a 'limit (cPar) 5' and a 'counter (int) 4'. The network diagram on the right shows a ring topology with two hosts, 'host[0]' and 'host[1]', connected by a message labeled '(cMessage)hello'. A 'Confirm' dialog box is open in the center, stating 'No more events, simulation completed -- at t=9s, event #9' with an 'OK' button. At the bottom, the event log shows the sequence of events from t=5 to t=9, all being 'hello' messages between the hosts.

```
> Network (Network) id=1
> scheduled-events (cEventHeap)

v Network (Network) id=1
  v host[0] (SimpleHost) id=2
    in (cGate) <-- host[1].out, (ned.DelayChannel)
    out (cGate) --> host[1].in, (ned.DelayChannel)
    limit (cPar) 5
    counter (int) 4
  v host[1] (SimpleHost) id=3
    in (cGate) <-- host[0].out, (ned.DelayChannel)
    out (cGate) --> host[0].in, (ned.DelayChannel)
    limit (cPar) 5
    counter (int) 4

Confirm
No more events, simulation completed -- at t=9s, event #9
OK

** Event #5 t=5 Network.host[1] (SimpleHost, id=3) on hello (omnetpp::cMessage, id=0)
** Event #6 t=6 Network.host[0] (SimpleHost, id=2) on hello (omnetpp::cMessage, id=0)
** Event #7 t=7 Network.host[1] (SimpleHost, id=3) on hello (omnetpp::cMessage, id=0)
** Event #8 t=8 Network.host[0] (SimpleHost, id=2) on hello (omnetpp::cMessage, id=0)
** Event #9 t=9 Network.host[1] (SimpleHost, id=3) on hello (omnetpp::cMessage, id=0)
<!-- No more events, simulation completed -- at t=9s, event #9
```

2. Tạo ra mạng dạng vòng (ring) với số lượng host tùy ý:

SimpleMessage.msg

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

lab4a2.ned

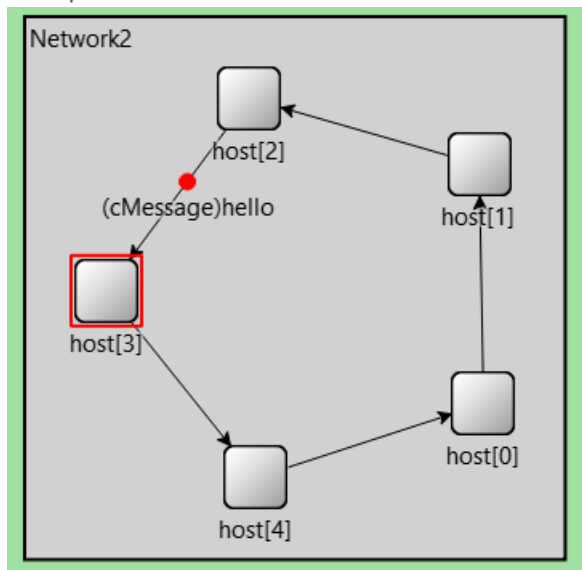
```
network Network2{
    parameters:
        int numHosts;
    submodules:
        host[numHosts]: SimpleHost;
    connections:
```

```

    host[numHosts - 1].out --> {delay = 1000ms; } --> host[0].in;
    for i = 0..numHosts-2{
        host[i].out --> {delay = 1000ms;} --> host[i+1].in;
    }
}

```

kết quả:



3. Thiết lập việc truyền tin trong mạng ring:

lab4a3.ned

```

simple SimpleHost3
{
    parameters:
        int limit;
    gates:
        input in;
        output out;
}

network Network3
{
    parameters:
        int numHosts;
    submodules:
        host[numHosts]: SimpleHost3;
    connections:
        host[numHosts - 1].out --> { delay = 1000ms; } --> host[0].in;
        for i=0..numHosts-2 {
            host[i].out --> { delay = 1000ms; } --> host[i+1].in;
        }
}

```

SimpleHost3.cc

```

#include <omnetpp.h>
#include <string.h>
#include <SimpleMessage_m.h>

```

```

using namespace omnetpp;

class SimpleHost3: public cSimpleModule {
private:
    int counter = 0;
    int limit;
    int numHosts;
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual SimpleMessage* generateMessage();
    virtual void forwardMessage(SimpleMessage *msg);
};

Define_Module(SimpleHost3);

void SimpleHost3::initialize() {
    limit = par("limit").intValue();
    numHosts = getParentModule()->par("numHosts").intValue();
    WATCH(counter);
    if (strcmp(getFullName(), "host[0]") == 0) {
        scheduleAt(0, new cMessage());
    }
}

void SimpleHost3::handleMessage(cMessage *_msg) {

    counter++;

    if (counter == limit && getIndex() == numHosts - 1){
        delete _msg;
        return;
    }

    if (getIndex() == 0) {
        delete _msg;
        send(generateMessage(), "out");
        return;
    }

    SimpleMessage *msg = check_and_cast<SimpleMessage*>(_msg);

    if (getIndex() == numHosts - 1) {
        EV <<"HOP COUNT = " << msg->getHopCount();
        delete msg;
        send(new cMessage(), "out");
    }
    else{
        forwardMessage(msg);
    }
}

SimpleMessage* SimpleHost3::generateMessage() {
    SimpleMessage *msg = new SimpleMessage();
    msg->setSource(0);
    msg->setDestination(numHosts - 1);
    msg->setHopCount(0);
    return msg;
}

```

```

void SimpleHost3::forwardMessage(SimpleMessage *msg) {
    msg->setHopCount(msg->getHopCount() + 1);
    send((cMessage*)msg, "out");
}

```

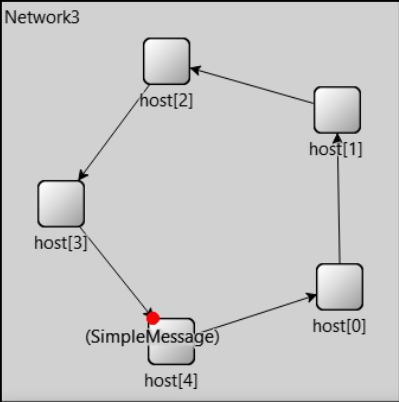
kết quả

> Network3 (Network3) id=1

> scheduled-events (cEventHeap)

Network3 (Network3) id=1

- numHosts (cPar) 5
- host[0] (SimpleHost3) id=2
- host[1] (SimpleHost3) id=3
- host[2] (SimpleHost3) id=4
- host[3] (SimpleHost3) id=5
- host[4] (SimpleHost3) id=6



Confirm

No more events, simulation completed -- at t=24s, event #25

```

** Event #21 t=20 Network3.host[0] (SimpleHost3, id=2) on (omnetpp::cMessage, id=8)
** Event #22 t=21 Network3.host[1] (SimpleHost3, id=3) on (SimpleMessage, id=9)
** Event #23 t=22 Network3.host[2] (SimpleHost3, id=4) on (SimpleMessage, id=9)
** Event #24 t=23 Network3.host[3] (SimpleHost3, id=5) on (SimpleMessage, id=9)
** Event #25 t=24 Network3.host[4] (SimpleHost3, id=6) on (SimpleMessage, id=9)
<!-- No more events, simulation completed -- at t=24s, event #25

```