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;
}
```

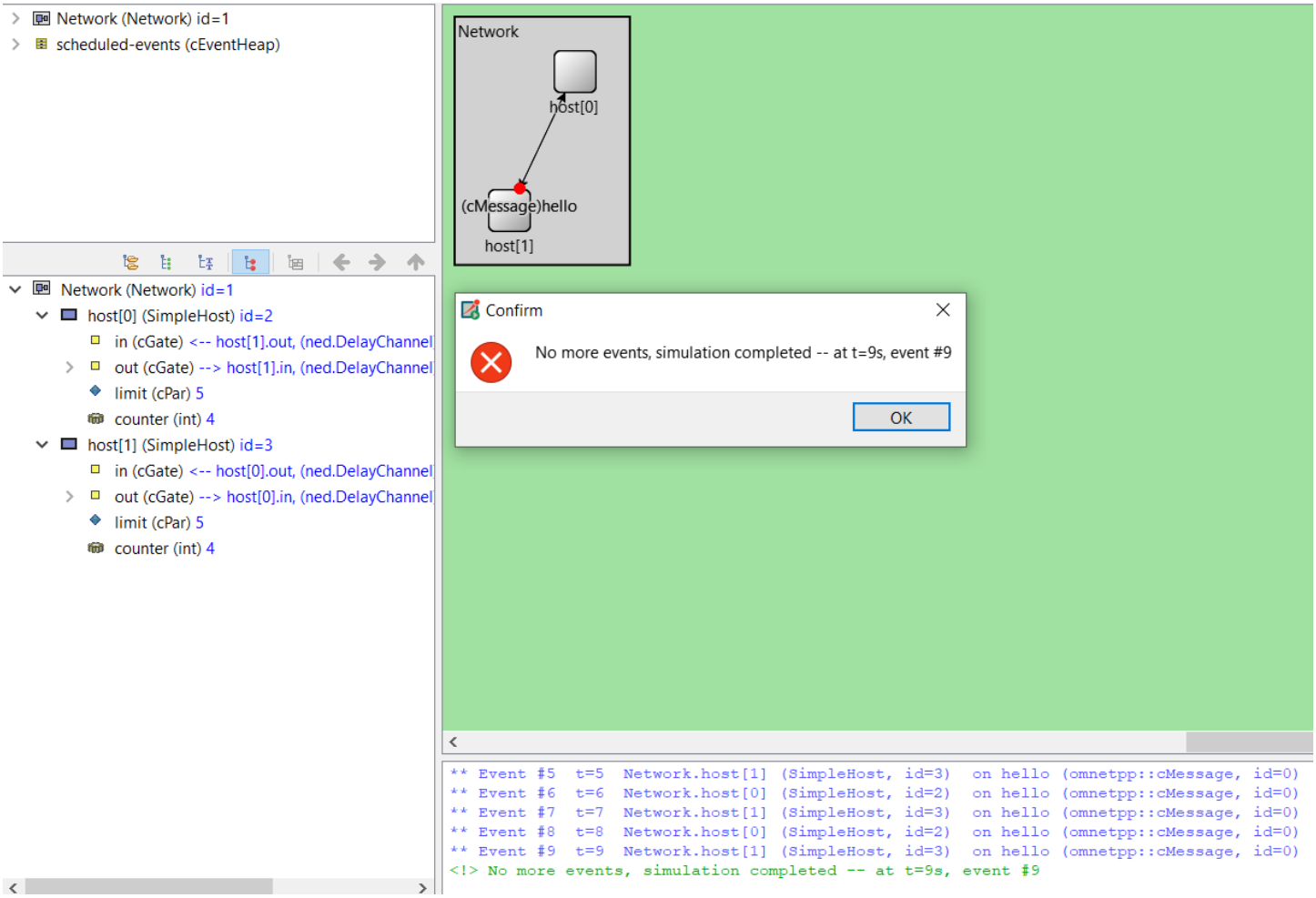
SimpleHost.cc

```
#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



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

SimpleMessage.msg

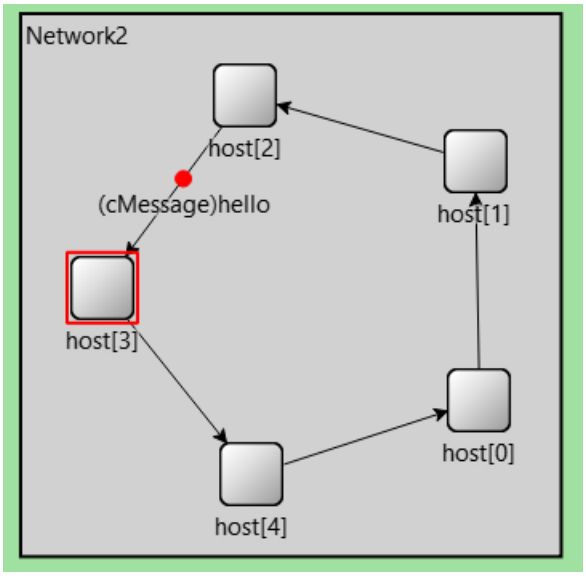
```
message SimpleMessage {
        int source;
        int destination;
        int hopCount = 0;
}
```

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