I pledge my honor that I have abided by the stevens honor system

My\_ARQ.ned

**network** My\_ARQ2

{

**@display**("BGB=214,190");

**submodules**:

source: ARQ\_source {

**@display**("p=31,99");

}

destination: DestARQ {

**@display**("p=226,129;is=vl");

}

**connections**:

source.gate\_out **-->** { ber = 1e-3; delay = 100ms; } **-->** destination.gate\_in;

destination.gate\_out **-->** { delay = 100ms; } **-->** source.gate\_in;

}

ARQ\_source.ned

**simple** ARQ\_source

{

**gates**:

**input** gate\_in;

**output** gate\_out;

}

ARQ\_source.h

**#ifndef** \_\_MY\_ARQ2\_ARQ\_SOURCE\_H\_

**#define** \_\_MY\_ARQ2\_ARQ\_SOURCE\_H\_

**#include** <omnetpp.h>

**using** **namespace** omnetpp;

/\*\*

\* **TODO** - Generated class

\*/

**class** ARQ\_source : **public** cSimpleModule

{

**private**:

simtime\_t timeout;

cMessage \*timeoutEvent;

cPacket \*pkt;

**public**:

**ARQ\_source**();

**virtual** **~ARQ\_source**();

**protected**:

**virtual** **void** **initialize**();

**virtual** **void** **handleMessage**(cMessage \*msg);

};

**#endif**

ARQ\_source.cc

**#include** "ARQ\_source.h"

Define\_Module(ARQ\_source);

**ARQ\_source::ARQ\_source**(){

timeoutEvent = **nullptr**;

}

**ARQ\_source::~ARQ\_source**(){

**cancelAndDelete**(timeoutEvent);

}

**void** **ARQ\_source::initialize**()

{

timeout=1.0;

timeoutEvent= **new** cMessage("timeoutEvent");

EV<<"Sending initial packet";

pkt = **new** cPacket("packet");

pkt ->**setBitLength**(100);

cPacket \*pkt\_copy = pkt ->dup();

send(pkt\_copy, "gate\_out");

**scheduleAt**(simTime()+timeout, timeoutEvent);

}

**void** **ARQ\_source::handleMessage**(cMessage \*msg)

{

**if**(msg==timeoutEvent){

EV<<"Timeout expired";

cPacket \*pkt\_copy = pkt ->dup();

send(pkt\_copy, "gate\_out");

**scheduleAt**(simTime()+timeout, timeoutEvent);

}

**else**{

cPacket\* pktr = check\_and\_cast<cPacket\*>(msg); //packet received from destination

**int** type = pktr ->getKind(); //assume type 0 means ACK and type 1 means NACK

**if** (type == 0){ //ACK

EV<<"Received ACK - cancel timeout";

**cancelEvent**(timeoutEvent);

pkt = **new** cPacket("packet");

pkt -> **setBitLength**(100);

cPacket \*pkt\_copy = pkt->dup();

send(pkt\_copy, "gate\_out");

**scheduleAt**(simTime()+timeout, timeoutEvent);

}

**else**{ //NACK

EV<<"NACK received - sending again";

cPacket \*pkt\_copy = pkt ->dup();

send(pkt\_copy, "gate\_out");

**cancelEvent**(timeoutEvent);

**scheduleAt**(simTime()+timeout, timeoutEvent);

}

}

}

destARQ.cc

**#include** "destARQ.h"

Define\_Module(DestARQ);

**void** **DestARQ::handleMessage**(cMessage \*msg)

{

cPacket\* pkt\_r = check\_and\_cast<cPacket\*>(msg);

**if**(pkt\_r ->hasBitError()){

EV<<"Packet received in error";

pkt\_r -> setKind(1); //NACK

pkt\_r -> setBitError(**false**); //resets the BER flag because we use the same packet

}

**else**{

EV<<"Packet received correctly";

pkt\_r -> setKind(0); //ACK

}

send(pkt\_r, "gate\_out");

}

destARQ.h

**#ifndef** \_\_MY\_ARQ2\_DESTARQ\_H\_

**#define** \_\_MY\_ARQ2\_DESTARQ\_H\_

**#include** <omnetpp.h>

**using** **namespace** omnetpp;

/\*\*

\* **TODO** - Generated class

\*/

**class** DestARQ : **public** cSimpleModule

{

**protected**:

**virtual** **void** **handleMessage**(cMessage \*msg);

};

**#endif**

destARQ.ned

**simple** DestARQ

{

**gates**:

**input** gate\_in;

**output** gate\_out;

}