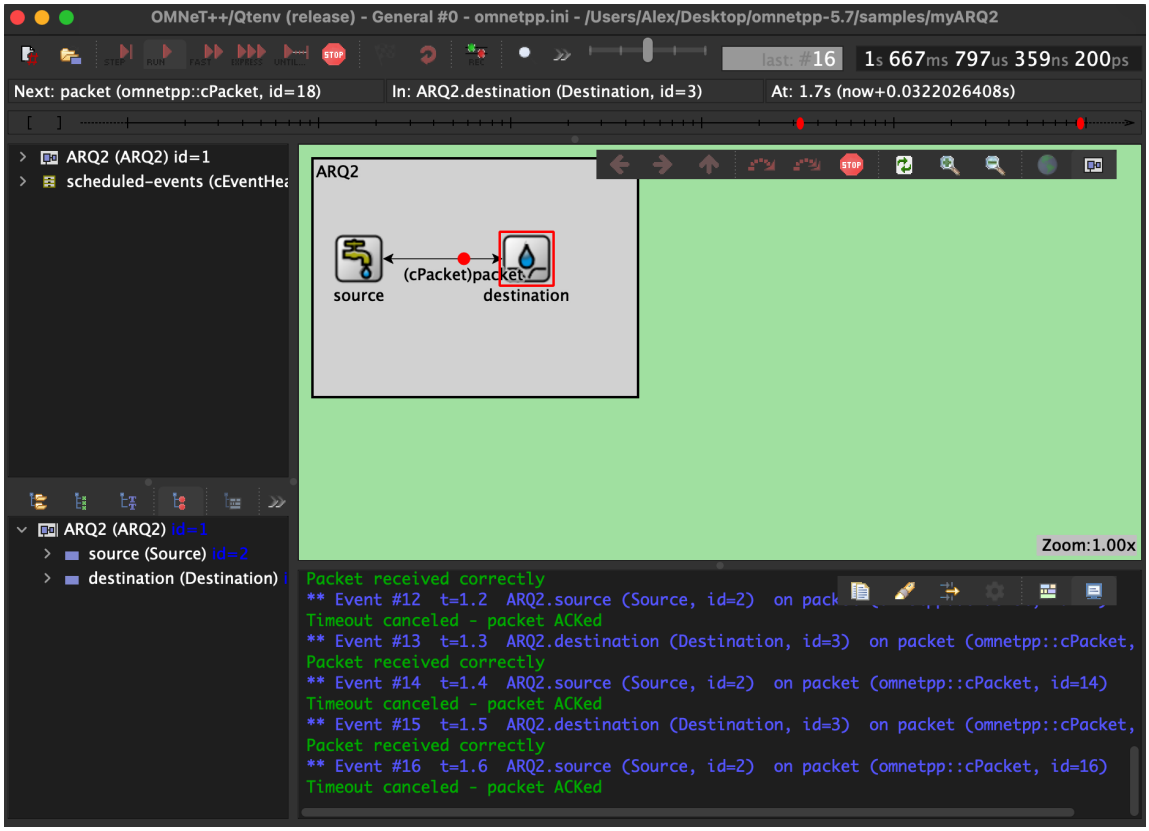
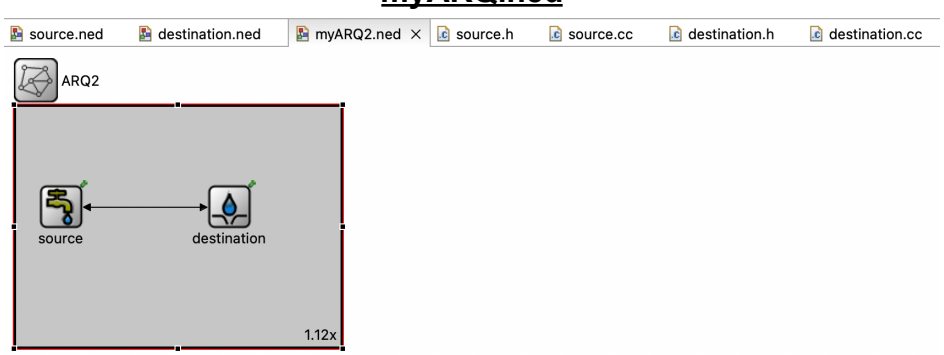
myARQ2



myARQ.ned




```
network ARQ2
{
    @display("bgb=264,193");
    submodules:
        source: Source {
            @display("p=37,81");
        }
        destination: Destination {
            @display("p=174,81");
        }
    connections:
        source.out -->{ber = 1e-3; delay = 100 ms;} --> destination.in;
        destination.out --> {delay = 100 ms;}--> source.in;
}
```

source.ned

```
simple Source
{
    parameters:
        @display("i=block/source");
    gates:
        input in;
        output out;
}
```

destination.ned

```
simple Destination
{
    parameters:
        @display("i=block/sink");
        gates:
            input in;
            output out;
}
```

source.h

```
#ifndef __ARQ2_SOURCE_H_
#define __ARQ2_SOURCE_H_

#include <omnetpp.h>

using namespace omnetpp;

class Source : public cSimpleModule
{
  private:
    simtime_t timeout;
    cMessage *timeoutEvent;
    cPacket *pkt;

  public:
    Source();
    virtual ~Source();

  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

#endif
```

source.cc

```
#include "source.h"

Define_Module(Source);

Source::~Source(){
    cancelAndDelete(timeoutEvent);
}

Source::Source(){
    timeoutEvent = nullptr;
}

void 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, "out");
   scheduleAt(simTime()+timeout, timeoutEvent);

}

void Source::handleMessage(cMessage *msg)
{
   if (msg==timeoutEvent){
       EV<< "Time out expired";
       cPacket *pkt_copy = pkt -> dup();
       send(pkt_copy, "out");
       scheduleAt(simTime()+timeout, timeoutEvent);
   }
   else {
       cPacket* pktr = check_and_cast<cPacket *> (msg); //we assume that Destination writes ACKs and NACKS
       int type = pktr->getKind();
              if (type == 0) {
                  //Destination sets the pkt type to 0 is ACK and 1 if NACK
                  EV<<"Timeout canceled - packet ACKed";
                  cancelEvent(timeoutEvent);
                  pkt = new cPacket("packet");
                  pkt -> setBitLength(100);
                  cPacket *pkt_copy = pkt -> dup();
                  send(pkt_copy, "out");
                  scheduleAt(simTime()+timeout, timeoutEvent);

              }
              else { //NACK
                  cPacket *pkt_copy = pkt -> dup();
                  send(pkt_copy, "out");
                  cancelEvent(timeoutEvent);
                  scheduleAt(simTime()+timeout, timeoutEvent);

              }
          }
       }
```

destination.h

```
#ifndef __ARQ2_DESTINATION_H_
#define __ARQ2_DESTINATION_H_

#include <omnetpp.h>

using namespace omnetpp;

class Destination : public cSimpleModule
{
  protected:
    virtual void handleMessage(cMessage *msg);
};

#endif
```

destination.cc

```
#include "destination.h"

Define_Module(Destination);


void Destination::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
            } else {
                EV<<"Packet received correctly";
                pkt_r ->setKind(0); //ACK
            }
            pkt_r -> setBitError(false);
            send(pkt_r, "out");
         }
```