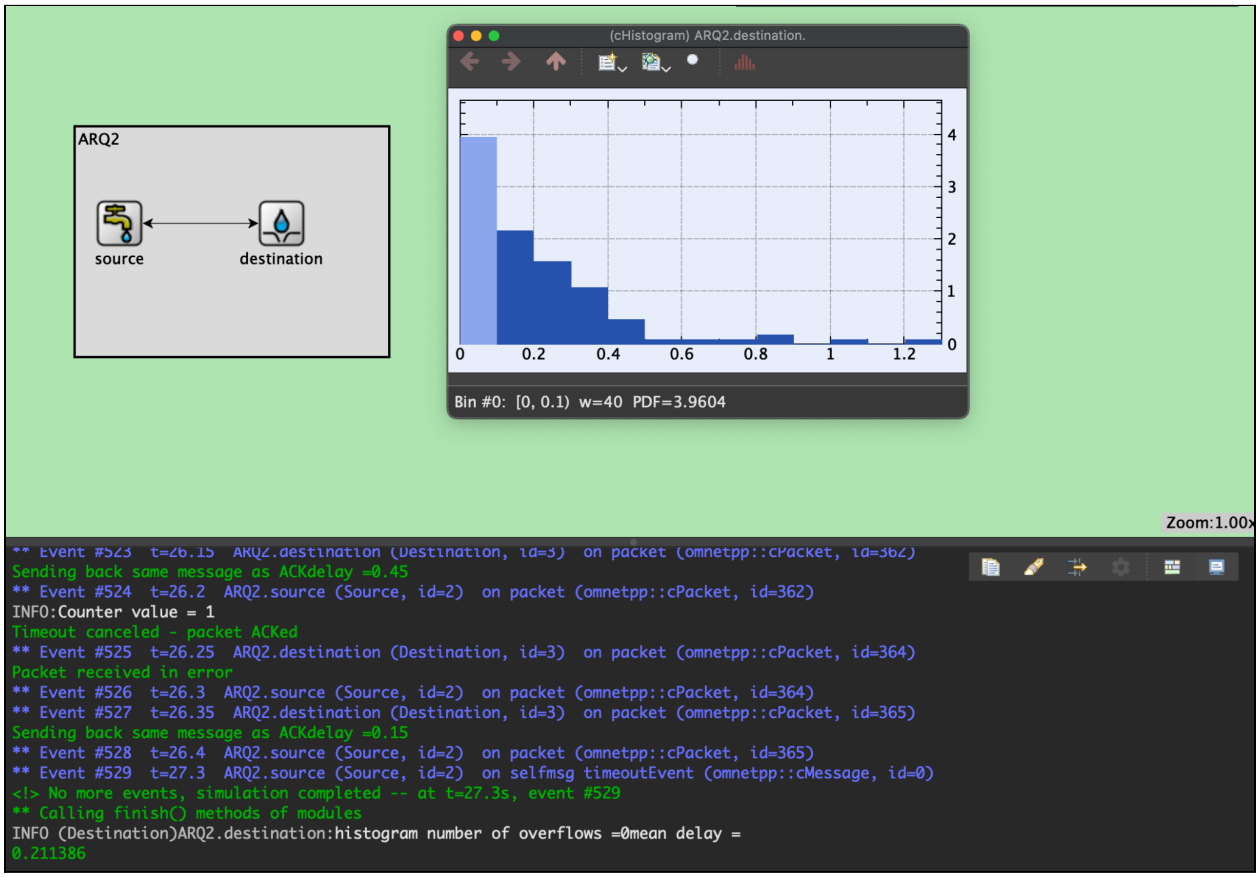
ARQ Simulation with Histogram



After adding to the ARQ program, a statistics function that read and displayed delay times to a histogram when the simulation was run has been successfully implemented as shown. The bit error rate in the .ned file is defined as 1e-2, as well as the delay at 50 ms.

Destination.h Code

```
#ifndef __ARQ2_DESTINATION_H_
#define __ARQ2_DESTINATION_H_

#include <omnetpp.h>

using namespace omnetpp;

class Destination : public cSimpleModule
{
    private:
        cStdDev delayStats;
        cHistogram histogram;
    protected:
        virtual void initialize();
        virtual void handleMessage(cMessage *msg);
        virtual void finish();
};

#endif
```

Destination.cc Code

```
#include "destination.h"

Define_Module(Destination);

void Destination::initialize() {
    delayStats.setName("Delay");
    histogram.setRangeAuto(5,1.5);
    histogram.setNumCells(10);
}


void Destination::handleMessage(cMessage *msg)
{
    cPacket* pkt = check_and_cast<cPacket *>(msg);
        if (pkt ->hasBitError()){
            EV<<"Packet received in error";
            pkt ->setKind(1); //kind =1 NACK
        }
        else {
            EV<<"Sending back same message as ACK";
            pkt ->setKind(0); //kind =0 ACK
            simtime_t delay = simTime()- pkt->getCreationTime();
            EV<<"delay =";
            EV<< delay;
            delayStats.collect(delay);
            histogram.collect(delay);
        }
        send(pkt, "out");
         }

void Destination::finish()
{
  recordScalar("mean delay",delayStats.getMean());
  recordScalar("max delay",delayStats.getMax());
  recordScalar("std deviation",delayStats.getStddev());
  recordScalar("variance",delayStats.getVariance());
  histogram.record();
  //test if the selected range was appropriate
  EV<<"histogram number of overflows =";
  EV <<histogram.getOverflowCell();
  EV<<"mean delay = \n";
  EV<<delayStats.getMean();
}
```

Calculating the Average Delay

The histogram provided the following delay values:

Bin	Delay Value
0 - 0.1	3.9604
0.1 - 0.2	2.17822
0.2 - 0.3	1.58416
0.3 - 0.4	1.08911
0.4 - 0.5	0.49505
0.5 - 0.6	0.0990099
0.6 - 0.7	0.0990099
0.7 - 0.8	0.0990099
0.8 - 0.9	0.19802
0.9 - 1.0	0
1.0 - 1.1	0.0990099
1.1 - 1.2	0
1.2 - 1.3	0.0990099

$$delay_{avg} = [(\frac{0.1}{2} \times 3.9604) + (\frac{0.2+0.1}{2} \times 2.17822) + (\frac{0.3+0.2}{2} \times 1.58416) + (\frac{0.4+0.3}{2} \times 1.08911) + (\frac{0.5+0.4}{2} \times 0.49505) + (\frac{0.6+0.5}{2} \times 0.0990099) + (\frac{0.7+0.6}{2} \times 0.0990099) + (\frac{0.8+0.7}{2} \times 0.0990099) + (\frac{0.9+0.8}{2} \times 0.19802) + (\frac{1+0.9}{2} \times 0) + (\frac{1.1+1}{2} \times 0.0990099) + (\frac{1.2+1.1}{2} \times 0) + (\frac{1.3+1.2}{2} \times 0.0990099)] / n_{bins}$$

The number of bins was set in the code to be 10, so n_{bins} = 10. Thus, the average delay is equal to **0.2113978575**, which is close to the experimental delay provided by the ARQ simulation at 0.211386. From the simulation process, the histogram fits the performance of the ARQ. On average, a delay buildup was quite common when multiple packets were transmitted simultaneously, which atoned for how common the higher delay value was in the distribution.