###### *CSE 473 – Introduction to Computer Networks*

Lab 4 Report – 70 Points

##### *Gilad Gabriel Tsehori, Gai Ashkenazy 1*

*Part A*. **[20 points]**Paste a copy of the completed source code for *Rdt* below. Highlight your changes by making them **bold**(you may omit sections of the original program that contain no added code). Remember to also place a complete copy in the repository before you make your final commit.

public void run() {

**long t0 = System.nanoTime();**

**long now = 0; // current time (relative to t0)**

**// set flag to prevent multiple sendings of duplicate acks**

**boolean flag = true;**

**while (!quit || /\* we still have un-acked packets \*/ sendBase != sendSeqNum) {**

**// if there are no un-acked packets, stop the timer**

**if (sendBase == sendSeqNum) {**

**t0 = System.nanoTime();**

**}**

**now = System.nanoTime() - t0;**

**// if receive buffer has a packet that can be delivered, deliver it to sink**

**if(recvBase != lastRcvd && lastRcvd != -1) {**

**toSnk.add(recvBuf[recvBase].payload);**

**recvBase = incr(recvBase);**

**}**

**// else if the substrate has an incoming packet**

**else if (sub.incoming()){**

**// get the packet from the substrate and process it**

**Packet inPack = sub.receive();**

**// if it's a data packet, ack it and add it to receive buffer as appropriate**

**if (inPack.type == 0){**

**// if expected packet is received**

**if(expSeqNum == inPack.seqNum){**

**//pass packet to recvBuf**

**recvBuf[inPack.seqNum] = inPack;**

**expSeqNum = incr(expSeqNum);**

**//Ack expected packet**

**Packet newACK = new Packet();**

**newACK.type = 1;**

**newACK.seqNum = expSeqNum;**

**while(!sub.ready()){ /\* busy waiting \*/ }**

**sub.send(newACK);**

**// increase lastRcvd right after sending ACK**

**lastRcvd = expSeqNum;**

**// else this is an unexpected packet**

**} else {**

**//Ack again for expected packet**

**Packet newACK = new Packet();**

**newACK.type = 1;**

**newACK.seqNum = expSeqNum;**

**while(!sub.ready()){ /\* busy waiting \*/ }**

**sub.send(newACK);**

**}**

**}**

**// if it's an ack, update the send buffer and**

**// related data as appropriate reset the timer if necessary**

**else { //expected or higher ACK**

**if ((sendBase + wSize < 2\*wSize && sendBase < inPack.seqNum) || (sendBase + wSize >= 2\*wSize && (sendBase < inPack.seqNum || ((sendBase + wSize) % (2\*wSize)) > inPack.seqNum))) {**

**sendBase = inPack.seqNum;**

**//reset timer**

**t0 = System.nanoTime();**

**now = 0; // current time (relative to t0)**

**//reset duplicate acks and flag**

**dupAcks = 0;**

**flag = true;**

**//duplicate acks**

**} else if (inPack.seqNum == sendBase) {**

**//third duplicate ACK**

**if (dupAcks == 2 && flag){**

**flag = false;**

**int diff = diff(sendSeqNum, sendBase);**

**short nextSend = sendBase;**

**while (diff > 0){**

**sub.send(sendBuf[nextSend]);**

**nextSend = incr(nextSend);**

**diff--;**

**}**

**//reset timer**

**t0 = System.nanoTime();**

**now = 0; // current time (relative to t0)**

**//reset duplicate acks**

**dupAcks = 0;**

**}**

**// increment dupAcks if flag is true**

**if (flag){**

**dupAcks++;**

**}**

**}**

**}**

**}**

**// else if the resend timer has expired, re-send all**

**// packets in the window and reset the timer**

**else if (now > timeout) {**

**int diff = diff(sendSeqNum, sendBase);**

**short nextSend = sendBase;**

**while (diff > 0) {**

**sub.send(sendBuf[nextSend]);**

**nextSend = incr(nextSend);**

**diff--;**

**}**

**//reset timer**

**t0 = System.nanoTime();**

**now = 0; // current time (relative to t0)**

**//reset duplicate acks**

**dupAcks = 0;**

**}**

**// else if there is a message from the source waiting to be sent and the send window is not full**

**// and the substrate can accept a packet create a packet containing the message,**

**// and send it, after updating the send buffer and related data**

**else if (fromSrc.size() > 0 && diff((short) ((sendBase + wSize) % sendBuf.length), sendSeqNum) > 0 && sub.ready()) {**

**//create a new packet to send**

**Packet newPacket = new Packet();**

**newPacket.type = 0;**

**newPacket.seqNum = sendSeqNum;**

**String payload = "";**

**try {**

**payload = fromSrc.take();**

**} catch (Exception e) {**

**System.err.println("Rdt: take exception" + e);**

**System.exit(1);**

**}**

**newPacket.payload = payload;**

**//send packet**

**sub.send(newPacket);**

**sendBuf[sendSeqNum] = newPacket;**

**//update sendSeqNum**

**sendSeqNum = incr(sendSeqNum);**

**}**

**// else nothing to do, so sleep for 1 ms**

**else {**

**try{**

**Thread.sleep(1);**

**} catch (Exception e) {**

**System.err.println("Rdt: sleep exception" + e);**

**System.exit(1);**

**}**

**}**

**}**

}

*Part B.* **[10 points]** Use the provided *script0* to test your client and server on a single computer. You may do this testing on any Unix (including MacOS) or Linux computer (shell.cec.wustl.edu or onl.wustl.edu). All you need to do is type

./script0

in the folder that contains all your code. Paste a copy of the output below.

**Note**: if there are errors in your code, you might need to kill the process running the server in order to free up the port this script uses. To do this, use the command “pkill -9 java”.

wSize= 5 timeout= .5 dropProb= 0.25

\*\*\*\*\*\*\*\*\*\*\*\*\*\* client report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[0] testing 0

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[1] testing 1

discarding data[2] testing 2

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[3] testing 3

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[1]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[5] testing 5

discarding data[6] testing 6

discarding data[2] testing 2

discarding data[3] testing 3

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[5] testing 5

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[6] testing 6

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[2]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[2] testing 2

discarding data[3] testing 3

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[5] testing 5

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[6] testing 6

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[3]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[3]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[7] testing 7

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[3] testing 3

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[5] testing 5

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[6] testing 6

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[7] testing 7

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[5]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[6]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[7]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[8]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[8] testing 8

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 data[9] testing 9

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[9]

/127.0.0.1:55841 received from /127.0.0.1:11313 ack[0]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[1] testing 1

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[0]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[2] testing 2

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[0]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[3] testing 3

discarding ack[0]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[1] testing 1

/127.0.0.1:55841 received from /127.0.0.1:11313 data[3] testing 3

/127.0.0.1:55841 received from /127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[0]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[0]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[0]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[0] testing 0

/127.0.0.1:55841 received from /127.0.0.1:11313 data[2] testing 2

/127.0.0.1:55841 received from /127.0.0.1:11313 data[3] testing 3

/127.0.0.1:55841 received from /127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[1]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[1]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[1]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[1]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[1] testing 1

/127.0.0.1:55841 received from /127.0.0.1:11313 data[2] testing 2

/127.0.0.1:55841 received from /127.0.0.1:11313 data[3] testing 3

/127.0.0.1:55841 received from /127.0.0.1:11313 data[4] testing 4

/127.0.0.1:55841 received from /127.0.0.1:11313 data[5] testing 5

discarding ack[2]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[3]

discarding ack[4]

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[5]

discarding ack[6]

/127.0.0.1:55841 received from /127.0.0.1:11313 data[5] testing 5

/127.0.0.1:55841 sending to localhost/127.0.0.1:11313 ack[6]

Sender: sent 25 data packets, 16 acks

discarded 5 data packets, 4 acks

runLength 3.013365329

Receiver: received 16 data packets, 15 acks

discarded 0 arrivals

runLength 2.913467684

SrcSnk: sent 10, received 6

runLength 0.3

\*\*\*\*\*\*\*\*\*\*\*\*\*\* server report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/127.0.0.1:11313 received from /127.0.0.1:55841 data[0] testing 0

/127.0.0.1:11313 received from /127.0.0.1:55841 data[1] testing 1

/127.0.0.1:11313 received from /127.0.0.1:55841 data[3] testing 3

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[1]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[5] testing 5

discarding ack[2]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 received from /127.0.0.1:55841 data[5] testing 5

/127.0.0.1:11313 received from /127.0.0.1:55841 data[6] testing 6

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[2]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[2] testing 2

/127.0.0.1:11313 received from /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 received from /127.0.0.1:55841 data[5] testing 5

discarding ack[3]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[6] testing 6

discarding ack[3]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[3]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[3]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[7] testing 7

discarding ack[3]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[3] testing 3

/127.0.0.1:11313 received from /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 received from /127.0.0.1:55841 data[5] testing 5

discarding ack[4]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[6] testing 6

/127.0.0.1:11313 received from /127.0.0.1:55841 data[7] testing 7

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[5]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[6]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[7]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[8]

/127.0.0.1:11313 received from /127.0.0.1:55841 data[8] testing 8

/127.0.0.1:11313 received from /127.0.0.1:55841 data[9] testing 9

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[9]

/127.0.0.1:11313 sending to /127.0.0.1:55841 ack[0]

discarding data[0] testing 0

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[1] testing 1

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[0]

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[2] testing 2

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[0]

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[3] testing 3

discarding data[4] testing 4

discarding data[0] testing 0

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[1] testing 1

discarding data[2] testing 2

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[3] testing 3

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[0]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[0]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[0]

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[0] testing 0

discarding data[1] testing 1

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[2] testing 2

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[3] testing 3

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[1]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[1]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[1]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[1]

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[1] testing 1

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[2] testing 2

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[3] testing 3

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[4] testing 4

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[5] testing 5

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[3]

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[5]

/127.0.0.1:11313 sending to /127.0.0.1:55841 data[5] testing 5

/127.0.0.1:11313 received from /127.0.0.1:55841 ack[6]

Sender: sent 21 data packets, 20 acks

discarded 5 data packets, 5 acks

runLength 2.910701777

Receiver: received 20 data packets, 12 acks

discarded 0 arrivals

runLength 3.011154605

SrcSnk: sent 6, received 10

runLength 0.3

1. Based on the report output, how many of the packets sent by the client were retransmissions? How many of these were caused by the discarding of the data packets and how many were caused by the discarding of acknowledgments?

*The client intended to send to the server 10 packets, but overall 25 packets were sent out; therefore 15 of them were retransmissions.*

*We will count how many packets were retransmitted because of discarding data packets; we can see 5 of the packets were discarded, hence all of the retransmissions (15) occurred because of discarded data packets.  
There can be a case that discarding acknowledgements will cause retransmissions; for example, if the client sent packets 0 to 4 to the server, and the server sends acknowledgements only for 0 to 3 (acknowledge 4 was discarded by the server) and then a timeout occurs, packet 4 will be retransmitted by the client. In our case, it didn’t happen.*

1. What was the specified run length for this test? How does that compare to the actual time it took to transfer all the packets?

*The specified run length of the test was 0.3 seconds.  
The actual time where all packets were transmitted between the client and the server is 3.011 seconds. Consequently, we can see that the time it took the threads to populate the packets and send them out is much faster (which is 0.3 seconds set by an argument) than the time it took for all packets to be received reliably in the other host (which took approximately 2.7 seconds, since the time of creation of the initial packets is included in this total time).  
We can also tell that this time goes (and drops) when we increase and decrease the probability that a packet is discarded by either host.*

*Part C*. **[5 points]** Use the provided *script1* to test your client and server on two computers in ONL, using the provided ONL configuration. To run the script, just type

./script1

in the folder ~/473/lab4 on your onl account. Your Java classes should be in this folder, along with the script. Paste a copy of the output below.

wSize= 5 timeout= .5 dropProb= .25

\*\*\*\*\*\*\*\*\*\*\*\*\*\* client report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[0] testing 0

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[1] testing 1

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[2] testing 2

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[3] testing 3

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[4] testing 4

discarding data[0] testing 0

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[1] testing 1

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[2] testing 2

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[3] testing 3

discarding data[4] testing 4

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[1]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[5] testing 5

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[2]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[6] testing 6

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[3]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[7] testing 7

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[4]

discarding data[8] testing 8

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[4]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[4]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[4]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[4] testing 4

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[5] testing 5

discarding data[6] testing 6

discarding data[7] testing 7

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[8] testing 8

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[5]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[0] testing 0

discarding ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[2] testing 2

discarding ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[3] testing 3

discarding ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[4] testing 4

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

discarding data[6] testing 6

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[7] testing 7

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[8] testing 8

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[4] testing 4

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[5] testing 5

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

discarding data[6] testing 6

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[7] testing 7

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[8] testing 8

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[3] testing 3

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[4] testing 4

/192.168.4.2:38601 received from /192.168.7.1:11313 data[5] testing 5

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[1]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[1] testing 1

/192.168.4.2:38601 received from /192.168.7.1:11313 data[2] testing 2

/192.168.4.2:38601 received from /192.168.7.1:11313 data[3] testing 3

/192.168.4.2:38601 received from /192.168.7.1:11313 data[4] testing 4

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[2]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[3]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[4]

discarding ack[5]

/192.168.4.2:38601 received from /192.168.7.1:11313 data[5] testing 5

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 ack[6]

discarding data[6] testing 6

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[7] testing 7

discarding data[8] testing 8

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[6]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[6] testing 6

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[7] testing 7

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[8] testing 8

discarding data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[7]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[8]

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[9]

/192.168.4.2:38601 sending to h7x1/192.168.7.1:11313 data[9] testing 9

/192.168.4.2:38601 received from /192.168.7.1:11313 ack[0]

Sender: sent 37 data packets, 14 acks

discarded 10 data packets, 4 acks

runLength 3.180379969

Receiver: received 14 data packets, 21 acks

discarded 0 arrivals

runLength 2.745700478

SrcSnk: sent 10, received 6

runLength 0.3

\*\*\*\*\*\*\*\*\*\*\*\*\*\* server report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/192.168.7.1:11313 received from /192.168.4.2:38601 data[0] testing 0

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[1] testing 1

/192.168.7.1:11313 received from /192.168.4.2:38601 data[2] testing 2

/192.168.7.1:11313 received from /192.168.4.2:38601 data[3] testing 3

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[2]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[3]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[4]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[5] testing 5

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[4]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[6] testing 6

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[4]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[7] testing 7

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[4]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[4] testing 4

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[5]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[5] testing 5

/192.168.7.1:11313 received from /192.168.4.2:38601 data[8] testing 8

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[9] testing 9

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[0] testing 0

discarding data[1] testing 1

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[2] testing 2

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[3] testing 3

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[4] testing 4

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

discarding data[5] testing 5

/192.168.7.1:11313 received from /192.168.4.2:38601 data[7] testing 7

/192.168.7.1:11313 received from /192.168.4.2:38601 data[8] testing 8

discarding ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[9] testing 9

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

discarding data[1] testing 1

discarding data[2] testing 2

discarding data[3] testing 3

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[4] testing 4

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[5] testing 5

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[7] testing 7

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[8] testing 8

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[9] testing 9

discarding ack[6]

discarding data[1] testing 1

discarding data[2] testing 2

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[3] testing 3

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[4] testing 4

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[5] testing 5

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[1]

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[1] testing 1

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[2] testing 2

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[3] testing 3

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[4] testing 4

/192.168.7.1:11313 sending to /192.168.4.2:38601 data[5] testing 5

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[2]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[3]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[4]

/192.168.7.1:11313 received from /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[7] testing 7

/192.168.7.1:11313 received from /192.168.4.2:38601 data[9] testing 9

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[6]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[6] testing 6

/192.168.7.1:11313 received from /192.168.4.2:38601 data[7] testing 7

/192.168.7.1:11313 received from /192.168.4.2:38601 data[8] testing 8

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[7]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[8]

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[9]

/192.168.7.1:11313 received from /192.168.4.2:38601 data[9] testing 9

/192.168.7.1:11313 sending to /192.168.4.2:38601 ack[0]

Sender: sent 21 data packets, 23 acks

discarded 7 data packets, 2 acks

runLength 2.773269061

Receiver: received 23 data packets, 10 acks

discarded 0 arrivals

runLength 2.864733642

SrcSnk: sent 6, received 10

runLength 0.3

*Part D*. **[15 points]** Use the provided *script2* to run this next test. Paste a copy of the output below. Also, paste a screen capture showing the two monitoring windows labeled “from/to hosts” and “inter-router traffic”. Make sure your screenshot shows the curves for the entire duration of the script run, and that the text labels are large enough to read on a printed copy. You will find it easier to do the screen capture if you first “stop” the chart by using the “Stop” menu item in the Options menu (to restart the chart, select this item again).

wSize= 5 timeout= .5 dropProb= .2

\*\*\*\*\*\*\*\*\*\*\*\*\*\* client report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 484 data packets, 144 acks

discarded 117 data packets, 25 acks

runLength 22.906953139

Receiver: received 144 data packets, 288 acks

discarded 0 arrivals

runLength 22.859482089

SrcSnk: sent 200, received 100

runLength 20.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\* server report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 182 data packets, 367 acks

discarded 38 data packets, 79 acks

runLength 22.824386695

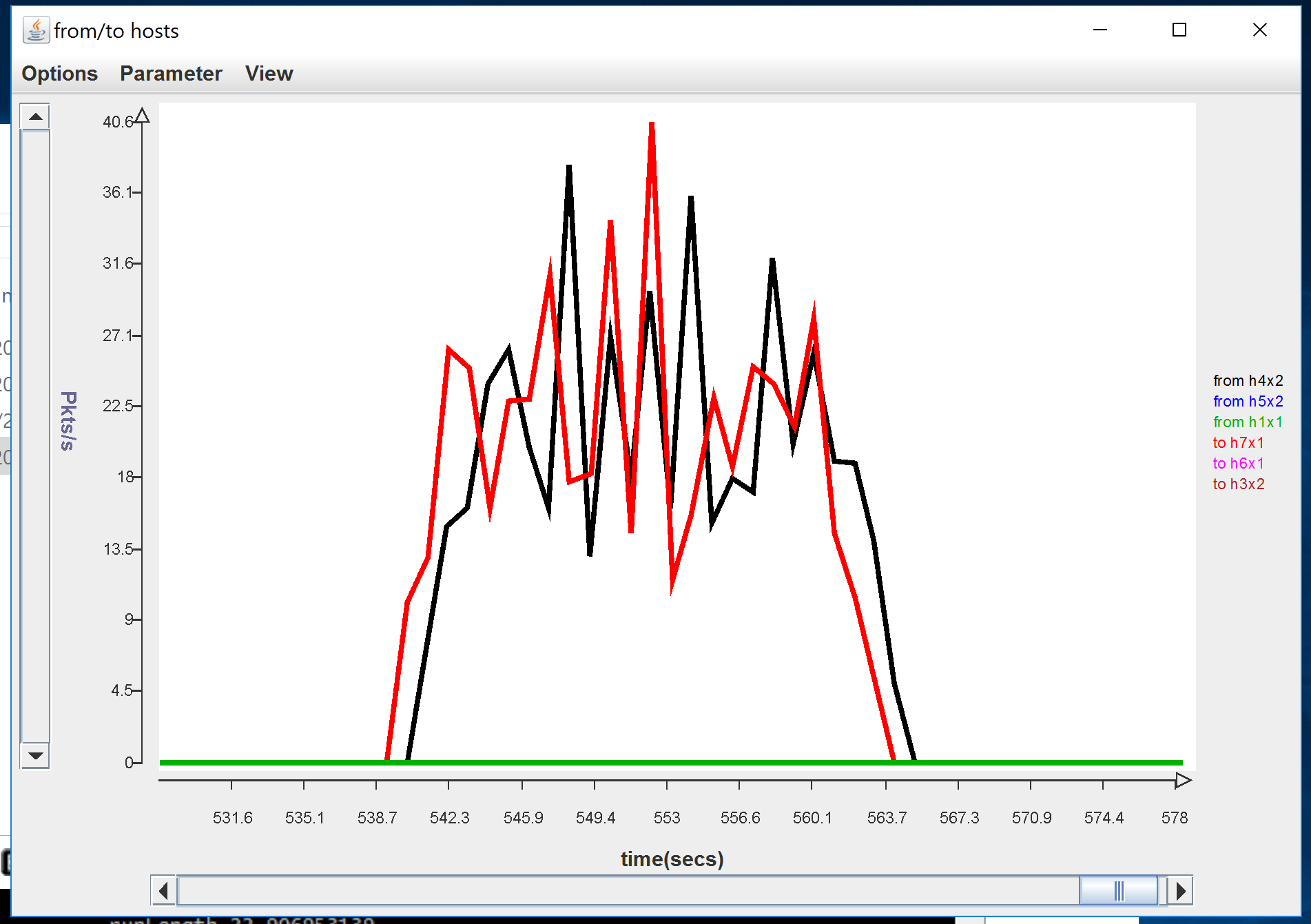
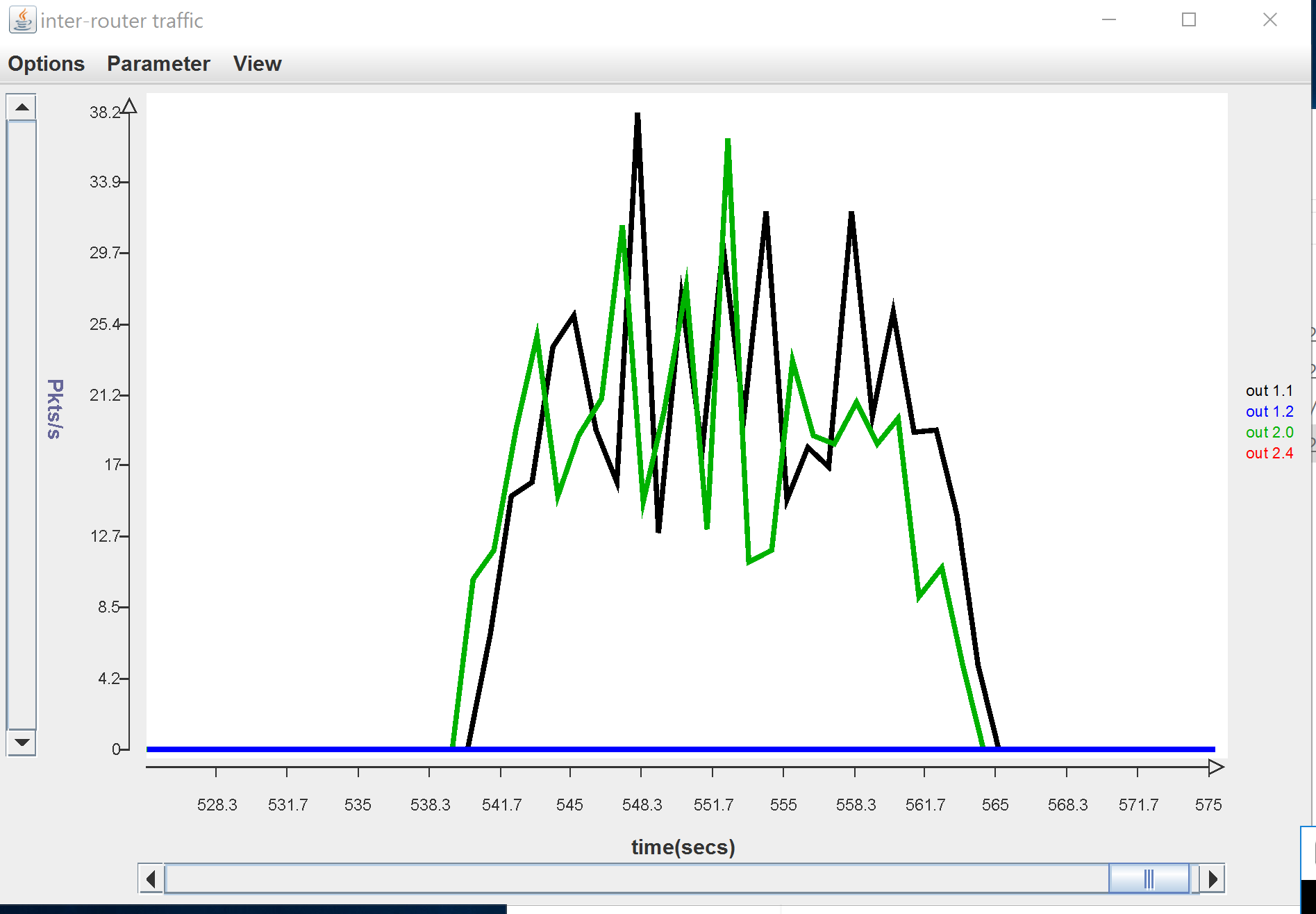
Receiver: received 367 data packets, 119 acks

discarded 0 arrivals

runLength 22.916975634

SrcSnk: sent 100, received 200

runLength 20.0

Answer the following questions, based on your the results of this test.

1. What was the specified run length for this script? What was the specified packet sending rate for the client and server? (You will need to examine the script in order to answer this question.)

*The specified run length was 20 seconds. To calculate the packet sending rate, we need to calculate the following for each: .  
For the client, we have: , and for the server we have: .*

1. How long did it take to deliver all the packets? What was the effective packet delivery rate from the client to the module at the server?

*It took 22.9069 seconds to deliver all packets from the client to the server (including the time of creating the packets).*

*The effective packet delivery rate is computed as follows:   
We can see that this rate is slower than the specified run length from the previous question.*

1. How many packets did the *Rdt* module at the client send (including retransmissions but excluding acks)? How many of these were retransmissions? What was the average sending rate for the client, including both retransmissions and acks?

*The client has sent 484 packets, including retransmissions and excluding acks. Out of those, 484 - 200 = 284 were retransmissions.*

*The average sending rate can be calculated as follows:*

*Part E.* **[20 points]** In this part you will be using the provided *script3* to answer some questions about the performance of your protocol when run from a client at *h4x2* to a server at *h7x1* (in this script, the server does not send any data packets). The script takes several arguments, whose values you will need to specify, when running the experiments needed to answer the questions below.

1. Determine the round-trip delay between *h4x2* and *h7x1* using *ping* (make sure you are using the correct addresses, so that your packets go through your experimental network, and not the ONL control network). What value did you get? Based on this, if your protocol is configured with a window size of 1 packet, what is the maximum rate at which it can send packets? What is the smallest window size that would allow it to send 1000 packets per second? Note that later answers in this section depend on your ability to answer this part correctly, so make sure you understand this.

*Using the ping command, we’ve seen that the average round-trip delay between the two hosts is 50.1 ms, therefore we will set the RTT in the arguments to about 51 ms.  
Based on this and a given window size of 1 packet, the maximum rate at which we can send packets is   
In order to find the smallest window size that would allow it to send 1000 packets per second, we will solve the following equation for x: .*

1. In this part, you will run *script3* with a timeout value of 0.6 seconds, a drop probability of 0 and a delta value of .004. What sending rate does this correspond to? Choose the smallest window size that is consistent with this sending rate and paste a copy of the output of your run below. Were the packets actually delivered to the destination at the specified sending rate?

*In this case, we have that RTT = 0.6 seconds (like the previous questions) and we also have a transmission rate that can be derived from the delta value; there is 1 packet sent every 0.004 seconds (4 ms), so in one second, packets per second, which is the sending rate.*

*The smallest window size that will correspond to this sending rate is: packets, so we round it up to 13.*

*The packets were indeed sent in approximately 250 packets per second, as .*

*Copy of the output:*

wSize= 13 timeout= 0.6 dropProb= 0 delta= .004

\*\*\*\*\*\*\*\*\*\*\*\*\*\* client report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 2500 data packets, 0 acks

discarded 0 data packets, 0 acks

runLength 9.995724497

Receiver: received 0 data packets, 2500 acks

discarded 0 arrivals

runLength 10.04893326

SrcSnk: sent 2500, received 0

runLength 10.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\* server report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 0 data packets, 2500 acks

discarded 0 data packets, 0 acks

runLength 9.984510102

Receiver: received 2500 data packets, 0 acks

discarded 0 arrivals

runLength 10.081581384

SrcSnk: sent 0, received 2500

runLength 10.0

1. In this part, you are to determine the maximum rate at which you can send traffic between the two routers. Determine the maximum sending rate by decreasing the delta value, while increasing the window size to match (keep the timeout value at 0.6 and the discard probability at 0). Observe the packet rate on the inter-router link using the monitoring window and stop decreasing delta when you no longer get any increase in the peak transfer rate observed. At this point, your sending rate is being constrained by the link’s ability to forward packets. Paste a copy of the script output from the run that achieves this maximum packet rate. Also, paste a screen shot showing all three of the monitoring windows from this run.

wSize= 1000 timeout= 0.6 dropProb= 0 delta= 0.0000004

\*\*\*\*\*\*\*\*\*\*\*\*\*\* client report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 131454 data packets, 0 acks

discarded 0 data packets, 0 acks

runLength 10.659573071

Receiver: received 0 data packets, 58433 acks

discarded 0 arrivals

runLength 10.624846952

SrcSnk: sent 25058, received 0

runLength 10.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\* server report \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sender: sent 0 data packets, 58433 acks

discarded 0 data packets, 0 acks

runLength 10.620514705

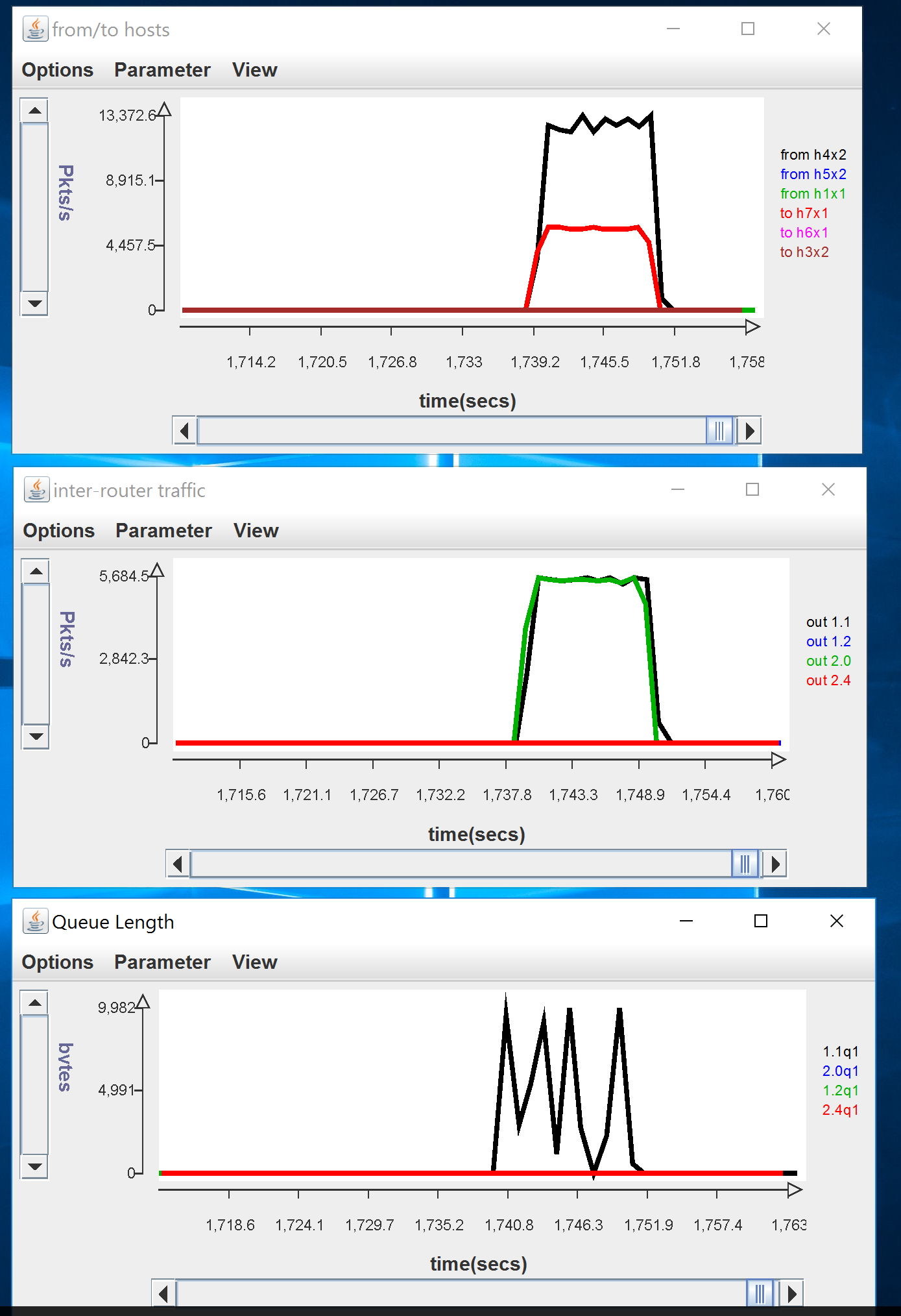
Receiver: received 58433 data packets, 0 acks

discarded 0 arrivals

runLength 10.713296267

SrcSnk: sent 0, received 25058

runLength 10.0



What was the maximum packet sending rate you were able to achieve? What was the specified sending rate? Did you observe any queueing at the inter-router link?

*In order to get to the maximum packet sending rate, we’ve started iterating from question 2’s variables; we did it by first increasing the window size until the rate did not improve, then decreasing the delta value and so forth, until convergence.*

*After experimenting with values, we reached an upper limit (maximum packet sending rate) of about packets/second.  
The specified sending rate, in this case, was packets/second. We can see that this rate was not reached (as expected) as the link capacity was much lower (by a factor of 5.3), and this specified sending rate cannot be satisfied.*

*When starting with the initial values we didn’t see any queuing at the inter-router link (shown in the last graph of the three). Queueing started to appear when the delta value was going very low, and we started to reach the maximum capacity of the link.*

1. Run *script3* with a windows size of 300, timeout of 0.6, discard probability of 0 and a delta of .00015. Now run it again with window sizes of 400, 450 and 500. For each of these runs note the maximum length of the queues at the inter-router link. What are these maximum queue lengths?

* *With wSize = 300, the maximum queue length is 782 packets.*
* *With wSize = 400, the maximum queue length is 5,336 packets.*
* *With wSize = 450, the maximum queue length is 7,636 packets.*
* *With wSize = 500, the maximum queue length is 9,982 packets.*

How does the throughput compare for these four cases?

*For the first, second and third cases, we can tell (by both the data in the outputs and the corresponding graphs) that the throughput was maximal (that is, the sending rate was not higher than the possible sending rate so no packets were discarded by routers and had to be resent).*

*More explicitly, we divide for each case the number of packets received by the receiver by the run length of the script:  
For the first case with wSize = 300, we have that packets/second, almost utilizes the full link speed.  
For the second case with wSize = 400, we have that packets/second.  
For the third case with wSize = 450, we have that packets/second (utilizes the full link speed).  
In the fourth case with wSize = 500, we will not that the full speed of the link is not utilized. In this case, we have that packets/second, so the throughput drops by more than a factor of 2.*

Explain the observed results as best you can. Hint: you may want to examine the queue table at port 1 of router 1.

*Observing both port 1 of router 1 and port 0 of router 2, we can see a large difference in the bandwidth of both routers, where the bottleneck of the link speed comes from the fact that the bandwidth of the queue in router 1 (2500) is significantly lower than the one on router 2 (4000).*

*Moreover, we can see this queueing and capacity difference come into play in the last question, where the window size was increased from 450 to 500 packets; what happened was that many more packets were sent from the sender to the receiver, but many of them were lost. Since the probability of a dropped packet (by network connectivity/errors) is 0, packet loss can only occur when a packet has no room at a queue in one of the routers.  
From our observation about the capacities of both queues, we can conclude that most of these lost packets did not go through router 1, which is the main bottleneck in the link between the two hosts.*