John Hodson
UFID 5244-0415
jhodson@ufl.edu

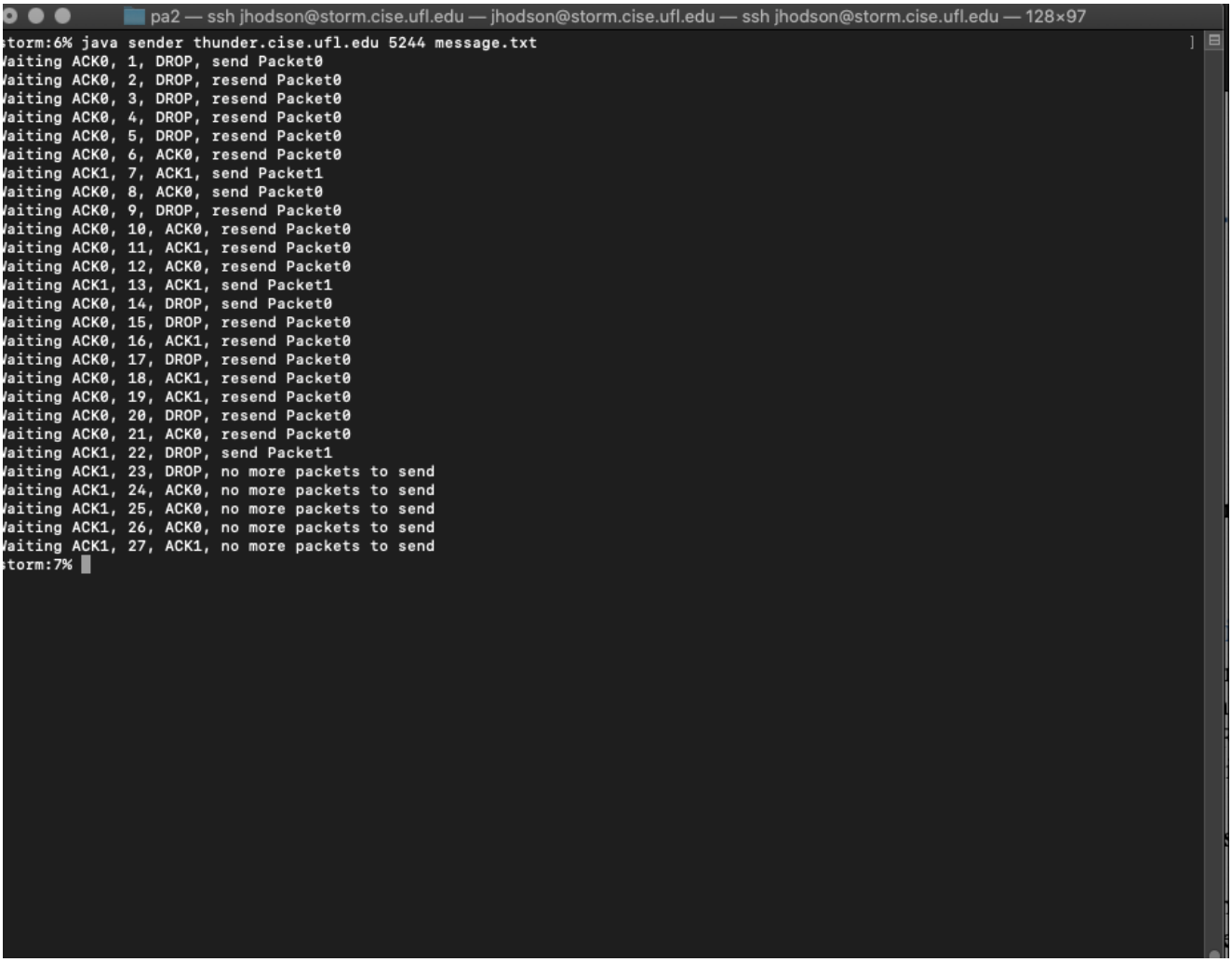
My code may be compiled and run under any environment in which the JDK and the javac compiler is available.

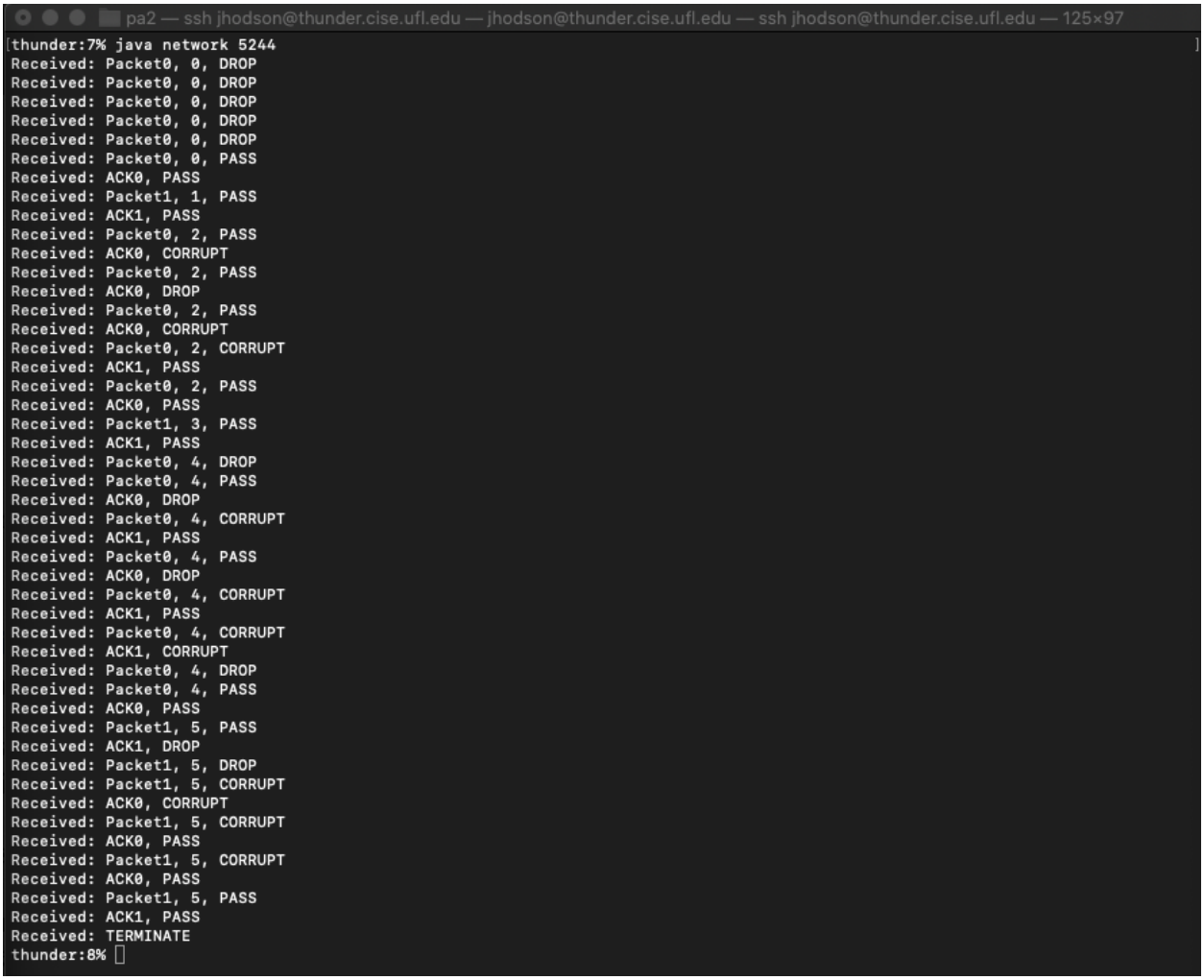
My code is organized as a sender, network, and a receiver. I designed static classes for each.

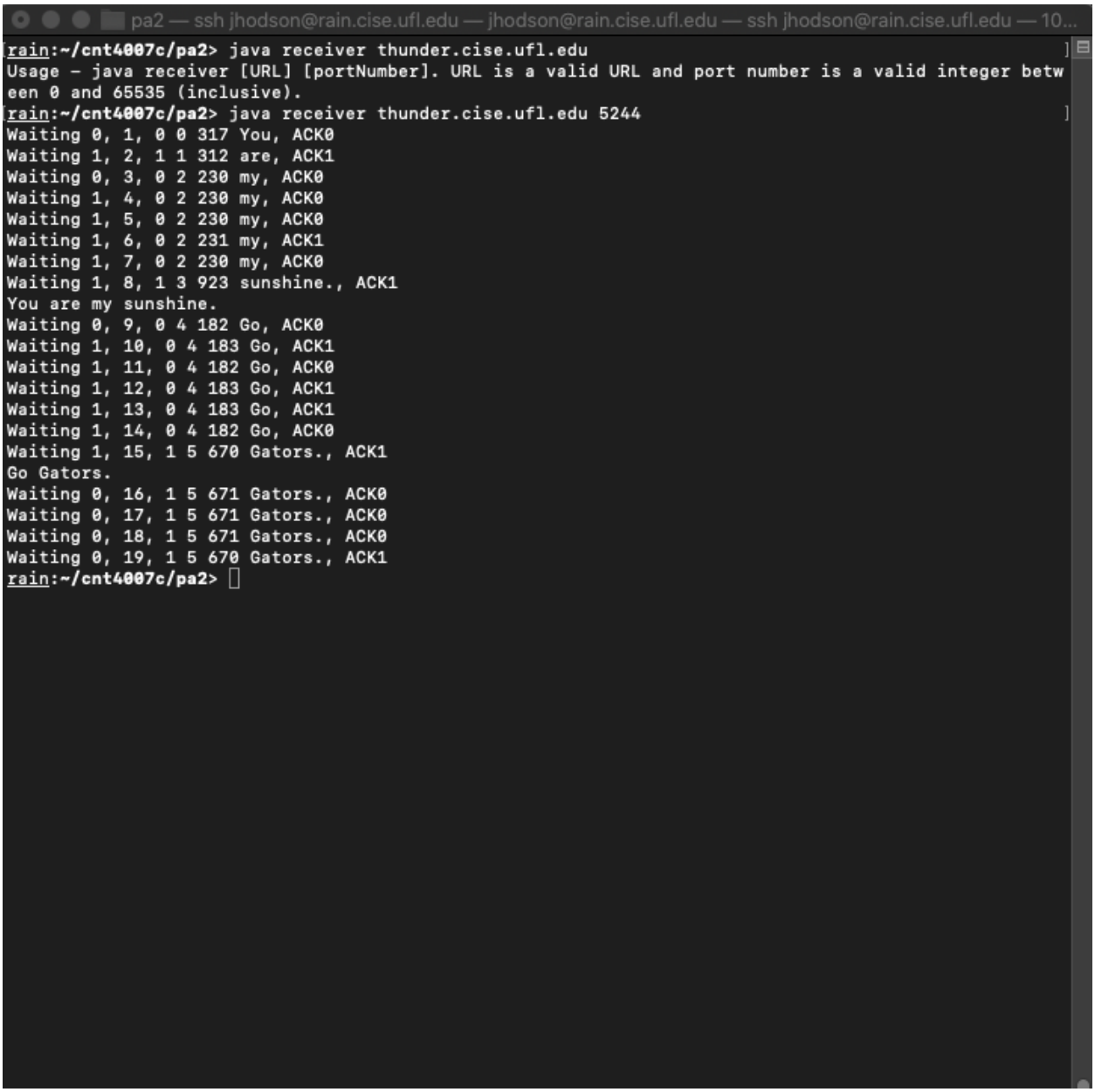
All programs accept command line arguments. The network program accepts a port number on which to run. The sender program accepts a host name and a port number to connect with the network, as well as the name of a file on the local file system from which to extract the messages from that will be sent over the network. The receiver program accepts a host name and a port number to connect with the network. This is implemented as requested in the specification document.

The network program must be launched first, followed by the receiver program, and finally followed by the sender program.

I tested my program on the CISE rain, thunder, and storm servers. A discussion between my three programs can be seen below. Here, my program has the same messages sent over the network as was sent over the network in the project specification.


pa2 — ssh jhodson@storm.cise.ufl.edu — jhodson@storm.cise.ufl.edu — ssh jhodson@storm.cise.ufl.edu — 128×97
storm:6% java sender thunder.cise.ufl.edu 5244 message.txt
Waiting ACK0, 1, DROP, send Packet0
Waiting ACK0, 2, DROP, resend Packet0
Waiting ACK0, 3, DROP, resend Packet0
Waiting ACK0, 4, DROP, resend Packet0
Waiting ACK0, 5, DROP, resend Packet0
Waiting ACK0, 6, ACK0, resend Packet0
Waiting ACK1, 7, ACK1, send Packet1
Waiting ACK0, 8, ACK0, send Packet0
Waiting ACK0, 9, DROP, resend Packet0
Waiting ACK0, 10, ACK0, resend Packet0
Waiting ACK0, 11, ACK1, resend Packet0
Waiting ACK0, 12, ACK0, resend Packet0
Waiting ACK1, 13, ACK1, send Packet1
Waiting ACK0, 14, DROP, send Packet0
Waiting ACK0, 15, DROP, resend Packet0
Waiting ACK0, 16, ACK1, resend Packet0
Waiting ACK0, 17, DROP, resend Packet0
Waiting ACK0, 18, ACK1, resend Packet0
Waiting ACK0, 19, ACK1, resend Packet0
Waiting ACK0, 20, DROP, resend Packet0
Waiting ACK0, 21, ACK0, resend Packet0
Waiting ACK1, 22, DROP, send Packet1
Waiting ACK1, 23, DROP, no more packets to send
Waiting ACK1, 24, ACK0, no more packets to send
Waiting ACK1, 25, ACK0, no more packets to send
Waiting ACK1, 26, ACK0, no more packets to send
Waiting ACK1, 27, ACK1, no more packets to send
storm:7%


pa2 — ssh jhodson@thunder.cise.ufl.edu — jhodson@thunder.cise.ufl.edu — ssh jhodson@thunder.cise.ufl.edu — 125×97
thunder:7% java network 5244
Received: Packet0, 0, DROP
Received: Packet0, 0, DROP
Received: Packet0, 0, DROP
Received: Packet0, 0, DROP
Received: Packet0, 0, DROP
Received: Packet0, 0, PASS
Received: ACK0, PASS
Received: Packet1, 1, PASS
Received: ACK1, PASS
Received: Packet0, 2, PASS
Received: ACK0, CORRUPT
Received: Packet0, 2, PASS
Received: ACK0, DROP
Received: Packet0, 2, PASS
Received: ACK0, CORRUPT
Received: Packet0, 2, CORRUPT
Received: ACK1, PASS
Received: Packet0, 2, PASS
Received: ACK0, PASS
Received: Packet1, 3, PASS
Received: ACK1, PASS
Received: Packet0, 4, DROP
Received: Packet0, 4, PASS
Received: ACK0, DROP
Received: Packet0, 4, CORRUPT
Received: ACK1, PASS
Received: Packet0, 4, PASS
Received: ACK0, DROP
Received: Packet0, 4, CORRUPT
Received: ACK1, PASS
Received: Packet0, 4, CORRUPT
Received: ACK1, CORRUPT
Received: Packet0, 4, DROP
Received: Packet0, 4, PASS
Received: ACK0, PASS
Received: Packet1, 5, PASS
Received: ACK1, DROP
Received: Packet1, 5, DROP
Received: Packet1, 5, CORRUPT
Received: ACK0, CORRUPT
Received: Packet1, 5, CORRUPT
Received: ACK0, PASS
Received: Packet1, 5, CORRUPT
Received: ACK0, PASS
Received: Packet1, 5, PASS
Received: ACK1, PASS
Received: TERMINATE
thunder:8%



My programs all work as expected. Observe the network program (the middle screenshot). The next message in the sequence is sent over the network ONLY when we receive a PASS for both a send and its corresponding acknowledgement. A message has a 1 in 4 chance of being correctly transmitted, which seems to line up with what we see in the print statements of the program statuses.

I was able to successfully implement every requested feature for programming assignment 2, with no bugs or limitations.