Ben Davis

Reliable Data Transfer

Table 1: achieved transfer times under given conditions.

Condition	Parameters	Time
Ideal	Loss rate = 0, dupe rate = 0, ooo rate = 0, ber = 0	174.79 seconds
Good	Loss rate = 0.01, dupe rate = 0.01, ooo rate = 0.01, ber = 1e-8	214.97 seconds
Medium	Loss rate = 0.02, dupe rate = 0.02, ooo rate = 0.02, ber = 1e-6	238.97 seconds
Terrible	Loss rate = 0.1, dupe rate = 0.03, ooo rate = 0.05, ber = 1e-4	Transfer unsuccessful (unreliable)

Protocol:

This is a bit of a messy implementation of reliable data transfer. In order to make my checksum I made my own method in the stop and wait class that adds together all the bytes in a payload. This will be a maximum of 4 digits long. To include sequencing, I created a sequence header that consists of one byte. This value counts up in sink on the sender and receiver sides to 255. Once this value is hit, the sequence number drops back down to 2 so that the value does not exec one byte. No modifications were made to any of the files except for stop and wait.

Limitations:

An error cannot be made when sending the “BEGIN”. This initial packet does not include a header and is important for the data transfer to begin

A duplicate or packet loss error cannot be made on the 255^{th} packet in the sequence, because the stop and wait program compares the sizes of the sequence numbers when one side may have already jumped down to 2 when the other has not.

Finally of note, a byte error cannot occur in the sequence number portion of the header. If this occurs the packet could be treated as a duplicate. There very well be other errors that can occur on top of each other to cause a failure in my protocol as well due to the robust nature of my code.