Jacob Wartofsky and Ian Moreno
Programming Assignment II RELDAT Documentation
April 3rd, 2017
CS 3251

RELDAT works similarly to the Go Back N procedure described in the book.

The client handles the majority of the logic. Upon starting, the client is given the ip, port, and window size. The client then asks the user to use a transform command or a disconnect command.

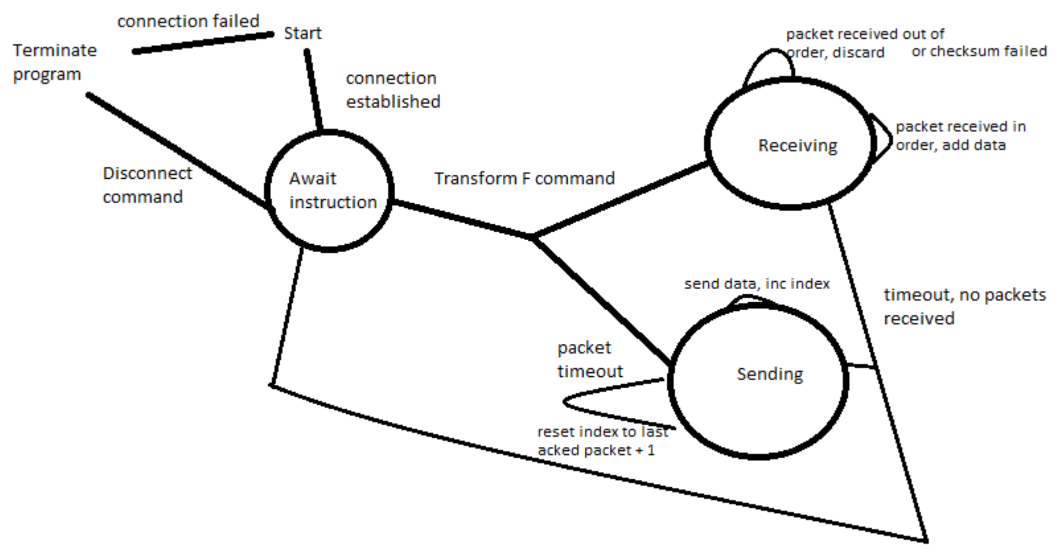
In the transform command, the client creates a thread to handle data that is received and a thread to handle sending data. The sending thread sends up to the window size in packets. Every time a packet is sent, a timeout is started for the packet after which if the packet has not been received, the index the sender uses to send the data is reverted to the index corresponding to the packet after the packet the server has last
received. Every time the client receives a packet, or a packet times out, more room is available in the send window and the client sends another packet.

The server is fairly simple. It uses a queue to ensure that it does not send more packets than allowed by the window. It then simply returns every packet it receives, if the packet arrived intact,
in uppercase to the client. Every packet has a string of data, an index, and a checksum all put together by pickle. Checksums are calculated using sha1.

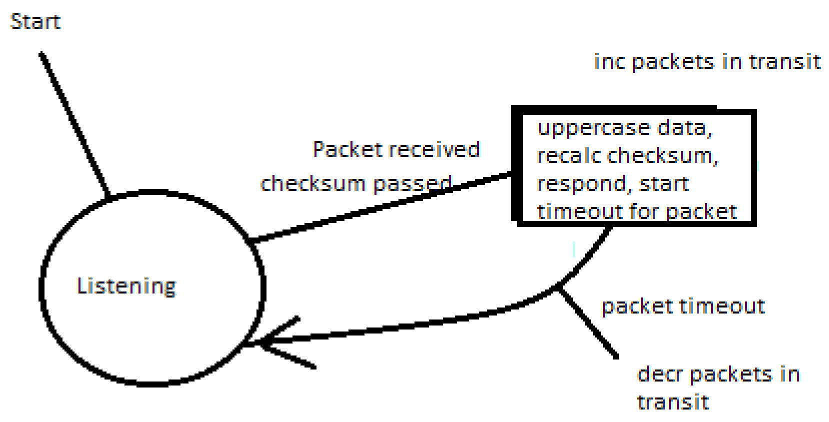
Packet design:
Every packet is pickled together using the pickle package from three parts. The first part is a string containing the data from the file to be sent in the packet. The second is an index, which describes where the packet belongs in order.
The third is the checksum, which is used to determine if a packet arrived intact.

FSM:

Client:



Server:



RELDAT establishes a connection by sending a message “syn” from the client to the server, from which the server responds with “SYN.” After connection is established, client awaits commands.

Connection termination is performed by calling the disconnect command on the client. When sending and receiving data during the transform command, if the receiver times out, the client reverts to awaiting commands. The server is always in the listening state after being started.

RELDAT discards any packets that are not the next packet that is expected and do not pass the checksum. If a packet times out, the RELDAT client will jump back to the packet after the last received packet, and begin sending packets again. The server simply responds with the altered packets as it receives them. Thus, RELDAT will only accept packets received in order.

RELDAT supports bi-directional data transfers by having the server respond to the client with packets. These packets contain the altered data as well as the index and a checksum, which the client can acknowledge to determine what to send next to the server. The server is simply reactive other than ensuring that it does not exceed the window size and the client controls all of the logic.

RELDAT uses pickling and packs the data from the file into packets of 1000 bytes including the index and checksum. RELDAT reads the file and adds the amount of chars necessary to fill up each packet to maximize efficiency. RELDAT also maintains the window size and will attempt to always have the window size of packets travelling in each direction.

The minimum packet size is determined by the size of the index and the checksum. 100 bytes are allocated for the header in each packet.