# UDP: RELIABLE DATA TRANSFER PROTOCOL

## Application Protocol:

The application is based on Reliable Data Transfer, stop and wait protocol. In our protocol, the client sends a datagram but does not send another until it receives the acknowledgement. After receiving the acknowledgement, the sender sends another datagram. Also I have implemented time out event which times outs if the socket is expecting an acknowledgement and has not received one. In that case, my application starts to resend the packet.

## Message Format:

Initially, I sent the command line arguments from the client to the server. In the command line arguments, I sent format type, file name and file length in the header. The server upon receiving the header information sends back an acknowledgement. After which the client proceeds to sends the data based on stop and wait protocol. The clients partitions the whole data into datagrams of fixed length. The last packet maybe of variable length.

The client sends the datagram, server may or may not receive the datagram. If server does not receive the datagram, the client side does not receive any acknowledgement and then sends the same packet again. On the other hand, if the server receives the datagram, the server proceeds to send the acknowledgement and client may or maynot receive the ack. If the client does not receive the ack, the client sends the datagram again until it finally receives the acknowledgement.

Initial header format:
[format] [file name length] [file name][file length]
Format is the format we want the input file be converted into. Format can range from 0 to 3.
File name length is sent to allocate space for file name in the server.

File name is the name of the file that we want the output file created by the server to have.
File length is the length of the input file.

Datagram format:
[frame size] [sequence number] [datagram]
Frame size is the size of the datagram. Frame size is 20 but the last datagram may not be equal to that number which I have used to my advantage in the application.
Sequence number is the sequence number of the datagram. It helps us in identifying duplicate packets. The value of sequence number can range from 0 to 1.
Datagram is the payload.

Use:
After creating the header files or the packet, the client sends the packet to the server through lossy_sendto function which uses two inputs in parameter, random seed and loss probability and determine whether the packet will be dropped or not based on some calculations.

Test Cases:

| Input File | Output |
|---|---|
| practice_project_test_file_1 | No error, format conversion occurred without any problems |
| practice_project_test_file_2 | No error, format conversion occurred without any problems |
| practice_work.c | Keeps sending files but does not change the format |

Instructions:

Compiling server

```
gcc -o dserver dserver.c
 ./dserver 9930 0.01 2234
```

Compiling Client

```
 gcc -o dclient dclient.c
 ./dclient 127.0.0.1 9930 practice_project_test_file_1 2 target 0.1 2234
```

Known Problems:
I have set my protocol to know the last datagram by telling the server to check if the last packet is less than the specified packet size which is 20. So if the original file length is divisible by 20, then my application will never get out of the loop and file processing will not occur.

Github link:
https://github.com/idiosincrasia/udp

Refrences:
https://stackoverflow.com/questions/13547721/udp-socket-set-timeout