Project Assignment 5

We were tasked with creating two scripts that will send and receive a file between by using multicasting similar to our code in Project Assignment 4 however with a larger payload. However the clients may or may not experience package loss when sending the file so a recovery algorithm needed to be implemented as well.

Steps taken: The specific steps for performing the discovery mentioned above are:
1. Created c beacon and client code to support multicasting
2. Developed algorithm for recovery
3. Added the code to specific nodes required
4. Ran client and beacon scripts on nodes
5. Collected data and answered questions about the lab

a. Our program detects datagram loss similarly to our Project Assignment 4 code. We implemented a method which simply expects the next package received will be last package N + 1. If it receives last package N + (2…N) instead then our algorithm flags that a package loss has occurred. If last package N - (2…N) is received by the client then it knows that the beacon is re-sending the file.

b. We choose the recovery algorithm to be an Airport Luggage Claim (Round robin) type of algorithm. The beacon continuously sends out datagrams in chunks of N bytes also containing the message ordering. The beacon also sends out a second message only used once by each client containing the total size of the file in chunks divided by N. The clients receives the total size of the file first in chunks of N bytes and then creates enough storage in memory to hold all of the packages expected. The client then continuously puts all packages received according to the package numbering in retrieved in said storage created. If a package was missed by the client at any given time then it will be received again at the next N iteration of packages sent by the beacon making sure that no synchronization is needed between clients.

c. The recovery mechanism works really well as the client can jump in at anytime during the beacon transmission of packages and retrieve the entire file sent.

d. The modification needed to the server code was to continuously send the file needed by clients in a loop divided into N chunks and send out two messages instead of 1. We choose two messages (1 small & 1 large) due to not wanting to impact the client file transfer initiation as the beacon can freely up the chunk size of packages sent without having to worry about loss rates.

e. There are some limitations on our client and beacon with the recovery algorithm. In extremely high package loss scenarios the client may need to wait for a long time to retrieve the last package as the chance of retrieving the last package is similar to the overall package loss chance for each iteration of sending the file. Changing package size and iteration speed does improve the overall speed of retrieving the file. The memory needed on the client side is equal to the file size received in chunks of N size which may have slight overhead by a couple of bytes. Another limitation is that the number may only contain N/2 chunks of numbers. If for example N = 4026 then the message may only contain 2013 characters and 2013 - 1 package orderings which then can represent 10^2012 different packages.

f. The recovery mechanism works well with any range of loss rates that aren't above 90%. If so then the final package may take a very long time to retrieve. Having a high loss rate around 50% and an extremely large file may also cause delays when the client is trying to retrieve the last package.

g. There is a slight overhead our recovery mechanism adds to standard unreliable multicasting which is the second message that is sent out which clients only use once before initiating file transfer retrieval.
h. The clients doesn't need to know about any other clients as they simply wait for their package to arrive and checks their own package storage after each package if the entire file has been successfully retrieved or if they still need to continue waiting.

i. The recovery mechanism scales with infinitely N clients recovering packages at the same time since there are no requests sent to the beacon to resend specific missing package possibly slowing down overall file transfer rates for other clients.

Proof of recovery mechanism:

Beacon output:

```
(base) Herculess-MacBook:groupProject5 HercHja$ ./beacon.o 7
Targetport used: 22207
IP used: 239.10.5.7
Num packages: 293
Server running!
```

Client output:

```
(base) Herculess-MacBook:groupProject5 HercHja$ ./client.o
Binding datagram to socket success.
Connected to multicast group.
Beacon package loop detected!
Datagram with order #: 117 recieved. Putting item in array location: 116
Progress: 1 out of 293 packages recieved
```

…..

```
Datagram with order #: 114 recieved. Putting item in array location: 113
Progress: 291 out of 293 packages recieved
Datagram with order #: 115 recieved. Putting item in array location: 114
Progress: 292 out of 293 packages recieved
Datagram with order #: 116 recieved. Putting item in array location: 115
Progress: 293 out of 293 packages recieved
Retrieved all data, disconnecting from beacon!
(base) Herculess-MacBook:groupProject5 HercHja$
```

Final output:

```
Datagram with order #: 116 recieved. Putting item in array location: 115
Progress: 293 out of 293 packages recieved
Retrieved all data, disconnecting from beacon!
(base) Herculess-MacBook:groupProject5 HercHja$
(base) Herculess-MacBook:groupProject5 HercHja$ ls
Project5datafile.txt    beacon.o                client.o
beacon.c                client.c                clientOutput.txt
(base) Herculess-MacBook:groupProject5 HercHja$ diff Project5datafile.txt clientOutput.txt
(base) Herculess-MacBook:groupProject5 HercHja$
```