

Kyle and Mati Project Proposal:

Design and Idea:

We are thinking of building a version of this default project in which we start with a central master server similar to what we learned about with Napster. When a peer connects to this server, it will tell the master a list of the files or data that it has stored, and gain access to information about what files other peers have stored. With this potential single point of failure design, we will add the option to include a backup server(s) that holds the data list of clients from the master in case the master goes down. When we implement this, we plan so that if the master goes down, the backup will simply become the new master. The backup server will ping the old master until it has come back online, and then tell the old master it is now the backup, sharing the data it is holding. Finally, for this fault tolerance, we will have the master blast out to each of the clients the new backup server, should this master fail at some point. When starting the original master up, we will have an option to declare how many backup servers we will have. Giving us the option for scalability, depending on how many expected clients will connect to the server. Then, clients can request data from this server, and it will begin to transfer data from peers. We plan to base the division of labor during a file transfer on all the clients that have the requested file (or portions of it) so for example if a client was requesting file A.txt and four other clients all had it we would expect the labor to be divided approximately equally among the four clients.

Technologies and Frameworks:

We plan to build our servers on similar Java xml frame work as we used in project 2 with the same sort of client facing capabilities. Since we have a master and specified number of backup servers on start up, we will need two different server types, one to be the master and one to be a backup. For our clients we are going to use a simple xmlrpc client like we created for project 2 and these clients will be run across a distributed network of computers provided to us by Prof. Barker. We also will need the time package to track how long client requests take during testing. We are not aware of any special packages we would need to transfer files or do any of the server work as of our current research and class discussions.

Timeline:

Checkpoint 1:

Work on setting up the master's code along with how the backup server will connect and store data.

Checkpoint 2:

Have the master be able to accept connections from clients who are requesting file transfers and be able to perform at least part of the transfer

Evaluation:

We will evaluate our system on the rate of data transferred per client helping in the transfer as well as look at the impact of the master server failing and backup taking over during a transfer. We will also compare the speed of transfers where clients have the whole file with the speed of clients that have incomplete portions of the file and mixes.

