# Comp50CP Final Project Initial Design

November 7, 2015

By Brinley Macnamara, Xuanrui (Ray) Qi & Benjamin Holen

Implementation:

1. Each node in the network will host a client and server process
2. Client
3. Server
    a. Built upon generic tcp server
    b. See Fig 4
4. Monitor
    a. See Fig 2
    b. Data Structures:
        i. A list of sibling Monitors to watch over
        ii. A table of the clients, their filenames, and the file's relevant hash so that the file can be downloaded to the client when they request it
        iii. A table of server processes so that they can be restarted and the client requests can be forwarded
    c. Utilizes the built-in supervisor utility to manage the various Servers
        i. Servers are dynamically monitored; as clients log in and out of the network the servers are started and stopped
            1. When a server goes down and needs to be restarted, there is a single Supervisor responsible for putting it back online. The server has a local directory where it has been keeping the stored files; it can reassemble the list of file hashes and be back in service.
    d. Utilizes co-monitoring of other monitors to allow for robust monitoring
        i. Handled by the built-in monitor module
            1. When a Monitor crashes and needs to be restarted, a sibling monitor will be tracking it. It is invariant that only a single sibling Monitor tries to restart a Monitor. The Monitor's state will then be reassembled by copying the state of the other monitors; they all share a state.
    e. Request Handling
        i. Clients do not send requests directly to the servers; incoming requests are handled by the monitors.
        ii. Upload Requests:
            1. The monitor will make an entry in its global list of file uploads using the user designated file name (necessary for retrieval), an md5 hash, and a client identifier
            2. It will then find a viable server and forward the request for a data upload to the server
                a. If a server cannot be found, it will send the request onto a sibling Monitor
        iii. Download Requests:

1. The Monitor will check the global list to see if such a file is in the distributed system.
2. The Monitor will then find a Server that has that file, and relay the client’s information along with the request so that the Server can handle the download.

Timeline:

1. Research - Due 11/5
    - a. Implement basic gen_tcp server - Brinley
    - b. Demo file library, MD5 hash, database - Ray
    - c. Implement basic monitor architecture - Ben
2. Implementation - Due 11/22
    - a. Write client, server, and supervisor interfaces (all running on one node)
    - b. Get client, server, and supervisor to talk to each other
    - c. Get client, server, and supervisor to send bitstreams between each other
    - d. Make things distributed!
        - i. Move supervisor to separate node
        - ii. Spawn many clients running on different nodes
    - e. Implement many monitors and client “clusters”
3. Add extra features - Due 12/9
    - a. Write encryption module
    - b. Client gui
    - c. “Striping” files across server nodes so that the file is stored in small chunks across many servers with some redundancy (ie. raid 5 or raid 10)
    - d. Version control