Comp50CP Final Project Proposal

October 24, 2015

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

Team Name: DistBackup

Objective: Build a distributed file backup service

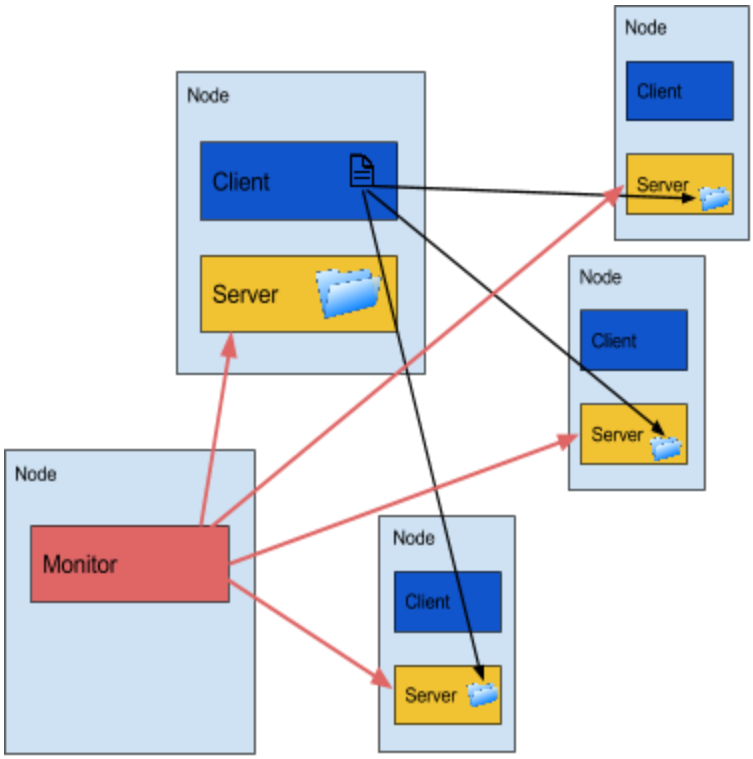
A distributed file backup service allows a client to upload a file to a network of servers, which will contain multiple copies of the client's file for safekeeping. When the client wants to re-download his file, he will be able to do so as long as one copy remains.

Language: Erlang

Core functions:

- Network of nodes, each with a process running a client and server
- Client process can send files to be stored on remote servers
- Server process listens for…
    - Requests to "post" a file -> stores file in data structure
    - Requests to "get" a file -> sends file to requesting Pid
- Network of node monitors ("trackers") to watch servers (and each other), alerting nodes if a server goes down

Picture:



“Stretch” functions:

- Client encrypts files so remote servers cannot look at them
- Client gui
- “Striping” files across server nodes so that the file is stored in small chunks across many servers with some redundancy

The biggest problems we foresee:

- *Problem*: *Sending files across the network.*
    - *Soln*: We will need to research Erlang’s tools to do this.
- *Problem: Managing redundancy*
    - *Soln*: Supervisors could have a list of files that each server has (the “master list”). When file is uploaded, it is added to supervisor’s “master list.” If a server goes down, and the supervisor notices a file’s redundancy has dropped below a certain threshold, the supervisor can increase its redundancy by telling a server that has the file to upload the file to another server.

Timeline:

1. Research - Due: 10/28
    a. Research Erlang OTP libraries for file handling (file), TCP server (gen_tcp)/SFTP server (ssh & ssh_sftp) (& encryption (crypto))
        i. Gen_tcp (more security) or gen_server?
        ii. File - handles file storage
    b. Find a way to distinguish files. I.e. each file needs a unique identifier.
        i. Use Erlang’s module, Crypto
2. Implementation (get core functions working)
    a. Write module to handle reading and writings files - Due: 11/4
    b. Define public interfaces for client and server - Due: 11/14
    c. Write client’s send and receive functions - Due: 11/22
    d. Write server’s callbacks and customer service apis - Due: 11/22
    Note: c and d should be done simultaneously
    e. Write monitor module for watching servers (and each other) - Due: 11/22
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)