E&CE 454: Spring 2012 Project 2
Design Document Due: 4:30 PM, 4th July 2012
Prototype Due: 4:30 PM, 22th July 2012
PeerBook

People today frequently have more than one computing device that they use in a non-trivial fashion. For example, they may have a desktop machine at work, a second such machine at home, a laptop, a netbook for travel and presentation purposes, a smartphone, *etc*. This leads to a problem: where are their files? In particular, the latest version of any given file could exist on any given machine; *e.g.,* a person may have made some last-minute changes to a file on their netbook just prior to giving a presentation; however, the netbook is probably not their main machine, and they likely have the most up-to-date versions of other files on one or other of their main machines.

Currently the typical solution to this is to have a back-end file server where all files reside; these days this is presumed to be in some "cloud" data center. The various devices then access the back-end file server over the network; the devices may or may not have additional local storage. There are several potential drawbacks with this approach, including

- network connectivity is variable and/or expensive (*e.g.*, consider the cost to access your files from your smartphone over 3G *vs.* over WiFi);
- users may well maintain local copies of files in a local file system, wrecking the overall approach;
- someone else may be controlling your data
- *etc.*

An alternate approach is to create a virtual distributed file system across the user's set of machines, presenting the user with a logical view of a file system, in which different logical versions of files only exist if the user has created such and wishes to access them as such. For example, the user may have files f_1, f_2, and f_3, where each file is available on each device. File f_1 may be some set of slides that the user prepared for a presentation; Version 1 of f_1 is the set that was presented at the first talk and the user may have decided to tag that as a stable version since it was used for a public talk; subsequent changes to the slides has resulted in version 2 of file f_1 being created; both versions 1 and 2 of f_1 would be available to the user from any device.

In such a virtual distributed file system the user may change some file on any device, but there is only one user; therefore concurrent changes will not generally occur; indeed, you can expect a reasonable delay between different devices being used to access any given file. That said, you cannot rule out the possibility that changes are made to some file(s) when a device is disconnected; further, you cannot assume that all other devices will be connected to the network at the time that a file is changed; therefore updates need to propagate as and when devices connect.

Clearly not every device can have an up-to-date copy of every file. In particular, smartphones lack sufficient storage, and netbooks, tablets, *etc.* may likewise be limited. Some decisions would have to be made as to which files should be made available where and when. Further, since most people already have computing devices and file systems, it is not reasonable to assume that all files will be immediately placed within such a system; some migration path from the current multi-machine environment would need to be envisioned. Third, it is unlikely that a user will never change their set of machines ever again; in particular, it is reasonable to assume that a user will acquire new devices periodically and retire devices periodically. Some thought will be needed in this regard.

Abstract File System

The API of any file system as a form something like this (though you are not constrained to this specific API):

```
int open(String filename, char operation);
int close(String filename);
int read(String filename, char buf[], int offset, int bufsize);
int write(String filename, char buf[], int offset, int bufsize);
```

where the possible open operations are 'r' (read) and 'w' (write). The integer returned are error codes and typically of the form1: 0 for correct operation, a positive number for a warning, a negative number for an error. For example, if a file is opened in "read" mode, any subsequent write operation should cause an error to be returned. A file opened in "write" mode may, however, be read as well as written to.

Requirements

You must design and prototype this virtual distributed file system. The design is worth 50% and the prototype is worth 50%. While there is a wide scope in how to approach this, some limits are that the system should be able to function when some devices are not connected; in particular, no device can be assumed to be always present.

There are a large number of unanswered questions in the above description. First, note that this is deliberate. The intent is to give you a lot of leeway to design the system as you think appropriate. Second, it is also intended to allow you to discover what problems are currently left unspecified that you need to resolve, and what issues you might choose to leave for a future version. Remember, as always, it is better to have a robust, but simple design, than one with many features than does not work well. You are not required, or expected, to resolve all issues in your design and prototype. However, you are expected to clearly identify what your design can and cannot do.

Finally, note that you are not required to have your prototype be as comprehensive as your design. Your design can think about and resolve issues that would take too long to prototype within the current timeframe. Please identify differences and limitations that your prototype has that are resolved by your design, as well as limitations of your design.

You should submit your design document and all code you write for your prototype separately by the deadlines. Your design document should *clearly* describe the limitations of your system.

You will be expected to demo your prototype to the TA.