System Programming Assignment 3

**Announced: Wednesday November 13, 2013.**

**Due: Tuesday November 26, 2013 23:59.**

**Synchronize Clocks of Clients and Servers**

**Introduction**

The goal of this programming assignment is to synchronize the clock on clients and server to assure that the time attributes of the files and directories are consistent. There are three tasks: synchronize the clocks among multiple computing devices, implement concurrent sessions on server, and link additional clients to the server.

In this homework, we do preparation for extending the architecture to multiple clients. **Requirement**

**1. Implement a clock synchronization section after login section.**

Because the system environment of clients and server may be different, the clocks on these devices could be out of sync for time being. One possible cause is clock skew[[1]](http://en.wikipedia.org/wiki/Clock_skew). Hence, we have to synchronize those information before providing service between all of them.

In distributed systems, there are numbers of clock synchronization protocols/algorithms. You may use the one described in [[2]](http://en.wikipedia.org/wiki/Network_Time_Protocol#Clock_synchronization_algorithm) to estimate the time difference among computing nodes. (Note that you may use different algorithms for this purpose.) To overcome clock skew among computing nodes, the server should update the time difference when the client connects to the server. (Note that this protocol leads to unnecessary clock synchronization overhead. There are other algorithms to avoid the unnecessary overhead.) Furthermore, the server should keep track of the time difference to avoid clock synchronization for every transmission between client and server. To summarize, the first task has the following three requirements.

1. Implement Clock Synchronization algorithm [[2]](http://en.wikipedia.org/wiki/Network_Time_Protocol#Clock_synchronization_algorithm) to estimate time difference between server and client.
2. Time zone supports: client and server may be located at different time zones.
3. Server stores all time difference of clients and use it when synchronize files. For example, the clock difference is shown below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | server | +0 | | client1 | +5 hr | | client2 | -5 hr | |  |

When client1 uploads a file which mtime is 12:00:00, server will change the mtime to 7:00:00 after received. After that, server synchronize the file to client2 with telling client2 changing the mtime to 02:00:00.

**2. Implement multiple I/O at server side and client side**

So far, the architecture of client and server can only handle one CSIEBox synchronization session at any time. In this assignment, you will extend the earlier version to handle multiple concurrent CSIEBox synchronization sessions. On server side, the server process handles requests from multiple client processes, and the client process handle requests from server process and inotify.

You should modify cisebox\_server.c::csiebox\_server\_run and the function which waits for inotify event. Use select to make them be able to handle multiple I/O.

**3. Implement new client link and file download**

When a new client is linked to the server, the server process will send files to the client and the client process should rebuild the directory tree. (The process is the reverse process of MP2.)

**Hints**

When implementing the homework, you may face a “bouncing” problem. Below is an example of bouncing problem. Assume there are two clients connect to a server with the same account.

1. client C1 update

2. client C1 inotify event triggered

3. client C1 update to server

4. server push change to client C2

5. client C2 update file

6. client C2 inotify event triggered … looping 1-6

You can assume that the user will modify the content of one file on at most one client device and the other client processes only **receive** updates from the server. Once a client process receives updates from server, the client process can ignore the succeeding inotify event to avoid *bouncing problem*.

**Testing**

For testing, you can hardcode a time difference in client side to test your program. TA will use VM to testing them.

TA will use testing scenario below.

1. Start server

2. Start client C1 with some files in home directory

3. Start client C2 with an empty home directory

\* The accounts of client C1 and client C2 are the same.

\* At the moment, client C2 will download files from server.

4. Touch a file in client C1 home directory

5. Stat the file in client C2

**Grading**

1. clock synchronization (2pts)

2. client and server multiple I/O (2pts)

3. client download (2pts)

4. Integration test (2pts)

**Contact**

email: [r01922084@ntu.edu.tw](mailto:r01922084@ntu.edu.tw)

or

p2: wmin0

Please use prefix [SP\_HW3] at the title. Thanks.