CMSC-621, PROJECT 2(BERKELEY'S CLOCK SYNC)
SHREYA DATE (PQ56297)

CODE DESIGN

The project Berkeley's Clock contains the following files:

1. TimeDaemon: contains the daemon class. Programmatically, it implements socket client APIs.

2. Server: This is the individual process/server in the distributed system whose clock is queried and synchronized by the time daemon. It uses the socket-server APIs.

3. Socket: contains wrappers over the socket system calls

4. SocketDef: contains structure for socket data

5. ServerMain.cpp : This is the process which kicks off server creation by using methods exposed by the Server class.

6. TimeDaemonMain: This is the process which kicks off the daemon tby using methods exposed by the TimeDaemon class.

7. input.csv : contains port number and server name which is input for the processes in the distributed system.
8. multiple_processes.sh: This shell script is used to spawn multiple Servers/Processes in the distributed system. The port number provided as parameter while running each server should be an entry in the input.csv file.

**Note that because of heavy modularization, code workflow is pretty clear and hence not commented much.

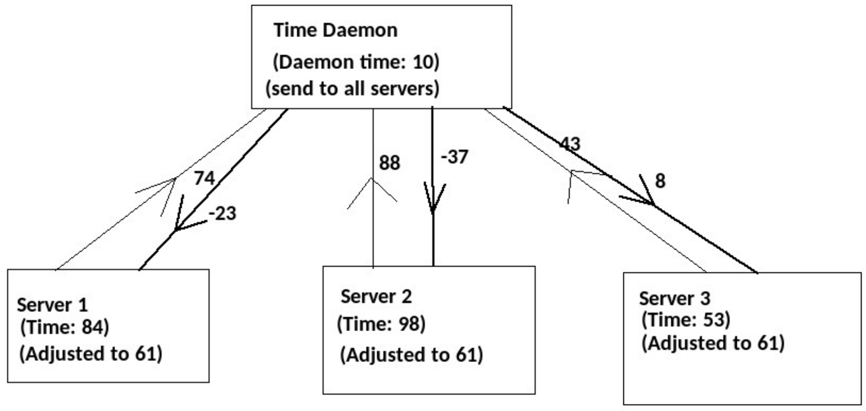
CODE WORKFLOW

Time Daemon:

1. Time daemon implements the socket client APIs
2. It connects with each server in the distributed system.
3. Time Daemon's clock is taken as input from the command line.
4. It sends it's clock to the servers in the distributed system.
5. It receives the time difference of each server.
6. Calculates Average (time_difference / total number of processes in the system including the daemon)
7. It calculates drift for each process in the distributed system.
8. Sends the drift to each server.
9. Calculates its own adjusted time.

Servers/Processes
1. Each server receives a connection request from the daemon,
2. It then sends a dummy “Hi” message to confirm that connection is successful.
3. It receives Time Daemon’s clock.
4. Calculates how much its own clock deviates from that of the time server.
5. Sends the offset to time daemon.
6. It receives the value by which it should adjust its clock from the time daemon.
7. Adjusts its own clock and prints it.

BERKELEY IMPLEMENTATION



LEARNINGS

Implemetation of Berkeley’s clock synchronization algortihm using sockets.

ISSUES

1. First needed to create a distributed system and design it well.
2. For example, time server which is the daemon had to implement client socket APIs and the other processes were multiple servers listening.
3. Had to record offset for each process and then calculate average.