Kierstin Matsuda, Kevin Garcia, Eric Cao

COP 5614 – Introduction to Operating Systems

Group 9 – Assignment 2

Professor Dong Chen

Wednesday March 27th

Smarter Home Design Document

Our assignment, Internet of Things - Smarter Home Edition, contains nine files, with one file for each device, sensor, server, and user. Each sensor, device, and user are given a unique integer ID that is used to verify that each message is sent to the correct executable. In addition to the previous assignment, there are now two new sensors named door and beacon. The door has two states, open and closed, and the user can query its status. The beacon sensor will send true if it senses the user's keychain in the house and false if it is not present in the house. The beacon is useful for determining if there is an intruder or the user in the house. This assignment was written entirely in Java.

Based on the instructions for assignment 2, we have split the gateway server into two tiers. The front tier server will handle all the queries and commands from the user and will store any changes into the back tier database server. The back tier server will also keep track of all attached sensor and device state changes. The database will keep the prior changes in a text file, with the title of each file being the time it was created. We used the bully algorithm to select the leader for clock synchronization. For clock synchronization, we used the Berkeley algorithm. We also used logical clocks, specifically Lamport clocks. Each node has a clock offset variable called timeOffset, which holds how much each clock has to be adjusted based on the Berkeley

algorithm in seconds. The actual clocks are not changed, but the offset is used to change their timestamps.

By having this clock synchronization, we can use this for event ordering. Now when the door is sensed to be open followed by motion sensed in the room, the smart home system will tell the user that the user has entered the house, the alarm system is set to HOME, and the lights will turn on due to motion sensed. On the other hand, if motion was sensed first then the door is opened, then the user has left the house and the alarm system is set to AWAY. Additionally, we will also use the beacon sensor to differentiate this event between the user and a possible intruder; if the beacon is not present when the following event occurs, then the smart home system will notify the user that there is an intruder at home and the lights will not turn on.

This assignment works by taking advantage of Java's ability to use multithreading, so we were able to use threads that independently handled each client on the server. This allowed each thread of the server to communicate with one another using shared memory. The program also uses sockets to communicate with each respective sensor, device, and user.

One problem we would like to fix in the future is that the thread created by the server when a user connects will not terminate if the user process abnormally exits. Two possible improvements are making the user know when the server has stopped running and vice versa. This will prevent any hanging executions from continuing to run. That way, the user can keep track of any problems or bugs.

In order to run the program, you first need to make sure that all programs have the same port and each client has the IP address of the server. The IP address and port of the server can be changed in all client programs by modifying the "SERVER_IP" and "PORT" constants at the top of each class. Also make sure that java and make are installed. As well as making sure that the

server is accepting incoming and outgoing requests on that port and IP. This would ensure that the server can receive and send information to the clients. Once that is completed, run the command “make” to compile all the files. Then run the files by using the command “java <Filename>”. For example, to run User.java, you need to enter the command “java User”. Run the server file first, then run all the clients. Each client can be run on any computer or directory. Once all programs are running, you should see the menu with options the user can choose, and occasional messages from the server to the user program. The messages will differ based on what options were chosen; for example, you should not see a message about an intruder if the security mode it set to HOME. Once you are done with the program, simply choose option 14 in the menu to exit. This will stop User.java, but the other programs will continue running. To stop the other programs from running press Control-C.