2020

COMS 487-Programming 1

DESIGN DOCUMENT

RAED ALBLOUSHY

Agent:

Running the agent you either give an userid in the command prompt like ./agent 4000 or without a custom id ./agent

Written in C language, consists of 5 main functions, GetLocalOS(), GetLocalTime(), beacon(),beaconSender() & CmdAgent().

beacon(): this is the function responsible for initializing my beacon that is to be sent to the manger, my data format for the beacon is *id* **c** *startuptime* **c** timeinterval **c** ip **c** cmdport , the c in between is a used as delimiter between the required information in a beacon and the actual beacon being sent as a character array. Given that information,

An example of a beacon would be :

5000c189090900c60c127.0.0.1c3000

decoding this beacon we would get

ID: 5000, startuptime=189090900 time interval=60 ip 127.0.0.1 cmdport=3000

In my beacon initialization if a user gives an id then we use that otherwise we generate a random number from 1000 to 10000. The startup time Is taken as timestamp from time() function. And the time interval is always going to be 60 secs and ip as local host, the cmdport is also randomly generated as number from 4500 to 6000.

beaconSender(): a function used in a thread to keep sending our beacon each 1 min. it just has function called sendto used to send the character array Str1 initialized by our beacon function . this act of sending using our established UDP connection with the manger using port 4444.

CmdAgent(): this is the function for the thread handling the Incoming TCP connection and the port is determined by the CMD port randomly generated. This function receives two commands one is for getting operating system and the other is to get local time. if the character 'o' is received that means that the function GetLocalOS() is called and a character array is filled and we send the response back to the manger , if the character 't' is received then the GetLocalTime() function is called and makes a response that is stored in a character array that is sent back to the manger.

Manger:

Running the manger, we do not take any parameters.

Written in Java language, consists of a main Manger Class that has three inner classes for the threads BeaconListener (), AgentMonitor(),ClientAgent(). The UDP connected agents are managed by an arraylist of string arrays storing their first sent beacon, that arraylist is called agents. BeaconListener () and AgentMonitor threads are started as soon as the manger starts running.

beaconListener(): this function is responsible for accepting incoming beacons using UDP and the UDP connection is hard coded on port 4444, I accept the incoming beacon in a byte array and then I convert it to a string and finally I decode the string using .split() in java to split our beacon using our char ‘c’ that was used as a delimiter . I store the incoming beacon in a string array, agent[0] = id, agent[1]=startuptime, agent[2]=timeinterval, agent[3]=ip and agent[4]=CMDport and I add another element to the beacon which is the last time a beacon was time transmitted from this agent, that is stored in agent[5] and I do this to see if an agent dies in the AgentMoniter thread. Once my beacon parsing is done , if this a new agent beacon then it is pushed to my agents arraylist otherwise it I update my last time of beacon transmission in agent[5]. I also check the if there is a different startuptime with same id for any of the existing beacons in my agents arraylist and if so I inform the manger.

AgentMonitor(): this thread is keeps checking the difference of the last time a beacon was received for each registered agent in the agents arraylist and if any of the agents has not transmitted for 2*thetimeinterval specified in the beacon then the agent is considered dead and removed from the list and the manger is informed.

ClientAgent(): this is the thread to connect using TCP to our agent and this thread takes the CMD port as a constructer and connects to that port. This thread is only spawned in the beaconListner() thread when it is determined that a new beacon has been received. I have a loop in the clientagent() thread that sends in first the character ‘ o’ to request the command for the local operating system and then once it gets a response, it prints that and sends the character ‘t’ to request the local time information from agent and once that is received, I break out of the loop and note the time it took to do both these requests.