CS4310: Computer Networks

Project #2

Due Dec 4 -- 11:59 PM (firm)

Please start early. Late submissions would incur a penalty of 20% per day for up to 2 days, and then they will not be accepted. Please read the description carefully and come see me (hopefully early) if you have any questions.

1. Overview

In this project, we are going to build a component of a Command and Control (C2C) server that is used to allow agents in a multi-agent system to become aware of each other. In particular, the C2C server should be able to maintain a list of active agents and respond to various actions as issued by the agents. For example, the agents could be a group of UAVs and the C2C server could be a satellite. The agents would need to communicate through the satellite if they are out-of-range of each other. The C2C server will run on a particular port (supplied by a command line argument) and should be able to respond to TCP connections from agents. Agents send various actions to the server to update the information maintained. The agent code is supplied in this assignment and is written in C. An agent can issue one of those actions:

1- #JOIN

When the C2C server receives a join request, it will add this agent to its current list of active agents and responds with “$OK”. An agent can only join once before it leaves. If an agent tries to join while it is already an active agent, the C2C server should respond with an error message “$ALREADY MEMBER”

2- #LEAVE

When the C2C server receives a leave request, it deletes the corresponding agent from its list of active agents and responds with “$OK”. An agent cannot leave if it has not already joined and the server responds with an error message “$NOT MEMEBER”

3- #LIST

When the C2C server receives a list request, it responds with the list of all active agents (agents that have joined, but not already left). The C2C server can only respond to list actions from active agents (otherwise ignored and no response is given). The list contains the IP address and duration in seconds (how many seconds since an agent joined) for each agent. Each entry should be in the form

of <agent_IP, seconds_online>. You may assume that each agent has a unique IP. Requests generated from the same IP address are treated as a single agent.

4- #LOG

When the C2C server receives a log request, it responds with the log file (log.txt) to the agent. Similar to list, only active agents can issue log requests. You can assume that the log requests itself should be logged into the log file before its gets sent to the agent. Note that the log file could be quite large, so make sure not to try to load it into a buffer all at once.

The agent takes three arguments in its command line: the C2C server IP, server port and the action string. It sends the action string and writes the response, if any, to the screen. The agent code is listed under resources on TRACS.

The agent would connect to the server on its listening port and waits for a response to its actions before it exits. All actions start with a '#' and all responses start with '$' followed by the data/message.

The C2C server should be able to listen for actions on the port it is running on and once a request is received, the server responds with the information/confirmation based on its current state (e.g., active list of agents, when they joined and the agent requesting the information) and records its interactions in the log file. Notice that the C2C server will never exit; it continues listening and servicing requests.

2. Log File

The C2C needs to maintain a log file the records any information about any given connection along with a time stamp (up to millisecond granularity) and the response sent (if any). You can assume the file is called log.txt. For example:

"TIME": Received a "#JOIN" action from agent "147.26.143.22"

"TIME": Responded to agent "147.26.143.22" with "$OK"

"TIME": Received a "#LIST" action from agent "147.26.12.11"

"TIME": No response is supplied to agent "147.26.12.11" (agent not active)

3. Where to start

First, you need to get familiar with Internet programming API. This will be based on the material discussed in class.

Given the agent application provided, start working on the server part. To test your C2C server, you need to generate multiple actions from different agents by running them

from different machines. You can run a number of agents from different terminals, but all of them will have the same IP address and thus would be treated as the same agent. You need to handle different cases such as a “#LEAVE” without joining, a “#LIST” from an inactive agent, among many others.

4. Submission

- You need to turn in a short design document stating the overall design decisions you made, and the data structures used.
- Please provide a single command line to test your code and make sure to provide a working example of the command line.
- Please provide the output log file for 2 or 3 agents (using different terminals from different machines such as zeus, eros and your machine) that perform the following actions in that order:

  1- Agent 1 joins
  2- Agent 2 joins
  3- Agent 3 joins
  4- Agent 1 issues list
  5- Agent 1 leaves
  6- Agent 1 issues list
  7- Agent 2 leaves
  8- Agent 2 leaves
  9- Agent 3 issues list
  10- Agent 3 issues log

5. Notes for future extensions [not required]–

In a real-world setting, we will need to add a layer of authentication in which the C2C server can verify that it is really communicating with an agent (as opposed to an attacker) and vice versa. Also, communication should be encrypted between them. Finally, the server should be a multithreaded one, so it can respond to various agents in parallel (for example by using pthreads).