Adam Składanek g. 11c s16787

SKJ project 1

Sailor Game – TCP p2p network

1. A general solution description

The goal of the project was to create a network of agents playing a tournament of sailor game: two players pick a number and count starting from one of them which one won. Agents could join and quit any time and every agent had to play with every agent. As additional tasks the game must be fair, and there has to be a monitor (http server) that can easily display results of matches played.

To properly run the project the monitor should be set up first. Source files can be found in folder *SailorGameMonitor.* After running the program, it should ask for specifying its ip address and port separated by : . If the proper address was entered, it should display a message:

```
Enter my ip and port [ip:port]
127.0.0.1:8888
Monitor is listening on /127.0.0.1:8888/monitor
```

After setting up the monitor the main application can be turned on. After compiling and entering IP and port of the monitor (again separated by :) there should be a list of available commands.

To add an agent enter n, then enter agent's name, IP, port and IP and port of the agent in the network to connect to (all separated by spaces). If the host agent is to be created, ip and port of the agent to connect to should be the same as IP and port of the agent. Whenever an agent joins the network, all members of the network will play a sailor game with him. This way every agent will play a match with every agent. You can also force 2 agents to play each other by entering p and agent names (the first one has to be local, but the second can be from any network). By entering r and an agent's name an agent can be removed from the network. When he is removed, after making sure he played every agent in the network and sending request to leave, he prints his match history and is deleted. Application also allows for displaying a list of agents with l, changing monitor address with m and removing all agents from the network and closing the application with q. The monitor should display results of matches. Examples:

```
Sailors Game - SKJ Project
Enter monitor ip and port (leave empty if monitor is not running) [ip:port]
127.0.0.1:8888
l - get list of agents
n - add new agent
p - play as an agent with another agent
r - remove agent
m - set monitor address
h - get list of commands
q - quit
n
Enter agent name, ip, port and point to connect to [name ip port connect_ip connect_port]
p1 127.0.0.1 11111 127.0.0.1 11111
p1 127.0.0.1 11111 is a host
p1 127.0.0.1 11111 is listening...
n
Enter agent name, ip, port and point to connect to [name ip port connect_ip connect_port]
p2 127.0.0.1 22222 127.0.0.1 11111
p2 127.0.0.1 22222 is listening...
p2 127.0.0.1 22222 is sending: JOIN p2 127.0.0.1 22222
p1 127.0.0.1 11111 recieved JOIN p2 127.0.0.1 22222
p2 127.0.0.1 22222 received: [p1 127.0.0.1 11111]
p1 127.0.0.1 11111 is sending: PLAY p1 p2 EHpsorPD58UPU00pstGYcwkHdPeI3JEuizAGwENfJSo=
p2 127.0.0.1 22222 recieved PLAY p1 p2 EHpsorPD58UPU00pstGYcwkHdPeI3JEuizAGwENfJSo=
p1 127.0.0.1 11111 received: p2 p1 30259
p2 127.0.0.1 22222 received: p1 p2 96270
p1 127.0.0.1 11111 received: p2 p1 30259
p2 127.0.0.1 22222 - 30259 | p1 127.0.0.1 11111 - 96270 | 126529 won
p1 127.0.0.1 11111 - 96270 | p2 127.0.0.1 22222 - 30259 | 126529 lost
Monitor recieved the message
p
Enter a local agent name, name of agent to play with and your number
p1 p2
p1 127.0.0.1 11111 is sending: PLAY p1 p2 80V9q+G0Eu1jdNVv6P47lpx2G3fcyA7MCWS3x2QdIZs=
p2 127.0.0.1 22222 recieved PLAY p1 p2 80V9q+G0Eu1jdNVv6P47lpx2G3fcyA7MCWS3x2QdIZs=
p1 127.0.0.1 11111 received: p2 p1 110571
p2 127.0.0.1 22222 received: p1 p2 576
p2 127.0.0.1 22222 - 110571 | p1 127.0.0.1 11111 - 576 | 111147 won
p1 127.0.0.1 11111 received: p2 p1 110571
p1 127.0.0.1 11111 - 576 | p2 127.0.0.1 22222 - 110571 | 111147 lost
Monitor recieved the message
r
Enter agent name
p2
p2 127.0.0.1 22222 is quitting.
Scores:
p2 127.0.0.1 22222 is sending: QUIT p2
p1 127.0.0.1 11111 recieved QUIT p2
[p2 127.0.0.1 22222 - 30259 | p1 127.0.0.1 11111 - 96270 | 126529 won, p2 127.0.0.1 22222 - 110571 | p1 127.0.0.1 11111 - 576 | 111147 won]
```

2. Precise solution description

Each agent can send and receive four types of signals: *JOIN*, *ADD*, *PLAY* and *QUIT.* A join request is sent whenever an application creates a new agent. The request contains agent's name, ip and port, the response holds names, addresses and ports of all agents that were connected to the network. After the agent is created, it sends a join request to the agent it wants to connect to, and the other agent responds with a list of agents he knows about. If the agent name is already taken, it returns an empty list.

Before an agent responds to a join request, it sends an add request to all connected agents. It contains name, IP and port of the agent that wants to join the network. After receiving an add request an agent adds the agent specified in the request to it's list of known agents.

Whenever an agent receives a join or an add request, or is forced to play via console input, it sends a play request to an agent who has joined the network. Play request contains sender, receiver, and a hashed (SHA-256) and encoded (base 64) number from 1 to 1000000. Agents respond to a play request by sending a response with a sender, receiver, and a number from 1 to 1000000, but not encrypted. When it is sent, the other agent sends a non encrypted number which he has sent before. Then the receiver checks if the hash from the number matches the hash sent before. This is to ensure that the game is fair, and the players do not

know each others numbers until they have both sent them. If the hashes match, 1 is sent, the game is saved and sent to the monitor. Otherwise the game does not count.

An agent can quit the network. First it checks if it has played at least one match with each of the agents in the network, then it sends a quit request containing its name and prints results of matches played. When another agent receives such a request, it deletes agent with a specified name together with history of matches played with him.

3. Observations, experiments and conclusions

The program works on multiple machines connected to a local network as long as the IP addresses are entered correctly. There can be multiple host on a single instance of a program and it can connect to many local networks at the same time.