ID1217 CONCURRENT PROGRAMMING

ASSIGNMENT -5

Distributed Computing with Message Passing

SUBMITTED BY-
KISHORE BAKTHA

PROBLEM 1-

Distributed Pairing 1 (a client-server application)

In this application, the aim is to have a server(teacher) to assign partners to different students(clients). First the server is started in port 8000 and waits for connections. The client is then started by calling 7 different threads and connecting to the server at port 8000.

The client sends a request to find the Lab Partner. The server has a synchronized object to control access the socket list. If the list is empty, the client socket is added to the list. Else, the first socket in the list is fetched and assigned to the current client socket. Once the socket is assigned, the server sends a response to both the partners. The process then repeats for the other clients until all the clients have been assigned partners.

The standard input and output stream is used for message passing. In addition, Gson library is used to serialize and deserialize the json objects in order to simplify the process of matching the client name to the client ID.

Output-

Client-

David 1: I've received a lab partner! {"labPartnerName":"Sam","labPartnerId":0}

Sam 0: I've received a lab partner! {"labPartnerName":"David","labPartnerId":1}

Rahul 3: I've received a lab partner! {"labPartnerName":"Aswin","labPartnerId":2}

Aswin 2: I've received a lab partner! {"labPartnerName":"Rahul","labPartnerId":3}

Vivek 5: I've received a lab partner! {"labPartnerName":"Rohit","labPartnerId":4}

Rohit 4: I've received a lab partner! {"labPartnerName":"Vivek","labPartnerId":5}

Mohit 6: I've received a lab partner! {"labPartnerName":"Mohit","labPartnerId":6}

PROBLEM 2-

Distributed Pairing 2 (a peer-peer application)

In this application, we again a set of clients and a single server. But the server just sends one message to the client and stops. The client processes then act as peers to form partners among themselves. In the server side, first the server waits until it accepts all the requests and sends a response back to all the clients. Then each client wait for a request from the server. The server has a list of ports that the clients are assigned to and picks up a random port and sends a request to that client and stops.

The first client receives a list of ports and then chooses some random port as its partner and then sends a request to that port. The port gets the request, generates a response and then sends it back to the client. Once the two ports

have become become partners, they are removed from the portlist and then a new request is started again to form the next set of partners.

The selection of partners is random has the ports is chosen randomly from the list of ports. Also, the server just sents one request and the other clients communicate with each other to form partners.

OUTPUT-

Server-

```
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
Starting to accept connections
Socket Created
{"requestType":"portRequest"}
{"requestType":"portRequest"}
{"requestType":"portRequest"}
{"requestType":"portRequest"}
{"requestType":"portRequest"}
{"requestType":"portRequest"}
{"requestType":"portRequest"}
 [8001, 8002, 8003, 8004, 8005, 8006, 8007]
{"portList":[8001,8002,8003,8004,8006,8007],"requestType":"FindLabPartnerR
equest"}
DONE!
```

Client-

```
{"port":8001,"requestType":"portResponse"}
```

{"port":8007,"requestType":"portResponse"}
here
{"port":8006,"requestType":"portResponse"}
here
here
{"port":8003,"requestType":"portResponse"}
here
{"port":8002,"requestType":"portResponse"}
here
{"port":8004,"requestType":"portResponse"}
here
{"port":8005,"requestType":"portResponse"}
here
{"portList":[8001,8002,8003,8004,8006,8007],"requestType":"FindLabPartnerRequest"}
{"studenName":"Rohit","studentId":4,"requestType":"LabPartnerRequest"}
Rahul 3: I've received a lab partner! LabPartner: Rohit
{"labPartnerName":"Rahul","labPartnerId":3,"requestType":"LabPartnerResponse"}
Rohit 4: I've received a lab partner! LabPartner: Rahul
{"portList":[8001,8002,8003,8006],"requestType":"FindLabPartnerRequest"}
{"studenName":"Mohit","studentId":6,"requestType":"LabPartnerRequest"}
Aswin 2: I've received a lab partner! LabPartner: Mohit
{"labPartnerName":"Aswin","labPartnerId":2,"requestType":"LabPartnerResponse"}
Mohit 6: I've received a lab partner! LabPartner: Aswin
{"portList":[8003,8006],"requestType":"FindLabPartnerRequest"}
{"studenName":"Sam","studentId":0,"requestType":"LabPartnerRequest"}
David 1: I've received a lab partner! LabPartner: Sam
{"labPartnerName":"David","labPartnerId":1,"requestType":"LabPartnerResponse"}
Sam 0: I've received a lab partner! LabPartner: David
{"portList":[],"requestType":"FindLabPartnerRequest"}
Vivek 5: I'm partner with myself!