2012

Universidad Carlos III de Madrid

Antón García Dosil
100277382

[LAB ASSIGNMENT 2: PROCESSES COMMUNICATION WITH SOCKETS]

Development of a multithreaded server, and a client using TCP sockets using C programming language for server and Java for clients

Table of Contents

1. PROGRAM DESIGN

1.1. SERVER GENERAL DESIGN

The main challenge for this assignment is to design the implementation of the multithreaded server over one single socket.

In the previous assignment I had incorrectly believed that the server could not be made multithreaded over the same port. This issue has been corrected in this assignment.

The main thread will be waiting passively for client requests. Once a connection is accepted, a new thread to dispatch the service will be created.

Once the service thread is created, the main thread will halt waiting for new connections, allowing several to be served at the same time.

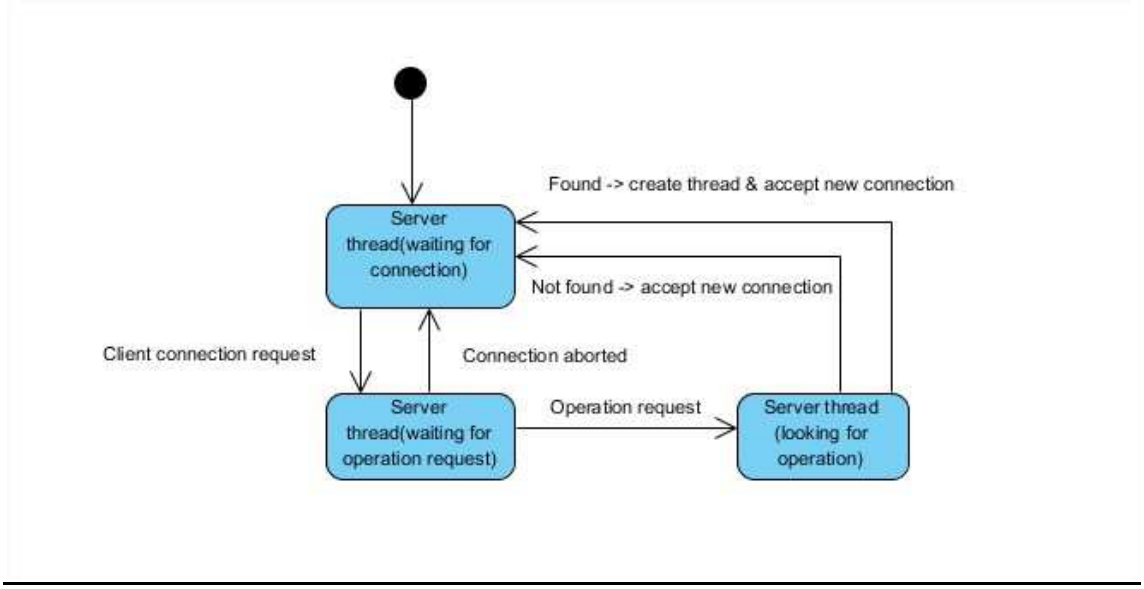

Image 1 Petition serving model.

1.2 CLIENT GENERAL DESIGN

The client design is simpler than the server side. There will be one method for each service request; handling connection to the server and package sending/receiving will be done separately in each method.

Server will fulfill the request through the according service thread.

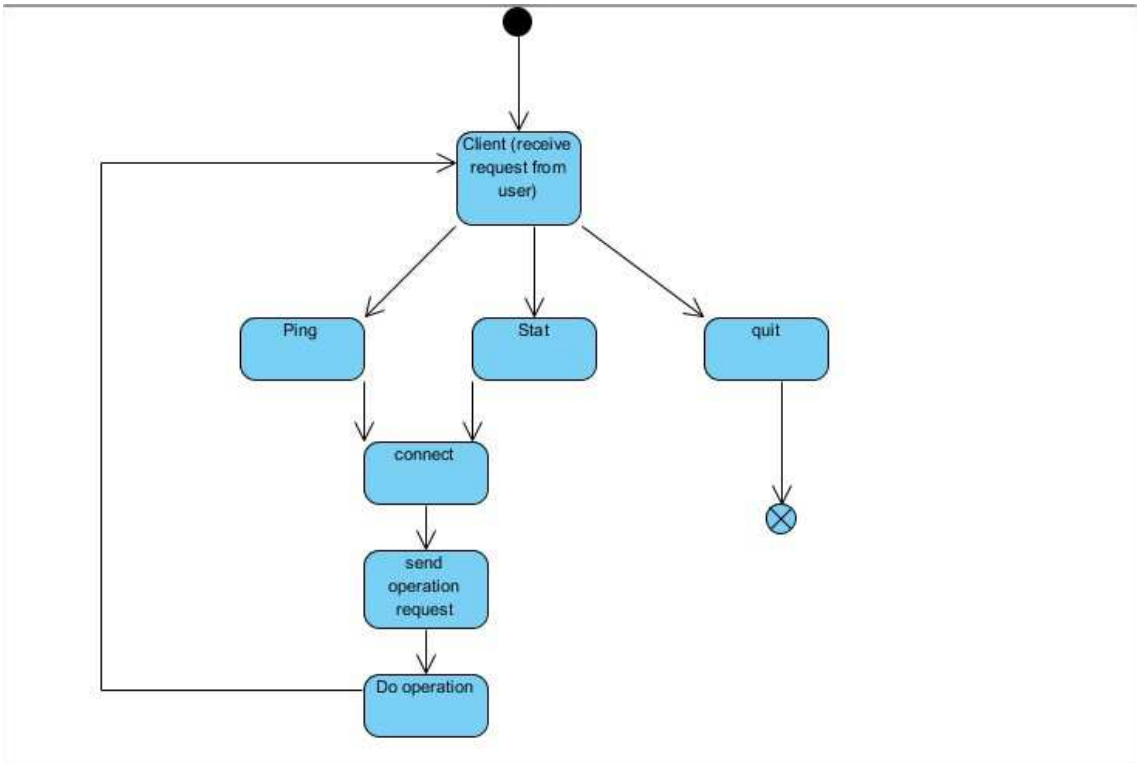

Image 2 Client model.

2. SERVICE DESCRIPTION

2.1. PING

The ping service is designed to give an immediate response to verify the connection between the client and the server. The client sends its IP and port to the server and receives a single byte in response.

2.2. SWAP

Not implemented due to problems with long sending

2.3. HASH

Not implemented due to problems with long sending.

2.4. CHECK

Not implemented due to problems with long sending.

2.5. STAT

Each time a service is requested, it will be logged in numeric format in a structure called st. When stat is called, each value of this structure is sent to the client, that will see the times each service has been called.

3. TESTS

The following tests have been made to verify the correctness of the solution. For the ping service only the general test cases have been applied, due to the development in earlier stages of the practice.

3.1 GENERAL TEST CASES

-To assure the connections are well done, services must be able to execute concurrently.

Tested creating stalls in server and creating more client connections. **OK.**

-The server must be able to communicate with different clients.

Make one/many requests to the server, exit the client and make a request from another client thread. **OK.**

-**The server must only exit when terminating via control + c.**

Test with many requests, checking if the server exits or not. **OK.**

2.5 STAT TEST CASES

-Check that the values given by stat service are correct.

Make a fixed number of requests and check if the stat response is correct. **OK.**

4. TASKS DONE

- **DEVELOP CLIENT AND SERVER FOR PING SERVICE - 5H**
- **DESIGN OF MULTITHREADED SERVER – 2H**
- **IMPLEMENTATION OF MULTITHREADED SERVER – 3H**
- **IMPLEMENTATION OF SERVICE FUNCTIONS AND CLIENT SERVER COMMUNICATION - 6H**
- **REPORT – 2H**

5. PERSONAL CONCLUSIONS

I have encountered many difficulties in communicating my C server and my Java client due to the issue of Big Endian vs Little Endian and signed vs unsigned variables. Although I haven't finished the practice, I have learned how to implement a concurrent server and that endianness varies in different circumstances. It was demoralizing to use around 5 java classes to send and receive data and none of them working although I used htons/htonl and ntohs/ntohl on server side.

I hope I may make up the mark lost on this practice with satisfactory RPC and web service hand-ins.