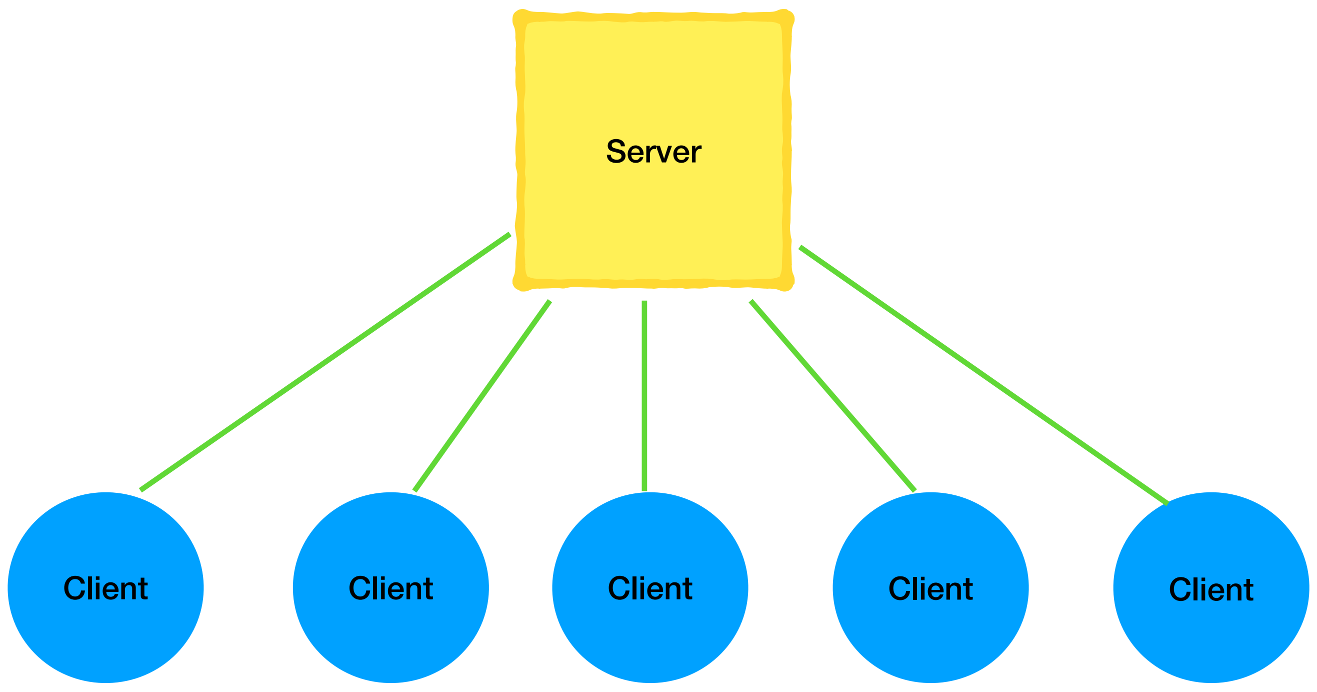
Internet Technology Assignment 1

Topic: Key-value store in a client-server based system using TCP socket

Name: Ritabroto Ganguly
Roll: 001910501090
BCSE-III A3



- The server stores the key-value pairs for each individual client.
- Each client is identified by an unique id.
- Each client can connect to the server using TCP sockets.
- The server can serve multiple clients simultaneously by creating a separate socket for each client connection.

<u>Implementation</u>

The server uses the fixed(for convenience) port number 5555 and the client port number is decide by the client's operating system.

- Well-known Ports — Ports in the range 0 to 1023 are assigned and controlled.
- Registered Ports — Ports in the range 1024 to 49151 are not assigned or controlled, but can be registered to prevent duplication.
- Dynamic Ports — Ports in the range 49152 to 65535 are not assigned, controlled, or registered. They are used for temporary or private ports. They are also known as private or non-reserved ports. Clients should choose ephemeral port numbers from this range.
- For our application, the server port could also have been an ephemeral port.

Each client connects to the server through port 5555 using TCP sockets.

<u>Client</u>

On successful connection,

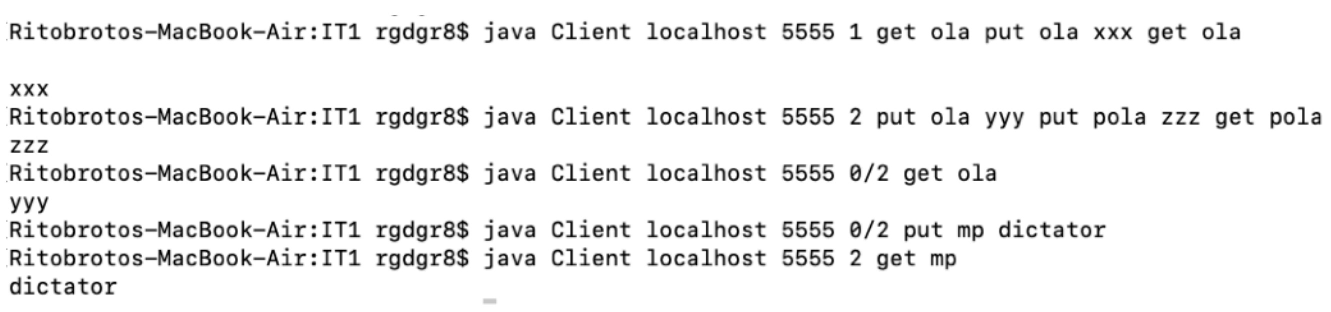
1. Client identification
   A. A single integer or
   B. Two integers separated by a slash("/")
      - The first integer is a manager id
      - The second integer is the client id whose key-value store the manager wants to access.
2. List of commands
   A. GET *key*
      - To get the value from the client's key-value store with the provided key if present, otherwise an empty string

B. PUT *key value*

- To store/update the value in the client’s key-value store

The above parameters are sent to the server from the client machine.

After sending all the above data, the client program sends an EXIT symbol to indicate the end of client’s requests to the server.



```
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Client localhost 5555 1 get ola put ola xxx get ola

xxx
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Client localhost 5555 2 put ola yyy put pola zzz get pola
zzz
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Client localhost 5555 0/2 get ola
yyy
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Client localhost 5555 0/2 put mp dictator
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Client localhost 5555 2 get mp
dictator
```

<u>Server</u>

The server stores the key-pairs as a map of maps.

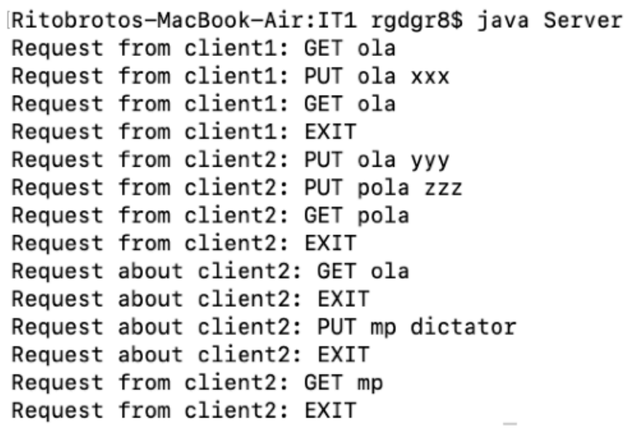
Client id is used as key in the outer map to get the key-value store(inner map) for each individual client.

The server gets the identification from the client machine and parses it and checks if the client is authorised for access.

If authorised, the server reads the client commands one by one parses them and returns the responses.

Authorisation can fail only when a guest client tries to access some other client’s key-value store. A static list maintains the ids of the managers.

If authorisation fails, the server sends an ERROR symbol back to the client and the client program exits.



```
Ritobrotos-MacBook-Air:IT1 rgdgr8$ java Server
Request from client1: GET ola
Request from client1: PUT ola xxx
Request from client1: GET ola
Request from client1: EXIT
Request from client2: PUT ola yyy
Request from client2: PUT pola zzz
Request from client2: GET pola
Request from client2: EXIT
Request about client2: GET ola
Request about client2: EXIT
Request about client2: PUT mp dictator
Request about client2: EXIT
Request from client2: GET mp
Request from client2: EXIT
```