CSE411: Distributed Computer Systems
Assignment (1)

Prepared By:

Salma Abdelfattah Fetouh Abdelfattah
Code: 1700622
Section: 2

Cairo, October 2021

Solution (i)

If mobile agents were used, we can assume three nodes in the system:

- Sensor nodes: from sensors that convert analog signals to digital numbers and signals in order to be able to do computations and complex processes with them.
- Computation nodes: clients that collect data from the information and signals coming from the sensors.
- Central nodes: servers in each area which are responsible for serving clients that collect data and process it from the clients.

These mobile agents are responsible for the process of transferring the data between sensor nodes & computation nodes in each area of the city, they also perform the necessary analyses on these collected data at the computation nodes, then they will transfer the results to central nodes after the necessary analyses and processing is done. The central nodes mobile agents are responsible for the recommendations of the best routes to take, they later send these data to the drivers.

Solution (ii)

The previous scenario can be implemented using a 3-tier architecture as follows:

Tier 1	**Tier 2**	**Tier 3**
Sensor Node Collect readings from sensors and send it to tier 2	**Computing Node** Perform analyses and computations on data sent from tier 1	**Central Node** Make recommendations and send them to drivers

Solution (iii)

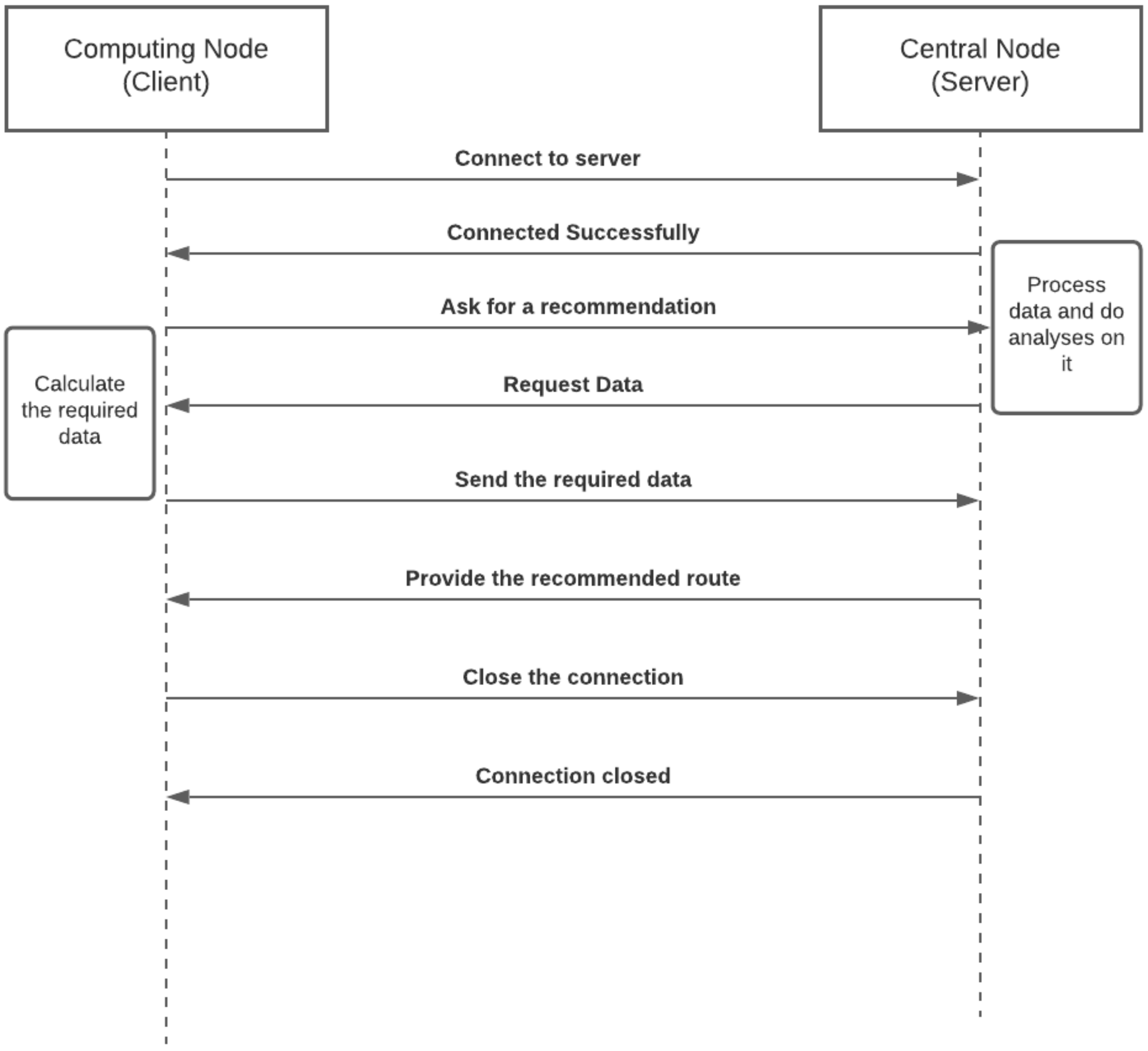

Figure 1-ALP diagram-designed by lucidchart

Solution (iv)

Code in details with Readme

Server Code

```
import java.io.*;
import java.net.*;

import handlers.ClientHandler;

public class Server {

   public static void main(String[] args) {
       try {
           // open a server socket
           ServerSocket sv = new ServerSocket(8080);
           System.out.println("Server waiting for a connection...");
           while (true) {
               // block until connection
               Socket s = sv.accept();
               System.out.println("Client Connected...");
               // open thread for this socket
               ClientHandler ch = new ClientHandler(s);
               Thread t = new Thread(ch);
               t.start();

           }

           // close the server
           // sv.close();
       } catch (IOException ex) {
           System.out.println(ex.getMessage());
       }
   }

}
```

Client Handler Code

```
package handlers;

import java.io.*;
import java.net.*;

public class ClientHandler implements Runnable {

    Socket s;

    public ClientHandler(Socket s) {
        this.s = s;
    }

    @Override
    public void run() {
        try {
            // create I/O streams
            DataInputStream dis = new DataInputStream(s.getInputStream()); //
for
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());

            // IO with client
            while (true) {
                // request the starting point
                dos.writeUTF("Please enter the starting point");
                dos.flush();
                String start = dis.readUTF();
                // request the destination
                dos.writeUTF("Please enter the destination");
                dos.flush();
                String destination = dis.readUTF();
                // Perform calculations to get the best route
                String best_route = getBestRoute(start, destination);
                dos.writeUTF("The best route would be " + best_route + "\n Start
Over? [y/n]?");
                dos.flush();
                String usr_choice = dis.readUTF();
                dos.flush();
                if (usr_choice.equalsIgnoreCase("n")) {
                    dos.writeUTF("bye");
                    System.out.println("Client disconnected");
                    dos.flush();
                    break;
                }
            }

            // close connection
            dis.close();
            dos.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public Socket getS() {
        return s;
    }

    public void setS(Socket s) {
        this.s = s;
    }

    public String getBestRoute(String start, String destination) {
        // dummy function for calculating best route
        return "Northeast";
    }

}
```

Client Code

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Client {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            // create socket to connect to the server
            Socket s = new Socket("127.0.0.1", 8080);
            // Create I/O streams
            DataInputStream dis = new
```

```java
DataInputStream(s.getInputStream());
            DataOutputStream dos = new
DataOutputStream(s.getOutputStream());

            // IO with server
            while (true) {
                // receive server command & print to user
                String srvr_msg = dis.readUTF();
                if (srvr_msg.equals("bye")) {
                    System.out.println("Session ended");
                    break;
                }
                System.out.println(srvr_msg);
                // take command from usr and send to the server
                String usr_msg = sc.next();
                dos.writeUTF(usr_msg);
                dos.flush();
            }
            // close the connections
            dis.close();
            dos.close();
            s.close();

        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```