# My ​Full Name ​as given in the class list

Chengzhi Fu

# My Student ID

8898807546

# What I have done in the assignment.

I implement a distributed system to generate customized resource allocation based on client queries. There are three hospitals (Hospital A, Hospital B, Hospital C), one client and one scheduler. A scheduler is responsible to receiving a client query then sends a request to the appropriate hospitals. Each hospital stores its own ​capacity​, ​availability and the ​map of LA​. Then, each hospital will reply to the scheduler. The scheduler will assign the client to the hospital with the highest score and inform the client and the hospitals of the decision. Communication message will be printed out.

# What my code files are and what each one of them does.

## scheduler.cpp

Scheduler establishes the UDP connection to each hospital and obtain the initial availability of the hospitals. Scheduler receives client location through TCP then send this location through UDP to available hospitals. Then scheduler will receive score from hospitals and choose hospital with the highest score, and inform the client through TCP and corresponding hospital to update availability through UDP.

Note: If there is a score and distance tie (same score and same distance), the rule scheduler will choose hospital A/B/C according to the alphabetical order.

## hospitalA.cpp

Hospital communicates with scheduler through UDP. Hospital reads in map.txt and construct the graph using adjacency matrix. It uses unordered\_map to re-index LA map. Take map\_simple.txt as an example,

Original location(key) # Re-indexed location(value) #

1. 0

12 1

2 2

8 3

18 4

14 5

10 6

1. 7

9 8

Then it can apply Dijkstras algorithm to find the shortest distance from client to this hospital. Then it will calculate the availability and score of this hospital. The it sends the score and distance to Scheduler. If scheduler choose this hospital to assign to a client, then occupation and availability of this hospital will be updated. UDP port number of hospital A is "30546".

## hospitalB.cpp

hospitalB.cpp boots up after hospitalA.cpp. hospitalB.cpp did almost the same job as hospitalA.cpp. It communicates with scheduler but the UDP port number is "31546".

## hospitalC.cpp

hospitalC.cpp boots up after hospitalA.cpp and hospitalB.cpp. hospitalC.cpp did almost the same job as hospitalA.cpp. It communicates with scheduler but the UDP port number is "32546".

## clinet.cpp

client communicates with scheduler through TCP. It sent query to Scheduler with its location. Then the client will terminate itself after it receives a scheduler reply and print out on screen message.

# The format of all the messages exchanged

All the communication messages send and receive between client, scheduler and hospitals are char array and words are separated by a whitespace. I use sprintf() function to put the message into a char array. To split the char array message from hospital, I use vector<string> to store each word. Here are some buffer I used.

## scheduler

char msg\_from\_hospitalA[MAXBUFLEN]; to store initialization data send from hospital, then use vector<string> to split the message.

char cli\_msg\_buf[MAXBUFLEN]; receive message from client

char hospital\_score\_disA/B/C[MAXBUFLEN]; store hospital’s score and distance and use vector<string> score\_disA/B/C to split them; If hospital is not available, then scheduler won't send request to this hospital then score is NONE, and distance is NO\_REPLY

char assign\_result[10]; put final compare result in which can be NONE, “A”, “B”, “C”

char \*location\_err = "location not found"; used to send location not found to client.

## hospitalA/B/C

char client\_loc\_char[MAXBUFLEN]; receive from scheduler. For location finding, upon receiving the input query.

If location is not found, send char \*not\_found = "location not found" to scheduler.

In order to update the occupation and availability, both of them are global variable with double type then they can participate calculation precisely. When print out change them as integer: (int)capacity\_num, (int)occupation\_num.

char score\_distance[MAXBUFLEN]; use sprintf() to store score and distance and send them to scheduler.

## client

char arg\_send[MAXDATASIZE]; After sending query to the

Schedulerbuf[numbytes]; After receiving output from the Scheduler or After receiving output the Scheduler, errors.

To better explain the format of all the messages exchanged, let’s give an example if hospitals read in map\_hard.txt.

## ./scheduler

The Scheduler is up and running.

The Scheduler has received information from Hospital A: total capacity is 10 and initial occupancy is 8

The Scheduler has received information from Hospital B: total capacity is 12 and initial occupancy is 8

The Scheduler has received information from Hospital C: total capacity is 15 and initial occupancy is 8

The Scheduler has received client at location 515154016 from the client using TCP over port 34546

The Scheduler has sent client location to Hospital A using UDP over port 33546

The Scheduler has sent client location to Hospital B using UDP over port 33546

The Scheduler has sent client location to Hospital C using UDP over port 33546

The Scheduler has received map information from Hospital A: the score = 0.005813 and the distance = 191.127401

The Scheduler has received map information from Hospital B: the score = 0.004747 and the distance = 274.777111

The Scheduler has received map information from Hospital C: the score = 0.009254 and the distance = 170.616152

The Scheduler has assigned Hospital C to the client

The Scheduler has sent the result to client using TCP over port 34546

The Scheduler has sent the result to Hospital C using UDP over port 33546

## ./hospitalA 255720539 10 8

Hospital A is up and running using UDP on port 30546.

Hospital A has total capacity 10 and initial occupancy 8.

Hospital A has received input from client at location 515154016

Hospital A has capacity = 10, occupation= 8, availability = 0.200000

Hospital A has found the shortest path to client, distance = 191.127401

Hospital A has the score = 0.005813

Hospital A has sent score = 0.005813 and distance = 191.127401 to the Scheduler

## ./hospitalB 545531658 12 8

Hospital B is up and running using UDP on port 31546.

Hospital B has total capacity 12 and initial occupancy 8.

Hospital B has received input from client at location 515154016

Hospital B has capacity = 12, occupation= 8, availability = 0.333333

Hospital B has found the shortest path to client, distance = 274.777111

Hospital B has the score = 0.004747

Hospital B has sent score = 0.004747 and distance = 274.777111 to the Scheduler

## ./hospitalC 919573377 15 8

Hospital C is up and running using UDP on port 32546.

Hospital C has total capacity 15 and initial occupancy 8.

Hospital C has received input from client at location 515154016

Hospital C has capacity = 15, occupation= 8, availability = 0.466667

Hospital C has found the shortest path to client, distance = 170.616152

Hospital C has the score = 0.009254

Hospital C has sent score = 0.009254 and distance = 170.616152 to the Scheduler

Hospital C has been assigned to a client, occupation is updated to 9.000000, availability is updated to 0.400000

## ./client 515154016

The client is up and running

The client has sent query to Scheduler using TCP: client location 515154016

The client has received results from the Scheduler: assigned to Hospital C

# Any idiosyncrasy of my project.

I tried some possible cases, for example, client not in map, client is the same as hospital location (d=0): the distance is None, one of the hospitals initial availability is None, all score is none and reply none. And I also tried lunch client multiply times to check the update of occupation and availability. There was no idiosyncrasy found.

# Reused Code:

1. Beej’s socket programming tutorial, A Simple Stream Server, A Simple Stream Client, Datagram Sockets. Functions like get\_in\_addr(), sigchld\_handler(), inet\_ntop(), send(). etc.
2. Dijkstras algorithm is modified from <https://www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/>. Functions like minDistance(), Dijkstra(). etc.