Uder Taxi Application

Software Specification

V3.0

Designed by AlphaGun

Chaoran Nong
Yijian Wang
Yanjie Xu
Yifan Xu
Xiaoyan Yang
Yan Zhang

Table of Content

1 Software Architecture Overview

1.1 Main data types and structures

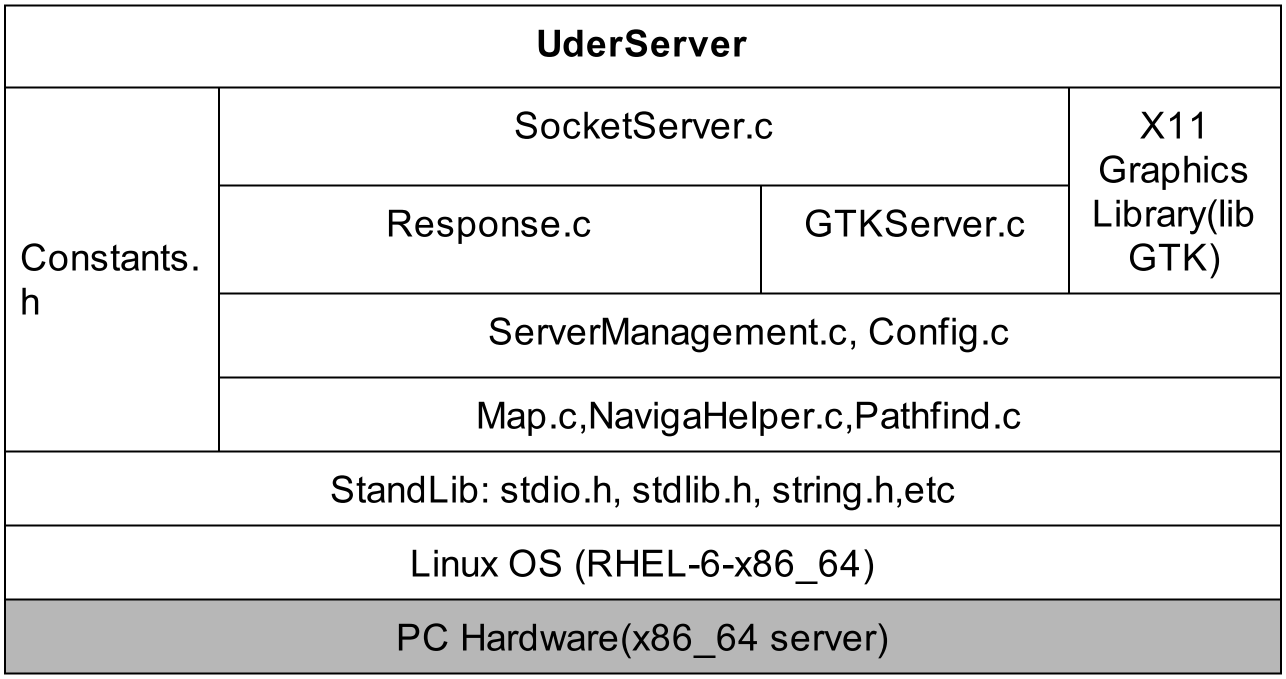
```
Typedef struct{
        Point position;
        Int status;  //0:occupied 1:
unoccupied
        Route route;
}Taxi;

Typedef struct{
        Int hour;
        Int minute;
}TIME;

Typedef struct {
        Point* routepoint;
}Route;

typedef struct {
        int time;
        Route route;
        Int fee;
        char* confirm;
} NaviReturn;

typedef struct Point_t {
   int x; int y;
} Point;

typedef struct TaxiStand_t {
   Point mark;
   int numTaxis;
   char name[LEN];
} TaxiStand;

typedef struct Landmark_t {
   Point mark;
   Point topLeft;
   Point botRight;
   char name[LEN];
} Landmark;

typedef struct Array_t {
   //int *items;
   void *items;
   int itemSize;
   int size;
   int capacity;
} Array;

typedef struct Map_t {
   char title[LEN];
   // street names
   int row;
   int col;
   int *rowNames;
   int *colNames;
   Array *landmarks;
   Array *stands;
} Map;

typedef struct{
   int x; int y;
} Point;

ILIST *CreatePosList(void);
void DeletePosList(ILIST *list);
void AppendPos(ILIST *list, Point *pos);
void DeleteLastPos(ILIST *list);
int PointSearch(Point* pos, ILIST* list);
ILIST* FindAdjacentSquare(Point pos, Map*
map);
```

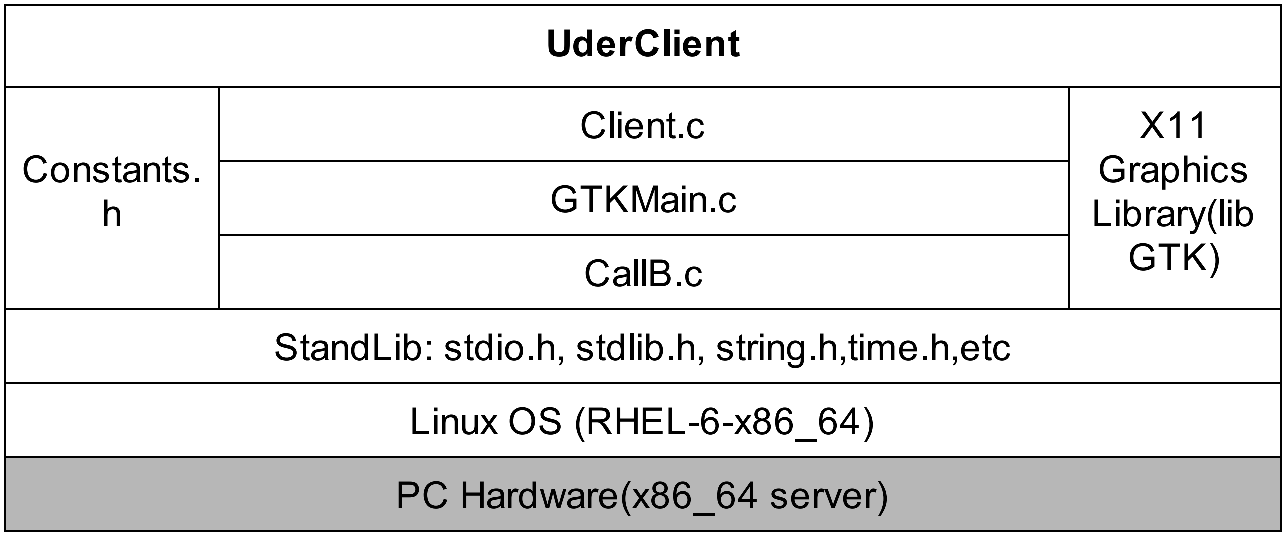
```
PosListEntry MinFSearch(ILIST *list);
```

1.2 Major software components

SocketServer.c, Response.c, GTKServer.c,ServerManagement.c, Config.c, Map.c, NavigaHelper.c, Pathfind.c, Client.c, GTKMain.c, CallB.c

• Diagram of module hierarchy


UderServer
Constants.h
SocketServer.c
X11 Graphics Library(lib GTK)
Response.c
GTKServer.c
ServerManagement.c, Config.c
Map.c,NavigaHelper.c,Pathfind.c
StandLib: stdio.h, stdlib.h, string.h,etc
Linux OS (RHEL-6-x86_64)
PC Hardware(x86_64 server)


UderClient
Constants.h
Client.c
X11 Graphics Library(lib GTK)
GTKMain.c
CallB.c
StandLib: stdio.h, stdlib.h, string.h,time.h,etc
Linux OS (RHEL-6-x86_64)
PC Hardware(x86_64 server)

1.3 Module interfaces

ServerManagement.h

```
Taxi* newTaxi();
int checkLandmark(Point pos, Map* newMap);
TaxiStand* Find_Stand(Point start_pos, Map* newMap);
Taxi Find_Taxi(Point pickup_pos, Map* newMap);
Route Navigation(Point start, Point end, Map* newmap);
int Taxi_Innit(Taxi* cab, Map* newMap);
double Fee(Route route,Map* newMap);
double Time(Route route);
int Taxi_Move(Taxi cab);
```

Utils.h:

```
Array* newArray(int itemSize);
void deleteArray(Array *array);
int addItem(Array *array, void *item);
void printArray(const Array *array, void *printFuncPtr(void *));
void setMapStreetNames(const char *str, const char *delim, int *names);
void setDefaultStreetNames(int *names, int size, int base);
void deleteStrings(int *strs, int size);
void printStrsVerbose(const int *strs, int size);
void printStrings(const int *strs, int size);
```

Callb.h:

```
void destroy(GtkWidget *widget, gpointer data);
void R_callback(GtkWidget *widget);
void P_callback(GtkWidget *widget, GtkWidget *entry);
void D_callback(GtkWidget *widget, GtkWidget *entry);
void Y_callback(GtkWidget *widget);
void N_callback(GtkWidget *widget);
BigP *CreateBigPointer();
void DeleteBigP(BigP *Pointer);
void copyString(char *str, char *entry);
void Crequest(BigP BP);
```

GTKMain.h:

```
char* GTK1(int argc, char *argv[]);
int GTK2(int argc, char *argv[], char* info);
```

<u>Map.h:</u>

```
void setPoint(Point *point, int x, int y);
int cmpPoint(const Point *point1, int x, int y);
Landmark* newLandmark();
void deleteLandmark(Landmark *landmark);
void setLandmark(const char *line, Landmark *landmark);
void printLandmark(const Landmark *landmark);
TaxiStand* newTaxiStand();
void deleteTaxiStand(TaxiStand *taxiStand);
void setTaxiStand(const char *line, TaxiStand *taxiStand);
void printTaxiStand(const TaxiStand *taxiStand);
int getIntegerIndex(const char *str, int *pIndex);
int getAlphabeticIndex(const char *str, int *pIndex);
void setObjName(const char *str, int *pIndex, char *name);
void reverse(char buffer[], int length);
void my_itoa(int num, char *buffer, int base);
void setDefaultStreetNames(int *names, int size, int base);
Map* newMap();
void deleteMap(Map *map);
int loadDefaultMap(Map *map);
int loadMap(const char *filename, Map *map);
void printMap(const Map *map);
```

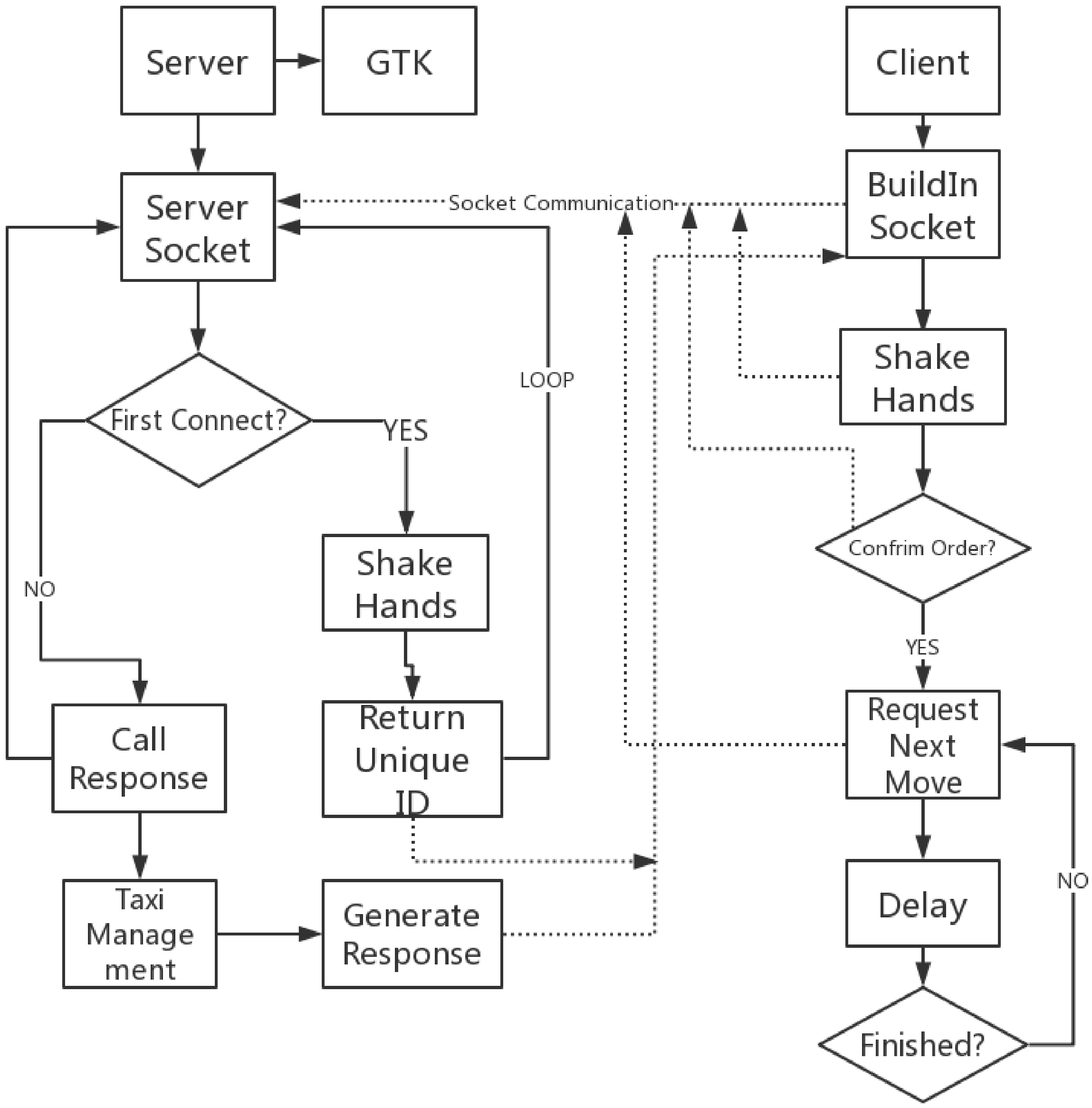
<u>Pathfind.h</u>

```
ILIST *CreatePosList(void);
void DeletePosList(ILIST *list);
void AppendPos(ILIST *list, Point *pos);
void DeleteLastPos(ILIST *list);
int PointSearch(Point* pos, ILIST* list);
ILIST* FindAdjacentSquare(Point pos, Map* map);
PosListEntry MinFSearch(ILIST *list);
```

<u>Response.h:</u>

```
char* Response(int Client_ID, char* req);
```

1.4 Overall program control flow



2 Installation

2.1 System requirements

Operating System:

Linux OS or Unix

Processer:

Intel Pentium (@ 1.5 GHz)
AMD ATHLON (@ 1.6GHz)
Or later versions

Graphic Requirements:

Nvidia GT 9800
AMD Radeon HD 2000
Intel HD graphics 500
Or later products

Disk Space:

At least 100M HDD space

RAM:

At least 10M

2.2 Setup and configuration

Type: “tar -zxvf Taxi_V1.0.tar.gz”
Then Tpye: make runc, to start Client
Or: make runs, to start Server

2.3 Uninstalling

Type “make clean”
Enter to confirm uninstallation.
Uninstallation finished

3 Documentation of packages, modules, interfaces

3.1 Detailed description of data structures

• Critical snippets of source code

```
Typedef struct{
        Point position;
        Int status;  //0:occupied 1: unoccupied
        Route route;
}Taxi;

Typedef struct{
```

```
            Int hour;
            Int minute;
}TIME;

Typedef struct {
            Point* routepoint;
            Int count;
}Route;

typedef struct {
            int time;
            Route route;
            Int fee;
            Char *confirm;
} NaviReturn;

typedef struct Point_t {
    int x; int y;
} Point;

typedef struct TaxiStand_t {
    Point mark;    // the mark position of taxi stand
    int numTaxis;   // the number of taxis located in this stand
    char name[LEN];  // name of the landmark
} TaxiStand;

typedef struct Landmark_t {
    Point mark;
    Point topLeft;  // top left point
    Point botRight; // bottom right point
    char name[LEN];  // name of the landmark
} Landmark;

/* generic array */
typedef struct Array_t {
    //int *items;     // pointer to the array of pointers
    void *items;      // pointer to the array of pointers
    int itemSize;   // the size of an item, =sizeof(itemType)
    int size;       // the number of elements stored
    int capacity;   // max number of elements allowed
} Array;

typedef struct Map_t {
    char title[LEN];    // title of the map
    // street names
    int row;    // number of rows
    int col;    // number of columns
    int *rowNames; // names of the rows
    int *colNames; // names of the columns
    Array *landmarks;
    Array *stands;  // pointer to array of taxi stands
} Map;

typedef struct{
        unsigned int Length;              /* Length of the list */
        PosListEntry *First;             /* Pointer to the first entry, or NULL */
        PosListEntry *Last;
        PosListEntry *LastSecond;
} ILIST;

typedef struct {                              /* Pointer to the list which this entry belongs to */
        PosListEntry *Next;
        Point *Pos;
        int F;
        int G;
        int H;
}PosListEntry;
```

3.2 Detailed description of functions and parameters

ServerManagement.h

```
Taxi* Scheduling(Point pickup_pos);
/*1.find_taxi() closest to pick-up point (2.consider pick-up time)*/
NaviReturn Navigation(Point start, Point end,map* newmap);
/*1.pick-up point to drop-off point 2.Taxi position to pick-up point*/
int Taxi_Info_Change(Taxi cab);
/*make taxi move from its location to the user location and to destination*/
Int Taxi_Innit(Taxi cab);
/*1.initialize taxi to unload type 2.set taxi route to nearest parking lot(taxi move) */
char *GenConfirm(NaviReturn navidata, int fee);
/*generate confirmation information and store in a file*/
int Clean_Confirm(char* confirm);
 /*clean confirmation*/
Int Fee(Route route);
/*Calculate route distance and decide route fee*/
```

```
Int Taxi_Move(Taxi cab);
/*change taxi location */
```

SocketServer.h

```
/* Print error diagnostics and abort */
void FatalError(const char *ErrorMsg);
/*Initialize Socket server */
int MakeServerSocket(uint16_t PortNo);
/* Fetch the top response from the client-request-stack and Return this request as a String */
void ServerMainLoop( int ServSocketFD,       /* server socket to wait on */
          ClientHandler HandleClient,        /* client handler to call */
          TimeoutHandler HandleTimeout,      /* timeout handler to call */
          int Timeout);
/* Main function of the Server Socket */
int main(int argc, char *argv[]);
```

Client.h

```
/* Print error diagnostics and abort */
void FatalError(const char *ErrorMsg);
/* Main function, connect and communicate with Server, send request and confirmation */
int main(int argc, char *argv[]);
```

Constants.h:

```
#define BaseFee 3.75                  /* Dollar */
#define DisCharge 2                   /* Dollar per mile */
#define Wage 0.8                      /* Dollar per mile */
#define MaxTime 20                    /* Minute */
#define MaxSpeed 0.75                 /* Mile per Minute */
#define MaxCar 12                     /* Max Car in a single stop */

#define TimeOutConstant 25000         /* Handle Timeout constant, microsecond */
#define LoopIntervalConstant 10000    /* Time Interval constant, millisecond */

#define OrderLength   5               /* Length of the unique Order Number */
#define OrderZero "90000"             /* Str: Number of Order Zero */
#define OrderZeroInt 90000            /* Int: Number of Order Zero */

#define LEN 100                       /* GTK String Maximum Length*/
```

Navigation.h:

```
/* Calculate the distance between two different points*/
int Distance(Point *p1, Point *p2);
```

```
 /* calculate the fee of the trap */
int Fee(int distance);
/* Get the user’s position, find a nearest taxi for him */
void FindTaxi(Point *p1);
```

3.3 Detailed description of routing and scheduling algorithms

• description of routing algorithm

For our navigation system, taxi will only be guided to the destination through the shortest route, which is the best strategy to decrease management cost. If a brunch of shortest routes available, the system will prefer to go Vertical(North/South) first instead of Horizontal(East/West), go North first instead of South, and go East instead of West.

• description of scheduling and communication with the taxi

All the taxis will be equally distributed to every stand and stay there until scheduled by the system. For those taxis which finished orders, they will be guided back to the nearest stand. Whenever a request is fetched, the system will calculate out an available taxi which located nearest to the guest and send confirmation request back to the client. During this period, this taxi will be set as reserved(cannot respond to any other requests) but keep his guidance(back to the nearest stand or stay inside a stand). A taxi will only be guided to pick up the guest after the order is confirmed by the guest.

3.4 Detailed Protocol Specification

– **Client Request**

<request>	REQUEST_TAXI <location> TO <location> <special>
	REQUEST_POSITION <taxi>
	CONFIRM <reservation>
	CANCEL <reservation>
<location>	<pos>
	<landmark_name>*
<special>	[ASAP]
	[AT <time>]
	[FOR <number_persons>]*
	[NON_STOP]*
<time>	<hours>:<minutes>

– **Server Response**

```
<response>
                OK <taxi> PICKUP <pos> AROUND [<time>] DROPOFF <pos>
                        [<time>]$<amount> CONFIRM <reservation>
                DECLINED <reason>
                OK <taxi> POSITION <position> [ETA <position> <time>]
                INVALID <reservation>
                ERROR <message>
<amount>        <dollars>.<cents>
```

*Request Marked with star will only be available in Advanced part.

4 Testing Plan

4.1 GTK Testing:

Before compiling into the main problem, the GTK part testing is depending on sending the data to the serve including:

Pick up location; Destination; number of people needed on one trip;
Car types; **Advanced data(time, multiple destination and pick up point)

The first step is to see if the program actually get the input data from user and send it to the computer. Simply using pointer to store the data user inputted and print out the data on the ASCII interface and see if it match.

Second step is to check if the data can send to the server. After making sure that the program is running fine on the local, upload the data to the server and using another simple program to download the data and print it out on ASCII interface to check if the data is correct.

4.2 Server Management Testing:

There is a test main function in the ServerManagement.c. And in the main loop, the test program will use a simple test data to test each function and enable the DEBUG mode to print each function’s return value and important data.

4.2 Socket Testing:

There will be a simple Test function to make sure socket communication between Server and Client can work. This function will simulate requests and responses on both Server and Client but will not call any other core functions.

4.3 System Testing:

There will be a simple TestModules executable file to call each module test function according to input. There will also be TestClient and TestServer executable files which are our main applications with debug mode on.

5 Development plan and timeline

5.1 Partitioning of tasks

Table 1.1 File List of Client		
Client.c	Connect and communicate with Server	Yifan Xu
Client GTK	User Interface of the Client(GTK)	Chaoran Nong Yifan Xu

Table 2.1 Files List of Server		
Response.c	Process Configuration, Translate Command fetch by Socket, call Server Core System	Yan Zhang
SocketServer.c	Connect and communicate with Clients	Yifan Xu
Test.c	Autotest Code for the whole program and each function	Yang Zhang
ServerGTK	User Interface of the Server(GTK)	Chaoran Nong
ServerCoreSystem	Yanjie Xu, Xiaoyan Yang, Yijian Wang, Yan Zhang	

Table 2.2 Functions List of Server Core System	
Fee()	Yijian Wang

Account()	Yijian Wang
Navigation()	Xiaoyan Yang
Distance()	Yanjie Xu
Taxi_Info_change()	Xiaoyan Yang
Taxi_Innit()	Xiaoyan Yang
Find_Taxi()	Yanjie Xu
Scheduling()	Yanjie Xu
Stop_Check()	Yanjie Xu
Navigate()	Xiaoyan Yang
Gen_confirm()	Yan Zhang
Clean_confirm()	Yan Zhang
Taxi_Move()	Yan Zhang

Table 3.1 Advanced Features List		
Appointment	Appointment system that let users reserve a future car	...
Carpool	Car share system that let users share a ride with others	...
...	…...	...

6. Appendix

6.1 Copyright

http://www.cnblogs.com/zhoug2020/p/3468167.html