CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

The implementations of many smart systems are increasingly depending upon wireless Ad-Hoc networks. Lack of need for centralized and dedicated infrastructure in wireless Ad-Hoc networks and comparatively lower power consumption of the wireless nodes make it more ample to be applied in many situations. An implementation of a smart elevator system based on a wireless multi-hop Ad-Hoc network is demonstrated.

For experimental simplicity, a single elevator in a five-storey building is considered. The traffic conditions of every floor are communicated to all other floors in a direct or multi-hop fashion. In the proposed system, the elevator serves the floor with greater traffic and hence achieving the objective of serving more people within a given time. As our proposed elevator system communicates in a multi-hop manner, this means that every node (floor) can only communicate with its adjacent nodes (floors). All nodes are capable of being informed of each other's traffic through this technique of wireless multi-hop communication.

We propose this multi-hop nature of interacting between nodes keeping in mind the inter-floor distance and the floor thickness that could affect communication if each floor was to be connected to every other node. To integrate our wireless network in controlling the elevator, a study on traditional elevator systems was made. The two basic algorithms that make the decision to stop the elevator car at a particular floor, called the dispatching algorithms, are (a) Based on the current direction of the elevator, and (b) Based on the time of request from each floor. These revolve around the objectives of either minimizing the consumption of power or minimizing the average waiting time for passengers.

In the first approach, if the elevator is currently moving in a certain direction, it will stop at floors in its way that have requests in that direction only and will change direction once it serves them and if there are requests in the opposite direction.

On the other hand, in the time-based approach, the requests are stored in a queue and ordered according to their time of arrival. The elevator then would serve the floors according to this queue.

An ad hoc network typically refers to any set of networks where all devices have equal status on a network and are free to associate with any other ad hoc network device in link range. Ad hoc network often refers to a mode of operation of IEEE 802.11 wireless networks.

1.1 MOTIVATION

We felt the need to make the current elevator system more effective by implementing a smart elevator system based on a wireless multi-hop Ad-Hoc network resulting in reduced waiting times for passengers.

1.2 PROBLEM STATEMENT

The traditional elevator moves to floor which has requested first regardless of the density of people at each floor. The system we propose calculates the density of people at each floor and serves the floor with larger people density.

1.3 PURPOSE

There are quite a few elevator dispatching algorithms being commonly implemented and used, like those above, there are several problems concerning the passengers and requesters. Firstly, considering the direction-based approach, if the elevator is moving in a particular direction (up/down), and the number of people requesting to go in that direction is less than those requesting to go the opposite way, a majority will be kept waiting. Secondly, in the time-based approach, if the queue of requests is haphazard, the elevator would waste a lot of power and trips. The elevator would be moving up and down the building and passing other requesting floors without serving them. Eventually it will return to them at some later point in time. So, in such cases, the elevator would be consuming more power and time than it would have to if it served the intermediate floors when it passed across them previously. We present a solution to such problems through the use of wireless ad-hoc networks connected together in a multi-hop ad-hoc manner.

1.4 SCOPE

The facility to specify the destination of the passengers beforehand. The system computes density of people at each floor to make the decision that results in lesser waiting time and the specification of threshold to make sure every floor is served.

1.5 REPORT ORGANISATION

Chapter 1 Introduction:

This chapter gives a brief introduction about the working of wireless ad-hoc networks.

Chapter 2 Literature survey:

This part gives an introduction of the work that has been carried out prior to the development of this project.

Chapter 3 System Design:

This chapter gives an overview of the proposed method used including the communication protocols and the algorithm logic used to drive our system.

Chapter 4 Implementation:

This chapter describes the algorithm used in the implementation of our system and also the interfaces used to run the algorithm.

Chapter 5 Testing:

This chapter specifies the various input test cases our algorithm is validated against.

Chapter 6 Results:

This chapter shows the results are observed for specific inputs of data from the GUI.

Chapter 7 Conclusion:

This chapter concludes our work on smart elevators and suggests future improvements that can be incorporated in the system.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

The following section enlists the title of the papers along with the brief description about them. The same were referenced during the course of completion of our project. These papers provided us with initial impetus and required information during the project completion.

Implementation of Smart Elevator System based on Wireless Multi-hop Ad-Hoc Sensor Networks by Hamza Ijaz Abbasi, Abdul Jabbar Siddiqui [1]:

In this implementation of the elevator system, each floor is represented and simulated by a laptop with the Java based application to run the simulations. The floors (or laptops) are connected with each other in a multi-hop Ad-Hoc fashion such that each can communicate only with its immediate neighbouring laptop (the floor above, and the floor below).Each floor (laptop) in our system is assigned a unique IP Address of the form “192.168.1.x” where ‘x’ corresponds to the respective floor-number. Eg: Floor#1 has the IP Address “192.168.1.1”.
In order to setup a wireless ad-hoc network amongst the laptops (floors), the standard IEEE 802.11 b/g wireless Ethernet cards on each laptop (which is simulating a floor) are used.
Initially (During the start-up of the elevator system program), elevator is always present at Floor-1. The floors intermittently communicate with each other in order to keep each other updated about the latest traffic information as well as the current location of the elevator. Each floor (sensor) that receives the Elevator (EIP) is entitled to take the decision about where to send it next based on the following criteria:

Let n be the floor which currently has the elevator. If the sum of the traffic (number of people) requesting from above floors and the up traffic (people requesting to go up) in the nth floor is greater than the sum of the traffic (number of people) requesting from down floors and the down traffic (people requesting to go up) in the nth floor, the result will be that the elevator will be sent to the above floor i.e. to Floor (n+1).
However, If the sum of the traffic (number of people) requesting from above floors and the up traffic (people requesting to go up) in the nth floor is lesser than the sum of the traffic (number of people)

requesting from down floors and the down traffic (people requesting to go up) in the nth floor, then the elevator will be sent downwards to Floor (n-1).

In the same way, the new floor which receives the elevator will then make the decision.

If the traffic above equals the traffic below, the elevator will continue to move in the same direction that it was previously travelling in, according to predefined default elevator dispatching algorithm(s).

If the total traffic in all floors is zero (i.e. there are no requests), the elevator will eventually return back to Floor 1 (the default stop).

After the elevator serves any floor, the traffic in that floor is updated to ‘0’ and communicated across other floors.

Ant Colony Optimization for Single Car Scheduling of Elevator Systems with Full Information by Zhen Shen and Qian-Chuan Zhao [2]:

The focus is on the single car, full information elevator problem. Here “full information” means that the arrival time, the origins and destinations of passengers are all assumed known beforehand. The aim is to find the best solution of serving the passengers, and the performance is measured by the average service time. The problem is modelled into a graph and the goal is converted to finding a path on the graph corresponding to the best performance. An algorithm of Ant Colony Optimization (ACO) is applied. the elevator problem into a graph. A node is either a loading or an unloading task of a passenger. The edge length is the cost to travel from one node to another, which is a function of the estimated time needed or a function of the distance. That the car should finish all the loading and unloading tasks corresponds to that the ant should visit all the nodes in the graph. And then solving the problem is converted to finding a feasible path with the best performance. It is a combinational optimization problem similar to TSP. This is one reason why we choose ACO to solve the problem. But different from the TSP, there are many constraints in the elevator problem. The candidate list is used to handle the constraints. Only the unvisited nodes without causing the infeasibility can be selected into the candidate list. The modelling and the handling of the constraints are the major contributions of this paper.

Here we shortly introduce how the Ant Colony Optimization (ACO) works. Reviews can be found in [15]. When applying ACO, the problem should be modelled into a graph. A path on the graph corresponds to a solution. A group of ants are created and placed at starting nodes. The starting nodes should be chosen according to the problem. In ACO, one important concept is the pheromone level, which indicates how desirable the edge or node is. The pheromone can be deposited either at the nodes or the edges. The ants will choose the next node according to the pheromone level and the length of the edge directing to the next node. The higher the pheromone level is and the shorter the edge is, the more desirable the node is.

After all the ants finish their paths, a global updating rule is implemented. The pheromone on each edge evaporates proportionally. And the pheromone on the best so far path is given an increment. Sometimes the pheromone is set with a threshold of minimum value to ensure the diversity of searching. After the operations above, a cycle of the algorithm is finished. The algorithm is run cycle by cycle until the stopping criteria are met. The pheromone stores the history information about the structure of the problem and acts as a tool for indirect communication. Local updating of pheromone and local search can be embedded into ACO to improve the performance of the algorithm. Decentralization, cooperation and iteration are the characteristics of ACO and also the reasons why it is effective.

Knowledge-Based Elevator Controller By G.C Clark. P Mehta, R Frowse [3]:

The traditional methods of controlling elevators in modem buildings are inaccurate and fall far short of the standard expected Ui large modem buildings where the control system must cope 1101 only with peak demands but also with changing traffic patterns over the life time of the building. The paper describes the design and performance of a knowledge based elevator control system that uses miles to compare possible movements and select the optimum route to maximise passenger transportation and minimise waiting time.

This part gives an introduction of what work has been carried out prior to development of this project. This talks about the overview of the following topics:

- JDK(JAVA,SWINGS,SOCKET PROGRAMMING)
- AD-HOC NETWORK

2.1 Existing System:

Current elevator system works on the basis of first in first out concept. The Drawback of this system is that it moves in the same direction until it reaches extreme end or there are no further requests in the same direction.

2.2 Proposed System:

Smart Elevator is a concept that overcomes the drawbacks of the traditional Elevator system. Our Elevator system makes decision on each floor about the direction of the Elevator based on the requests received from its above and below floors.

2.3 JDK

JDK stand for Java Development Kit, a software package that can be used to write, compile, debug and run Java applications and applets. Basically it contains minimal set of tools which are need to develop Java application. Just starting from version 1.1 and now the latest version is 1.7. Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

2.3.1 JAVA

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarity on any supported hardware/operating-system platform. One should be able to write a program once, compile it once, and run it anywhere. This is achieved by compiling the Java language code, not to machine code but to Java bytecode-instructions analogous to machine code but intended to be interpreted by a virtual machine(VM) written specifically for the host hardware.

2.3.2 SWINGS

Java Swings consist of

- Look and feel
- Accessibility
- Java 2D

- Drag and Drop etc

If you do not explicitly add a GUI component to a container, the GUI component will not be displayed when the container appears on the screen. Swings, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing gui Apps as well as Servlets and Applets. It employs a model/view design architecture. Swing is more portable and more flexible than AWT. Swing is built on top of AWT and is entirely written in Java, using AWT's lightweight component support.

2.3.3 SOCKET PROGRAMMING

The endpoint in an inter-process communication is called a socket, or a network socket for disambiguation. Since most communication between computers is based on the Internet Protocol, an almost equivalent term is Internet socket. The data transmission between two sockets is organised by communications protocols, usually implemented in the operating system of the participating computers. Application programs write to and read from these sockets. Therefore, network programming is essentially socket programming.

2.4 AD-HOC NETWORK

A wireless ad hoc network is a decentralized type of wireless network. The network is ad hoc because it does not rely on a preexisting infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity. In addition to the classic routing, ad hoc networks can use flooding for forwarding the data.

An ad hoc network typically refers to any set of networks where all devices have equal status on a network and are free to associate with any other ad hoc network device in link range. Ad hoc network often refers to a mode of operation of IEEE 802.11 wireless networks.

It also refers to a network device's ability to maintain link status information for any number of devices in a 1-link (aka "hop") range, and thus, this is most often a Layer 2 activity. Because this is only a Layer 2 activity, ad hoc networks alone may not support a routable IP network environment without additional Layer 2 or Layer 3 capabilities.

CHAPTER 3
SYSTEM DESIGN

CHAPTER 3

SYSTEM DESIGN

3.1 REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS

- JDK 1.7
- OS-WINDOWS 7

HARDWARE REQUIREMENTS

- WIFI ENABLED LAPTOPS

3.2 Assumptions and Constraints

We make the following assumptions:

• We assume that all the parameters of the car and the building are known, which includes the number of the floors of the buildings, the time needed for the car to move from one floor to another.

• We assume the speed of the car is not affected by the number of the passengers in it.

• The door open time, the door close time, and the door dwell time are fixed and given.

• Sometimes the best service of an individual passenger conflicts with the global best scheduling. Here we assume that the passengers obey the scheduling of the algorithm, i.e. the passengers' subjective choice is forbidden.

We set up the following constraints:

• The passenger cannot de-board the car halfway and board again.

• Once a passenger boards a car, s/he must be delivered to their destination floor without passing by the destination floor and then returning to it.

• The capacity of the car is measured by the number of the passengers. The car cannot have passengers more than the capacity in it.

3.3 METHODOLOGY

The proposed smart elevator system utilizes a wireless multi-hop ad-hoc network and will be simulated and prototyped depicting a building comprising of five floors. In fig 3.1 we emulate each floor using a single laptop to ease simulation and implement a graphical view of the system at work. Therefore, three laptops will be connected together in an ad hoc multi-hop manner such that the nth laptop (emulating the nth floor) will be connected only to the (n-1)th laptop and (n+1)th laptop with the exception of 1st and 3rd laptops respectively where 1st laptop will only be connected to 2nd laptop whereas 3rd laptop will be only connected to 2nd laptop. Each floor could be equipped with some external keypad which could then feed a people counting algorithm and know the number of people waiting in that floor to go upwards and downwards. The laptop will then communicate and pass this information to the next connected laptops (multi-hop). In this way, the information will be shared among all the laptops. However, the final decision will only be made by that laptop (i.e., floor) where the elevator is currently present. This decision will only be made by this laptop after running some algorithm.

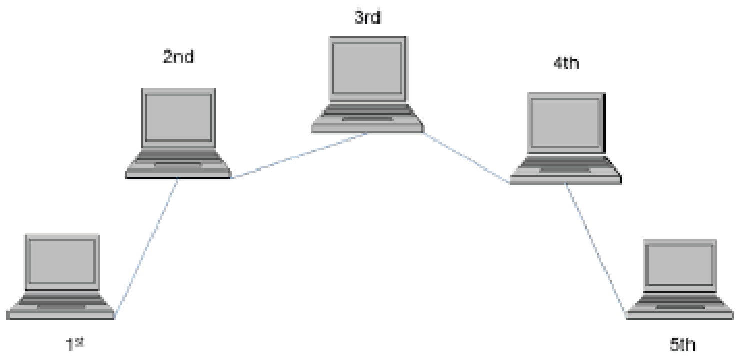


Figure 3.1 Laptops (floors) connected in multi-hop fashion

3.4 FLOW DIAGRAM OF THE PROPOSED METHOD

The system begins by checking if there is a request for a floor. If a button press has been detected the input at that floor is collected and the algorithm to serve the passengers is run. After the algorithm is run it checks for any additional requests. If yes then repeat the above process else stop the system.

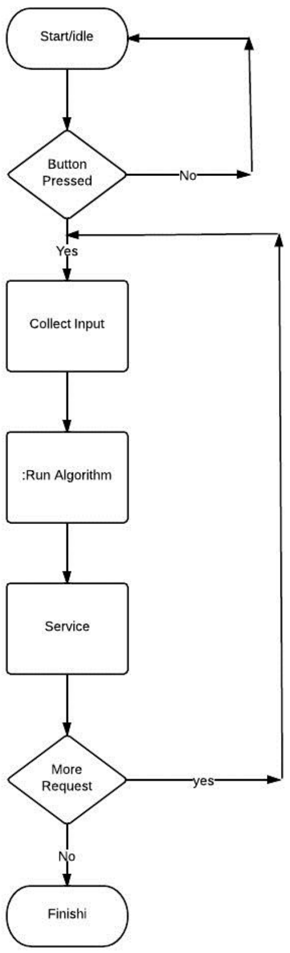


Figure 3.2 Flow diagram of Elevator system

:RUN ALGORITHM :This is called when the elevator is in current floor to make the decision.

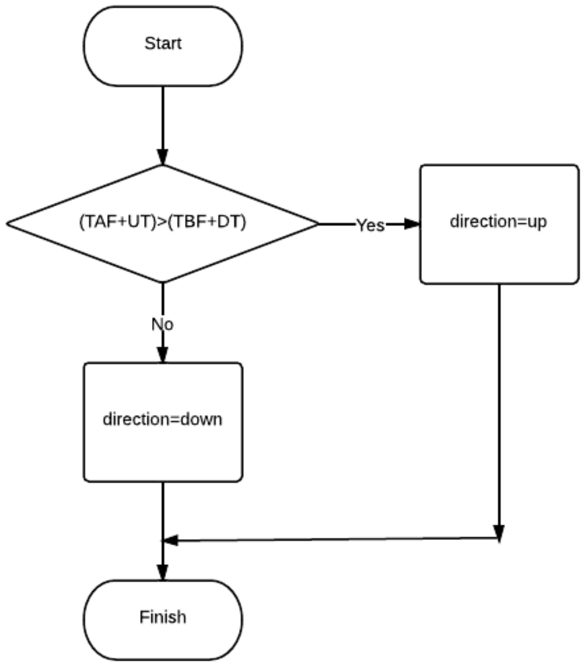


Figure 3.3 Flow diagram of run algorithm

3.5 COMMUNICATION

In order to setup a wireless ad-hoc network amongst the laptops (floors), we use the standard IEEE 802.11 b/g wireless Ethernet cards on each laptop (which is simulating a floor). For the inter-laptop (inter-floor) communication, the datagram communication protocol we have used in our system is known as UDP (User Datagram Protocol) which is a connectionless protocol, as opposed to the TCP (Transfer Control Protocol) which is a connection-oriented protocol. UDP serves our purpose best because of its simplicity and wing to the following reasons. Firstly, our system does not require connections between floors to be continuously established as the communication between floors is required only in few circumstances like change in traffic or elevator position. Secondly, the information to be transferred is simply packets of a maximum of 4 bytes and not streams of data. Hence, UDP is more efficient for our system than TCP as it emphasizes on greater speeds and less delays.So, maintaining a connection between them would be a waste of power. Hence we exploit the connectionless nature of UDP. On the other hand, using TCP will maintain the connection for a relatively longer period of time, thus causing power wastage as opposed to the UDP.

3.5.1 Information Awareness Mechanism

Floor Information Packets (FIPs)

- These are UDP datagram packets having information such as the floor-number which it belongs to, the amount of traffic (upward plus downward traffic), the amount of downward traffic. The structure of the packet is shown below.

Floor ID	Density
3	2

Elevator Information Packets (EIPs):

- These UDP packets provide information about the location of the elevator (i.e. current floor).

Floor ID	Destination
5	4

Flag Information Packets:

- These UDP packets provide flag array (i.e., an array that holds the request) with respect to the particular floor.

Floor ID	Flag Array
4	[1,0,1,0,1]

CHAPTER 4
IMPLEMENTATION

CHAPTER 4

IMPLEMENTATION

4.1 ELEVATOR DISPATCHING METHOD

Initially (During the start-up of the elevator system program), elevator is always present at Floor-5. The floors intermittently communicate with each other in order to keep each other updated about the latest traffic information as well as the current location of the elevator. Each floor that receives the Elevator (EIP) is entitled to take the decision about where to send it next based on the following criteria:

- Let n be the floor that currently has the elevator. If the sum of the traffic (number of people) requesting from above floors and the up traffic (people requesting to go up) in the nth floor is greater than the sum of the traffic (number of people) requesting from down floors and the down traffic (people requesting to go down) in the nth floor, the result will be that the elevator will be sent to the above floor i.e. to Floor (n+1).
- However, If the sum of the traffic (number of people) requesting from above floors and the up traffic (people requesting to go up) in the nth floor is lesser than the sum of the traffic (number of people) requesting from down floors and the down traffic (people requesting to go down) in the nth floor, then the elevator will be sent downwards to Floor (n-1).
- In the same way, the new floor which receives the elevator will then make the decision.
- If looping is occurring between floors, the elevator ignores all requests and goes to the unserved floors.

After the elevator serves any floor, the traffic in that floor is updated to ‘0’ and communicated across other floors.

The Figure 4.1 explains the concept with greater detail, considering the example of a five floor building with a single elevator. As depicted in the figure below, since traffic above Floor_3 + up-traffic at Floor_3 (i.e., 0+6=6) is greater than traffic below Floor_3 + down-traffic at Floor_3 (i.e., 3+2=5), the elevator is sent upwards to Floor_4 (to be sent to Floor_5).

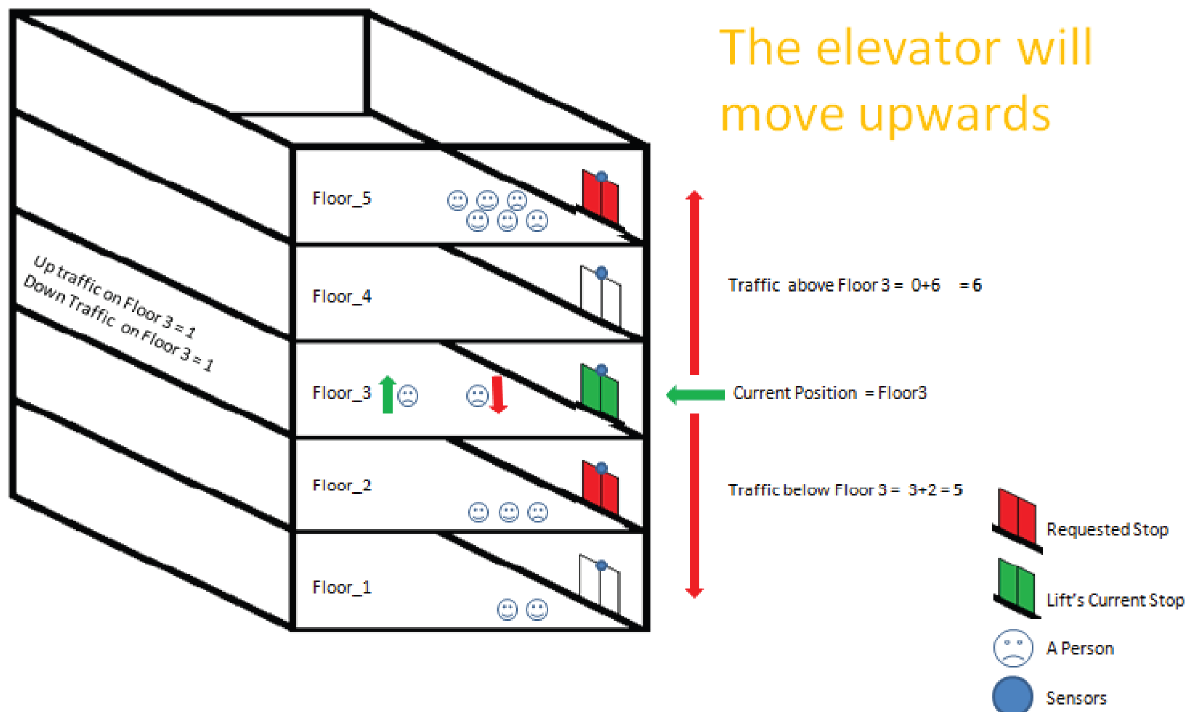

The elevator will move upwards
Floor_5
Floor_4
Floor_3
Floor_2
Floor_1
Up traffic on Floor 3 = 1
Down Traffic on Floor 3 = 1
Traffic above Floor 3 = 0+6 = 6
Current Position = Floor3
Traffic below Floor 3 = 3+2 = 5
Requested Stop
Lift's Current Stop
A Person
Sensors

Figure 4.1 Proposed Elevator Dispatching Method

4.2 INTERFACING

NetBeans IDE provides first-class comprehensive support for the newest Java technologies and latest Java enhancements before other IDEs. It is the first IDE providing support for JDK 7, Java EE 6, and JavaFX2.With it's constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, NetBeans IDE sets the standard for developing with cutting edge technologies out of the box.

NetBeans IDE provides different views of your data, from multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily, while giving you versioning tools via Subversion, Mercurial, and Get integration out of the box.

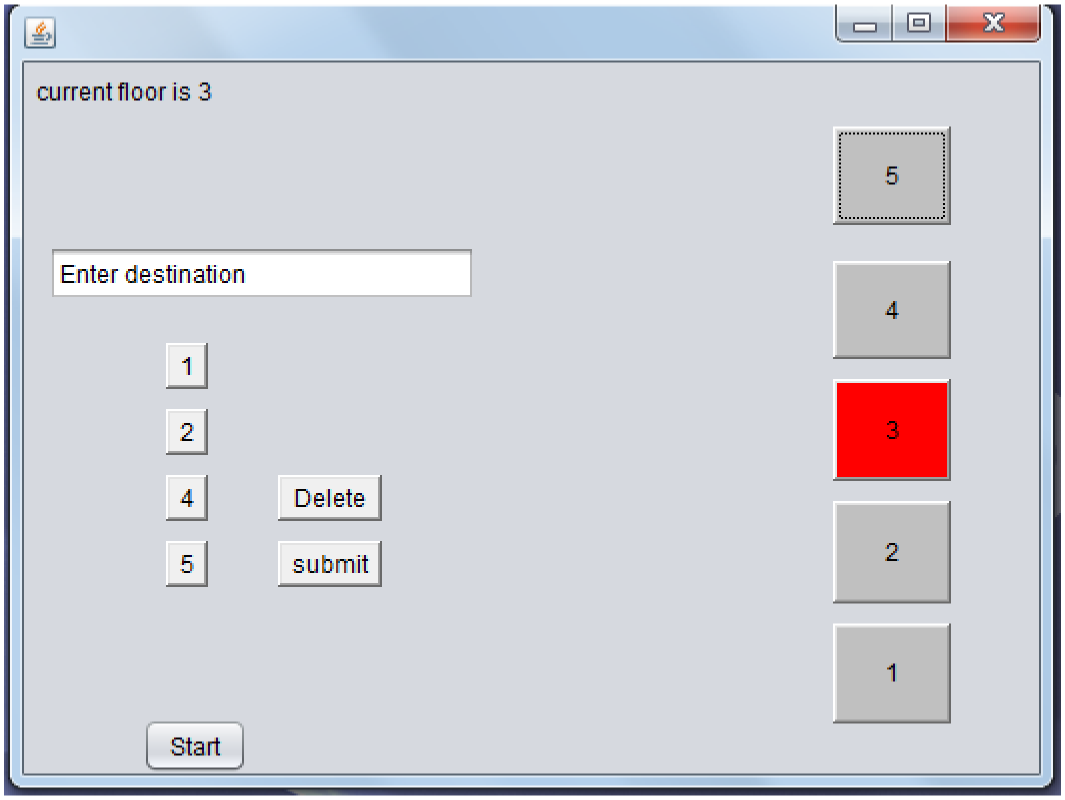


Figure 4.2 Graphical user Interface for Elevator system

For Java SE applications, the NetBeans GUI Builder automatically takes care of correct spacing and alignment, while supporting in-place editing, as well. The GUI builder is so intuitive that it has been used to prototype GUIs at customer presentations.

NetBeans provides static analysis tools, especially integration with the widely used FindBugs tool, for identifying and fixing common problems in Java code. In addition, the NetBeans Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs.

The NetBeans Profiler provides expert assistance for optimizing your application's speed and memory usage, and makes it easier to build reliable and scalable Java SE, JavaFX and Java EE applications. NetBeans IDE includes a visual debugger for Java SE applications, letting you debug user interfaces without looking into source code. Take GUI snapshots of your applications and click on user interface elements to jump back into the related source code.

4.3 Module 1: Code snippet for current floor

This algorithm is executed when the elevator is in current floor. It receives the requests from other floors and considers the input given at the current floor to make the decision of sending the elevator.

The code is given below:

```
if(curclr==java.awt.Color.RED)
{
        eleLoc=Integer.parseInt(f5.getActionCommand());
        for(;top!=-1;)
       {
                 int i=1;
                 element=stack.pop();
                 top--;
                 while(i!=element)
                       i++;
                   c[i]++;
      }
      for(int k=1;k<=5;k++)
      {
                 if(k<eleLoc)
                 DT+=c[k];
                 else
                 UT+=c[k];
      }


 try{
                 client nw;

                 nw=new client(flag,c);

                  if((nw.TAF+UT)<(nw.TBF+DT))

                 {

                         for(int i=(eleLoc-1);i>=1;i--)

                         {

                                 if(flag[i]==1)

                                 {

                                         p=i;break;

                                 }

                         }

                 }

                 else if((nw.TAF+UT)>(nw.TBF+DT))

                 {

                         for(int i=eleLoc+1;i<=5;i++)

                         {

                                 if(flag[i]==1)

                                 {

                                         p=i;break;

                                 }

                         }

             switch(p)

       {

                   case 3:f3.setBackground(java.awt.Color.RED);

                   f5.setBackground(java.awt.Color.LIGHT_GRAY);

                   flag[3]=0;
```

```
                     c[3]=0;
                    eleLoc=3;
                    break;
            case 4:f4.setBackground(java.awt.Color.RED);
                     f5.setBackground(java.awt.Color.LIGHT_GRAY);
                    flag[4]=0;
                    c[4]=0;
                    eleLoc=4;
                   break;
    }


     sendfloor();
    }
    catch(Exception e){
    }
```

4.4 Module 2: Code snippet when elevator is not in the current floor:

This algorithm is executed when the elevator is not in the current floor. It sends its requests to the current floor elevator and also the threshold is checked avoid the looping condition.

The code is given below:

```
else{
      try{
         DatagramSocket elsesck=new DatagramSocket(1223);
         byte []emptyData=new byte[10];
         DatagramPacket emptyrec=new DatagramPacket(emptyData,emptyData.length);
         byte []elseData=new byte[10];
```

```
        elsecount=0;
      switch(eleLoc)
     {
       case 3:
       DatagramPacket elsesend1=new DatagramPacket(elseData,elseData.length,IP1,1223);
        elsesck.send(elsesend1);
        break;
   }
   if(elsecount==1){}
   else{
            byte b1[]=new byte[10];
           b1[0]=5;
           b1[1]=(byte)(top+1);
           try{
                  server c=new server(b1,flag);
               }
          catch(Exception e){}
     }
      receiveflr();
    switch(eleLoc)
    {
          case 3:
           f3.setBackground(java.awt.Color.RED);
           f2.setBackground(java.awt.Color.LIGHT_GRAY);
           f1.setBackground (java.awt.Color.LIGHT_GRAY);
           f4.setBackground (java.awt.Color.LIGHT_GRAY);
          f5.setBackground (java.awt.Color.LIGHT_GRAY);

           break;


          }

      }

     catch(Exception e)

     {

      }
```

CHAPTER 5
TESTING

Chapter 5

TESTING

To demonstrate the working of our proposed wireless multi-hop ad-hoc network based elevator system, a Java based Graphical User Interface (GUI) was designed and implemented on the laptops representing the floors. Based on these, several combinations of traffic conditions and changes were tested, of which we present the test-case as described in Fig 5.2.

5.1 The Graphical User Interface (GUI)

The GUI has been designed in such a way so as to receive input from the user about the up/down traffic of a particular floor, and to display the current status/position of the elevator, the current traffic at each floor, etc. A log of all received FIPs is also displayed at each floor which helps trace back the steps of the elevator, status of each Floor during these steps, etc. during the analysis. The following snapshot(Fig.5.1) shows and describes the GUI that we have implemented to simulate our elevator system:

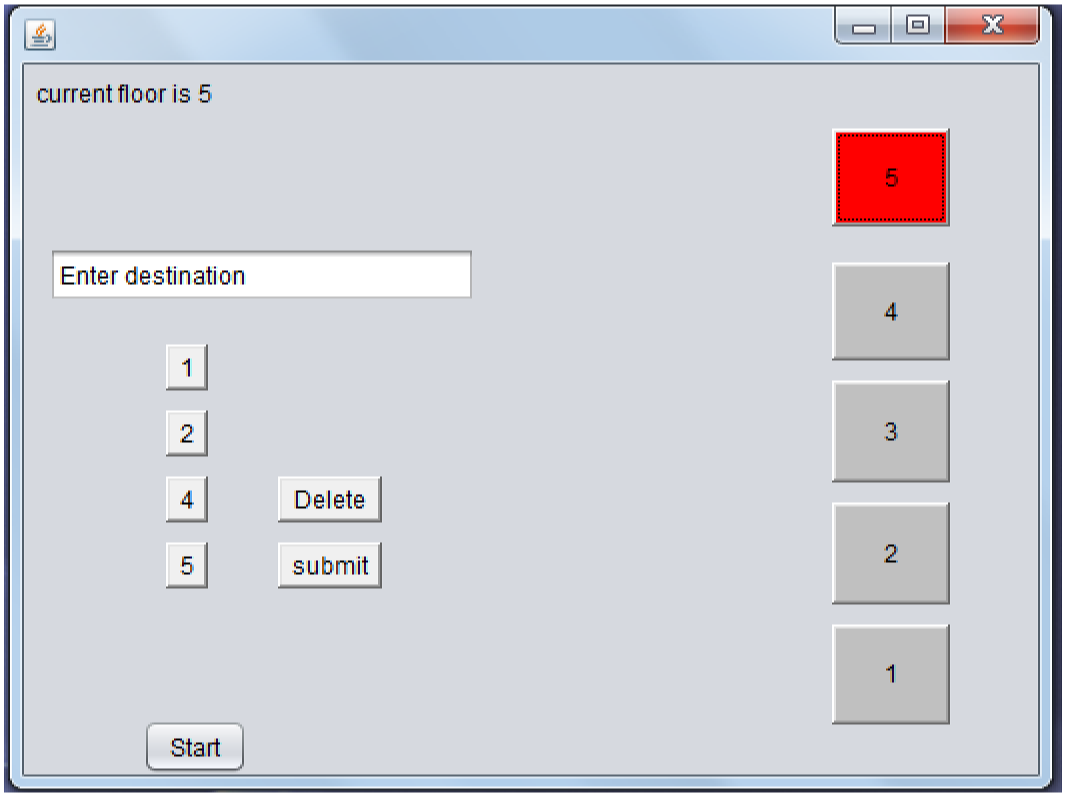
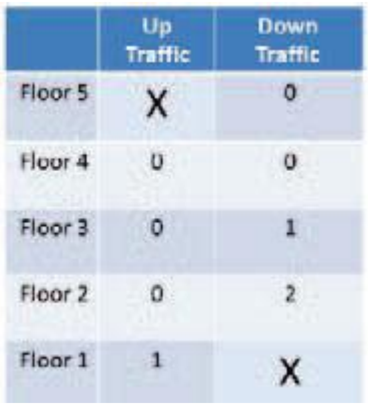


Figure 5.1 A snapshot of the GUI

5.2 A Sample Test-Case

We have tested our proposed smart elevator system considering a five floor building with a single elevator. With the traffic conditions depicted below in Figure 5.2, the proceeding points describe how the algorithm operates.



	Up Traffic	Down Traffic
Floor 5	X	0
Floor 4	0	0
Floor 3	0	1
Floor 2	0	2
Floor 1	1	X

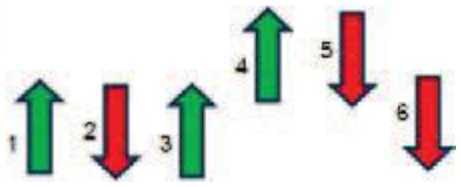


Figure 5.2 Sample test-case for our smart elevator system

To begin with, the elevator is present at Floor-1. The following steps describe how the different floors/people will be served and what exactly happens at each stage.

1) At Floor-1:

Traffic of Above Floors= 3, Up-Traffic = 1, Traffic of Floors Below = 0 (no floor below), Down-Traffic = 0. Since (2+1 + 1) > (0 + 0), Elevator is sent upwards to Floor-2 and Floor 1 up-Traffic becomes 0 (Now served).

2) At Floor-2:

Traffic of Above Floors= 1, Up-Traffic = 0, Traffic of Floors Below= 0, Down-Traffic =2. Since (1 + 0) < (0 +2), Elevator is sent downwards to floor 1 and Floor 2 down-Traffic becomes 0 (Now served).

3) At Floor-1:

Traffic of Above Floors= 1 (at Floor-3), Up-Traffic = 0,

Traffic of Floors Below = 0, Down-Traffic =0, Since
(1+ 0) > (0 + 0), Elevator is sent upwards to Floor-2 to be
sent to Floor-3.

4) At Floor-2:
Traffic of Above Floors= 1, Up-Traffic = 0, Traffic of
Floors Below = 0, Down-Traffic =0, Since (1 + 0) > (0 +
0), Elevator is sent upwards to Floor-3.

5) At Floor-3:
Traffic of Above Floors= 0, Up-Traffic = 0, Traffic of
Floors Below= 0, Down-Traffic =1, As (0 + 0) < (0 + 1),
Elevator is sent downwards to Floor-2.

6) At Floor-2:
Traffic of Above Floors= 0, Up Traffic = 0, Traffic of
Floors Below= 1, Down Traffic =0, Since (0 + 0) < (0 +),
therefore the Elevator is sent downwards to Floor-1.

Finally, if there are no requests from any floor, the elevator remains at the current floor and continuously waits for any new requests or changes in traffic conditions.

CHAPTER 6
RESULTS

CHAPTER 6

RESULTS

Case 1: The elevator is in the 5th floor. It takes the traffic information from the other floors and decision is made based on the collected traffic information.

The current floor is 5:

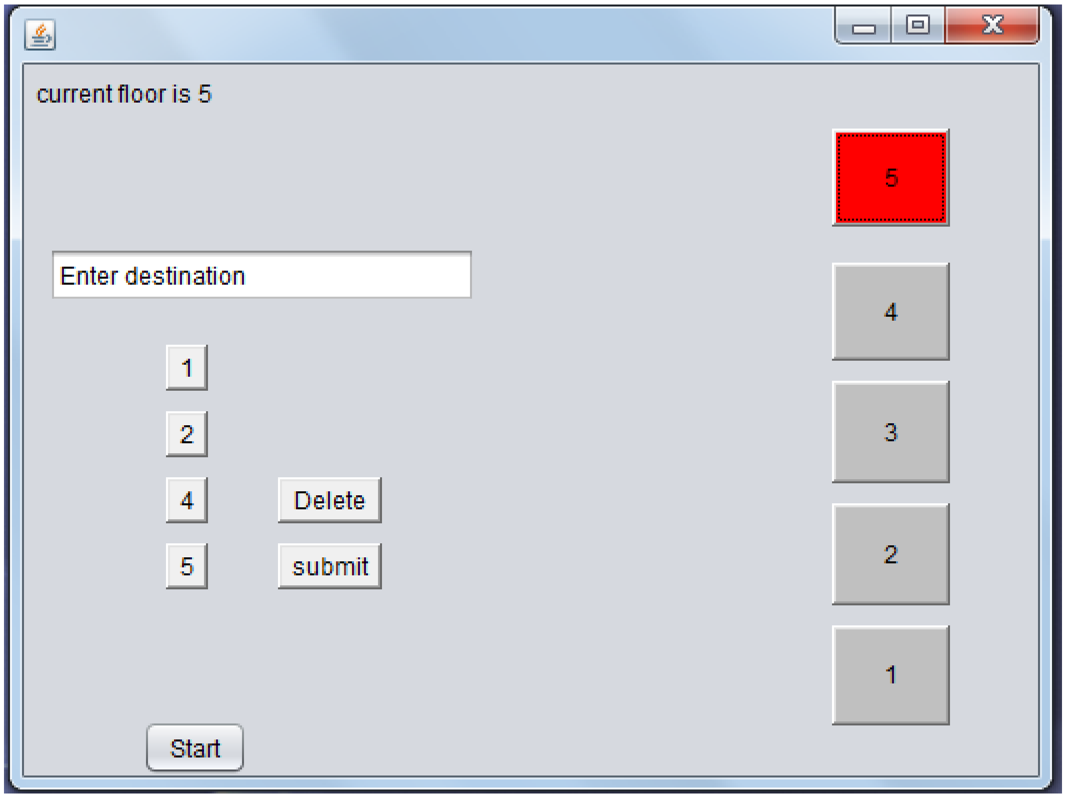


Figure 6.1 Initial state of the elevator

The inputs are:

Floor no	Inputs	UT	DT	TAF	TBF	Total Up traffic	Total down Traffic	Elevator position
5	4	0	1	0	4	0	5	5
4	3,2							5
3	2,4							5

UT=0   DT=1

TAF=0   TBF=4

(UT+TAF)=0 (DT+TBF)=5

Since down-traffic > up-traffic the elevator moves downwards.

Due to the decision made in the 5th floor, the elevator moves to the 4th floor as show in the figure 6.2.

Case 2: The elevator is in the 4th floor. It takes the traffic information from the other floors and decision is made based on the collected traffic information

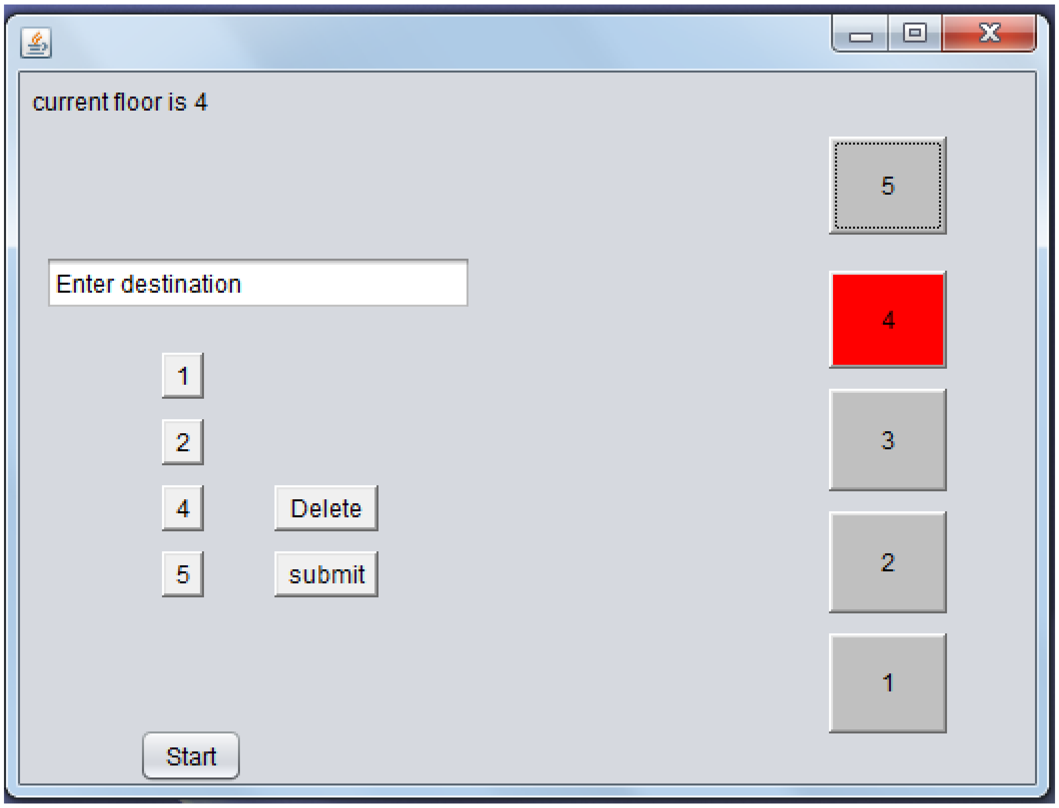


Figure 6.2 Second state of the elevator

The inputs are:

Floor no	Inputs	UT	DT	TAF	TBF	Total Up traffic	Total down Traffic	Elevator position
5	2,3							4
4	1,5	1	3	2	3	3	6	4
3	5							4

UT=1   DT=3

TAF=2   TBF=3

(UT+TAF)=3 (DT+TBF) =6

Since down-traffic > up-traffic the elevator moves downwards.

Due to the decision made in the 4th floor, the elevator moves to the 3rdth floor as show in the figure 6.3.

Case 3: The elevator is in the 3th floor. It takes the traffic information from the other floors and decision is made based on the collected traffic information.

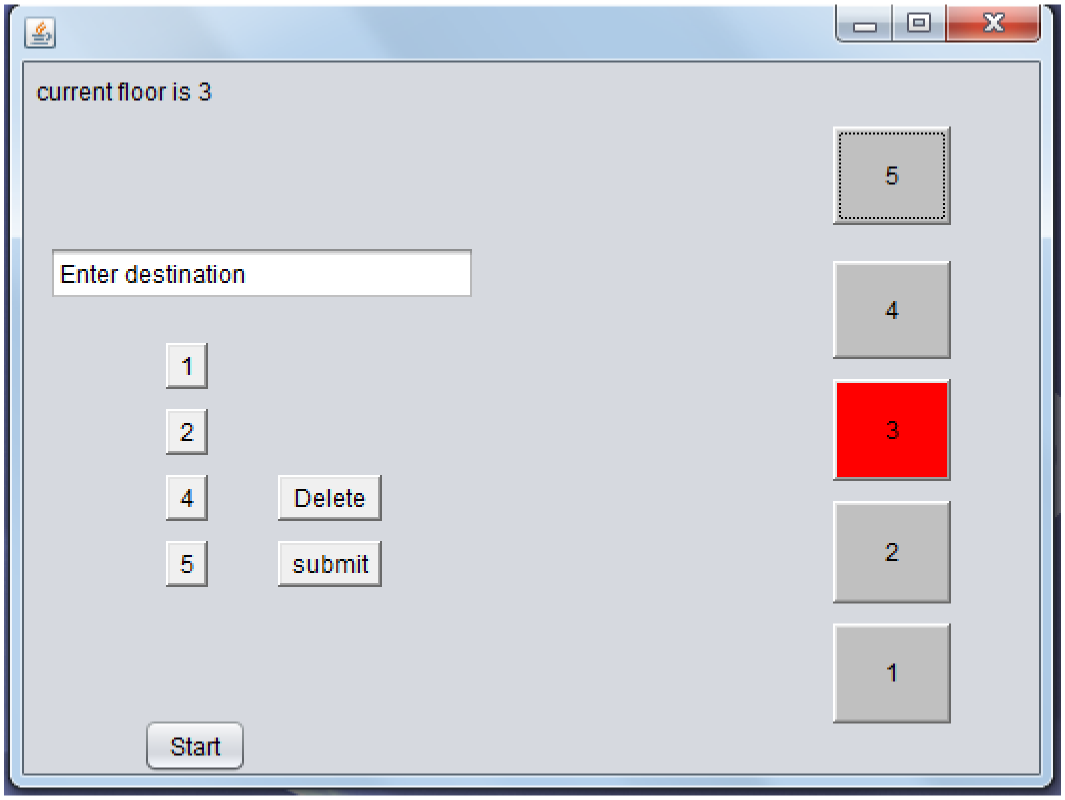

current floor is 3

Figure 6.3 Third state of the elevator

The inputs are:

Floor no	Inputs	UT	DT	TAF	TBF	Total Up traffic	Total down Traffic	Elevator position
5	1,4							3
4	2							3
3	2,4	4	4	5	0	9	4	3

UT=4   DT=4

TAF=5   TBF=0

(UT+TAF)=9 (DT+TBF)=4

Since down-traffic < up-traffic the elevator moves upwards.

Due to the decision made in the 3th floor, the elevator moves to the 4th floor as show in the figure 6.4.

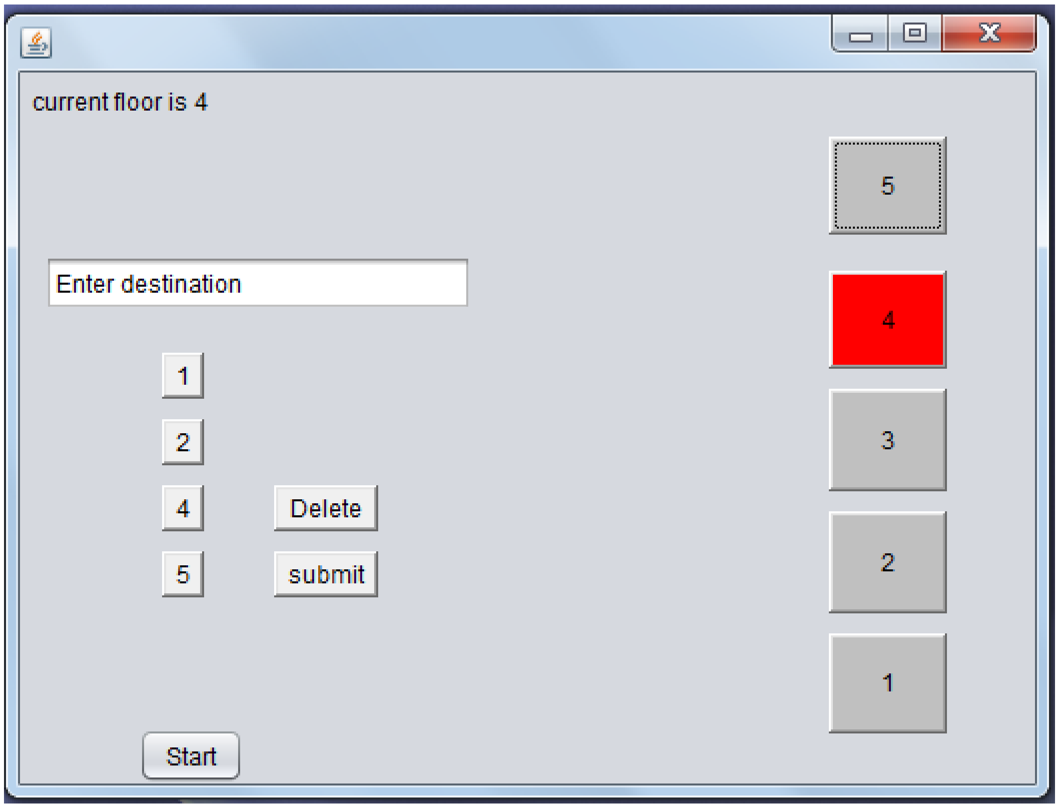


Figure 6.4 Fourth state of the elevator

6.1 APPLICATIONS

a) Educational Institutions with facilities located at the higher floors.

b) To provide an alternative instead of escalators in shopping malls to reach higher numbers of floors in lesser time.

c) High Storied Buildings.

CHAPTER 7
CONCLUSION

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

A smart elevator system has been designed, implemented and tested utilizing wireless ad-hoc nodes that are connected in a multi-hop fashion and sense passenger-traffic in real-time. Furthermore, an algorithm developed to implement and prioritize the elevator system based on the traffic inflow such that all the floors communicate with each other and share their traffic information to optimize and control the movement of the elevator. To summarize, the elevator system is implemented in such a way that it serves those floors first which have the greater traffic.

7.2 FUTURE WORK

In buildings where there is a group of elevators and not just a single one as we consider in this project, more complex inter-floor traffic information awareness mechanism and elevator dispatching algorithms would have to be developed to achieve the main goal of serving more people in lesser time. Furthermore, the smart elevator system should be equipped with some mechanism to keep track of people inside it at any given time, and also of people going in/out as there are certain situations when the proposed algorithm could face bottlenecks when it has reached its passenger limit.

Other issues such as those related to security of the wireless ad-hoc network of laptops, failure of nodes, information update delays in case of high-rise buildings, enabling emergency or manual interruptions and controls, etc. also need to be researched and analysed in order to come up with a completely robust and applicable solution of a smart elevator system.

Considering the results obtained by testing our proposed system and the above mentioned challenges, there is definitely a wide array of opportunities for research into the implementation of a smart elevator system based on wireless multi-hop ad-hoc networks.

REFERENCES

REFERENCES

[1] Hamza Ijaz Abbasi, Abdul Jabbar Siddiqui, Implementation of Smart Elevator System based on Wireless Multi-hop Ad-Hoc Sensor Networks, IEEE 2011.

[2] Zhen Shen , Qian-Chuan Zhao, "Ant Colony Optimization for Single Car Scheduling of Elevator Systems with Full Information", IEEE 2009.

[3] G.G Clerk, P Mehta, R Prowse,"Knowledge-Based Elevator Controller", IEEE 1994.

[4] http://docs.oracle.com/javase/tutorial/networking/sockets/ ,website link for information related to java socket programming

[5] Herbert Schildt, the Complete Reference Java-7th Edition, TMH.

.

APPENDIX

APPENDIX

Table 1. Acronyms

API	Application Program Interface
JDK	Java Development Kit
UT	Up Traffic of current floor
DT	Down Traffic of current floor
TAF	Traffic of Above Floors
TBF	Traffic of Below Floors
JVM	Java Virtual Machine
AWT	Abstract Window Toolkit
UDP	User Datagram Protocol
FIP	Floor Information Packet
EIP	Elevator Information Packet