Autonomous Buses
Dynamic Bus Routes

Ian Richardson
B.A.(Mod.) Computer Science
Final Year Project April 2011
Supervisor: Dr. Donal O'Mahony

School of Computer Science and Statistics
O'Reilly Institute, Trinity College, Dublin 2, Ireland
Autonomous Buses

April 11, 2011
Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted as an exercise for a degree at any other university.

______________________________  April 11, 2011
Ian Richardson
Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

________________________________________  April 11, 2011
Ian Richardson
Acknowledgments

I would like to thank my supervisor Dr. Donal O’Mahony for the wealth of advice and encouragement he has given me throughout the project.

I would also like to thank all my friends and family for all their help and support during this project.
Abstract

Fixed bus routes can appear to be wasteful of both resources and time. Transport systems are now being designed with technology integrated into their infrastructure to improve efficiency levels. This report discusses these transport systems, such as Nakamura cities On-Demand bus service, and documents the design and testing of 2 dynamic bus routes. The approach taken in this project was to compare both dynamic bus routes against a fixed bus route already in operation using the designed simulator created for this project. Performing a number of tests, each test containing a map of 23 bus stops and 210 passengers, against an existing fixed route is used to establish efficiency levels of both dynamic routing techniques. The tests show the quality at which a bus can perform using either technique. The results confirmed that both dynamic routing methods performed better than the fixed route in relation to buses needed for passengers served.
Contents

1 Introduction 10
  1.1 Motivation ............................................ 10
  1.2 Project Goal .......................................... 10
  1.3 Outline ................................................. 10

2 State of the Art 12
  2.1 Introduction ........................................... 12
  2.2 Intelligent Transport Systems .......................... 12
    2.2.1 Smart Cars ........................................... 12
    2.2.2 Monitoring Systems .................................. 13
    2.2.3 On-Demand Bus Services ............................ 15
  2.3 School Bus Route Problem ............................. 18
    2.3.1 Algorithmic Approach to the SBRP ............... 18
    2.3.2 RTPI Approach to the SBRP ....................... 20
  2.4 Wireless Communication and Handheld Technologies 22
    2.4.1 GSM/GPRS ............................................ 22
    2.4.2 GPS .................................................. 22
    2.4.3 Smart Phones ....................................... 22
    2.4.4 High Speed Fibre Links ............................ 23
    2.4.5 Web Based Maps .................................... 23
  2.5 Review .................................................. 25

3 Design 26
  3.1 Introduction ........................................... 26
  3.2 Components ............................................. 26
    3.2.1 The passengers ..................................... 26
    3.2.2 The Bus Stops ...................................... 27
    3.2.3 The Bus ............................................. 28
    3.2.4 Further Decisions - The Roads .................... 29
    3.2.5 Representation of Interaction ..................... 31
  3.3 Routing Techniques ...................................... 33
List of Figures

1  Information Displayed To Passenger[17] ......................... 14
2  Current Issues With On-Demand[22] ............................. 16
3  Concept of ARL Strategy[6] ................................. 19
4  Variables and Constraints used for Heuristic method[26] ....... 20
5  The Passenger Object ........................................ 27
6  The Bus Stop Object .......................................... 28
7  The Bus Object ............................................. 29
8  The Road Object ............................................. 30
9  UML Class Diagram .......................................... 31
10 Zero-Lookahead Flow Chart ..................................... 34
11 Formula used to calculate potential diversions: Zero-Lookahead 35
12 One-Lookahead Flow Chart ..................................... 37
13 Formula used to calculate potential diversions: One-Lookahead 38
14 Current Position of Bus ....................................... 41
15 Check All Bus Stops .......................................... 42
16 Create Passengers from Log File ................................ 43
17 Mapped Area[9] .............................................. 49
18 Fixed Route Estimations ....................................... 51
19 Dynamic Route Estimations ................................... 52
20 Histograms of Time Spent on a Bus ............................ 53
21 Graph of Time Spent at a Bus Stop ............................ 55
22 Test 1 Results ................................................ 57
23 Histograms of Time Spent on a Bus ............................ 57
24 Graph of Time Spent at a Bus Stop ............................ 59
25 Test 2 Results ................................................ 61
26 Histograms of Time Spent on a Bus ............................ 62
27 Graph of Time Spent at a Bus Stop ............................ 64
28 Test 3 Results ................................................ 65
1 Introduction

1.1 Motivation

Currently, 25 percent of the world's greenhouse gas emissions come from transportation[11]. New methods of creating and maintaining transport systems in order to reduce this figure are being deployed. These transport systems are known as Sustainable Transport systems.

"A sustainable transport system is one that is accessible, safe, environmentally-friendly, and affordable." European Conference of Ministers of Transport (ECMT 2004)

A sustainable transport system has many different definitions, but all have the same underlying meaning, a transport system that is cost effective for both the organization who own it and the passengers who uses it while limiting the damage done to the environment.

Fixed bus routes are perceived to be wasteful. They can result in such things as Ghost Buses, where a bus travels the entire route without picking up a single passenger. Fixed routes can also be extremely inefficient for passengers. As the bus is required to travel a pre-determined route, this predetermined route might not be the most optimal or efficient route in order to get the passenger to their desired destination.

1.2 Project Goal

The aim of this project is to investigate the efficiency of a dynamic bus routing system over a fixed bus routing system.

1.3 Outline

State Of The Art discusses how IT integrated into transport systems have been used to improve their efficiency. Design discusses the design of a simulator and
two dynamic routing algorithms used to do the comparison of bus routing. **Implementation** discusses what language and data structures were used to create the simulator and algorithms and why they were chosen. **Evaluation** discusses the testing that took place in order to measure the efficiencies of all 3 routing techniques and each method’s results. **Future Work** discusses different ways in which this project can be scaled up. **Conclusion** concludes the project with a brief overview of the entire project, why the project was beneficial and what the project found.
2 State of the Art

2.1 Introduction

This Chapter discusses the background to, and advantages of Intelligent Transport Systems as well as looking at how different technologies and methods have been used to improve the efficiency of transport systems in recent years. This Chapter will also outline a common algorithmic problem, known as the School Bus Routing Problem, and some of the possible solutions that have been put forward to address it.

2.2 Intelligent Transport Systems

Intelligent Transport Systems are transport systems that incorporate IT into their infrastructure in an effort to improve certain aspects such as distance traveled by the vehicle, passenger journey time/satisfaction and fuel consumed by the vehicle. There are many different technologies that have been used in order to improve these systems such as multiple forms of wireless communication technologies[18][12], like 3G and WiMAX, positioning technologies such as GPS and sensing technologies[16], such as sonar.

2.2.1 Smart Cars

The ability for a car to drive itself has become a more realistic goal with improvements in sensing and wireless communications technologies. Google are just one of a number of companies who have done large amount of research in the area of driverless smart cars. These smart cars are also able to communicate with other smart cars within an area in order transfer information about possible obstructions or traffic congestion.

Google have produced a vehicle that uses video cameras, radar sensors and a laser range finder to identify surrounding traffic and drive safely along a predetermined map. An algorithm that interleaves mapping with a probabilistic
technique to identify measurements is used to dynamically map 2d and 3d environments. These sensors combined with the algorithm to map 3d environments dynamically[15] have allowed the Google car to successfully navigate more than 140,000 miles with a human chaperone[14].

2.2.2 Monitoring Systems

The monitoring of vehicles in a transport network is currently growing in popularity. Systems are being deployed which allow passengers the ability to see real time information about their desired vehicle, such as time of arrival or current location.

The Tottori University in Japan[17] developed a bus monitoring system that would give people the facility to remotely view where a bus is at any given time. Smart phones equipped with GPS capabilities are installed into each bus. A message is then relayed back to a central data base. This message contains information such as transmission time, bus stop departure id and current location co-ordinates. The central database updates the details displayed on its website about each vehicle when ever a message is received. These messages are sent every minute. The passenger is able to review information such as what bus stop the bus is closest to and what route the bus will be taking before it arrives at their desired stop, all shown in Figure 1.
The system also calculates the possible delay of the bus arriving at a bus stop by an algorithm that uses the straight line distances from the bus' current location to all bus stops. The difference between passage schedule time of the bus stop that has the shortest distance and the current time is the estimated delay for all bus stop arrival times. This system was deployed in the local surroundings of the University with great success due to its low cost compared to other monitoring systems and its accurately displayed passenger information.

Dublin City Council have also started deployment of a more sophisticated bus monitoring system\cite{21}\cite{2}. Automatic Vehicle Location and Control (AVCL) systems have been installed throughout the Dublin Bus fleet. These systems transmit each bus’ current location back to the closest monitoring point and to Dublin City Councils SCATS(Sydney Coordinated Adaptive Traffic System) traffic control central computer via the existing bus radio system, thus allowing for greater bus prioritization at signal controlled junctions. The systems will also update the estimated time of arrival at each bus stop to ensure the Real Time Passenger Information (RTPI) is accurately displayed. Fibre optic links that can transfer at approximately 10GB/sec are used to link the passenger displays to CIE central database. The GPS devices within each bus relays it’s co-ordinates, along with
other information such as vehicle number and destination, once every 20 seconds depending on radio coverage and the database translates this information into RTPI requirements and updates relevant displays.

There are also bus monitoring tools available that allow individuals or organizations to design and manage their own bus route system. Futurefleet is one such product[3]. The system is broken up into three specific service styles. The management of passenger services, the in-vehicle services and the operations services. These three services allow for passengers to monitor how long it will take for a bus to arrive at their desired bus stop, allow for the system to monitor how long each journey is taking and how much money is being charged, and also allow for the bus to be monitored as regards their current position and their schedule. This system uses the Google Maps API which enables a visual representation of the current location of the bus. Each bus is equipped with a Mobile Data Terminal (MDT). This terminal has a powerful GPS receiver built-in which enables for more accurate real time capabilities. The central server can monitor the movements of each vehicle and record the vehicles progress and also update a Google Map. The MDT also allows each bus to be provided with location based schedule adherence information as they drive which ensures a more predictable service provided to passengers and reduces the chance of bus bunching.

2.2.3 On-Demand Bus Services

On-Demand bus services means that buses are only dispatched to bus stops specified by users via information service terminals[8]. Such systems results in a much more passenger-efficient service being provided, while also reducing the amount of buses required to service an area. Nakamura City, in Japan, has adopted an On-Demand bus service system. Passengers reserve the time, via an information service terminal or by telephoning an operator, at which they require a bus to arrive at their stop. If no passenger has placed a reservation, then no bus will leave the terminus. Nakamura City have now seen its number of bus users quadruple with a bus service that offers 8 times as much service provided to passengers at
only 2.4 times the cost of conventional circulating buses (On-Demand runs at a third of the cost of conventional)[8]

On-Demand bus services can have issues around time keeping[22]. The general idea is that passengers traveling in similar directions travel on the same bus. That bus then makes only slight diversions for other passengers. The problem arises with the diversions and the fact that these systems do not take into account possible time constraints of passengers or priority passengers (if a passenger places their reservation before other passengers).

![Figure 2: Current Issues With On-Demand][22]

A passenger may place a reservation, via an information service terminal or by telephoning an operator, to arrive at a certain destination at a specific time. The shortest route is then calculated and a pick-up time reported. If a new passenger places a similar request, the system decides on a new route which accommodates both passengers, but then invalidates the estimated arrival time as shown in Figure 2. With the introduction of priority passengers, and time constraints, the on demand bus system would be able limit the amount of diversions it would make in a journey. The system would also be able to give more accurate estimations on a passengers arrival time at their destination.

With the introduction of such constraints, a vehicle choosing algorithm and a routing algorithm must be used in unison to decide which passengers are to be
picked up by what buses[25]. Firstly the vehicle that is closest to the reservation is calculated. A direction vector for a customer is declared based on their requested point of collection and destination. Similarly, a direction vector is defined for each bus. A direction decision variable is then defined from the $\cos$ of the angle between the two vectors. The bus with the lowest value then has a routing algorithm based on a time limit and capacity levels performed on it. If either the request exceeds the capacity levels of the bus or time limits of passengers on the bus, then the next closest bus is calculated and the routing algorithm is performed on the new bus.
2.3 School Bus Route Problem

The School Bus Route Problem (SBRP) has been the starting point for many research groups\[6\]|\[7\]|\[26\] that are trying to devise more efficient transport systems. The SBRP deals with calculating the most efficient route a school bus should take in order to deliver students to their designated schools. This route must be calculated while satisfying certain constraints of the traveller, the bus and the school. This problem has been undertaken by either redesigning the areas of pick up for students and then the sequence which a bus traverses these areas\[6\], or by using web based maps and wireless communication technologies to improve the performance of existing routes\[7\]|\[26\]. This section will discuss an algorithmic approach to redesigning a bus route based on passenger demand. This method requires prior knowledge to all passenger locations. This section will also discuss an algorithm to divert a bus based on real-time information. This method relies on prior information about passenger locations in order to plot a potential route for the bus and then calculates possible diversions, in real-time, from all passenger requests received after the bus has left the terminus.

2.3.1 Algorithmic Approach to the SBRP

In 1995, a heuristic algorithm that would tackle the issue of the SBRP was devised by the University of Waterloo\[6\]. The algorithm was designed around satisfying three specific criteria: efficiency, effectiveness and equity.

Firstly, efficiency was used to measure the level of service provided compared to the cost of resources used. Secondly, effectiveness measures how well the passenger demand is satisfied by the service. And finally, equity looks at the fairness of the service, for example assuring passengers are not on the bus significantly longer than other passengers. The algorithm also takes into account certain route constraints such as bus capacity, travel time on the route, and travel time of the passenger.

The development of the heuristic algorithm was divided into three sub problems. The location of the bus stops, assigning students to the bus stops, and then
generating a route that traverses the bus stops. The location of the bus stops was generated using both Location-Allocation-Routing (LAR) and Allocation-Routing-Location (ARL). LAR assigns students to already placed bus stops that are of maximum distance to them and then decides a route based on these bus stops. ARL was more advantageous for this algorithm as it defined the locations of students around the school and then placed bus stops based on clusters of students. This section will therefore only discuss the ARL method shown in Figure 3 and not the LAR method because of the improved solution method discussed below.

Figure 3: Concept of ARL Strategy[6]

Using the ARL strategy leaves only one issue to be resolved: to define an appropriate route for the bus to take. From the bus stop locations, calculated using the ARL strategy, an efficient route is calculated for each bus by the using 3 algorithms sequentially. The first algorithm defines a set of bus stops for all possible routes. Every student within that region is then assigned to a bus stop of maximum distance from their location. The next algorithm then selects the route with the least total weighted distance of the students from their bus stops. And finally the third algorithm used, attempts to reduce the selected routes total weighted distance, by introducing more bus stops to the route. The results from these algorithms then defines the most efficient route for a bus to take while observing the previous constraints.
This method shows that both the selection of the bus stops, and how the route is calculated for a single school are NP-hard problems\cite{23}.

### 2.3.2 RTPI Approach to the SBRP

In 2009, wen-juan wang et al. \cite{26} purposed a paper that documented a simulation of the SBRP on a college campus and which also allowed the vehicle divert from its planned course. This method puts forward a way of reducing cost when density of customer requests is low and the need for a new bus route is not efficient. The method assumes that the most efficient route and location of bus stops have already been defined that attribute to passenger demand. The method simulates a Demand Responsive Transit (DRT) system that allows for the immediate response to random requests of passengers during a vehicle’s route.

\[ V \] Vehicle;  
\( L_i \) The location of the depot \( i \);  
\( t_o \) Call-in time of \( i \);  
\( t_r \) Response time of \( i \);  
\( t_{pi} \) Pick-up time of \( i \);  
\( t_{wi} \) Waiting time response to \( i \);  
\( c_{ij} \) Cost per mile from \( i \) to \( j \);  
\( T_i \) The time window of passenger \( i \);  
\( n \) The current total number of passengers;  
\( c_{fi} \) The flexible cost of passenger inconvenience;  
\( c_{si} \) The static cost of passenger inconvenience;  
\( k_i \) Quantity of the static cost of passenger inconvenience.

\[
\text{Min} \left[ \sum_{i=1}^{n} |L_i - L_j| c_{ij} + \sum_{i=1}^{n} \frac{t_{pi} - t_{wi}}{T_i} c_{si} + \sum_{i=1}^{n} k_i c_{si} \right] / n  
\text{ s.t. } t_r \geq t_{pi}  
\text{ and } t_{wi} \leq t_{pi} - t_{wi} \tag{2}  
T_i \geq t_{wi} \tag{4}  
k_i = \begin{cases} 0 & t_{pi} \in [t_{wi}, t_{wi} + T_i] \\ 1 & t_{pi} \notin [t_{wi}, t_{wi} + T_i] \end{cases} \tag{5}  
c_{ij} \geq 0 \tag{6}  
c_{ji} \geq 0 \tag{7}  
c_{si} \geq 0 \tag{8}  
\forall i, j \in n \text{ and } i \neq j \tag{9}
\]

Figure 4: Variables and Constraints used for Heuristic method\cite{26}

The route of the bus is calculated in two ways. Firstly, a static algorithm is used on requests received prior to the bus’ departure. This generates the most optimal initial route based on passenger clusters. Secondly, while the bus is on route, it’s schedule is potentially updated based on the results from the model shown in
Figure 4. Figure 4 (a) defines the variables that are used, and (b) defines how the constraints are calculated using those variables. The first formula on Figure 4 (b) is used to calculate the minimum average cost along with passenger inconvenience.

The subsequent formulae check the variables against constraints. If the constraints, displayed in Figure 4(b), are met and the average passenger cost satisfies an optimization algorithm, then the current route is adjusted. Otherwise the diversion is not deemed efficient and the bus does not re-route. The optimization algorithm used is a combination of the initial route calculation algorithm and ant colony algorithm. Ant colony algorithm is where the shortest path between two points is built from a combination of several paths.[10]

The simulation that was documented was based on exploiting the advances of communication systems, and the ability to have real-time information. GIS (Geographic Information System) information that has become readily accessible in recent years[7] is also a major influence on this simulation.
2.4 Wireless Communication and Handheld Technologies

Many different technologies have been utilized to improve the efficiency of public transport. This section will discuss some of the technologies that have been or are currently being implemented in the design and running of public transport systems.

2.4.1 GSM/GPRS

GSM is a common architecture for mobile communication networks. GPRS is an extension of GSM and allows for higher data rate transmissions[13]. Such communication technologies have enabled passengers to receive details about transport networks directly to their phone. The introduction of 3G and WiMax networks have also improved data transfer speeds allowing for more accurate RTPI displayed. The deployment of these services on a large scale by telecommunication companies have allowed transport organizations to use these technologies to improve their systems at minimum cost. Providing information directly to passengers phones is also a lot cheaper than sending information to bus stop displays. A charging mechanism can also be introduced to passengers to subsidize this cost.

2.4.2 GPS

GPS is a global navigation system that provides accurate information about a receivers current location. GPS capabilities are now becoming a standard additions to many new devices. Its integration into transport systems means that passengers can either monitor the exact location of their desired vehicle, or have information about the vehicle displayed to them. With the combination of the previously discussed wireless technologies and GPS, the information displayed to passengers is becoming more accurate.

2.4.3 Smart Phones

As the number of smart phone users continues to grow[20], transport networks have ability now the to deliver more information to their passengers by designing
applications that give passengers real time information directly to their phone. Applications such as Avego Driver[3] allow individuals to use their own vehicle to pick up people. They use the GPS and wireless network capabilities of a Smart Phone to share the current location of a vehicle with other passengers using the application. Passengers then have the ability to see many different vehicles on different routes, in real time. If a route displayed by the application is useful to the passenger, they can send a request to the driver to travel with them. The driver then receives the request, in real time, and can either confirm or deny to collect the passenger. If a driver accepts the passengers request, a pick up point is agreed upon. The driver has complete discretion over who they collect and if possible diversions are optimal for their desired journey. This type of application is exempt from possible Taxi Licensing issues because it is for non-profit use. This system alleviates the need for people traveling in similar directions to travel in separate vehicles.

Smart Phones have also been installed into current transport systems[17] as a monitoring tool. The cost to install and maintain these systems is relatively low.

2.4.4 High Speed Fibre Links

Many transport monitoring systems are designed using a central server[21][17] which contains all the information about the network. Buses relay GPS information to this server and the server then distributes relevant information to the passenger displays. The bus monitoring system in Dublin uses high speed fibre optic cables to relay the information to the passenger displays. These cables allow for faster transfer of information which has less chance of interference when compared to wireless technologies.

2.4.5 Web Based Maps

Google Maps and Open Street Map are web mapping service applications that provides people with detailed information about particular areas. Monitoring systems [21][17] use these applications by projecting the location of their vehicles onto these maps so that users have a visual representation of the current location of the
vehicle. For the Google Maps API, General Transit Feed Specification (GTFS) is a format used specifically to update a Google Map about a transport system. These formats comprise of various information such as bus stop locations, bus routes, arrival times and current bus locations along with other attributes. These files can then be uploaded through the API to the transport organizations map that displays the information to its passengers. Avego\cite{3} use these file formats for both their ride-share application and also their bus monitoring tool.
2.5  Review

Many different transport systems that utilize today's technology have been put forward in this section. It has also looked at a common problem, the school bus routing problem, which itself has triggered much research on finding a more efficient way to route buses. Finally the section documented some of the current technologies that are available and in use by many transport systems.

All the systems and methods that have been described throughout this chapter have common features. They each:

• Calculate a pre-determined route
• Have knowledge of entire system of bus stops
• Have knowledge of all new passenger requests
• Re-route a bus based on all the new passenger requests

With these features, each system is able to guide their vehicles to picking up as many passengers as possible in the most efficient way with regards to the system itself.

In order to ensure passenger satisfaction, certain systems introduced constraints/limits that would favour on the side of the passenger efficiency. These constraints/limits comprised of:

• Distance from a bus stops
• Time spent on a bus by a passenger
• Time to arrive at a passenger's destination

Some of the transport systems documented[22][26] have used these constraints in similar ways to create more efficient bus routes.
3 Design

3.1 Introduction

This Chapter will document the design of the simulator and both dynamic bus routing algorithms used in order compare a fixed bus route against two dynamic bus routing algorithms. The Chapter is broken up into three sections. Firstly, section 3.2 will discuss the elements common to all routes and then how they will be represented within the simulation. Secondly, section 3.3 discusses the design of the two dynamic algorithms that will be implemented within the simulation for comparison against the fixed route. Finally, 3.4 gives an overview of the design of the simulator and both dynamic methods.

3.2 Components

The common elements in all bus routes are the passengers, the bus stops and the bus itself. These elements need to be defined accurately in the simulator so that each method of route calculation can use the same objects. The following sections will define each element’s representation within the simulator. The section will also discuss how elements interact with each other.

3.2.1 The passengers

Passengers in a public transport system have basic characteristics. They have a place where they enter the bus and a place where they exit the bus. For the remainder of this report these points will be referred to as a passenger’s origin and destination. The point of this project is to compare efficient routing methods of buses, so the time at which the passenger arrives at their specified place of origin is also important. The time at which a passenger gets on the bus, and the time at which a passenger gets off the bus are recorded and will be used to calculate the journey time of that passenger.
Figure 5: The Passenger Object

Figure 5 shows how the passenger object will be represented within the simulator. The methods that are shown are all directly related to manipulating the attributes except for `returnTimeOnBus()` which is used for calculating the time spend on the bus by the passenger. This method, in conjunction with other methods, will be used to calculate the efficiency of a routing technique.

3.2.2 The Bus Stops

Bus stops have two major design attributes. A bus stop number, and a list of passengers associated with that bus stop. Figure 6 shows the bus stop object. Each bus stop has the ability to store details about multiple passengers. Each bus stop can also return or remove the details about individual passengers based on parameters such as a specific passenger or a passenger in a certain position in the list.
3.2.3 The Bus

Depending on the routing technique used during the simulation, certain attributes of the bus have the potential to change such as the list of bus stops the bus will be passing, the roads the bus intends to travel on and the passengers that are on the bus. However regardless of the technique used, the bus will always need to track its current location. The buses location will be monitored with reference to bus stops.

Figure 7 displays how the object is represented within the simulation. The bus will have different capabilities available depending on the routing technique used during the simulation (Fixed or Dynamic), but the common features with all the techniques will be the ability to add and remove passengers from the bus and report the current location of the bus. The current location of the bus is key to the dynamic routes and will be discussed in further detail in Section 3.3.

Methods such as `changeRoute(newRoute : LinkedList<busStop>)` will only be used during the simulation of a dynamic route.
3.2.4 Further Decisions - The Roads

The previous sections have defined the basic parts needed in order to simulate an accurate system. During the simulation’s design it became evident that there would need to be a further element added to the overall structure of the simulation.

The representation of roads was established as being the necessary missing element. For simulation purposes, the road object will only have basic attributes. The roads will contain data defining the bus stops at either end of it and also whether or not the road is a one way or a two way street. This will allow for backward traversal of routes. The road object is shown in Figure 8.
<table>
<thead>
<tr>
<th>roads</th>
</tr>
</thead>
<tbody>
<tr>
<td>name : String</td>
</tr>
<tr>
<td>oneWay : boolean</td>
</tr>
<tr>
<td>bS1 : busStop</td>
</tr>
<tr>
<td>bS2 : busStop</td>
</tr>
<tr>
<td>streetType()</td>
</tr>
<tr>
<td>returnName()</td>
</tr>
<tr>
<td>returnBusStops()</td>
</tr>
<tr>
<td>returnStart()</td>
</tr>
<tr>
<td>returnEnd()</td>
</tr>
</tbody>
</table>

Figure 8: The Road Object
3.2.5 Representation of Interaction

Figure 9: UML Class Diagram
Figure 9 displays the relationship between each component of the simulator. Accurate interaction between the components is key to the successful implementation of the simulator. The first elements that need to have interactions defined are the bus stops, as these are directly linked to all other elements. When a bus stop is initialized, it is given a number which represents its position in the topology. The initialization of the bus stop then allows for any number of passengers to be initialized with that bus stops number as their point of origin. When a passenger has been added to a bus stop, that passenger’s details can only be accessed through that bus stop. When more than one bus stop is being initialized, roads are needed to represent the link between these bus stops. A road must be initialized with two bus stops. These represent the start and end points of a road. Defining roads as a separate entity will also allow users of the simulator the ability to name roads and also define if they are one or two way. For this simulation, all roads have been defined as one way. The last element to be initialized is the bus. When initializing a bus, at least two bus stops are required and one road. These are used to symbolize the firstStation where the bus will start and the stations that the bus intends to pass. The stations are either fixed or dynamically altered. When the bus is first initialized, the firstStation is also the current bus stop. The current bus stop is one of the features used to establish what passengers, if any, will be getting on or off the bus. For this simulation, all buses will start at the first bus stop initialized. (Bus Stop Number 1)
3.3 Routing Techniques

This section discusses the two types of dynamic routing techniques that have been implemented for this project. The first routing technique that will be discussed, the Zero-Lookahead algorithm, uses some of the formulae and constraints discussed in the paper put forward by wen-juan wang et al. [26].

The second routing technique that will be discussed, the One-Lookahead algorithm, also uses some of the same constraints put forward by wen-juan wang et al. [26], but the way in which they are used is in the algorithm is different.

Both algorithms look at how long a passenger has been waiting at a bus stop, both algorithms give preference to the passengers already on a bus and both algorithms determine who to pick up based on the time it would take to reach a desired location, the length of time spent at the bus stop by the passengers and the length of time spent on the bus by the passengers.

3.3.1 Zero-Lookahead

The original idea behind the Zero-Lookahead algorithm is that a bus traveling towards a bus stop will make its decision on what direction to take based on the passenger attributes at the oncoming bus stop and the passengers, if any, currently on the bus. Issues arose from the fact that if there was not a potential route proposed to the bus when a bus picks up a passenger, then there would be no way of calculating estimated journey times. In order to perform any comparison, a route needs to be established before the bus leaves the first stop. When a bus picks up its first passenger, it calculates the shortest route between its current location and the passengers desired destination. This route then becomes the basis for any calculations or comparisons done when deciding what other passengers to collect. Once a bus has been given a potential route, it is then possible to decide what passengers are allowed onto this specific bus and if the potential route can be edited in such a way that the bus can service as many bus stops as possible with minimum inconvenience to the passengers currently on the bus. The passengers currently on the bus have priority.
Figure 10: Zero-Lookahead Flow Chart
Figure 10 shows the flow chart for the Zero-Lookahead algorithm. When a bus is on its way towards the next bus stop on its potential route, it checks if there are any passengers at that bus stop. If the bus stop is empty, and has not been requested by a passenger currently on the bus as their destination the bus does not stop at the bus stop. If however there are passengers at the oncoming bus stop, the bus needs to know whether or not to stop.

The bus communicates with the bus stop to get the details about passengers from that bus stop. The passengers at the bus stop are stored in a list in the order that they arrived thereby giving preference to the passengers who have been there longer. The eligibility of each passenger is checked. Firstly, if the passengers destination is on the current potential route then the bus must stop and pick them up. If there destination is not on the route, but can be appended onto the end of the current potential route then the bus must stop and pick them up. The last condition that is checked is the diversion possibility. A diversion is only possible if it does not interfere with any requests from passengers who are already on the bus.

\[(A + B) \leq (\text{Threshold} - C)\]

Figure 11: Formula used to calculate potential diversions: Zero-Lookahead

In Figure 11, \(A\) represents the estimated time to take the diversion from the current location, \(B\) represents the time needed to travel the distance between the end of the diversion and the passengers destination, already on the bus, who is currently being checked, \(C\) then represents the time spent by the passenger on the bus and \(\text{Threshold}\) represents the limit set for passengers to spend on the bus.

If the estimated time to take the diversion combined with the estimated time from the end of the diversion to the passengers destination is less than or equal to the maximum time set for any passenger to be on the bus, then the diversion is possible and the current potential route is edited accordingly.
Once all of the passengers have been checked at the bus stop the bus then moves to the next bus stop along its current potential route.

### 3.3.2 One-Lookahead

One-Lookahead checks similar constraints as to the Zero-Lookahead algorithm. The major difference between both algorithms is the group of passengers that each compare the passengers currently on the bus too. The One-Lookahead algorithm calculates its route based on a potentially far larger group of passengers. A potential route still needed for this algorithm.

This potential route is calculated by looking at the passenger request from the starting position and the immediate surrounding bus stops. Figure 12 documents how the route of the bus is updated, it also shows how passengers are deemed to be acceptable for the bus.

When a bus is traveling towards a bus stop on its current potential route, it looks at the oncoming bus stop and the bus stops that are within one hop of the oncoming bus stop. As we can assume the oncoming bus stop has been taken into account during the previous use of the algorithm, we only need to check if there are any passengers at the bus stop and if so, have their destinations been accommodated for in the buses current potential. The bus checks all the passengers at the bus stop and makes it’s decision on whether to stop based on their destination requests. Once the bus has checked all, if any, passengers at the oncoming bus stop, the bus then checks the passenger details at the bus stops that are one hop away. If there are no passengers at the surrounding bus stops then it continues on its route. Otherwise it checks the details of all passengers at the surrounding bus stops.
Passengers are organized into two lists. One orders the passengers by longest waiting time at the bus stops. The other orders people based on similarity of directions requested. The algorithm then switches between each list when checking the suitability of passengers. Firstly, if the passengers’ destination is already on the proposed route then the route of the bus is updated so that the bus passes the passengers’ origin. If a passengers’ destination can be appended to the end of the route, the buses route is edited so that the bus passes the passengers’ origin. Finally, if there exists a diversion then the route is edited to accommodate this diversion and the passengers’ origin and diversion is edited into the buses current potential route.

\[ A + (B + C) \leq (\text{Threshold} - D) \]

Figure 13: Formula used to calculate potential diversions: One-Lookahead

Figure 13 displays the formula used to calculate the diversion with this algorithm. \( A \) represents the estimated journey time from the current location to the passengers’ origin, \( B \) represents the estimated time to take the diversion form the passengers’ origin, \( C \) represents the time needed to travel the distance between the end of the diversion and the passengers destination, already on the bus, who is currently being checked, \( D \) represents the time already spent on the bus by the passenger and \( \text{Threshold} \) represents the limit set for passengers to spend on the bus. If this constraint is satisfied with all passengers currently on the bus, then the bus re-directs itself to accommodate the requesting passenger. Once all of the passengers have been checked at all surrounding bus stops, the bus then moves to the next bus stop along its current potential route and preforms the same methods among the new group of bus stops.
3.4 Review

This section has defined four key elements for the simulation.

- The Passengers
- The Bus Stops
- The Bus
- The Roads

It has also discussed each of the elements key attributes and dependency relationships between the elements. The elements that have been defined are key to all types of route generation. The implementation of this simulator allows for multiple types of route generation to be simulated. Two types of dynamic route generation were discussed during this section.

- Zero-Lookahead
- One-Lookahead

Zero-Lookahead allows for a possible route change based on the passengers currently on the bus and the passengers at the oncoming bus stop. The passengers on the bus have priority over the direction that the bus will take. Passengers at the bus stop can only enter the bus if their desired destination is already on the potential route, can be appended onto the route or a diversion exists that does not interfere with passengers already on the bus.

One-Lookahead allows for a possible route change based on the passengers currently on the bus, the passengers at the oncoming bus stop and the passengers at the bus stops that are one-hop away from the oncoming bus stop. Again passengers on the bus have priority over the direction that the bus will take. The bus will re-route itself to accommodate more passengers only if these passengers satisfy the routing constraints. One-Lookahead checks similar constraints to Zero-Lookahead but on a larger scale.
4 Implementation

4.1 Introduction

This chapter discusses how the simulation and routing methods were implemented. The chapter is divided into four sections. Firstly, Section 4.2 will discuss the choice of language made to complete the implementation of both the simulator and the dynamic routing techniques. Section 4.3 discusses the principal features of the simulated bus route and how these features were implemented accurately. Section 4.4 will discuss the different data structures used within each component of the simulation and the reasons why the chosen data structures are used. Lastly, Section 4.5 will give a brief review of the chapter and discuss a test that was run in order to clarify that the simulation was working.

4.2 Language and Libraries

The simulator that has been designed has many objects that need to interact with each other. Due to its highly object orientated design[1], Java was chosen as the best language to create the simulator along with each method of dynamic route calculation. Java would also facilitate the design of a simulator that is easily portable to a multitude of different platforms.

The Java language was also chosen due to the availability of libraries that were required for this project. A Java library that enabled the creation and use of specific data structures, which will be discussed in Section 4.4, was used[19]. This language also enabled the use of the Java Swing library which was used to create a suitable GUI.

The Java IO library enables data to be written too and from various files. This allowed for log file inputs of various passenger details and bus stop topology details which will be covered in the following sections.
4.3 Principal features

There are multiple features that are of importance in a system like this. Time is a key factor that needs to be represented within the construction of this simulator. This will then allow for the comparison of outputs with a defined unit of measurement. Buses need to have the ability to arrive at a bus stop and, based on their routing method, decide what passengers need to get on or off the bus. Passenger actions such as having the passengers arrive at the bus stops in a simulated real-time way need to be simulated. Dynamic routes also need to be calculated depending on the routing technique chosen.

4.3.1 Time

The simulation that has been designed in this paper has been slowed down to move through each state one second at a time. After every second the system performs various operations depending on the routing technique chosen, then pauses for the rest of the second. As the passenger comparisons and bus movements in the simulation are performed sequentially, Thread.sleep(1000); is the command called after every state.

4.3.2 Bus Movement

When a bus is given a bus route it must drive along that route. The system simulates a bus traversing its list of bus stops by incrementing to the next bus stop in the list every three seconds. If there are no passengers at the oncoming bus stop, then the bus is incremented to the next bus stop after only two seconds.

```
//check to see if the current stop is actually the last stop
if(buses.get(0).stations.getLast().equals(buses.get(0).returnCurrent()))
```

Figure 14: Current Position of Bus

Figure 14 shows the condition to check if the bus has reached the last stop along its current route. If this condition is satisfied, the simulation then checks for remaining passengers at any of the bus stops.
Figure 15: Check All Bus Stops

Figure 15 shows that if there are passengers remaining at any of the bus stops a boolean called `peopleLeft` is set. This boolean is then used to determine if a new bus is required. However, if `peopleLeft` is not set, this means that the current bus has reached its last bus stop along its route and all the passengers requests from the input file have been satisfied.

### 4.3.3 Passenger Actions

There are two phases required with describing the passenger actions. Firstly, the way in which a passenger arrives at their designated bus stops. And secondly how the passengers interact with the bus.

A passenger’s details are read in from a log file. Each line in the log file represents a new passenger. Passengers are created from this information. Figure 16 shows that the information taken from the log file is each passengers `origin`, `destination` and then the `time`. The passengers `origin` and `destination` represent the number of bus stop that they are designated to and requesting to travel to. The third value taken from the log file, the `time`, is used to represent the specific passengers arrival time at their `origin`. As the simulation is executing, after every `Thread.sleep(1000);` instruction an integer variable `i` is incremented.
The passengers that have been taken in from the file each have their time value checked against the current time of the simulation. If a passengers time stamp matches that of the current time of the simulation, that passenger is simulated to be arriving at the bus stop and must be registered to the bus stop associated with their origin value. Once the bus stop that matches the passengers origin is found, the passenger is subsequently added to the bus stop and a record is taken of the exact time the passenger arrives at the bus stop.

When a bus arrives at a bus stop the bus must check if any passengers are eligible to get onto the bus and also check if any passengers are eligible to leave the bus and then adds or removes accordingly.

If a passengers destination is equal to the bus stops number that the bus is currently at, then the bus removes the passenger from the bus and also takes record of the time that the passenger has disembarked. Recording the endTime of the passenger will enable a diagnostics of the simulation at the end to be printed out.

The bus must also check if there are any passengers at the current bus stop that
are eligible to enter the bus. This will depend on the routing method implemented in the simulation and will be looked at in further detail in Section 4.3.4. If a passenger has been deemed acceptable then they must be both added to the bus and then removed from the bus stop.

4.3.4 Route Generation

There are two routing techniques, Zero-Lookahead and One-Lookahead, used in this simulation. As the overall goal of this work is to compare dynamic versus fixed bus routing techniques, a fixed route was also implemented. Using the Zero-Lookahead algorithm, firstly a new bus is sent out along the topology of bus stops and it is designated a route in one of two ways. Either a passenger at the starting terminus defines a potential route for the bus, or there is no passengers at the starting terminus and a passenger somewhere along the topology requiring a bus defines the route. If the former is the case, then the bus must work out the shortest route get the passengers to their destination. If the later is the case, then the bus will not know where the passenger wants to go until it collects them and then calculates the shortest route. In this case, the passenger that has been waiting at a bus stop the longest, is the one who’s destination is taken into consideration. Dijkstra’s shortest path algorithm is used to calculate the roads between the current position and the passenger requested destination. Then the bus stops that are along the roads are calculated. After the potential route has been set for the bus, the bus proceeds along the route. Passengers are added to the bus if their desired destination is currently on the route, can be appended to the route or a diversion is possible. If a diversion is possible, the bus must re-calculate its route to accommodate both the requesting passenger and the passengers on the bus.

If there exists a route between the current bus stop and the requested bus stop, then this route is checked against each passengers destination, currently on the bus. Once the number of possible diversions is the same as the number of passengers, then there exists a diversion that does not interfere with any of the passengers destination requests currently on the bus. The shortest diversion is then chosen and subsequently compared against a threshold travel time of the passengers on
the bus.

Constraints are then checked against all passenger on the bus (the constants used are specific to the simulation ran and will be explained in Chapter 5), then the number of passengers that the diversion satisfies is then checked to ensure all passengers are ok with the diversion. If the threshold value is not reached for all passengers, then the bus route is edited. It is edited by performing an Ant Colony based approach[10]. The route from the original bus stop to the current bus stop is stored in one list. Then the new diversion is stored in another list. And finally, the remaining route from the end of the diversion to the remaining passenger stops is stored in a third list. The three lists are then appended together, and the bus continues traversing its new route.

One-Lookahead is similar too Zero-Lookahead in the way in which it checks if passengers are eligible to enter the bus with regards to appending destination requests to the end of the route and adding passengers if their destination is on the route already. Its route editing is based on a potentially larger passenger data base. The entire range of passenger associated with the oncoming bus stop and the bus stops that are one hop away are listed in two list. One contains the list of passengers in order of waiting time. The other contains the list of passengers in order of similar directions of destinations. The bus checks the number of diversions possible and then compares the diversions against each list of passengers, checking the constraints in a similar manor to to the Zero-Lookahead. The first route to be return that does not over go the threshold set for the passengers on the bus is chosen. The buses potential route is then edited in an Ant Colony fashion[10].

The fixed routes composed of hard coding in the route to the bus and not allowing any diversion to take place. The bus is then only capable of visiting these bus stops and serving the passengers who have origins and destinations along the route.
4.4 Data Structures

As there are so many different object types being represented in this simulation, using the correct data structures to store each of these is vital. The main data structure that is require for this simulation is the one used to represent the bus stop topology. The reason the bus stop elements of the simulation are the most important objects to structure accurately is because the passenger objects and bus objects are the data flow and the bus stop locations are static within the simulation.

For the bus stop topology a directed graph data structure was used. As this data structure is not standard to the Java language, a library was required. The information about a bus stop topology is stored in a log file. Each new integer value found in the log file is recognized as a new bus stop. When a value is read in, a new bus stop is created and added to the graph, unless the graph already contains the bus stop. As a directed graph is being used, the way in which the bus stops are added and connected to other bus stops is important. The log file contains the starting bus stop number, the ending bus stop number and the name of the road between them. The bus stop numbers were used to represent vertices and roads were used to represent the edges between these vertices.

If one of the bus stops read in from the log file is already represented in the graph, then it is not re-added. Otherwise both bus stops are added to the graph. The final piece of data taken from the log file is the name of the road between the two bus stops. This road is created with the first bus stop being the start point and the second bus stop being the end point. The graph checks if the edge already exists and then adds the edge accordingly. All roads added to this directed graph are one-way.

When a passenger arrives at a bus stop the bus stop must store the passengers information. Then when a bus comes along to that bus stop, it must be able to search that information easily to find the passengers eligible to enter the bus. The data structure chosen to store passenger information at the bus stops must be
able to add, remove and search through passenger details easily. A linked list was used to store passenger information at bus stops because they can grow and shrink dynamically and they allow for constant removal and insertion of passenger objects.

Storing the passenger details on the bus is also necessary. Being able to add and retrieve passengers is the main goal for this data structure. A hash table was originally used to store the details as it stored people with the same bus stop destination in the same location on the hash table. So when a bus was checking if passengers have requested to leave at the current bus stop, the bus will be able to see if there are multiple passengers who have requested this bus stop with one search. This was also the drawback to using this data structure. Hash tables lose their efficiency when there are continuous collisions and as this is a simulation of a bus network, there is expected to be multiple collisions. Therefore a linked list was chosen as the data structure to store passenger details on buses.

As the bus stops and roads that the bus will be traversing have the potential to change depending on the routing technique, the bus stops and roads have been stored in individual linked lists because of the dynamic size and ease of insertion and removal.
4.5 Review

This Chapter discusses the languages and libraries used to create the simulator and algorithms which will be used to compare fixed versus dynamic bus routes. The Chapter explains why Java was the language chosen due to its specific library accessibility, and also describes how actions within the simulation were created using the Java Language.

- Time
- Bus Movement
- Passenger Actions
- Route Generation

These are actions within the simulator that are necessary to creating an accurate simulation. Representing how time passes, how a bus moves through a topology of bus stops, how passengers interact with bus stops and bus and how routes are decided upon are all basic necessities to a public transport simulation. Implementing these actions also involved using multiple types of data structures. The two most commonly used data structures with relevance to these actions were:

- Directed Graph
- Linked List

Following the implementation of these actions and data structures, a small test was created to ensure the simulator was working accurately. This test composed of create a bus topology of nine bus stops and then populating that topology with passengers. Buses were then ran with both dynamic routing techniques and a fixed route. The interaction between elements on the system was successful following this test.
5 Evaluation

5.1 Introduction

This chapter discusses the tests that were run in order to obtain results regarding the efficiency differences between fixed routes and dynamic routes. As this project is based on routing buses, a map of real bus locations is necessary to acquire interpretable results. The area that was chosen to be mapped for the simulations was Dublin city centre. The reason for choosing this as the test area is due to the high density of bus stops within the city centre and the multitude of different routes between these bus stops.

![Figure 17: Mapped Area](image)

Figure 17: Mapped Area[9]

Figure 17(a) displays the map of the test area and the bus stops that will be used during the simulations. Figure 17(b) displays the fixed route that has been chosen through the test area. This fixed route is the real 48a route, along which 6 bus stops are served. The 23 bus stops that are highlighted on both maps are
the locations of real bus stops. It is assumed that the bus stops have been placed in high passenger density areas.

The rest of this chapter is broken up into three sections. Section 5.2 will discuss the 3 tests and give the estimated results of each test. Section 5.3 will discuss how each technique performed in the tests, giving the actual results obtained and discussing the over or under performance of each technique in comparison to the estimated results. Finally, Section 5.4 will review the 3 tests.

5.2 Tests to Simulate Passenger Traffic

Three tests were performed on each routing algorithm. Each test comprised of 210 passengers with random places of origin, destination and time of arrival to their origin. These tests were used to measure the amount of passengers picked up by each bus, the amount of time spent on the bus by each passenger and the amount of time spent at the bus stop by each passenger. Bus stops in Dublin city centre are regularly placed every 250 metres. This was used to calculate overall distances. For the purpose of these tests, time will be measured in seconds as the simulation progresses in seconds. During the simulation, the time taken to travel between 2 bus stops was 3 seconds. The threshold value for time spent on the bus per passenger with the dynamic routes was set at 15 seconds.

5.2.1 Estimated Results

Figure 17(b) shows the fixed route that was chosen can only serve 6 bus stops out of 23. This means that, from each test of randomly placed passengers, a bus using the fixed route would serve on average 55 passengers from the maximum 210 passengers.

As a bus takes 3 seconds to travel between bus stops the maximum time a passenger could be waiting for a bus is 15 seconds, passenger arrives as bus leaves. This then establishes and average waiting time of 7 seconds. As the journey time is set between bus stops, the time spent on the bus is proportional to the distance.
traveled on the bus. 3 seconds is the estimated minimum travel time spent on the bus, 15 seconds is the maximum time spent on the bus and 9 seconds is estimated at the average. Figure 22 shows the estimations tabulated.

<table>
<thead>
<tr>
<th>Passenger Pick Up</th>
<th>Distance Traveled</th>
<th>Time At Stop</th>
<th>Time On Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>250</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>105</td>
<td>750</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>210</td>
<td>1250</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 18: Fixed Route Estimations

The dynamic bus routes do not require a predetermined route. They are able to serve any of the 23 bus stops on the map. However, all roads on the map are one way. Therefore, the selection of a particular bus stop limits the next stops the bus can travel to.

It is possible for a passenger to request a journey that is longer than the threshold value of 15 seconds. This is due to the fact that certain routes take longer than 15 seconds to traverse. The estimated maximum journey time for a passenger within the given topology is 33 seconds, the longest single route is 11 bus stops, with a minimum journey time of 3 seconds. The average journey time is then calculated at 18 seconds.

The two types of algorithms implemented, Zero-Lookahead and One-Lookahead, have different limits of knowledge about the topology of bus stops and the passengers at each bus stop. With the limited knowledge it is possible for a passenger to be waiting at a bus stop for 3465 seconds(Half Number of Passengers * Longest Travel Time). The potential of this maximum value occurring is extremely low as it is estimated on the basis that ever passenger is randomly placed at the starting terminus before a bus has reached its last stop, starving other bus stops of service. The average pick up time is 1734 seconds. Figure 19 shows the estimations tabulated for 1 passenger, 105 passengers and 210 passengers.
<table>
<thead>
<tr>
<th>Passenger Pick Up</th>
<th>Distance Traveled</th>
<th>Time At Stop</th>
<th>Time On Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>105</td>
<td>1500</td>
<td>1734</td>
<td>18</td>
</tr>
<tr>
<td>210</td>
<td>2750</td>
<td>3465</td>
<td>33</td>
</tr>
</tbody>
</table>

Figure 19: Dynamic Route Estimations
5.3 Testing

Three tests of the simulation were performed using 210 random passengers for each test. The following sections will discuss the results from each test.

5.3.1 Test 1

![Histograms of Time Spent on a Bus](image)

Figure 20: Histograms of Time Spent on a Bus

Figure 20 shows the results of the time spent by passengers on a bus for the first test on all three routing techniques. In each histogram the Y-axis represents the number of passengers and the X-axis represents the time spent on the bus by the passengers from their origin to their destination.

Figure 20(Left) shows that the Fixed route served only 51 of the 210 randomly placed passengers. The majority of these passengers (28 passenger) spent over 10 seconds on the bus. The average time spent on the bus by the 51 passengers was 9 seconds and the maximum time spent by any passenger was 14 seconds. For these passengers to be served, the simulation calculated that 6 buses were required. This means that on average, 1 bus serves approximately 8 passengers.
Figure 20(Centre) shows that the Zero-Lookahead algorithm served 210 passengers. The majority of these passengers (140 passengers) are on a bus between 3-15 seconds. 28 passengers were on a bus for over 30 seconds. The maximum time spent on a bus was 44 seconds. The average time spent on a bus for the Zero-Lookahead technique was 17 seconds. The simulation calculated that 8 buses were required in order to serve all 210 passengers. The average passenger time on a bus has almost doubled however the number of passengers being served has quadrupled. This was achieved with only 2 additional buses. Using the Zero-Lookahead algorithm, each bus served on average 26 passengers.

Figure 20(Right) shows that the One-Lookahead served 210 passengers. The majority of passengers(155 passengers) spent between 3-15 seconds on a bus. The maximum time spent on a bus has increased to 55 seconds. The average time spent on a bus has decreased to 14 seconds because more passengers were on the bus for a shorter period. The number of buses calculated by the simulation was 10. This increase in bus numbers results in an average of 21 passengers per bus which is better than the fixed route, but not as good as the Zero-Lookahead.

The estimations for passenger time spent on a bus are shown in Figure 18 and Figure 19. The buses using a Fixed route performed almost as estimated. The buses using a Fixed route were able to collect 51 passengers (estimated 55 passengers) with an average journey time of 9 seconds per passenger (estimated journey time 9 seconds). The Zero-Lookahead algorithm returned an average passenger journey time of 17 seconds (estimated 18 seconds) and the One-Lookahead algorithm returned an average of 14 seconds (estimated 18 seconds). The maximum time spent on a bus by passengers using either dynamic routing technique (Zero-Lookahead 44 seconds)(One-Lookahead 55 seconds) was greater than the estimated 33 seconds. As the journey estimation time is not calculated from the starting terminus. It is possible for a bus to collect multiple passengers at the starting terminus and calculate a route to satisfy all passengers that could possibly result in the maximum passenger journey time being exceeded.
The graphs shown in Figure 21 plot the results of the time spent by passengers at a bus stop from the first test on all three routing techniques. In each graph the Y-axis represents the time spent at a bus stop and the X-axis represents the index for each passenger who entered the bus (The passengers are indexed from 1 to 210 as they enter the bus). The simulation running the fixed route did not serve as many passengers as either dynamic method.

The graph in Figure 21(Left) plots the waiting time of each individual passenger using the Fixed route. As estimated, no passenger waited longer than 15 seconds at a bus stop and the average waiting time was 6 seconds. The maximum waiting time for the Fixed route was 14 seconds. The average waiting time is less than the estimated average waiting time because 16 passengers arrived at their designated bus stops as the bus arrived as shown in the graph (Waited 0 seconds at their bus stops).

The graph in Figure 21(Centre) plots the waiting time of the passengers using the Zero-Lookahead routing technique. The first 52 passengers waited an average of 12 seconds. The waiting time increased substantially for, approximately, the next 93 passengers because they were waiting at bus stops located at a significantly
increased distance from the terminus. The remain passengers, approximately 65, have dramatic increases to their waiting times because their bus stops are located at outlying points furthest away from the terminus. In the last group, even though they were at outlying bus stops some passengers waiting times fell below the average. This can be explained by the fact that they arrived at bus stops as a bus was approaching. The maximum waiting time for the Zero-Lookahead is 188 seconds. The average waiting time is 43 seconds. The average estimation was 1734 seconds Figure 19.

The graph in Figure 21(Right) plots the waiting times of passengers with the One-Lookahead algorithm. The average waiting time is 75 seconds for the One-Lookahead algorithm. Similar to Zero-Lookahead algorithm the waiting time increased for passengers at bus stops located further from the terminus. In this test, the graph shows that the maximum passenger waiting time was 335 seconds.

The Fixed route performed better than expected. The average waiting time at a bus stop was 6 seconds against (estimated 7 seconds) and the longest waiting time was 14 seconds (estimated 15 seconds). Both dynamic routes also performed better than estimated. Zero-Lookahead had an average waiting time of 43 seconds (estimated 1734 seconds) and a maximum waiting time of 188 seconds (estimated 3465 seconds). One-Lookahead had an average waiting time of 75 seconds (estimated 1734 seconds) and a maximum of 335 seconds (estimated 3465 seconds). The Zero-Lookahead produced better results than the One-Lookahead for both the maximum and average passenger waiting times.
<table>
<thead>
<tr>
<th></th>
<th>Fixed Route</th>
<th>Zero-Lookahead</th>
<th>One-Lookahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time on Bus</td>
<td>9</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>Time at Bus Stop</td>
<td>6</td>
<td>43</td>
<td>75</td>
</tr>
<tr>
<td>Number of Buses</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Passengers Served</td>
<td>51</td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>

Figure 22: Test 1 Results

Test 1 has shown that both dynamic routes performed better than the Fixed route. For this test, the Zero-Lookahead technique has performed better than One-Lookahead technique. This is clear from an improved passenger to bus numbers and the average waiting time being lower. The One-Lookahead performs better with respect to the equity of passenger service.

5.3.2 Test 2

Figure 23: Histograms of Time Spent on a Bus

Figure 23 shows the results of the time spend by passengers on a bus for the second test on all three routing techniques. The Y-axis represents the number of passengers and the X-axis represents the time spent on a bus by the passengers getting
from their origin to their destination.

Figure 23(Left) shows that the Fixed route was able to serve 37 of the 210 passengers. The majority of these passengers (21 passengers) spent over 10 seconds on a bus. The average time of a passenger on a bus using this technique was 9 seconds and the maximum time spent on a bus was 14 seconds. As less passengers were served, the simulation calculated that only 5 buses were required. This means that on average, 1 bus serves approximately 7 passengers. However, this is not as good as the average bus service provide in Figure 20(Left) due to the fact that there are 14 less passengers along the Fixed route from Figure 20(Left) (51 passengers).

Figure 23(Centre) shows that the Zero-Lookahead algorithm served 210 passengers. The majority of passengers (approximately 164 passengers) were on the bus between 3 - 15 seconds. Approximately 7 passengers were on the bus for over 30 seconds. The maximum time spent on a bus was 38 seconds. The average time spent on a bus was 11 seconds and. The simulation calculated that the number of buses needed to serve all 210 passengers was 12. In comparison to the Fixed route, with just over double the number of buses it is possible to serve almost 6 times as many passengers while still keeping the average journey low at 11 seconds. This means that on average 1 bus can serve 17 passengers. This decrease in bus efficiency is due to the decrease in passengers being placed along a similar route (less passenger on the Fixed route). The average time has also decreased from Figure 20(Centre) (average journey time 17 seconds) due to the increased number of buses used.

Figure 23(Right) shows that the One-Lookahead algorithm served 210 passengers. The majority of passengers (approximately 165 passengers) were on the bus between 3 - 15 seconds. The maximum time spent on a bus has decreased to 25 seconds. The average time spent on a bus by any passenger was 12 seconds. The number of buses calculated by the simulation was 10. The decrease in bus numbers resulted in a larger average time spent on a bus. On average, 1 bus can serve
roughly 21 passengers. This algorithm is performing a better passenger to bus service than both the Fixed route and Zero-Lookahead. However, One-Lookahead has a larger average wait time than both methods.

The estimations for passenger time spend on a bus are shown in Figure 18 and Figure 19. The buses using a Fixed route performed worse than estimated. The buses using a Fixed route were able to collect 37 passengers (estimated 55 passengers) with an average journey time of 9 seconds (estimated journey time 9 seconds). Both dynamic routes returned better average journey times than estimated. The Zero-Lookahead technique returned an average passenger journey time of 11 seconds (estimated 17 seconds) and the One-Lookahead technique returned an average journey time of 12 seconds (estimated 17 seconds). The maximum time spent on a bus by a passenger using the Zero-Lookahead technique was 38 seconds (estimated 33 seconds) and using the One-Lookahead technique was 25 seconds (estimated 33 seconds). The reasons that the maximum journey time can be exceeded are stated in Test 1. The One-Lookahead technique required less buses compared to the Zero-Lookahead technique because this algorithm can see more passengers across the topology and having the ability to see more bus stops enabled each bus to serve more passengers.

Figure 24: Graph of Time Spent at a Bus Stop
The graphs shown in Figure 24 plot the results of the time spent at a bus stop for each routing technique in Test 2. The Y-axis represents the time spent at a bus stop and the X-axis represents the passengers (The passengers are indexed from 1 to 210 as they enter the bus).

The graph Figure 27 (Left) plots the waiting times for each individual passenger using the Fixed route. As estimated, no passenger waited longer than 15 seconds at a bus stop and the average time spent at a bus stop was 9 seconds. The average waiting time is larger than the average wait time in Figure 21 (Left) (6 seconds) due to the fact that less passengers (7 passengers) arrived at the bus stop as the bus arrived (16 passengers in Figure 21 (Left) arrived as the bus arrived).

The graph Figure 24 (Centre) plots the waiting times for each individual passenger using the Zero-Lookahead technique. The first 65 passengers waiting average time was 10 seconds. The waiting time increased substantially for, approximately, the next 110 passengers because they were waiting at bus stops located at a significantly increased distance from the terminus. The remain passengers, approximately 75, have dramatic increases to their waiting times because their bus stops are located at outlying points furthest away from the terminus. In the last group, even though they were at outlying bus stops some passengers waiting times fell below the average. This can be explained by the fact that they arrived at bus stops as a bus was approaching. The maximum waiting time for the Zero-Lookahead is 295 seconds. The average waiting time is 43 seconds. The average estimation was 1734 seconds.

The graph in Figure 24 (Right) plots the waiting times of passengers with the One-Lookahead algorithm. The average waiting time is 59 seconds for the One-Lookahead algorithm. Similar to Zero-Lookahead algorithm, the waiting time increased for passengers at bus stops located further from the terminus. In this test, the graph shows that the maximum passenger waiting time was 304 seconds.

The Fixed route performed worse than expected. The average waiting time at
a bus stop was 9 seconds (estimated 7 seconds) and the longest waiting time was 15 seconds (estimated 15 seconds). Both dynamic routes performed better than estimated. Zero-Lookahead had an average waiting time of 43 seconds (estimated 1734 seconds) and a maximum waiting time of 295 seconds (estimated 3465 seconds). One-Lookahead had an average waiting time of 59 seconds (estimated 1734 seconds) and a maximum of 304 seconds (estimated 3465 seconds).

<table>
<thead>
<tr>
<th></th>
<th>Fixed Route</th>
<th>Zero-Lookahead</th>
<th>One-Lookahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time on Bus</td>
<td>9</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Time at Bus Stop</td>
<td>9</td>
<td>43</td>
<td>59</td>
</tr>
<tr>
<td>Number of Buses</td>
<td>5</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>Passengers Served</td>
<td>37</td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>

Figure 25: Test 2 Results

Test 2 has shown that both dynamic routing techniques performed better than both the Fixed route. However, both dynamic routes performed differently. From an economic point of view, One-Lookahead is better as less buses are needed to serve all 210 passengers. From a passengers point of view, using the Zero-Lookahead, passengers have lower waiting time and lower time spent on a bus.
5.3.3 Test 3

Figure 26 shows the results of the time spent by passengers on a bus for Test 3 on all three routing techniques. The Y-axis represents the number of passengers and the X-axis represents the time spent on the bus by the passengers from their origin to their destination.

Figure 26(Left) shows that the Fixed route served only 32 of the 210 randomly placed passengers. The average time spent on the bus by these passengers was 9 seconds and the maximum time spent by any passenger was 14 seconds. For these passengers to be served, the simulation calculated that 5 buses were required. This means that on average, 1 bus serves approximately 6 passengers. This is a lower average from previous tests.

Figure 26(Centre) shows that the Zero-Lookahead technique served 210 passengers. The majority of these passengers (160 passengers) were on the bus for between 3 - 15 seconds. Approximately 7 passengers were on the bus for over 30 seconds. The average time spent on the bus was 13 seconds and the maximum time was 38 seconds. The average time spent on the bus has increased from Test 2
Figure 23(Centre). The simulation calculated that 10 buses were required to serve all 210 passengers. For this test, in comparison to the Fixed Route, double the number of buses can serve over 6 1/2 times the number of passengers. Using this technique, each bus serves on average 21 passengers.

Figure 26(Right) shows that the One-Lookahead technique served 210 passengers. The majority of passengers (156 passengers) were on a bus for between 3 - 15 seconds. No passengers exceeded the limit of 33 seconds. The average time for passengers using this technique has decreased to 11 seconds and the maximum time spent by any passenger has again decreased to 23 seconds. The simulation calculated that 11 buses were required to serve the 210 passengers. Each bus then serves on average 19 passengers. The extra bus has resulted in an improved average time spent on a bus, but a reduced passenger to bus average.

The estimations for a passenger time spent on a bus are shown in Figure 18 and Figure 19. The bus using a Fixed Route perform worse then estimated. The bus using a Fixed route were able to collect 32 passengers (estimated 55 passengers) with an average time of 9 seconds (estimated 9 seconds). Both dynamic routing techniques had better average times spent on a bus than estimated. The Zero-Lookahead algorithm returned an average passenger journey time of 13 (estimated 18 seconds) and the One-Lookahead algorithm returned an average of 11 seconds (estimated 18 seconds). The maximum time spent on a bus by a passenger using the Zero-Lookahead technique was 38 seconds (estimated 33 seconds) and using the One-Lookahead technique was 23 seconds (estimated 33 seconds). The reasons that the maximum journey time can be exceeded are stated in Test 1. The One-Lookahead technique had an improved average with an added bus from both previous tests.
The graphs shown in Figure 27 plot the results of the time spent at a bus stop for the 3 routing techniques. The Y-axis represents the time spent at a bus stop and the X-axis represents the order in which the passengers enter the bus (The passengers are indexed from 1 to 210 as they enter the bus).

The graph Figure 27(Left) plots the waiting times for each individual passenger using the Fixed route. As estimated, no passenger waited longer than 15 seconds at a bus stop. The average passenger waiting time is 5 seconds. The average waiting time is smaller than the both previous tests because 20 passengers arrived at bus stops with a bus arriving within 4 second.

The graph Figure 27(Centre) plots the waiting times for the passengers using the Zero-Lookahead technique. The first 52 passengers waited an average of 10 seconds. The waiting time increased substantially for, approximately, the next 90 passengers because they were waiting at bus stops located at a significantly increased distance from the terminus. The remaining passengers, approximately 68, have dramatic increases to their waiting times because their bus stops are located at outlying points furthest away from the terminus. In the last group, even though they were at outlying bus stops some passengers waiting times fell below
the average. This can be explained by the fact that they arrived at bus stops as a bus was approaching. The maximum waiting time for the Zero-Lookahead is 246 seconds. The average waiting time is 43 seconds. The average estimation was 1734 seconds.

The graph Figure 27(Right) plots the graph for the wait times of passengers with the One-Lookahead algorithm. The 65 passengers have an average pick up time of 10 seconds. The average passenger waiting times is 49 seconds. Similar to Zero-Lookahead algorithm, the waiting time increased for passengers at bus stops located further from the terminus. The maximum waiting time of any passenger was 261 seconds.

The Fixed route performed as estimated. The average waiting time at a bus stop was 5 seconds against (estimated 7 seconds) and the longest waiting time was 15 seconds (estimated 15 seconds). The average waiting time was reduced because passengers arrived at bus stops closer to the buses time of arrival. Both dynamic routes performed better than estimated. Zero-Lookahead had an average waiting time of 43 seconds (estimated 1734 seconds) and a maximum waiting time of 246 seconds (estimated 3465 seconds). One-Lookahead had an average waiting time of 49 seconds (estimated 1734 seconds) and a maximum of 261 seconds (estimated 3465 seconds).

<table>
<thead>
<tr>
<th></th>
<th>Test 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed Route</td>
</tr>
<tr>
<td>Time on Bus</td>
<td>9</td>
</tr>
<tr>
<td>Time at Bus Stop</td>
<td>5</td>
</tr>
<tr>
<td>Number of Buses</td>
<td>5</td>
</tr>
<tr>
<td>Passengers Served</td>
<td>32</td>
</tr>
</tbody>
</table>

Figure 28: Test 3 Results

Test 3 has shown that for the One-Lookahead algorithm, with an extra bus added to its fleet, it is possible to again achieve better average journey times with a lower
maximum journey time. However, Zero-Lookahead might be a better option from an economic point of view as less buses are required with minimum change to the average passenger journey time. Again both dynamic routing methods perform better than the fixed route.
5.4 Review

The chapter discussed the three test that were run. Each test contained 210 passenger with randomly picked points of origin, destination and time of arrival to their point of origin. Certain estimations were subsequently made before the test were run based.

- Time on Bus
- Passenger Pick up Numbers
- Time Spent At a Bus Stop

The tests were then carried out and within each test there was 3 sets of results, one for each of the algorithms performed. Each test was then discussed and analyzed. In Test 1 it was shown that the Zero-Lookahead performed only slightly better than either of the other two methods due to its reduced bus requirements and better ratio of passengers collected to buses required. In Test 2, both dynamic routes were shown to perform on a better level than the fixed route. However, determining which of the dynamic routes was more efficient was not as clear. Both routes performed on differently with respect to different values. The Zero-Lookahead was found to be more better from an economical point of view, and the One-Lookahead was better from a passenger point of view. Test 3 returned similar results too Test 2 in the way the dynamic routes performed against each other.

5.4.1 Observations

As the number of passengers on the Fixed bus route has decreased, the overall performance of the One-Lookahead algorithm has increased. This is because the One-Lookahead technique has a slightly wider knowledge of the topology and as passengers are seen to be placed at more spread out locations( Less passenger being placed along the Fixed Route), the One-Lookahead algorithm is able to create a route suitable for more passengers. Using the One-Lookahead technique, passengers being on the bus for longer than the estimated maximum time significantly decreased with respect to the Zero-Lookahead algorithm. The One-Lookahead
works better for passengers who are evenly dispersed out across a topology.

The Zero-Lookahead’s average wait time did not change throughout each test because the bus did not know where passengers were placed along the topology. It could only make decision based on the immediate bus stops passengers, and the passengers on the bus itself.

For both dynamic routing techniques, an increase of bus numbers improved the overall average time spent on a bus.

As both dynamic routes have only a limited range of knowledge about the passenger density, there is potential for bus stop starvation for outlying bus stops. Wider range of knowledge of the bus stops would overcome this issue.
6 Future Work

As the tests that were discussed in chapter 5 were based on randomly generated passenger information, performing further tests on real passenger data would return results that show the effectiveness of these algorithms in real situations.

Following a discussion with Dr. Rene Mieir from the Distributed Systems Group in Trinity College Dublin, it was decided that the following additions would be of benefit for gathering further results:

- Further tests in different environments
- Increased number of fixed routes
- Increased number of passengers
- Actual passenger data
- Introduction of traffic congestion into the simulator
- Implement algorithms into existing simulators
- Introduction of peak hour testing

Presenting this study to transport organizations will allow them to include these results in any feasibility testing that they might run.

Implementing these algorithms in real buses could also increase the inevitability of having driverless buses. Combined with the advances made with the Google car, these algorithms could make for a completely automated transport service.

Algorithms[26][22][25] that have been designed with an increased overall knowledge appear to be more efficient for passengers. Improvements to the new algorithms, used in this project, by increasing the information provided(Two-Lookahead, Three-Lookahead) will further increase passenger efficiency and keeps bus topology knowledge low.
7 Conclusion

Transport is the cause for 25 percent of the worlds carbon emissions\cite{11} so improving the efficiency of public transport systems is an important issue. This project was undertaken because of the perceived inefficiencies of transport systems that use fixed routes. The overall objective of this project was to design new routing algorithms (Zero-Lookahead, One-Lookahead) to compare against the current routing system (Fixed Routes). Previous systems, discussed in section 2, have used algorithms that incorporate overall knowledge of both the topology of bus stops and their passengers. This project designed 2 new algorithms that routed buses with reduced information supplied to the buses. The idea in reducing information was to reduce transmission costs. The two new algorithms used in this project achieved successful results. A transport system simulator was also successfully created in order to compare methods of routing.

It is clear from the results of the tests that buses operating with either dynamic routing technique can serve on average 4 times as many people with only minimal increases in bus numbers compared to buses operating on a fixed route. The tests results also show that both dynamic routing techniques are more efficient for passengers than fixed routes. The aims of the project were completed with success and future work possibilities are shown in section 6.
8 Appendix A

Please see code package provided on attached CD.
References


72


