University of Southampton
Faculty of Engineering and Applied Science
Department of Civil and Environmental Engineering

SIMULATING ADVANCED BUS PRIORITY STRATEGIES AT TRAFFIC SIGNALS

By
Birendra Prasad Shrestha

March 2003

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy.

This thesis is dedicated to my parents.

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

Doctor of Philosophy

SIMULATING ADVANCED BUS PRIORITY STRATEGIES AT TRAFFIC SIGNALS

by Birendra Prasad Shrestha

Providing priority to buses plays an important role in protecting bus services from the effects of traffic congestion and in improving speeds and reliability. Among the measures available, bus priority at traffic signals is the most relevant where opportunities for segregated systems are not available and/or where numerous traffic signals exist. Again, there is scope for many forms of bus priority at signalised junctions. One such method is differential bus priority in which buses are given different levels of priority according to their individual requirement. This method allows a range of priority strategies to be implemented, depending upon the lateness of buses and junction capacity constraints. Exploring the performance of different priority strategies within this method has been the main aim of this research.

The literature review showed that there is a need for a model for modelling differential priority in detail. Hence, a microscopic simulation model SIMBOL (Simulation Model for Bus priority at traffic signal) was developed. SIMBOL simulates a bus route in a field situation, taking account of the characteristics of buses, bus stops, traffic signals, the AVL system and differential bus priority systems.

The model was calibrated and validated with the field data from a bus route in Southampton. The validated model was then used to formulate 7 different types of priority strategies and to simulate them under various scenarios. These strategies varied in terms of the level of priority to be provided, based on the lateness of the buses. The different scenarios in which these strategies were simulated included: the present field condition, different types of bus generation, holding early buses, potential errors in location systems and changes in bus operations. The performance of these priority strategies was evaluated using the output results from the model.

The simulation results showed that all bus priority strategies simulated give benefit to a bus system including buses and passengers. However, the type of benefits and their magnitude differ from one strategy to another. The results illustrated the strength and weaknesses of different priority strategies under different field conditions. This showed that the selection of a best-suited priority strategy depends upon the aim of a priority scheme in terms of total benefit or punctuality. Research included the development, modelling and recommendation of a new mixed priority strategy giving good benefits across a range of scenarios. The research also demonstrated the usefulness of SIMBOL as a valuable tool for modelling differential bus priority under different field conditions and as a basis for further research.

Content

List of tables

List of figures

Acknowledgement

I would like to thank the following individuals whose kind guidance, help and support enabled me to start, continue and complete this thesis.

I would like to express my sincere gratitude to my supervisor, Dr. N.B. Hounsell, whose constant guidance and encouragement throughout the study period made it all possible complete this research.

I would like to thank TRG staff, Dr. T. Cherrett for his help in SCOOT data collection. I would also like to thank my friend, Dr. Bidur Rajbhandari and family for their support during this study. I would also like to thank my friends Debra Grunnham and Graham Wall for their help in language correction.

Finally, I would like to thank my wife Lata, whose encouragement and support made it all possible to complete this work. I would also like to thank my son Binay for his kind cooperation during thesis writing period beside his unlimited queries about my research and his interests.

Abbreviations used

The abbreviations used throughout this thesis are as follows

AVL	Automatic Vehicle Location
DoS	Degree of Saturation
SCOOT	Split Cycle Offset Optimisation Techniques
SPRINT	Selective PRIority Network Techniques
SVD	Selective Vehicle Detection
UTC	Urban Traffic Control

Chapter One

Introduction

1.1 Background

In recent years, there have been growing concerns about optimising the existing road-network rather than expanding it. In this regard, public transport having a very high passenger carrying capacity can help to optimise the use of the existing network. This is receiving much more political attention as well as research and development emphasis. Cities like Southampton now have transport policies with greater emphasis on running attractive and effective public transport (SCC, 1999). To fulfil this goal, public transport should have at least some better quality than private vehicles. King (1992) recognises that in order to achieve the desired major increase in bus use, a high quality bus system with, at least, some key components is needed. These components are fast, reliable, convenient, comfortable, reasonably priced and a good information system. In the past, there has been too much attention paid to the movement of vehicles rather than the movement of people and goods (King, 1992). This still remains to some extent and public transport especially buses are now suffering from problems such as congestion, low operating speed, bunching and irregularity.

Buses are the predominant form of public transport in most towns and cities in many countries, including the U.K (Hounsell and McLeod, 1999). With their large carrying capacity, buses make effective use of limited road space, and can therefore make a substantial contribution to reducing traffic congestion (Cheney, 1992). However, buses themselves are often affected by congestion, leading towards a decrease in speed and an increase in bus travel time variability and service irregularity. In extreme cases bus 'bunching' may occur, where buses form groups of two or more vehicles. This makes the timetable of the bus operation very uncertain which can degrade the faith of passengers in the bus service.

Providing priority to buses plays an important role to protect bus services from the effects of traffic congestion and to improve route frequencies, speeds and reliability (IHT, 1997). These measures of providing priority to buses are getting more and more attention in these recent years. The U.K. government also supports bus priority schemes and hence these priority schemes are eligible for Department of Transport /Government Office funding, as part of bids for Supplementary Credit Approval (IHT, 1997). 'Keeping Buses Moving' (DETR, 1997) outlines that the potential impact of new traffic management measures on bus services should always be considered at an early stage in the planning of any scheme, whether its main intention is to improve bus services or not.

Priorities currently used for bus operations include segregated systems such as with-flow and contra-flow bus lanes, banned turn exemption provision like bus only turn (bus gates), improved design of bus bays, stops and priority at traffic signals (DETR, 1997). The choice of bus priority measures to implement varies according to the situation. When considering the congested urban situation with many signalised junctions and limited roadspace, bus priority at signals may be the most effective option to consider. There are a number of ways in which buses can be given priority at traffic signals. Among them, there are a number of examples where priority is given to all approaching buses based on selective vehicle detection (SVD). In recent years, interest has grown in using differential bus priority, which can give a varying level of priority to the buses according to their need (e.g. their adherence to schedule). This system not only helps to regularise the bus operation but also minimises the negative impacts to other traffic, while giving priority (because fewer buses are given priority). These systems typically rely on Automatic Vehicle Location system (AVL). This research is concerned specifically with the evaluation of differential priority to the buses using AVL in terms of potential impacts, control strategies and benefits. The objectives of this research work are outlined below.

1.2 Objectives

The main objective of this research is to explore ways of providing selective priority to buses at traffic signals using AVL. The general objectives of the research are given below.

- To review the existing state of art of bus priority at traffic signals.
- To explore the requirements of advanced bus priority strategies at traffic signals.
- To develop a simulation model capable of modelling the details of different bus priority strategies incorporating AVL.
- To apply the simulation model to a range of operating scenarios, to evaluate the performance of different bus priority strategies

The methodology to be followed to achieve these objectives is discussed in the next section.

1.3 Methodology

As already discussed in the background of the research, a statement of the problem is the first step to be started. At this stage, the necessity of the research to be carried out in the field of differential priority to the buses at traffic signals is defined and the objectives of the research are set out. This is followed by a review of literature available in the field of bus priority including differential priority and use of automatic vehicle location (AVL) systems. A review of commercially available relevant computer software is carried out to identify potentially useful packages in this research given the modelling requirements specified. Once the review is completed, the proposed modelling methodology is the next to be developed. This includes the finalisation of modelling components and techniques to model them. This will be followed by the collection of data from the field to make a base case model. The developed base case model is then calibrated to simulate the field situation. The model is then modified to incorporate differential priority strategies to give priority to buses. The model is then used to simulate different bus priority strategies under a range of scenarios. The results obtained from the model application are then analysed to evaluate the performance of the strategies and to draw conclusions. The sequence of the research process is illustrated in Figure 1.1.

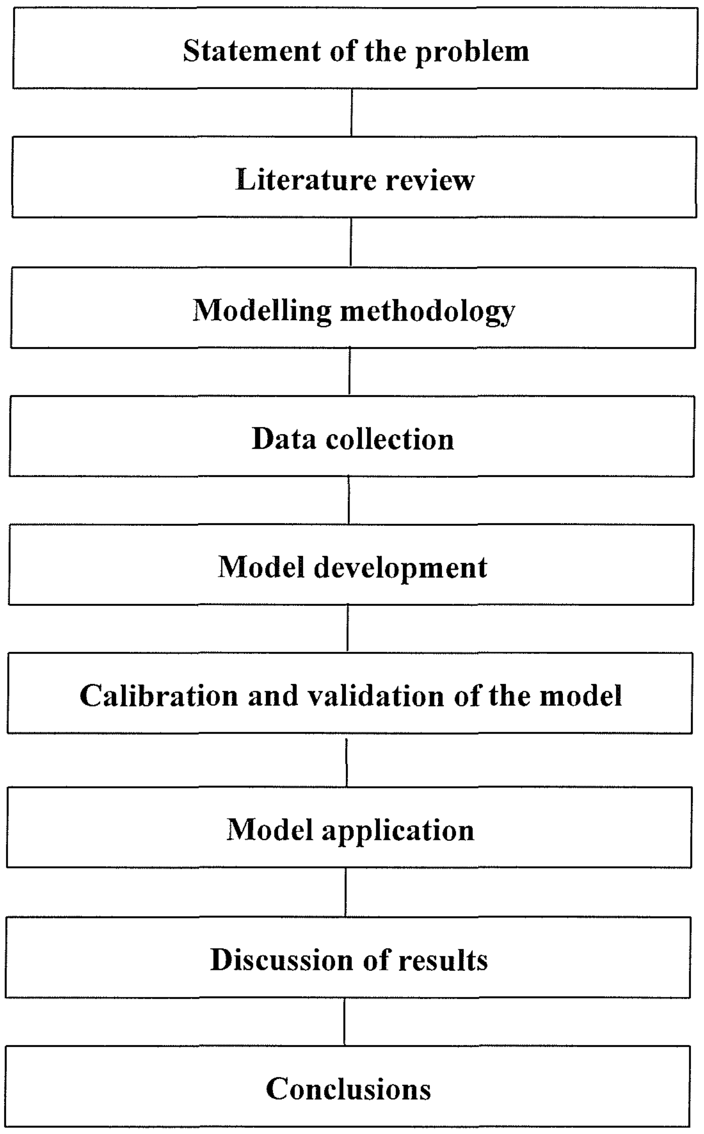


Figure 1.1 Outline of the research

Chapter Two

Literature Review

2.1 Bus priority

Giving priority to buses is an important measure to protect bus services from the effects of traffic congestion and to improve their speed and reliability. These measures vary in scale and impact from a simple exemption from a manoeuvre prohibited to other traffic, to area-wide measures such as priority in traffic control systems (IHT, 1987). The appropriateness of a measure may depend upon the aim and objectives of the scheme. There may be several aims and objectives of providing priority measures to the bus. One of them may be to reduce delays to buses arising from traffic congestion and thus save bus operating costs, passengers' travel-time costs and bus-fleet requirement. Another may be to increase accessibility to major traffic generators, like shopping centres and inter-modal transport exchanges (IHT, 1997). But a key factor is to improve the reliability and regularity of bus services which makes travel by bus more attractive and could increase bus patronage (IHT, 1987). Oakes et. al. (1994) also state that the overall objectives of bus priority can be encompassed by a reduction in bus journey time and improved reliability.

There are many different ways of giving priority to buses. Some of them are link-based and others are junction-based. These measures are used according to the need and the feasibility to provide them. In most of the cases buses are given priority by specifically re-allocating existing highway capacity at junctions and on links to them (Holmann and Willumsen, 1991). Some of the bus priority measures detailed in 'Keeping buses moving' (DETR, 1997) include with-flow lanes, contra-flow bus lanes, bus only streets, bus only turns (bus gates) and bus priority at signalised junctions

With-flow lanes, contra-flow lanes, bus only streets, busways, parking control along bus routes and improved design of bus stops are examples of linked-based priority measures. Bus only turns (bus gates) and bus priority at signals are examples of junction based priority measures. With-flow bus lanes are the most common form of bus priority measure. Keeping Buses Moving (DETR, 1997), describes with-flow lanes as a reserved traffic lane, usually on the nearside, for the use of buses and may accommodate bicycles. A with-flow bus lane enables buses to bypass traffic queues, usually approaching traffic signals. This will often mean substantial time savings to buses and their passengers, offset by some additional delay to vehicles which have been overtaken.

A contra-flow bus lane is a lane where buses are allowed to travel against the main direction of traffic flow in a one-way street. This operation enables buses to avoid unnecessary diversions, to maintain route patterns when new one-way streets are introduced, and to gain better access to business and shopping areas (DETR, 1997). These contra-flow bus lanes are usually introduced in area-wide one way traffic systems, where the effect is to create a two-way road with 'buses only' allowed in one direction, and all types of vehicle including buses, in the other (IHT, 1997).

A bus-only street is a section of road for the use of buses only. It may be a section of road enabling buses to take a more direct route or a "pedestrianised" street in a town centre where buses are exempt from a prohibition on other vehicles (DETR, 1997). Such a street enables buses to maintain route patterns in areas where traffic flow patterns have been changed and to gain close access to business and shopping areas where it is denied to other vehicles (IHT, 1997). This close access to the main attractions can be advantageous in encouraging the use of buses.

Busways are substantial corridors or networks of bus-only sections of road constructed specifically for the exclusive use of buses. Busways are designed to segregate buses from general traffic that protect them from congestion. For reasons of economy and land requirement, automatically guided or tracked busways may be preferred over busways ralying on manual steering (IHT, 1987).

Another form of urban bus priority measure is bus priority at junctions. A simple type of bus priority measure may be giving turning exemption at a junction, which allows buses to make a turn that is prohibited to other traffic. This is an inexpensive measure, which can give a considerable advantage to buses by allowing them to take a shorter route than other traffic. The exemptions can be used to enter into a contra-flow bus lane or a bus only street. In signalised junctions, the segregated type of priority such as with-flow bus lanes may be combined with the signalling measures. Queue relocation (also known as traffic metering) is one such measure in which the flow of traffic is controlled at upstream junctions by adjusting signal to reduce capacity so that this junction becomes more critical than the downstream (IHT, 1997). The downstream junction is the main junction whereas the upstream junction is the metered junction. Along with this, the bus lane running up to the upstream stopline enables buses to by-pass the relocated traffic queue. Other different types of signalling measures to give priority at signalised junctions are discussed in subsequent sub-sections.

2.1.1 Bus priority at traffic signals

In signalised junctions, there is scope for many forms of bus priority options (Hounsell, 1995). The most straightforward form of priority at traffic signals is to bias signal timings so that the approach having higher bus flow gets more effective green time than it would have done otherwise. The other approaches then share the remaining part of the cycle time. This is sometimes called passive bus priority because no equipment is required on the buses to 'actively' request priority. This form of priority could be implemented by optimising signal timings to minimise delay to people at the junction, rather than vehicles, as is traditionally undertaken. There is an argument that 'minimum person delay' should, anyway, be a preferred criterion for signal optimisation. In practice, passive priority often has to be implemented through urban traffic control (UTC) systems, and options available in the UK include:

a) BusTRANSYT (Robertson and Vincent, 1975), where optimum signal offsets in fixed time systems are calculated to reflect the different bus occupancy and performance in signalised networks relative to other traffic;

b) Facilities in SCOOT (Split Cycle and Offset Optimisation Technique) (Hunt et. al., 1981) such as 'split and offset weighting' (Hounsell and McLeod, 1999) where preference can be given to specific links in the network (e.g. those containing high bus flows).

The benefits of passive priority are generally limited and not related to the needs of individual buses. This can be resolved by providing 'active' priority. This means that bus priority is triggered only after detecting a bus in the traffic stream. This is achieved by making the traffic signal responsive to the arrival of the bus by using some sort of detection technique available. Active bus priority at traffic signals was first trialled at isolated junctions in London in the 1970s before the large-scale trial in the outer London bus district of SELKENT (South East London and Kent) in 1987 (University of Southampton, 1987). This large-scale trial in SELKENT covered 56 signalled non-UTC junctions and included the fitting of some 900 London buses with transponders to activate priority (Hounsell and Landles, 1995).

The success of the SELKENT scheme in London led to the system being extended to cover 300 outer London junctions and the fitting of all buses (about 4500) with transponders. Then it was followed by the development of bus priority systems using selective vehicle detection (SVD) at fixed time and traffic responsive urban traffic control (UTC) systems. The PROMPT (Priority and Informatics in Public Transport) (Hounsell et al, 1996) project developed bus priority at traffic signals under traffic responsive SCOOT UTC system and SPRINT (Selective Priority Network Technique) (Hounsell et al, 1997) was developed for those under fixed time UTC in London. More about this system of giving bus priority with selective vehicle detection is described in the next sub-section.

2.1.2 Bus priority with selective vehicle detection

In this method, buses are given priority at traffic signals by utilising selective vehicle detection (SVD) techniques and hence it is commonly known as bus priority with SVD. In this system, buses fitted with some form of electronic device are detected by different techniques before reaching the junction. Alternatively, some advanced detection techniques can detect buses in mixed traffic streams without needing on-bus equipment. Once a bus is detected, a priority is given by altering the traffic signal timing in its favour. Priority to buses may be given by extending the green time until the clearance of the bus or by recalling the next green time to give green to the bus more quickly. These methods of extension and recall are described and illustrated in Section 2.2.

Bus priority with selective vehicle detection is an effective system in giving priority to all detected buses. This system gives a good base for bus priority by minimising the bus delay at traffic signals and hence reducing the journey time of the buses and journey time variability. However, in some cases, there may be greater disruption to other traffic without giving a real benefit to the buses. For example, buses running early or on time may not need priority from a punctuality or regularity consideration. 'Punctuality' (i.e. a measure of buses departing on time) is the term used to measure operational performance in timetabled services. 'Regularity' (i.e. a measure of buses departing in regular interval) is the term used in headway-based service. 'Punctuality' and 'regularity' are important measures for the evaluation of public transport operations (Rudnicki, 1997). These measures directly affect the waiting time of passengers. The relation between 'punctuality/regularity' and passenger waiting times is described as follows according to the type of bus operation (i.e. timetable-based or headway-based).

In timetabled services, buses are scheduled to arrive at a stop at predefined times. In this service, generally the frequency of buses is quite low (typically less than 5 buses per hour) and the passengers arrive at bus stop according to the time of bus arrival. In this case, passengers arriving on time miss buses running ahead of timetable. They have to wait for the next bus coming and hence there will be a considerable increase in waiting time. On other hand, late buses will increase the waiting time of all passengers at that stop. Hence, the punctual bus gives lower average passenger waiting time at the bus stop.

In headway-based services, buses are operated under a predefined headway (gap) between them. The gap between buses is given rather than the departure time of the next bus. This is generally a high frequency service (typically more than 5 buses per hour). Since passengers do not know the departure time of next bus, they arrive randomly at bus stops. In this case, the buses running with lesser headway may cause a bigger headway to the bus behind. The passengers who have just missed the bus have to wait for a longer duration than the scheduled headway, increasing the average passenger waiting time at bus stops. McLeod (1999) has shown that the expected passenger waiting time is minimum when buses are regular. Hence, the regular bus service gives lower passenger waiting time at bus stops.

A SVD priority system providing the same level of priority to all buses may not often be that effective in maintaining service punctuality. Hence, although the system may be effective in reducing the journey time of buses, it may not be that effective in reducing passenger waiting time at bus stops. One of the options for making buses more punctual and regular is by giving them priority according to their need to maintain their punctuality or regularity. In recent years, concern has grown to refine systems to give priority to the buses only if they need it according to prescribed criteria (e.g. lateness). This can be termed as 'differential priority' in which buses are given different levels of priority according to their need for the priority. This system of giving priority is explained in more detail in next sub-section.

2.1.3 Differential bus priority at traffic signals

In this system of differential priority to buses, priority is given to the buses according to their individual requirement and or fulfilling the other criteria imposed. Here, once the bus is detected, it is checked with pre-defined criteria to identify whether the bus is eligible for priority or not. The criteria can depend upon site-specific requirements and may differ from place to place. The criteria may be based on the lateness of a bus and the situation of the junction only or a combination of several other factors such as delay to other traffic and passengers.

Where the criterion is based on the time gap between the buses, known as headway, priority may be termed 'headway-based differential bus priority'. With this system, higher levels of priority can be given to buses with headways higher than scheduled and this should improve bus service regularity (McLeod, 1998). A study undertaken by Transportation Research Group for London Transport Buses (University of Southampton, 1996) has shown that selecting buses for priority according to their headway (relative to the scheduled/average frequency) could have the following benefits (Hounsell et. al., 1997).

- Improved service regularity
- Targeting buses with a higher than average occupancy
- The provision of a higher level of priority without significantly affecting general traffic
- Less disruption to general traffic

Headway-based differential priority criteria depend upon the lateness of the bus and the situation of the junction. If the bus is lagging behind by more than the threshold defined, then it may be eligible for priority. The bus that has higher lagging may get the higher priority. But this may be constrained by the availability of spare capacity in the junction. The higher the saturation at a junction, the lower the chance of getting priority, at least in the current system operated in the UK. Combining these constraints, a priority algorithm developed by Hounsell and McLeod (1999) was used to investigate the effect of differential priority. The algorithm was incorporated into a simulation model SPLIT (Selective priority for Late buses Implemented at Traffic signals) to assess the impact of differential priority on passenger waiting and travel times to improve bus service regularity.

The algorithm for defining the priority level requested by each bus is based on the adherence to its actual headway relative to the scheduled headway. It provides different levels of priority according to the size of the bus headway relative to the expected average headway. Two different factors, the ratio (R) and the difference (D) between actual headway and scheduled headway, were considered as measures of 'lateness' of a bus. The level of priority assigned to a bus would then be determined by comparing its lateness with pre-set lateness threshold. The values taken by lateness threshold and the definition of the different priority levels are site specific. By modifying the values of these threshold ratios and meaning of priority levels, it is possible to 'fine-tune' the priority system to improve results at the site.

Another algorithm developed by Chang and Su (1995) is based on a performance function that depends upon the current queue length, bus loading factors and bus schedule delay. In this model, the benefit of giving a green signal to a bus is compared with that of terminating it by computing the tradeoffs incurred in passenger, vehicle and schedule delays. This model needs to compute a performance index, PI, that evaluates the effect of bus priority. The performance index is calculated by summing up passenger delay, vehicle delay and schedule delay. In a multi-phase control intersection, the PI value is the sum of PIs for each competing phase. When PI is non-negative, then only the bus priority is given by extending current green time for T seconds.

One of the criteria needed for schedule-based services is based on the deviation of buses from their scheduled timetable. With this system, buses running behind their timetable may be made eligible for priority. The bus that has higher lagging may get the higher priority and that may be constrained by the availability of spare capacity in the junction. Such priority may help them to improve their adherence to the timetable and to reduce passenger waiting time. The method of implementing awarded bus priority at traffic signals is described in the next section.

2.2 Implementation of bus priority at traffic signals

Under current UK practice, priority to buses is implemented mainly in two ways (McLeod, 1999). One is 'Extensions' where the green time is extended until the clearance of the bus, if it is expected to otherwise just miss the end of the present green period. The second is 'Recalls' where the green time is recalled to give green to the bus more quickly once a bus is expected to arrive at red period. Khasnabis and Rudraraju (1997) have categorised one more strategy as red interruption. In this red interruption, a short green phase, not contiguous with the adjacent green, is injected within the red phase along the bus route. Other options using special phases for buses are also used in some cases. The method of implementing bus priority using 'extension' and 'recall' is illustrated in a distance time diagram in Figure 2.1.

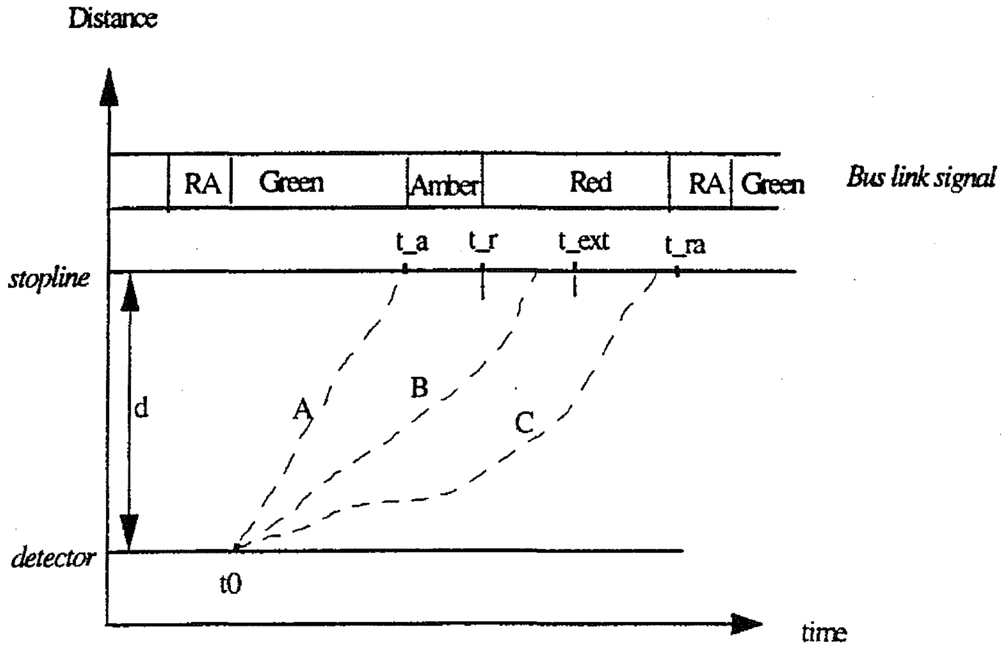


Figure 2.1: Space time representation of bus priority (Liu et al, 1999)

In the diagram, the signal setting for the bus link is shown on the top, with t_r and t_ra representing the start and end time of the red aspect. The end of extension time allowed, t_ext = t_r + E_max, where E_max is a user specified maximum allowed extension time. The distance from the detector to the stop line is d. Three trajectories from the detector to the stop line are drawn as dashed lines to represent probable bus arrival time at the stop line. If a bus is predicted to arrive at the stop line at a green signal, then the signal timings will not be changed. If it is expected to arrive just after the start of the red signal (case B), then the bus green time will be extended to allow the bus to exit. The extension is just sufficient for the bus to exit and may be less than the maximum defined. And if a bus is predicted to arrive during the red (case C), then the duration of the red aspect may be reduced by a constant amount. This is the case of recalls.

McLeod (1999) states that extensions generate higher benefit to individual buses than recalls. This is because the delay associated with buses which are eligible for an extension but 'just miss' the green light is greater than that for buses which are eligible for a recall which could arrive at a traffic signal at any time during the red period. In addition, extensions cause less disruption to other traffic than recalls since there is less interference with traffic signal settings. However, the proportion of buses gaining an extension is often much lower than those gaining recalls, so that the net benefit to all buses is often greater from recalls.

For the implementation of priority, a traffic signal controller should know about the arrival time of a bus at the traffic signal. This arrival time of a bus is estimated by detecting a bus approaching at the traffic signal. This is then communicated to the control centre where a priority requirement is assessed and the bus priority is awarded. The two main operational aspects such as detection system and communication architecture are described in the subsequent sub-sections.

2.2.1 Detection techniques used in bus priority

The use of detection techniques for bus priority at traffic signals started in the form of transponders and inductive loops detecting buses on the signal approach (Hounsell and McLeod, 1999). But the disadvantages of this system arise from the vulnerability of the loop to damage and the short life of the bus transponder (Hounsell and Landles, 1995). Hence, in recent years, the use of AVL systems such as beacon-based and GPS-based is increasing. The increasing use of AVL systems has utilised a number of different AVL techniques. These available AVL techniques used in bus priority at traffic signals are described in next sub-section.

2.2.2.1 Automatic vehicle location techniques

With the rapid advancement of the communication sector, there are different Automatic Vehicle Location (AVL) techniques available now. The main aim of all the techniques is to estimate the position of a vehicle in space by measuring it precisely in time. The various technologies that can be used for establishing vehicle location include (Lobo, 1998):

- Dead reckoning (e.g. odometers);

- Beacon based (e.g. microwave, infra-red, inductive loops);
- Radio triangulation (e.g. Loran-C, Datatrack, Omega);
- Satellites (e.g. GPS)

In the dead reckoning technique, the distance travelled by a bus is calculated from the number of revolutions made by tyre. The technique includes the rate gyro, the road wheels odometer and speedometer (Krakiwsky, 1995). In a fixed route, it is possible to fix the location of a bus using this technique. But the odometer is capable of measuring only linear distances, any changes in route would mean that the new route would have to be surveyed and the relative positions should be mapped on to the absolute locations. This simple technique requires no roadside infrastructure and hence costs less. However, it is vulnerable to inaccuracies in distance and direction. Again, it also requires regular calibration of odometers for incorporating any change in circumference of the wheel, that may be due to change in tyre pressure or worn tyres (Lobo, 1998).

Beacon-based AVL systems utilise several beacons or road loops positioned at known locations along the route. Once the instrumented vehicle passes through these inductive loops or beacons nearby, it is detected and the location of the vehicle is known. This technology includes various ways of detecting vehicles such as pressure detection, magnetic detection, inductive loops, microwave and optical (infrared) beacons (Lobo, 1998). The main advantage of the beacon-based system is that it gives an exact location of the vehicle when it passes the beacon. Again, these beacons can also be utilised to transmit priority requests directly from the bus to the downstream traffic signal while using in bus priority schemes (Hounsell et. al., 1997). But since it requires roadside infrastructure to be ready before use, it is limited to those routes that have beacons installed. And hence it is not possible to track the vehicles if they divert from the fixed route for any reason. Again, failure of any one of the beacons affects all the buses in the route at that point.

Radio triangulation systems establish vehicle position relative to the navigation network by calculating the distance between the receiver on the vehicle and the radio transmitters that are the basis of the navigation network (Lobo, 1998). Examples of such radio-frequency-based techniques include Omega, Loran C and Datatrack (Krakiwsky, 1995). A radio triangulation technique is independent of route and roadside infrastructure and requires no initialisation prior to daily use. But there is difficulty in obtaining radio channels due to crowding of the frequency spectrum and hence the operating cost of the system depends on the subscription cost to the network of an existing provider. In the case of establishing a user specific radio network, high cost requires many users or potential users to share the costs (Lobo, 1998).

The Global positioning system (GPS) is a satellite-based positioning system developed by the US Department of Defence and is available to civil users under certain accuracy and reliability constraints (Krakiwsky, 1995). In this system, vehicles fitted with GPS receivers are able to determine their three-dimensional position continuously anywhere in the world in all weather conditions. GPS (also known as NAVSTAR, NAVigation System using Time And Ranging), which consists of 24 satellites orbiting some 20,000 km above the surface of the earth (Ibrahim, 2000). Each satellite transmits two radio frequencies for positioning purposes; if a satellite transmits a signal at a known time, the bus receives the signal some time later and the distance between the receiving unit and the satellite can be calculated (Garmin, 2000). Prior to 2000, locational accuracy using GPS was only within the range of 100 metres which was insufficient for bus priority purposes. Differential GPS (DGPS) provided substantially improved accuracy e.g. 5 metres (Hill, 2000), but at a cost. However, 'Selective Availability' was removed by the US Department of Defence in May 2000 (Ochieng and Sauer, 2002), so that GPS accuracy is now typically to within 10 metres, which makes this cost-effective system now potentially suitable for bus priority. The main advantages of GPS are that it is independent of route and roadside infrastructure, relatively low cost, no initialisation requirement and no subscription costs. But the system suffers from possible obscuring of signals especially in built-up areas, tall buildings and tunnels.

With number of different AVL technologies available, these technologies each have their own advantages and disadvantages. Some technologies are cheaper but more inaccurate, some are more accurate but fixed to pre-determined route and some are flexible in positioning but costly. Hence the suitability of any one of these AVL technologies depends upon the functional requirement. For example, if high accuracy is needed (e.g. for bus priority at traffic signal), then selection criteria should be locational accuracy rather than flexibility and costs. But whenever the moderate accuracy will also do (e.g. for bus operation control), then flexibility and cost is also important when selecting the technology. Table 2.1 summarises the advantages and disadvantages of various automatic vehicle location systems discussed above (Lobo, 1998).

Table 2.1: Advantages and disadvantages of AVL technologies (Lobo, 1998)

Technology	Advantages	Disadvantages
Dead reckoning	Low cost No roadside infrastructure	Vulnerable to inaccuracies in distance and direction Require regular calibration of odometer
Beacon- Based	Exact position known at beacon	Roadside infrastructures required Lack of flexibility in route choice
Radio Triangulation	No roadside infrastructure Flexible route choice No initialisation	High installation /subscription Vulnerable to interference
GPS	No roadside infrastructure Flexible route choice No initialisation	Inaccuracies near high buildings System owned by US Defence

Since the dead reckoning system is vulnerable to inaccuracies, it may not be possible to use the system alone to form an AVL system. It must then be used in conjunction with some other system for reasonable level of accuracy. Beacon-based systems have been the most common AVL system used for public transport applications in the U.K. (Hounsell et. al., 1996) and in Germany (Nickel, 1997). However the early type of this system (loop and transponder) is now less popular in the U. K. because of greater maintenance combined with costly reposition and installation requirement (Lobo

1998). Radio triangulation is a flexible system that performs well in urban areas but relatively lacks in positional accuracy. Hence radio triangulation systems have not been generally taken up for public transport applications in the U.K. (Hounsell and McLeod, 1999). GPS is more extensively used in North America than any other place although implementation is increasing rapidly in other parts of the world particularly with the accuracy now available through 'normal' GPS.

Combining two or more technologies may be advantageous to exploit the strong points of each technology. There are many places where two or more technologies combine to form a hybrid system. In many systems, dead reckoning is combined with other high accuracy systems such as beacon-based and GPS. The dead reckoning system used fills the gap of high accuracy system between their known locations. Examples are DGPS (Differential Global Positioning System) integrated with DR (Dead Reckoning) such as in Patra (Kontaratos, 1996). The use of hybrid AVL system is common in the public transport sector. The present form of AVL application in bus priority is described in the next sub-section.

2.2.1.2 Application of AVL system in bus priority

The key components of AVL systems used in public transport are location, communications and central processing and control units. These are often categorised into 5 functional components (Hounsell and McLeod, 1998):

- On-vehicle equipment to track the position of the vehicle in real-time and provide a variety of other on-bus functions. Tracking may be through location systems discussed earlier.
- Outside equipment to aid vehicle location and data transfer. These may be beacons or satellites. They may be active, sending a location code continuously, or dormant, where the location code is only transmitted to the bus on request.
- Control centre to monitor the location of the bus fleet and to take various manual and automatic functions to optimise the management of the fleet and to process data for other functions such as passenger information and bus priority requests.
- Communications systems to communicate between the control centre and the vehicles, with each having a radio transmitter and receiver.

- Signal controller and/or bus stop equipment to give priority at signals and/or passenger information at bus stops.

Obtaining real-time locational information of all buses is the key aspect of the AVL systems. It is carried out with the help of on-vehicle and outside equipment based on the techniques discussed in the previous section. This information about the location of a vehicle is usually transferred to the control centre by regular polling of the vehicles. The polling of the vehicle is carried out in regular intervals by the control centre via the communication system in place (Cassidy, 1995). The control centre utilises the collected data to check priority requirements of a bus approaching a traffic signal and request priority, if needed. Some of the AVL systems used in bus systems in the UK, with their positioning techniques, communication method, polling rate, stated accuracy and adherence to schedule are summarised in Table 2.2.

Table 2.2: Use of AVL systems in bus systems in the UK (Maxwell, 2000)

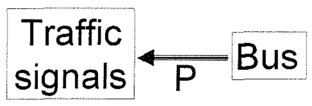
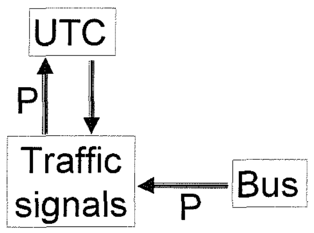
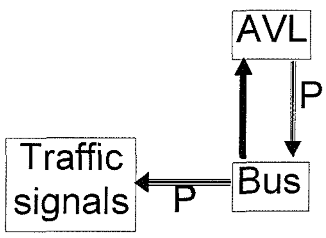
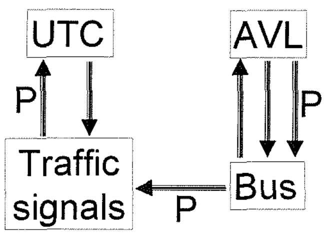
Location	Positioning techniques	Communication techniques	Polling cycle	Stated accuracy	Adherence to schedule
London, Uxbridge Rd.	Beacon	Band III radio	30 sec	2m	Headway
Cardiff	DGPS	VHF Radio	20 sec	5m	Schedule
Maidstone	DGPS	VHF Radio	30 sec	5m	None
Southampton	Beacon	Band III Radio	15 sec	-	Schedule
Leicester	DGPS	VHF Radio	20 sec	10m	Schedule
Sheffield	DGPS	VHF Radio	Event-based	10m	Schedule

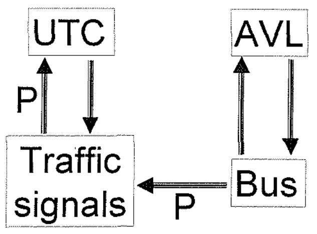
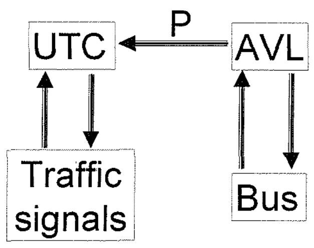
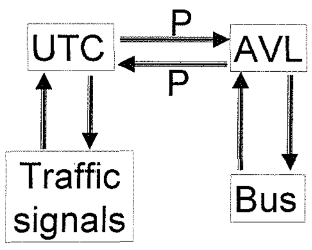
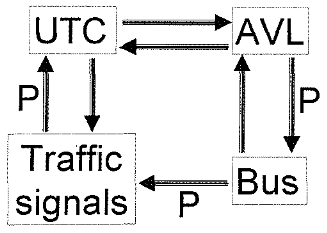
The table shows different types of positioning and communication techniques used at different places in the UK. Additionally, the method of communication within the similar techniques can be altered to form different priority architectures while giving priority to buses. These different priority architectures use different ways of requesting and implementing bus priority at traffic signals. Some of these different bus priority architectures are described in the next sub-section.

2.2.2 Bus priority architectures

The increasing use of AVL in bus priority at traffic signals has developed a variety of bus priority architectures. These different priority architectures differ in the way of interaction between bus, traffic signal, AVL system and UTC system. These different ways of interaction between different components change the method of priority request and priority implementation. A recent study in PRISCILLA (2001) has identified some 8 categories of priority architecture in use across the world (Hounsell and Wall, 2001). These are illustrated in Table 2.3.

Table 2.3: Example of bus priority architectures (Hounsell and Wall, 2001)

Architecture Category	Architecture (P = priority request)	Examples/ Cities	Priority options	
			Centralised	Decentralised
1.		Examples in many European cities		V
2		Examples in many European cities	V	V
3		Aalborg Helsinki		V
4		London	V	V

5		Zurich	v	
6		Southampton Toulouse Turin Cardiff Gothenburg	v v v v v	 v v
7		CGA	v	
8		Genoa	v	v

These can be summarised as follows:

- **Category 1.** This architecture involves bus priority at isolated junctions, without the use of AVL or UTC. Buses are typically detected using transponders, tags, or through entering an infra-red detection zone.
- **Category 2.** This architecture is as Category 1, except that the traffic signals and the priority provided operate under UTC.

- **Category 3.** This involves the use of AVL to determine bus-specific priority levels, which are then transmitted from the bus to each traffic signal controller on the route. With no UTC involved, signal control is isolated/decentralised.
- **Category 4.** This architecture is similar to category 3, except that the traffic signals are under UTC. There is no communication between AVL and UTC, so bus-specific priority requests are routed from the AVL centre to UTC via the bus and traffic signal controllers.
- **Category 5.** With this architecture, used in Zurich, Switzerland, AVL is used predominantly for fleet management. Buses and trams are given 'absolute' priority using loop detection.
- **Category 6.** This involves one-way communication of bus location and priority requirements from an AVL centre directly to UTC. AVL becomes the primary source of bus location upstream of signalised junctions for priority purposes.
- **Category 7.** Common in many French Cities, involving centralised UTC/AVL integration; UTC plays an active role in informing AVL of each proposed signal stage change at each junction, and requesting the location of any approaching buses or trams which should influence the stage change time (i. e. where priority is needed).
- **Category 8**. This architecture illustrates the highest level of two-way communication between the system components. In the example in Genoa, Italy, buses are allocated a priority level by the AVL centre and transmit this directly to the traffic signals for implementation subject to UTC commands. At a higher level, strategic data is transferred between the AVL and UTC centres.

Out of all these different architectures, two main categories can be identified in the UK. They are 'category 6' (Centralised AVL-UTC communications) and 'category 4' (Decentralised AVL-UTC communications). The workings of these two architectures are illustrated and described below (Hounsell and McLeod, 1998).

2.2.2.1 Centralised AVL-UTC communications (Category 6)

This architecture (Figure –2.2) uses AVL as the primary bus location function with a direct interface between AVL and UTC centres. This system involves the use of AVL for bus location and the use of a priority algorithm in the AVL centre to determine priority requirements for each bus (e.g. according to its headway or adherence to schedule). A priority request is then passed to UTC control. Where UTC is centralised (e.g. SCOOT), signal timings are then re-optimised centrally to incorporate the priority request and transmitted to the local signal controller for implementation. The priority is provided if the bus is located on a junction approach and within a pre-defined distance of the junction. This type of system has been trialled in Southampton where the SCOOT centralised traffic responsive UTC system is operational (Hounsell and McLeod, 1999).

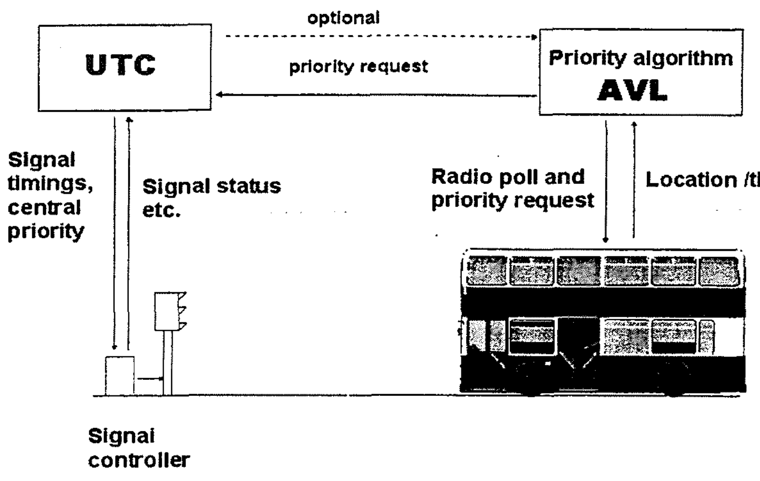


Figure – 2.2: Centralised AVL-UTC communications (Hounsell and McLeod, 1999)

2.2.2.2 Decentralised AVL-UTC communications (Category 4)

In this architecture (Figure –2.3) also the calculation of individual bus priority requirements (e.g. according to its headway or adherence to schedule) is carried out at the AVL centre based on AVL information about the positions of buses. However, priority requests are then returned to individual buses with the radio polling message, rather than to the UTC centre. Priority requests are then transmitted either directly from the bus to the downstream traffic signal (e.g. by short range radio) or, as currently proposed for London, transmission is via a roadside beacon, with cable or radio communications from the beacon to the traffic signal controller. This architecture has been chosen for London, with trials undertaken on the Uxbridge Road during the summer of 1999 (Hounsell and McLeod, 1999). The result from the trial is referred while discussing the results from this research in Section 7.3.1.

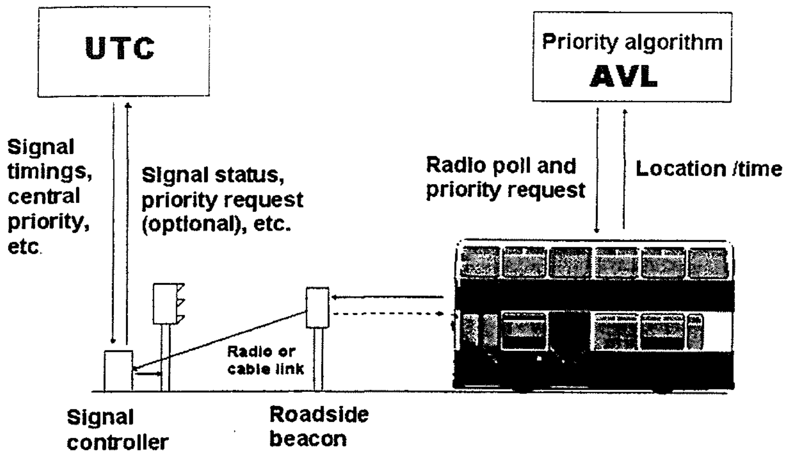

UTC
Priority algorithm
AVL
Signal timings, central priority, etc.
Signal status, priority request (optional), etc.
Radio poll and priority request
Location /time
Radio or cable link
Signal controller
Roadside beacon

Figure –2.3: Decentralised AVL-UTC communications (Hounsell and McLeod, 1999)

These architectures can accommodate various AVL techniques available at present. The first trial of beacon-based decentralised AVL-UTC architecture was carried out in London, whereas the trial of beacon-based centralised AVL-UTC architecture was carried out in Southampton. Since then, there are more and more systems in implementation for bus priority in the UK. Table 2.4 gives examples of AVL systems used for bus priority in conjunction with SCOOT in the UK, with their different positioning techniques and communication architecture between AVL-UTC.

Table 2.4: Use of AVL systems for bus priority in SCOOT in the UK (Maxwell, 2000)

Location	Positioning techniques	Communication architecture
London, Uxbridge Rd.	Beacon	Decentralised
Cardiff	DGPS	Decentralised
Maidstone	DGPS	Decentralised
Southampton	Beacon	Centralised
Leicester	DGPS	Decentralised
Sheffield	DGPS	Centralised

This table shows that the use of GPS-based AVL system and decentralised communication architecture in bus priority at traffic signals is relatively common in the UK. One reason for growing use of GPS is the flexibility which it provides in the case of changing bus routes and for the allowance of 'virtual' detectors to be 'sited' (and changed) according to requirements. A 'virtual' detector is where the co-ordinates of the detectors are programmed into the on-bus software, so that when the bus reaches this location (as determined from the on-board GPS), an action is triggered (e.g. a priority request is sent). In principle, 'virtual' detectors can be sited anywhere on the bus route, with no need for physical infrastructures . The use of more decentralised architecture may be due to its capability of including isolated junctions and complex nature of centralised communication architecture. These were some of the key reasons for adopting this architecture in London (Hounsell and McLeod, 1998). This 'Decentralised AVL-UTC communications' architecture using GPS positioning techniques is the system chosen for bus priority in this research.

With this growing use of AVL systems in bus priority, there is scope for a variety of strategies to be implemented. The benefits from these bus priority strategies are different in type and magnitude. For example, the strategy giving priority to all buses effectively reduces the journey times but does not improve passenger waiting time that effectively. The strategy giving priority to late buses improves waiting time but gives lesser journey time benefit. The strategy giving a higher amount of priority using higher target degree of saturation (DOS) for side road may give more journey time benefits but give more delays to the non-priority traffic. Additionally, the performance of such strategies may be influenced by field factors such as passenger demand, intervention option (holding early buses) available and type of bus operation (timetable/scheduled). Exploring the performance of different priority strategies under various field conditions is the focus of this research. Although few field trial results are yet available, the scope for testing a variety of strategies is limited. Hence it has been decided in this research to undertake simulation modelling of the system to evaluate a range of priority strategies. The next section deals with the modelling options available to analyse the system for this research work.

2.3 Modelling bus priority at traffic signals

In general, modelling requirements depend upon the aim and objectives of the work. Since the main aim of this research work is to model different bus priority strategies under various scenarios, a detail modelling of main components of a bus system is required. These main components are bus, bus stops, traffic signal, AVL system and bus priority. Each of these components includes modelling issues to define its characteristics. Some of the main modelling issues are: bus - generation and movement; bus stop - passenger generation, waiting time and bus dwell time calculation; traffic signal - signal timings and delay calculation; AVL system - bus positioning; and bus priority - priority criteria and implementation method. So a model, capable of modelling these components and their modelling issues in detail, is required for this research work. A review of available models carried out to find their suitability for this research work is outlined in the sub-section below.

2.3.1 Review of existing models

With rapid growth in computer technology, there are numbers of methods and programs available for modelling urban traffic. Most of them have become increasingly sophisticated and now attempt to deal with network interaction (Willoughby and Emmerson, 1999). Grossly, these are divided into two groups according to their modelling approach. These are:

- Aggregate methods
- Simulation modelling

2.3.1.1 Aggregate methods

These models are the most common models in use for traffic network modelling, incorporating traffic assignment. These traffic models use formulae with average values of the parameters such as vehicle flow and capacity over a period of time (Willoughby and Emmerson, 1999). CONTRAM, TRIPS and SATURN are the examples of this type of model (although CONTRAM can be disaggregated to an individual vehicle level). These traffic assignment models can be used in the evaluation and assessment of traffic management schemes. Though all of these models contain some facilities to model public transport within them, it is not their main concern; they are more aimed at network traffic assignment rather than public transport modelling. In particular, these models do not have the facility to model many of the elements required for this research including AVL systems and differntial bus priority. Hence these models are found to be not suitable to carry out this type of research having the objectives of modelling different issues of bus priority in greater detail.

2.3.1.2 Simulation models

With simulation, an attempt is made to model the behaviour of each vehicle/driver as it moves through the road network, based on the characteristics of the vehicle and driver behaviour. In Czogalla's (1997) view, microscopic simulation environment provides an excellent experimental base for the design of control strategies which enables the visual and statistical evaluation of the achieved results. Santhakumar and Harihara (1992) and Agrawal (1994) find the simulation technique as the ideal tool for the study of bus operation in a transport corridor. There are a number of packages now available for simulation modelling of traffic.

The SMARTEST Project (Bernauer et al, 1997) found more than 60 developers producing modelling programs around the world (Bagot, 1999). They mentioned that it is not yet clear whether such software has reached maturity, and whether it can provide the level of reliability that invokes full confidence in the user. Out of the models they found in the marketplace, only PARAMICS and VISSIM have got facility for modelling public transport priority. Other than these, NETSIM is also found to have facility for modelling public transport.

TRAF-NETSIM is a microscopic simulation model that provides a detailed evaluation of proposed operational improvements in a signalised network. The model can deal with signal-controlled intersections and interaction between cars and buses explicitly. It applies interval-based simulation to describe traffic operations where each vehicle is a distinct object that is moved every second. Vehicles are moved according to car-following logic, response to traffic control devices and response to other demands such as buses stopping to pick up passengers at bus stop. There are some studies carried out by Khasnabis et.al. (1996), Al-Sahaili and Taylor (1996) and Khasnabis and Rudraraju (1997) which have used TRF-NETSIM to evaluate different bus priority strategies in different parts of USA. But Khasnabis and Rudraraju (1997) have mentioned that there are limited applications of NETSIM with a bus as the primary vehicle. Again, it does not model differential priority and AVL systems in detail.

PARAMICS is a suite of high performance software tools for microscopic traffic simulation. It provides simulation, visualisation, interactive network creation and editing, interactive adaptive signal control, on-line simulation data and statistics and others in one package (Bernauer et al, 1997). PARAMICS, includes a microscopic car-following and lane-changing model, dynamic and intelligent routeing and inclusion of intelligent transport systems (Druitt, 1998). PARAMICS also takes account of public transport and its interaction with other modes at bus-stops and through bus priority measures. However, there is no literature suggesting the modelling of differential priority options and AVL systems in PARAMICS.

VISSIM is a general purpose computer-based traffic simulation system (Fellendorf, 1996). VISSIM models links, roundabouts, priority junctions, signalised intersections and networks at a high level of detail. The model contains a psycho-physical car following model for longitudinal vehicle movement and rule-based algorithm for lane changing (Bernauer et al, 1997). Its signal control program uses detectors for microscopic and macroscopic measurements (i.e. speeds, volumes, travel times) of traffic to decide the current signal aspects. The simulation is microscopic and stochastic with fixed time-slices (one-second interval). In this case also, there is no literature specifying the modelling of differential priority or an AVL system as required in this research.

PRISCILLA (2001) carried out a review of some simulation models commercially available in the market. The review included two more simulation models other than those reviewed above. A summary of the features of these simulation models with regards to the modelling components required for the research is given in Table 2.5 (PRISCILLA, 2001).

Table 2.5: Comparison of commercially available models with research requirements

Modelling components	Simulation models				
	FLEXSYT II	HUTSIM	PARAMICS	TRAF_ NETSIM	VISSIM
Bus	✓	✓	✓	✓	✓
Bus stop	×	×	×	✓	✓
Traffic signals	✓	✓	✓	✓	✓
Bus priority	✓	✓	✓	✓	✓
Differential bus priority	×	×	×	×	×
AVL system	×	×	×	×	×

KEY:

✓ - feature is reported as being modelled

× - feature is not available as built-in facility

As shown in the table, modelling of differential bus priority and AVL system is not specifically mentioned in the available literature of these simulation models. These commercially available models are more complex in their treatment of general traffic, but less complex in their treatment of buses than required for this research. Their use would require significant program development which could not be guaranteed within the scope of this research. Hence a review of other research models carried out to find their suitability to the project is outlined below.

2.3.1.3 Other research models

There are a number of other research models developed for the study of specific bus operation and priority schemes. One such simulation model developed by Santhakumar and Harihara (1992) was aimed at evaluating the performance of urban bus routes in Triruchirapalli in India. The main objective was to find an optimum combination of different transportation system management (TSM) options to get maximum reduction in journey time. Another simulation model developed by Agrawal et al (1994) was used for performance evaluation of bus operations in the ring road of New Delhi, India. In this case, the existing bus operation system was compared with the simulated system by changing bus stop spacing, removing some of them, varying bus frequency and introducing bus lanes. Both of these models were targeted to find better system management rather than the bus priority options of interest in this research.

A microscopic model developed by Liu et al (1999) was used to study bus priority measures for guided bus. Though this model was primarily intended to model guided bus it is relevant to the modelling of other buses also. The model incorporates modelling of bus services, bus stops and selective vehicle detection. The main two priority options modelled are extensions and recalls with only one signal change (extension or recall) permitted per cycle. Though the approach of the model is suitable to this research, it does not model AVL and differential priority.

A model developed by Salter and Shahi (1979) was aimed at predicting the effect of bus priority using computer simulation. The model incorporates passenger modelling by distributions representing their arrivals, boarding time and alighting time at each bus stop. The model also uses simulation of non-bus vehicles in which vehicles are generated before the junction to ensure correct level of service at the junction. The model uses uniform time scanning for this microscopic section of the simulation. But this model also lacks modelling differential priority and AVL, which is the main objective of this research work.

The simulation model SPLIT (Selective Priority to Late buses Implemented at Traffic Signal), developed by McLeod (1999), simulates the differential priority for buses at traffic signal depending upon the headway. The model was used to model survey data from the Uxbridge Road in London including 29 bus stops and 14 signalised junctions in a linear network. The model was used to test 9 alternative priority strategies based on the headway algorithm developed assuming that the signals are under SCOOT UTC control. This model is very much compatible with the nature of this research, but the model also does not address all the modelling aspects required by the research objectives. The model does not cater for schedule-based bus service, traffic signal timings and AVL systems. The model also does not calculate delay savings to the buses and delays for general traffic on an individual basis. Beside that the model also lacks visual representation of system operation which is very important in terms of understanding the happenings in the system during simulation period.

2.4 Summary

Bus priority is considered as one of the most influencing factors for making bus services regular and attractive. The priority measures include with-flow lanes, contra-flow bus lanes, bus only streets, busways, bus only turns (bus gates) and bus priority at signalised junctions. Bus priority at traffic signals is most effective in congested city areas with many signalised junctions. Nowadays, bus priority with selective vehicle detection is common in a number of developed countries. In recent years, there is an interest in giving priority to the buses according to their individual requirement and constrained by junction factors.

Bus priority in the UK is generally implemented by the method of green time extensions and recalls. It needs some form of detection of buses to find their arrival time to match the priority award. The use of automatic vehicle location (AVL) techniques for this purpose is growing. The different AVL techniques in use are dead reckoning, beacon based, radio triangulation and satellite based. There are several different architectures in which these AVL systems can be used. AVL systems used for bus priority in the U.K. have developed recently in two forms. These are centralised AVL-UTC communications and decentralised AVL-UTC communications. These different priority architectures have made it possible to use several different bus priority strategies. Modelling these different strategies and their performances under various field scenarios is the main focus of this research. This will be carried out by simulation modelling of a bus priority system.

The main modelling components of bus priority at traffic signals are bus, bus stop, traffic signal, AVL systems and bus priority. To determine the suitability of existing models, both aggregate methods and simulation modelling programs available for modelling traffic were reviewed. Though the aggregate models have got some facility to model public transport, they do not model bus priority in detail. In simulation approach also, commercially available packages, such as PARAMICS, TRAF-NETSIM and VISSIM, are not found to be modelling differential bus priority and AVL system in detail.

There are other research models developed for study of specific bus priority schemes. The models of Santhakumar (1992), Agrawal (1994), Liu et al (1999) and Salter and Shahi (1979) were found to be useful in designing the model but were not appropriate to use directly. The simulation model SPLIT developed by McLeod (1998) is particularly related to this research. The model simulates differential priority for buses at traffic signals depending upon the headway. But the model also does not model signal control strategy nor AVL nor calculate traffic delays in detail.

Since the models reviewed were not suitable to carry out this research to achieve set objectives, the only way ahead was to develop a simulation model specifically for this research programme. The modelling methodology adopted to develop the model is detailed in Chapter-3 and the development of the model follows in the subsequent chapters. With development of the model, it is expected to develop strategies for differential priority to the buses at traffic signals.

Chapter Three

Modelling Methodology

3.1 Introduction

Developing a simulation model of a bus operation system needs careful assessment of all the associated parameters. The key factors that will affect the performance of buses in a route are their starting condition (e.g. lateness at the start of the route), their movement along the route, bus stops, traffic signals and interaction with other traffic. Hence the simulation model of a bus system should be able to model these key factors in depth to mimic the field situation. In this research, the main objective being the simulation of advanced bus priority strategies at traffic signals, there are other parameters that the model should address also in detail. For this purpose, the model should be capable of modelling details of differential bus priority options at traffic signals and be able to incorporate AVL systems to locate buses. Finally the model should provide enough information about the impact of different levels of bus priority in the system.

In summary, the key requirements of the model in order to carry out the research are set as follows. The model should

- Represent a proper linear route having a number of traffic signals and bus stops;
- Have a controlled method of generating buses in different intervals to accommodate both headway based as well as schedule based bus operation;
- Model bus stops with estimation of boarding and alighting passengers, their waiting times and dwell times;
- Be able to model the different possible AVL systems available with different accuracies and polling frequencies;
- Model traffic signals with their signal timing and delays;
- Model different bus priority strategies;

- Be able to estimate benefits to the buses and disbenefits to general traffic from the bus priority options;
- Be able to produce output such as the progression of bus along the route, journey time, delays at intersections, dwell time at bus stops, passengers boarding and alighting at each stop, waiting time of passengers, bus occupancy and punctuality.

The methodology of modelling main components along with their modelling issues is described in the subsequent sub-sections. The method of incorporating these components into the model is described in Chapter 5 (Model development).

3.2 Buses

Buses are the vehicles of main interest in the model. Their generation, movement and interaction with other components are the main focus of this model. Each bus generated at this stage is assigned a unique identification number that will relate to the parameters representing all its functional aspects. These parameters include bus size, its type (Double deck/single, one door/two door), passenger capacity and exit node. The main modelling issues of this bus component are bus generation and movement. These issues are discussed in next sub-sections.

3.2.1 Bus generation

Theoretically, buses are generated according to their time headways or their schedule. In headway-based operation, buses are generated at a defined time interval. Whereas, in scheduled-based operation, buses are generated according to their timetable. But in the field, generation of buses is affected by several factors such as bus operator, availability of buses, their condition and weather condition. Hence buses often do not start at the beginning of their route exactly at the defined headway or timetable. This is an issue to be reflected in the model while generating buses. This fluctuation in generating buses can be incorporated in a model by using a distribution for generating buses. An appropriate distribution based on field data collected for a long time is used in the model. Using the distribution, the time headway of the next bus is calculated when the first bus is generated. Then the next bus is generated in the system using this headway.

3.2.2 Bus movement

Once the buses are generated at the origin, they start moving in the route links according to the movement parameters assigned. Bus movement parameters describe the behaviour of bus movements on the route to the destination once started from the origin. In practice, the movement of a bus is based on various parameters such as condition of the bus, condition of the road, weather condition, special events, roadside parking and variation in traffic conditions. Especially, roadside parking adjacent to a bus stop severely disrupts buses while getting in and out of the bus stops. However, simulating all these parameters to calculate the journey time of a bus would make the program unnecessarily complex. Again, with simulation, the parameters are kept constant anyway when comparing strategies. Therefore modelling all aspects of random variability is thought to be unnecessary unless they affect strategy evaluation. Beside that, it is easier to collect link journey time of buses in the case of the bus service. A link in this case is described as a section of a bus route between two consecutive points where a bus is very likely to stop (e.g. bus stops, traffic signals). A link journey time collected in the field incorporates the effect of traffic condition in that link. This average link journey time is used in modelling bus movements in the model.

3.3 Bus stops

Bus stops are the place where passengers arrive and wait for a bus and where boarding and alighting takes place. When a bus comes, it stops there while the passengers inside alight and the waiting passengers board. The generation of passengers, their waiting time and dwell time of buses are the main modelling issues of a bus stop. These issues are discussed separately in subsequent sub-sections below.

3.3.1 Passenger generation

Passenger demand is the main factor deciding route, frequency, bus stops and bus operation as a whole. The estimation of numbers of boarding and alighting passengers at a bus stop is therefore is an important component of a model. There are different ways in which the passenger arrival at a bus stop can be modelled. In general, these can be represented by a distribution over a time period (Salter and Shahi, 1974). This may be a normal distribution with a mean representing the average passenger flow to the bus stop and a variance fixed for all bus stops (Liu et al, 1999); or it may be a simple relationship between the number of boarding and alighting passengers in terms of rate of boarding and alighting passengers and the headway (Hounsell and McLeod, 1999). Assuming the random arrival of passengers, the relationships then used are as follows.

*Number of boarders = boarding rate * bus headway*

*Number of alighters = alighting rate * bus headway*

Hence by having a boarding rate and alighting rate from a survey, the number of passengers boarding and alighting can be calculated. This random passenger arrival assumption is reasonable (Seddon and Day, 1974) for a high frequency service where the passenger has no expectation about the arrival time of the bus before reaching to the bus stop. However in the case of low frequency service, the passengers tend to arrive at the bus stop towards the scheduled time of the bus arrival and hence the assumption is not valid. The bus headway used here is the difference between the departure times of two buses. Since departure time is not that straightforward to calculate, the number of passengers generated is carried out in two parts. Firstly, the number of passengers arrived between departure of first bus and arrival of second bus is calculated. Then the number of passengers arrived during the dwell time are calculated and added to the total number of boarding passengers.

Although the number of boarding passengers may depend upon the bus headway, this may not be the case in terms of alighting passengers. Since the number of alighting passengers at any stop will be dependent upon the passengers already inside the bus, it may not be affected by the present headway at the bus stop where the passengers are alighting. Hence, the number of alighting passengers is estimated according to the fixed percentage of total passenger inside assigned for each bus stop. The relationship for calculating these alighting passengers is:

*Nos. of alighters =passenger inside a bus * % of passenger alighting*

3.3.2 Waiting time:

The waiting time of a passenger can be the time between the arrival of a passenger to the arrival of a bus or to the boarding of the passenger or to the departure of the bus. While using these different definitions, the average passenger waiting time differs slightly depending upon the bus dwell time at a bus stop. However, the change being very small, it does not influence the result while comparing one priority strategy with another. In the model, the waiting time is used as the time between the arrival of the passenger at a stop and the arrival of a bus in which the passenger boarded into. While considering uniform rate of arrival of passengers at a bus stop for boarding into a bus, the waiting time can be approximated by half the time between the departure of first bus and the arrival of second. This can be expressed in following mathematical form:

*Average waiting time = 0.5 * time gap between buses*

This is the case for the services where passengers arrive in a regular basis without knowing the actual arrival time of a bus. However in case of schedule-based operation (low frequency service), passengers often time their arrival according to the timetable with an appropriate allowance for punctuality. In the model, the waiting time is calculated by using this simple relationship that is valid for high frequency services. The total waiting time of passengers for a bus is then calculated by multiplying the average waiting time by the number of passengers boarding.

3.3.3 Dwell time

The amount of time spent by buses at bus stops while passengers board and alight is a very significant part of overall journey time of bus services (York, 1993). Based on a study of London bus services, York found that the stop time of a bus could be expressed in terms of dead times, passenger alighting time and boarding time. The dead time is the fixed time, dependent on the particular bus type, which mainly consists of the time taken to open or close the door and time taken to check the traffic. Alighting and boarding times are the times taken by passengers while alighting and boarding. These times are the product of number of passengers and the time per passenger. The time per passenger varies between different bus types and fare collection systems. The stop time T can be related to the number of passengers alighting a and the number of passengers boarding b_i using the ith process of boarding, where there are m different processes of boarding by expression:

For buses with single door

$$T = D + Aa + \sum_{i=1}^{m} B_i b_i$$

For buses with two doors

$$T = D_b + \sum_{i=1}^{m} B_i b_i$$

or

$$T = D_a + Aa$$

whichever is greater.

Where, A is alighting time per passenger, B is boarding time per passenger, D is 'dead time' for one door buses, and 'D_a and D_b' are dead times for two door buses.

These variables are all dependent upon the type of the bus, which includes the size of a bus, its number of decks and the mode of paying for tickets. The values of these variables will be obtained from the field survey. Another formula used by Liu et al (1999) gives the bus dwell time (T) at a bus stop that is related to the number of passenger (N) waiting at the bus stop by:

$$T = (a * N + b)$$

Where, a is the boarding time per passenger (including paying for ticket) and b is the time for the bus' door being opened and closed.

Though this formula is simple in nature but does not consider the effect of the number of passengers alighting and the variation in bus type. Hence, York's formula is used in calculating dwell time.

3.4 Traffic signal

Traffic signals are the location where buses may be given priority, which is the main focus of this research. The modelling of a traffic signal consists of modelling its position, signal timings and delay savings calculation. While modelling, each traffic signal is represented according to its relative position in the route. Each of the signals is given a unique identification number to refer to its individual properties that include signal timings and general traffic flows in each arm etc. These are discussed in subsequent sub-sections.

3.4.1 Signal timings

Modelling the real field situation of traffic signal timing is not a simple issue. In the model, this issue is simplified by modelling fixed-time signals in the place of SCOOT controlled signals. Here, the sequence and length of stages of a signal are kept fixed for whole modelled period. This occurs predominantly within SCOOT. Average stage length and cycle time duration was then calculated from the individual stage lengths and cycle times obtained from SCOOT data for each junction. This simplification was done because of the complex nature of SCOOT (e.g. varying levels of traffic, signal timings and co-ordination) which can cause significant random variability that can mask the impacts of the priority strategies if it were to be modelled.

3.4.2 Delay calculation

Delay calculation is a very important part of traffic signal modelling. Since the concept of differential priority to buses is based on reducing delays of late buses at traffic signals, it is a very important part of this research work. Again, it forms the basis for calculating resulting benefits and disbenefits from bus priority at traffic signals. In this model, the delays to the buses are calculated separately from general traffic. The delay to a bus is calculated on an individual basis whereas that of general traffic is calculated using aggregate values.

The delays at signals can be divided into two parts; delays due to the red signal and delays due to blocking by vehicles in front. Hence while modelling the delays to a bus at traffic signal, both the delays due to signal timing and due to presence of other traffic in front should be accounted for. If a bus arrives at a signal in the green phase and has no other vehicles in front, then there will be no delay to the bus. But if the bus arrives at the back of a queue, the bus will have to follow other traffic to cross that signal. Even arriving on the green phase, there may be some delay associated with it.

Hence the departure time of a bus is determined taking notice of other traffic too. Now knowing the departure time of a bus, taking account of general traffic in front and the arrival time of the bus at the signal, the delay at that signal is calculated.

In the model, the general traffic is generated according to the average arrival flow at all approaches of a signal. When the signal is red the generated traffic is queued one after another. But when the traffic signal turns to green, the traffic is discharged at the saturation flow rate. The discharged general traffic is not modelled after crossing the stop line but buses will continue their journey into next link and so on. The calculation of delays to the general traffic will be carried out on a similar basis to that of SCOOT (Hunt et. al., 1981) system. However, the delay calculation is simplified by taking average inflow and outflow instead of cyclic profile as in case of SCOOT. The delay calculation method used in the model is illustrated in figure 3.1.

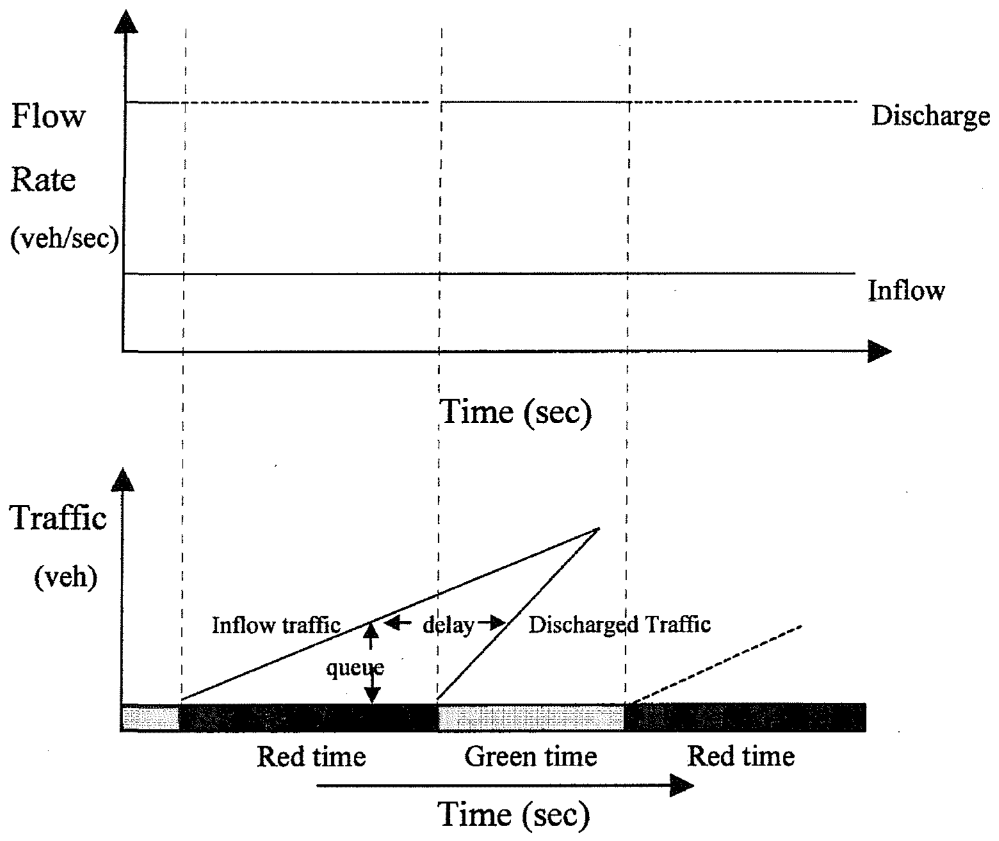


Figure 3.1: Illustration of delay calculation at traffic signal

As illustrated in the figure, the traffic is generated according to the average flow during the whole modelled period and is discharged at the rate of saturation flow during the green period. To simplify the model, it is assumed that all the traffic accumulated during the red period will be discharged completely during green time when there is no priority. The delay of these vehicles is calculated using the time between their arrivals at the back of the queue to their discharge at stopline. However in the field, inflow and discharge vary according to time rather than being constant. This variation in inflow varies the junction delays and degree of saturation and ultimately the amount of priority that can be provided to the buses. However, modelling these variations makes a model very complicated and does not make significant difference while comparing one strategy with another. Hence average values of inflow and discharge are used in the model to calculate junction delays.

3.5 Bus priority methods

Modelling bus priority at traffic signals is the main focus of this research. The modelling of bus priority has been started with the system of giving priority to all the buses at traffic signals. In this method, a bus is detected by a detector placed upstream of a traffic signal and its arrival time at the stopline of that traffic signal is estimated. Then the signal period for that expected arrival time of the bus is calculated. If the bus is estimated not to arrive during a green period, then a priority is awarded within the allowable limit. The main parameters to be modelled in this type of bus priority are bus detection position and method of priority implementation. These are discussed in subsequent sub-sections.

3.5.1 Detector positioning

The position of the bus detector plays an important role in the efficiency of bus priority. The detector should be located at an optimum position so that there will be less uncertainty in journey time prediction but sufficient time for priority process. A longer detector distance (from detector to stopline) gives a higher journey time to reach the stopline and the allowable extension time can be increased. However, it makes estimation difficult due to variability in journey time in the field. An earlier study (Bretherton et al, 1996) recommended a location of 70-100 m upstream of the junction. However, the literature stressed that the detectors must be located downstream of bus stops. These recommendations will be taken into account while deciding the detector locations in the model.

3.5.2 Method of Extension and Recall

Once a bus is detected at the detector upstream of a traffic signal, its expected arrival time at the stopline of the signal is estimated. In the model, the estimation of journey time between a detector and stopline of the downstream traffic signal is based on the average link speed. This estimated journey time is added to the detected time to give the expected arrival time (EAT) of the bus at the stop line of the downstream signal. If EAT of a bus is during a green period, then there is no need of priority. If the arrival time is just after end of green time (EoG) then the time difference between the EoG and EAT of the bus is calculated. If this difference is within the maximum allowable limit, then extension of that amount equal to the difference is granted. If the difference is more than the maximum allowable limit, then the extension will not be granted. Instead, the maximum allowable amount of recall is granted. The examples of giving priority using 'extension' and 'recall' along with a recovery method are shown in Figure 3.2 and Figure 3.3.

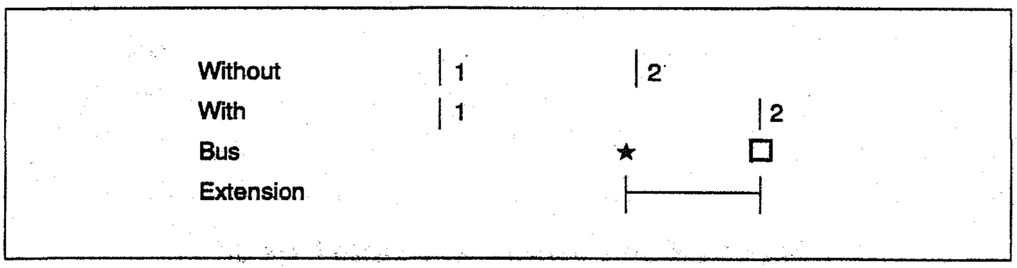


Figure 3.2: Example of an extension (Bowen, 1997)

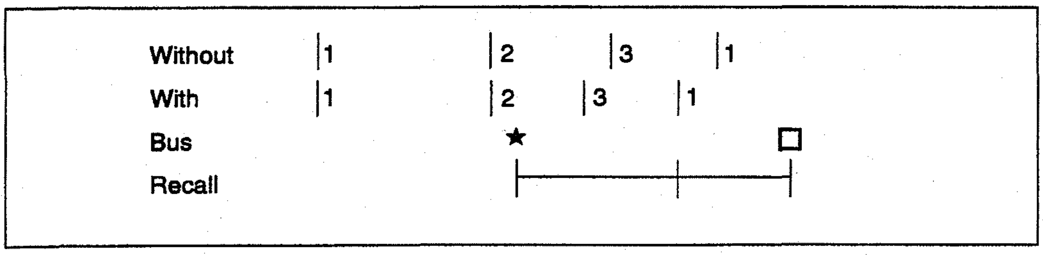


Figure 3.3: Example of a recall (Bowen, 1997)

In the figures, the symbol ★ indicates the detection of a bus and the symbol □ indicates the bus crossing stopline. In Figure 3.2, a bus is detected in stage 1 (green period for the bus approach) and is expected to miss the green period. Hence an extension is provided by extending the stage until the bus crosses stopline. In Figure 3.3, a bus is detected in stage 2 and hence a recall is provided by recalling the stage 1 (green period for the bus approach) earlier.

The maximum amount of recall allowed is dependent upon the target Degree of Saturation (DOS), which is also commonly abbreviated as 'x', for non-priority traffic in a junction (The term DOS is used in the following text, as being most commonly adopted by practitioners in this field). The target DOS is used to check the amount of time that can be provided without exceeding this DOS for the non-priority links. In the case of an extension, the maximum allowable amount is constrained by the detector distance. The maximum amount of extension is less than or equal to the amount of journey time from the detector to the stopline of the downstream signal. This is to make sure that the extension awarded is the continuation of the 'green period' at the time of detection.

Once this method of giving priority to all buses was completed, then the model was further developed to incorporate differential bus priority. In this case, the maximum amount of priority allowed may be different for different buses depending upon their performance (i.e. lateness). This is dependent upon the type of priority strategy in use. More about the modelling of these strategies is described in next sub-section.

3.5.3 Differential priority strategies

Modelling of differential priority requires a clear definition of the priority strategy to give priority to the buses. This needs defined criteria to check the eligibility of the priority requirement of a bus approaching a traffic signal. A headway based differential priority criteria (as mentioned earlier in Section 2.1.3), depending upon the lateness of a bus, was used to select priority levels in the London system (Hounsell and McLeod, 1999). In that system, the lateness was defined in terms of headway ratio 'R' of actual bus headway to the expected headway. This headway ratio was used to decide the level of priority to be awarded. The different priority strategies formulated and tested are shown in table 3.1 (McLeod, 1998).

Table3.1: Alternative priority strategies tested (McLeod, 1998)

Logic no.	Description
1	No priority for bus
2	Extensions only for all buses
3	Extensions + R_{HIGH} for all buses
4	Extensions + R_{HIGH} for buses with $R \geq 1.00$
5	Extensions + R_{HIGH} for buses with $R \geq 1.25$
6	Extensions + R_{HIGH} for buses with $R \geq 1.50$
7	Extensions + R_{HIGH} for buses with $R \geq 1.75$
8	Extensions + R_{HIGH} for buses with $R \geq 1.0$ Extensions only for buses with $R \leq 1.00$

Notes:

R_{HIGH} = recalls using a 'high' target degree of saturation threshold (110%)

R_{NORMAL} = recalls using the 'normal' target degree of saturation threshold (95%)

These priority strategies are shown here as an example and they form the starting base for modelling differential bus priority strategies in this research. Further strategies will be formulated and explored later in the application of the model (Chapter 7).

3.6 AVL systems

The AVL system is used to determine the locations of the buses. The modelling of AVL is concentrated on beacon-based and satellite based (GPS) systems. These different systems help a bus in determining its position in a network. This positional information is then transferred to the AVL centre by means of polling. Hence the modelling of beacon-based AVL system is divided into two parts:

- Locational techniques
- Polling

3.5.1 Locational techniques

In the case of a beacon based system, the location of each bus is determined primarily using roadside beacons and then it is refined by odometer readings. While modelling, each of the beacons placed in the network are given an identification number relating to their position in the network. The identification number (ID number) and their distance from a reference point (generally from origin) is given as input in the model. While updating the position of each bus in every second, the simulation model checks buses equipped with AVL equipment to see if a beacon has been passed during the update of the bus position. If a beacon lies between the updated and the previous bus position, then the individual beacon code with its positional distance is transferred to the associated bus. The distance between two successive beacons is determined by using odometer readings. Odometer readings are modelled by measuring the distance travelled since the last beacon passed, and then errors in odometer measurements are added to the distance travelled.

In the case of a GPS system, the location of each bus is determined continuously by using a GPS receiver onboard. In principle, this continuous positioning of a bus could be used as a multiple detection of a bus especially while approaching traffic signal for priority. However, there are no traffic signal priority strategies able to take full advantage of this potential yet. The GPS system modelled in here is therefore based on single detection method used in Cardiff (Hill, 2000) and elsewhere. In this case, the main modelling involves the estimation of GPS error incurred during the location determination. A recent test carried out for determination of GPS error (using a Garmin 12x1 GPS) showed the accuracy within +/- 9.8 metres at 99.73% confidence (Rupprecht, 2001). The test result showed that the GPS error follows a normal random behaviour with mean 0.0 metre and standard deviation of 3.3 metres. Hence the GPS error is modelled by generating a random number from normal random distribution with mean 0 and standard deviation 3.3. This error is added to the actual position of a bus at that moment stored as the virtual position of the bus. This is the position recorded by the computer on board of a bus with the aid of GPS equipment.

This estimated position of a bus along the route is communicated to the AVL centre. This communication between AVL centre and buses is carried out by polling, in which the buses are contacted by the AVL centre to find their position. The modelling of polling is further discussed in the next sub-section.

3.5.2 Polling

The AVL centre polls each bus in the route at a regular time interval to determine its position in the route. The polling interval depends upon the accuracy requirement of the system as well as the number of buses to be polled in the system relative to the communication capacity. Polling rates are typically 20-30 seconds per bus (Hounsell and Wall, 2001). Polling information, passed from bus to the AVL centre, may include bus number, route number, code of last beacon passed and odometer reading. In the model, polling is represented by a 'blinked' signal on the bus. This information about the position of a bus is used to check the priority requirement of that bus while approaching a traffic signal.

3.6 Other issues

Other issues of modelling bus priority at traffic signals include bus bunching and the treatment of overlapping routes. These issues are the operational side of a bus operation system. These issues with their modelling concepts are discussed in subsequent sub-sections.

3.6.1 Bus bunching

Bus bunching is the outcome of the deviation of a bus service from its schedule in terms of time. In this process, the bus lagging behind the schedule picks up more passengers than average and hence takes more time at each bus stop and lags a bit more again. Hence the headway goes on increasing and bunches with other buses behind and two or more buses will move together in a bunch. In some cases, the buses behind the lagged bus may simply follow to the destination. This may be due to reluctant driver or due to space availability at bus stops. However, in some other case, the bus behind the lagged one may overtake it to pick up passengers. But then this bus may also face high passenger boarding numbers and may be overtaken by the lagged bus behind it. Then a sort of leap-frogging action may occur. As such, there is no evidence that a bus will follow particular pattern in case of bunching.

Given these uncertainties, bunching is modelled here by estimating the departure time of a bus at a stop with the arrival time of the next bus approaching that stop. When a bus comes at the bus stop, its departure time is estimated considering the dwell time needed for the passengers at the stop. If the arrival time of next bus is later than the departure time of the first bus, the next bus stops to alight and board passengers. But in case of the arrival time of next bus being earlier than the departure time of the first bus, the next bus only stops if there are alighting passengers. This provides the opportunity of overtaking buses at bus stops. Since this not being the universal case of bus bunching, the generalisation may not be appropriate. Additionally, it is noted here that bus bunching is more prevalent for high frequency services in congested networks, which is different from the route modelled. In this route, bunching is not an issue for most of the time, and the assumptions concerning bus bunching are not crucial to the results from the model.

3.6.2 Overlapping bus routes

Whenever there are more than one bus services in a route or part of it, then the concept of overlapping bus route comes into play. If the services are identical, then the headway between the buses can be calculated by considering that all the buses are of the same service. But whenever the services are not identical then other issues arise from that. The passenger at the bus stop will have a choice of taking the bus from any one of the services. The passenger may take the fist bus arriving or may wait for another if it is perceived to be better (e.g. in terms of journey time). This makes the modelling more complicated. Modelling this issue needs origin-destination data of each passenger. Data collection in this way would be very extensive and has been shown to be unnecessary in this research, given the characteristics of the network chosen for study, as described in Chapter 4. However, a simple case of overlapping routes is modelled by considering different services serving the route as identical services.

3.7 Model output

There are a number of parameters required as an output to explore the possible benefit of differential priority at traffic signals. These include:

- The progression of each bus along the route
- Punctuality at bus stops (a measure of variation with actual bus schedule)
- Total journey time
- Number of passengers boarding and alighting at each stop
- Waiting time of passengers
- Total dwell time at bus stops
- Total passenger journey time
- Bus delays savings at signals
- Delays to the other traffic

These outputs have been used to evaluate the performance of the different priority strategies at traffic signals as described in Chapter 7.

3.8 Summary

The main modelling components of a simple bus priority at traffic signal are origin, route links, bus stops, traffic signals, bus priority and AVL systems. Modelling buses includes the issue of generation and movement. This generation is carried out according to a distribution obtained from field data collection. The modelling of bus movement along the route is based on average link journey time. Bus stops are important components responsible for generating passengers. Issues include passenger generation, their waiting time and the time required for boarding and alighting. Traffic signals are modelled as fixed time signals with proper modelling of signal timings and delay calculation. The bus priority modelling will be based on differential priority. Modelling the AVL system at the beginning is concentrated on the modelling of a beacon-based system. Other issues like bus bunching and overlapping routes are modelled at the final stage of the model. The main outputs generated from the model are total journey time, delays at intersections, number of passengers boarding and alighting, their waiting time, bus occupancy, and delays to the other traffic. These parameters will be used to evaluate the performance of different priority strategies.

Chapter Four

Data Collection

4.1 Introduction

The modelling methodology (Chapter 3) described the methodology to be adopted while modelling a bus route for this research. It identified and described the main modelling components of a bus route. This chapter describes the data collection process to define the components needed to build a model. This data is required in order to build a model to utilise the methodology and to show the characteristics of the chosen route. All the data collected to build a model of the chosen route is called the model building data. There is also other data required in order to validate the model in field conditions. The data and their collection methods are described in detail in Chapter 6 (Model Validation). This chapter concentrates on the collection of model building data.

This chapter starts by stating the data required (section 4.2) from the field. It then describes the methodology/ survey plan (section 4.3) explaining where and how the data has been collected. It then describes the main survey (section 4.4) carried out for data collection. The next section describes the data checks (section 4.5) to find possible errors in the collection process and to refine the data (section 4.6) to make it suitable for use in the model. This chapter is summarised in the chapter summary (section 4.7).

4.2 Data Requirement

Various data is needed to define and describe the different components of a bus priority model. These data are to be collected from the field in order to make a valid model for the chosen site. The required data, to be collected from the field, are grouped according to their related components.

4.2.1 Route data

This data is required to define the characteristics of the links of the route. The data includes the number of links, start and end of links, link lengths, number of lanes in each link, width of lanes, number of bus stops and number of junctions.

4.2.2 Bus data

The bus data describes the characteristics of the buses in the system. The data includes the time of bus generation, service number, origin, destination, type of the bus, passenger capacity, bus patronage at the start of the network and average link speed.

4.2.3 Bus stop data

This data defines the characteristics of the bus stops. The data includes the position of bus stops, bus services, the timetable of different bus services, the starting time of passenger arrivals, rate of passengers boarding, percentage of passengers alighting and dwell time parameters.

4.2.4 Traffic signal data

The traffic signal data is there to represent the characteristics of the traffic signals. This data includes the position of traffic signals, starting time of signal, total cycle time, green time, amber time, number of arms, arm flow rate, arm delays, saturation flows and amount of allowable priority etc.

The next section describes the selection of the site and details the methods to be adopted for the collection of this data.

4.3 Data collection planning

The first part of this section describes the site selection for the collection of data and is followed by describing the method of data collection.

4.3.1 Site selection

The site selected for the data collection was the Portswood corridor bus route in Southampton. The location of the route in the map of Southampton is shown in Figure 4.1.

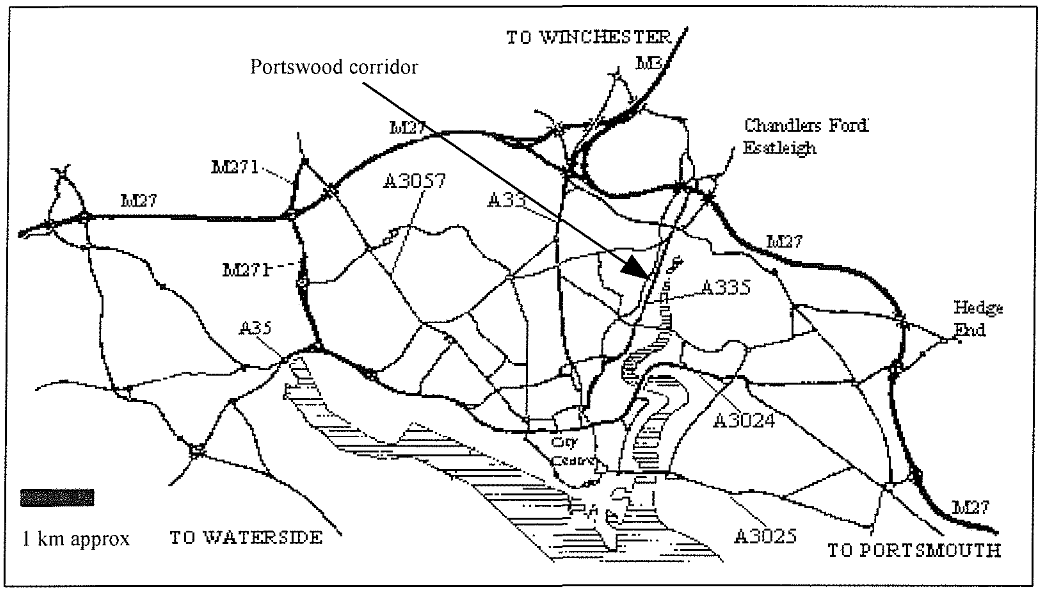


Figure 4.1: Portswood Corridor in Southampton

The bus route extends from Swaythling junction to the city centre. The route is 4.32 kilometres long and has 16 bus stops and 11 signalised junctions. The location of the bus stops and signalised junctions on the route are shown in Figure 4.2. This route is served by 3 main bus services that are divided into 5 different services. These bus services are 11/11A, 3/3A and 101. Only one service runs throughout the route whereas the remaining four enter and/or exit within the route. The total frequency of the services at any bus stop on the route is 6 buses per hour or more. For this research, traffic has been modelled in only one direction towards City Centre. The route being easily accessible from the university, has attracted earlier research also. The earlier study of the route provided valuable preliminary information about the route along with an opportunity to compare results. The route having a number of bus stops, traffic signals and bus services within a reasonable distance provided a chance of studying a variety of bus operation aspects such as a simple case of overlapping routes.

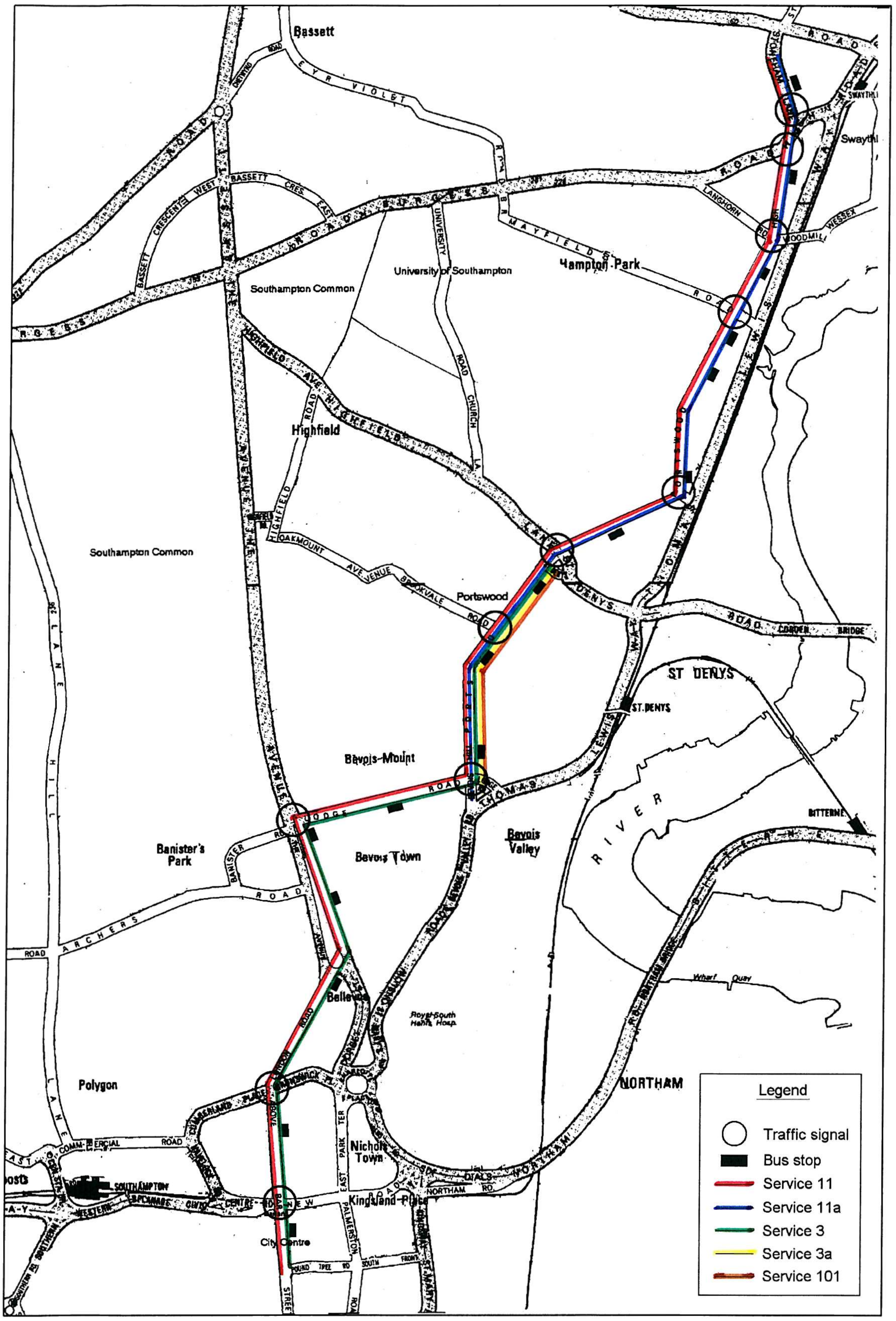


Figure 4.2: Details of Portswood Corridor bus route in Southampton

The survey time selected was between 10:00:00 and 12:00:00. This time interval is after the morning peak. This period was chosen due to the inability of SIMBOL at its development stage in this PhD research to cater adequately for oversaturated junctions. During peak hours, some of the junctions get oversaturated and queues build up that cannot be discharged in a single cycle. This is different from the present assumption of signal modelling in SIMBOL that all the traffic generated are discharged in a single cycle under normal operations (see Section 3.4.2).

4.3.2 Data collection method

At the beginning of data collection process, all the preliminary data required to build a basic model of the bus route were collected together. This data included route details such as number of links, length of links, number of lanes, width of lanes, number of bus stops and their positions in the route, number of traffic signals and their positions in the route, number of bus services and their starting and ending points within the route, timetable of bus services etc. This data was collected at the beginning part of the data collection process. Table 4.1 shows the route data collected as part of this process.

Table 4.1 : Swaythling - Portswood - Lodge Road - City Route

S.N.	Type	Name	Distance	Cum. Dist	Link No.	Bus stop No.	Signal No.
1	Bus stop	Stoneham road	0.0	0.0	0	0	
2	Traffic signal	Stoneham road junction	50.0	50.0	1		1
3	Traffic signal	Swaythling junction	50.0	100.0	2		2
4	Bus stop	McDonald restaurant	50.0	150.0	3	1	
5	Traffic signal	Woodmill road junction	250.0	400.0	4		3
6	Bus stop	Woodmill road	40.0	440.0	5	2	
7	Traffic signal	Mayfield road junction	230.0	670.0	6		4
8	Bus stop	Mayfield road	50.0	720.0	7	3	
9	Bus stop	Sirdar road	180.0	900.0	8	4	
10	Bus stop	Somerset road	350.0	1250.0	9	5	
11	Traffic signal	Talking head junction	40.0	1290.0	10		5
12	Bus stop	Bus depot	320.0	1610.0	11	6	
13	Traffic signal	Highfield lane junction	100.0	1710.0	12		6
14	Bus stop	Somerfield	90.0	1800.0	13	7	
15	Traffic signal	Brookvale road junction	160.0	1960.0	14		7
16	Bus stop	Safeway	100.0	2060.0	15	8	

17	Bus stop	Spring crescent	350.0	2410.0	16	9	
18	Traffic signal	Lodge road junction	30.0	2440.0	17		8
19	Bus stop	Cedar road	180.0	2620.0	18	10	
20	Traffic signal	Stage gate junction	330.0	2950.0	19		9
21	Bus stop	Stage gate	40.0	2990.0	20	11	
22	Bus stop	Middle street	220.0	3210.0	21	12	
23	Bus stop	Law court	280.0	3490.0	22	13	
24	Traffic signal	Cumberland place jn	380.0	3870.0	23		10
25	Bus stop	Cenotaph	140.0	4010.0	24	14	
26	Traffic signal	New road junction	220.0	4230.0	25		11
27	Bus stop	Marland center	90.0	4320.0	26	15	

The other remaining data needed was the data describing the characteristics of buses, bus stops and traffic signals. This data included bus data (average link speed of buses), bus stop data (passenger boarding and alighting rate at each bus stop, dwell time parameters, starting time of passenger arrival) and signal data (signal timings, flows at each arms of a junction, delays associated with general traffic). This data is changeable according to time and hence proper planning was required to gain suitable representative data. From the method of collection, this data was grouped into two parts as 'bus journey data' including bus and bus stop data and 'junction data' which includes signal timings, flows and delays. 'Bus journey data' was collected by travelling on board a bus serving the route in that 2-hour period and collecting operational data using a palmtop computer (PTC). 'Junction data' was collected automatically using the SCOOT loop detectors available in the field and other information about signal timings.

Pilot surveys were carried out to check the preliminary data collected and assess the survey plan before carrying out the main survey. A pilot survey was carried out to collect 'bus journey data' by boarding a bus, with a second pilot survey to collect bus arrival data at a representative bus stop (in Portswood). These surveys provided an opportunity to become more aware of the route. The following interesting observations were made during the survey at Portswood :

- Passengers had a choice of bus services (11/11A, 3/3A or 101). Passengers waiting for other services such as 1 and 20A did not take the above services which may be due to a different operator and/or route. Again, very few passengers boarded the 48 service (serving intercity service). Hence, it was decided to model only bus services 11/11A, 3/3A and 101 serving the Portswood corridor as overlapping routes.

- Some of the early buses were found to be waiting to match their timetable shown at the bus stop. To explore the impact of this process, it was decided to make a provision in SIMBOL for holding buses at bus stops to match their timetable.

These observations were taken into consideration while collecting the main data and were concentrated on collecting data from only those 5 services. The next section describes how the main survey was carried out.

4.4 Main survey

As discussed in section 4.3.2, the survey was carried out in two different parts. 'Bus journey time' data was collected on board using a PTC and 'junction data' was collected automatically. The details of these different surveys are explained in the subsequent sub-sections.

4.4.1 Bus journey data

This 'bus journey survey' was aimed at collecting detailed data of the bus and bus stop in order to build a model to represent the field condition on the Portswood corridor. The survey was carried out between 12/11/2001-30/11/2001 during the period 10:00:00 - 12:00:00. The survey was carried out by noting down the arrival and departure time (in real time) of buses at every signal and bus stop using a handheld computer (Hewlett Packard 1600T). All 30 buses (of service 11/11A, 3/3A and 101) serving the corridor in the specified period between 10:00:00 - 12:00:00 were boarded during this period of 3 weeks. Appendix A (Table A1) shows the schedule of the survey carried out between 12/11/2001 and 30/11/2001 in the different buses. The survey collected the data of the bus services on an inbound direction towards the City Centre.

The survey collected data regarding bus passenger numbers and bus journey time (including stoppage) data for the full length of the route between Swaythling and the City centre. The arrival and departure time (in real time) of buses at every signal and bus stop, along with the number of passengers alighting and boarding at every bus stop, was collected from the survey. This enabled the average link speed of a bus, delay at traffic signals, dwell time at bus stops, boarding and alighting passenger rates and dwell time parameters (refer section 3.4.1 modelling methodology) to be calculated.

To ensure there was no error in the survey timing, the time in the palmtop was checked with British standard time every morning. The collected data was also transferred into the table format developed to ensure no loss of valuable data. A sample of data collected manually inside a bus with the help of a palmtop and its transfer of data onto paper is shown in Appendix A (Table A2 and Table A3). While carrying out the survey, the bus was boarded at least one stop earlier and alighted one stop later than the bus stop where the data of bus arrival and departure was needed. This made sure that the counting of passengers already inside the bus, the number of passengers alighting and the passengers boarding was accurate. Again, this also made sure that the person collecting the data did not influence the dwell time of buses at bus stops recorded during the survey process.

4.4.2 Junction data

Junction data from all 11 junctions on the route was collected automatically at the ROMANSE centre via SCOOT loops. The data collection was carried out for a week on 05/11/2001-09/11/2001 between 10:00:00 - 12:00:00. This timing was matched with the weeklong validation data (described in Section 6.4) collection date. Since the model uses average values while defining junction characteristics, there was no problem in collecting bus journey data and junction data on separate dates (i.e. the interest here is in obtaining reasonably representative data values rather than needing a strict validation). The data collected for each junction included the green and intergreen time for every stage, delay data for 5-minute intervals and link flows data for every 5 minute interval. The 'signal timing data' data file contained data of the different stages of a signal including total period, green time and intergreen time of each stage. The 'link flows data' data file contained the data regarding flows on each arm at 5-minute interval. The third 'link delay data' data file contained delay data per unit time interval. This data needed to be checked before use in the model to make sure that the data was free of errors.

4.5 Data Checks

The collected data was checked to make sure that it was error free. Precautions were taken to minimise error while collecting bus journey data manually. This included the recording of entries in both PTC as well as in the survey form designed during the survey period. The collected data was entered into an Excel format and checked to make sure of the error free transfer of data. Table 4.2 shows an example of the final stage of 'bus journey data' collected.

Table 4.2: An example of final stage of bus journey data collected

Date : 19/11/02 Time : 11:23 Bus type : WD Pass at start : 15									
Type	Name	Dist (m)	Arrival time (hh:mm:ss)	Alight (no.)	Board (no.)	Depart time (hh:mm:ss)	Dwell time (hh:mm:ss)	Signal delay (hh:mm:ss)	Journey time (hh:mm:ss)
Bus stop	Somerfield	1800.0	11:21:03	3	2	11:22:58	0:01:55	-	0:00:00
Signal	Brookvale rd	1960.0	11:23:47	-	-	11:24:35	-	0:00:48	0:00:49
Bus stop	Safeway	2060.0	11:24:58	1	0	11:25:00	0:00:02	-	0:00:23
Bus stop	Spring crescent	2410.0	11:25:42	0	2	11:26:03	0:00:21	-	0:00:42
Signal	Lodge road Jn	2440.0	11:26:21	-	-	11:26:52	-	0:00:31	0:00:18
Bus stop	Cedar road	2620.0	11:27:42	1	0	11:27:45	0:00:03	-	0:00:50
Signal	Stage gate Jn	2950.0	11:28:43	-	-	11:29:00	-	0:00:17	0:00:58
Bus stop	Stage gate	2990.0	11:29:16	0	4	11:29:47	0:00:31	-	0:00:16
Bus stop	Middle street	3210.0	11:30:12	0	0	11:30:12	0:00:00	-	0:00:25
Bus stop	Law court	3490.0	11:30:41	3	0	11:30:55	0:00:14	-	0:00:29
Signal	Cumberland Pl	3870.0	11:32:25	-	-	11:32:25	-	0:00:00	0:01:30
Bus stop	Cenotaph	4010.0	11:32:46	2	0	11:32:51	0:00:05	-	0:00:21
Signal	New road Jn	4230.0	11:33:45	-	-	11:34:35	-	0:00:50	0:00:54
Bus stop	Marland (City)	4320.0	11:35:03	12	0	11:35:26	0:00:23	-	0:00:28
Total				22	8				0:14:00

The junction data collected automatically from SCOOT loops was vulnerable for error. Due to its size and the automatic method adopted to collect it, there was a possibility of error in the data. So, this data was carefully checked to find possible errors while calculating averages to be used in the model. This data collected from SCOOT loops was in its standard format and hence proper formatting was needed in order to calculate the average values. Due to the large amount of data, a Macro was written in Excel to speed up the process.

In the checking process, a large amount of data was found to be missing from the delay data collected on 6th November. The data was collected for only 30 minutes out of the targeted 2-hour period. The data for this day was not appropriate and hence discarded while taking the average of delays. Again, it is to be noted that there was no detector to collect delay and flow data on one arm of traffic signal no.7 (Safeway junction). This arm, serving only car park of the Portwood shopping centre, is a minor arm and does not play an important role.

Again, flow data of some of the arms were missed in some of the days. Missed data were carefully noticed while calculating average values. In this process the flow data of 2nd day i.e. 06/11/2001 was found to be of shorter period. Hence the average value of flow data for that day is based upon the shorter period of time. Beside that, the average flow of side arm of Mayfield junction was calculated using 2 days' data because of missing data for other 3 days. In other cases, the average flows were calculated using 5-day data. The day-to-day variability of the flows during the week was very small. The average values of mean and standard deviation of flows in the different arms of junction were calculated as 338.66 veh/hr and 15.87 veh/hr. There was not much difference in using the average flows calculated from 4 days' (similar to delay data) or 5 days' (all available data) data. The difference between these two average flows was only 1.48 veh/hr (0.37%). So, the average flows were calculated from all available 5-day data.

It is also to be noted that the flow and delay data obtained from SCOOT are converted from Link Profile Unit (LPU) using a link specific conversion factor. Since the conversion depends upon the vehicle gaps and number of lanes, using a fixed conversion factor may involve some error in absolute values of these data. However, the results are being used here for comparative purposes (of strategies) rather than for interpretation in terms of absolute values. Errors in absolute values therefore do not play any significant role in the outcome of the research.

4.6 Data Refinement

After checking the collected data, it was refined for input into the model. The junction data collected was in a different file for different parameters such as signal timing, arm flows and arm delays. However, in case of bus journey time, many parameters describing bus and bus stop characteristics were to be obtained from the single data collected format. The different types of data deduced from the survey are explained in the subsequent sub-sections.

4.6.1 Junction data

All the junction data was collected automatically at the ROMANSE centre for one week. This data collected using SCOOT loops was very bulky and needed refinement in order to use in SIMBOL. The refinement of the data collected is described separately below.

4.6.1.1 Signal timing data

The signal timing data was extracted from the automatic collection of signal timing data (the SCOOT m37 message). The signal data collected from SCOOT gave the signal data for every signal at every stage. The data contained information such as time of the end of each stage (hour:minute:sec) format, the junction number, stage number, intergreen time, green time and total stage length. An example of m37 data in its raw state is given in Appendix A (Table A4).

The data needed to be sorted according to the junction and then the stage. Then the green time and intergreen times of all stages were grouped together to calculate the total cycle time. All these stage times and cycle times were then averaged to get average stage lengths and cycle time for that period of the day. These average stage lengths and cycle times of each day were also averaged to get final average signal timings. This procedure was adopted for all the traffic signals on the route. The data was also used to calculate the starting time of each signal to represent their time in the simulation model. The start of the stage time was calculated by deducting the stage length from the end of the stage time. The start time was taken as the starting time of the last cycle just before 10:00 o'clock. These starting times of each signal were then averaged to obtain the starting times of each signal. Table 4.3 shows the average stage lengths of traffic signals along with their starting time obtained from the collected data.

Table 4.3 : Stage lengths and start time of traffic signals along Portswood corridor

S.N.	Traffic signal name	Stage lengths (seconds)								Cycle time (secs)	Start time (hh:mm:ss)
		1		2		3		4			
		green	IG	green	IG	green	IG	green	IG		
1	Stoneham Rd	71	9	0	0	5	4	12	7	108	9:59:01
2	Swaythling	48	6	1	0	2	5	28	15	104	9:59:27
3	Woodmill Rd	30	7	34	8	20	7	1	1	108	9:58:51
4	Mayfield Rd	21	7	8	7	0	0	0	0	43	9:59:39
5	Talking head	45	6	9	7	0	0	0	0	67	9:59:24
6	Highfield Rd	18	7	2	2	17	10	11	8	75	9:59:40
7	Brookvale Rd	28	12	3	3	17	6	7	6	82	9:59:15
8	Lodge Rd	31	8	13	8	2	4	0	0	67	9:59:31
9	Stage gate	30	9	11	6	22	6	0	0	84	9:58:57
10	Cumberland P	44	13	16	9	11	10	0	0	103	9:59:01
11	New Road	19	15	9	9	16	15	8	9	100	9:58:27

4.6.1.2 Flow data

Link flows were obtained from the detector flow data (the SCOOT u07 message). Flow data of each arm from all the signals collected from the SCOOT were recorded for every 5-minute (300 secs) interval. The data file contained speed and flow data from each detector for 5-minute intervals. The data was collected for a whole 24-hour period for that week, making the collection process easier. While taking the average of the data, it was difficult to import it into Excel due to the number of rows needed being bigger than Excel's capacity. A program called "U07Prog" developed in TRG was used to import the data between 9:45 to 12:30 for those 11 junctions. An example of flow data (u07) in its raw stage is given in Appendix A (Table A5). The data was transferred into Excel format and then sorted according to junction and SCOOT loop number. The average value for each loop was then calculated for the period from 10:00 to 12:30 for a day. These average values were again used to calculate a single average value for 5 days of the week. These values are shown in Table 4.4.

4.6.1.3 Delay data

Delay data was obtained from detector delay data (the SCOOT m02 message) processed from SCOOT loops. The data file contained delay data at each detector per unit time for every 5 minute interval. An example of delay data (m02) in its raw stage is given in Appendix A (Table A6). The data was then sorted according to junction and SCOOT loop number and the average value for each loop was calculated for a day. These values were then again averaged to get a single average value for the week. Due to missing data, the average was taken using data from 4 days only instead of 5 days (similar to flow data). This might have caused mismatching of the data if the flows varied widely during the survey period. However, the day-to-day variation of flows was very small (see Section 4.5) and these average values were just needed to build a reasonably representative model; so there was no problem in using such data. Table 4.4 shows the average flow and delay data from the main and side arms for signals in the direction of the city centre.

Table 4.4 : Average flows and delays of junction arms in the direction of the city

S.N.	Junction name	Junction No.	Bus route road		Left side road	
			Av Flows (veh/hr)	Av Delays (sec/veh)	Av Flows (veh/hr)	Av Delays (sec/veh)
1	Stoneham Rd	5121	263	38.27	711	2.78
2	Swaythling	5131	293	5.89	-	
3	Woodmill Rd	5111	311	31.98	308	22.21
4	Mayfield Rd	5142	336	9.00	69	15.50
5	Talking head	6252	339	4.81	89	56.42
6	Highfield Rd	6121	289	26.33	403	36.07
7	Brookvale Rd	6111	501	22.19		
8	Lodge Rd	6211	502	-	293	15.34
9	Stage gate	4141	439	39.69	-	-
10	Cumberland Pl	7331	429	51.77	671	20.03
11	New Road	7411	122	38.23	349	45.94

In the table, '-' shows that there is no side road in the modelled direction for that junction. Blank spaces means that there was no data for those arms. The arm flow for the side road of junction 7 leading towards the car park was assumed to be 120 vehicles per hour.

4.6.2 Bus journey data

The bus journey data collected was the main source of model building data. The collected data was then refined to get values of average bus speed for each link, starting times of each bus stop, number of passengers boarding and alighting and dwell time parameters. These refined data to be used in SIMBOL are described in the following subsequent sub-sections.

4.6.2.1 Average link speed

The average time taken by buses to travel a link, collected during the bus journey time data, was used to calculate the average link speed of buses on each link. A link is a section of a bus route between two consecutive points where a bus is very likely to stop (e.g. bus stops and traffic signals). The link time of a bus at each link was calculated as the difference between the arrival time of the bus at a link and the departure time of the bus from the last link. This link time excluded any stoppage time at a bus stop or a traffic signal. The link times of all the buses were then averaged to get an average link time. This average link time was used to calculate the average link speed for the known distance of the link. The 'space mean speed' hence calculated was then used in the model. Table 4.5 shows the average link time calculated to use in the model.

Table 4.5: Average link speed of buses along the bus route

S.N.	Type	Name	Distance (m)	Average time	Speed (km/hr)
1	Bus stop	Stoneham road	0	-	-
2	Traffic signal	Stoneham road junction	50	0:00:11	16.74
3	Traffic signal	Swaythling junction	50	0:00:16	11.37
4	Bus stop	McDonald restaurant	50	0:00:17	10.80
5	Traffic signal	Woodmill road junction	250	0:00:29	30.95
6	Bus stop	Woodmill road	40	0:00:18	7.89
7	Traffic signal	Mayfield road junction	230	0:00:21	39.74
8	Bus stop	Mayfield road	50	0:00:10	18.78

9	Bus stop	Sirdar road	180	0:00:26	25.17
10	Bus stop	Somerset road	350	0:01:09	18.28
11	Traffic signal	Talking head junction	40	0:00:05	26.58
12	Bus stop	Bus depot	320	0:00:36	32.30
13	Traffic signal	Highfield lane junction	100	0:00:17	21.71
14	Bus stop	Somerfield	90	0:00:34	9.53
15	Traffic signal	Brookvale road junction	160	0:00:41	13.96
16	Bus stop	Safeway	100	0:00:23	15.85
17	Bus stop	Spring crescent	350	0:00:40	31.25
18	Traffic signal	Lodge road junction	30	0:00:15	7.32
19	Bus stop	Cedar road	180	0:00:45	14.27
20	Traffic signal	Stage gate junction	330	0:00:42	28.12
21	Bus stop	Stage gate	40	0:00:15	9.93
22	Bus stop	Middle street	220	0:00:23	34.43
23	Bus stop	Law court	280	0:00:47	21.64
24	Traffic signal	Cumberland place jn	380	0:01:22	16.70
25	Bus stop	Cenotaph	140	0:00:23	21.60
26	Traffic signal	New road junction	220	0:00:44	17.90
27	Bus stop	Marland centre	90	0:00:30	10.68

4.6.2.2 Boarding and alighting passenger rate

The number of passengers alighting and boarding at each bus stop were obtained from the collected data from all 30 buses. These were then used to calculate a boarding passenger rate at every bus stop assuming that the passengers arrive randomly. The assumption was based on a recent study carried out in the same route showing random passenger arrival (Rajbhandari, 2002). This means a uniform rate of arrival of passengers can be used to estimate the rate of passenger arrival at a bus stop. However, the fixed rate may not be valid for all period during a day. The rate may be considerably higher in peak period than off-peak making the dwell time of a bus considerably different. However, since the total modelling period in this case is within off peak period, the variation in the rate is not considerable. Additionally, some random variation in passenger arrival within the period would have been smoothened in longer run to get an average rate. Hence a uniform rate is used in the model to estimate boarding passengers. The uniform rate was calculated by summing up the total number of passengers arriving at each bus stop and then diving the sum by the period (2-hr i.e. 7200 seconds). This arrival rate was presented in the form of time gaps in order to generate a passenger for the model input.

Rate of passenger boarding = 7200/total passenger number   sec per passenger

In the case of alighting passengers, the rate was expressed in terms of a percentage of passengers already inside the bus. For this purpose, the total numbers of passengers alighting at a bus stop and the total number of passengers inside the bus were summed for 2-hour period. The ratio of alighting passengers to total numbers inside was then calculated.

$$Ratio\ of\ passenger\ alighting = \frac{Numbers\ of\ alighting\ passenger}{Total\ numbers\ of\ passenger\ inside}$$

The ratio of passengers alighting and the rate of passengers boarding, calculated from the survey data for all the bus stops of the Portswood route, are shown in Table 4.6.

Table 4.6 : Alighting and boarding passenger rates obtained from 2hour period

S.N.	Name	Alighting Pass (nos.)	Boarding Pass (nos.)	Inside Pass (nos.)	Alighting ratio	Boarding rate time gaps (sec/pass)
1	Stoneham road	4	40	114	0.035	180.0
2	McDonald Restaurant	2	50	150	0.013	144.0
3	Woodmill road	1	14	198	0.005	514.3
4	Mayfield road	1	28	211	0.005	257.1
5	Sirdar road	1	15	238	0.004	480.0
6	Somerset road	0	13	252	0.000	553.8
7	Bus depot	1	5	265	0.004	1440.0
8	Somerfield	182	103	596	0.305	69.9
9	Safeway	25	51	517	0.048	141.2
10	Spring crescent	4	21	543	0.007	342.9
11	Cedar road	4	8	262	0.015	900.0
12	Stage gate	5	20	266	0.019	360.0
13	Middle street	8	2	281	0.028	3600.0
14	Law court	31	3	275	0.113	2400.0
15	Cenotaph	58	1	247	0.235	7200.0
16	Marland (City centre)	154	0	190	0.811	1440.0
	Total passengers	481	374			

The unequal number of passenger boarding and alighting are due to the number of passengers already inside the bus at the beginning of the modelled route. The actual route starts upstream of the starting bus stop (i.e. Swaythling and Portswood) of the modelled route.

4.6.2.3 Dwell time parameters

Dwell time parameters are necessary to estimate the dwell time of buses (see section 3.4.1) at a bus stop calculated from the number of passengers alighting and boarding at any bus stop. Since the bus journey time survey had collected the dwell time of buses at all bus stops along with the number of passengers alighting and boarding, it was possible to obtain dwell time parameters from the data collected. The data collected showed that there was a clear distinction between the type of buses used in different services. Most of the buses (85%) serving the route 11/11A and 101 were double decker buses whereas most of the buses serving 3/3A were single deck low floor buses. Hence two separate dwell time parameters were calculated to use in the model for different services. A similar method was adopted by York (1993) while calculating dwell time parameters in a London study. Multiple linear regression was used to calculate the coefficients of the dwell time equation discussed in Section 3.3.3. The dwell time expressions for two types of buses obtained from the analysis are:

$$T = 6.85+1.69*a+9.00*b, \textit{ for double deck buses (11/11a/101services);}$$

$$T = 3.30+1.96*a+9.04*b, \textit{ for low floor buses (3/3a services);}$$

These relationships give total dwell time '*T*' in seconds in terms of number of alighting passengers '*a*' and number of boarding passengers '*b*'. The R^2 values calculated for the equations were 0.67 and 0.92 respectively. The first value of R^2 shows a reasonably acceptable fit whereas the second one shows a good fit of the equation to the collected data. The t-ratios of the coefficient of input variables of the first equation were 5.01 and 15.10. These significantly higher values than theoretical t-values show that there is a big influence of alighting and boarding passengers in total dwell time. Similarly the t-ratios of the coefficient of input variables the second equation were 5.08 and 13.42. These values also being significantly higher than the theoretical t-values, the influence of the input variables was significant in case of second equation too. Furthermore, the equations were found to be very close to those obtained by York (1993), as illustrated in Table 4.7.

Table 4.7: Dwell time parameters from field data and York (1993)

Bus type	Data type	Dwell time parameters		
		D	**A**	**B**
Double deck	Field	6.85	1.69	9.00
	York	5.42	1.48	9.15
Low floor	Field	3.30	1.96	9.04
	York	3.55	1.99	9.18

These both sets of values were then statistically tested using t-test for paired data. The test showed that the difference between the field parameters and York's parameters was found to be not significant at the 5 percent level. This confirmed the compatibility of the field parameters with York's parameters.

4.6.2.4 Starting time of passenger generation

The starting time of passenger generation is important for the proper estimation of the passenger numbers for the first bus arriving at each bus stop. A wrong estimation of this start of passenger generation time affects the dwell time of the first few buses. If passengers are generated at all the bus stops at the same time, then the first bus reaching the bus stop will have to carry a lot of passengers which is not realistic. Here the starting time is calculated by using the average journey time taken by the buses of the first service to arrive at a different bus stop. The starting time of the first bus stop was based on the scheduled time interval and average deviation of buses for the service. Then the start time for the rest of the bus stops was calculated by adding average departure time of buses from those bus stops to that start time at the first bus stop.

Since the first bus to arrive at all the bus stops just before the start of simulation period is of service 11, its arrival time at the first bus stop and its average journey time data were used to calculate the starting time of passenger generation at all bus stops. At the first bus stop, the average deviation of buses of service 11 were included in the scheduled time of the bus. After including a deviation of -3 seconds (early) in the scheduled starting time of –540 seconds (9:51 - 9 minutes earlier than 10:00), it became -543 seconds. The starting times for the rest of the bus stops were calculated by using the average travel time of buses of service 11. Table 4.8 shows the starting time of passenger generation at all bus stops for the route.

Table 4.8: Passenger generation start time of bus stops along the route

S.N.	Bus stop	Passenger generation start time (seconds)
1	Swaythling (Stoneham Rd)	-543
2	McDonald restaurant	-437
3	Woodmill road	-340
4	Mayfield road	-294
5	Sirdar road	-241
6	Somerset road	-150
7	Bus depot	-85
8	Portswood (Somerfield)	-8
9	Safeway	139
10	Spring crescent	208
11	Lodge road (Cedar road)	305
12	Stage gate	395
13	Middle street	437
14	Law court	492
15	Cenotaph	645
16	City centre (Marland centre)	761

4.6.2.5 Scheduled timetable

The scheduled timetables of all the bus services along the route were collected from the field. The field timetable was based on the timetable stated in a booklet provided by the bus operator. As there are 16 bus stops instead of the 4 stated in the booklet, the timetable of the bus stops not included in the booklet were found to be the same as that of the last bus stop with a scheduled time in the booklet. The timetable in the field was found to be repeated for all the bus stops until the next bus stop with a scheduled time in the booklet was reached. Table 4.9 shows the field timetable of different bus services displayed at each bus stop along the route.

Table 4.9: Timetable of buses displayed at bus stops along the route

S.N.	Bus stop	Distance	Bus services				
			11	11A	3	3A	101
1	Swaythling (Stoneham rd)	0.0	660	60	-	-	-
2	McDonald restaurant	150.0	660	60	-	-	-
3	Woodmill road	440.0	660	60	-	-	-
4	Mayfield road	720.0	660	60	-	-	-
5	Sirdar road	900.0	660	60	-	-	-
6	Somerset road	1250.0	660	60	-	-	-
7	Bus depot	1610.0	660	60	-	-	-
8	Portswood (Somerfield)	1800.0	1080	480	180	780	1140
9	Safeway	2060.0	1080	480	180	780	1140
10	Spring crescent	2410.0	1080	480	180	780	1140
11	Lodge road (Cedar road)	2620.0	1260	-	360	-	-
12	Stage gate	2990.0	1260	-	360	-	-
13	Middle street	3210.0	1260	-	360	-	-
14	Law court	3490.0	1260	-	360	-	-
15	Cenotaph	4010.0	1920	-	1020	-	-
16	City centre (Marland centre)	4320.0	1920	-	1020	-	-

The times in the table are given in terms of seconds past every hour of the first bus of each service. The same time is given for 7 different bus stops on a 1.6 kilometres route showing the need to refine the timetable to provide more accurate information. The modification of this field timetable is required in order to calculate lateness of buses for determining their priority requirement in the case of differential bus priority strategies.

The modification of the timetable was based on the average journey time of buses and their deviation at the start of the route. The timetable of the buses at the origins was changed taking account of deviation in the starting time. The deviation in the starting time of buses from the field timetable for services 11/11a and 3/3a were found to be 19.5 seconds and 152 seconds. These values were rounded down to 0 and 120 seconds and added to the starting time for 11/11a service and 3/3a service. The timetables for rest of the bus stops were modified depending upon the average journey time of buses. The journey time was calculated as the time between the departure time from a bus stop to the departure time from next on the downstream. Using these modifications, the timetable based on the journey time of buses is shown in Table 4.10. This timetable is used as a base timetable in the model.

Table 4.10: Modified timetable used in the model

S.N.	Name	Distance	11	11A	3	3A	101
1	Swaythling (Stoneham Rd)	0.0	660	60	-	-	-
2	McDonald Restaurant	150.0	780	180	-	-	-
3	Woodmill road	440.0	840	240	-	-	-
4	Mayfield road	720.0	900	300	-	-	-
5	Sirdar road	900.0	960	360	-	-	-
6	Somerset road	1250.0	1020	420	-	-	-
7	Bus depot	1610.0	1080	480	-	-	-
8	Portswood (Somerfield)	1800.0	1200	600	300	900	1140
9	Safeway	2060.0	1320	720	420	1020	1260
10	Spring crescent	2410.0	1380	780	480	1080	1320
11	Lodge road (Cedar road)	2620.0	1440	-	540	-	-
12	Stage gate	2990.0	1500	-	600	-	-
13	Middle street	3210.0	1500	-	600	-	-
14	Law court	3490.0	1560	-	660	-	-
15	Cenotaph	4010.0	1680	-	780	-	-
16	City centre (Marland)	4320.0	1800	-	900	-	-

4.7 Chapter summary

This chapter has described the data collection process involved in this research in detail. The data collection was carried out for the Portswood corridor in Southampton. All the data needed to develop a model of that route was collected. The data was collected by desktop study, manual collection on site and automatic collection using SCOOT data. The collected data was checked for possible errors and then refined into a proper format that was usable for the model. The refined data was then ready to use as input parameters into the model. The development of the simulation model based on the methodology described in Chapter 3 and the data collection described in this chapter is detailed in the next chapter.

Chapter Five

Model Development

5.1 Introduction

This chapter describes the model developed for the purpose of this research. The model has been developed based on the modelling methodology (Chapter 3) and data collection (Chapter 4). The model has been called SIMBOL (SImulation Model for Bus priOrity at traffic signaLs). It is a microscopic simulation model with a fixed time scanning interval of one second. The programming language used in developing the model is C++ (Borland Builder version5). It is a widely popular object oriented programming language (Parsons, 1997). The complete source code of the model is given in Appendix B. While modelling, the model is divided into six different modules. These are main, bus, bus stop, traffic signal, bus priority and GPS module. In the model, the main module takes input, interacts with other modules and produces final output. A simple diagrammatic model of SIMBOL is shown in Figure 5.1.

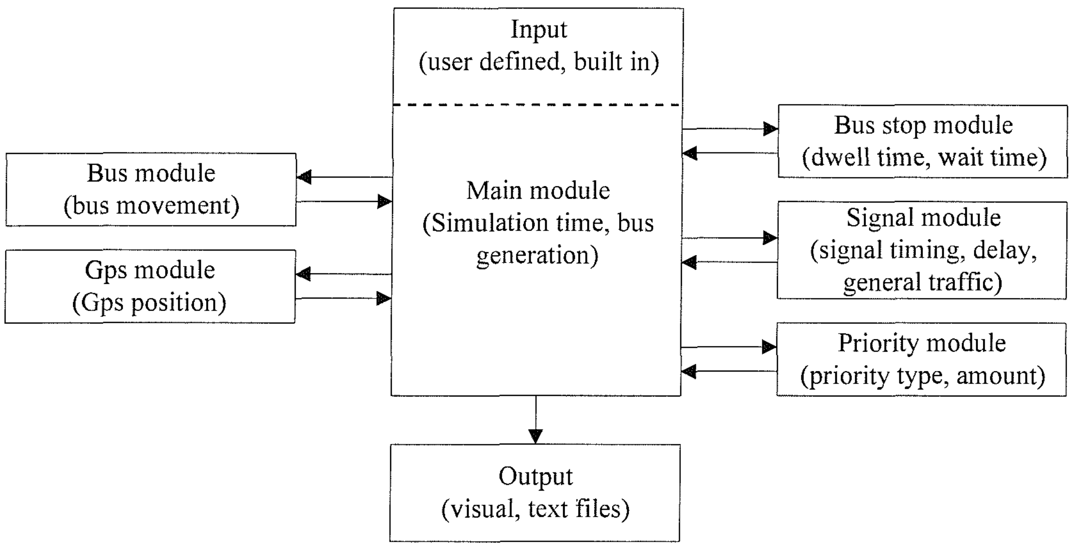


Figure 5.1: Diagrammatic model of SIMBOL

The main module takes input parameters for all the modules designed to perform some specific tasks. The main module then interacts with these modules at every change in simulation time to carry out those tasks. The main module produces visual and text output based on the outcome of the tasks. The working of all these modules are described in the subsequent sections.

5.2 Main module

The main module is responsible for steering the overall simulation by taking input, coordinating the interaction between all other modules and generating output. It is responsible for starting, taking input, generating final output and ending of the simulation model. The input, output and working of the module are described in the following sub-sections.

5.2.1 Input

The main module takes the input such as simulation period, simulation speed and priority strategy while starting a simulation run. These are modelled as user defined input parameters for the simulation model. All other input parameters for buses, bus stops, signals and priority are modelled as built-in input inside this main module. Hence, these input parameters can be modified only before running the simulation model. These input parameters for each of the modules are given in the respective module.

5.2.2 Working of the module

At the start of the simulation, the module passes all the information for the different modules to their respective module. The main module then increments the simulation time every second and updates the condition of all the activities. At every change in simulation time, the module updates signal periods and number of cars in each arm of all signals by interacting with the signal module. This is followed by checking the time to generate a bus according to a predefined generation time. The module then takes every generated bus in turn and updates its actual and GPS position. The updated position of each bus is checked with bus stop, traffic signals or priority modules to find its presence there. The module then visually displays all the activities which occurred during the increment in the simulation period and finally generates concise output at the end. The flowchart showing the working of the module is shown in Figure 5.2.

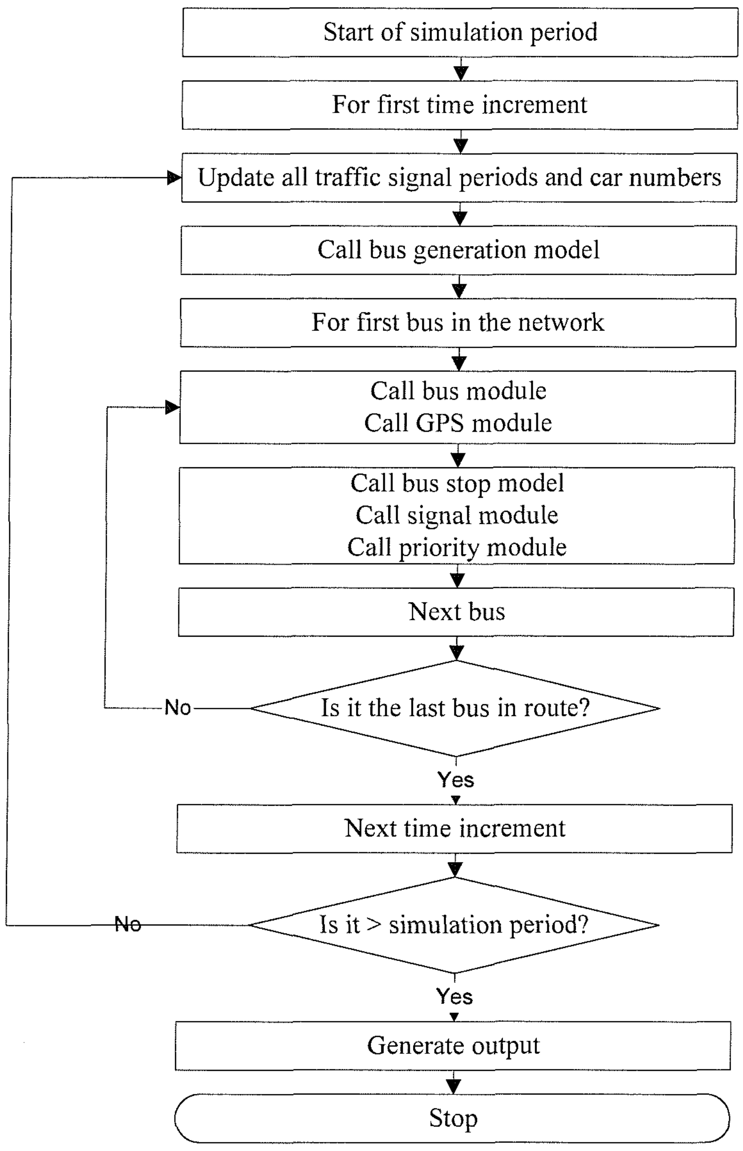

Start of simulation period
For first time increment
Update all traffic signal periods and car numbers
Call bus generation model
For first bus in the network
Call bus module
Call GPS module
Call bus stop model
Call signal module
Call priority module
Next bus
No
Is it the last bus in route?
Yes
Next time increment
No
Is it > simulation period?
Yes
Generate output
Stop

Figure 5.2: Flowchart of main module

Beside the interaction with modules, the main function of the main module is to manage the simulation time and generate buses. These two main functions are described in subsequent sub-sections.

5.2.2.1 Simulation period

The total simulation period is defined by the user. The model updates the simulation time in one second intervals and scans all the activities until the simulation period is completed. Since the buses are generated till the end of the simulation period, those generated towards the end of the simulation may not reach their destination by the end of the simulation period. Hence the simulation is continued to allow all the buses generated to reach their destination but no more buses are generated after the length of simulation period.

5.2.2.2 Bus generation

The module continuously checks whether the simulation time has reached the predefined bus generation times. Once such a time is reached, a bus is generated at the specified bus stop. There are two places for the generation of buses according to the service. Buses of service 11/11a are generated at Swaythling (bus stop 0) and buses of the other services 3/3A/101 at Portwood (bus stop7). Service 11 and 3 continue to the city whereas 11a and 3a deviate from Lodge Road junction. Although different services start and end at different bus stops, they all reach the city with a little deviation on the way. All these services are modelled as identical so that passengers board the first bus to arrive at a bus stop.

5.2.3 Output

The module continuously updates the status of different components in visual output and produces text files at the end.

5.2.3.1 Visual output

The module displays the progress of each component in the system continuously. The output is shown with the reference to a linearly represented route. The positions of all the buses, bus stops and traffic signals are arranged on this linear route. The bus stops are represented by blue rectangles, signals by black rectangle with three coloured circles, cars by small black rectangles and buses by big red rectangles. The positions of buses, signal period and number of cars are visually updated every second. An example of the visual output of a simulation run of SIMBOL is shown in Figure 5.3.

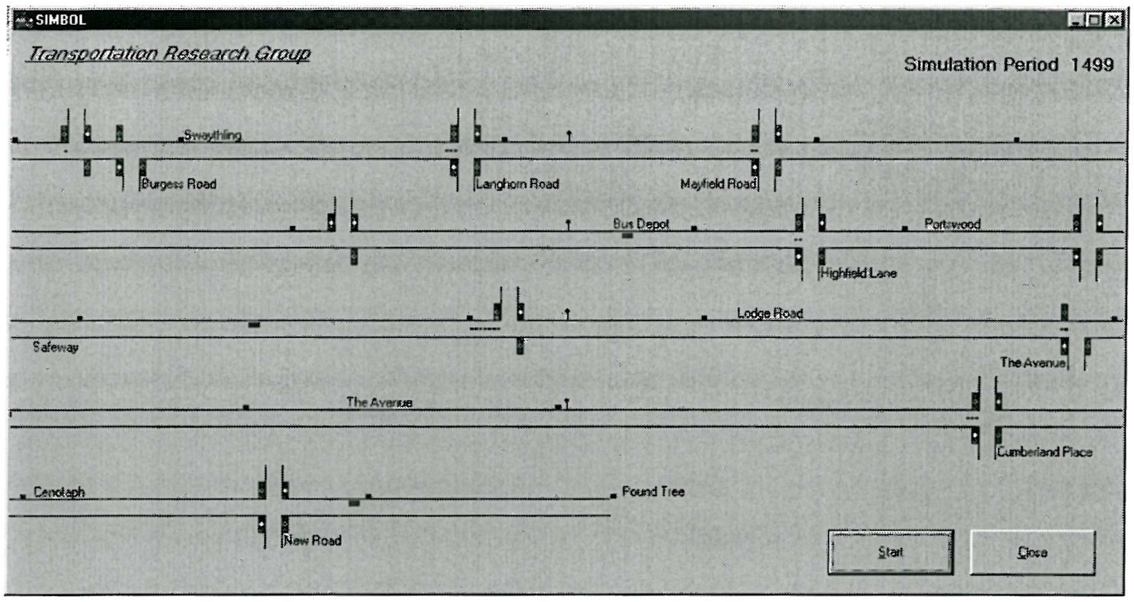


Figure 5.3: An example of visual output from SIMBOL

5.2.3.2 Text output

At the end of the simulation run, the main module generates a concise text file containing total and averages of performances of different modules. The main output parameters are total passenger waiting time, passenger journey time, bus journey time and delays to cars at signals on a per hour basis. The other parameters in the output contain the average number of passengers alighting and boarding at bus stops, bus occupancy, waiting time, car delay per junction and bus delay per junction. The main output parameters are used in the further analysis to evaluate the performance of different bus priority strategies.

5.3 Bus module

This module models the movement of all buses and keeps a record of all of them in the system. The input parameters for this module are identification number, origin, destination, service number and capacity. These are defined at the time of generation by the main module. The bus module then continuously checks the bus speed and updates the position of all buses in the system.

The movement of a bus is modelled by updating its position every second. The new position is obtained by adding the distance travelled in one second to the last position of the bus. The distance travelled is calculated from the present speed of the bus. The present speed of the bus is assigned as the average link speed according to a link if it is moving. When it reaches a bus stop or signal and has to stop, then the speed is changed to zero. At this time, an interaction between the bus and these components (e.g. bus stop, signal) is modelled by the respective module. The flow chart showing the working of the bus module is shown in Figure 5.4.

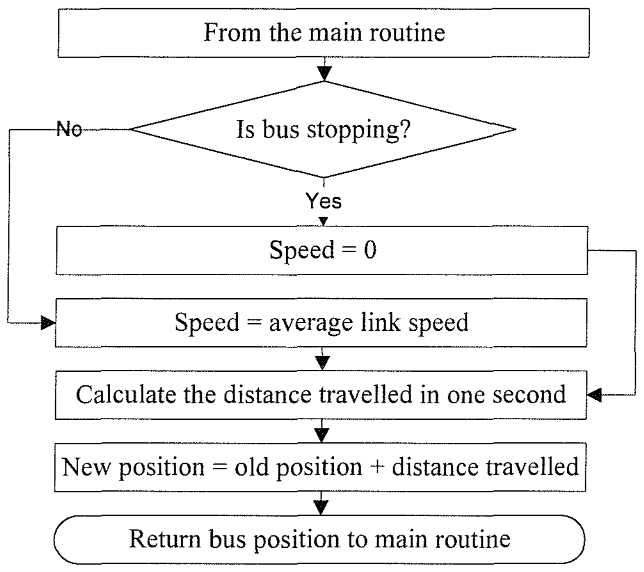

From the main routine
Is bus stopping?
No
Yes
Speed = 0
Speed = average link speed
Calculate the distance travelled in one second
New position = old position + distance travelled
Return bus position to main routine

Figure 5.4: Flowchart of bus module

5.4 Bus stop module

The bus stop module models the interaction between buses and passengers at a bus stop. It mainly estimates the dwell time of a bus from the number of passengers at a bus stop and the waiting time of those passengers and calculates the lateness of the bus at departure. The input parameters required for this module are bus stop identification number, position of the bus stop, percentage of alighting passengers, rate of boarding passengers and the timetable of the different services. At the time of a bus departure, the module produces a detailed output for verification purposes. The output includes the arrival and departure time of a bus, the time gap, number of alighting and boarding passengers, average waiting time of passengers, dwell time of the bus, occupancy of the bus and the deviation of the bus from the schedule. The working of the module is described in the next sub-section.

5.4.1 Working of the module

The main module calls bus stop module to check whether a bus has reached any bus stops. The bus stop module then proceeds according to the flow chart shown in Figure 5.5.

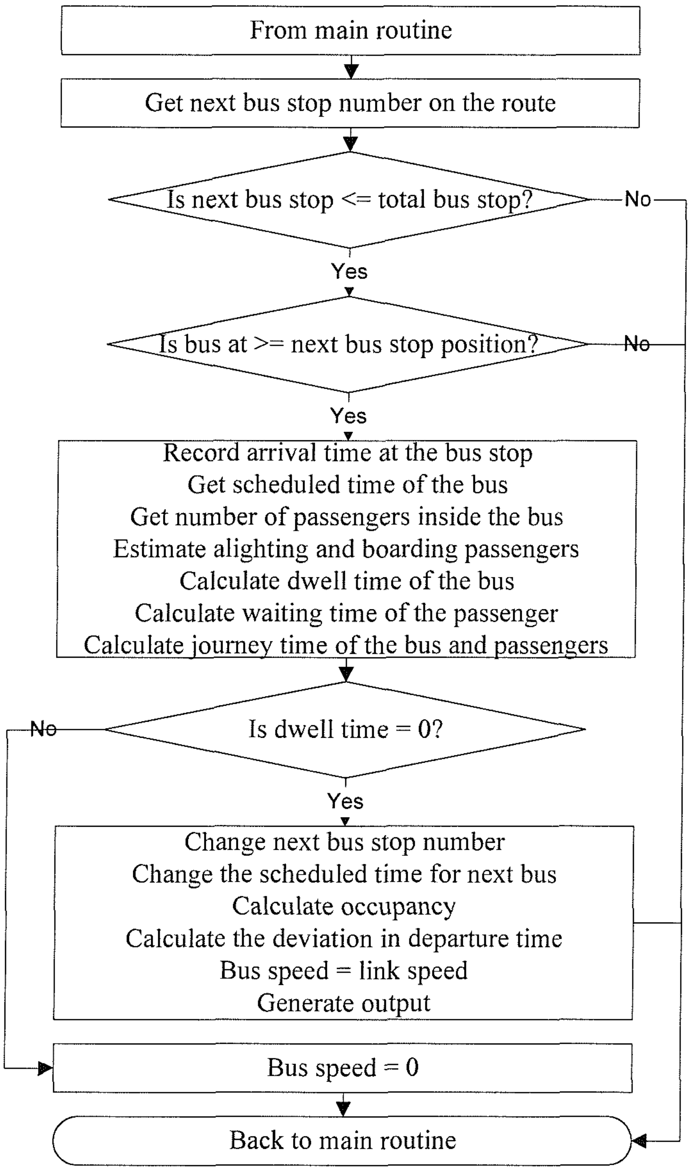

From main routine
Get next bus stop number on the route
Is next bus stop <= total bus stop?
No
Yes
Is bus at >= next bus stop position?
No
Yes
Record arrival time at the bus stop
Get scheduled time of the bus
Get number of passengers inside the bus
Estimate alighting and boarding passengers
Calculate dwell time of the bus
Calculate waiting time of the passenger
Calculate journey time of the bus and passengers
Is dwell time = 0?
No
Yes
Change next bus stop number
Change the scheduled time for next bus
Calculate occupancy
Calculate the deviation in departure time
Bus speed = link speed
Generate output
Bus speed = 0
Back to main routine

Figure 5.5: Flowchart of bus stop module

The module first checks whether a bus has arrived at any of the bus stops. Once a bus reaches a bus stop the bus identification number along with its arrival time, the service number and the number of passengers inside is obtained. With this information, the module calculates the time gap from the earlier bus, the number of alighting and boarding passengers, the total dwell time needed, total passenger waiting time and the departure time. The bus is stopped until the departure time by changing its speed to zero. Once the time reaches the departure time, its speed is changed to the average link speed. At this moment, total passengers inside, the lateness of the bus and the scheduled timetable for the next bus is calculated.

When a bus is stopped at a bus stop, the bus gets a reserved number from the bus stop. If a second bus arrives at the bus stop before the departure of the first, then the second bus will only stop to alight passengers. If the first bus is still there at the end of the alighting time, the second bus departs without boarding any passengers. All the passengers board the first bus even if there are two buses. But, if the first bus departs before the departure of the second bus, the second bus boards the passengers while stopping. In case of 'holding' option, stopping may be for matching the timetable of an early bus. All the passengers generated during this period board the bus. The flowchart showing the detail process once a bus is at a bus stop is shown in Figure 5.6.

The various functions, carried out by the module, are described in following sub-sections as time gap calculation, estimation of passengers, dwell time calculation, waiting time calculation, bus occupancy, journey time and punctuality of buses.

5.4.2 Time gap calculation

The calculation of time gap is the first step taken once a bus arrives at a bus stop. This is used in calculating the number of boarding passengers and their waiting time. While doing the calculation, all the buses serving a bus stop are considered identical. Hence the time gap is the time between the arrival time of a bus and the departure time of the earlier bus. For this purpose, the arrival and departure time of all buses serving a bus stop are noted. With this information, the time gap of every arriving bus is calculated from the departure time of the last bus at that bus stop. The time gap for the first bus arriving at each bus stop is calculated by using the start time of passenger generation calculated from field data (Section 4.6.2.4). This time gap is used to estimate the number of passengers boarding using the rate of boarding passengers.

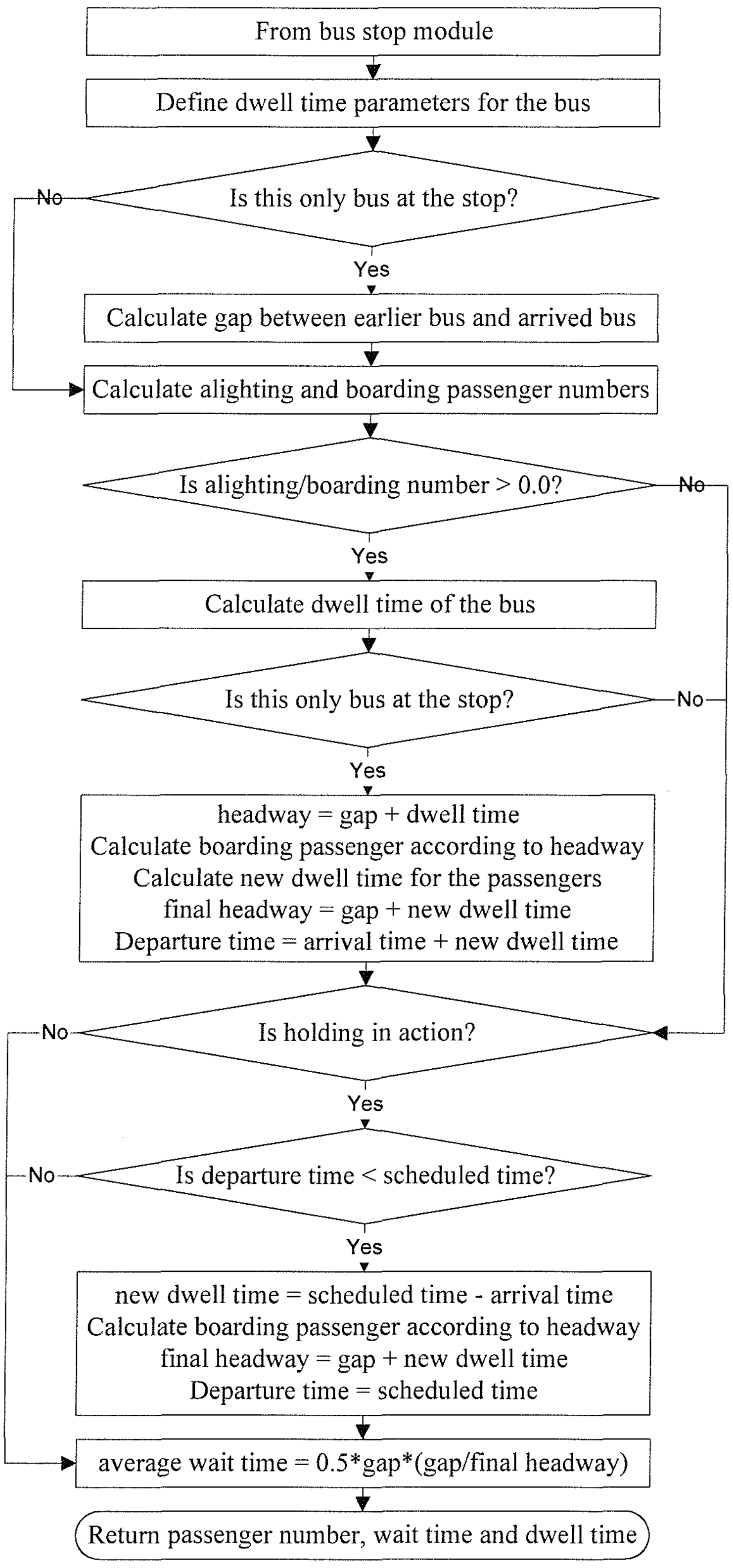

From bus stop module
Define dwell time parameters for the bus
Is this only bus at the stop?
No
Yes
Calculate gap between earlier bus and arrived bus
Calculate alighting and boarding passenger numbers
Is alighting/boarding number > 0.0?
No
Yes
Calculate dwell time of the bus
Is this only bus at the stop?
No
Yes
headway = gap + dwell time
Calculate boarding passenger according to headway
Calculate new dwell time for the passengers
final headway = gap + new dwell time
Departure time = arrival time + new dwell time
Is holding in action?
No
Yes
Is departure time < scheduled time?
No
Yes
new dwell time = scheduled time - arrival time
Calculate boarding passenger according to headway
final headway = gap + new dwell time
Departure time = scheduled time
average wait time = 0.5*gap*(gap/final headway)
Return passenger number, wait time and dwell time

Figure 5.6: Flowchart of detail calculation inside bus stop module

5.4.3 Alighting Passenger estimation

As a bus arrives at a bus stop, the bus stop module obtains the number of passengers inside the bus. Then the number of alighting passengers is estimated by multiplying the total passengers inside by a fixed ratio assigned to that bus stop. The numbers are calculated by following relationship:

$$no.\ of\ alighting\ passengers = no.\ of\ passengers\ inside * ratio\ of\ alighting\ passengers$$

5.4.4 Boarding Passenger generation

The number of boarding passengers is estimated for a time gap between departure-to-departure times of buses. Since the departure time cannot be calculated until the dwell time is known, it is calculated recursively. First, the boarding passengers are estimated for the time gap between departure to arrival. Then the dwell time is estimated using these passengers along with alighting passengers. The passengers generated during the dwell time period is then estimated. Then again dwell time for these passengers is estimated. This total number of passengers gives the total number of boarding passengers. The estimation of the number of boarding passengers is calculated by using following relationship.

$$No.\ of\ boarding\ passengers = time\ gap / rate\ of\ passenger\ arrival\ (time\ gap)$$

5.4.5 Dwell time

The bus stop module obtains the service number of the bus just arrived at a bus stop to choose the suitable dwell time equation. Then the model uses the equation to calculate the dwell time for the estimated number of passengers alighting and boarding. The equations obtained from the field data (Section 4.6.2.3) are:

$T = 6.85+1.69*a +9.00* b$, for double deck buses of (11/11a/101 services) and;

$T = 3.30+1.96*a +9.04*b$, for low floor buses (3/3a services)

Where,

T is dwell time in seconds, *a* is number of alighting passengers and *b* is number of boarding passengers.

Since the estimation of boarding passengers is carried out in a repetitive way, the total dwell time is also carried out in the repetitive way to cater for the passengers estimated to arrive during the dwell time period. The total of these two dwell times give the total dwell time.

5.4.6 Waiting time

The waiting time of a passenger is calculated as the time between the arrival of a passenger at a stop and the arrival of a bus which the passenger boards. The model calculates the waiting time in a cumulative basis using a time gap between buses (Section 5.4.2). The average waiting time for all the passengers generated during a time gap is half of that time gap between two buses, assuming uniform rate of passenger arrival (Section 3.3.2).

$$\textit{average waiting time} = 0.5 * \textit{time gap between buses}$$

This relationship is valid for this high frequency route (6 buses per hour) in the field with passengers arriving randomly (Rajbhandari, 2002). This gives the average waiting time for the passengers arrived during the time gap. To make it the average of all the passengers boarded into the bus, a weighted average is taken for the total numbers of passengers arrived during time gap and the dwell time. Since the rate of arrival is the same, the time difference is used to take weighted average. In the model, the weighted average waiting time is calculated by,

$$\textit{average waiting time} = 0.5 * \textit{time gap} * \frac{\textit{time gap}}{(\textit{time gap} + \textit{dwell time})}$$

5.4.7 Bus occupancy

The module passes the number of passengers alighted and boarded at each bus stop at the time of departure to the bus module. The bus module then calculates the bus occupancy (i.e. the number of passengers inside a bus) by subtracting the number of passengers alighted and adding the number of passengers boarded to the number of passengers already inside at the time of arrival at that bus stop.

5.4.8 Journey time

The journey time of buses between two bus stops is calculated using their arrival times at subsequent bus stops. The bus module then calculates the passenger journey by multiplying bus journey time between the stops by the number of passengers inside the bus at the last bus stop. Hence the journey time of a passenger is calculated as the time between the bus arrival times at origin and destination of that passenger.

5.4.9 Punctuality of buses

A bus is classified as punctual if it departs within an acceptable window either side of the schedule time. The punctuality window used in the model is between 1 minute early to 5 minute late (CPT, 2001). While departing from a bus stop, the deviation of the bus departure time from the scheduled time is calculated and the next scheduled time of the next bus of the same service is updated. If the bus deviation falls within the acceptable punctuality range, then the bus is added to the category of punctual bus. Then the punctuality is calculated as the percentage of these punctual buses.

5.5 Signal module

The signal module models all the activities that take place at a traffic signal. It mainly calculates the signal period and delays at traffic signals every second. The input parameters needed for this module are signal identification number, position of signals, cycle time, green time, amber time, lag, arm flows, saturation flows and the maximum amount of extension and recall. The module produces a detailed output for verification purposes at the time of a bus departure. The output includes the arrival and departure time of a bus, the signal delay, the queuing delay and the total delays. The working of the module is described in the next section.

5.5.1 Working of the module

The main module continuously calls the signal module for an update of signal stages and car numbers at each arm of each junction. The signal module calculates the signal stages, generates cars at all the arms of the junction and then passes to the main module for visual display. It checks whether a bus has reached any traffic signals. Once the module finds a bus at a traffic signal, then the interactions of the signal with the bus starts. The flow chart showing the working of the signal module is shown in Figure 5.7.

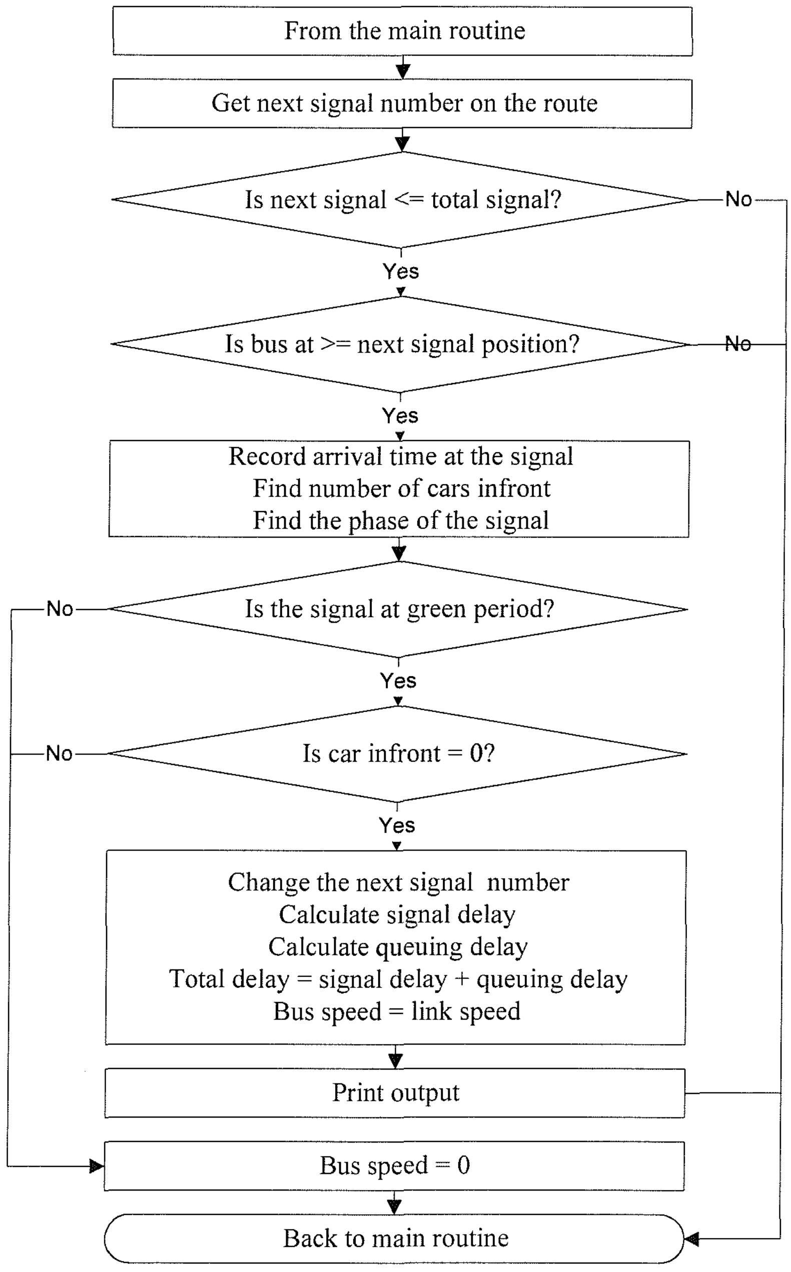

From the main routine
Get next signal number on the route
Is next signal <= total signal?
No
Yes
Is bus at >= next signal position?
No
Yes
Record arrival time at the signal
Find number of cars infront
Find the phase of the signal
No
Is the signal at green period?
Yes
No
Is car infront = 0?
Yes
Change the next signal number
Calculate signal delay
Calculate queuing delay
Total delay = signal delay + queuing delay
Bus speed = link speed
Print output
Bus speed = 0
Back to main routine

Figure 5.7: Flowchart of signal module

Once a bus reaches a traffic signal the bus identification number and the arrival time is noted. With this information, the module calculates the signal period and the number of cars in front of the bus. If the bus has arrived during the green period and there are no preceding cars, then the bus crosses the stop line without stopping. But in the other case, the bus is stopped until the condition is met that the signal is green and there are no preceding cars. Once both conditions are met, the speed of the bus is changed to the average link speed and the bus moves. While crossing the stop line, the total delay to the bus at the signal including signal delay and queueing delay is calculated. In priority mode, the priority award made by the priority module is taken into account while calculating the signal periods.

The main functions such as signal timing calculation, general traffic modelling and bus delays calculation carried out by the module are described in the following separate sub-sections.

5.5.2 Signal timings

The model uses the average length of green time and intergreen time of stages collected from field data (Section 4.6.1.1) in seconds. The signal sequence followed in the model is green-amber-red-red/amber-green. However, for simplification of visual output, red period is displayed in place of red/amber period. Hence, the signal period change from green to amber to red and back to green (without red/amber). Traffic is discharged during the green period of the signal according to saturation flows. Amber period is used in the model for display purpose only and modelled as red period in discharge calculation. No vehicle is discharged when these amber and red periods are displayed. The signals at different arms of a junction are modelled as separate signals linked with each other. The linking is done by the use of a common cycle time and lags to start the green period differently within the cycle time.

The main module continuously asks the signal module to calculate the present signal period for every change in simulation time. While calculating the present signal period, the signal module first calculates the cycle number for the present time. Then it looks for the priority to be provided for the cycle number, as directed by the priority module (priority provision is discussed in Section 5.6.1). The module then calculates the signal period for the present time using the cycle time and lengths of different periods. Figure 5.8 shows the method adopted in calculating the signal period for a given time.

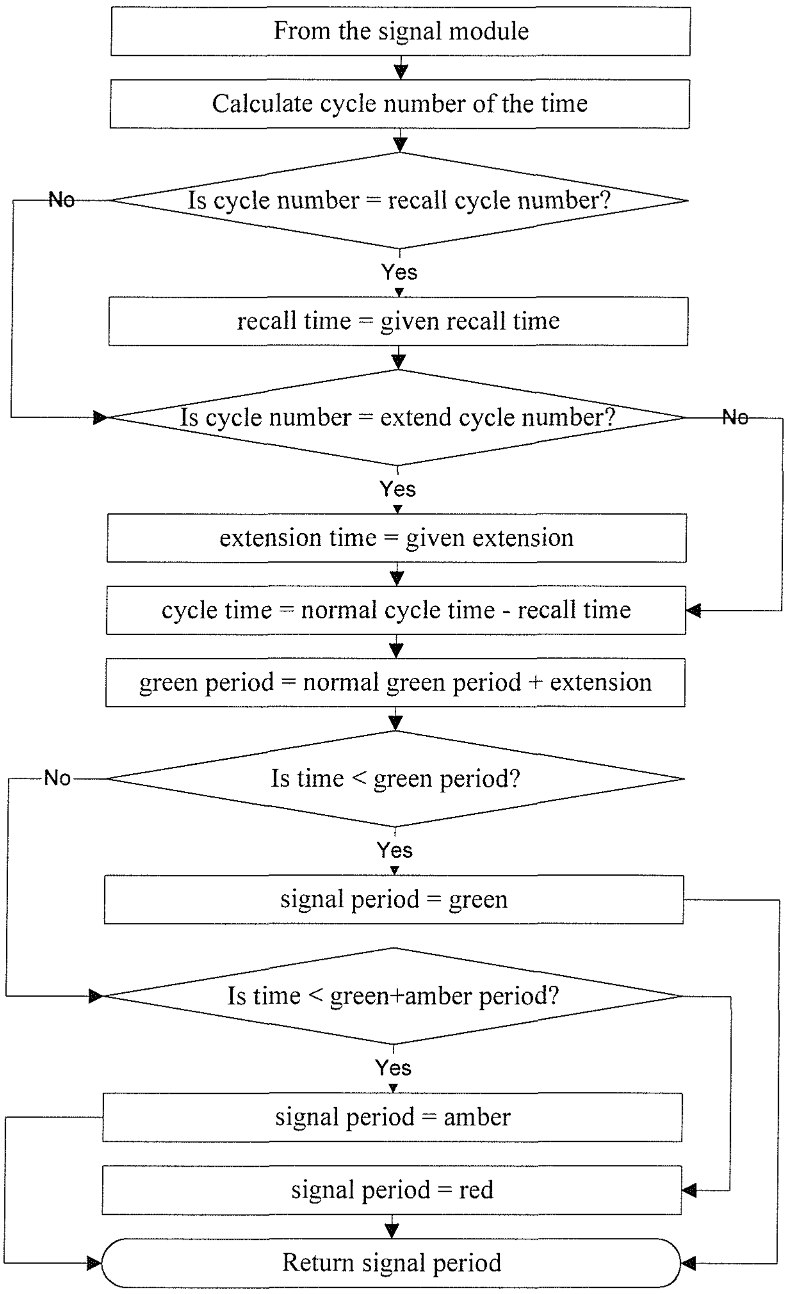

From the signal module
Calculate cycle number of the time
Is cycle number = recall cycle number?
No
Yes
recall time = given recall time
Is cycle number = extend cycle number?
No
Yes
extension time = given extension
cycle time = normal cycle time - recall time
green period = normal green period + extension
Is time < green period?
No
Yes
signal period = green
Is time < green+amber period?
Yes
signal period = amber
signal period = red
Return signal period

Figure 5.8: Flowchart of signal period calculation

5.5.3 General traffic modelling

General traffic is modelled in SIMBOL for estimating delays at traffic signals. The model calculates the time interval to generate a car for each arm of a junction based on the flow data (Section 4.6.1.2). The cars are generated in every time interval passed (based on the flow data) depending only upon the time. When the signal is in the red period the generated traffic is queued one after another, but when the traffic light turns green, the cars are discharged at a time interval based on the saturation flow of the arm. During this process, the number of vehicles at each arm at every second is counted. These number of vehicles present at an arm gives the delays of that many vehicle-seconds for that arm. Since the buses are modelled in one direction, the total delay is calculated for the half of the junction. Hence the total delays at a signal are calculated from the bus route arm and the arm left side of it. The flowchart of general traffic modelling in SIMBOL is shown in Figure 5.9.

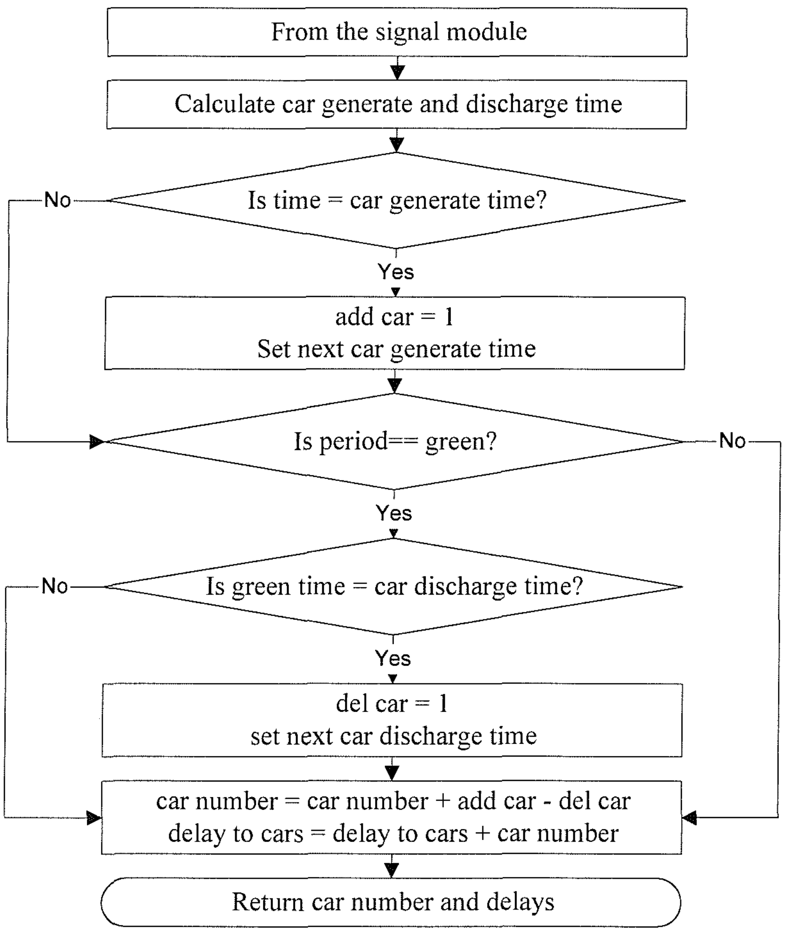

From the signal module
Calculate car generate and discharge time
Is time = car generate time?
No
Yes
add car = 1
Set next car generate time
Is period== green?
No
Yes
Is green time = car discharge time?
No
Yes
del car = 1
set next car discharge time
car number = car number + add car - del car
delay to cars = delay to cars + car number
Return car number and delays

Figure 5.9: Flowchart of general traffic modelling

Visually, the discharged cars disappear at the stop line of a junction. Again to simplify the model, all the cars generated are queued in a single lane rather than the actual number of lanes on the junction. The reduction in the numbers of lanes is taken into account by increasing the saturation flow rate. Hence, though the queueing length is longer visually, the delay associated will be same as if there were multiple lane queueing. Even then, there may be a potential problem of blocking back of the upstream junction if the distance is short and the flow is high. However, the problem did not arise in the route because of longer distances between junctions.

5.5.4 Bus delay calculation

Once a bus reaches the stopline of a traffic signal, the module records the signal period and the number of cars in front depending upon the arrival time. If the bus has arrived during the green period, the time is also noted as start of green period. If the bus has arrived in another period, then the time of actual start of green period is noted as the start of green period. The bus is stopped until the signal is green and there are no cars in front. When the bus crosses the stopline, the time between the arrival and the departure of the bus at the stopline is calculated as the total delay to the bus. The time between the arrival and the start of green period is the signal delay whereas the time between the start of the green period and the departure of the bus is the queuing delay.

Since the buses are modelled for the whole route, unlike cars that are modelled only at junctions, buses are kept in a separate lane from cars. But the effect of queuing delay due to cars is taken account by stopping buses until all the cars in front discharge. This is taken as a simplified method of modelling a complicated signalling system with reasonably accurate delay calculation.

5.6 Priority module

This module is activated when the model runs in priority mode. This module is the one responsible for providing priority to the buses at traffic signals.

5.6.1 Working of the module

The main module calls the priority module to check whether any a bus has reached any detectors/virtual detectors upstream of signals. Then the priority module starts working according to the flowchart shown in Figure 5.10.

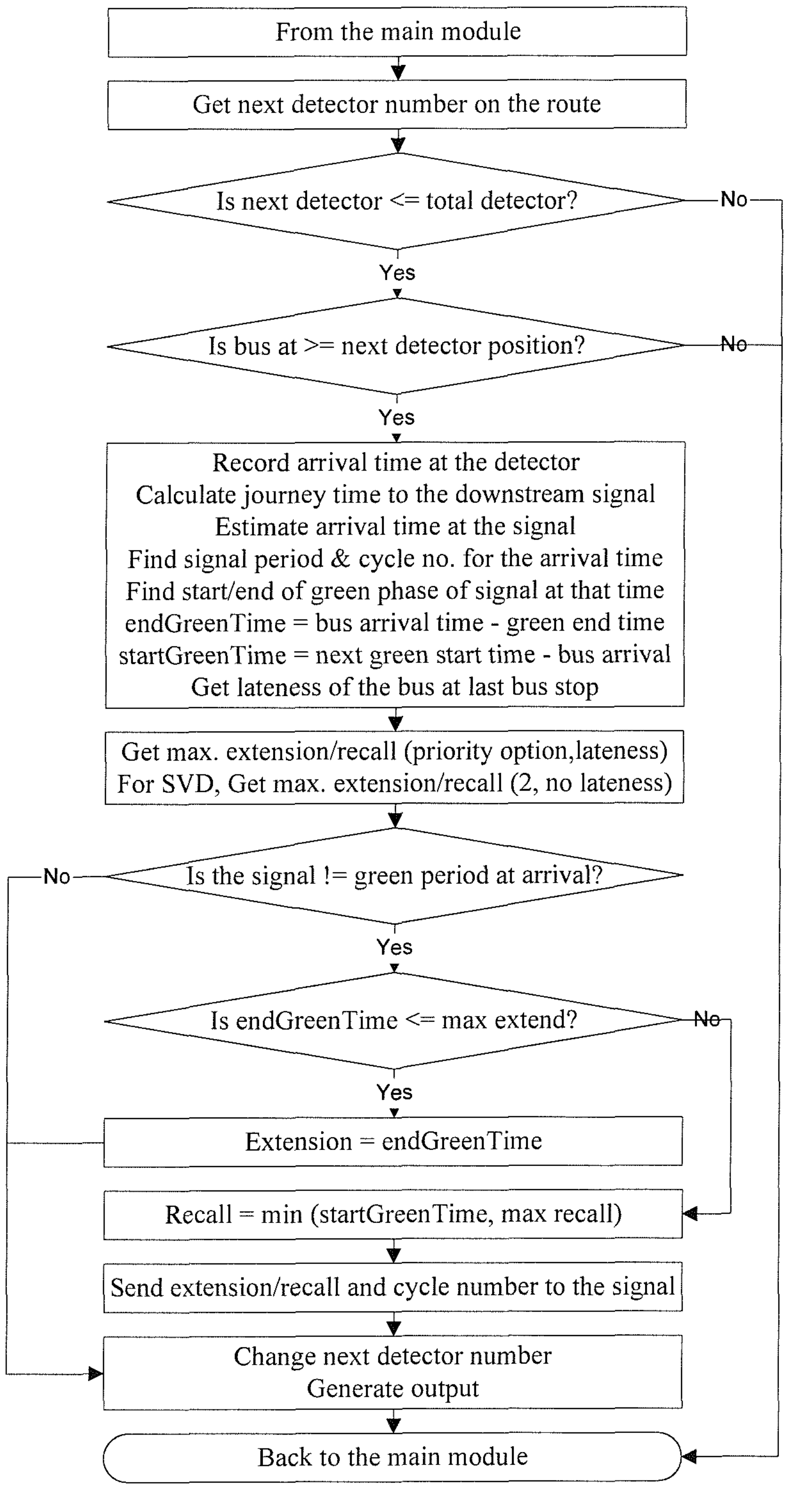

From the main module
Get next detector number on the route
Is next detector <= total detector?
No
Yes
Is bus at >= next detector position?
No
Yes
Record arrival time at the detector
Calculate journey time to the downstream signal
Estimate arrival time at the signal
Find signal period & cycle no. for the arrival time
Find start/end of green phase of signal at that time
endGreenTime = bus arrival time - green end time
startGreenTime = next green start time - bus arrival
Get lateness of the bus at last bus stop
Get max. extension/recall (priority option,lateness)
For SVD, Get max. extension/recall (2, no lateness)
No
Is the signal != green period at arrival?
Yes
Is endGreenTime <= max extend?
No
Yes
Extension = endGreenTime
Recall = min (startGreenTime, max recall)
Send extension/recall and cycle number to the signal
Change next detector number
Generate output
Back to the main module

Figure 5.10: Flowchart of working of the priority module

The module first checks whether a bus has arrived at any of the detectors/virtual detectors. Once a bus approaching a traffic signal crosses a detector/virtual detector placed upstream of the signal, the module obtains the arrival time, the identification number of the downstream signal, the distance, the average link speed of the bus, and the lateness of the bus at upstream bus stop. With this information, the module estimates the time of arrival of the bus at the downstream signal. Then the module calls the signal module to calculate signal period, cycle number, time to the next green period and the time from the present green period. The module also obtains the maximum amount of extension and recall allowed depending upon the priority strategy (Table 5.9) and the lateness of the bus. In case of SVD (priority to all buses), the priority amount is calculated from the second priority strategy that does not take account of lateness. This amount is then checked with the signal state to determine type and amount of priority to be given to the bus.

If the signal is at green period at present and the 'end green time' (bus arrival time-end of green time) is less than the maximum extension allowed, then an extension equal to the 'end green time' is provided. If not, the module checks the amount of recall needed to the start of the next green period. If the recall needed is less than the maximum recall allowed then the recall needed is provided. If the recall needed is more than the maximum allowed then the maximum recall allowed is provided. The amount of extension or recall provided is then sent to the signal for implementation.

The various functions carried out by the module are described in separate sub-sections such as detector positioning, journey time calculation, maximum allowable priority and priority calculation.

5.6.2 Detector positioning

The positions of detectors were calculated from a consideration of the upstream bus stop position and the distance required for the maximum extension period. While calculating the distance from the upstream bus stop, a distance travelled in one second was deducted. This was to make sure that a bus is not at a detector whilst being at a bus stop due to 1 second scanning interval. However, the maximum detector distance was limited to 70 metres. This distance was taken from the recommended distance of 70-100 metres taking care of extension time and journey variability (Bretherton et al, 1996). The minimum out of two distances (just after bus stop and 70 metres) was selected as the optimum detector distance. Table 5.1 shows the distances from the bus stop consideration and the selected optimum distance.

Table 5.1: The optimum location of the detectors

Detector no.	Junction name	Detector distance (m)	
		Bus stop constraint	Optimum distance
1	Stoneham road Jn	45.0	45.0
2	Swaythling Jnction	46.0	46.0
3	Woodmill road Jn	247.0	70.0
4	Mayfield road Jn	227.0	70.0
5	Talk Head	34.0	34.0
6	Highfield lane Jn	91.0	70.0
7	Brookvale road Jn	157.0	70.0
8	Lodge road Jn	21.0	21.0
9	Stage gate Junction	326.0	70.0
10	Cumberland place Jn	373.0	70.0
11	New road Junction	214.0	70.0

5.6.3 Journey time between detector and stop line

The model uses the average link speed of a bus to calculate the journey time between detector and the stop line of the downstream traffic signal. Since the queuing delay at traffic signal (Section 5.5.4) is modelled separately, the blocking due to general traffic to reach to stop line is not considered. Since all the cars are discharged in the green period of a cycle time, the modelling does not have a problem due to blocking while giving priority.

5.6.4 Maximum allowable priority

In the model, the maximum allowable recall amount is constrained by degree of saturation (DOS) whereas maximum extension amount is constrained by journey time between the detector and the stopline. There are two maximum allowable recall amounts calculated for 95% (normal) and 110% (high) Degree of Saturation (DOS) for side roads. However, there is only one set of maximum amount of extensions allowed for both DOS levels. Since the arm flows and signal stages are fixed in the model, the allowable amount is fixed for the whole modelled period.

Out of these 11 signals, signals 5 and 6 have minimum green time for the side road. Hence, it was not possible to reduce these green times to give recall priority to the bus route arm. The only option was to give extension only and not a recall in this case. In case of signals 6 and 11, recall priority is possible only when target DOS is set to 110%. Table 5.3 shows the maximum allowable amount of extension and recall allowed looking towards the detector position and DOS.

Table 5.2: Maximum allowable extension and recall allowed

No	Name	Max. amount allowed (sec)			Maximum amount allowed (sec)			
		Detector constraint	DOS (95%)	DOS (110%)	Normal (95%)		High (110%)	
					Extend	Recall	Extend	Recall
1	Stoneham road Jn	10.0	48.0	51.0	10.0	48.0	10.0	51.0
2	Swaythling Junction	15.0	0.0	0.0	15.0	0.0	15.0	0.0
3	Woodmill road Jn	9.0	22.0	23.0	20.0	22.0	9.0	23.0
4	Mayfield road Jn	7.0	6.0	6.0	7.0	0.0	7.0	0.0
5	Talk Head	5.0	3.0	3.0	5.0	0.0	5.0	0.0
6	Highfield lane Jn	12.0	0.0	3.0	12.0	0.0	12.0	3.0
7	Brookvale road Jn	19.0	10.0	11.0	19.0	10.0	19.0	11.0
8	Lodge road Junction	11.0	16.0	18.0	11.0	16.0	11.0	18.0
9	Stage gate Junction	9.0	0.0	0.0	9.0	0.0	9.0	0.0
10	Cumberland place Jn	16.0	23.0	26.0	16.0	23.0	16.0	26.0
11	New road Junction	15.0	0.0	3.0	15.0	0.0	15.0	3.0

5.6.5 Priority calculation

The module calculates the amount of priority (both extension and recall) allowed based on the criteria of the chosen strategy. In the model, it is based on the DOS level awarded depending upon the lateness of the bus at the upstream bus stop. There are 3 basic strategies included in the model: No priority; Priority to all buses and Priority to late buses only. The 'no priority' is the base case situation. The 'priority to all buses' is the case of selective vehicle detection (SVD) in which all the buses are given priority. The 'priority to late buses only' is a simple form of differential priority. These basic forms of priority strategies are shown in Table 5.4.

Table5.3: Basic form of priority strategies modelled in SIMBOL

Strategy no.	Description
1	No priority (Base case)
2	Priority to all buses (SVD case)
3	Priority to late buses only (Differential priority case)

These basic strategies are further modified and several different strategies based on lateness and DOS are formulated. The formulation and exploration of these different strategies are described in Chapter 7 (Model application).

5.6.7 Priority implementation

The bus priority awarded is implemented by the traffic module by extending the present green period or recalling next green period earlier. While giving extension, the green period of the priority side is increased by the amount of extension time. The red period of the side road is extended by the same amount of time. Once the priority award is completed, the side road green starts and continues for normal period. This increases the total cycle time in that particular cycle. In case of a recall, the red period of the bus route side is reduced by an amount of recall to start the green period early. The same amount of time is deducted from the green period of the side road side. After the priority, the bus route side ends at normal period and hence the side road side starts earlier than usual. In this case of recall, the total cycle time is shortened in that particular cycle. These methods are illustrated in Figure 5.11 and Figure 5.12.

Once the priority process is completed, the normal stage timings start working without compensating the green period lost by the side road side. This process of resynchronisation with the normal stage timings after bus priority has finished is also known as 'recovery'. The model uses 'Do nothing (DN)' recovery method, which is one of 4 methods of recovery available in bus priority in SCOOT (Bowen, 1997). The 'Do nothing (DN)' recovery is the simplest in concept and good method when offset is not important (Bowen, 1997). Hence this Do nothing recovery method is adopted in the model. The examples of do nothing recovery after an extension and recall are shown in Figure 5.11 and Figure 5.12.

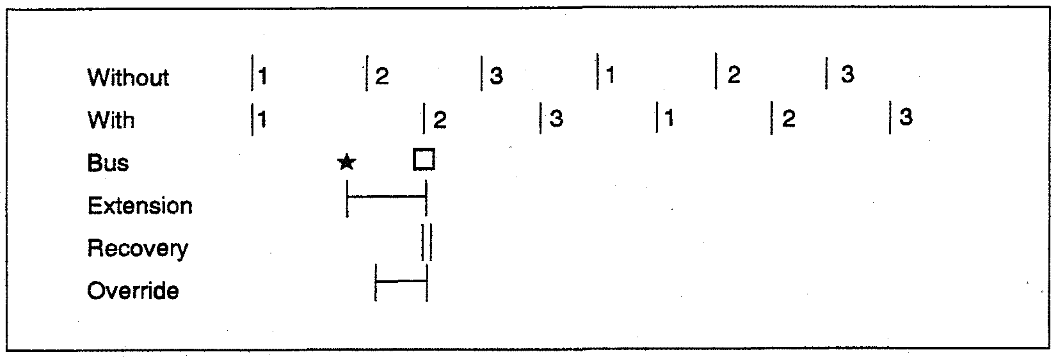


Figure 5.11: Example of 'Do nothing' recovery after an extension (Bowen, 1997)

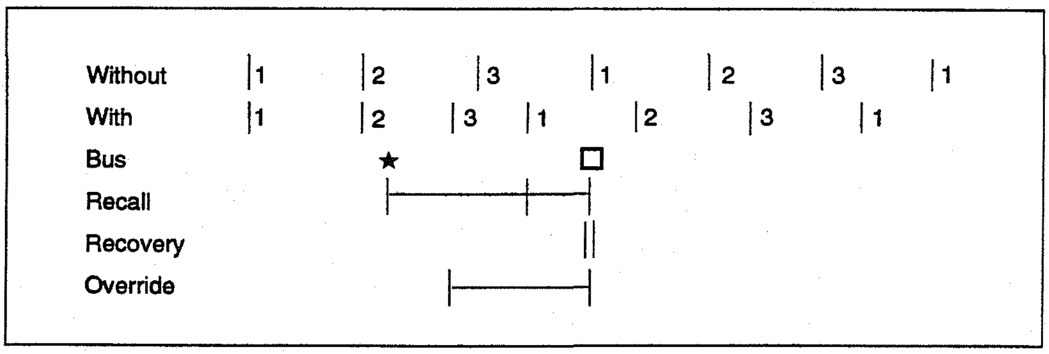


Figure 5.12: Example of 'Do nothing' recovery after a recall (Bowen, 1997)

5.7 GPS module

The GPS module provides the GPS location of buses to detect a bus approaching any traffic signals. This detection is necessary to activate the priority module for the ascertainment and activation of the priority. The detection of an approaching bus is carried out by the use of virtual detectors based on the GPS position of a bus incorporating the error in the GPS system. This module continuously estimates the error and provides the GPS position of a bus based on the actual bus position and the error.

5.7.1 Working of the module

The main module calls the GPS module to get the GPS position of all the buses every second. The GPS module generates a location error for a bus at that moment. The error is estimated from the normal random distribution with mean 0.0 and standard deviation 3.8 (Rupprecht, 2001). The error is then added to the actual position of the bus giving the GPS position of the bus. This GPS position of a bus is used to check bus arrival at a virtual detector and activation of bus priority module (section 5.6). The modelling is based on Decentralised AVL-UTC communication architecture described in Section 2.2.2.2. However, there is no communication delay modelled in the model while ascertaining the priority requirement. This is like the bus keeping the lateness information and deciding the priority requirement itself. This is a possible architecture in the case of timetabled services. The flowchart of GPS module for bus position is shown in Figure 5.11.

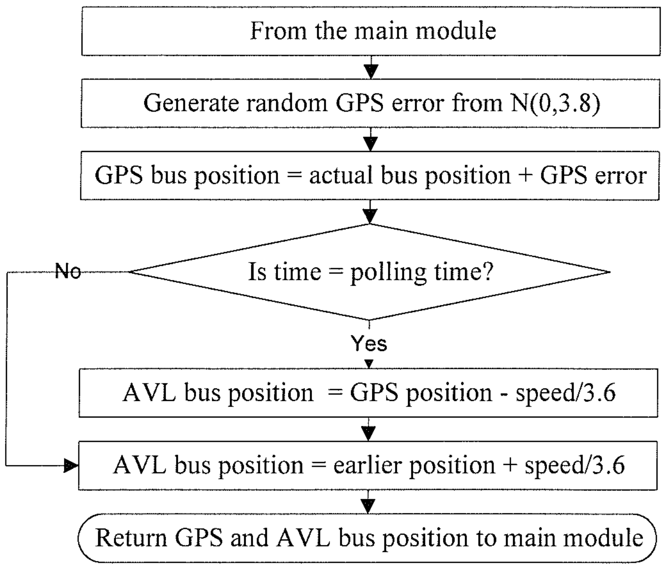

From the main module
Generate random GPS error from N(0,3.8)
GPS bus position = actual bus position + GPS error
No
Is time = polling time?
Yes
AVL bus position = GPS position - speed/3.6
AVL bus position = earlier position + speed/3.6
Return GPS and AVL bus position to main module

Figure 5.13: Flowchart of GPS module

The GPS position of buses is used to detect a bus approaching a traffic signal. The detection is carried out by a virtual detector upstream of the signal. More about the virtual detector is described in next sub-section.

5.7.2 Virtual detectors

In case of GPS system, a bus is detected by virtual detectors while approaching a traffic signal. The functioning of these virtual detectors is similar to the detectors placed on the road for selective vehicle detection (SVD). However, the virtual detectors are marked by their positions on the bus computer rather than placing on the road. This makes the positions of these virtual detectors very flexible and can be easily changed, if needed.

The positions of the virtual detectors are the same as ordinary detectors (given in Table 5.1), when the GPS system is modelled with no error. However, when the GPS error was modelled as maximum +/- 10 metres, the maximum detector distance from bus stop position constraint was reduced by 10 metres. This was to ensure that a bus at an upstream bus stop is not detected due to GPS error. Hence the optimum distances of the virtual detectors based on the bus stop position earlier were changed. Table 5.4 shows the location of the virtual detectors placed in the systems.

Table 5.4: The optimum location of the virtual detectors with GPS error

Detector no.	Junction name	Detector distance (m)	
		Bus stop constraint	Optimum distance
1	Stoneham road Jn	35.0	35.0
2	Swaythling Jnction	36.0	36.0
3	Woodmill road Jn	237.0	70.0
4	Mayfield road Jn	217.0	70.0
5	Talk Head	24.0	24.0
6	Highfield lane Jn	81.0	70.0
7	Brookvale road Jn	147.0	70.0
8	Lodge road Jn	11.0	11.0
9	Stage gate Junction	316.0	70.0
10	Cumberland place Jn	363.0	70.0
11	New road Junction	204.0	70.0

This change in detector distance changed the maximum allowable extension time of the signals. However, in the model, the reductions in allowed extension time are modelled by giving extra amount of extension than actually required.

5.8 Chapter summary

This chapter has described the development process of the simulation model SIMBOL. The workings of different modules of the model are described with the aid of flowcharts. Since the model is based on the data from the bus route in Southampton, it is necessary to check whether the model works satisfactorily as in the field. The next chapter 'Model Validation' describes the method adopted in validating the model with the validation data collected.

Chapter Six

Model Verification and Validation

6.1 Introduction

Chapter 5 described the process of model development based on the methodology described in chapter 3 and data collection in chapter 4. This chapter describes the method adopted for checking the developed model to eliminate errors and to ensure that it works as intended. The first portion of this chapter, model verification, describes the checking of the model to make sure that it is error-free. The second portion, model validation, describes validation of the model in the field by checking the model output against the field data collected.

6.2 Model verification

Davies and O'Keefe (1989) have defined verification as the tasks associated with checking the model and corresponding programs to ascertain that they perform as intended. This outlines the requirement of a model verification process. Here, verification of the model was carried out by checking the output from the series of test runs using hypothetical data. Specifically, the detailed checking was carried out using the following output:

- Visual output during simulation runs;
- Text output files detailing the performances of each of the modelling parameters such as bus stop, traffic signals and buses.

6.2.1 Verification using visual output

Verification started with checking the generation of buses at different bus stops at their generation time. Then the bus journey along the route to the destination was checked including interactions at bus stops and traffic signals. This was to ensure that the buses were complying with the following requirements: stopping at bus stops to let passengers alight and board; stopping at the red period of traffic signals and going at green periods; and allowing the vehicles in front of them to go first at the green period.

The signals were visually checked for the changing of signal phases, stopping of vehicles at the red period, increase in queue lengths during the red period and discharging of vehicles at the green period. Looking at signal functioning in more detail, the lengths of each signal period were checked against the input data, as well as the number of cars generated during red periods and discharged during green periods. The synchronisation of different signal heads at the same junction was another aspect to be verified. Once signal functioning had been verified, priority operation was checked. In this process, the length of extension and recall were noted to make sure that they are within specified limits. For all this visual verification, the speed of the simulation was made equal to real time by adjusting the simulation speed input. More detailed verification of the model was carried out by checking the text output files giving details of bus, bus stop, signal and detector.

6.2.2 Verification using text output

The model was verified in detail using the output files generated by different modules. The main module produced the output file giving positions of all buses in the system at every second. This information was used to check the distance travelled by a bus within every second and to countercheck the output from other modules. This detailed output gave information about bus travel as well as stoppage at bus stops or traffic signals. The stoppage information was counterchecked with the data from the bus stop module or signal module. To simplify the checking process, input data such as a bus speed of 18 km/hr (i.e. 10 m/sec) and a 100 second cycle time with 50 second green time were used. Beside that, the output from all the modules were checked themselves or counterchecked with output from other modules. For example, the output from the signal module was checked against that of the detector module. If the detector predicted that the bus would get an extension of 10 seconds, the traffic signal module output was inspected to see whether the bus got that extension of green period or suffered any delays at all.

The working of the bus stop module was verified using the information such as arrival and departure time of a bus, its gap with the preceding bus, numbers of passenger boarding and alighting and dwell time obtained from the output file. Arrival and departure time were checked against the bus model to verify when it really stopped and moved from the bus stop. Calculation of the time gap between the departure of one bus and the arrival time of the next bus at bus stops were also checked. Estimated numbers of passengers boarding and alighting were checked with the time gap, the arrival rate of boarding passengers, the number of passengers inside and the percentage of alighting passengers at that bus stop. The dwell time of buses was also checked with the number of passengers boarded and alighted.

Regarding the signal module, the signal period and the number of cars in front at the time of bus arrival were calculated from signal input data and verified against the output from the module. The departure of the bus and the signal period were then crosschecked. Signal delay and queuing delay were then checked using available data. In the case of the priority option, the arrival time of a bus as predicted from bus detection was checked against the actual arrival of the bus at the stop line of the downstream traffic signal. The priority need of a bus predicted by the detector was compared with the actual priority provided by the downstream traffic signal. The detailed output produced by the GPS module giving details of the error generated along with actual bus position was also checked against the bus position from the main module.

6.3 Model calibration

The calibration of the model was carried out to set saturation flows of arms of all junctions. Additionally, the bus stop dwell time parameters for different bus services serving the route were also calibrated (Section 4.6.2.3). Since SIMBOL has modelled any number of lanes in an arm as a single lane, saturation flows were to be calibrated for the arms of all junctions. Saturation flow values were initially set at levels referred in the available literature for different lanes (Salter and Hounsell, 1996). They were then modified by trial and error method until the resulting delays matched those collected from SCOOT data. In this process, care was taken to avoid unrealistic figures for the widths and number of lanes available at a junction arm. Realistic saturation flows were used in the model even though some of the arm delays estimated by the model did not match the SCOOT prediction.

6.4 Model validation

Validation of the model was carried out to make sure that it closely reflected the system in the field. This was done mainly by comparing the journey times of an individual bus obtained from the model to that collected in the field, typically over a 2-hour period. Both graphical and statistical methods were used. Details of these validation methods are described in the following sub-sections.

For the validation of the model, the arrival times of consecutive buses at origins and destinations were collected in the field for a week. Data for consecutive buses were required because the dwell time of a bus at a bus stop is modelled as depending upon the gap between consecutive buses. A weeklong survey, to collect arrival time of consecutive buses at origins and destinations, was planned separately from the model development data collection described in chapter 4.

Bus arrival data was collected from 05/11/2001 to 09/11/2001 (one week) between 10:00 – 12:00 (two hours) for the buses of services 11/11A, 3/3A and 101 going towards the City Centre. The data was collected simultaneously by 3 persons standing at 3 different bus stops - Swaythling, Portswood and City Centre. Here, Swaythling is the origin of 11/11a buses and Portswood is the origin for 3/3a/101 buses. City Centre is the destination stop for buses of 11 and 3 services. Data collected at each stop included the arrival and the departure time of buses along with their service number and vehicle registration number. The service and registration numbers were noted so that buses could be tracked at different bus stops. To record the arrival time of all buses generated between 10:00 and 12:00, the survey period was longer at Portswood and City Centre. At the City Centre bus stop, data was collected until the last bus from Swaythling and Portswood arrived. To keep survey times synchronised and accurate at all three bus stops, digital watches were used. These were checked against British Standard Time every morning.

Data collected at all three bus stops for each day were combined by checking the service number and the registration number of the buses. The combined data was then sorted according to the generation time of buses at their origin (Swaythling or Portswood). This was necessary to order the generation time of buses at different origins. The rank number of each bus in the ordered series was used as its identification number in the model. The journey times from origins to destination were then calculated. Here, the journey times of service 11 buses were calculated for the entire route between Swaythling and City centre. The journey times of service 11a buses were calculated for the portion of the route between Swaythling and Portswood, and for service 3 buses between Portswood and City Centre. Five tables showing generation times of buses at origin and journey times to destination were prepared for the 5 days of data collection. Table 6.1 shows a sample of these formatted data for one day (05/11/2001).

Table 6.1: Formatted validation data collected on 05/11/2001

		Arrival time (hr:min:sec)			JourneyTime (O-D)		Start time
BusId	Service	Swaythling	Portswood	City	(hour:min:sec)	(seconds)	(seconds)
1	3		10:01:29	10:13:27	00:11:58	718	89
2	11a	10:02:46	10:13:33		0:10:47	647	166
3	11	10:08:57	10:17:44	10:28:25	0:19:28	1168	537
4	3a		10:19:54	10:28:20	00:08:26	506	1194
5	11a	10:20:53	10:30:38		0:09:45	585	1253
6	3		10:21:16	10:31:40	00:10:24	624	1276
7	101		10:22:51				1371
8	11	10:28:19	10:38:28	10:51:45	0:23:26	1406	1699
9	3a		10:35:47	10:48:25	00:12:38	758	2147
10	11a	10:38:57	10:48:25		0:09:28	568	2337
11	3		10:46:41	10:58:30	00:11:49	709	2801
12	11	10:48:27	11:04:20	11:17:00	0:28:33	1713	2907
13	101		10:48:29				2909

14	3a		10:57:28	11:08:24	00:10:56	656	3448
15	11a	11:00:11	11:12:56		0:12:45	765	3611
16	3		11:03:26	11:16:40	00:13:14	794	3806
17	11	11:11:45	11:20:34	11:31:46	0:20:01	1201	4305
18	3a		11:13:31	11:21:55	00:08:24	504	4411
19	101		11:20:14				4814
20	3		11:22:36	11:32:01	00:09:25	565	4956
21	11a	11:24:09	11:34:40		0:10:31	631	5049
22	11	11:30:12	11:37:35	11:49:02	0:18:50	1130	5412
23	3a		11:35:26	11:46:03	00:10:37	637	5726
24	3		11:40:05	11:51:48	00:11:43	703	6005
25	11a	11:40:44	11:55:07		0:14:23	863	6044
26	11	11:48:11	11:56:59	12:08:24	0:20:13	1213	6491
27	101		11:52:13				6733
28	3a		11:57:23	12:05:12	00:07:49	469	7043
29	11a	11:58:54	12:08:45		0:09:51	591	7134
30	3		12:00:11	12:10:03	00:09:52	592	7211

The collected data was then used to generate buses at origins and to check the output from the model in terms of bus journey times.

6.4.1 Journey time validation

The journey time validation was carried out by comparing the journey time of each bus as given by the model with that obtained from the field data. The field journey times of the buses were obtained above and the model journey times of buses from the simulation runs. To get model journey times of buses, 5 different simulation runs were carried out similar to those 5 days of data collection. Buses were generated at Swaythling and Portswood according to their arrival time at these bus stops in the field data. The simulation runs were continued until all these buses generated at Swaythling and Portswood had arrived at the city centre. At the end of the simulation, the arrival times of buses at their destination were produced. Bus journeys were then calculated by subtracting generation time at origin from arrival time at destination stop. Since origin and destination differed for different services, individual journey times of buses were also calculated for different portions of the route. The journey times of service 11 buses were calculated for the whole route, those of 11a buses for the Swaythling to Portswood portion and those of service 3 buses for the Portswood to City Centre portion.

The journey times thus obtained from the model were validated against the field journey times. The initial journey time validation was carried out by plotting, for each bus, the model journey time against the field journey time. This plotting of journey times of buses is shown in Figure 6.1.

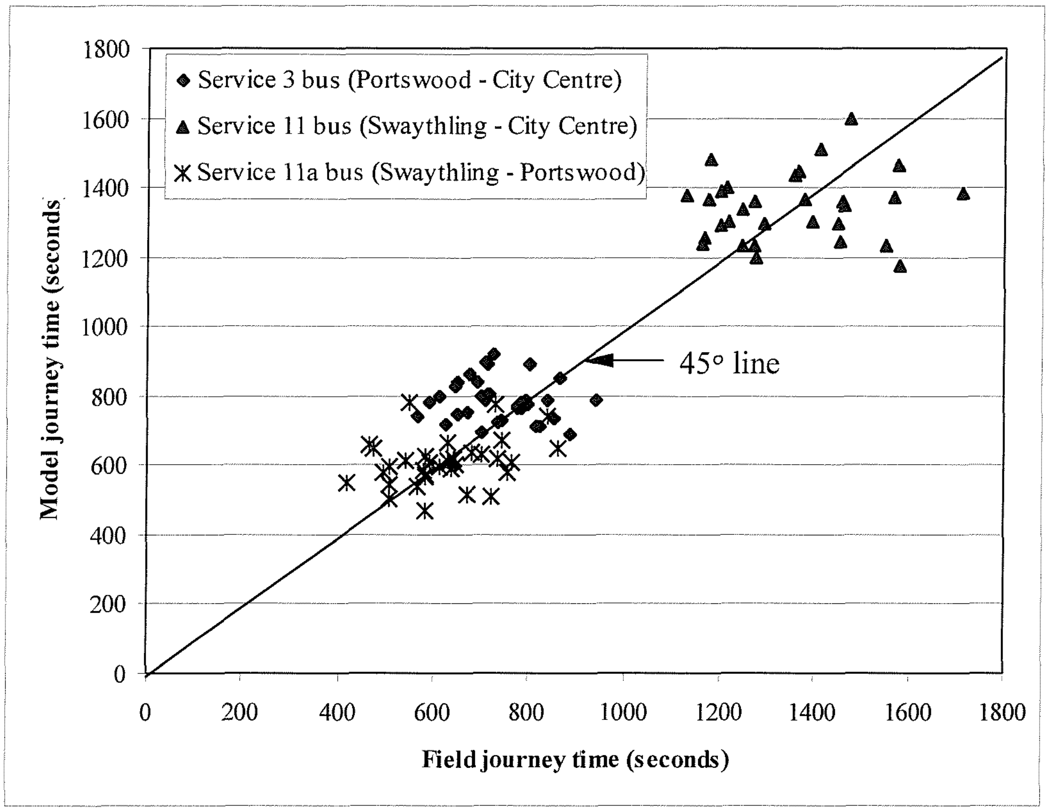


Figure 6.1: Model bus journey time against field journey time

The graph shows a close compatibility between the model journey times of buses and field data. Despite the use of average link speed for all the buses in the model, SIMBOL prediction shows considerable variation in bus journey time. The variation is due to the different dwell times and delays at signal for different buses. Again, the model variation is quite compatible with the field variation of journey times. This shows the strong performance of the model in field conditions. The R^2 value for the data is calculated as 0.84, showing a strong correlation between model result and field data.

These sets of values were then statistically tested using t-test for paired data. Here, the model journey times and field data of individual buses were compared to see if they differed significantly from each other at 5 percent level. While testing, the difference between field data and simulation results was found to be not significant at the 5 percent level. This again confirmed the compatibility of the model results with the field data.

6.4.2 Bus delay validation

The saturation flows of each arm of all junctions were calibrated by matching average delays to vehicles from the model and from SCOOT (Section 6.3). However, bus delay being modelled on an individual basis in SIMBOL, it was thought sensible to compare the model prediction of bus delay with the average vehicle delay obtained from the SCOOT data. Table 6.2 shows the comparison made between model predicted bus delays at the bus route arm of traffic signals and the average delay data from the SCOOT system.

Table 6.2: Comparison of field delay data against bus delays from SIMBOL

Signal	Delays (sec/veh)		Difference
number	Field data (average delays)	Model data (bus delays)	(sec/veh)
1	38.27	45.40	-7.13
2	5.89	5.88	0.01
3	31.98	21.30	10.68
4	9.00	7.52	1.48
5	4.81	3.40	1.41
6	26.33	22.24	4.09
7	22.19	20.32	1.87
8	-	24.60	-
9	39.69	36.16	3.53
10	51.77	48.24	3.53
11	38.23	39.18	-0.95

These sets of values were then tested statistically using t-test for paired data. The test showed that the difference between the field data and simulation results was found to be not significant at the 5 percent level. Hence the model delay calculation is found to be valid in estimating field delay within accepted statistical limits.

6.5 Chapter summary

This chapter has described the verification and validation process carried out for the model to eliminate errors and to ensure that it works as intended. The process began with verification and was followed by calibration and then validation of the model. Validation data were collected in the field for a week. The collected data were then used to validate bus journey times. This showed the model results to be compatible with field data. It is hence concluded that the model is valid for the field conditions of the Portswood corridor. The validated model can therefore be used to explore different bus priority options and their impacts on the system in order to develop strategies for bus priority at traffic signals. The applications of the model for different scenarios of bus priority options are described in Chapter 7.

Chapter Seven

Model Application

7.1 Introduction

Chapter 6 described the validation process of the model. The validated model was then used to simulate different bus priority strategies to compare their performances in various scenarios. These scenarios were built by making possible changes in the different key components of the model. The aim of the application was to explore the impact of the changes on the priority strategies with different scenarios that might not be possible to test in the field. Another aim of the application was to progress towards determining suitable strategies under different circumstances.

This chapter starts by describing the methodology adopted to compare different priority strategies. This is then followed by a series of sections describing the simulations carried out to model changes to different components of the system. This includes the description of the scenario tested, simulation results and discussion. The chapter concludes with a summary.

7.2 Methodology

The simulation model provided an opportunity to simulate various priority strategies and to build different scenarios by changing certain parameters. Since it was not feasible to study all possible strategies and scenarios, certain key ones were selected. These are described in the subsequent sub-sections along with the method of evaluation.

7.2.1 Types of priority strategies

The first step in applying the model was to select priority strategies for comparison. At the initial stage, 3 basic priority strategies were selected for detailed exploration. These are as follows:

- Priority to all buses
- Priority to late buses only
- Mixed priority (late buses and all buses)

The first strategy is a common form of priority in which all buses are given priority within the allowable limit of a traffic signal. This type of strategy helps buses to reduce signal delay and to improve journey time. It is widely used in the form of "Selective Vehicle Detection' (SVD), described in Section 2.1.2.

The second strategy, giving priority to late buses only, is more advanced. It helps to make buses punctual and hence reduce passenger waiting time. This is the simplest form of 'Differential Priority' described in Section 2.1.3. This type of strategy is further enhanced by changing the level of priority depending upon the lateness of a bus.

In the course of the application of the model, a third priority strategy was developed by mixing the first two. This was named 'Mixed Priority' and gives priority to late buses in the earlier part of the route and priority to all buses in the latter part of the route. The rationale behind this strategy is discussed in the next-sub-section.

7.2.1.1 Mixed priority

In general, bus stops along the latter part of a route have more alighting passengers and few boarding passengers. In such circumstances, the waiting time saving from a punctual service has only a small effect on total passenger waiting time. Making buses punctual by giving priority only to late buses therefore may not be the preferred option. Rather, passenger journey time can be reduced by giving priority to all buses. The passengers onboard would be more than happy if they arrive earlier than scheduled timetable rather than late. In this case, the strategy giving priority to all the buses would be more beneficial.

In contrast, bus stops along the early part of the route have more boarding passengers and less alighting passengers. Reducing average waiting time per passenger therefore has a bigger impact on the total passenger waiting time. Conversely, with lower number of passengers on board, the benefit of reduced journey time may have less effect on the total passenger journey time. Additionally, the main role of differential priority is to play a corrective role to prevent late buses from further deterioration further along the route. The priority provided at the early part of the route may reduce the increase of lateness with distance/time.

Hence the mixed priority strategy was formed by combining the strategy giving priority to late buses and the strategy giving priority to all buses. This strategy should therefore combine the better passenger waiting time from the strategy giving priority to late buses with the passenger journey time benefit from the strategy giving priority to all buses.

7.2.2 Selection of scenarios

The selected priority strategies were first compared using field data. Then a further study was carried out by simulating the strategies under different scenarios. The scenarios were constructed by changing the characteristics of the main components of the bus operation system. The simulation model provided ample choice to change characteristics of different components. However, it was not possible to investigate all different combinations hence the change in only one main modelling issue of each main component was considered. The modelling issues selected for scenario formation were based on what can practically be implemented or already occur. These included issues such as changes in Degree of Saturation (DOS) at traffic signals, changes in bus generation etc. Taking one main issue of each component such as bus, passengers, bus stops, signals and GPS system, the different scenarios formed for model application were as follows:

- Target DOS at signals

- Types of bus generation
- Change in passenger demand
- Intervention (holding of early buses at selected bus stops)
- Error in GPS
- Change in bus operation (headway-based service)

These scenarios were simulated separately for the priority strategies discussed earlier. The simulation results for each priority strategy were then used to explore and compare the strategies under these different scenarios. The method of evaluation of these priority strategies is discussed in the next sub-section.

7.2.3 Method of evaluation

The evaluation of different priority strategies was based on a comparison of their respective performances. The performance of each strategy was indicated by the output file generated at the end of each simulation run. Performance criteria included the change in bus delays at traffic signals, change in punctuality of buses and economic evaluation of the benefit obtained. The economic evaluation was based on key performance parameters such as bus journey time, passenger journey time, passenger waiting time and car delays at traffic signals.

The economic values for these parameters were calculated as the resource values of time per person from Highways Economics Note No.2 (HEN2, 1997). This reference gives the resource values of time per person for driver and passengers of cars and public service vehicles; based on 1994 prices. In the present study, the same 1994 prices were used. Updating was considered unnecessary because a cross-strategy comparison would be unaffected by the exact value. The reference also gives average vehicle occupancies for cars and public service vehicles. These values are useful in calculating average resource values on a per vehicle basis. The average resource value of time per car, taking care of passengers, was obtained from Table 2.4 in the same reference. In case of buses, the average occupancy values were obtained directly from the simulation results in the present study.

Based on the note, passenger journey time value was taken as the standard non-working time from table 2.1 (HEN, 1997); and the value of passenger waiting time as double the passenger journey time value. The bus driver's journey time was taken as the working time of a PSV driver. The car occupants' (including driver and passenger) journey time value was taken from the average value given in Table 1.3 of the note. This took account of car occupancy during both working and non-working times. The economic values of the different parameters are therefore as follows:

- Passenger journey value = £3.15 (per passenger-hour journey time)
- Passenger waiting value = £6.30 (per passenger-hour waiting time)
- Bus driver's journey value = £9.83 (per bus-hour journey time)
- Car occupants' journey value = £6.74 (per car-hour journey time)

These economic values were used to convert the benefit and disbenefit obtained from a particular priority strategy into economic terms. This formed one of the main bases for comparison of the different priority strategies.

7.3 The base case scenario

Application of the model started with a simulation of the bus route in its current state. In this scenario, simulations were carried out for different priority strategies as well as the no priority case. All components were modelled based on their characteristics in the field. However, because the field timetable given by the bus operator was very unrealistic, a modified timetable (Table 4.10) was used instead. The modification to the timetable was based on the field journey time of buses and the actual starting deviation at the origin. As is the practice in the field, buses were not stopped at all bus stops, even if they were earlier than their scheduled timetable. Furthermore, in contrast to the usual assumption of a timetable service, the passenger arrival was modelled as random due to the high frequency of buses (6 buses per hour). Again, this was as observed in the field for this corridor (Rajbhandari, 2002). Bus priority was implemented by the method of 'extensions' and 'recalls' described in Section 3.5.2. Recovery of the signal setting after a priority award used in the model was the 'Do Nothing (DN)' method described in Section 5.6.6. Using this method, no compensation of lost green time is given to the side road following the 'Recall' process.

The output from each simulation run for each type of priority strategy was stored and compared against the output for the no priority case. A comparison of the results is described in the next sub-section.

7.3.1 Simulation results

Four simulation runs were carried out, which covered the no priority case and 3 other priority strategies. The main impact of bus priority at traffic signals being reduced bus delay at junctions, the delay saving per bus per junction was compared for different priority strategies. The average delay per bus per junction was 24.7 seconds, which is almost equal to the car delay per junction (24.9 secs) in the no priority case. Compared to this, average bus delay savings of 10 seconds per junction were achieved in case of priority to all buses, 4 seconds for priority to late buses and 6 seconds for mixed priority. Figure 7.1 shows the bus delay savings obtained from different types of priority strategies.

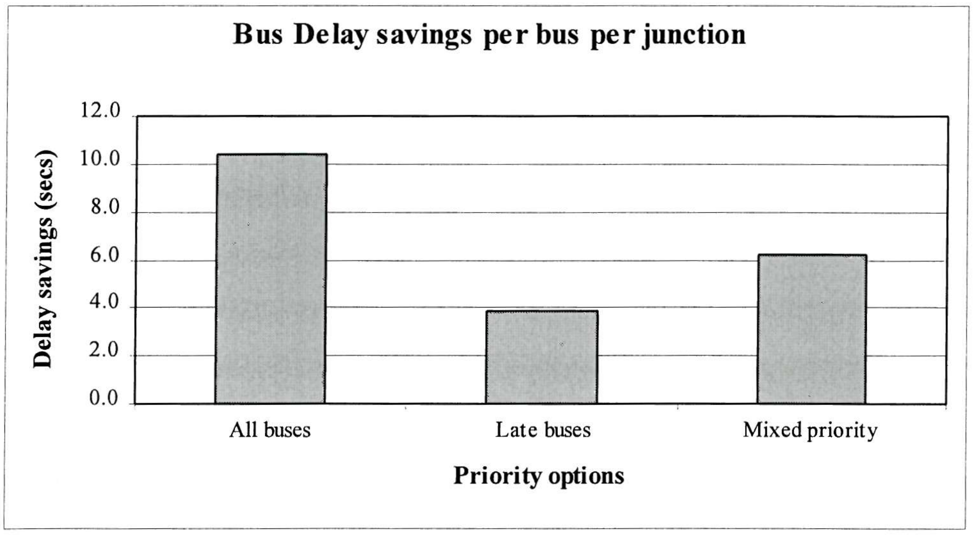


Figure – 7.1: Bus delay savings achieved through different priority strategies

The delay saving of 10 seconds per bus per junction from priority strategy giving priority to all buses is slightly higher than reported in other studies. However, the earlier field trials carried out in London and Southampton (Bretherton et al., 1996) suggested that the delay saving of that level is possible in low saturation levels. Since the junctions were modelled for off-peak periods, the saturation level was quite low and hence these higher delay savings are possible. The reduction in delay savings from a strategy giving priority to late buses only is due to the exclusion of early or on-time buses from getting priority. Delay savings from mixed priority is only a little better, because priority is given to all buses only at 4 out of 11 junctions.

Beside the bus delay savings, punctuality of buses is another aspect to be compared. This is the criterion by which the performance of most timetable services is judged. An improvement in bus punctuality also reduces passenger waiting time (Section 2.1.2). The punctuality window used by Confederation of Passenger Transport is between 1 minute early to 5 minutes late (CPT, 2001). Using this window of punctuality, the percentage of punctual buses at each bus stop is shown in Figure 7.2a.

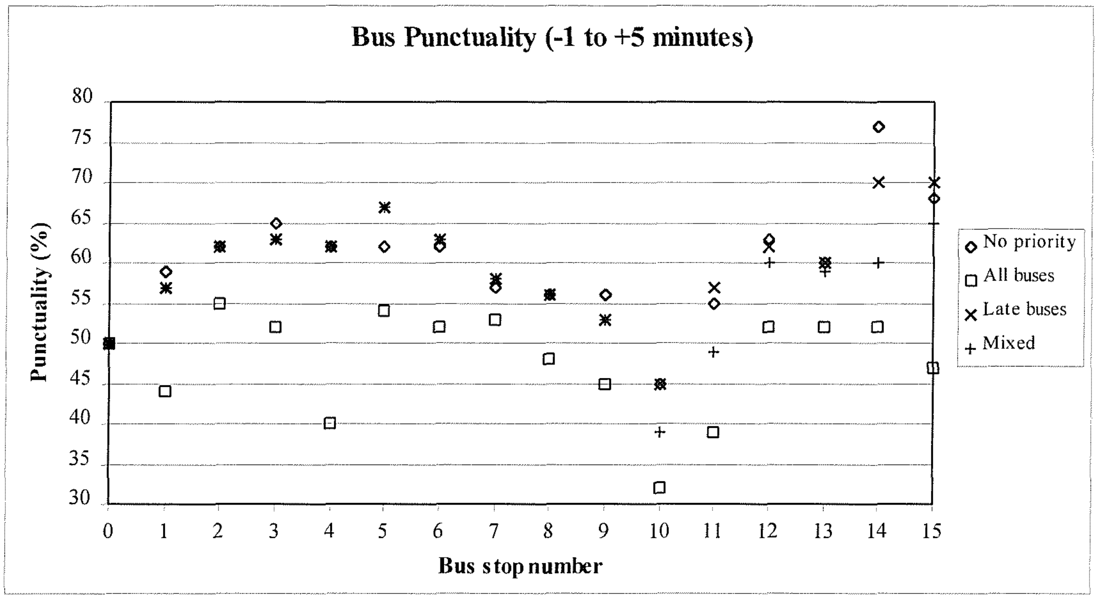


Figure – 7.2a: Punctuality of buses (1-5) from different priority strategies

In many cases, the punctuality of buses is worsened while giving them priority because the proportion of early buses increases. Furthermore, the punctuality is worst in case of the strategy giving priority to all buses. The main reason for this poor performance is the presence of more early buses (72% at bus stop 0) due to non-holding practice in the field. In such a situation, the strategy that gives priority to all buses makes more buses early and puts them out of the punctuality range.

Additionally, the better performance of the 'no priority' case is due to the big window of punctuality. This big window makes 'late' buses also punctual. For this 10-minute frequent bus service, the punctuality window of 6 minute is quite high. Hence, the window of punctuality for this 10-minute service was redefined as 1 minute early to 3 minutes late. With this new definition, the percentage of punctual buses at each bus stop is shown in Figure 7.2b.

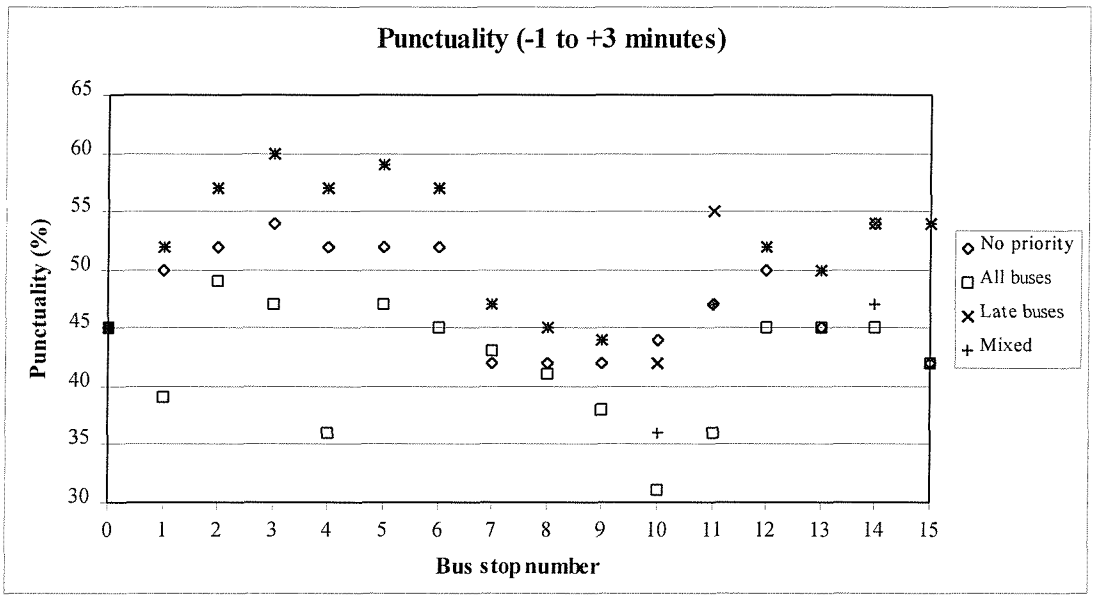


Figure – 7.2b: Punctuality of buses (1-3) from different priority strategies

This graph shows that the strategies giving priority to late buses improves the punctuality of buses. In contrast, the strategy giving priority to all buses is the worst in terms of punctuality because it makes earlier buses even more early. Here again, the performance of this strategy is influenced by more early buses due to no-holding practices in the field. The mixed priority strategy performs equally to the late buses priority strategy up to bus stop 9 and remains good even after the start of the priority to all buses strategy. Its performance for the later bus stops is among the best due to the effect of good performance earlier.

Since priority to all buses was found to provide the best bus delay savings, and the mixed priority and priority to late buses strategies the best punctuality, it was necessary to evaluate these strategies on a common ground of economic benefit. As mentioned earlier (Section 7.1.3), passenger waiting time, bus journey time, passenger journey time and delays to general traffic were taken as the parameters for economic assessment. The values of these parameters used for economic assessment were taken for a one-hour period. Table 7.1 shows the total times of each parameter in one hour and economic values of benefit for each parameter after deducting from the no priority case.

Table 7.1: Economic benefits from different priority strategies

Priority	Total times per hour (in hours)			Priority benefit per hour (in £)				
	wait	jrTime	carDelay	passWait	jrTime	busTotal	carDelay	Total
No priority	13.77	51.06	45.25	-	-	-	-	-
All buses	13.52	47.24	44.94	1.58	13.70	15.28	2.09	17.37
Late buses	13.35	49.48	45.19	2.65	5.78	8.42	0.40	8.83
Mixed	13.36	48.75	45.03	2.58	8.35	10.93	1.48	12.41

The table shows that all priority strategies give some benefit as compared to the no priority case. Giving priority to all buses provides the most overall benefit. This strategy also gives the maximum benefit while considering buses only ('busTotal' column of above table). Although this strategy gives the least waiting time benefit, it is the best overall performer due to journey time savings. Giving priority to late buses provides the best waiting time benefit, but due to lesser journey time benefit it gives the lowest overall benefit. The mixed priority strategy is found to be in the middle ground, giving good waiting time benefit along with good journey time savings. Even so, this strategy gives lesser overall benefit than the strategy giving priority to all buses.

Journey time savings are found to be the main contributor to overall benefit for all the priority strategies. Its impact is the greatest for the strategy giving priority to all buses. In addition to journey time savings, it is interesting to find some waiting time savings for the strategy giving priority to all buses. Intuitively, this strategy is thought not to alter the gap between the buses, nor therefore the waiting time of passengers. In reality however, even this strategy rectifies the distortions in the gaps between buses caused by red periods of a signal: a bus arriving during the green period will neither need nor get a priority whereas a bus arriving in a red period will get priority, thus reducing the distortion in the gap. Hence there will be some waiting time benefit from the strategy giving priority to all buses. This is supported by the field trial carried out in London (Hounsell et al, 2000), which found a waiting time benefit in the range of 2.5% from a similar strategy. However, the strategies giving priority to late buses and mixed priority provide more waiting time savings than this strategy.

It is also interesting to find that cars gain benefits from buses being given priority. One reason for this is the availability of spare green time in junction in under saturation conditions. Using this spare green time in bus priority may not disbenefit cars. A study by Bretherton et.al. (1996) also found similar benefit to cars at junctions with lower saturation levels in London. Furthermore, the bus route modelled in this research passes through the main arm of most of the junctions. It follows that giving more green time to the bus route also benefits a greater number of cars than those in side roads. The change in the total amount of green time in the bus route and side road sides along after giving priority with different strategies are shown in Table 7.2.

Table 7.2: Change in total green time with different priority strategies

Priority	Total green time (in hours)		Change (in hours)	
	Bus Route	Side Road	Bus Route	Side Road
No priority	39.60	28.93	0.00	0.00
All buses	39.89	28.55	+0.18	-0.45
Late buses	39.76	28.68	+0.05	-0.32
Mixed	39.81	28.67	+0.10	-0.33

The total green time given is that calculated for all 11 signals along the route for a simulation period of nearly 10 hours. The changes in total green time in the 2-right-hand columns are obtained by subtracting the ‘no priority’ times from other ‘priority strategies’ times. The increase in total green time for the bus route, being the main arm of most of the junctions, supports the improved delay situation.

It is also to be noted that the contribution of waiting time savings is smaller. This is disadvantageous to the priority strategies improving waiting times rather than journey times. One of the reasons for this is the random arrival of passengers assumed in the model for this high frequency service route. Another reason is the low passenger demand in the route at this stage. The situation would change if the passenger demand were increased. This scenario is explored in Section 7.6.

7.3.2 Discussion

This scenario has shown that the strategy of giving priority to all the buses gives the highest overall benefit. However, strategies giving priority to late buses are better while considering punctuality and passenger waiting time benefit. Mixed priority, combining good points of both other priority strategies, closes the gap in overall benefit between them. It is to be noted that these results are based on the field practice of non-holding early buses and random arrival of passengers, which are not considered common in timetable services. These are particularly advantageous to the strategy giving priority to all buses. More discussion of the effects of the field situation in overall results is given in Chapter 8.

For the field condition assumed here, the strategy of giving priority to all buses is the best from an economic point of view; but with respect to bus punctuality and passenger waiting time, the mixed priority strategy is the best. The benefit from the different strategies may change if a higher degree of saturation (DOS) is set for the non-priority traffic arm while giving priority. Since changing the DOS level is possible in the field, this scenario is simulated next.

7.4 Effect of high DOS

The degree of saturation (DOS) for non-priority traffic stage at a traffic signal can be set at a higher level making more availability of recall time for the priority stage. The increase in priority amount improves bus journey times but it may be at the cost of higher delay to the non-priority traffic. The severity of the impact depends upon the level of Degree of saturation (DOS) and the number of priority awards made. The number of priority awards depends upon the type of priority strategy used. Hence this scenario is aimed to explore the impact using one high DOS level and different priority strategies.

Simulation was started by taking the same 3 priority strategies compared earlier. These were priority to 'all buses', priority to 'late buses' and 'mixed priority'. All the data used in the model were kept the same except the change in DOS and the recall amount allowed. The priority recall time in some of the traffic signals was increased by using target DOS for non-priority links as 110% rather than 95%. The increased recall amount at different traffic signals while using 110% DOS is given in Section 5.6.4. The analysis of results from simulation using these increased allowable recall amounts is given below.

7.4.1 Simulation results

Three simulation runs were carried out to cover 3 strategies mentioned above. The priority benefits obtained were then compared with the similar strategies using normal DOS. The change in priority benefits from different strategies, while changing the degree of saturation (DOS) level from normal to high, is shown in Figure 7.3.

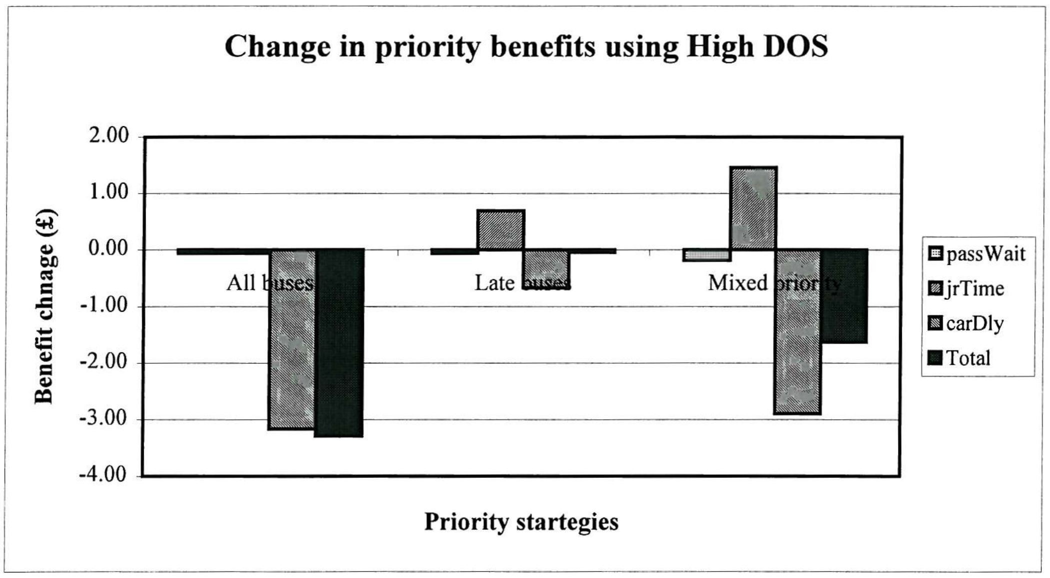


Figure – 7.3: Change in priority benefits while using High DOS

There is a decrease in overall benefits in all priority strategies while changing DOS from normal to high. However, there is an increase in bus related benefits in 'late buses' and 'mixed priority' strategies while not taking car delays into account. Despite some improvement in journey time, the total benefit from all the strategies is decreased due to the increase in car delays. The extent of the increase in car delay is lowest in the strategy giving priority to late buses and highest in the strategy giving priority to all buses. With this largest increase in car delays and having no increase in other benefits, the total benefit from 'priority to all buses' suffered the most. In the case of 'priority to late buses', the mild increase in car delays is counteracted by the increase in journey time benefit making total benefit unchanged. In the case of mixed priority, the disbenefit from the increase in car delays is counteracted by some increase in journey time benefit.

This clearly shows that the increase in car delays made the main impact when changing DOS from normal to high. The main reason for this increase in car delays is the type of recovery method opted after giving priority. The recovery method used in SIMBOL is 'Do Nothing (DN)' recovery (Section 5.6.6) that is one of four methods implemented in SCOOT (Bowen, 1997). In this method, the green time lost by side road stage during recall process is not compensated afterwards. However, by keeping the length of priority stage same, the side road green period starts earlier than usual. Even so, this will make the total amount of green time of the side road lower than in case of no priority. Though this has a little adverse effect in the case of unsaturated conditions, it worsens the situation in the case of over-saturated conditions. Additionally, SIMBOL is validated for unsaturated junctions and hence the results may be less accurate when a junction is over-saturated. This over-saturation may happen when a full recall amount is used with DOS setting higher than 100%. In such conditions, the vehicles will be carried over to subsequent cycles to clear and hence making car delays higher than in the field under SCOOT control.

The above figure also shows the improvement in bus journey time despite increased car delays in the case of high DOS. The increase in car delays was greater when giving priority to more buses. Hence it was thought that restricting priority to fewer buses would achieve journey time benefits with a minimum increase in car delays. For this purpose, the lateness of a bus at each upstream bus stop was taken into consideration. The lateness here used is the amount of time between departure time of a bus and its scheduled time in the last bus stop. This is positive if a bus is late and negative if a bus is early. In this case, another strategy giving high priority to a bus whose lateness is more than 60 seconds was considered. This is termed here as 'Late buses (>60)' priority. This is also a kind of 'differential priority' discussed in Section 2.1.3.

This idea of giving higher priority to buses that are more than 60 seconds late is then modified to incorporate both 'normal DOS' and 'high DOS' in a single priority strategy. In this strategy, priority using normal DOS is given to a bus with lateness between 0 and 60 seconds and priority using high DOS is given to a bus with lateness more than 60 seconds. This is another type of 'differential priority' strategy and here termed as 'Late buses (0&60)' priority.

This type of differential priority strategy is then used to form a new type of 'mixed priority' similar to one discussed earlier in Section 7.2.1.1. In this mixed priority, 'Late buses (0&60)' priority is used in the early part of the route and 'All buses (normal)' priority in the later part. This type of mixed priority is here termed as 'Mixed (differ & all)' priority. These 3 new strategies named 'Late buses (>60)', 'Late buses (0 & 60) and 'Mixed (Differ & all)' were simulated again using the same field data. The economic evaluation of results from simulations is tabulated in Table 7.3.

Table 7.3: Economic benefits from priority strategies using high DOS

	Economic benefits per hour (in £)				
Priority strategies	Pass wait time	Journey time	Bus total	Car Delay	Total
Late buses (>60)	0.69	3.33	4.03	-0.74	3.29
Late buses (0&60)	2.96	7.33	10.29	-1.21	9.08
Mixed (Differ & All)	2.71	10.37	13.08	1.75	14.83

The table shows that the 'Late buses (>60)' strategy gave the least benefit among all the priority strategies. The performance of 'Late buses (0 & 60)' strategy is better, giving the best passenger waiting time benefit. The 'Mixed (Differ & All)' strategy gave the best result among all the strategies using high DOS. The total benefit from this strategy is almost equal to the maximum benefit obtained earlier from the strategy giving priority to all buses using normal DOS. This benefit of 'Mixed (Differ & All)' strategy is achieved from good passenger waiting time benefit using 'Late buses (0 & 60)' strategy at the early part of the route and an increase in journey time benefit using 'All buses (Normal)' strategy in the later part.

7.4.2 Discussion

This scenario has shown that the priority strategies using higher DOS might not give better overall benefit to a system when used alone. Despite better journey time benefits, the strategies using high DOS gave higher car delays and hence may result in lesser total benefit. In such a situation, the selection of a priority strategy depends upon the aim of the scheme. If the focus is on buses rather than overall benefits, then the stronger priority strategy using high DOS should be selected. However, if the aim is to get a better overall benefit, then a priority strategy using only high DOS may not be suitable.

The result has shown that the strategies using high DOS in conjunction with normal DOS give better results than the strategies using high DOS alone. These strategies using different levels of DOS for different levels of lateness give the best passenger waiting time benefits along with good journey time benefits. These strategies, using high DOS to the buses with higher lateness, utilise the facility of high DOS more effectively than other strategies not differing in the use of high DOS from normal DOS.

Overall, the strategies using high DOS to 'All Buses' and 'Mixed priority (Late & all)' did not perform well when compared to similar strategies using normal DOS. Beside that, the strategy using high DOS to 'Late buses (>60)' also did not give good results. Hence, none of these 3 strategies were added to the list of priority strategies for further study. However, the strategy using high DOS to 'Late Buses' performed reasonably well when compared to the same strategy using normal DOS. Along with that, the strategies using both high DOS and normal DOS were found to be well when compared to the same strategy using normal DOS. Along with that, the strategies using both high DOS and normal DOS were found to be performing very well. Hence, these 3 strategies were added to the list of priority strategies. This made a total of 6 strategies including 3 from base case scenario and 3 added now. These 6 strategies selected for further simulations to explore their performance under different scenarios are shown in Table 7.4.

Table7.4: Priority strategies selected for further study

Strategy	Description
All buses (Normal)	Priority (Normal DOS) to all buses
Late buses (Normal)	Priority (Normal DOS) to late buses
Late buses (High)	Priority (High DOS) to late buses
Late buses (0&60)	Priority (Normal DOS) to late buses (>0 sec and <=60 sec) and priority (High DOS) to late buses (>60 sec)
Mixed (Late & All)	Priority (Normal) to late buses at early part and all buses (later part)
Mixed (Differ & All)	Priority (Late buses (0&60)) at early part and all buses (later part)

7.5 Effect of bus generation

It has been shown that buses in the field often deviate from their starting time specified in the timetable. The deviation in bus generation may be due to various factors, including weather conditions, road conditions, traffic conditions or operational problems. This deviation in starting times is one of the major sources making buses unpunctual. A small lateness at the origin becomes enlarged at each bus stop with more passengers than average, increasing dwell times and leading to more enlargement of the gap. This will conversely reduce the passengers for the following bus if on time thereby reducing dwell time and the gap. Hence the gap between the bus and the earlier bus to that will get enlarged whereas the gap between the following bus will get smaller. This may lead ultimately towards bus bunching.

The deviation in bus generation (lateness) at the origin may vary from day to day. Some days it may be very close to schedule whereas on others it may be greatly deviated. The effect of such deviation on the performance of a bus priority strategy may be considerable. Hence this scenario is simulated to explore the effects of different patterns of bus generation on priority strategies. In order to approach the field situation, a distribution similar to the lateness of buses at the origin in the field is used. The lateness profile of buses at their origin in the field is shown in Figure 7.4.

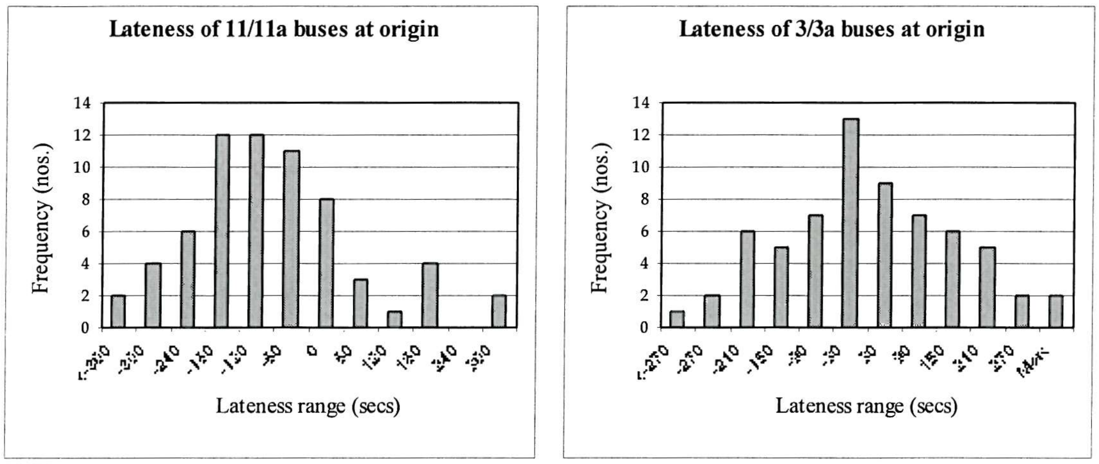


Figure – 7.4: Lateness profile of bus generation in the field

The shape of the lateness profile of buses at origin is close to the Normal distribution (This was verified by carrying out the 'goodness of fit' test using chi-square distribution. The difference between the Normal distribution and the field profiles was found to be not significant at the 95% confidence level). Retaining a normal distribution, 3 different bus generation profiles were generated by changing the mean and/or standard deviation as follows:

- N $(0,60^2)$ - Normal distribution (mean = 0 sec, standard deviation = 60 secs);
- N $(0,120^2)$ - Normal distribution (mean = 0 sec, standard deviation = 120 secs);
- N $(60,180^2)$ - Normal distribution (mean = 60 sec, standard deviation = 180 secs)

These 3 profiles are considered to represent 3 different types of days of bus operation in the field. Under good conditions, most of the buses will be generated on time; the mean and standard deviation will therefore be smaller and this is represented by N $(0,60^2)$. However it may not be possible to achieve this under normal conditions and buses may be generated with a larger standard deviation. This situation is represented by N $(0,120^2)$. In worse cases, the mean itself is shifted and standard deviation becomes even larger. This situation, where more buses are late and deviated is represented by N $(60,180^2)$. The results from the simulation runs and their discussion are found in the next sub-section.

7.5.1 Simulation results

In this scenario, 21 simulation runs were carried out to cover 7 different priority strategies in 3 different profiles of bus generation. Other model data defining the route, bus stops and signals were kept constant. Each type of generation was used to simulate all the priority strategies selected earlier. The results for these different profiles are given separately in 3 different tables below. Table 7.5a gives the priority benefit from different strategies while generating buses using N $(0,60^2)$ profile; Table 7.5b gives that of N $(0,120^2)$ profile; and Table 7.5c gives that of N $(60,180^2)$ profile.

Table 7.5a: Comparison of priority benefits for bus generation using N $(0,60^2)$ profile

	Economic benefits per hour (in £)				
Priority strategies	Pass wait time	Journey time	Bus total	Car delay	Total
No priority – total value	72.83	184.88	257.70	304.92	562.62
All buses (Normal)	+1.39	+14.68	+16.06	+1.21	+17.27
Late buses (Normal)	+2.02	+11.56	+13.58	+0.34	+13.92
Late buses (High)	+2.08	+11.69	+13.77	-2.22	+11.54
Late buses (0&60)	+1.70	+11.69	+13.39	-1.08	+12.31
Mixed (Late & All)	+2.02	+13.43	+15.45	+0.74	+16.19
Mixed (Differ & All)	+1.89	+13.50	+15.39	+1.62	+17.00

Table 7.5a represents the best base case scenario with least deviation amongst all. In this situation, the passenger waiting time is relatively low even in the base case (no priority case). Additionally, the mean being 0, almost half of the buses will be late and half early at origin. However, there are some more late buses in the later part of route due to the timetable design (Section 4.6.2.5). This makes only a little more than half of the buses getting priority from the strategies using the lateness check. Hence these strategies lag behind the strategy giving priority to all buses in terms of journey time.

So, the priority 'All buses (Normal)' performed well in this condition. However, it is to be noted that the performances of both types of 'Mixed priority' are also among the best.

Table 7.5b:Comparison of priority benefits for bus generation using N(0,120^2) profile

	Economic benefits per hour (in £)				
Priority strategies	Pass wait time	Journey time	Bus total	Car Delay	Total
No priority – total value	81.59	184.03	265.61	304.99	570.60
All buses (Normal)	-0.13	+15.40	+15.27	+1.95	+17.23
Late buses (Normal)	+2.77	+11.47	+14.24	+1.01	+15.25
Late buses (High)	+2.77	+10.97	+13.74	-1.28	+12.46
Late buses (0&60)	+2.77	+9.45	+12.22	-0.54	+11.68
Mixed (Late & All)	+2.65	+12.99	+15.64	+1.48	+17.12
Mixed (Differ & All)	+2.65	+10.72	+13.36	+1.82	+15.18

In the case of bus generation using the N (0,120^2) profile (Table 7.5b), buses are more deviated from their scheduled timetable. With more deviation in bus generation, the gaps between them are more uneven and hence there is increased passenger waiting time. The table shows an increase in passenger waiting time than the generation using N(0,60^2) profile. In such a situation, the lateness-based strategies perform better in terms of passenger waiting time benefit and improve the total benefits. In this case also, 'Mixed (Late & all)' priority is among the best strategies.

Table 7.5c:Comparison of priority benefit for bus generation using N(60,180^2) profile

	Economic benefits per hour (in £)				
Priority strategies	Pass wait time	Journey time	Bus total	Car Delay	Total
No priority – total value	86.12	180.04	266.16	304.92	571.08
All buses (Normal)	+0.00	+16.89	+16.89	+1.75	+18.64
Late buses (Normal)	+0.88	+12.81	+13.69	+1.48	+15.17
Late buses (High)	+1.20	+12.46	+13.66	-0.81	+12.85
Late buses (0&60)	+1.32	+12.97	+14.29	-0.61	+13.68
Mixed (Late & All)	+0.88	+13.50	+14.38	+1.48	+15.87
Mixed (Differ & All)	+1.39	+13.35	+14.73	+1.82	+16.55

With a change in both mean and deviation (Table 7.5c), buses are more late and deviated from their scheduled timetable; this further increases the passenger waiting time in the system. This enables lateness-based strategies to give priority to more buses. The performance of all these strategies is improved due to an increase in journey time benefits. However, with more buses getting priority, the improvement in passenger waiting time is lesser than earlier distributions. In this bus generation profile also, the performance of 'All buses' and 'Mixed' priorities are among the best. The collective comparison of these priority strategies with different types of bus generation is shown in Figure 7.5.

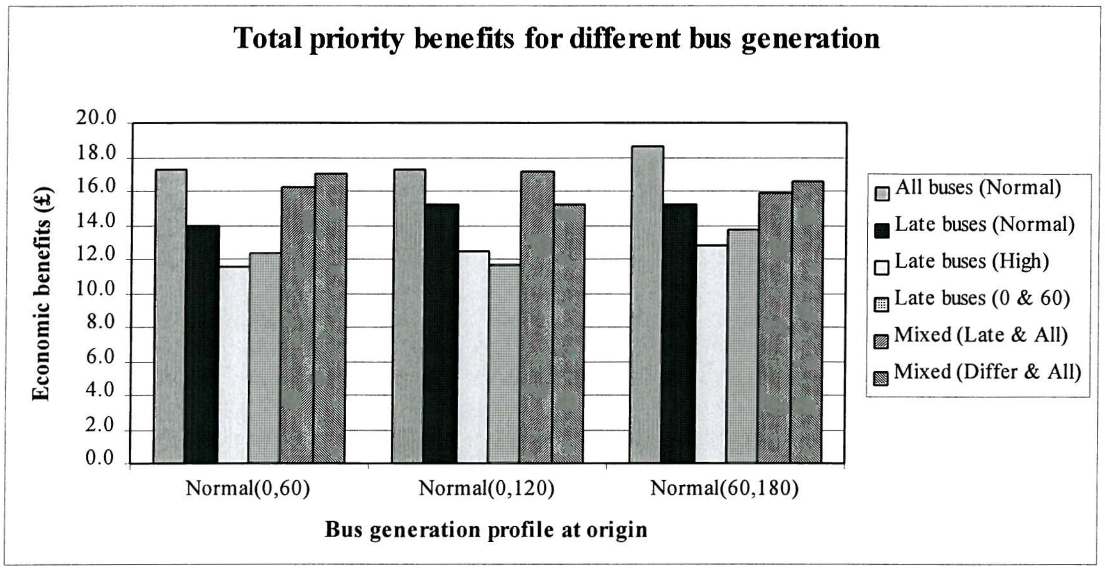


Figure – 7.5: Comparison of priority benefits for different bus generation profiles

The figure shows that the change in bus generation alters the performance of different priority strategies. The performance of the 'All buses' strategy is best with all 3 different profiles. The performances of both 'Late buses (Normal)' and 'Late buses (High)' strategies get better with deterioration in bus generation profile. In all types of generation profiles, 'Mixed' priority is among the best performing strategies.

7.5.2 Discussion

The result has shown that the strategies giving priority to late buses perform better in the case of system having more late buses. The performance is particularly enhanced by the greater number of buses getting priority and hence the improvement in journey time. The performance of these strategies are comparable to best performing 'All buses' strategy in such situations. The 'Mixed' priority is found to be performing very well in wider range of bus generation. Its strong performance came from strong passenger waiting time benefit along with moderate journey time benefit. Even then, the contribution of passenger waiting time benefit in total benefit is quite low. It only contributes around 10-20% of the total benefit of any strategy. This is mainly affecting the performance of strategies giving priority to late buses that improves the passenger waiting time rather than journey time. One of the reasons for this is the low passenger demand in the route at present. This scenario may change if the passenger demand is increased. The effect of change in passenger demand is therefore explored in next section.

7.6 Effect of increase in passenger demand

The field data showed that the passenger demand in the route is quite low. The total numbers of passengers boarding at all 16 bus stops were 378 in total over a 2-hour period. In average, only 25 passengers boarded each bus at 16 bus stops (with 15 buses per bus stop in average). The main effect of the low passenger demand is that the overall impact from bus priority can be influenced more by vehicular traffic than by its effect on bus passengers. In such a situation, the influence of passenger waiting time is overshadowed by bus journey time and car delays. This gives an advantage to the strategy giving priority to all buses over the strategy giving priority to the late buses only. Particularly because, in this case, cars benefit with increasing bus priority. The situation may be different if the passenger demand is higher. The increase in the passenger demand may happen due to better service punctuality or real time information systems or other initiatives that make bus journeys more attractive. In such a circumstance of higher passenger demand, the performance of priority strategies may be different. Hence this scenario is simulated to find the performance of different priority strategies under increased passenger demand situation.

7.6.1 Simulation results

In this scenario, 21 simulation runs were carried out with double the passenger numbers to cover 7 different priority strategies for each of 3 different types of bus generation. The results are tabulated in Table 7.6.

Table 7.6: Increase in performance parameters while doubling the passengers

	Increase in economic parameters per hour (in hours)				
Bus generation	Pass wait time	Journey time	Bus total	Car delay	Total
Field	92.04 (106%)	96.48 (54%)	188.52 (71%)	0.00	188.52 (33%)
N(0,60)	81.33 (112%)	97.47 (53%)	178.80 (69%)	0.07	178.87 (32%)
N(0,120)	87.38 (107%)	107.54 (58%)	194.92 (73%)	0.07	194.99 (34%)
Average	86.92 (108%)	109.76 (55%)	188.52 (71%)	0.04	187.46 (33%)

With double the passenger numbers arriving at bus stops, the change in performance parameters across all different types of bus generation is similar. With increase in passenger numbers, the main increase is in passenger waiting time and journey time. The average increase in waiting time is 108% whereas that of journey time is 61%. The bigger increase (more than 100%) in passenger waiting time is due to more passengers and an increase in dwell time at bus stops. A lower increase (less than 100%) in journey time is due to the same number of buses serving the route despite increased passenger demand and the unchanged occupancy at the beginning of the route. There is no change in car delay making more change in bus related benefits ('Bus total') than the overall benefits ('Total'). With the big increase in passenger waiting time and smaller in total, the effect of waiting time on total system cost increased considerably. Table 7.7 shows the passenger waiting time benefit and total priority benefits for different strategies under different types of bus generations.

Table 7.7: Priority benefits while doubling the passengers

	Priority benefit per hour (in £)					
Priority strategies	Field generation		Normal $(0,60^2)$		Normal $(0,120^2)$	
	passWait	Total	passWait	Total	passWait	Total
All buses (Normal)	3.84	24.87	6.61	21.37	3.53	20.72
Late buses (Normal)	5.17	16.88	8.32	19.92	7.37	15.48
Late buses (High)	3.53	13.12	8.13	20.06	7.31	17.24
Late buses (0&60)	3.53	13.07	8.19	21.93	6.43	16.64
Mixed (Late & All)	4.85	20.44	8.25	19.87	7.43	16.64
Mixed (Differ & All)	3.28	18.97	8.00	21.22	6.62	19.33

The table shows the greater contribution of passenger waiting time to total priority benefits in this case. The contribution is higher in priority strategies giving priority to late buses. This enabled the priority benefits from the lateness-based strategies to be comparable with 'All buses' priority. However, the 'All buses' priority is among the best even in this case of increased passenger demand. Additionally, the performance of 'Mixed (Differ and all)' priority strategy is among the best across all types of bus generation.

7.6.2 Discussion

This scenario has shown that the increase in passenger demand increases the influence of passenger parameters in a bus priority system. It considerably increases the total passenger waiting time and total journey time with no or little change in general traffic delays. The increase in passenger numbers increases the dwell time at bus stops that further increases total waiting time and journey time. This makes the overall priority benefit to be influenced by passengers rather than vehicles. This is advantageous to the priority strategies reducing average passenger waiting time. The simulation results have shown that the 'All buses' and 'Mixed priority' strategies give the best overall benefit in case of higher passenger demand.

7.7 Effect of holding early buses

Buses arriving early at bus stops may occur due to such factors as a change in the passenger demand, favourable traffic conditions or poor timetabling. The effect of an early bus is more severe than a late bus by the same amount, if passengers tend to arrive near the scheduled timetable. The passengers arriving on time miss the early buses and have to wait for the next bus to arrive. This increases passenger waiting time and deteriorates passenger confidence in bus arrival time at a bus stop. Hence it is desirable to avoid early running of buses. One such method to check early buses is by holding them at a bus stop. This is a simple method in which a driver stops at a bus stop until its scheduled departure time is reached. It is a simple way of making buses punctual but is relevant to timetabled services only.

Holding early buses makes buses punctual at downstream bus stops and hence improves average passenger waiting time. However, this method increases the journey time of passengers already inside the bus. Additionally, it underutilises the resources by wasting bus and bus driver's time. The field data revealed that almost half of the early buses were stopping to match their timetable whereas the other half did not take notice of their earliness at the Portswood bus stop. Hence it was thought to be necessary to explore the implication of holding buses on bus priority. This scenario was simulated to assess the impact of holding early buses at a main stop on different priority strategies.

7.7.1 Simulation results

The simulation started by taking 4 different base cases (no priority) based on the type of bus generation. The simulations were carried out by stopping buses at the Portswood bus stop, if found earlier than their scheduled timetable. The change in economic parameters for different types of bus generation, while holding early buses is shown in Figure 7.6.

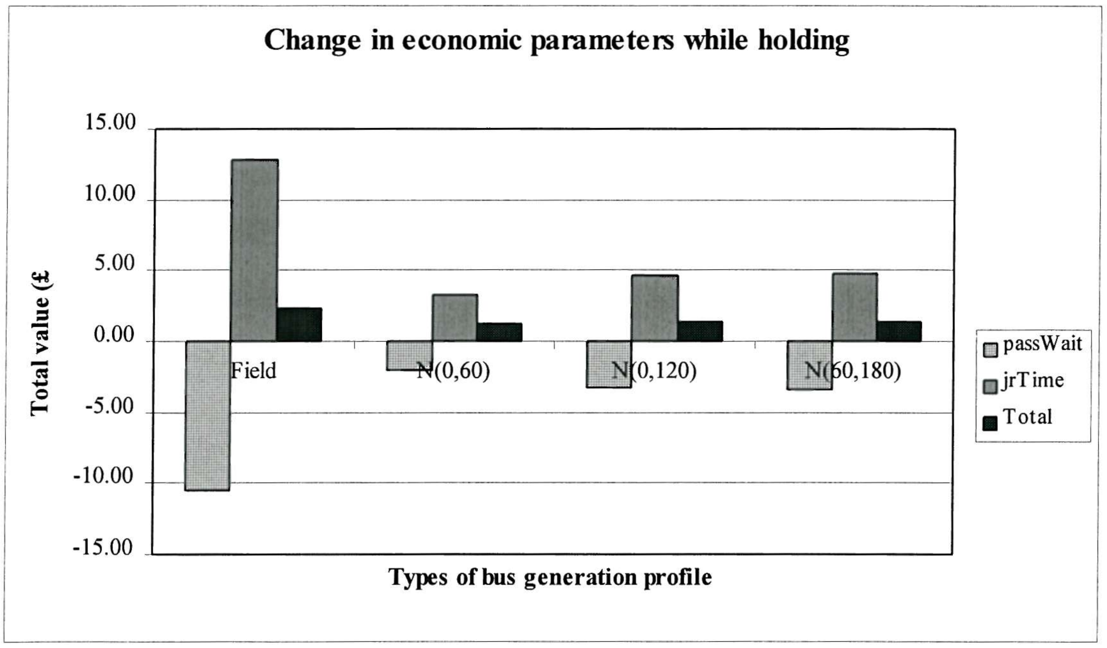


Figure – 7.6: Change in performance parameters while holding buses

There is a decrease in passenger waiting time and increase in journey time across all types of bus generation, when holding early buses. These changes are almost equally opposite in this case hence the total change in benefit is very small. However, the extent of these changes varied according to the type of bus generation. The extent is highest while generating buses according to field data and lowest while generating buses with N $(60,180^2)$ profile. The highest change in field data is due to the presence of large number of early buses (72% of 11/11a buses at their 'origin').

With this preliminary analysis, more simulations were carried out to explore the impact of holding on different priority strategies. All 7 priority strategies were simulated with generating buses according to field data. The change in main performance parameters while holding buses against non-holding is tabulated in Table7.8.

Table 7.8 Comparison of priority benefits while holding early buses

Priority strategies	Change in benefit of main parameters per hour (in £)		Total priority benefit per hour for holding (in £)
	Pass wait time	Pass Journey	Total
All buses (Normal)	0.50	-1.67	15.21
Late buses (Normal)	0.57	1.42	10.97
Late buses (High)	0.82	0.82	8.02
Late buses (0&60)	0.50	0.38	9.52
Mixed (Late & All)	0.50	-0.72	11.22
Mixed (Differ & All)	0.69	-1.76	12.05

While holding early buses, there is an increase in passenger waiting time benefit in all the priority strategies. By holding early buses and giving priority to buses, all priority systems make the gap between buses more even and improve passenger waiting time. Furthermore, there is an increase in journey time benefits from the strategies giving priority to late buses. However, there is a decrease in journey time benefit in 'All buses' priority strategies and 'Mixed' priority (which uses the 'all buses' priority strategy in the latter part of the route). By stopping all early buses at Portswood, it counteracts the priority given to the early buses. This action wastes the priority given to the early buses by 'All buses' priority and hence reduces the journey time.

7.7.2 Discussion

Holding is a simple strategy for making early buses punctual. By making buses more punctual, it improves the passenger waiting time but increases journey time. However, the influence of holding depends upon the proportion of early buses. The effect of holding is more with more early buses in the system and less with lesser early buses. One of the drawbacks of this method is that it is applicable to the timetable service only. In case of headway based service, there is no reference for a driver to check his/her 'earliness'. Furthermore, the effectiveness of the measure is dependent upon the driver perception and behaviour and hence susceptible to be error-ridden. And again, it requires regular standardisation of time for checking earliness of buses. However, the use of a digital display, showing present time and the scheduled time onboard, can avoid these time checking problems.

The above result and discussion has shown that the holding of early buses improves the waiting time of passengers. However, this action wastes the priority given to early buses in case of 'All buses' strategy. This reduces the journey time benefit from the strategy and makes it less attractive. In contrast, this method gives advantages to the strategies giving priority to the late buses in terms of both passenger waiting time benefit and journey time benefit. Hence, the combination of 'holding' option with the strategies giving priority to late buses was found to be working better. This scenario has shown the positive impact of 'holding' option on lateness based priority strategies for a timetable service. This supports the argument that 'holding early buses' should be normal practice in the case of timetable services.

7.8 Effect of GPS error

The error in a GPS system creates an uncertainty in the predicted position of a bus. The uncertainty in the position of a bus could have an impact on the bus priority system in which accurate bus position information plays a vital role. If the GPS location of a bus puts it further from the junction than it actually is, then the green extension time awarded would be greater than needed, and wasteful. Conversely, if the GPS location of a bus puts it closer to the junction than it actually is, then the green extension time awarded could be too short for the bus to clear the junction. A significant extra delay could then result. Especially, the positional accuracy is quite crucial while detecting a bus at a virtual detector using the GPS position of the bus. It is common to place a virtual detector just downstream of a bus stop. With the presence of GPS error, a bus may be detected at a virtual detector while being at a bus stop just upstream. In this case, the priority may be triggered/awarded while the bus is at the bus stop and any priority awarded could be completely wasted. To avoid this uncertainty, a safe distance (i.e. distance of maximum error - 10 metres in this case) between the virtual detector and bus stop is kept (Section 5.7.2). This reduced length of detection reduced the extension time allowed in some of the traffic signals. This amount is modelled as an extra time above the actual extension time required for buses. Hence this scenario is simulated to explore the performance of different strategies while incorporating GPS error.

7.8.1 Simulation results

This simulation was carried out by inserting a GPS error of maximum +/-10 metres. The GPS error was sampled from a normal random distribution with a mean 0 metre and standard deviation 3.3 metres obtained from earlier research (Rupprecht, 2001). A total of 6 simulation runs were carried out to model 6 different priority strategies with buses generated according to field data. The change in total priority benefits after the introduction of this GPS error is given in Table 7.9.

Table 7.9 Change in total priority benefits after introduction of GPS error

	Total priority benefits per hour (in £)		
Strategies	No GPS error	GPS error	Change
All buses (Normal)	17.37	16.79	-0.58
Late buses (Normal)	8.83	8.41	-0.41
Late buses (High)	8.79	8.21	-0.57
Late buses (0&60)	9.08	8.57	-0.51
Mixed (Late & All)	12.41	12.25	-0.16
Mixed (Differ & All)	14.83	14.11	-0.72

The result shows that the system with GPS error gives lesser benefit than the system without error. The total priority benefit is reduced in the range of 5% across the strategies. The reduction is mainly caused by a decrease in maximum extension amount allowed. Even in this scenario, the 'All buses' and 'Mixed (Differ and all)' strategies are the best performing strategies.

7.8.2 Discussion

It is expected to get a reduction in priority benefits once the GPS error is introduced. This is due to the reduction in priority time allowed due to the shifting of detectors at a safer distance from a bus stop and requirement of extra time. These are needed to ensure that a bus does not miss a priority awarded to it. Besides that, the comparison with a 100% accurate GPS system is not quite realistic because there are no 100% accurate GPS systems in use. However, the 100% accuracy modelled can be obtained in case of the system using detectors or beacons on the road.

While comparing with these 100% accurate system, their benefit is overshadowed by their rigidity in positioning. If it is needed to change the extension amount, then the detector should be reinstalled again. This can be easily done in the case of a GPS system by altering the virtual detector location in the onboard computer. Additionally, the GPS positioning of buses can be used in other areas such as 'passenger information at bus stops' and 'fleet management'. These are not possible with using detectors on the approaches of traffic signals only. These positive elements of a GPS system counterbalance the reduction in priority benefit while using it. With greater flexibility, wider areas of use and greater opportunity for combining different function/operation of a bus operation system, the use of GPS is appropriate.

7.9 Effect of change in operation

The route modelled in SIMBOL (Portswood corridor, Southampton) has a bus frequency of 6 or more buses per hour. A recent study (Rajbhandari, 2002) showed that the passenger arrival in this route is random (i.e. passengers do not take any notice of the timetable). This may be due to the frequent services, the non-punctual nature of the bus services, or both. Besides that, the present timetable is very unrealistic because of the repetition of the same timetable at up to 5-6 bus stops. This random arrival of passengers is a more common characteristic of headway-based bus services rather than a timetable service. In such a situation, it is interesting to explore the implications of bus services running under headway-based operation rather than timetabled. This scenario is intended to explore the implications of using headway-based service instead of timetable service.

7.9.1 Simulation results

The simulation was carried out by changing the bus operation from timetable-based to headway based. The headway between the buses was taken as the time interval between buses of similar bus service (i.e. buses of 11/11a) in the timetable-based operation. The standard headway between buses was 600 seconds. The scenario was simulated for 2 different types of bus generations with each having 5 different lateness based priority strategies. The bus generation profiles used were 'Field data' (having more early buses) and 'Normal(60,180^2) distribution' (having more late buses). The comparison of simulation results in both types of bus generations is tabulated in Table 7.10.

Table 7.10: Comparing benefits while changing from timetable to headway service

	Total priority benefits per hour (in £)			
	Field generation		Normal (60,180^2)	
Priority strategies	Timetable	Headway	Timetable	Headway
Late buses (Normal)	8.83	12.47 (+41%)	15.17	9.61 (-37%)
Late buses (High)	8.79	9.86 (+12%)	12.85	8.38 (-35%)
Late buses (0&60)	9.08	9.80 (+8%)	13.68	10.20 (-25%)
Mixed (Late & All)	12.41	16.23 (+31%)	15.87	12.58 (-21%)
Mixed (Differ & All)	14.83	14.94 (+1%)	16.55	13.47 (-19%)

The table shows two different patterns of results for 2 different types of bus generations. There is an increase in priority benefits while shifting to headway-based operation in the case of field generation. Conversely, the benefit is decreased in case of 'Normal(60,180^2)' generation. Even after the change, the 'Mixed' priorities are the best performing strategies. Here, the biggest change in priority benefit is in case of the

'Late buses (normal)' strategy in both types of bus generation. Hence this strategy is further analysed to explore the reasons behind the change in priority benefits. For both types of bus generation, the change in economic benefits of 'Late buses (Normal)' strategy while shifting from timetable-based to headway-based operation is shown in Figure 7.7.

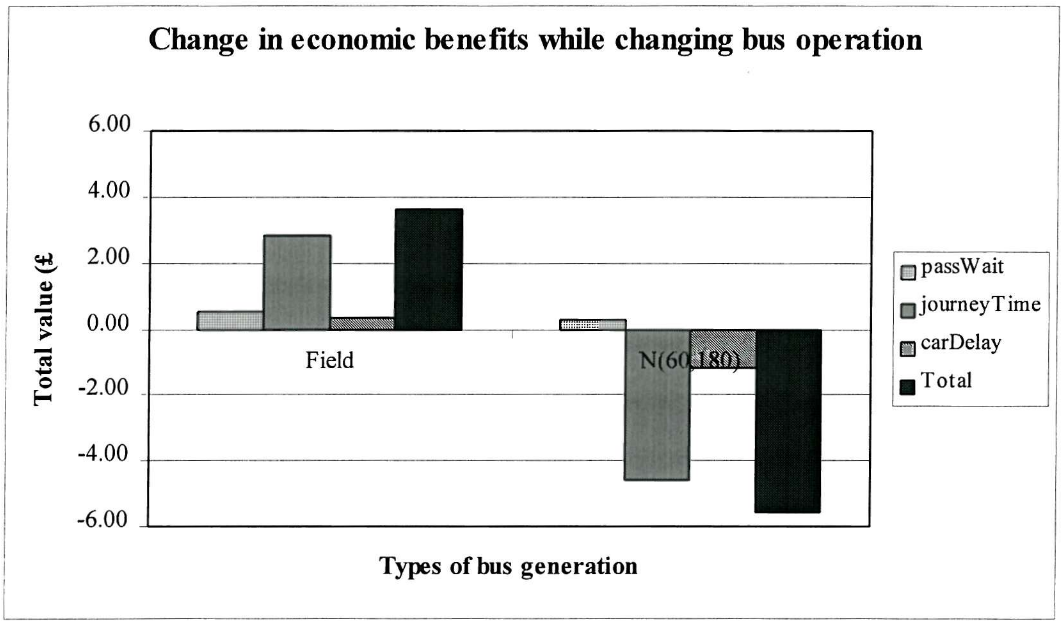


Figure – 7.7: Change in economic benefits of 'Late buses (Normal) strategy

The increase in priority benefits in case of field generation is contributed mainly by journey time and slightly by passenger waiting time. Since the strategy is targeted for bigger headway, the buses with more passengers get priority and hence there is an effective improvement in passenger journey time and waiting time. The decrease in priority benefits in the case of 'Normal(60,180^2)' generation is mainly due to the reduction in journey time benefit. However, even in this case, there is an improvement in waiting time of passengers due to better regularity. In this generation profile, there are more late buses in the system. Hence, while shifting to headway-based operation, some of the late buses with lesser gap (that would have got priority in case of timetable service) do not get priority. In such a situation, there are lesser buses getting priority than timetable service and hence a reduction in journey time benefit. The change in the number of priority awards made in both cases of bus generation while shifting from timetable to headway based operation is shown in Table 7.11.

Table 7.11: Number of priority awards under different types of bus operation

	Number of priority awards					
	Field generation			Normal(60,180)		
Priority type	Timetable	Headway	Change	Timetable	Headway	Change
Extension	65	66	1	95	59	-36
Recalls	109	113	4	175	126	-49
Total	174	179	5	270	185	-85

In the case of field generation, there is around only 3% increase in numbers of priority award made to the buses while changing the service from timetable to headway based. However, there is 46% reduction in numbers of priority award made to the buses in case of Normal (60,180) generation. This large reduction in numbers of priority award made is the main cause for the big decrease in journey time benefits. Hence the total priority benefit is reduced in this case of the system having more late buses. However, the bus priority is more in case of headway-based operation if the same number of buses gets priority. Furthermore, headway based priority is better for regularity of buses and improvement in passenger waiting time.

7.9.2 Discussion

With passengers arriving randomly at a bus stop, it was expected that the headway-based service would give better priority benefit than a timetable service. The better priority benefit from the system is due to targeting buses with more passengers (i.e. bigger headway). This was supported by simulation results. However, the priority benefit may be less if there are too many late buses if compared with timetable services. In normal case, headway-based priority is better for the system where passengers arrive randomly without taking notice of timetable. Though it is quite difficult to keep up the time, the timetable makes clear and easy for priority implementation. In case of priority implementation, the timetable can be set into the bus computer so that a bus can calculate its lateness itself. This will allow a bus to decide its priority requirement without referring to the AVL centre. This will cut the communication loss between bus and the AVL centre to ascertain the priority need and increase the amount of priority time.

7.10 Chapter summary

This chapter describes the application of SIMBOL in exploring different scenarios of bus priority in Portswood corridor bus route. Application of the model was started by simulating a base case scenario and taking 3 basic types of bus priority strategies. The strategies were further modified to use the high DOS facility available at traffic signals. The strategies were then simulated under different scenarios such as different types of bus generation, increase in passenger demand, holding early buses, error in GPS and change of bus operation. The results from these simulations were tabulated and discussed in respective sections. The overall discussion of these various strategies and different aspects of bus operation influencing bus priority is presented in Chapter 8.

Chapter Eight

Discussion

8.1 Introduction

Applications of SIMBOL to explore the performance of different bus priority strategies under various scenarios were described and analysed in Chapter 7. The strategies were compared and discussed under a particular scenario. In this chapter, all the strategies are collectively compared along with the discussion of field characteristics that might have an influence on the results.

8.2 Comparison of results

Chapter 7 showed that the performance of a strategy is often better in some aspects and weaker in some others. Hence, to ascertain the best performing strategy, all the strategies are compared collectively by using a performance rating between 'A' to 'D' based on total economic benefit. The ratings are: 'A' for priority benefit between £15-£20 per hour; 'B' for £10 - £14; 'C' for £5 - £9; and 'D' for less than £5 per hour. The performance rating of all strategies is given in Table 8.1.

Table 8.1: Comparison of different strategies under various scenarios

Strategy	Field data	N(0,60^2)	N(0,120^2)	N(60,180^2	Holding	GPS error	Headway
All buses (Normal)	A	A	A	A	A	A	A
Late buses (Normal)	C	B	A	A	B	C	B
Late buses (High)	C	B	B	B	C	C	B
Late buses (0&60)	C	B	B	B	B	C	B
Mixed (Late & All)	B	A	A	A	B	B	A
Mixed (Differ & All)	A	A	A	A	B	B	A

Notes:

Field data – Buses generated according to field data

$N(0,60^2)$ - Buses generated from Normal distribution (mean = 0 & SD =60 sec)

$N(0,120^2)$ - Buses generated from Normal distribution (mean = 0 & SD =120 sec)

$N(60,180^2)$ - Buses generated from Normal distribution (mean = 60 & SD =180 sec)

Holding – Stopping early buses at Portswood bus stop in place

GPS error – Inclusion of GPS error while locating buses for priority

Headway – Buses under headway based operation

The performance-rating table shows that the 'All buses' strategy gives the best economic benefit under all the scenarios. 'Mixed priorities' also shows a strong performance under many scenarios. The performance of the 'All buses' strategy is particularly better while using field data (containing more early buses) and that of 'Mixed priorities' and 'Late buses' are better in the case of more late buses e.g. generating buses with N $(60,180^2)$. Among the scenarios compared, the field scenario containing a large percentage of early buses is less likely to be the case in most situations. In normal conditions, buses generated according to N $(0,120^2)$ is more likely to be the field case rather than N $(0,60^2)$. Both 'holding' and 'GPS error' scenarios give possible benefits from the scenarios while generating buses according to field data. Hence their results also incorporate the shortcomings of the field data. In the circumstances of the high frequency service with random passenger arrival as found in the field, headway based operation is the most likely scenario in general. In these scenarios (i.e. 'N $(0,120^2)$' and 'Headway'), the performance of 'Mixed priority' is among the best (almost as good as 'All buses' priority). Beside the strong performance in terms of economic benefit, 'Mixed priority' is better in terms of punctuality/regularity consideration (than 'All buses' priority'). From these considerations, the 'Mixed priority' is the best strategy among all the strategies compared. The strategy comparisons in Table 8.1 are therefore specific to the corridor modelled and only partially transferable. However, it can be concluded that the strategies with differential priority are more expensive and complex to achieve, because of their infrastructure requirements.

It is to be noted that the passenger journey time makes the most contribution of around 50-70% in total economic benefit. The higher percentage contribution is in case of strategy giving priority to all buses. On other hand, passenger waiting time's contribution is in the range of 10-25% only. The upper limit of passenger waiting time is found in strategies other than 'All buses'. Furthermore, it is interesting to find the positive contribution of car delays benefit in some of the strategies including 'All buses' while giving bus priority.

The big contribution of passenger journey time, less influence of passenger waiting time and car delays benefit have given advantage to the strategy giving more priority to buses. In such a situation, a strategy that gives priority to all buses gets advantage over other strategies that give priority to late buses and make them more punctual. However, this is not representative of typical field conditions everywhere. Some of the field characteristics of the modelled route that might have an influence in these results are: random passenger arrivals, non-holding of buses, unsaturated junctions and bus route passing through the main arms at junctions. This analysis also does not account for any issues of passenger confidence (e.g. a punctual service may be more important to passengers than an improved service speed). These field characteristics are discussed in the context of the results below.

8.2.1 Random passenger arrival

In this research, the passenger arrival at a bus stop is assumed to be random making average passenger waiting time dependent upon the gap between buses. In this condition, early running buses with a regular gap do not cause any waiting time disbenefit. However, if it would have been the case that passengers arrive around the scheduled time then passengers arriving near the scheduled timetable miss early running buses and have to wait a very long time for next bus to arrive. This would cause a lot of passenger waiting time disbenefit in the system. This not being the case in the field, the 'All buses' strategy, giving priority to even early buses, has escaped the consequences. On the contrary, the other strategies making buses more punctual do not result in bigger passenger waiting time benefit. This is the reason for lesser passenger waiting time benefit and the weaker performances of the strategies making buses more punctual.

8.2.2 Non-holding of buses

Holding early buses should be a common practice in a timetable service so that no bus departs earlier than its scheduled timetable from a bus stop. However, the field practice in this study shows that buses in most of the bus stops do not take account of that at all. In such conditions, fewer buses get priority from strategies giving priority to late buses. However, this makes no difference to the 'All buses' strategy that gives priority to all buses. Furthermore, with non-holding of early buses, they are allowed to complete their journey as soon as possible. Hence the priority benefit from this 'All buses' strategy is higher than others in terms of journey time when there are more early buses. The field data revealed that 72% of buses of 11/11a service were early at the origin and in this case, 'All buses' gave the best benefit. However, while shifting the timetable by 120 seconds so that around 63% buses were late, 'Mixed' priority performed the best. The change in priority benefit while shifting the timetable by 120 seconds is given in Table 8.2.

Table 8.2: The change in priority benefit while shifting timetable by 120 seconds

	Priority benefits per hour (in £)		
Priority strategies	Field data	120 secs shift	Change
All buses (Normal)	17.37	17.37	0.00
Late buses (Normal)	8.83	13.92	+5.09 (58%)
Late buses (High)	8.79	12.30	+3.52 (40%)
Late buses (0&60)	9.08	14.52	+5.44 (60%)
Mixed (Late & All)	12.41	16.26	+3.85 (31%)
Mixed (Differ & All)	14.83	18.22	+3.39 (23%)

The priority benefit is increased while shifting the timetable to make more buses late. The increase in benefit is up to 60% of the earlier benefit in the case of the strategy giving priority to late buses. This big improvement is obtained due to more buses becoming eligible for priority that has increased the journey time savings. In this case, the maximum priority benefit is obtained from 'Mixed (Differ and all)' strategy. This is the maximum benefit obtained for field passenger numbers, among all the scenarios.

8.2.3 Unsaturated junctions

All the junctions modelled in this study are in unsaturated condition (see Section 3.4.2). The available spare green time in these junctions can be used in bus priority without severely disrupting other traffic. Hence the bus delay savings achieved in this study is in the upper limit of that found in earlier studies (Hounsell et al, 1996). These big delay savings at junctions has made the journey time savings the biggest contributor in economic analysis of priority benefit. This influential role of journey time gives an advantage to the 'All buses' strategy that gives priority to all buses and hence more journey time benefit. This situation would have been different if the junctions would be in higher saturation level. In such situation, the spare green time will be less and so will the journey time benefit and its influence on total benefit. Furthermore, a high number of traffic signals (11 signals in 4.32 kilometres) in the route further enhanced the benefits from journey time savings to contribute the most in overall priority benefits.

8.3.4 Bus route through the junctions

The bus route modelled in this research passes through the main arm of most of the junctions. In such conditions, general traffic gets benefit from increased green time due to priority to the buses. In this case, the more the priority award, the more is the benefit to general traffic. This gives a clear advantage to the strategy giving more priority i.e. strategy giving priority to all buses. However, the situation may be different if the bus route passes through the minor arms of the junctions. In this case, the priority given to buses may cause more disbenefit to other traffic than benefit.

8.2.5 Bus punctuality and passenger confidence

The punctuality of buses is a key criterion on which most of the timetable services are measured. The percentage of buses at a bus stop within the specified window of deviation (earliness/lateness) is the punctuality of buses. This is used as the performance measure of timetable services. The higher the punctuality, the better is the service. This also helps in improving passenger waiting time (where passenger arrive according to the timetable) and is helpful in developing passenger confidence. However, the value of punctuality and passenger confidence developed from it is not considered here in the economic evaluation of the priority benefit. This gives a significant disadvantage to the strategies giving priority to late buses and making buses more punctual. If this is also taken into account in the economic analysis, the performance of strategies would be different.

8.2.6 Bus timetable

The discussion of the prevailing field characteristics showed that the presence of a lot of early buses could have a big influence on the outcome of a bus priority strategy. Beside non-holding practice, another major factor for so many early buses in the field at origin is due to poor timetabling. In the field timetable, the journey time for the first part of the route (Swaythling to Portswood) is around 120 seconds less than the average journey time of buses (Section 4.6.2.5). So, most of the buses were starting early to be on time at Portswood bus stop which the main bus stop in the middle of the route. This resulted in very few buses getting priority from differential bus priority strategies and hence less priority benefits. This shows that a proper timetabling is an important issue while implementing bus priority at traffic signals. So, first step towards the bus priority implementation for a timetabled system should be the preparation of a proper timetable. Incorporating proper timetabling, a plan for implementing bus priority in timetabled service is shown in Figure 8.1.

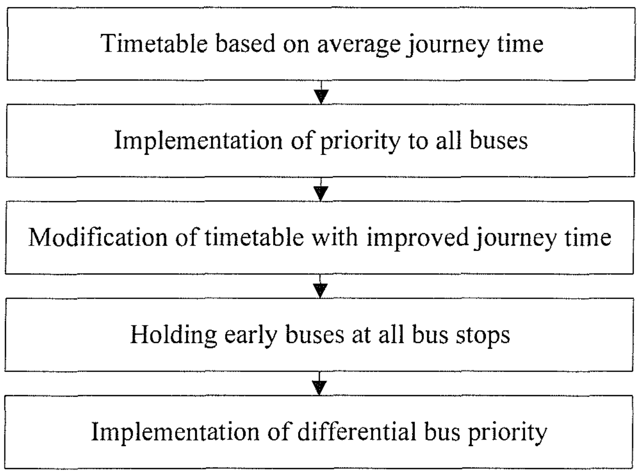

Timetable based on average journey time
Implementation of priority to all buses
Modification of timetable with improved journey time
Holding early buses at all bus stops
Implementation of differential bus priority

Figure 8.1: A plan for implementing bus priority in timetabled service

The plan starts with the preparation of timetable based on the journey time of buses without giving priority. The next step in this process is to give priority to all buses and obtain the improvement in journey time. Incorporating the improvement of the journey times, a modified timetable should be prepared and used for priority implementation. Then a differential bus priority strategy can be used for bus priority to account for within and between day variability in bus operations. In this case, beside the benefits discussed earlier (passenger waiting time, journey time etc.), there is a benefit of fleet management. The journey time saved from improved timetable (reduction in journey time) can be utilised by putting resources into other routes or services or by increasing frequency of the services (if necessary). This will help to improve the quality of bus services and minimise the operational costs, which may lead towards the increased patronage and help in modal shift from private to public transport. These other benefits of bus priority are not covered within the scope of this research. This area of fleet management can be further explored by modifying the existing model. This is one of the possible areas for further work.

8.3 Chapter summary

The overall comparison of the simulation results showed that the strategy giving priority to all the buses gives the highest overall benefit to a system in most cases. The performance of 'mixed priority' is also found to be among the best in many scenarios. Further discussion showed that 'All buses' priority is particularly favoured by the characteristics of the modelled route in this study. These characteristics are non-holding of buses, unsaturated condition of junction and passenger arrival profile. The discussion showed that the performance of a strategy might be different in different field conditions. This also pointed to the issue of suitability of a priority strategy depending upon the field condition. However, the performance of 'Mixed' priority, being the hybrid of two different types of strategies, is unaffected in most cases. The discussion also showed that a bus timetable plays an influential role in a bus priority system. Hence, a simple plan of priority implementation taking account of timetable improvement is formulated.

Chapter Nine

Summary and Conclusions

9.1 Introduction

This chapter summarises the work carried out during this PhD research work and draws out the main conclusions. The aim of this research was to investigate and develop recommendations for advanced bus priority strategies at traffic signals using a new simulation model developed within this research. The research began with a review of available literature on bus priority, priority mechanisms, options and available modelling tools. It was clear from the literature review that a new simulation model was necessary to meet the requirements of detailed modelling of advanced bus priorities at traffic signals incorporating Automatic Vehicle Location (AVL) systems. The modelling methodology developed was based on the specific research requirements and available literature. The data collection was then carried out along the Portswood corridor bus route in Southampton. A simulation model was built based on the field data, then verified and validated. The completed model was used to explore the performance of different priority strategies under different scenarios. The simulation results were then discussed in the context of the modelled route. The key findings from the application of the model are given in this chapter. The chapter starts with a description of the key features of the model, followed by the findings from the application and possible areas for future research.

9.2 Main features of the model

One of the main objectives of this research was to develop a simulation model for detailed study of advanced bus priority strategies at traffic signals. In order to achieve this, a microscopic simulation model, SIMBOL (Simulaton Model for Bus priOrity at traffic signal), was developed. SIMBOL is capable of simulating a bus route taking account of buses, bus stops, traffic signals, AVL systems and advanced bus priority depending on lateness and target degree of saturation (DOS) at junctions. The delays to general traffic and impacts of priority on general traffic are also modelled in SIMBOL. The main modelling features of SIMBOL are summarised in Table 9.1.

Table 9.1: Modelling features of SIMBOL

Component	Characteristics	Methods
Bus system	Bus operation Overlapping services	Timetable, headway Multiple origin-destination
Bus	Generation Movement	Timetable, Distribution Average link journey time
Bus stop	Passenger generation Alighting passenger Waiting time Dwell time calculation Holding early buses	Regular interval, Distribution % of passenger inside Average, individual basis York's, field parameters Optional
Traffic signal	Cycle time Bus delays General traffic delays	Fixed time Individual basis By generation and discharge of cars
Bus priority	Priority methods Priority strategies	Extension and recall Selective detection, Differential and Mixed priority
AVL system	GPS based system Detection	Error sampled from Normal random distribution Virtual detectors, detectors
Input	Model building data Simulation data	Built in (can be user defined) User defined for each simulation
Output	Visual Text files	Shows buses, traffic lights and traffic queues Concise and detailed files

Application of the model was focussed on simulating the range of different scenarios affecting the performance of a bus priority strategy. Various bus priority strategies under different scenarios were simulated. The main findings from the application and discussion of the results are in next section.

9.3 Main findings of the research

The application of the model to the simulation of different scenarios is described in chapter 7. The simulation was based on a field route with a high frequency (but timetabled) bus service (10 minute frequency), unsaturated junctions, low passenger volume and random passenger arrivals. Based on the analysis of results and discussion from these simulations, the following conclusions can be drawn:

- Bus priority at traffic signals is a useful measure to give effective priority to buses in urban areas where there are a number of traffic signals present. There is a wide range of priority strategies for giving priority to buses.
- Giving priority to all buses is the simplest form of bus priority strategy at traffic signals. By giving priority to all buses, this strategy gives the best journey time benefits of all available priority strategies. However, this strategy by making early buses even earlier is the least preferred in terms of bus punctuality and passenger waiting time.
- Mixed priority giving priority to late buses at the starting part of the route and all buses at the latter part, is found to be among the best strategies under all different scenarios. The performance of this priority is found to be most resilient due to a combination of the best points of other strategies. It gives very good overall benefit, including good punctuality and passenger waiting time benefit.
- One of the factors affecting priority strategy is lateness in starting of buses at origin. The strategy giving priority to all buses is favoured by more early buses at the start. The differential priorities, being targeted at late buses, are found to be more effective when there are more late buses than early.
- Holding of early buses at bus stops is a simple measure that considerably improves the waiting time of passengers. However, this action increases journey times and wastes the priority given to early buses. In this sense, 'holding' works better with strategies giving priority to late buses only.

- The error in GPS system used for bus location reduces the bus priority benefits. One of the main reasons for this is the reduction in priority extension time to ensure that a bus does not miss a priority extension provided. Despite this reduction in benefit, its flexibility and multi purpose application makes a GPS system worth using in the strategies giving priority to late buses.
- Shifting from a high frequency timetable service to a headway-based operation is found to benefit systems with passengers arriving randomly at bus stops. The headway-based priority is better because it makes buses more regular (i.e. even gaps between buses) and hence improves the passenger waiting time.
- A proper timetabling is an important factor while implementing bus priority in a timetabled service. A bus timetable could have a big influence on the outcome of a bus priority strategy. So, the preparation of a proper timetable is the first step towards the bus priority implementation for a timetabled system. The timetable should then be modified to take advantage of the benefits provided by bus priority.

Overall, the simulation results show that the strategy giving priority to all buses is better in terms of overall benefit but not in terms of punctuality and passenger waiting time. The strategies giving priority to late buses are better in terms of punctuality and passenger waiting time but not in terms of total benefit. Hence the selection of best-suited priority strategy depends upon whether the priority scheme is aiming to achieve total benefit or better punctuality. Between these extremes, mixed priority is the best strategy giving very good overall benefit combined with better punctuality and passenger waiting time.

9.4 Possible areas for further work

The research concluded with the development of a simulation model and its application in simulating different scenarios to explore the performance of advanced bus priority strategies at traffic signals. The new simulation model has opened up possible areas for further work. This may include both further application of the existing model and development of the model itself to model other issues which it does not yet cater for. Both of these possible areas of further work are described below.

9.4.1 Further applications of the current model

This research was carried out using a system with random passenger generation as is the case for high frequency services. Since the model is able to model individual passenger generation, there is an opportunity to explore bus priority implications for a low frequency timetabled service. The passenger arrival rate is then higher near to the bus arrival time. This can be modelled once a relation/distribution is defined. More applications of the model are as follows:

- One of the scenarios modelled here was headway-based bus operation along the existing modelled route. This facility of modelling headway-based bus operation could be further used to explore a system by taking an actual example of a true headway-based bus operated system.
- The overlapping route in the present study was modelled by considering only identical services. The model can also be applied to situations where there is a choice of services. This will depend upon the field data collection of passenger O-D matrix and characteristics of bus services.
- The study was carried out taking one out of several priority architectures available. Different priority architectures with their communication lags can be modelled and explored using the model.
- In the present study, only 4 types of bus generation profiles were simulated. More bus generation profiles representing day-to-day variations in field conditions can be modelled for further exploration.
- The model can be used to model bus priority in a fixed-time UTC (Urban Traffic Control) such as the SPRINT system being trialled in London.
- The modelling of timetable modifications in response to improvements generated by bus priority and its implications in terms of fleet management and resource utilisation.

9.4.2 Further development of the model

The model at this stage is capable of modelling on a route basis. There may be wider issues such as conflicting priority requirements at junctions and passenger route choice, once a network level of bus operation is considered. Further development of the model is needed to model a bus operation system at network level. This would be a more representative model of the field condition.

There is a practice of running buses continuously around a route, i.e. Origin-Destination-Origin. In such cases, any deviation at the starting time at origin of the first leg may affect the starting at the second leg (returning leg) of the journey. Modelling this continuous circulation of buses on a route would be useful to explore possible bus fleet management and optimisation. This requires further model development, which is possible within the framework of the current model.

The present modelling of journey time modelling between a detector and downstream traffic signal is based on the average speed of the bus obtained from the field data collection. However, there may be a considerable variation in this portion of the road due to queuing vehicles. A further work is necessary to model this variation in journey time of buses near the signalised junctions. Along side, more detail modelling of junction can be carried out using varying level of signal timings and traffic flows. However, this requires significant amount of further development of the model. Additionally, the model can be further developed to explore the new methods and technologies (i.e. the use of multiple detection of buses using GPS system while giving priority). These show the diverse range of potentials the model has for further developments.

References

AGARWAL P. K., JAIN S. S., KHANNA S. K., 1994, Development of a Simulation Model for Performance Evaluation of Bus Operation on Urban Transportation Corridor, Proceedings of 36th Annual Conference of Transportation Research Forum, Florida, pp759-76

AL-SAHAILI K. A., TAYLOR W. C., 1996, Evaluation of Bus Priority Signal strategies in Ann Arbor, Michigan, Transportation Research Record. 1996. (1554) pp74-79

BAGOT N., 1999, Try Before You Buy: Trends in Traffic Simulation, Traffic Technology International, pp68-74

BERNAUER E., BREHERET L., ALGERS S., BOERO M., TARANTO C. D., DOUGHERTY M., FOX K., GABARD J. F., 1997, SMARTEST: Review of Micro-Simulation Models, Institute for Transport Studies, University of Leeds, United Kingdom

BOWEN G. T., 1997, Bus priority in SCOOT, TRL Report 255, Transport Research Laboratory, Old Wokingham Road, Crowthorne, United Kingdom, page 8-11

BRETHERTON D., HOUNSELL N., RADIA B., 1996, Public Transport Priority in SCOOT, Proceedings of the Third World Congress on Intelligent Transport Systems, Orlando, Florida, 1996, pp71

CALTABIANO R., CAMUS R., GERIN R., ONGO G., 1997, Implementation of an Advanced AVM (Automatic Vehicle Monitoring) System for the Trieste Bus Network, Proceedings of Seminar K: Traffic Management and Road Safety, 25th PTRC European Transport Forum, Brunel University, Volume P419, pp9-17

CASSIDY S., 1995, " What you don't know won't hurt you!": The Impacts Of Automatic Vehicle Location Systems on Bus Operations and Planning, Proceedings of Seminar D: Public Transport Planning and Operations, 23rd European Transport Forum, University of Warwick, 11-15 September, pp. 35-48.

CHANG G. L., VASUDEVAN M., 1995, Modeling and Evaluation of Adaptive Bus-Preemption Control With and Without AVL Systems, Proceedings of the 6th International VNIS, 30 July - 2 August 1995, Washington, USA, pp305-316.

CHENEY C. N., 1992, Keeping Buses Moving, Proceedings of Seminar D: Public Transport Planning and Operations, 20th European Transport Forum, PTRC, pp. 129-140.

CPT, 2001, The Factors Affecting Bus Reliability, Confederation of Passenger Transport, Imperial House, London.

CZOGALLA O., HOYER R., 1997, Simulation Based Design of Control Strategies for Urban Traffic Management and Control, Proceedings of the 4th World Congress on Intelligent Transport Systems, 21-24 October 1997, Berlin (Paper No. 2265)

DAVIES R. M., O'KEEFE R. M., 1989, Simulation Modelling with Pascal, First edition, Prentice Hall International (UK) Group, pp 146-147.

DEPARTMENT OF THE ENVIRONMENT, TRANSPORT AND THE REGIONS, 1997, Keeping Buses Moving: A Guide To Traffic Management To Assist Buses In Urban Areas, Local Transport Note LTN 1/97, The Stationery Office, 49 High Holborn, London, Wc1v 6hb, United Kingdom.

DEPARTMENT OF THE ENVIRONMENT, TRANSPORT AND THE REGIONS, 1998, A New Deal for Transport: Better for Everyone, The Government's white paper on the future of transport, HMSO, London, United Kingdom.

DRUITT S., 1998, An Introduction to Microsimulation, Traffic Engineering and Control, Vol. 39 (9), pp480-483

FELLENDORF M., 1996, VISSIM for Traffic Signal Optimisation, Traffic Technology International '96, 1996, pp190-192

GARMIN INC., 2000, GPS Guide for Beginners, GARMIN International, Inc, Kansas, U.S.A.

HEN2, 1997, Highway Economics Note No. 2, Section 2, Values of Time and Vehicle Operating Costs, Chapter 2, Design Manual for Roads and Bridges, Volume 13, November 1997, Highways Agency, HMSO

HILL R., 2000, Real Time Passenger Information and Bus Priority System in Cardiff, From Vision to Reality: Proceedings of the 7th World Congress on Intelligent Transport Systems, 6-9 November 2000, Turin, Italy

HOLMAN S., WILLUMSEN L., 1991, Computer Assisted Design of Bus Priority Schemes, Proceedings of Seminar H: Public Transport Planning and Operations, 19th PTRC European Transport Forum, University of Sussex, 9-13 September 1991, pp. 79-91

HOUNSELL N., WALL G., 2002, Examples of New Intelligent Transportation Systems Applications in Europe to Improve Bus Services, Proceedings of 2002 Annual Meeting of Transportation Research Board, Washington, 10-12 January 2002, paper no 02-3451.

HOUNSELL N. B., MCLEOD F. N., GARDNER K., HEAD J. R., COOK D., 2000, Headway-Based Bus Priority in London using AVL: First Results, Proceedings Of 10th International Conference On Road Transport Information And Control, London, 4-6 April, 2000, IEE Conference Publication NO. 472, pp218-222

HOUNSELL N., MCLEOD F., 1999, Automatic Vehicle Location and Bus Priority: The London System, World Transport Research, Selected Proceedings of the 8th World Conference on Transport Research, Volume 2, Planning, Operation, Management and Control, pp. 279-292

HOUNSELL N., MCLEOD F., 1998, Automatic Vehicle Location Implementation, Application and Benefits in the United Kingdom, Transportation Research Record 1618, pp155-162

HOUNSELL N. B., BOWEN G. T., COOK D. J., GARDNER K., 1997, SPRINT: Active Bus Priority in Fixed Time UTC in London, Proceedings of Seminar K: Traffic Management and Road Safety, 25th PTRC European Transport Forum, Brunel University, 1-5 September 1997. Volume P419, pp75-86.

HOUNSELL N. B., MCLEOD F. N., LANDLES, J. R., GARDNER K., 1996, Bus Priority in London: Building on Prompt, Proceedings of the Third World Congress on Intelligent Transport Systems. Location: Orlando, USA, October 1996

HOUNSELL N. B., MCLEOD F. N., BRETHERTON R. D., BOWEN G.T., 1996, PROMPT: Field Trial and Simulation Results of Bus Priority in SCOOT, Proceedings of International Conference on Road Traffic Monitoring and Control, IEE Publication No. 422, 23-25 April 1996, pp 90-94

HOUNSELL N. B., LANDLES J. R., 1995, Public Transport Priority at Traffic Signals In London: Progress, Performance And Opportunities, Proceedings Of The Second World Congress On Intelligent Transport Systems '95 Yokohama. Volume 1. 1995/11. Pp273-8

HUNT, P. B., et. al, 1981, SCOOT: A Traffic Responsive Method for Co-ordinating Signals, Transport and Road Research Laboratory Report LR 1014, Crowthrone, UK

IBRAHIM D., 2000, Improving Accuracy for GPS Vehicle Navigation Systems in London, Traffic Engineering and Control, June 2000, pp 228-232.

INSTITUTE OF HIGHWAY AND TRANSPORT, 1987, Roads and Traffic in Urban Areas, HMSO, pp. 214- 228

INSTITUTE OF HIGHWAY AND TRANSPORT, 1997, Transport in the Urban Environment, HMSO, pp. 329- 348

KHASNABIS, S., KARNATI, R. R., RUDRARAJU, R. K., 1996, NETSIM-Based Approach to Evaluation of Bus Preemption Strategies, Transportation Research Record. 1996. (1554) pp80-89

KING, G. N., 1992, London-Wide Bus Priority – Achievable in the 1990's?, Proceedings of Seminar D: Public Transport Planning and Operations, 20th European Transport Forum, PTRC, pp. 153-164

KONTARATOS M., BALIS V., LIAPAKIS C., 1996, GPS-Based AVL System: A Tool for Supporting Public Transport Operation in the Urban Area, Proceedings of the Third World Congress on Intelligent Transport Systems. Location: Orlando, Florida, 1996. pp102-

KRAKIWSKY E. J., 1995, Analysis of Automatic Vehicle Location and Navigation Systems Built Worldwide, Proceedings of 2nd World Congress on Intelligent Transport Systems '95 Yokohama, November 9-11, 1995, Vol. V, pp 2216- 2220

LIU R., CLARK S., MONTGOMERY F., WATLING D., 1999, Microscopic Modelling of Traffic Management Measures for Guided Bus Operation, Proceedings of 8th World Congress on Transport Research, Volume – 2, Planning, Operation, Management and Control, Pergamon, 1999, pp367-80

LOBO A. X., 1998, A Review of Automatic Vehicle Location Technology and Its Real-Time Applications, Transport Reviews, 1998, Vol. 18, No. 2, pp165-191

MCLEOD F., 1998, Headway-Based Selective Priority to Buses, Mathematics in Transport Planning and Control, Pergamon, pp. 69-78

MIORANDI J., CAMPBELL J., 1997, Waterworks Road Intelligent Bus Priority Pilot, Proceedings of the 4th World Congress on Intelligent Transport Systems, 21-24 October 1997, Berlin (Paper No. 2265)

NICKEL B. E., 1997, Telematic Applications in German Public Transport, Mobility for Everyone: Proceedings of the 4th World Congress on Intelligent Transport Systems, 21-24 October, 1997, Berlin, 9 pages

OAKES J. A. J., THELLMANN A. M., KELLY I. T., 1994, Innovative Bus Priority Measures, Proceedings of Seminar J: Traffic Management and Road Safety, 22nd PTRC European Transport Forum, Vol. 381, pp. 153-164

OCHIENG W.Y., SAUER K., 2002, Urban Road Transport Navigation: Performance of the Global Positioning System after Selective Availability, Transportation Research, Part C, 2002, pp 171-187

PARSONS D., 1997, Object Oriented Programming with C++, 2nd Edition, Letts Educational, Aldine Place, London

PRISCILLA, 2001, Bus Priority Strategies and Impact Scenarios Development on a Large Urban Area, Deliverable 2, Public Transport Priority: State of the Art Review, March 2001, pp. 66-67,108

RAJBHANDARI B., 2002, Modelling Intelligent Transport System Applications for Public Transport, PhD Thesis, Department of Civil Engineering, University of Southampton, pp. 113-114

ROBERTSON D. I., VINCENT R. A., 1975, Bus Priority in a Network of Fixed Time Signals, Transport and Road Research Laboratory Report LR666, Crowthorne.

RUDNICKI A., 1997, Measures of Regularity and Punctuality in Public Transport Operation, Transportation Systems, Preprints of 8th IFAC/IFIP/IFORS Symposium, Chania, Greece, 16-18 June, 1997.

RUPPRECHT W., 2001, Post SA GPS Accuracy Measurements, wolfgang@charlotte.wsrcc.com

SALTER R. J., HOUNSELL N. B., 1996, Highway Traffic Analysis and Design, 3rd edition, McMillan Publishing Company, UK

SALTER R. J., SHAHI J., 1979, Prediction of Effects of Bus-Priority Schemes by Using Computer Simulation Techniques, Transportation Research Record No. 718, pp1-5

SANTHAKUMAR S. M., HARIHARA P., 1992, Transportation Systems Management Options to Improve Urban Bus Route Performance Using Computer Simulation, Transportation Research Record No. 1338, pp22-27

SEDDON P. A., DAY M. P., 1974, Bus passenger Waiting Times in Greater Manchester, Traffic Engineering and Control, January 1974, pp 442- 445

SHRESTHA B. P., 2002, Simulating Differential Bus Priority at Traffic Signals for Improving Bus Regularity, 34th UTSG Annual Conference, Transport Research Institute, Napier University, 3-5 January 2002, Volume II, paper no. 46

SMITH M. W., NELSON, J. D., BELL, M. G. H., DICKINSON, K. W., 1994, Developing the Concept of Buses as Probes: The Integration of Automatic Vehicle

Locationing and Urban Traffic Control Systems, 7th IFAC/IFORC Symposium on Transportation Systems, TianJin, China, 1994, pp637-42

SOUTHAMPTON CITY COUNCIL, 1999, Public Transport Development study, Summary Report, March 1999

TRIPS Version Seven Manual, 1996, MVA Systematica, 1996

UNIVERSITY OF SOUTHAMPTON, 1988, Evaluation of SELKENT Bus Priority Scheme, Final Report to the Traffic Control Systems Unit, London

UNIVERSITY OF SOUTHAMPTON, 1996, AVL/Bus Priority Feasibility Study, Final Report to London Transport Buses

UNIVERSITY OF SOUTHAMPTON, 1997, Bus Priority at Traffic Signal: AVL/BP Headway Algorithm, Final Report to London Transport Buses

WILLOUGHHBY P., EMMERSON P., 1999, Network Interaction – A Review of Existing Modelling Techniques, Traffic Engineering and Control, Vol. 39(2), pp81-82

WREN A., 1996, ROMANSE – Road Management System for Europe, Proceedings of Seminar H : 24th PTRC European Transport Forum, Vol. 407

WREN A., 1996, ROMANSE - Information Dissemination, Proceedings of the Third World Congress on Intelligent Transport Systems, Orlando, Florida, 1996. pp287-288

YORK I. O., 1993, Factors Affecting Bus-Stop Times, TRL Project Report 2, T1/25, Transport Research Laboratory, Crowthorne, UK

APPENDIX A: Data Collection Forms

Table A2: Data collected using handheld computer

```
  Route 3, Portswood to Town Centre

  FILE NAME = WW094927.13
DATE IS 13 11 2001
  14.Bus stop        Somerfield
 Arrival    time   9 49 48  5
 Arrival    time  10  4 54 98
 Departure time  10  6 15 94
  15.Traffic signal  Brookvale Road junction
 Arrival    time  10  6 44 83
 Departure time  10  6 45 66
  16.Bus stop        Safeway
 Arrival    time  10  7  3 56
 Departure time  10  7 43  0
  17.Bus stop        Spring Crescent
 Arrival    time  10  8 27 38
 Departure time  10  8 55 33
  18.Traffic signal  Lodge Road junction
 Arrival    time  10  9 11 10
 Departure time  10  9 15 44
  19.Bus stop        Cedar Road
 Arrival    time  10 10 14 92
 Departure time  10 10 15 53
  20.Traffic signal  Stag Gates junction
 Arrival    time  10 11  6 39
 Departure time  10 11 40 22
  21.Bus stop        Stag Gates
 Arrival    time  10 11 54 23
 Departure time  10 12  6  9
  22.Bus stop        Middle Street
 Arrival    time  10 12 26 58
 Departure time  10 12 27 13
  23.Bus stop        Law Courts
 Arrival    time  10 13 15 35
 Departure time  10 13 40 51
  24.Traffic signal  Cumberland Place junction
 Arrival    time  10 15 17 40
 Departure time  10 15 22 45
  25.Bus stop        Cenotaph
 Arrival    time  10 15 46 73
 Departure time  10 16  4  3
  26.Traffic signal  New Road junction
 Arrival    time  10 16 51 15
 Departure time  10 17 57  1
  27.Bus stop        Marlands
 Arrival    time  10 18 24 31
 Departure time  10 18 56 66
  28.Bus stop        Pound Tree
```

Table A3: Manual data collection inside a bus

Portswood - Lodge Road - City Route (3)

WW0904927.13

Date : 13/11/2001

Time : 10:03

Day : Sunny Cold

Bus: Single (white) wpx Already = 22

S.N.	Type	Name	Dist	Arrival	Alight	Board	Departure
1	Bus stop	Portswood/Somerfield	1800.0	10:04:54	10	7	10:06:15
2	Traffic signal	Brookvale road Jn	1960.0		-	-	
3	Bus stop	Safeway	2060.0		1	3	10:07:43
4	Bus stop	Spring crescent	2410.0		0	1	
5	Traffic signal	Lodge road Junction	2440.0		-	-	
6	Bus stop	Cedar road	2620.0		0	0	
7	Traffic signal	Stage gate Junction	2950.0		-	-	
8	Bus stop	Stage gate	2990.0		1	1	
9	Bus stop	Middle street	3210.0		0	0	
10	Bus stop	Law court	3490.0		3	0	
11	Traffic signal	Cumberland place Jn	3870.0		-	-	
12	Bus stop	Cenotaph	4010.0		3	1	
13	Traffic signal	New road Junction	4230.0		-	-	
14	Bus stop	Marland	4320.0	10:18:24	12	0	
15	Bus stop	Pound tree	4540.0	—	—	—	-
					0.0	0.0	

Transferred ✓
Checked ✓

Table A4: Signal stage data from SCOOT

Hour	Minute	Secs		Junction	Stage	Intergreen	Green	Total
9	59	15	11	N04141	3	6	23	29
9	59	58	11	N04141	1	9	34	43
10	0	11	11	N04141	2	6	7	13
10	0	39	11	N04141	3	6	22	28
10	1	26	11	N04141	1	9	38	47
10	1	39	11	N04141	2	6	7	13
10	2	10	11	N04141	3	6	25	31
10	2	54	11	N04141	1	9	35	44
10	3	7	11	N04141	2	6	7	13
10	3	35	11	N04141	3	6	22	28
10	4	14	11	N04141	1	9	30	39
10	4	27	11	N04141	2	6	7	13
10	4	51	11	N04141	3	6	18	24
10	5	38	11	N04141	1	9	38	47
10	5	51	11	N04141	2	6	7	13
10	6	18	11	N04141	3	6	21	27
10	6	58	11	N04141	1	9	31	40
10	7	11	11	N04141	2	6	7	13
10	7	39	11	N04141	3	6	22	28
10	8	18	11	N04141	1	9	30	39
10	8	31	12	N04141	2	6	7	13
10	9	0	11	N04141	3	6	23	29
10	9	34	11	N04141	1	9	25	34
10	9	47	11	N04141	2	6	7	13
10	10	21	11	N04141	3	6	28	34
10	10	57	11	N04141	1	9	27	36
10	11	10	11	N04141	2	6	7	13
10	11	42	11	N04141	3	6	26	32
10	12	17	11	N04141	1	9	26	35
10	12	30	11	N04141	2	6	7	13
10	12	55	11	N04141	3	6	19	25
10	13	41	11	N04141	1	9	37	46
10	13	54	11	N04141	2	6	7	13
10	14	22	11	N04141	3	6	22	28
10	14	57	11	N04141	1	9	26	35
10	15	10	11	N04141	2	6	7	13
10	15	39	11	N04141	3	6	23	29

Table A5: Junction flowdata from SCOOT

Hours	Minutes	Seconds	Tick	SCN	period	stp	Dly*10	flow	cong	raw	flts
10	0	30	12	N04141A	300	109	15	121	0	0	0
10	5	30	12	N04141A	300	54	9	58	0	0	0
10	10	30	12	N04141A	300	90	11	91	0	0	0
10	15	30	12	N04141A	300	93	9	93	0	0	0
10	20	30	12	N04141A	300	108	13	109	0	0	0
10	25	30	12	N04141A	300	136	12	137	0	0	0
10	30	30	12	N04141A	300	66	7	66	0	0	0
10	35	30	12	N04141A	300	85	11	87	0	0	0
10	40	30	12	N04141A	300	100	6	102	0	0	0
10	45	30	12	N04141A	300	96	10	103	0	0	0
10	50	30	12	N04141A	300	64	4	70	0	0	0
10	55	30	12	N04141A	300	83	4	93	0	0	0
11	0	30	12	N04141A	300	39	2	47	0	0	0
11	5	30	12	N04141A	300	113	8	127	0	0	0
11	10	30	12	N04141A	300	90	8	91	0	0	0
11	15	30	12	N04141A	300	113	8	113	0	0	0
11	20	30	12	N04141A	300	151	9	155	0	0	0
11	25	30	12	N04141A	300	123	10	129	0	0	0
11	30	30	12	N04141A	300	91	7	94	0	0	0
11	35	30	12	N04141A	300	98	8	108	0	0	0
11	40	30	12	N04141A	300	165	17	165	0	0	0
11	45	30	11	N04141A	300	103	7	103	0	0	0
11	50	30	12	N04141A	300	126	14	135	0	0	0
11	55	30	12	N04141A	300	140	9	140	0	0	0
12	0	30	12	N04141A	300	191	20	194	0	0	0
12	5	30	12	N04141A	300	159	16	159	0	0	0
12	10	30	12	N04141A	300	143	16	157	0	0	0
12	15	30	12	N04141A	300	168	15	183	0	0	0
12	20	30	12	N04141A	300	135	9	141	0	0	0
12	25	30	12	N04141A	300	171	24	172	0	0	0
10	0	30	12	N04141B	300	483	29	529	0	0	0
10	5	30	12	N04141B	300	155	7	427	0	0	0
10	10	30	12	N04141B	300	378	18	511	0	0	0
10	15	30	12	N04141B	300	360	19	459	0	0	0

Table A6: Junction delay data from SCOOT

Time		Junction	Speed Mile	Speed	Flow	Occ	Hr	Sr	Count
10:00:00	10000011	04141A	19	30	13	3	9031	160	10
10:05:00	10050011	04141A	20	32	8	1	11270	130	10
10:10:00	10100011	04141A	20	32	11	2	9870	130	10
10:15:00	10150011	04141A	16	25	8	2	10665	135	10
10:20:00	10200011	04141A	14	22	12	2	9041	158	10
10:25:00	10250011	04141A	18	28	14	2	8955	144	10
10:30:00	10300011	04141A	14	22	9	2	9898	101	10
10:35:00	10350011	04141A	11	17	9	2	9875	125	10
10:40:00	10400011	04141A	14	22	9	2	10055	165	10
10:45:00	10450011	04141A	19	30	13	2	9160	140	10
10:50:00	10500011	04141A	11	17	9	1	9728	71	10
10:55:00	10550011	04141A	17	27	11	2	9782	117	10
11:00:00	11000011	04141A	8	12	6	1	10543	56	10
11:05:00	11050011	04141A	19	30	13	2	9542	157	10
11:10:00	11100011	04141A	20	32	12	2	9762	137	10
11:15:00	11150011	04141A	19	30	12	2	9090	110	10
11:20:00	11200011	04141A	24	38	18	3	7605	190	10
11:25:00	11250011	04141A	28	45	17	3	7835	175	10
11:30:00	11300011	04141A	12	19	8	1	10982	117	10
11:35:00	11350011	04141A	22	35	13	3	9778	221	10
11:40:00	11400011	04141A	27	43	20	3	7501	198	10
11:45:00	11450011	04141A	15	24	11	2	9101	98	10
11:50:00	11500011	04141A	19	30	18	3	7890	150	10
11:55:00	11550011	04141A	17	27	14	2	8826	173	10
12:00:00	12000011	04141A	22	35	21	4	6829	210	10
12:05:00	12050011	04141A	27	43	18	3	8285	214	10
12:10:00	12100011	04141A	28	45	21	3	7942	157	10
12:15:00	12150011	04141A	22	35	19	4	7468	172	10
12:20:00	12200011	04141A	19	30	14	3	8613	186	10
12:25:00	12250011	04141A	25	40	22	4	6527	172	10
10:00:00	10000011	04141B	22	35	53	13	3303	289	10
10:05:00	10050011	04141B	25	40	45	9	2960	219	10
10:10:00	10100011	04141B	24	38	53	10	2822	229	10
10:15:00	10150011	04141B	25	40	45	9	2846	226	10

APPENDIX B: Source Code of SIMBOL

```
//---------------------------------------------------------------------------
#ifndef InputDialogFH
#define InputDialogFH
//---------------------------------------------------------------------------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TInputDialog : public TForm
{
__published:
	TButton *OKBtn;
	TButton *CancelBtn;
	TBevel *Bevel1;
	 TEdit *editSimPeriod;
	 TLabel *Label2;
	 TLabel *Label3;
	 TEdit *editSimSpeed;
	 TRadioButton *rbNoPrior;
	 TRadioButton *rbSVDPrior;
	 TRadioButton *rbAVLPrior;
	 TLabel *Label4;
	 TLabel *Label5;
	 TLabel *Label6;
	 TRadioGroup *rgAvl;
	 TCheckBox *cbPass;
	 TLabel *Label1;
	 TComboBox *cbLogicNos;
	 TLabel *lblPriorityStrategy;
	 TCheckBox *cbCar;
	 TLabel *Label7;
	 TCheckBox *cbSignal;
	 TLabel *Label8;
	 TLabel *lblPollingFrequency;
	 TEdit *editPollingFrequency;
	 TLabel *lblGpsAccuracy;
	 TEdit *editGpsAccuracy;
	 TLabel *lblOdometerAccuracy;
	 TEdit *editOdometerAccuracy;
	 TCheckBox *cbHolding;
	 TLabel *Label9;
	 TCheckBox *cbDoublePass;
	 TLabel *Label10;
	 TCheckBox *cbGpsError;
	 TLabel *Label11;
	 void __fastcall FormCreate(TObject *Sender);
	 void __fastcall rbNoPriorClick(TObject *Sender);
	 void __fastcall rbAVLPriorClick(TObject *Sender);
	 void __fastcall rbSVDPriorClick(TObject *Sender);
	 void __fastcall editSimPeriodKeyPress(TObject *Sender, char &Key);
	 void __fastcall rgAvlClick(TObject *Sender);
	 void __fastcall editSimSpeedKeyPress(TObject *Sender, char &Key);
	 void __fastcall editPollingFrequencyKeyPress(TObject *Sender, char
&Key);
private:
public:
	virtual __fastcall TInputDialog(TComponent* AOwner);
	 int newPriorityOption;
	 int newSimPeriod;
	 int newSimSpeed;
	 int newPollingFrequency;
	 int newGpsAccuracy;
	 int newOdometerAccuracy;

	 int newLogicNumber;
```

```
        int newAvlSystem;
        int holdingBusstop;
        int gpsError;
        float doublingPassenger;
        int newPassGenerateMode;
        int signalPhasesVisible;
        int carGeneratedVisible;
        bool __fastcall  Execute (void);
};
//---------------------------------------------------------------------------
extern PACKAGE TInputDialog *InputDialog;
//---------------------------------------------------------------------------
#endif
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "InputDialogF.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TInputDialog *InputDialog;
//---------------------------------------------------------------------------
__fastcall TInputDialog::TInputDialog(TComponent* AOwner)
      : TForm(AOwner)
{
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::FormCreate(TObject *Sender)
{
newSimPeriod=36400;//29100;//7590;//7390;//7570;//7415;
newSimSpeed=100000;      //Default=1000
newPollingFrequency=1;//20; //Default=30 secs
newGpsAccuracy=0;       //Default=10
newOdometerAccuracy=100;
}
//---------------------------------------------------------------------------
bool __fastcall TInputDialog::Execute(void)
{
 editSimPeriod->Text=IntToStr(newSimPeriod);     //These are the start values
 editSimSpeed->Text=IntToStr(newSimSpeed);
 editPollingFrequency->Text=IntToStr(newPollingFrequency);
 editGpsAccuracy->Text=IntToStr(newGpsAccuracy);
 editOdometerAccuracy->Text=IntToStr(newOdometerAccuracy);
 cbLogicNos->ItemIndex=1;//2;//3;//2;

 ActiveControl=editSimPeriod;

 if(ShowModal () == mrOk)
 {
  if(rbNoPrior->Checked)  newPriorityOption=0;
  else if(rbSVDPrior->Checked) newPriorityOption=1;
  else  newPriorityOption=2;

  if(rgAvl->ItemIndex==0) newAvlSystem=1;
  else if(rgAvl->ItemIndex==1) newAvlSystem=2;
  else {}

  if(cbHolding->Checked)  holdingBusstop=7;
  else   holdingBusstop=99;
  if(cbGpsError->Checked)  gpsError=1;
  else   gpsError=0;
  if(cbDoublePass->Checked)  doublingPassenger=2.0;
  else   doublingPassenger=1.0;
  if(cbPass->Checked) newPassGenerateMode=1;
  else newPassGenerateMode=0;
  if(cbSignal->Checked) signalPhasesVisible=1;
  else signalPhasesVisible=0;
  if(cbCar->Checked)  carGeneratedVisible=1;
  else   carGeneratedVisible=0;

  newSimPeriod=StrToInt(editSimPeriod->Text);
  newSimSpeed=StrToInt(editSimSpeed->Text);
  newPollingFrequency=StrToInt(editPollingFrequency->Text);
  newGpsAccuracy=StrToInt(editGpsAccuracy->Text);
  newOdometerAccuracy=StrToInt(editOdometerAccuracy->Text);
  newLogicNumber=(cbLogicNos->ItemIndex)+1;
  return true;
 }
 else return false;
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::rbNoPriorClick(TObject *Sender)
{
 rgAvl->Visible=false;
 cbLogicNos->Visible=false;
 lblPriorityStrategy->Visible=false;
 lblPollingFrequency->Visible=false;
 editPollingFrequency->Visible=false;
 lblGpsAccuracy->Visible=false;
 editGpsAccuracy->Visible=false;
 lblOdometerAccuracy->Visible=false;
 editOdometerAccuracy->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::rbSVDPriorClick(TObject *Sender)
{
 rgAvl->Visible=false;
 cbLogicNos->Visible=false;
 lblPriorityStrategy->Visible=false;
 lblPollingFrequency->Visible=false;
 editPollingFrequency->Visible=false;
 lblGpsAccuracy->Visible=false;
 editGpsAccuracy->Visible=false;
 lblOdometerAccuracy->Visible=false;
 editOdometerAccuracy->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::rbAVLPriorClick(TObject *Sender)
{
 rgAvl->Visible=true;
 rgAvl->ItemIndex=0;
 cbLogicNos->Visible=true;
 lblPriorityStrategy->Visible=true;
 lblPollingFrequency->Visible=true;
 editPollingFrequency->Visible=true;
 lblGpsAccuracy->Visible=true;
 editGpsAccuracy->Visible=true;
 lblOdometerAccuracy->Visible=false;
 editOdometerAccuracy->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::rgAvlClick(TObject *Sender)
{
 if(rgAvl->ItemIndex==0)
 {
  lblPollingFrequency->Visible=true;
  editPollingFrequency->Visible=true;
  lblGpsAccuracy->Visible=true;
  editGpsAccuracy->Visible=true;
  lblOdometerAccuracy->Visible=false;
  editOdometerAccuracy->Visible=false;
 }
 if(rgAvl->ItemIndex==1)
 {
  lblPollingFrequency->Visible=false;
  editPollingFrequency->Visible=false;
  lblGpsAccuracy->Visible=false;
  editGpsAccuracy->Visible=false;
  lblOdometerAccuracy->Visible=true;
  editOdometerAccuracy->Visible=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::editSimPeriodKeyPress(TObject *Sender,char
&Key)
{
 if((Key<'0')||(Key>'9'))
 {
  MessageBeep(0);
  Key='\0';
 }
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::editSimSpeedKeyPress(TObject *Sender,char &Key)
{
 if((Key<'0')||(Key>'9'))
 {
    MessageBeep(0);
    Key='\0';
  }
}
//---------------------------------------------------------------------------
void __fastcall TInputDialog::editPollingFrequencyKeyPress(TObject
*Sender,char &Key)
{
 if((Key<'0')||(Key>'9'))
 {
  MessageBeep(0);
  Key='\0';
 }
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
#ifndef MainSimFH
#define MainSimFH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>

#include "BusList.h"
#include "BusStop.h"
#include "Link.h"
#include "Bus.h"
#include "Signal.h"
#include "Detector.h"
#include "Beacon.h"
#include "VirtualDetector.h"
#include "InputDialogF.h"

#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TMainSim : public TForm
{
__published: // IDE-managed Components
        TButton *StartBtn;
        TButton *CloseBtn;
        void __fastcall StartBtnClick(TObject *Sender);
        void __fastcall CloseBtnClick(TObject *Sender);
private:        // User declarations

        int enterSimTime;
        int tempBusNumber;
        int routeLength;
        int windowWidth;
        int simulationPeriod;
        float timeFactor;
        int simulationSpeed;
        int busGenerateTime;
        int  pollingFrequency;
        int gpsAccuracy;
        int odometerAccuracy;
        int holdBusStop;
        float passengerRateFactor;

        int priorityOption;
        int logicNumber;
        int avlSystem;
        int passGenOption;
        int signalVisible;
        int carVisible;

        int totalBusstopNo;
        int totalSignalNo;
        int totalLinkNo;
        int totalDetectorNo;
        int totalBeaconNo;
        int totalVirtualDetectorNo;

        void RouteParameters(int startLag,float passRateFactor);
        void DummyParameters();
        void DrawRoute();
        void DrawSignalPhase();
        void ChangeSignalPhase(int times);
        void DrawDetectorBeacon();
        void DisplaySimTime(int SimTime);
        int CountDigit(int number);

        void ChangeBusNumber(int busnos);
        int StartBusNumber();
        void Simulate(int time,int busNumber,float busPosi2,int holdBusStop);
        void MoveBus(int oldpos, int newpos, int nosBuses);
        void GenerateCar(int times);
        void DrawCar();
        void DrawPass(int times);
        void ChangeLinkPara(float busPosi2);
        void CheckBusstops(int times,int busNumber,float busPosi2,int
holdBusStop);
        void CheckSignals(int times,int busNumber,float busPosi2);
        void CheckSVDPriority(int times,int busNumber,float busPosi2);
        void CheckAVLPriority(int times,int busNumber, float avlBusPosi2);
        void CheckBeacons(int times,int busNumber,float busPosi2);

        float AvlSystem(int time,int busNumber,float busPosi2);
        int RandomBusGenerate();

        void OutputHeaders();
        void OutputFile(int busNumber,float passRateFactor);
        void BusGenerate(int option, int times);

        void BusData(int time);

        BusList* list;
        BusStop*busstop[16];
        Signal*signal[40];
        Link*link[28];
        Detector*detector[12];
        Beacon*beacon[5];
        VirtualDetector*vDetector[12];
        Bus*currentBus;
        BusStop* currentBusstop;
        Signal* currentSignal;
        Link* currentLink;
        Detector* currentDetector;
        Beacon* currentBeacon;
        VirtualDetector* currentVDetector;

        int stopOut[16][300];//150];  more simu time

public:             // User declarations
        __fastcall TMainSim(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TMainSim *MainSim;
//---------------------------------------------------------------------------
#endif
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include <stdio.h>
#include "MainSimF.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainSim *MainSim;
FILE* stream,*stream1,*stream2,*stream21,*stream3,*stream4,*stream5,*stream6,
*stream7,*stream8,*stream9,*stream10,*stream11;
//---------------------------------------------------------------------------
__fastcall TMainSim::TMainSim(TComponent* Owner)
        : TForm(Owner)
{
 stream11 = fopen("ComparativeChart.txt", "w+");
 fprintf(stream11, "\n \n %s","Comparative chart of different bus   priority
options");
 fprintf(stream11, "\n \n%s %s %s %s %s %s %s %s","SimTime",
"priority","TotPass","  TotPassWait"," TotBusJr"," TotPassJr","
TotCarDly","TotCost");
 fprintf(stream11, "\n %s %s %s %s %s %s %s %s", "(secs)", " (Nos)","
(Nos)","      (hrs)","      (hrs)","     (hrs)","      (hrs)", "      (£)");
 fclose(stream11);
}
//---------------------------------------------------------------------------
void __fastcall TMainSim::StartBtnClick(TObject *Sender)
{
 if(InputDialog->Execute())    //for Dialog Box
 {
  priorityOption=InputDialog->newPriorityOption;
  simulationPeriod=InputDialog->newSimPeriod;
  simulationSpeed=InputDialog->newSimSpeed;
  logicNumber=InputDialog->newLogicNumber;
  if(priorityOption!=2) logicNumber=0; //to avoid priority logic for No/SVD
  avlSystem=InputDialog->newAvlSystem;
  pollingFrequency =InputDialog->newPollingFrequency;
  //gpsAccuracy=InputDialog->newGpsAccuracy;
  gpsAccuracy=InputDialog->gpsError;
  odometerAccuracy =InputDialog->newOdometerAccuracy;
  holdBusStop=InputDialog->holdingBusstop;
  passengerRateFactor=InputDialog->doublingPassenger;
  passGenOption=InputDialog->newPassGenerateMode;
  signalVisible=InputDialog->signalPhasesVisible;
  carVisible=InputDialog->carGeneratedVisible;
 }
 else Close();
//---------------------------------------------------------------------------
 //Initialising
//---------------------------------------------------------------------------
 tempBusNumber=1;
 routeLength = 4325; //Marland is at 4320 & 5 m extra for Visual Correction
 windowWidth = 1000;          //default =1000
 timeFactor =1.0;             //default = 1.0 which divides 1 sec to get
update inetrval 40.0 corresponds to real time
 int generateFactor1=0;
 int generateFactor2=0;
 int generateFactor3=0;
busGenerateTime=600;//300;          //default =300   needed for differential
//---------------------------------------------------------------------------
 BusList* list = new BusList();      //Defining List for buses
 RouteParameters(0,passengerRateFactor);//100);    //100 seconds lag
 DrawRoute();         //Drawing roads, Signals, Busstops and place names
 OutputHeaders();
//---------------------------------------------------------------------------
 //Start of Simulation - Generating Buses at Origin according to time
//---------------------------------------------------------------------------
 randomize();
 //int startTime=1000;
 int simuperiod=simulationPeriod;
```

```
  enterSimTime=simuperiod;
  for (int time = 0; time <simulationPeriod; time++)
  {
   DisplaySimTime(time/timeFactor); //Displays time
   float realTime = (time/timeFactor);
   fprintf(stream, "\n %c %6.2f %c", ' ', realTime, ' ');
   fprintf(stream8, "\n %c %6.2f %c", ' ', realTime, ' ');
   fprintf(stream9, "\n %c %6.2f %c", ' ', realTime, ' ');
   if (realTime<999) fprintf(stream21, "\n %5.1f",realTime);
   //GenerateCar(time/timeFactor);
   ChangeSignalPhase(time/timeFactor);         //Changing Traffic Signal phase
   GenerateCar(time/timeFactor);
   if(signalVisible==1)DrawSignalPhase();      //Drawing Traffic Signal phase
   if(carVisible==1)DrawCar();                 //Drawing cars at the signals
   if(passGenOption==1)DrawPass(time/timeFactor);        //Drawing passengers
//----------------------------------------------------------------------------
   if(time<simuperiod)  //for not generating buses after simulation period
   {
    int option=100;//00;//100;
    if(option==00)
    {
     //generates buses in a fixed time interval
     if (time==(generateFactor1*busGenerateTime*timeFactor+60+45))
     {
      int routeSpecifier=generateFactor1%2;
      if(routeSpecifier==0) list->addBus(new
Bus("Bus1",111,18.0,0.0,90,0,0,0,0,9,8,8,8,10,2450.0)); //Service 11a
      if(routeSpecifier==1) list->addBus(new
Bus("Bus1",110,18.0,0.0,90,0,0,0,0,15,30,11,11,10,4330.0)); //Service 11
      generateFactor1++;
     }
     if (time==(generateFactor2*busGenerateTime*timeFactor+300+45))
     {
      int routeSpecifier=generateFactor2%2;
      if(routeSpecifier==0) list->addBus(new
Bus("Bus2",30,18.0,1800.0,90,7,7,7,7,15,30,11,11,22,4330.0));//Service 3
      if(routeSpecifier==1) list->addBus(new
Bus("Bus2",31,18.0,1800.0,90,7,7,7,7,9,8,8,8,22,2450.0)); //Service 3a
      generateFactor2++;
     }
     if (time==(generateFactor3*1800*timeFactor+1140+45))
     {
      list->addBus(new
Bus("Bus2",1010,18.0,1800.0,90,7,7,7,7,9,8,8,8,15,2450.0)); //Service 101
      generateFactor3++;
     }
    }

    if(option==100)
    {
     stream110 = fopen("busGenTime1.txt", "r");
     int temp1;
     char msg1[20];
     fseek(stream110, 0, SEEK_SET);
     for (int i =0; i<30; i++)
     {
     fgets(msg1,10,stream110);
     temp1 = atoi(msg1);
     if (time==temp1) list->addBus(new
Bus("Bus1",110,18.0,0.0,90,0,0,0,0,15,30,11,11,10,4330.0));
     }
     fclose(stream110);

     stream111 = fopen("busGenTime2.txt", "r");
     int temp2;
     char msg2[20];
     fseek(stream111, 0, SEEK_SET);
     for (int i =0; i<31; i++)
     {
     fgets(msg2,10,stream111);
     temp2 = atoi(msg2);
     if (time==temp2) list->addBus(new
 Bus("Bus1",111,18.0,0.0,90,0,0,0,0,9,8,8,8,10,2450.0));
     }
     fclose(stream111);

     stream30 = fopen("busGenTime3.txt", "r");
     int temp3;
     char msg3[20];
     fseek(stream30, 0, SEEK_SET);

     for (int i =0; i<31; i++)
     {
     fgets(msg3,10,stream30);
     temp3 = atoi(msg3);
     if (time==temp3) list->addBus(new
 Bus("Bus2",30,18.0,1800.0,90,7,7,7,7,15,30,11,11,22,4330.0));
     }
     fclose(stream30);

     stream31 = fopen("busGenTime4.txt", "r");
     int temp4;
     char msg4[20];
     fseek(stream31, 0, SEEK_SET);
     for (int i =0; i<30; i++)
     {
     fgets(msg4,10,stream31);
     temp4 = atoi(msg4);
     if (time==temp4) list->addBus(new
 Bus("Bus2",31,18.0,1800.0,90,7,7,7,7,9,8,8,8,22,2450.0));
     }
     fclose(stream31);

     stream1010 = fopen("busGenTime5.txt", "r");
     int temp5;
     char msg5[20];
     fseek(stream1010, 0, SEEK_SET);
     for (int i =0; i<20; i++)
     {
     fgets(msg5,10,stream1010);
     temp5 = atoi(msg5);
     if (time==temp5) list->addBus(new
 Bus("Bus2",1010,18.0,1800.0,90,7,7,7,7,9,8,8,8,15,2450.0));
     }
     fclose(stream1010);
    }
   }
//------------------------------------------------------------------------
   //Changing attributes of buses in a loop
//------------------------------------------------------------------------
   int tempNumber=StartBusNumber();
   for(int number=tempNumber; number <=list->getLength(); number++)
   {
    int busNumber = number;
    currentBus = list->getBus(number-1);
    int numberOfBuses = list->getLength();
    float busPosi1 = currentBus->oldPosition();
    float busPosi2 = currentBus->newPosition(timeFactor);
    MoveBus (busPosi1,busPosi2,numberOfBuses);

    fprintf(stream, "%6.2f %c", busPosi2, ' ');

    Simulate(time,busNumber,busPosi2,holdBusStop);

    if(time==simulationPeriod-1)  //this is for making sure that all buses in
 route reach destination
    {
     if(busPosi2<4326) simulationPeriod++;
     else simulationPeriod=simulationPeriod;
    }
    if(busPosi2==9999) ChangeBusNumber(busNumber);
   }
  DrawDetectorBeacon();     //Blinking of Beacons and repainting of Detectors
 }
 OutputFile(list->getLength(),passengerRateFactor);    //Formatted output
}
//---------------------------------------------------------------------------
void TMainSim::ChangeBusNumber(int busnos)
{
 if (tempBusNumber==busnos)
 {
  tempBusNumber=busnos+1;
 }
}
int TMainSim::StartBusNumber()
{
 return tempBusNumber;
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void TMainSim::Simulate(int time,int busNumber,float busPosi2,int
holdBusStop)
{
 float avlBusPosi2=AvlSystem(time,busNumber,busPosi2);
 ChangeLinkPara(busPosi2); //Changing link parameters
 CheckBusstops(time/timeFactor,busNumber,busPosi2,holdBusStop);   //Checking
Buses at Bus Stops
 CheckSignals(time/timeFactor,busNumber,busPosi2);    //Checking Buses at
signals
 if(priorityOption==1) CheckSVDPriority(time/timeFactor,busNumber,busPosi2);
//Giving priority using Detectors
 if(priorityOption==2)
CheckAVLPriority(time/timeFactor,busNumber,avlBusPosi2);  //Giving priority
using AVL data
 CheckBeacons(time/timeFactor,busNumber,busPosi2);   //Detecting  Buses at
Beacons
}
//---------------------------------------------------------------------------
void TMainSim::ChangeLinkPara(float busPosi2)
{
 int nextLinkNos=currentBus->nextLink();
 if (nextLinkNos<totalLinkNo)
 {
  currentLink = link[nextLinkNos];
  if (busPosi2>=(currentLink->linkPosition()))
  {
   float presentSpeed = currentLink->thisLinkSpeed();
   currentBus->getSpeed(presentSpeed);
  }
 }
}
//---------------------------------------------------------------------------
void TMainSim::CheckBusstops(int times,int busNumber,float busPosi2,int
holdBusStop)
{
 //int passGenOption=0;
 if(passGenOption==1)
 {
  int nextBusstopNos=currentBus->nextBusstop();
  if (nextBusstopNos<=currentBus->LastBusstop())//totalBusstopNo)
  {
     int reserver;
     int headway;
     int waitTime;
     int schTime1;
     int busstopNo=nextBusstopNos;
     int arrivalTime=currentBus->arriveBusstopTime(times,busstopNo);
     int passJourney=currentBus->PassJourneyTime ();
```

174

```
     stopOut[busstopNo][busNumber-1]=  arrivalTime;  //stored for formatted
 output generated at the last
     int passInside=currentBus->totalPassInside(0,0);
     int service=currentBus->ServiceNumber();
     if (service>100&&service<1000) schTime1=currentBusstop-
>ScheduleStartTime11();
     if (service<100) schTime1=currentBusstop->ScheduleStartTime3();
     if (service>1000) schTime1=currentBusstop->ScheduleStartTime101();
     if (busstopNo!=99) schTime1=0;        //not checking schedule time
     if((currentBusstop->ReservedNum1()==0))//&&(currentBusstop-
>ReservedNum2()==0))
     {
     headway = currentBusstop-
>headwayCalculate(times,busNumber,passInside,schTime1,0,service);  //it does
not make any difference in passBoard but shows only different in headway
      currentBusstop->GetReserveBus1(busNumber);
//this difference in headway in compesated in Busstop ::headwayCalculate-
schTime check
     }
//But makes difference in case of bus staying for alighting passenger after
departure of first bus
     else
//It may have some passenger during that period -calculated differently
     {
      //headway = currentBusstop-
>headwayCalculate(arrivalTime,busNumber,passInside,schTime1,1);
      headway = currentBusstop-
>headwayCalculate(times,busNumber,passInside,schTime1,1,service);
      currentBusstop->GetReserveBus2(busNumber);
     }
    int passAlight = currentBusstop->alightPass(passInside,busNumber);
    int boardPassenger = currentBusstop-
>boardPassDischarge(times,busNumber,passAlight); //have to use this for
dwellTime

    if((busNumber==currentBusstop-
>ReservedNum1())&&(busNumber==currentBusstop->ReservedNum2()))currentBusstop-
>GetReserveBus2(0);
    if(busNumber==currentBusstop->ReservedNum1())reserver=1;    //these are
to include boarded pass after departure of first bus
    if(busNumber==currentBusstop->ReservedNum2())reserver=2;
    currentBus-
>StopBusstopTime(0,busstopNo,headway,passAlight,0,waitTime,reserver);

     if (boardPassenger==0)
     {
      currentBus->changeSpeed(0);
      currentBus->getNextBusstop(busstopNo+1);
      int passBoard=currentBusstop->boardPassNos();
      int dwellT=currentBusstop->DwellTime();
      waitTime=currentBusstop->waitingTime(timeFactor);
      int totalPassInside=currentBus->totalPassInside(passAlight,passBoard);
      float occupy=currentBus->occupancy(totalPassInside);
      int schTime=currentBusstop->ScheduleStartTime11();
      //int deviatedTime=currentBusstop-
>DeviationInSchedule(arrivalTime,busGenerateTime);
      int deviatedTime=currentBusstop-
>DeviationInSchedule11(times,busGenerateTime);
      currentBus->GetBusstopDeviation(deviatedTime);
      int busJourney=currentBus->BusJourneyTime ();
      currentBusstop-
>GenerateOutput(occupy,passInside,deviatedTime,busJourney,waitTime,passAlight
,passBoard,dwellT);
      currentBusstop->GetReserveBus1(0);
      fprintf(stream1, "\n");  //busstop output
      fprintf(stream1, "%3d %3d %5d %c %5d %5d %5d %5d %5d %c %5.1f %c %3d
%5d %5d %5d %5d",
       busstopNo,busNumber,arrivalTime,'
',times,headway,passAlight,passBoard,
       totalPassInside,' ',occupy,'
',dwellT,waitTime,passJourney,schTime,deviatedTime);
     }
```

```
       else
       {
        currentBus->changeSpeed(1);
        currentBusstop->GetReserveBus1(busNumber);
       }
      }
     else currentBus->getNextBusstop(nextBusstopNos+1);
    }
   }
  }

  else if(passGenOption==0)    //generating passenger using uniform rate
  {
   int nextBusstopNos=currentBus->nextBusstop();
   if (nextBusstopNos<=currentBus->LastBusstop())//totalBusstopNo)
   {
    currentBusstop = busstop[nextBusstopNos];
    if (busPosi2>=(currentBusstop->busstopPosition()))
    {
     int reserver;
     int schTime1;
     int headway;
     int busstopNo=nextBusstopNos;
     int arrivalTime=currentBus->arriveBusstopTime(times,busstopNo);
     stopOut[busstopNo][busNumber-1]= arrivalTime; //stored for formatted
output generated at the last
     int passInside=currentBus->totalPassInside(0,0);
     int service=currentBus->ServiceNumber();
     if (service>100&&service<1000) schTime1=currentBusstop-
>ScheduleStartTime11();
     if (service<100) schTime1=currentBusstop->ScheduleStartTime3();
     if (service>1000) schTime1=currentBusstop->ScheduleStartTime101();
     if (busstopNo!=holdBusStop) schTime1=0;    //not checking schedule time
     if((currentBusstop->ReservedNum1()==0))//&&(currentBusstop-
>ReservedNum2()==0))
     {
      headway = currentBusstop-
>headwayCalculate(times,busNumber,passInside,schTime1,0,service); //it does
not make any difference in passBoard but shows only different in headway
      currentBusstop->GetReserveBus1(busNumber);
//this difference in headway in compesated in Busstop ::headwayCalculate-
schTime check
     }
//But makes difference in case of bus staying for alighting passenger after
departure of first bus
    else
//It may have some passenger during that period -calculated differently
     {
      headway = currentBusstop-
>headwayCalculate(times,busNumber,passInside,schTime1,1,service);
      currentBusstop->GetReserveBus2(busNumber);
     }
    float passAlight = currentBusstop->AlightPass2();
    float passBoard = currentBusstop->BoardPass2();
    int totDwellTime=currentBusstop->DwellTime2();
    float waitTime=currentBusstop->WaitTime2();
    if((busNumber==currentBusstop-
>ReservedNum1())&&(busNumber==currentBusstop->ReservedNum2()))currentBusstop-
>GetReserveBus2(0);
    if(busNumber==currentBusstop->ReservedNum1())reserver=1;    //these are
to include boarded pass after departure of first bus
    if(busNumber==currentBusstop->ReservedNum2())reserver=2;
    currentBus-
>StopBusstopTime(totDwellTime,busstopNo,headway,passAlight,passBoard,waitTime
,reserver);
    int dwellTime=currentBus->BusstopDwellTime();

    if (dwellTime==0)
    {
     int schTime;
     int deviatedTime;
     currentBus->changeSpeed(0);
```

```
      currentBus->getNextBusstop(busstopNo+1);
      float totalPassInside=currentBus->totalPassInside(passAlight,passBoard);
      float occupy=currentBus->occupancy(totalPassInside);
      if (service>100&&service<1000) schTime=currentBusstop-
>ScheduleStartTime11();
      if (service<100) schTime=currentBusstop->ScheduleStartTime3();
      if (service>1000) schTime=currentBusstop->ScheduleStartTime101();
      if (service>100&&service<1000) deviatedTime=currentBusstop-
>DeviationInSchedule11(times,busGenerateTime);
      if (service<100) deviatedTime=currentBusstop-
>DeviationInSchedule3(times,busGenerateTime);
      if (service>1000) deviatedTime=currentBusstop-
>DeviationInSchedule101(times,busGenerateTime);
      currentBus->GetBusstopDeviation(deviatedTime);
      currentBus->GetLateColor(deviatedTime);

      int schHeadway=currentBusstop->ScheduleHeadway(service);
      currentBus->GetBusstopSchedule(schHeadway);

      int busJourney=currentBus->BusJourneyTime();
      headway=currentBus->BusHeadway();
      float passAlight=currentBus->PassAlighted();
      float passBoard=currentBus->PassBoarded();
      float waitTime=currentBus->PassWaited();
      int totDwellTime=currentBus->TotalBusstopDwellTime();
      float passJourney=currentBus->PassJourneyTime ();

      currentBusstop-
>GenerateOutput(occupy,passInside,deviatedTime,busJourney,waitTime,passAlight
,passBoard,totDwellTime);
      if(currentBusstop->ReservedNum1()==busNumber)currentBusstop-
>GetReserveBus1(0);
      if(currentBusstop->ReservedNum2()==busNumber)currentBusstop-
>GetReserveBus2(0);
      {
      fprintf(stream1, "\n");  //busstop output
      fprintf(stream1, "%3d %3d %5d %c %5d %5d %5.2f %5.2f %5.1f %c %3.1f %c
%5d %5.1f %5.1f %5d %5d",
       busstopNo,busNumber,arrivalTime,' ',times,headway,passAlight,passBoard,
       totalPassInside,' ',occupy,'
',totDwellTime,waitTime,passJourney,schTime,deviatedTime);
      }
     }
     else
     {
      currentBus->changeSpeed(1);
     }
    }
   }
  }
}
//----------------------------------------------------------------------------
void TMainSim::CheckSignals(int times,int busNumber,float busPosi2)
{
 int nextSignalNos=currentBus->nextSignal();
 if (nextSignalNos<=currentBus->LastSignal())//totalSignalNo)
 {
  currentSignal = signal[nextSignalNos];
  if (busPosi2>=(currentSignal->signalPosition()))
  {
   int totalDelay;
   int greenTime;
   int signalNo=nextSignalNos;
   int arrivalTime=currentBus->arriveSignalTime(times,signalNo);
   int carInfront = currentSignal->frontCarNumber(busNumber);
   int sigStage = currentSignal->signalStage(times);
   if (sigStage==1)
   {
    greenTime=currentSignal->GreenStartTime(times,busNumber);
   }

   if (sigStage==1&&carInfront==0) //Stopping with taking care of Cars
```

```
    {
     currentBus->changeSpeed(0);
     currentBus->getNextSignal(signalNo+1);
     int carInfront1 = currentSignal->frontCarFirst();
     int carDelay1 = times-greenTime;
     int signalDelay1=greenTime-arrivalTime;
     totalDelay=times-arrivalTime;
     currentSignal->GetBusDelay(totalDelay);
     fprintf(stream2, "\n");
     fprintf(stream2, "%c %2d %c %2d %c %5d %c %5d %c %2d %c %3d %c %3d %c
%3d",
      ' ',signalNo,' ',busNumber,' ',arrivalTime,' ',times,' ',carInfront1,'
',carDelay1,' ',signalDelay1,' ',totalDelay);
    }
    else currentBus->changeSpeed(1);
   }
  }
 }
//---------------------------------------------------------------------------
void TMainSim::CheckSVDPriority(int times,int busNumber,float busPosi2)
{
 int nextDetectorNos=currentBus->nextDetector();
 if (nextDetectorNos<=currentBus->LastDetector())//totalDetectorNo)
 {
  currentDetector = detector[nextDetectorNos];
  if (busPosi2>=(currentDetector->DetectorPosition()))
  {
   int extension=0;
   int recall=0;
   int detectorNo=nextDetectorNos;
   int distance=currentDetector->DetectorDistance();
   int sigNum=currentDetector->SignalIdentity();
   int journeyTime=currentBus->JourneyTime(distance, timeFactor);
   int arrivalTime= times+journeyTime;

   currentSignal=  signal[sigNum] ;
   int sigStage = currentSignal->signalStage(arrivalTime);
   int cycleNumber=currentSignal->CycleNumber(arrivalTime);
   int endGreenTime=currentSignal->EndGreenState(arrivalTime);
   int startGreenTime=currentSignal->StartGreenState(arrivalTime);

   currentSignal->PriorityOption(2,9999,9999);
   int maxExtensionAllow=currentSignal->MaxExtensionTime();
   int maxRecallAllow=currentSignal->MaxRecallTime();

   int presentSigPeriod=currentSignal->signalStage(times);
   if((presentSigPeriod==1)&&(sigStage!=1 &&
endGreenTime<=maxExtensionAllow))
   {
    extension=endGreenTime;
    recall=0;
    currentSignal->GetExtension(extension,cycleNumber);
    int sideSig=currentSignal->SideSignalNumber();
    currentSignal=signal[sideSig];
    currentSignal->GetRecall(-extension,cycleNumber+1);
   }
   else if (sigStage!=1)
   {
    extension=0;
    recall=startGreenTime;
    if(recall>maxRecallAllow) recall=maxRecallAllow;
    currentSignal->GetRecall(recall,cycleNumber);
    int sideSig=currentSignal->SideSignalNumber();
    currentSignal=signal[sideSig];
    currentSignal->GetExtension(-recall,cycleNumber+1);
   }
   currentBus->getNextDetector(detectorNo+1);
   fprintf(stream5, "\n");
   fprintf(stream5, "%c %3d %c %3d %c %5d %c %3d %c %5d %c %3d %c %3d %c %3d
%c %3d",
   ' ',detectorNo,' ',busNumber,' ',times,' ',journeyTime,' ',arrivalTime,'
',sigNum,' ',sigStage,' ',extension,' ',recall);
```

```
    }
   }
  }
  //-----------------------------------------------------------------------
  void TMainSim::CheckAVLPriority(int times,int busNumber,float avlBusPosi2)
  {
   int nextVDetectorNos=currentBus->NextVDetector();
   if (nextVDetectorNos<=currentBus->LastVDetector())//<totalVirtualDetectorNo)
   {
    currentVDetector = vDetector[nextVDetectorNos];
    if (avlBusPosi2>=(currentVDetector->VirtualDetectorPosition()))
    {
     int extension=0;
     int recall=0;
     int vDetectorNo=nextVDetectorNos;
     int distance=currentVDetector->VirtualDetectorDistance();
     int sigNum=currentVDetector->SignalIdentity();
     int journeyTime=currentBus->JourneyTime(distance, timeFactor);
     int arrivalTime= times+journeyTime;

     currentSignal=  signal[sigNum] ;       //needs checking
     int sigStage = currentSignal->signalStage(arrivalTime);
     int cycleNumber=currentSignal->CycleNumber(arrivalTime);
     int endGreenTime=currentSignal->EndGreenState(arrivalTime);
     int startGreenTime=currentSignal->StartGreenState(arrivalTime);
     int restriction=currentSignal->RecallRestriction();
     int lateness=currentBus->LastBusstopDeviation();
     int scheduleHeadway=currentBus->LastBusstopSchedule();
     if(scheduleHeadway<=0)scheduleHeadway=1;
     currentSignal->PriorityOption(logicNumber,lateness,scheduleHeadway);
     int maxExtensionAllow=currentSignal->MaxExtensionTime();
     int maxRecallAllow=currentSignal->MaxRecallTime();

     int presentSigPeriod=currentSignal->signalStage(times);
     if((presentSigPeriod==1)&&(sigStage!=1 &&
 endGreenTime<=maxExtensionAllow))
     {
      extension=endGreenTime;
      recall=0;
      currentSignal->GetExtension(extension,cycleNumber);
      int sideSig=currentSignal->SideSignalNumber();
      currentSignal=signal[sideSig];
      currentSignal->GetRecall(-extension,cycleNumber+1);
     }
     else if (sigStage!=1)
     {
      extension=0;
      recall=startGreenTime-restriction;
      if(recall<0) recall =0;
      if(recall>maxRecallAllow) recall=maxRecallAllow;
      currentSignal->GetRecall(recall,cycleNumber);
      int sideSig=currentSignal->SideSignalNumber();
      currentSignal=signal[sideSig];
      currentSignal->GetExtension(-recall,cycleNumber+1);
     }
     currentBus->GetNextVDetector(vDetectorNo+1);
     fprintf(stream10, "\n");
     fprintf(stream10, "%c %3d %c %3d %c %5d %c %3d %c %3d %c %5d %c %3d %c %3d
 %c %3d %c %3d %5d %c %5d",
      ' ',vDetectorNo,' ',busNumber,' ',times,' ',presentSigPeriod,'
 ',journeyTime,' ',arrivalTime,' ',sigNum,' ',sigStage,' ',extension,'
 ',recall,lateness,' ',scheduleHeadway);
    }
   }
  }
 //-----------------------------------------------------------------------
 void TMainSim::CheckBeacons(int times,int busNumber,float busPosi2)
 {
  int beaconNo=currentBus->nextBeacon();
  if (beaconNo<totalBeaconNo)
  {
   currentBeacon = beacon[beaconNo];
   if (busPosi2>=(currentBeacon->BeaconPosition()))
   {
    currentBeacon->GetBeaconColor();
    int beaconPosi = currentBeacon->BeaconPosition();
    currentBus->GetBeaconDistance(beaconPosi);
    currentBus->getNextBeacon(beaconNo+1);
    fprintf(stream7, "\n");
    fprintf(stream7, " %5d %5d %5d %5d",beaconNo,busNumber,times,beaconPosi);
   }
  }
 }
 //----------------------------------------------------------------------
 //----------------------------------------------------------------------
 //This section is for drawing route, signals, busstops, buses and cars
 //----------------------------------------------------------------------
 //----------------------------------------------------------------------
 void TMainSim::MoveBus(int oldpos, int newpos, int nosBuses)
 {
  int oldPosition = oldpos%windowWidth;      //horizontal shift to other lines
  int y1=102+(int (oldpos/windowWidth))*80;  //vertical shift to other lines
  Canvas->Pen->Color = clSilver;
  Canvas->Brush->Color = clSilver;
  Canvas->Rectangle ((oldPosition),y1,((oldPosition)-10),y1+5);

  if(newpos<routeLength)
  {
   int newPosition = newpos%windowWidth;      //horizontal shift in other lines
   int y2=102+(int ((newpos-2)/windowWidth))*80;//vertical shift in other line
   Canvas->Pen->Color = clMaroon;
   int busColor=currentBus->BusColor();
   if (busColor==2)  Canvas->Brush->Color = clPurple;
   else if (busColor==3)  Canvas->Brush->Color = clFuchsia;
   else Canvas->Brush->Color = clRed;
   Canvas->Rectangle (newPosition,y2,((newPosition)-10),y2+5);
   if(priorityOption==2)//blinking at bus if AVL is acivated
   {
    int changeColor=currentBus->PollingColor();
    if (changeColor>0)                        //changing colour while polling
    {
     Canvas->Brush->Color = clAqua;
     Canvas->Rectangle (newPosition,y2,((newPosition)-5),y2+5);
    }
   }
   int lateColor=currentBus->LatenessColor();
   if (lateColor>0)                           //changing colour while polling
   {
    Canvas->Brush->Color = clBlack;
    Canvas->Rectangle (newPosition-6,y2,(newPosition-9),y2+5);
   }
  }
  for (int j=1; j<500000000/(simulationSpeed*nosBuses); j++)
  {
  }
 }
 //----------------------------------------------------------------------
 void TMainSim::ChangeSignalPhase(int times)
 {
  for (int j=0; j<totalSignalNo; j++)
  {
   currentSignal = signal[j];
   currentSignal->signalStage(times);
  }
 }
 //----------------------------------------------------------------------
 void TMainSim::GenerateCar(int times)
 {
  for (int j=0; j<totalSignalNo; j++)
  {
   currentSignal = signal[j];
   currentSignal->CalculateCarNumber(times);
   if (j<21&&times<999)
   {
    int stage = currentSignal->SignalStageState();
    int cars = currentSignal->newCarNumber();
    if (j>0)fprintf(stream21, "%3d %2d", stage,cars);
   }
  }
 }
 //---------------------------------------------------------------------------
 float TMainSim::AvlSystem(int time,int busNumber,float busPosi2)
 {
  float virtualBusPosi2;
  if(avlSystem==1)          //GPS based system
  {
   float gpsError=currentBus->GpsErrorGenerateNorm(gpsAccuracy);
   virtualBusPosi2=busPosi2+gpsError;
  }
  else if(avlSystem==2)     //beacon-based system
  {
   virtualBusPosi2=currentBus->VirtualPosition(busPosi2,odometerAccuracy);
  }
  fprintf(stream8, "%6.2f %c", virtualBusPosi2, ' ');

  //AVL polling
  if ((time%pollingFrequency)==((busNumber-1)%pollingFrequency))
  {
   currentBus->GetPollingColor();
   currentBus->GetAvlPosition(virtualBusPosi2);
  }

  //AVL Centre bus position
  float avlBusPosi2=currentBus->AvlPosition(); //it gives the position at AVL
 centre
  fprintf(stream9, "%6.2f %c", avlBusPosi2, ' ');//but there is a problem at
 First stop

  //return avlBusPosi2;
  return virtualBusPosi2;     //The GPS position of bus used fro priority
 request
 }
 //---------------------------------------------------------------------------
 int TMainSim::RandomBusGenerate()
 {
  //randomize();
  //float valuePx1 = rand()%400;
  //int valueX1 = (((log((valuePx1/100000)*60))*-60)+240); //for mean =300,
 st. dev = 60

  int valueX1=0;
  while(valueX1==0)
  //do
  {
   float valuePx;
   int valueX = rand()%1500;
   if (valueX>240)
   {
    valuePx = ((exp(-1*((valueX-240)/60)))/60);  //for mean =300, st. dev = 60
   }
   else valuePx=-1.0;

   float valueY = (rand()%1000)/250000.0;
   if(valueY<=valuePx)
   {
    valueX1= valueX;
   }
   else valueX1=0;
  }
  //while  (valueX1==0);
  return valueX1;
 }
 //---------------------------------------------------------------------------
 void __fastcall TMainSim::CloseBtnClick(TObject *Sender)
 {
  Close();
```

```
}
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
void TMainSim::RouteParameters(int startLag,float passRateFactor)
{
  //Timetable based on Shifted Portswood (Field) and Journey time
  //BusStop::BusStop(int busstopId,int linkNo,int busstopP,int
busstopStartTime,
  //float passRa,float passRb,int scheduleStartT11,int scheduleStartT3,int
scheduleStartT101,BusList* blist)
  busstop[0] = new BusStop(0,1,0,-
543,0.035,180.0/passRateFactor,60,60,60,list);
  busstop[1] = new BusStop(1,4,150,-
437,0.013,144.0/passRateFactor,180,60,60,list);
  busstop[2] = new BusStop(2,6,440,-
357,0.005,514.3/passRateFactor,240,60,60,list);
  busstop[3] = new BusStop(3,8,720,-
298,0.005,257.1/passRateFactor,300,60,60,list);
  busstop[4] = new BusStop(4,9,900,-
259,0.004,480.0/passRateFactor,360,60,60,list);
  busstop[5] = new BusStop(5,10,1250,-
161,0.00,553.8/passRateFactor,420,60,60,list);
  busstop[6] = new BusStop(6,12,1610,-
116,0.004,1440.0/passRateFactor,480,60,60,list);
  busstop[7] = new
BusStop(7,14,1800,23,0.305,69.9/passRateFactor,600,300,1140,list);
//218->32
  busstop[8] = new
BusStop(8,16,2060,136,0.048,141.2/passRateFactor,720,420,1260,list);
  busstop[9] = new
BusStop(9,17,2410,191,0.007,342.9/passRateFactor,780,480,1320,list);
  busstop[10] = new
BusStop(10,19,2620,286,0.015,900.0/passRateFactor,1440,540,1920,list);
  busstop[11] = new
BusStop(11,21,2990,379,0.019,360.0/passRateFactor,1500,600,1920,list);
  busstop[12] = new
BusStop(12,22,3210,412,0.028,3600.0/passRateFactor,1500,600,1920,list);
  busstop[13] = new
BusStop(13,23,3490,472,0.113,2400.0/passRateFactor,1560,660,1920,list);
  busstop[14] = new
BusStop(14,25,4010,627,0.235,7200.0/passRateFactor,1680,780,1920,list);
  busstop[15] = new BusStop(15,27,4320,757,0.811,99999.0,1800,900,1920,list);
  totalBusstopNo =16;

  //Defining Traffic Signals for South-bound Service
  //Signal(int junctionId, float unctionPosi, 1int signalId,2int sideSig,3int
//linkNo,4float signalP,5int cycleT, 6int greenT, 7int amberT,8int
//offset,9int flowArm,10int satFlow,int maxExtend, int maxRecal)
  signal[0] = new Signal (0,0,1,0,0,-100,10,11,0,0,0.0001,9999,0,0,0,0,0);
//non-existing signal for better output
  signal[1] = new Signal
(1,50,1,12,2,50,108,12,7,78+startLag,21.88,4800,10,48,10,51,8);
  signal[2] = new Signal
(2,100,1,0,3,100,104,76,21,76+startLag,24.41,2000,15,0,15,0,13);
  signal[3] = new Signal
(3,400,1,13,5,400,108,30,7,177+startLag,25.88,2000,9,22,9,23,28);
  signal[4] = new Signal
(4,670,1,14,7,670,43,21,7,64+startLag,28.02,1700,7,0,7,0,0);
  signal[5] = new Signal
(5,1290,1,15,11,1290,67,45,6,103+startLag,28.23,2400,5,0,5,0,2);
  signal[6] = new Signal
(6,1710,1,16,13,1710,75,18,7,95+startLag,24.05,2400,12,0,12,3,17);
  signal[7] = new Signal
(7,1960,1,17,15,1960,82,31,15,127+startLag,41.76,2200,19,10,19,11,0);
  signal[8] = new Signal
(8,2440,1,18,18,2440,67,15,12,58+startLag,41.79,3600,11,16,11,18,0);
  signal[9] = new Signal
(9,2950,1,0,20,2950,84,22,6,91+startLag,36.57,1800,9,0,9,0,17);
  signal[10] = new Signal
(10,3870,1,19,24,3870,103,16,9,105+startLag,35.74,3200,16,23,16,26,0);
  signal[11] = new Signal
(11,4230,1,20,26,4230,100,16,15,141+startLag,10.17,2000,15,0,15,3,18);
```

```
  //Side signals Southbound (towards City centre)---------------------------
  //signal[12] = new Signal
(12,0,26,20050,108,71,9,167+startLag,38.83,4000,21,21,21,21,0);
  signal[12] = new Signal
(1,50,2,0,26,20050,108,71,9,167+startLag,59.24,3600,21,21,21,21,0);
  signal[13] = new Signal
(3,400,2,0,26,20400,108,34,8,248+startLag,30.65,3600,21,21,21,21,0);
  signal[14] = new Signal
(4,670,2,0,26,20670,43,8,7,79+startLag,5.73,1800,21,21,21,21,0);
  signal[15] = new Signal
(5,1290,2,0,26,21290,67,9,7,119+startLag,7.38,1200,21,21,21,21,0);
  signal[16] = new Signal
(6,1710,2,0,26,21710,75,17,10,141+startLag,33.58,2000,21,21,21,21,0);
  signal[17] = new Signal
(7,1960,2,0,26,21960,82,17,6,163+startLag,10.0,1500,21,21,21,21,0);      //no
flow and delay data safeway junction car park
  signal[18] = new Signal
(8,2440,2,0,26,22440,67,31,8,98+startLag,24.38,1500,21,21,21,21,0);
  signal[19] = new Signal
(10,3870,2,0,26,23870,103,44,13,162+startLag,55.89,3600,21,21,21,21,0);
  signal[20] = new Signal
(11,4230,2,0,26,24230,100,19,15,193+startLag,29.08,2000,21,21,21,21,0);
  //Side signals Northbound (towards Swaythling)----------------------------
  signal[21] = new Signal
(11,4230,4,0,26,34835,100,19,15,210+startLag,27.56,2000,21,21,21,21,0);
  signal[22] = new Signal
(10,3870,4,0,26,35195,103,44,13,187+startLag,51.02,1800,21,21,21,21,0);
  signal[23] = new Signal
(9,2950,4,0,26,36115,84,30,9,158+startLag,27.50,1800,21,21,21,21,0);
  signal[24] = new Signal
(7,1960,4,0,26,37105,82,17,6,169+startLag,19.68,1800,21,21,21,21,0);
  signal[25] = new Signal
(6,1710,4,0,26,37355,75,11,8,141+startLag,28.22,3000,21,21,21,21,0);
  signal[26] = new Signal
(4,670,4,0,26,38395,43,8,7,79+startLag,7.66,1800,21,21,21,21,0);
  signal[27] = new Signal
(3,400,4,0,26,38665,108,20,7,248+startLag,10.0,1800,21,21,21,21,0);      // no
flow and delay data data
  signal[28] = new Signal
(2,100,4,0,26,38965,104,48,6,137+startLag,29.62,3600,21,21,21,21,0);
  //Main signals Northbound (towards Swythling) Reverse order start from City
  signal[29] = new Signal
(11,4230,3,0,26,14835,100,16,15,141+startLag,12.78,1800,21,21,21,21,0);
  signal[30] = new Signal
(10,3870,3,0,26,15195,103,11,10,105+startLag,11.19,1800,21,21,21,21,0);
  signal[31] = new Signal
(9,2950,3,0,26,16115,84,11,6,91+startLag,13.75,1800,21,21,21,21,0);
  signal[32] = new Signal
(8,2440,3,0,26,16625,67,33,12,58+startLag,55.81,1800,21,21,21,21,0);
  signal[33] = new Signal
(7,1960,3,0,26,17105,82,28,12,127+startLag,33.30,1800,21,21,21,21,0);
  signal[34] = new Signal
(6,1710,3,0,26,17355,75,20,9,95+startLag,40.50,1800,21,21,21,21,0);
  signal[35] = new Signal
(5,1290,3,0,26,17775,67,45,6,103+startLag,30.81,1800,21,21,21,21,0);
  signal[36] = new Signal
(4,670,3,0,26,18395,43,21,7,64+startLag,29.89,1800,21,21,21,21,0);
  signal[37] = new Signal
(3,400,3,0,26,18665,108,30,7,177+startLag,23.77,1800,21,21,21,21,0);
  signal[38] = new Signal
(2,100,3,0,26,18965,104,28,15,76+startLag,25.62,1800,21,21,21,21,0);
  signal[39] = new Signal
(1,50,3,0,26,19015,108,71,9,78+startLag,42.92,1800,21,21,21,21,0);
  totalSignalNo = 40;

//-------------------------------------------------------------------------
  //Defining Links for South-bound Service
  //Link(int lID, int lStart,int lDist,float lTime,float lSpeed)
  link[0] = new Link (0,-10,-10,0.0,3.6);  //non-existing link for better
output
  link[1] = new Link (1,0,50,0.0,16.74);
  link[2] = new Link (2,50,50,0.0,11.37);
```

```
   link[3] = new Link (3,100,50,0.0,10.80);
   link[4] = new Link (4,150,250,0.0,30.95);
   link[5] = new Link (5,400,40,0.0,7.89);
   link[6] = new Link (6,440,230,0.0,39.74);
   link[7] = new Link (7,670,50,0.0,18.78);
   link[8] = new Link (8,720,180,0.0,25.17);
   link[9] = new Link (9,900,350,0.0,18.28);
   link[10] = new Link (10,1250,40,0.0,26.58);
   link[11] = new Link (11,1290,320,0.0,32.30);
   link[12] = new Link (12,1610,100,0.0,21.71);
   link[13] = new Link (13,1710,90,0.0,9.53);
   link[14] = new Link (14,1800,160,0.0,13.96);
   link[15] = new Link (15,1960,100,0.0,15.85);
   link[16] = new Link (16,2060,350,0.0,31.25);
   link[17] = new Link (17,2410,30,0.0,7.32);
   link[18] = new Link (18,2440,180,0.0,14.27);
   link[19] = new Link (19,2620,330,0.0,28.12);
   link[20] = new Link (20,2950,40,0.0,9.93);
   link[21] = new Link (21,2990,220,0.0,34.43);
   link[22] = new Link (22,3210,280,0.0,21.64);
   link[23] = new Link (23,3490,380,0.0,16.70);
   link[24] = new Link (24,3870,140,0.0,21.60);
   link[25] = new Link (25,4010,220,0.0,17.90);
   link[26] = new Link (26,4230,90,0.0,10.68);
   link[27] = new Link (27,4320,0,0.0,36.0); //non-existing link to stop buses
at destination
   totalLinkNo = 28;                       //the speed 36 is to cross busstop

//---------------------------------------------------------------------------
   //Defining Detectors for South-bound Service
   //Detector(int detectId, int linkNO, int signalNO,float detectP, int
detectDist)
   detector[0] = new Detector (0,-10,0,-10,0);  //non-existing detector for
better output
   detector[1] = new Detector (1,1,1,5,45);
   detector[2] = new Detector (2,2,2,54,46);
   detector[3] = new Detector (3,4,3,330,70);
   detector[4] = new Detector (4,6,4,600,70);
   detector[5] = new Detector (5,10,5,1256,34);
   detector[6] = new Detector (6,12,6,1640,70);
   detector[7] = new Detector (7,14,7,1890,70);
   detector[8] = new Detector (8,18,8,2419,21);
   detector[9] = new Detector (9,20,9,2880,70);
   detector[10] = new Detector (10,24,10,3800,70);
   detector[11] = new Detector (11,26,11,4160,70);
   totalDetectorNo =12;

   //Defining Beaconss for South-bound Service
   //Avl(int detectId,int linkNo,int detectP);
   beacon[0] = new Beacon (0,0,-10);  //non-existing link for better output
   beacon[1] = new Beacon (1,1,500);
   beacon[2] = new Beacon (2,2,1500);
   beacon[3] = new Beacon (3,3,2500);
   beacon[4] = new Beacon (4,4,3500);
   totalBeaconNo=5;

   //Defining Detectors for South-bound Service
   //Detector(int detectId, int signalNO,float detectP, int detectDist)
   vDetector[0] = new VirtualDetector (0,0,-10,0);  //non-existing link for
better output
   vDetector[1] = new VirtualDetector (1,1,5,45);
   vDetector[2] = new VirtualDetector (2,2,54,46);
   vDetector[3] = new VirtualDetector (3,3,330,70);
   vDetector[4] = new VirtualDetector (4,4,600,70);
   vDetector[5] = new VirtualDetector (5,5,1256,34);
   vDetector[6] = new VirtualDetector (6,6,1640,70);
   vDetector[7] = new VirtualDetector (7,7,1890,70);
   vDetector[8] = new VirtualDetector (8,8,2419,21);
   vDetector[9] = new VirtualDetector (9,9,2880,70);
   vDetector[10] = new VirtualDetector (10,10,3800,70);
   vDetector[11] = new VirtualDetector (11,11,4160,70);
```

```
  //vDetector[0] = new VirtualDetector (0,0,-10,0);  //non-existing link for
 better output
  //vDetector[1] = new VirtualDetector (1,1,5,35);
  //vDetector[2] = new VirtualDetector (2,2,54,36);
  //vDetector[3] = new VirtualDetector (3,3,330,70);
  //vDetector[4] = new VirtualDetector (4,4,600,70);
  //vDetector[5] = new VirtualDetector (5,5,1256,24);
  //vDetector[6] = new VirtualDetector (6,6,1640,70);
  //vDetector[7] = new VirtualDetector (7,7,1890,70);
  //vDetector[8] = new VirtualDetector (8,8,2419,11);
  //vDetector[9] = new VirtualDetector (9,9,2880,70);
  //vDetector[10] = new VirtualDetector (10,10,3800,70);
  //vDetector[11] = new VirtualDetector (11,11,4160,70);
  totalVirtualDetectorNo =12;
}
//-----------------------------------------------------------------------------
//These are non changing functions - Drawing Routes, showing time and
changing signal
//-----------------------------------------------------------------------------
void TMainSim::DrawRoute()
{
 int routeSegmentNos=routeLength/windowWidth;      //for horizontal shift in
other lines
 Canvas->Pen->Color = clSilver;
 Canvas->Brush->Color = clSilver;
 Canvas->Rectangle (0,0,windowWidth,routeSegmentNos*120);   //cleanup for re-
start

 for (int i=0; i<=routeSegmentNos; i++)
 {
  int segmentLength=windowWidth;      //for horizontal shift in other lines
  if (i==routeSegmentNos) segmentLength=routeLength%windowWidth;
  int y=100+i*80;      //for vertical shift in other lines
  Canvas->Pen->Color = clBlack;
  Canvas->MoveTo (0,y);
  Canvas->LineTo (segmentLength,y);       //draw a line
  Canvas->MoveTo (0,y+15);
  Canvas->LineTo (segmentLength,y+15);
  Canvas->Pen->Color = clSilver;
  Canvas->Brush->Color = clSilver;
  Canvas->Rectangle (0,y+2,segmentLength,y+13);   //road area

  for (int j=0; j<totalSignalNo; j++)    //Drawing Side Roads in position
  {
   currentSignal = signal[j];
   int junctionPosi = currentSignal->junctionPosition();
   int signalLoc = currentSignal->signalIdentity();
   int signalPosition = junctionPosi%windowWidth;      //for horizontal shift
in other lines
   int y=100+(int(junctionPosi/windowWidth))*80;
   Canvas->Pen->Color = clBlack;
   if ((signalLoc==2)||(signalLoc==4))
   {
   Canvas->Pen->Color = clBlack;
   Canvas->MoveTo (signalPosition,y+(int(signalLoc/4)*45));
   Canvas->LineTo (signalPosition,y+(int(signalLoc/4)*45)-30);
   Canvas->MoveTo (signalPosition+15,y+(int(signalLoc/4)*45));
   Canvas->LineTo (signalPosition+15,y+(int(signalLoc/4)*45)-30);
   Canvas->Pen->Color = clSilver;
   Canvas->Brush->Color = clSilver;
   Canvas->Rectangle
(signalPosition+1,y+(int(signalLoc/4)*45)+1,signalPosition+15,y+(int(signalLo
c/4)*45)-31);   //road area
   }
  }
 }

 for (int j=0; j<totalSignalNo; j++)    //Drawing Traffic Signals in position
 {
  currentSignal = signal[j];
  int junctionPosi = currentSignal->junctionPosition();
  int signalLoc = currentSignal->signalIdentity();
  int signalPosition = junctionPosi%windowWidth;      //for horizontal shift
 in other lines
  int y1=85+(int (junctionPosi/windowWidth))*80;      //for vertical shift in
 other lines
  Canvas->Pen->Color=clBlack;         // should be off if all others are on
  Canvas->Brush->Color=clBlack;
  if (signalLoc==1) Canvas->Rectangle(signalPosition,y1,signalPosition-
 6,y1+16);
  else if (signalLoc==2) Canvas-
 >Rectangle(signalPosition+21,y1,signalPosition+21-6,y1+16);
  else if (signalLoc==3) Canvas-
 >Rectangle(signalPosition+21,y1+30,signalPosition+21-6,y1+46);
  else if (signalLoc==4) Canvas-
 >Rectangle(signalPosition,y1+30,signalPosition-6,y1+46);
 }

 for (int j=0; j<totalBusstopNo; j++)     //Drawing Bus Stops in position
 {
  currentBusstop = busstop[j];
  int busstopPosi = currentBusstop->busstopPosition();
  int busstopPosition=busstopPosi%windowWidth;      //for horizontal shift in
 other lines
  int y=96+(int (busstopPosi/windowWidth))*80;      //for vertical shift in
 other lines
  Canvas->Pen->Color = clBlue;
  Canvas->Brush->Color = clBlue;
  Canvas->Rectangle (busstopPosition,y,busstopPosition+5,y+4);       //bus
 stop1 (110)
 }

 //if(priorityOption==2)
 if(priorityOption==2&&avlSystem==2)
 {
  for (int j=0; j<totalBeaconNo; j++)    //Drawing Bus Stops in position
  {
   currentBeacon = beacon[j];
   int beaconPosi = currentBeacon->BeaconPosition();
   int beaconPosition=beaconPosi%windowWidth;      //for horizontal shift in
 other lines
   int y=100+(int (beaconPosi/windowWidth))*80;      //for vertical shift in
 other lines
   Canvas->Pen->Color = clBlack;
   Canvas->MoveTo (beaconPosition,y);
   Canvas->LineTo (beaconPosition,y-10);
   Canvas->Brush->Color = clGreen;
   Canvas->Pen->Color = clGreen;
   Canvas->Ellipse (beaconPosition-2,y-6,beaconPosition+2,y-10);       //bus
 stop1 (110)
  }
 }
//----------------------------------------------------------------------
//For displaying Place names
 Canvas->Font->Size=8;
 Canvas->Font->Color = clBlack;
 Canvas->Brush->Color = clSilver;
 Canvas->TextOut((155%windowWidth),((155/windowWidth)*80+87),"Swaythling");
 Canvas->TextOut((117%windowWidth),((117/windowWidth)*80+131),"Burgess
Road");
 Canvas->TextOut((417%windowWidth),((417/windowWidth)*80+131),"Langhorn
Road");
 Canvas->TextOut((600%windowWidth),((600/windowWidth)*80+131),"Mayfield
Road");
 Canvas->TextOut((1540%windowWidth),((1540/windowWidth)*80+87),"Bus Depot");
 Canvas->TextOut((1727%windowWidth),((1727/windowWidth)*80+131),"Highfield
Lane");
 Canvas->TextOut((1820%windowWidth),((1820/windowWidth)*80+87),"Portswood");
 Canvas->TextOut((2652%windowWidth),((2652/windowWidth)*80+87),"Lodge Road");
 Canvas->TextOut((2020%windowWidth),((2020/windowWidth)*80+117),"Safeway");
 Canvas->TextOut((2889%windowWidth),((2889/windowWidth)*80+131),"The
Avenue");
 Canvas->TextOut((3302%windowWidth),((3302/windowWidth)*80+87),"The Avenue");
 Canvas->TextOut((3887%windowWidth),((3887/windowWidth)*80+131),"Cumberland
Place");
 Canvas->TextOut((4022%windowWidth),((4022/windowWidth)*80+87),"Cenotaph");
 Canvas->TextOut((4247%windowWidth),((4247/windowWidth)*80+131),"New Road");
 Canvas->TextOut((4330%windowWidth),((4330/windowWidth)*80+87),"City
Centre");
}
//---------------------------------------------------------------------------
void TMainSim::DisplaySimTime(int SimTime)   //For displaying Simulation Time
{
  char *str;
  double num;
  int dec, sign, ndig = 0;//5;
  num = SimTime;///60.0;//2.0;
  str = fcvt(num, ndig, &dec, &sign);
  int totalTime = StrToInt(str);
  int sec = totalTime%60;
  int minute = (totalTime%3600)/60;
  int hour = totalTime/3600;

  Canvas->Pen->Color = clSilver;
  Canvas->Brush->Color = clSilver;
  Canvas->Rectangle (550,10,568,12);
  Canvas->Font->Color=clMaroon;
  Canvas->Font->Size=12;
  Canvas->Font->Style=TFontStyles()<<fsBold<<fsItalic<<fsUnderline;
  //Canvas->Font->Style=TFontStyles()<<fsUnderline;
  Canvas->TextOut(10,10,"Transportation Research Group");
  Canvas->Font->Color=clNavy;//Maroon;
  Canvas->Font->Size=12;
  Canvas->Font->Style=TFontStyles()<<fsBold;
  Canvas->TextOut((windowWidth-200),20,"Simulation Period");
  Canvas->TextOut((windowWidth-50),20,str);

  //Time display in hour:minute:second format
  int secDigit=CountDigit(sec);
  int minuteDigit=CountDigit(minute);
  int hourDigit=CountDigit(hour);
  Canvas->Brush->Color = clSilver;
  Canvas->Rectangle ((windowWidth-400),20,(windowWidth-500),40);
  Canvas->TextOut((windowWidth-480),20,"0");
  Canvas->TextOut((windowWidth-450),20,"0");
  Canvas->TextOut((windowWidth-420),20,"0");
  Canvas->TextOut((windowWidth-(400+secDigit*10)),20,sec);
  Canvas->TextOut((windowWidth-430),20,":");
  Canvas->TextOut((windowWidth-(430+minuteDigit*10)),20,minute);
  Canvas->TextOut((windowWidth-460),20,":");
  Canvas->TextOut((windowWidth-(460+hourDigit*10)),20,hour);

  Canvas->Font->Style = TFontStyles(); //clears
}
//---------------------------------------------------------------------------
int TMainSim::CountDigit(int number)   //For displaying Simulation Time
{
  int digit;
  if(number<=9) digit =1;
  else digit =2;
  return digit;
}
//---------------------------------------------------------------------------
void TMainSim::DrawSignalPhase()
{
 for (int j=0; j<totalSignalNo; j++)
 {
  int signalPosition;  int factor1;  int factor2;
  currentSignal = signal[j];
  int stage = currentSignal->SignalStageState();
  int junctionPosi = currentSignal->junctionPosition();
  int signalLoc = currentSignal->signalIdentity();
  signalPosition=junctionPosi%windowWidth;
  if (signalLoc==1)
  {
     factor1=0;    factor2=0;
    }
    else if (signalLoc==2)
    {
     factor1=1;    factor2=0;
    }
    else if (signalLoc==3)
    {
     factor1=1;    factor2=1;
    }
    else if (signalLoc==4)
    {
     factor1=0;    factor2=1;
    }
    int y=85+factor2*30+(int (junctionPosi/windowWidth))*80;      //for vertical
 shift
    Canvas->Brush->Color=clBlack;
    Canvas->Pen->Color=clLime;
    Canvas->Ellipse(signalPosition+21*factor1-1,y+11-
 10*factor1,signalPosition+21*factor1-5,y+15-10*factor1);
    Canvas->Pen->Color=clRed;
    Canvas->Ellipse(signalPosition+21*factor1-
 1,y+1+10*factor1,signalPosition+21*factor1-5,y+5+10*factor1);
    Canvas->Pen->Color=clYellow;
    Canvas->Ellipse(signalPosition+21*factor1-1,y+6,signalPosition+21*factor1-
 5,y+10);

    if (stage==1)
    {
     Canvas->Pen->Color=clLime;    Canvas->Brush->Color=clLime;
     Canvas->Ellipse(signalPosition+21*factor1-1,y+11-
 10*factor1,signalPosition+21*factor1-5,y+15-10*factor1);
    }
    else if (stage==2)
    {
     Canvas->Pen->Color=clYellow;    Canvas->Brush->Color=clYellow;
     Canvas->Ellipse(signalPosition+21*factor1-1,y+6,signalPosition+21*factor1-
 5,y+10);
    }
    else
    {
     Canvas->Pen->Color=clRed;    Canvas->Brush->Color=clRed;
     Canvas->Ellipse(signalPosition+21*factor1-
 1,y+1+10*factor1,signalPosition+21*factor1-5,y+5+10*factor1);
    }
  }
}
//---------------------------------------------------------------------------
void TMainSim::DrawCar()
{
 for (int j=0; j<totalSignalNo; j++)
 {
   currentSignal = signal[j];
   int junctionPosi = currentSignal->junctionPosition();
   int signalLoc = currentSignal->signalIdentity();
   int signalPosition = junctionPosi%windowWidth;       //for horizontal shift
in other lines
   int y=85+(int(junctionPosi/windowWidth))*80;//+22;  //22 is vertical
displacement of cars from bus
   int newCarNos = currentSignal->newCarNumber();
   int carPosi=signalPosition;

   Canvas->Pen->Color = clSilver;
   Canvas->Brush->Color = clSilver;
   if(signalLoc==1) Canvas->Rectangle (carPosi,y+22,(carPosi-
(newCarNos+3)*4),y+22+2);  //deleting cars
   if(signalLoc==2) Canvas->Rectangle (carPosi+14,y+15,carPosi+14-3,(y+15-
(newCarNos+2)*3));  //deleting cars
   if(signalLoc==3) Canvas->Rectangle
(carPosi+16,y+24,(carPosi+16+(newCarNos+3)*4),y+24+2);  //deleting cars
   if(signalLoc==4) Canvas->Rectangle
(carPosi+2,y+30,carPosi+2+3,(y+30+(newCarNos+2)*3));  //deleting cars
```

```
   if(newCarNos>0)
   {
    for(int i=0;i<newCarNos;i++)
    {
     Canvas->Pen->Color = clMaroon;
     Canvas->Brush->Color = clRed;
     if(signalLoc==1)Canvas->Rectangle (carPosi-(i*4),y+22,(carPosi-3-
 (i*4)),y+22+2);  //adding cars
     if(signalLoc==2) Canvas->Rectangle (carPosi+14,y+15-(i*3),(carPosi+14-
 3),y+15-2-(i*3));  //adding cars
     if(signalLoc==3)Canvas->Rectangle
 (carPosi+16+(i*4),y+24,(carPosi+16+3+(i*4)),y+24+2);  //adding cars
     if(signalLoc==4) Canvas->Rectangle
 (carPosi+2,y+30+(i*3),(carPosi+2+3),y+30+2+(i*3));  //adding cars
    }
   }
  }
 }
//---------------------------------------------------------------------------
void TMainSim::DrawDetectorBeacon()
 {
  if(priorityOption==1)
  {
   {
    for (int j=0; j<totalDetectorNo; j++)     //Redrawing Detectors in position
    {
     currentDetector = detector[j];
     int detectorPos = currentDetector->DetectorPosition();
     int detectorPosition=detectorPos%windowWidth;
     int y=101+(int (detectorPos/windowWidth))*80;
     Canvas->Pen->Color = clDkGray;
     Canvas->Brush->Color = clSilver;
     Canvas->Rectangle (detectorPosition+6,y+1,detectorPosition,y+4);
    }
   }
  }
  if(priorityOption==2&&avlSystem==2)
  {
   for(int j=0; j<totalBeaconNo; j++)     //Changing color of Beacons
   {
    currentBeacon = beacon[j];
    int beaconPosi = currentBeacon->BeaconPosition();
    int beaconPosition=beaconPosi%windowWidth;
    int y=100+(int (beaconPosi/windowWidth))*80;
    Canvas->Pen->Color = clGreen;
    Canvas->Brush->Color = clGreen;
    int changeColor=currentBeacon->BeaconColor();
    if (changeColor>0)
    {
     Canvas->Pen->Color = clYellow;
     Canvas->Brush->Color = clRed;
    }
    Canvas->Ellipse (beaconPosition-2,y-6,beaconPosition+2,y-10);
   }
  }
}
//---------------------------------------------------------------------------
void TMainSim::DrawPass(int times)   //only valid if there is no more than
one passenger difference in each time segment
{
 for (int j=0; j<totalBusstopNo; j++)
 {
  currentBusstop = busstop[j];
  int busstopPosi = currentBusstop->busstopPosition();
  int busstopPosition = busstopPosi%windowWidth;
  int y1=85+(int(busstopPosi/windowWidth))*80+5;  //22 is vertical
displacement of cars from bus
  int newPassNos = currentBusstop->boardPassGenerate(times);

  int passPosi=busstopPosition;
  Canvas->Pen->Color = clSilver;
```

```
   Canvas->Brush->Color = clSilver;
   Canvas->Rectangle (passPosi,y1,(passPosi-(newPassNos+1)*3),y1+4);
 //deleting passengers

   if(newPassNos>0)
   {
    for(int i=0;i<newPassNos;i++)
    {
     int passPosi=busstopPosition-(i*3);
     Canvas->Pen->Color = clBlack;
     Canvas->Brush->Color = clBlack;
     Canvas->Rectangle (passPosi,y1,(passPosi-2),y1+4);  //adding passengers
    }
   }
  }
}
//---------------------------------------------------------------------------
// The section generates formatted outputs only
//---------------------------------------------------------------------------
void TMainSim::OutputHeaders()     //Defining output files
{
 stream = fopen("SimOut.txt", "w+");
 fprintf(stream, "\n %c %c %s %c %s %c", ' ',' ',"Time", ' ', "Distance", '
');

 stream1 = fopen("BusstopOut.txt", "w+");
 fprintf(stream1, "\n %s %s %s %s %s %s %s %s %s %s %s %s %s %s",
"Stop","Bus","Arrive","Depart","Headway","PassA","PassB","PassIn","Occupy","D
well","Wait","JourT","SchT","Late");

 stream2 = fopen("SignalOut.txt", "w+");
 fprintf(stream2, "\n %s %s %s %s %s %s %s %s %s",
"Signal","BusNo","Arrive","Depart","QCar","CrLag","SgLag","TotLag","DelayToCa
rs");

 stream21 = fopen("SignalStages.txt", "w+");
 fprintf(stream21, "\n %s %s %s %s %s %s %s %s %s",
"1","2","3","4","5","6","7","8","9");
 fprintf(stream21, "\n %s %s %s %s %s %s %s %s %s %s %s %s",
"Stage","Car","Stage","Car","Stage","Car","Stage","Car","Stage","Car","Stage"
,"Car");

 stream3 = fopen("SimbolOutput.txt", "w+");  //for formatted output of
Busstop time
 //stream3 = fopen("SimbolOutput.txt", "a+");  //a+ adds the output at the
end rather than overwrites

 stream4 = fopen("LinkOut.txt", "w+");
 fprintf(stream4, "\n %s %s %s", "LinkNo","BusNo","Time");

 stream5 = fopen("DetectOut.txt", "w+");
 fprintf(stream5, "\n %s %s %s %s %s %s %s %s %s",
"Detect","BusNo","Time","JrTime","Arrive","Signal","Period","Extend","Recall"
);

 stream6 = fopen("CarDelaysOut.txt", "w+");
 fprintf(stream6, "\n %c %s %s %c %s %c %s %c %s %c %s %c %s %c %s %c %s %c
%s %c %s %c %s %c %s",
  ' ',"Time","0",' ',"1",' ',"2",' ',"3",' ',"4",' ',"5",' ',"6",' ',"7",'
',"8",' ',"9",' ',"10",' ',"11");

 stream7 = fopen("BeaconOut.txt", "w+");
 fprintf(stream7, "\n %s %s %s %s", "Beacon","BusNo","Time","BeaconPosi");

 stream8 = fopen("BusAVLPosition.txt", "w+");
 fprintf(stream8, "\n %c %c %s %c %s %c", ' ',' ',"Time", ' ', "Distance", '
');

 stream9 = fopen("AVLCentrePosition.txt", "w+");
 fprintf(stream9, "\n %c %c %s %c %s %c", ' ',' ',"Time", ' ', "Distance", '
');
```

```
 stream10 = fopen("VirtualDetectOut.txt", "w+");
 fprintf(stream10, "\n %s %s %s %s %s %s %s %s %s %s %s",
"VDetect","BusNo","Time","period","JrTime","Arrive","Signal","Period","Extend
","Recall","Lateness");

 //stream11 = fopen("ComparativeChart.txt", "w+");
 stream11 = fopen("ComparativeChart.txt", "a+");
 //fprintf(stream11, "\n %s %s %s %s %s %s %s %s",
"PAlight","PBoard","DwellT","PWaitT","BJourneyT","PJourneyT","CDelayT","BDela
yT");
 //fprintf(stream11, "\n %s %s %s %s %s %s %s %s", "(Nos)","  (Nos)","
(sec)"," (sec)","  (sec)","   (sec)","   (sec)","   (sec)");
}
//---------------------------------------------------------------------------
void TMainSim::OutputFile(int busNumber,float passRateFactor)
{
 float averageJourneyTimeOfBuses=0.0;
 float totalJourneyTimeOfBuses=0.0;
 float averagePassJourneyTimePerBus=0.0;
 float totalPassJourneyTime=0.0;
 float averagePassWaitingTimePerBus=0.0;
 float totalPassWaitingTime=0.0;
 float totalBusDelays=0.0;
 float carDelayPerJunction=0.0;
 float totalCarDelays=0.0;
 float grandTotalCarDelays=0.0;
 float busDelayPerJunction=0.0;
 float totalNumberOfAlightingPass=0.0;
 float averageAlightingPass=0.0;
 float totalNumberOfBoardingPass=0.0;
 float averageBoardingPass=0.0;
 float totalDwellTimeAtStop=0.0;
 float averageDwellTime=0.0;
 float totMainRoadGreen=0.0;
 float totSideRoadGreen=0.0;

 fprintf(stream3, "\n %s","SIMBOL Output");
 fprintf(stream3, "\n %s","---------------------------------------------");
 fprintf(stream3, "\n %s","Simulation of Portswood Corridor, Southampton");
 fprintf(stream3, "\n %s","Priority Type =");
 if(priorityOption==0) fprintf(stream3, "%s","No Priority");
 else if(priorityOption==1) fprintf(stream3, "%s","SVD Priority");
 else//(priorityOption==2)
 {
  fprintf(stream3, "%s","Differential Priority");
  fprintf(stream3, "\n %s","AVL System =");
  {
   if(avlSystem==1) fprintf(stream3, "%s","GPS System");
   else if(avlSystem==2) fprintf(stream3, "%s","Beacon System");
  }
  fprintf(stream3, "\n %s %2d","Logic Number =",logicNumber);
 }
 if(holdBusStop==7) fprintf(stream3, "%s",", Holding option - Yes");
 else if(holdBusStop==99) fprintf(stream3, "%s",", Holding option - No");
 fprintf(stream3, " %s %2.1f",", Passenger generation factor
=",passengerRateFactor);
 fprintf(stream3, "\n %s %5d %s","Simulation Period =",enterSimTime,"secs");
 fprintf(stream3, " %s %5d %s","Actual Simulation Period
=",simulationPeriod,"secs");

 //fprintf(stream3, "\n %s %s %c","
","BusStops",' ');
 fprintf(stream3, "\n \n %s","BusStp");
 //fprintf(stream3, "\n %s","BusNum");
 for (int i = 0; i<totalBusstopNo; i++) fprintf(stream3, "%6d",i);

 fprintf(stream3, "\n %s","Distce");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  int dist=currentBusstop->busstopPosition();
  fprintf(stream3, "%6d",dist);
 }

 fprintf(stream3, "\n %s","BusNum");

 for (int j=0; j<busNumber; j++)
 {
  fprintf(stream3, "\n %4d %c",j+1, ' ');
  for (int i = 0; i<totalBusstopNo; i++)
  {
   fprintf(stream3, "%6d",stopOut[i][j]);    //Bus arrival at different stops
  }
 }

 fprintf(stream3, "\n \n %s","BusStp");
 for (int i = 0; i<totalBusstopNo; i++) fprintf(stream3, "%6d",i);

 fprintf(stream3, "\n %s","noBsBs");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  int numBus=currentBusstop->NumberOfBuses();
  fprintf(stream3, "%6d",numBus);
 }

 fprintf(stream3, "\n %s","relib1");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  int relBus=currentBusstop->ReliableBuses();
  fprintf(stream3, "%6d",relBus);
 }

 fprintf(stream3, "\n %s","relib2");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  int relBus2=currentBusstop->ReliableBuses2();
  fprintf(stream3, "%6d",relBus2);
 }

 fprintf(stream3, "\n %s","avDvBs");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float deviate=currentBusstop->AverageScheduleDeviation();
  fprintf(stream3, "%6.1f",deviate);
 }

 fprintf(stream3, "\n %s","avMdDv");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float modDeviate=currentBusstop->AverageModulusDeviation();
  fprintf(stream3, "%6.1f",modDeviate);
 }

 fprintf(stream3, "\n %s","sdDvBs");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float sdDeviate=currentBusstop->SdOfDeviation();
  fprintf(stream3, "%6.1f",sdDeviate);
 }

 fprintf(stream3, "\n %s","sdMdDv");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float sdDeviate=currentBusstop->SdOfModulusDeviation();
  fprintf(stream3, "%6.1f",sdDeviate);
 }
 fprintf(stream3, "\n %s","avWtPs");
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float waitBus=currentBusstop->WaitingTimePerBus();
  float pBoard1=currentBusstop->AveragePassBoarded();
  float avPassWait=waitBus/(pBoard1);
  fprintf(stream3, "%6.1f",avPassWait);
 }

 fprintf(stream3, "\n %s","avWtBs");          //average waiting time per bus
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float waitBus=currentBusstop->WaitingTimePerBus();
  float totalWaitBus=currentBusstop->TotalWaitingTime();
  fprintf(stream3, "%6.1f",waitBus);
  //fprintf(stream3, "%6.1f",totalWaitBus/3600.0);
  averagePassWaitingTimePerBus+=waitBus;
  totalPassWaitingTime+=totalWaitBus;
 }

 fprintf(stream3, "\n %s","avOpBs");          //average occupancy per bus
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float occupy=currentBusstop->AverageOccupancy();
  fprintf(stream3, "%6.1f",occupy);
 }

 fprintf(stream3, "\n %s","avJrBs");         //average bus journey time
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float bJourney=currentBusstop->AverageBusJourney();
  float totalBJourney=currentBusstop->TotalBusJourney();
  if(i>0)
  {
   fprintf(stream3, "%6.1f",bJourney);
   if(i>0)averageJourneyTimeOfBuses+=bJourney;
   if(i>0)totalJourneyTimeOfBuses+=totalBJourney;
  }
  if(i==0)fprintf(stream3, "%s","   0.0");
 }

 fprintf(stream3, "\n %s","psJrBs");          //passenger journey time per
bus
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float pJourney=currentBusstop->AveragePassJourney();
  float totalPJourney=currentBusstop->TotalPassJourney();
  fprintf(stream3, "%6.1f",pJourney);
  averagePassJourneyTimePerBus+=pJourney;
  totalPassJourneyTime+=totalPJourney;
 }

 fprintf(stream3, "\n %s","psAlBs");          //passenger journey time per
bus
 for (int i = 0; i<totalBusstopNo; i++)
 {
  currentBusstop=busstop[i];
  float pAlight=currentBusstop->AveragePassAlighted();
  fprintf(stream3, "%6.1f",pAlight);
  totalNumberOfAlightingPass+=pAlight;
  if(i>0)averageAlightingPass=totalNumberOfAlightingPass/i;
 }

 fprintf(stream3, "\n %s","psBdBs");          //passenger journey time per
bus
 for (int i = 0; i<totalBusstopNo; i++)
 {
   currentBusstop=busstop[i];
   float pBoard=currentBusstop->AveragePassBoarded();
   fprintf(stream3, "%6.1f",pBoard);
   totalNumberOfBoardingPass+=pBoard;
   if(i>0)averageBoardingPass=totalNumberOfBoardingPass/i;
  }

  fprintf(stream3, "\n %s","psDwBs");          //passenger dwell time per bus
  for (int i = 0; i<totalBusstopNo; i++)
  {
   currentBusstop=busstop[i];
   float pDwellT=currentBusstop->AverageDwellTime();
   fprintf(stream3, "%6.1f",pDwellT);
   totalDwellTimeAtStop+=pDwellT;
   if(i>0)averageDwellTime=totalDwellTimeAtStop/i;
  }

   float totPass1 =0.0;
  fprintf(stream3, "\n %s","psBdNo");          //total passenger number per
 bus stop
  for (int i = 0; i<totalBusstopNo; i++)
  {
   currentBusstop=busstop[i];
   float pBoard1=currentBusstop->AveragePassBoarded();
   int numBus1=currentBusstop->NumberOfBuses();
   float totPass=pBoard1*numBus1;
   totPass1+=totPass;
   fprintf(stream3, "%6.1f",totPass);
  }
  fprintf(stream3, "\n %s","psBdTt");          //total passenger number in
 system
  fprintf(stream3, "%6.1f",totPass1);

  fprintf(stream3, "\n \n %s \n %s %s %s %s","signal
 performance","signalNo","carDely","avCarDely","avBusDelay");
  for (int j=0; j<21; j++)    //only one side of junction taken into account
  {
   currentSignal = signal[j];
   float carQueues=currentSignal->TotalDelayCarNumber();
   float averageCarDelays=currentSignal->AverageCarDelay(simulationPeriod);
   float busDelay=currentSignal->AverageBusDelay();
   float carDelayHr=carQueues/simulationPeriod;//8000.0;
   float totGreenTime=currentSignal->TotalGreenTime()/3600.0;
   fprintf(stream3, "\n %5d %8.1f %8.1f %6.1f %6.3f %5.2f",
 j,carQueues,averageCarDelays,busDelay,carDelayHr,totGreenTime);  //Delay at
 different signals
   totalCarDelays+=averageCarDelays;//carQueues;
   if(j>0)carDelayPerJunction=totalCarDelays/j;//(j-1);
   if(j>0 && j<12)
   {
    totalBusDelays+=busDelay;
    if(j>0)busDelayPerJunction=totalBusDelays/j;//(j-1);
   }
   grandTotalCarDelays+=carQueues;
   if(j>0 && j<12) totMainRoadGreen+=totGreenTime;
   if(j>11) totSideRoadGreen+=totGreenTime;
  }

 fprintf(stream3, "\n\n %s %10.1f %s","Av nos of pass alight per bus per
busstop =",averageAlightingPass,"nos");
 fprintf(stream3, "\n %s %10.1f %s","Av nos of pass board per bus per busstop
=",averageBoardingPass,"nos");
 fprintf(stream3, "\n %s %10.1f %s","Average dwell time per bus per busstop
=",averageDwellTime,"secs");
 fprintf(stream3, "\n\n %s %10.1f %s","Pass Waiting Per Bus
=",averagePassWaitingTimePerBus,"secs");
 fprintf(stream3, "\n %s %10.1f %s","Journey Time Per Bus
=",averageJourneyTimeOfBuses,"secs");
 fprintf(stream3, "\n %s %10.1f %s","Pass Journey Per Bus
=",averagePassJourneyTimePerBus,"secs");
 fprintf(stream3, "\n %s %10.1f %s","Total car delays
=",grandTotalCarDelays,"secs");
```

```
 fprintf(stream3, "\n %s %10.1f %s","Average car delay per junction
=",carDelayPerJunction,"secs");
 fprintf(stream3, "\n %s %10.1f %s","Average bus delay per junction
=",busDelayPerJunction,"secs");
 fprintf(stream3, "\n %s %10.2f %s","Total bus route side green time
=",totMainRoadGreen,"hours");
 fprintf(stream3, "\n %s %10.2f %s","Total side road side green time
=",totSideRoadGreen,"hours");

 float
totalCost=totalPassWaitingTime*((374*passRateFactor)/2)/(3600*totPass1)*6.30+
                totalJourneyTimeOfBuses/(3600*10)*9.83+
totalPassJourneyTime*((374*passRateFactor)/2)/(3600*totPass1)*3.15+
                grandTotalCarDelays/simulationPeriod*6.74;

 fprintf(stream3, "\n\n %s %10.2f %s","Total Pass Waiting
=",totalPassWaitingTime*((374*passRateFactor)/2)/(3600*totPass1),"hours");
 fprintf(stream3, "\n %s %10.2f %s","Total Bus Journey
=",totalJourneyTimeOfBuses/(3600*10),"hours");
 fprintf(stream3, "\n %s %10.2f %s","Total Pass Journey
=",totalPassJourneyTime*((374*passRateFactor)/2)/(3600*totPass1),"hours");
 fprintf(stream3, "\n %s %10.2f %s","Total car delays
=",grandTotalCarDelays/simulationPeriod,"hours");
 fprintf(stream3, "\n %s %10.2f %s","Total cost per hour
=",totalCost,"pounds");

 fprintf(stream11, "\n %6d %3d%1d %10.1f %10.2f %10.2f %10.2f %10.2f %10.2f",
//Gap gives gap in the output

simulationPeriod,priorityOption,logicNumber,totPass1,totalPassWaitingTime*((3
74*passRateFactor)/2)/(3600*totPass1),

totalJourneyTimeOfBuses/(3600*2),totalPassJourneyTime*((374*passRateFactor)/2
)/(3600*totPass1),grandTotalCarDelays/simulationPeriod,totalCost);

 fprintf(stream6, "\n");
 for (int j=0; j<totalSignalNo; j++)
 {
  currentSignal = signal[j];
  int carQueues=currentSignal->TotalDelayCarNumber();
  fprintf(stream6, "%6d", carQueues);
 }
}
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_LINK)
#define EXAMPLE_BUS_LINK

#include <stdlib.h>
#include <iostream.h>

class Link
{
  public:
    Link(int lID, int lStart,int lDist,float lTime,float lSpeed);
    ~Link();
    int linkIdentity();
    int linkPosition();
    int thisLinkTime(int times);
    float thisLinkSpeed();

  private:
    int linkID;
    int linkStart;
    int linkDist;
    float linkTime;
    float linkSpeed;

};

#endif // end of !defined(EXAMPLE_RACE_TRACK) check
```

```
#include "link.h"


Link::Link(int lID,int lStart,int lDist,float lTime,float lSpeed)
{
  linkID = lID;
  linkStart = lStart;
  linkDist = lDist;
  linkTime= lTime;
  linkSpeed = lSpeed;
}
//--------------------------------------------------------------------------
Link::~Link()                {}
int Link::linkIdentity()     {return linkID;}
int Link::linkPosition()     {return linkStart;}
int Link::thisLinkTime(int times)  {return linkTime;}
float Link::thisLinkSpeed()        {return linkSpeed;}
//--------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_BUS)
#define EXAMPLE_BUS_BUS

#include <iostream.h>
#include <string.h>

class Bus
{
  public:
    Bus(char* nm,int serveNo,float sp,float oldp,int capa,int firstStop,int
firstSig,int firstDetect,int firstVDetect,int lastStop, int lastSig,int
lastDetect,int lastVDetect,int passStart,float finalp);
    ~Bus();
    int ServiceNumber();
    int nextLink();
    int FirstBusstop();
    int LastBusstop();
    int LastSignal();
    int LastDetector();
    int LastVDetector();
    void getSpeed(float lSpeed);
    int arriveBusstopTime(int times,int busstopNumber2);
    void StopBusstopTime(int times,int busstopNumber2,int bHeadway,float
pAlight,float pBoard,float pWait,int reserveNos);
    int BusstopDwellTime();
    int TotalBusstopDwellTime();
    int BusHeadway();
    float PassAlighted();
    float PassBoarded();
    float PassWaited();

    void getNextBusstop(int busstopNums);
    int nextBusstop();
    int arriveSignalTime(int times,int busstopNumber2);
    void getNextSignal(int signalNums);
    int nextSignal();
    int totalSignalTime();
    void getNextDetector(int detectorNums);
    int nextDetector();
    void getNextBeacon(int beaconNums);
    int nextBeacon();
    void GetNextVDetector(int vDetectorNums);
    int NextVDetector();

    void changeSpeed(int times);
    float oldPosition ();
    float newPosition(float timeFactor);
    float totalPassInside(float passAlight,float passBoard);
    float occupancy(float totalPassg);

    int JourneyTime (int distance,float timeFactors);
    int PassJourneyTime ();
    int BusJourneyTime();

    void GetBeaconDistance(float distance);
    float VirtualPosition(float realBusPosi, int odoAccuracy);
    void GetAvlPosition(float distance);
    float AvlPosition();

    void GetBusstopDeviation(int deviTimes);
    int LastBusstopDeviation();
    void GetBusstopSchedule(int schTimes);
    int LastBusstopSchedule();

    void GetPollingColor();
    int PollingColor();
    void GetLateColor(int timings);
    int LatenessColor();
    int BusColor();

    float GpsErrorGenerateNorm(int maxError);
    private:
      int serviceNos;
      float speed;
      float normalSpeed;
      char* name;
      float oldBusPosition;
      int nextBusstopNos;
      int nextSignalNos;
      int nextDetectorNos;
      int nextVDetectorNos;
      int firstBusstopNos;
      int lastBusstopNos;
      int lastSignalNos;
      int lastDetectorNos;
      int lastVDetectorNos;
      float passAlready;
      float finalDist;

      int busstopDelayTime;
      int busstopNumber1;
      int signalDelayTime;
      int signalNumber1;
      int totalSignalDelayTime;
      int passCapacity;
      float linkSpeed;
      float pSpeed;
      int nextBeaconNos;

      int linkNumber1;
      int nextLinkNos;
      int busstopArriveTime;
      int signalArriveTime;
      int busstopNumber11;
      int busstopStopTime;
      int totalBusstopStopTime;
      int busHeadway;
      float passAlight;
      float passBoard;
      float passWait;
      int alreadyReserved;

      int previousTimes;
      int passengerJourneyTime;
      int busJourneyTime;

      float lastBeaconPosi;

      float lastPollingPosi;
      int lastPollingTime;
      int nextPollingTime;

      int lastBusstopDeviation;
      int lastBusstopSchedule;
      int timeDuration;
      int timeDuration1;
};

#endif
```

```
#include "bus.h"


Bus::Bus(char* nm,int serveNo,float sp,float oldp,int capa,int firstStop,int
firstSig,int firstDetect,int firstVDetect,int lastStop,int lastSig,int
lastDetect,int lastVDetect,int passStart,float finalp)
{
  serviceNos=serveNo;
  speed = sp;
  normalSpeed = sp;
  name = new char[100];
  strncpy(name, nm, 100);
  oldBusPosition = (oldp - (speed/3.6)); //for making the position zero
  passCapacity = capa;
  nextBusstopNos =firstStop;//0;
  nextSignalNos =firstSig;//0;
  nextDetectorNos =firstDetect;//0;
  nextVDetectorNos =firstVDetect;//0;
  firstBusstopNos=firstStop;
  lastBusstopNos =lastStop;//0;
  lastSignalNos =lastSig;//0;
  lastDetectorNos =lastDetect;//0;
  lastVDetectorNos =lastVDetect;//0;
  passAlready = passStart;//0;
  finalDist=finalp;
  linkNumber1 = 0;
  busstopNumber1 = 100;
  signalNumber1 = 100;
  nextLinkNos =0;
  nextBeaconNos=0;       //need to change for 3 service
  busstopArriveTime =0;
  signalArriveTime=0;
  busstopNumber11=100;
  busstopStopTime=0;
  totalBusstopStopTime=0;
  busHeadway=0;
  passAlight=0.0;
  passBoard=0.0;
  passWait=0.0;
  alreadyReserved=0;
  previousTimes=0;
  passengerJourneyTime=0;
  busJourneyTime=0;
  lastBeaconPosi=0.0;
  lastPollingPosi=0.0;
  lastBusstopDeviation=0;
  lastBusstopSchedule=0;
  timeDuration1=0;
}
//-----------------------------------------------------------------------------
Bus::~Bus()     {delete[] name;}
int Bus::ServiceNumber()  {return serviceNos;}
int Bus::nextLink()  {return nextLinkNos;}
int Bus::FirstBusstop() {return firstBusstopNos;}
int Bus::LastBusstop() {return lastBusstopNos;}
int Bus::LastSignal() {return lastSignalNos;}
int Bus::LastDetector() {return lastDetectorNos;}
int Bus::LastVDetector() {return lastDetectorNos;}

//-----------------------------------------------------------------------------
void Bus::getSpeed(float lSpeed)
{
  normalSpeed=lSpeed;
  nextLinkNos++;
}
//-----------------------------------------------------------------------------
void Bus::changeSpeed(int times)
{
  int delayTimes= times;
  if (delayTimes!=0) {speed = 0.0;}
  else {speed = normalSpeed;}
}
```

```
//-----------------------------------------------------------------------
float Bus::oldPosition ()      {return oldBusPosition;}
//-----------------------------------------------------------------------
float Bus::newPosition (float timeFactor)
{
 if(oldBusPosition<-0.1)     //  if(oldBusPosition<=0.0)
 {
  oldBusPosition=oldBusPosition+(speed/(3.6))-(speed/(3.6*timeFactor));
 }
 float newBusPosition = oldBusPosition + (speed/(3.6*timeFactor));
 if(newBusPosition>finalDist) {newBusPosition=9999.0;}      //terminate
 oldBusPosition = newBusPosition;
 return newBusPosition;
}
//-----------------------------------------------------------------------
int Bus::arriveSignalTime(int times,int signalNumber2)
{
 if (signalNumber2!=signalNumber1)
 {
  signalNumber1=signalNumber2;
  signalArriveTime= times;
 }
 return signalArriveTime;
}
//-----------------------------------------------------------------------
void Bus::getNextSignal(int signalNums)     {nextSignalNos=signalNums;}
int Bus::nextSignal()                       {return nextSignalNos;}
int Bus::totalSignalTime()                  {return totalSignalDelayTime;}
//-----------------------------------------------------------------------
int Bus::arriveBusstopTime(int times,int busstopNumber2)
{
 if (busstopNumber2!=busstopNumber1)
 {
  busstopNumber1=busstopNumber2;
  busstopArriveTime= times;
  if (busstopNumber2!=firstBusstopNos)
  {
  passengerJourneyTime= passAlready*(times-previousTimes);
  busJourneyTime= times-previousTimes;
  }
  previousTimes=times;
 }
 return busstopArriveTime;
}
//-----------------------------------------------------------------------
int Bus::PassJourneyTime()      {return passengerJourneyTime;}
int Bus::BusJourneyTime()       {return busJourneyTime;}
//-----------------------------------------------------------------------
void Bus::StopBusstopTime(int times,int busstopNumber2,int bHeadway,float
pAlight,float pBoard,float pWait,int reserveNos)
{
 if (busstopNumber2!=busstopNumber11)
 {
  busstopNumber11=busstopNumber2;
  busstopStopTime= times;
  totalBusstopStopTime= times;
  busHeadway=bHeadway;
  passAlight=pAlight;
  passBoard=pBoard;
  passWait=pWait;
  alreadyReserved=reserveNos;
 }
 else { busstopStopTime--;}
 if(reserveNos!=alreadyReserved)     //here if first bus moves, second stays
 {                                   //second have some boarding passengers
  alreadyReserved=reserveNos;        //departure of the first bus
  busHeadway=bHeadway;
  passBoard=pBoard;
 }
}
//-----------------------------------------------------------------------
int Bus::BusstopDwellTime()                  {return busstopStopTime;}
int Bus::TotalBusstopDwellTime()                {return totalBusstopStopTime;}
int Bus::BusHeadway()                           {return busHeadway;}
float Bus::PassAlighted()                       {return passAlight;}
float Bus::PassBoarded()                        {return passBoard;}
float Bus::PassWaited()                         {return passWait;}

void Bus::getNextBusstop(int busstopNums)        {nextBusstopNos=busstopNums;}
int Bus::nextBusstop()                           {return nextBusstopNos;}
//-------------------------------------------------------------------------
float Bus::totalPassInside (float passAlight,float passBoard)
{
  passAlready = passAlready-passAlight+passBoard;
  return passAlready;
}
//-------------------------------------------------------------------------
float Bus::occupancy (float totalPassg)
{
 float busOccupy = 1.0*totalPassg; //float busOccupy =
(100.0*totalPassg/passCapacity);
 return busOccupy;
}
//-------------------------------------------------------------------------
int Bus::JourneyTime (int distance,float timeFactors)
{
  //int journeyTime= ((distance/normalSpeed)*3.6*timeFactors);
  double journeyTime= ((distance/normalSpeed)*3.6*timeFactors);
  double journeyTime1=ceil(journeyTime);      //For rounding up
  return journeyTime1;
}
//-------------------------------------------------------------------------
void Bus::GetBusstopDeviation(int deviTimes)
{lastBusstopDeviation=deviTimes;}
int Bus::LastBusstopDeviation()               {return lastBusstopDeviation;}
void Bus::GetBusstopSchedule(int schTimes)  {lastBusstopSchedule=schTimes;}
int Bus::LastBusstopSchedule()               {return lastBusstopSchedule;}
//-------------------------------------------------------------------------
//This part is for AVL assistance
//-------------------------------------------------------------------------
void Bus::GetBeaconDistance(float distance)
{
 lastBeaconPosi=distance;
 if(lastBeaconPosi<0)lastBeaconPosi=0.0;
}
//-------------------------------------------------------------------------
float Bus::VirtualPosition(float realBusPosi, int odoAccuracy)  //For Beacon
{
 float errorDistance=realBusPosi-lastBeaconPosi;
 float virtualAddDist=errorDistance*odoAccuracy/100.0;
 float virtualBusPosition=lastBeaconPosi+virtualAddDist;
 return virtualBusPosition;
}
//-------------------------------------------------------------------------
void Bus::GetAvlPosition(float distance)       //transfer from polling
{
 lastPollingPosi=distance-speed/3.6;
 if(lastPollingPosi<0)lastPollingPosi=0.0;
}
//-------------------------------------------------------------------------
float Bus::AvlPosition()
{
 if(lastPollingPosi>0.0)
 {
  lastPollingPosi=lastPollingPosi+speed/3.6;
 }
 return lastPollingPosi;
}
//-------------------------------------------------------------------------
void Bus::getNextDetector(int detectorNums)    {nextDetectorNos=detectorNums;}
int Bus::nextDetector()                        {return nextDetectorNos;}
void Bus::getNextBeacon(int beaconNums)        {nextBeaconNos=beaconNums;}
int Bus::nextBeacon()                          {return nextBeaconNos;}
void Bus::GetNextVDetector(int vDetectorNums)
```

```
{nextVDetectorNos=vDetectorNums;}
int Bus::NextVDetector()                    {return nextVDetectorNos;}
//-----------------------------------------------------------------------------
void Bus::GetPollingColor()    {timeDuration=2;}

int Bus::PollingColor()
{
 int pollingColor=timeDuration;
 if (timeDuration>0)timeDuration--;
 return pollingColor;
}
//-----------------------------------------------------------------------------
void Bus::GetLateColor(int timings)    {timeDuration1=timings;}
//-----------------------------------------------------------------------------
int Bus::LatenessColor()
{
 int lateColor=timeDuration1;
 return lateColor;
}
//-----------------------------------------------------------------------------
int Bus::BusColor()
{
 int busColor;
 switch (serviceNos)
 {
  case 110:
  busColor=1;
  break;
  case 111:
  busColor=1;
  break;
  case 30:
  busColor=2;
  break;
  case 31:
  busColor=2;
  break;
  case 1010:
  busColor=3;
 }
 return busColor;
}
//-------------------------------------------------------------------------
float Bus::GpsErrorGenerateNorm(int maxError)
{
  float mean=0.0;
  float stDeviation=3.3;
  int multiplier=2*(3*stDeviation)*1000;     //bigger range of random nos.
  float divider=multiplier/(2.0*(3*stDeviation));  //for bringing back to
normal number

  float valueX1=99999;
  while(valueX1==99999)
  {
    float valueX = (rand()%multiplier)/divider-10.0;
    float valueZx = (valueX-mean)/stDeviation;
    float valuePx = ((exp(-
0.5*(valueZx*valueZx)))/((sqrt(2*3.14159))*stDeviation));

    float valueY = (rand()%1000)*0.001/((sqrt(2*3.14159))*stDeviation);
    if(valueY<=valuePx)
    {
      valueX1= valueX;
    }
    else valueX1=99999;
  }
  if (maxError==0) valueX1=0.0;
  return valueX1;
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_BUSSTOP)
#define EXAMPLE_BUS_BUSSTOP

#include <stdlib.h>
#include <iostream.h>
#include "buslist.h"

class BusStop
{
  public:
    BusStop(int busstopId,int linkNo,int busstopP,int busstopStartTime,float
passRa,float passRb,int scheduleStartT11,int scheduleStartT3,int
scheduleStartT101,BusList* blist);
    ~BusStop();
    int busstopLinkNumber();
    int busstopIdentity();
    int busstopPosition();
    int alightPass(int passInside,int busArrive1);
    int boardPassGenerate(int times);
    int boardPassDischarge(int times, int busArrive, int alightPass);
    int DwellTime();
    int boardPassNos();
    int waitingTime(int timeFactor);
    int ScheduleStartTime11();
    int ScheduleStartTime3();
    int ScheduleStartTime101();
    int DeviationInSchedule11(int times,int busFrequency);
    int DeviationInSchedule3(int times,int busFrequency);
    int DeviationInSchedule101(int times,int busFrequency);
    int ScheduleHeadway(int serviceNos);
    int BusstopHeadway();
    void GetReserveBus1(int resNum);
    void GetReserveBus2(int resNum);
    int ReservedNum1();
    int ReservedNum2();
    int PassGenerateExpo(int meanPassHeadway);

    int headwayCalculate(int times,int busArrive2,int passInside,int
schTime,int reserved,int serviceNo);
    float AlightPass2();
    float BoardPass2();
    int DwellTime2();
    float WaitTime2();
    float WaitTime3();
    float WaitTime4();
    float WaitTime5();

    void GenerateOutput(float occupancy,float passIn,int deviation,int
busJourney,float avWaitTime,float passAlighted,float passBoarded,float
dwellTimeAtStop);
    int NumberOfBuses();
    int ReliableBuses();
    int ReliableBuses2();
    float AverageOccupancy();
    float AverageScheduleDeviation();
    float SdOfDeviation();
    float AverageModulusDeviation();
    float SdOfModulusDeviation();
    float AverageBusJourney();
    float TotalBusJourney();
    float AveragePassJourney();
    float TotalPassJourney();
    float WaitingTimePerBus();
    float TotalWaitingTime();
    float AveragePassAlighted();
    float AveragePassBoarded();
    float AverageDwellTime();

  private:
    int busstopID;
    int linkNos;
    int busstopPosi;
```

```
     float alightPassRate;
     float boardPassRate;
     int timeBus1;
     int startTime;
     int scheduleStartTime11;
     int scheduleStartTime3;
     int scheduleStartTime101;
     BusList* buslist;
     int numberPass1;
     int passGenerateTime;
     int passGenerateFactor;
     int passDischargeFactor;
     int busAlready;
     int checkinTime;
     int alightPassTime;
     int numberBoardPass;
     int totalDwellTime;
     int passengerA;
     int busAlready1;
     int passengerB;
     int waitTime;
     float deadTime;
     float passAlightingTime;
     float passBoardingTime;
     int finalWaitTime;
     int busstopReserved1;
     int busstopReserved2;
     int headway;
     int busAlready2;
     float passengerA2;
     float passengerB2;
     int dwellTime2;
     int cumuHeadwaySquare;
     int cumuHeadway;
     int cumuHeadwaySquare5;
     int cumuHeadway5;
     float averageWaitTime;
     float cumuPassWait;
     float cumuPassenger;
     float averageWaitTime3;
     float averageWaitTime4;
     float averageWaitTime5;

     int numberOfBuses;
     int reliableBuses;
     int reliableBuses2;
     float cumuOccupancy;
     float averageOccupancy;
     float cumuDeviation;
     float averageDeviation;
     double cumuModulusDev;
     float sumOfDeviationDiffer;
     float sdOfDeviation;
     float sumOfModulusDeviationDiffer;
     float sdOfModulusDeviation;
     float avModulusDeviation;
     float cumuBusJourney;
     float averageBusJourney;
     float cumuPassIn;
     float averagePassIn;
     float averagePassJourney;
     float cumuPassJourney;
     float cumuWaitTime;
     float averageWaitTimePerBus;
     float cumuPassAlighted;
     float averagePassAlighted;
     float cumuPassBoarded;
     float averagePassBoarded;
     float cumuDwellTime;
     float averageDwellTime;
};
#endif // end of !defined(EXAMPLE_RACE_TRACK) check
```

```
#include "busstop.h"


BusStop::BusStop(int busstopId,int linkNo,int busstopP,int busstopStartTime,
        float passRa,float passRb,int scheduleStartT11,int
scheduleStartT3,int scheduleStartT101,BusList* blist)
{
  busstopID = busstopId;
  linkNos = linkNo;
  busstopPosi = busstopP;
  alightPassRate = passRa;
  boardPassRate = passRb;
  timeBus1= busstopStartTime;
  startTime=busstopStartTime;
  scheduleStartTime11=scheduleStartT11;
  scheduleStartTime3=scheduleStartT3;
  scheduleStartTime101=scheduleStartT101;
  buslist = blist;
  numberPass1 =0;
  passGenerateTime=99999; // to avoid pass generate until initialisation
  passGenerateFactor=1;
  passDischargeFactor=1;
  busAlready =0;
  checkinTime=0;
  alightPassTime=0;
  numberBoardPass=0;
  totalDwellTime=0;
  passengerA=0;
  busAlready1=0;
  passengerB=0;
  waitTime=0;
  finalWaitTime=0;
  busstopReserved1=0;
  busstopReserved2=0;

  headway=0;
  busAlready2=0;
  passengerA2=0.0;
  passengerB2=0.0;
  dwellTime2=0;
  cumuHeadwaySquare=0;
  cumuHeadway=0;
  cumuHeadwaySquare5=0;
  cumuHeadway5=0;
  averageWaitTime=0;
  cumuPassWait=0.0;
  cumuPassenger=0.0;
  averageWaitTime3=0.0;
  averageWaitTime4=0.0;
  averageWaitTime5=0.0;

  numberOfBuses=0;
  reliableBuses=0;
  reliableBuses2=0;
  cumuOccupancy=0.0;
  averageOccupancy=0.0;
  cumuDeviation=0.0;
  averageDeviation=0.0;
  cumuModulusDev=0.0;
  avModulusDeviation=0.0;
  sumOfDeviationDiffer=0.0;
  sdOfDeviation=0.0;
  sumOfModulusDeviationDiffer=0.0;
  sdOfModulusDeviation=0.0;
  cumuBusJourney=0.0;
  averageBusJourney=0.0;
  cumuPassIn=0.0;
  averagePassIn=0.0;
  averagePassJourney=0.0;
  cumuPassJourney=0.0;
  cumuWaitTime=0.0;
  averageWaitTimePerBus=0.0;
```

```
   cumuPassAlighted=0.0;
   averagePassIn=0.0;
   cumuPassBoarded=0.0;
   averagePassBoarded=0.0;
   cumuDwellTime=0.0;
   averageDwellTime=0.0;
 }
 //-----------------------------------------------------------------------
 BusStop::~BusStop() {}
 int BusStop::busstopLinkNumber()  {return linkNos;}
 int BusStop::busstopIdentity()  {return busstopID;}
 int BusStop::busstopPosition()  {return busstopPosi;}
 //-----------------------------------------------------------------------
 int BusStop::headwayCalculate(int times,int busArrive2,int passInside,int
 schTime,int reserved,int serviceNo)
 {
  if(serviceNo>100) {deadTime = 6.85; passAlightingTime=1.69;
 passBoardingTime=9.00;}
  //{deadTime = 5.42; passAlightingTime=1.48; passBoardingTime=9.15;}
 //York's parameters
  else              {deadTime = 3.30; passAlightingTime=1.96;
 passBoardingTime=9.04;}
  //{deadTime = 3.55; passAlightingTime=1.99; passBoardingTime=9.18;}
 //York's parameters

  if(busAlready2!=busArrive2)
  {
   int timeBus2=times;
   if(reserved==0)
   {
    headway = timeBus2-timeBus1;
    timeBus1 = timeBus2;
   }
   else headway=0;

   if(headway<0) headway=0;
   passengerA2 = passInside*alightPassRate;
   passengerB2 = headway/boardPassRate;
   int tempPassengerB2=passengerB2;    //to consider passenger at dwell time
   if(passengerA2>0.0||passengerB2>0.0)
   {
   dwellTime2 =
deadTime+passengerA2*passAlightingTime+passengerB2*passBoardingTime;
    if(reserved==0)
    {
     int headwayTotal = headway+dwellTime2;      //for taking account of
passengers at dwelltime
     passengerB2 = headwayTotal/boardPassRate;
     dwellTime2 =
deadTime+passengerA2*passAlightingTime+passengerB2*passBoardingTime;
     passengerB2 = (headway+dwellTime2)/boardPassRate;  //This is thirs
iteration for passenger number
     timeBus1=timeBus2+dwellTime2;               //for changing headway into
time gap (depart-arrive)
    }
   }
   else {dwellTime2=0;}

   if((times+dwellTime2)<schTime)      //Check for scheduled timetable
   {
    dwellTime2=schTime-times;
    if(reserved==0)
    {
     int headwayTotal = headway+dwellTime2; //for taking account of passengers
at dwelltime
     passengerB2 = headwayTotal/boardPassRate;
     timeBus1=timeBus2+dwellTime2; //for changing headway into time gap
(depart-arrive)
    }
   }

  busAlready2=busArrive2;
```

```
   cumuHeadwaySquare+=headway*headway;
   cumuHeadway+=headway;
   if ((headway+dwellTime2)>0)
   {
    averageWaitTime=0.5*headway*headway/(headway+dwellTime2); //wait time
 using Gap for total Pass
   }
   else averageWaitTime=0;
   cumuHeadwaySquare5+=(headway+dwellTime2)*(headway+dwellTime2);
   cumuHeadway5+=(headway+dwellTime2);
   cumuPassWait+=0.5*headway*tempPassengerB2;
   cumuPassenger+=passengerB2;
 }
  return headway;
 }
 //-----------------------------------------------------------------------------
 float BusStop::AlightPass2()    {return passengerA2;}
 float BusStop::BoardPass2()     {return passengerB2;}
 int BusStop::DwellTime2()       {return dwellTime2;}
 float BusStop::WaitTime2()      {return averageWaitTime;}
 //-----------------------------------------------------------------------------
 int BusStop::DeviationInSchedule11(int times,int busFrequency)
 {
   int deviationInSchedule=times-scheduleStartTime11;
   scheduleStartTime11+=busFrequency;
   if (busstopID>9)  {scheduleStartTime11+=busFrequency; }
   return deviationInSchedule;
 }
 //-----------------------------------------------------------------------------
 int BusStop::DeviationInSchedule3(int times,int busFrequency)
 {
   int deviationInSchedule=times-scheduleStartTime3;
   scheduleStartTime3+=busFrequency;
   if (busstopID>9)  {scheduleStartTime3+=busFrequency; }
   return deviationInSchedule;
 }
 //-----------------------------------------------------------------------------
 int BusStop::DeviationInSchedule101(int times,int busFrequency)
 {
   int deviationInSchedule=times-scheduleStartTime101;
   scheduleStartTime101+=1800;//busFrequency;  // Frequency of 101 is 30 min.
   if (busstopID>9)  {scheduleStartTime101+=busFrequency; }
   return deviationInSchedule;
 }
 //-----------------------------------------------------------------------------
 int BusStop::ScheduleHeadway(int serviceNos)
 {
   int schHeadway;
   if(busstopID<7) {schHeadway=600;} //for 11&11a service up to Portswood
   if(busstopID>=7&&busstopID<=9)   //for 11&11a, 3&3a, 101 service from
 Portswood to Lodge road
   {
    if(serviceNos<100)
    {
     if
 (scheduleStartTime11>=scheduleStartTime101||scheduleStartTime101>scheduleStar
 tTime3){schHeadway=scheduleStartTime3-scheduleStartTime11;}
     else {schHeadway=scheduleStartTime3-scheduleStartTime101;}
    }
    if(serviceNos>100&&serviceNos<1000)
    {
     if
 (scheduleStartTime3>=scheduleStartTime101||scheduleStartTime101>scheduleStart
 Time11){schHeadway=scheduleStartTime11-scheduleStartTime3;}
     else {schHeadway=scheduleStartTime11-scheduleStartTime101;}
    }
    if(serviceNos>1000)
    {
     if
 (scheduleStartTime3>=scheduleStartTime11){schHeadway=scheduleStartTime101-
 scheduleStartTime3;}
```

```
      else {schHeadway=scheduleStartTime101-scheduleStartTime11;}
     }
    }
    if(busstopID>9) //for 11&3 service from Lodege Road to city centre
    {
     if(serviceNos<100) {schHeadway=scheduleStartTime3-scheduleStartTime11;}
     if(serviceNos>100) {schHeadway=scheduleStartTime11-scheduleStartTime3;}
    }
    //if(busstopID<7) {schHeadway=600;} //for 11&11a service up to Portswood
    return schHeadway;
  }
  //-----------------------------------------------------------------------
  int BusStop::BusstopHeadway()
  {
    int schHeadway;
    if(busstopID<7) schHeadway=600; //for 11&11a service up to Portswood
    if(busstopID>=7&&busstopID<=9)schHeadway=300;   //for 11&11a, 3&3a, 101
  service from Portswood to Lodge road
    if(busstopID>9) schHeadway=600;//for 11&3 service from Lodege Road to city
  centre
    return schHeadway;
  }
  //-----------------------------------------------------------------------
  int BusStop::ScheduleStartTime11()        {return scheduleStartTime11;}
  int BusStop::ScheduleStartTime3()         {return scheduleStartTime3;}
  int BusStop::ScheduleStartTime101()       {return scheduleStartTime101;}
  void BusStop::GetReserveBus1(int resNum)  {busstopReserved1=resNum;}
  void BusStop::GetReserveBus2(int resNum)  {busstopReserved2=resNum;}
  int BusStop::ReservedNum1()               {return busstopReserved1;}
  int BusStop::ReservedNum2()               {return busstopReserved2;}
  //-----------------------------------------------------------------------
  //This section is for generating individual passengers (not required at this
  stage)
  //-----------------------------------------------------------------------
  //-----------------------------------------------------------------------
  int BusStop::alightPass(int passInside, int busArrive1)
  {
    if(busAlready1!=busArrive1)
    {
      passengerA = alightPassRate*passInside;  //alightPassRate is passenger
  arrival rate
      busAlready1=busArrive1;
      finalWaitTime=waitTime;  //for storing the waiting time refered to
  arrival of bus
    }
    return passengerA;
  }
  //-----------------------------------------------------------------------
  int BusStop::boardPassGenerate(int times)
  {
    int addPass=0;

    int genTime = times-startTime;   //startTime is busstop start time
    if (genTime>=passGenerateFactor*boardPassRate)
    {
      addPass=1;
      passGenerateFactor++;
    }

    int numberPass2 = numberPass1+addPass;
    //if(numberPass2<1) numberPass2 =0 ;   //there is a problem if I don't use
  this statement
    if(numberPass2<1) numberPass2 =0;
    waitTime=waitTime+numberPass1*1;///timeFactor//1 is time factor
    numberPass1=numberPass2;
    return numberPass2;
  }
  //-----------------------------------------------------------------------
  int BusStop::boardPassDischarge(int times, int busArrive, int alightPass)
  {
    int delPass=0;
    if(busAlready!=busArrive)
    {
      checkinTime = times;
      busAlready = busArrive;
      passDischargeFactor=1;
      alightPassTime = alightPass*passAlightingTime;
      numberBoardPass=0;
    }
    int alightStartTime=times-checkinTime-8-alightPassTime; //8 is dead time &
 alightPassTime is time for pasenger alighting
    if (alightStartTime>=passDischargeFactor*passBoardingTime)
    {
      delPass=1;
      passDischargeFactor++;
    }
    //int *change = &numberPass1;
    //*change -=delPass;         //Instead I could have done
    numberPass1=numberPass1-delPass;//0;
    numberBoardPass=numberBoardPass+delPass;
    totalDwellTime=times-checkinTime;
    return numberPass1;
 }
 //------------------------------------------------------------------------------
 int BusStop::boardPassNos()      {return numberBoardPass;}
 int BusStop::DwellTime()         {return totalDwellTime;}

 int BusStop::waitingTime(int timeFactor)
 {
   //int outWaitTime=waitTime/timeFactor-numberBoardPass*passBoardingTime;//to
 make the wait time calculation at the start of boarding
   //int outWaitTime=finalWaitTime/timeFactor;
   if(numberBoardPass==0) numberBoardPass=1;
   int outWaitTime=(waitTime/timeFactor)/numberBoardPass;
   waitTime=0;
   return outWaitTime;
 }
 //------------------------------------------------------------------------------
 int BusStop::PassGenerateExpo(int meanPassHeadway)
 {
   //randomize();
   //float valuePx1 = rand()%400;
   //int valueX1 = (((log((valuePx1/100000)*60))*-60)+240); //for mean =300,
 st. dev = 60

   int largestHeadway=10*meanPassHeadway;
   float dividerValue=1000.0*meanPassHeadway;

   int valueX1=0;
   if(meanPassHeadway==99999) valueX1=99999; //to ban generating passengers at
 exit
   while(valueX1==0)
   {
     float valuePx;
     int valueX = rand()%largestHeadway;
     valuePx = ((exp(-1*(valueX/meanPassHeadway)))/meanPassHeadway);  //for
 mean =300, st. dev = 60

     float valueY = (rand()%1000)/dividerValue;
     if(valueY<=valuePx)
     {
       valueX1= valueX;
     }
     else valueX1=0;
   }
   return valueX1;
 }
 //------------------------------------------------------------------------------
 //This section is for Output generation purpose only
 //------------------------------------------------------------------------------
 //------------------------------------------------------------------------------
 void BusStop::GenerateOutput(float occupancy,float passIn,int deviation,int
 busJourney,float avWaitTime,float passAlighted,float passBoarded,float
 dwellTimeAtStop)
```

```
{
 numberOfBuses++;
 if (deviation >-60&&deviation<180) reliableBuses++;
 if (deviation >-60&&deviation<300) reliableBuses2++;
 cumuOccupancy+=occupancy;
 averageOccupancy=cumuOccupancy/numberOfBuses;
 cumuDeviation+=deviation;
 averageDeviation=cumuDeviation/numberOfBuses;
 int modulusDeviation=sqrt(deviation*deviation);
 cumuModulusDev+=modulusDeviation;//sqrt(deviation*deviation);
 avModulusDeviation=cumuModulusDev/numberOfBuses;
 sumOfDeviationDiffer+=(deviation*deviation);
 if(numberOfBuses>1)sdOfDeviation=sqrt(sumOfDeviationDiffer/(numberOfBuses-
1));
 sumOfModulusDeviationDiffer+=((modulusDeviation)*(modulusDeviation));
 if(numberOfBuses>1)sdOfModulusDeviation=sqrt((sumOfModulusDeviationDiffer-
numberOfBuses*avModulusDeviation*avModulusDeviation)/(numberOfBuses-1));
 cumuBusJourney+=busJourney;
 averageBusJourney=cumuBusJourney/numberOfBuses;
 cumuPassIn+=passIn;
 averagePassIn=cumuPassIn/numberOfBuses;
 averagePassJourney=averagePassIn*averageBusJourney;
 cumuPassJourney+=(passIn*busJourney);
 cumuWaitTime+=(avWaitTime*passBoarded);
 averageWaitTimePerBus=cumuWaitTime/numberOfBuses;
 cumuPassAlighted+=passAlighted;
 averagePassAlighted=cumuPassAlighted/numberOfBuses;
 cumuPassBoarded+=passBoarded;
 averagePassBoarded=cumuPassBoarded/numberOfBuses;
 cumuDwellTime+=dwellTimeAtStop;
 averageDwellTime=cumuDwellTime/numberOfBuses;
}

int BusStop::NumberOfBuses()              {return numberOfBuses;}
int BusStop::ReliableBuses()              {return
reliableBuses*100/numberOfBuses;}
int BusStop::ReliableBuses2()             {return
reliableBuses2*100/numberOfBuses;}
float BusStop::AverageScheduleDeviation() {return averageDeviation;}
float BusStop::SdOfDeviation()            {return sdOfDeviation;}
float BusStop::AverageModulusDeviation()  {return avModulusDeviation;}
float BusStop::SdOfModulusDeviation()     {return sdOfModulusDeviation;}
float BusStop::AverageOccupancy()         {return averageOccupancy;}
float BusStop::AverageBusJourney()        {return averageBusJourney;}
float BusStop::TotalBusJourney()          {return cumuBusJourney;}
float BusStop::AveragePassJourney()       {return averagePassJourney;}
float BusStop::TotalPassJourney()         {return cumuPassJourney;}
float BusStop::WaitingTimePerBus()        {return averageWaitTimePerBus;}
float BusStop::TotalWaitingTime()         {return cumuWaitTime;}
float BusStop::AveragePassAlighted()      {return averagePassAlighted;}
float BusStop::AveragePassBoarded()       {return averagePassBoarded;}
float BusStop::AverageDwellTime()         {return averageDwellTime;}
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_SIGNAL)
#define EXAMPLE_BUS_SIGNAL

#include <iostream.h>

class Signal
{
  public:
    Signal(int signalId,int sideSig,int linkNo,float signalP,int cycleT,
           int greenT,int amberT,int offset,float flowArm, int satFlow,
           int maxExtend1, int maxRecal1,int maxExtend2,int maxRecal2,int
recallRestrict);
    ~Signal();
    int signalIdentity();
    int SideSignalNumber();
    int signalLinkNumber();
    float signalPosition();
    int GreenStartTime(int times, int busNumber2);
    void GetRecall(int times, int cycleNos);
    void GetExtension(int times, int cycleNos);
    int CycleNumber(int times);
    int signalStage(int times);
    int SignalStageState();
    int TotalGreenTime();
    void CalculateCarNumber(int times);
    void CalculateCarNumber1(int times);
    int newCarNumber();
    int frontCarNumber(int busNumber2);
    int frontCarFirst();
    int carPositionNos(int busNumber2);
    int carPosition(int nums);//int times, int busNumber2);
    void changeFrontCarNos();
    int LockBusNos();
    int demoFrontCarNumber();
    int EndGreenState(int times);
    int StartGreenState(int times);
    int RecallRestriction();

    int DelayCarNumber();
    int TotalDelayCarNumber();
    float AverageCarDelay(int totalTime);
    int MaxExtensionTime();
    int MaxRecallTime();
    void PriorityOption(int logicNo, int lateTime, int scheduleTime);

    void GetBusDelay(int busDelay);
    float AverageBusDelay();


    private:
    int signalID;
    int sideSignalNos;
    int linkNos;
    float signalPosi;
    int cycleTime;
    int greenTime;
    int amberTime;
    int offsetTime;
    int maxExtension1;
    int maxRecall1;
    int maxExtension2;
    int maxRecall2;
    int recallRestrictionTime;

    int stageNumber;
    int numberCar;
    int numberCar1;
    int numberCar2;
    int addCars;
    int delCars;
    int numberFrontCar;
    int busNumber1;
```

```
     float flowArms;
     int satFlows;
     int carGenerateFactor;
     int carDischargeFactor;
     int delCar;
     int busNum1;
     int numberFrontCar1;
     int carDischargeFactor1;
     int frontCarAtFirst;
     int greenTimes;

     int busCarNum1;
     int numberFrontCar11;
     int numberFrontCar111;

     int busNumbs1;
     int greenStartTime;
     int lockBusNumber;

     int recallCycleNumber;
     int givenRecallTime;
     int recallTime;
     int extendCycleNumber;
     int givenExtensionTime;
     int extensionTime;
     int extensionTimeAllowed;
     int recallTimeAllowed;
     int previousExtensionTime;
     int previousRecallTime;
     int totalGreenTime;

     int delayToCars;
     int maxCarNumber;

     int totalDelayCarNumber;
     int recallCounter;
     int  recallCounter1;

     int numberOfBuses;
     float cumuBusDelay;
     float averageBusDelay;

};

#endif // end of !defined(EXAMPLE_RACE_TRACK) check
```

```
#include "signal.h"


Signal::Signal(int signalId,int sideSig,int linkNo,float signalP,int cycleT,
               int greenT, int amberT,int offset,float flowArm,int satFlow,
               int maxExtend1, int maxRecal1,int maxExtend2,int maxRecal2,int
recallRestrict)
{
 signalID = signalId;
 sideSignalNos=sideSig;
 linkNos = linkNo;
 signalPosi = signalP;
 cycleTime = cycleT;
 greenTime = greenT;
 amberTime = amberT;
 offsetTime = offset;         //offset helps to start with the phase wanted
 stageNumber=0;
 numberCar1 =0;
 numberCar2 =0;
 addCars=0;
 numberFrontCar=0;
 busNumber1=100;
 flowArms = flowArm;
 satFlows = satFlow;
 maxExtension1=maxExtend1;
 maxRecall1=maxRecal1;
 maxExtension2=maxExtend2;
 maxRecall2=maxRecal2;
 recallRestrictionTime=recallRestrict;
 carGenerateFactor=1;  //for generating the car after First interval
 carDischargeFactor=1; //for discharging the car after First interval
 delCar=0;
 busNum1=100;
 numberFrontCar1=0;
 carDischargeFactor1=1;
 frontCarAtFirst=0;
 busNumbs1=100;
 greenTimes=0;

 totalDelayCarNumber=0;

 greenStartTime=0;

 busCarNum1=100;   numberFrontCar11=0; numberFrontCar111=0;
 lockBusNumber=0;
 recallCycleNumber=0;  givenRecallTime=0; recallTime=0;
 extendCycleNumber=0;  givenExtensionTime=0; extensionTime=0;
 extensionTimeAllowed=0;   recallTimeAllowed=0;
 previousExtensionTime=0;  previousRecallTime=0;
 totalGreenTime=0;


 maxCarNumber=0;
 recallCounter=0;  recallCounter1=0;
 numberOfBuses=0;  cumuBusDelay=0.0;  averageBusDelay=0.0;
}
//-------------------------------------------------------------------------
Signal::~Signal()            {}
int Signal::signalIdentity()   {return signalID;}
int Signal::SideSignalNumber() {return sideSignalNos;}
int Signal::signalLinkNumber() {return linkNos;}
float Signal::signalPosition() {return signalPosi;}
//-------------------------------------------------------------------------
int Signal::GreenStartTime(int times, int busNumber2)
{
  if (busNumber2!=busNumbs1)
  {
    busNumbs1=busNumber2;
    greenStartTime=times;
  }
  return greenStartTime;
```

```
}
//---------------------------------------------------------------------------
int Signal::EndGreenState(int times)
{
  int checkTime = (times+offsetTime)%cycleTime;
  int endGreenState=checkTime-(greenTime-1);   //1 is deducted because Green
starts from 0 not 1
  if (endGreenState<0) endGreenState=0;
  return endGreenState;
}
//---------------------------------------------------------------------------
int Signal::StartGreenState(int times)
{
  int checkTime = (times+offsetTime)%cycleTime;
  int startGreenState=cycleTime-checkTime;
  return startGreenState;
}
//---------------------------------------------------------------------------
int Signal::RecallRestriction()   {return recallRestrictionTime;}
//---------------------------------------------------------------------------
int Signal::CycleNumber(int times)
{
  int cycleNumbers=(times+offsetTime)/cycleTime;
  return cycleNumbers;
}
//---------------------------------------------------------------------------
void Signal::GetRecall(int times, int cycleNos)
{
  givenRecallTime = times;
  recallCycleNumber =cycleNos;
}
//---------------------------------------------------------------------------
void Signal::GetExtension(int times, int cycleNos)
{
  givenExtensionTime = times;
  extendCycleNumber =cycleNos;
}
//---------------------------------------------------------------------------
int Signal::signalStage(int times)
{
  //int stageNumber;
  int cycleNum=(times+offsetTime)/cycleTime;
  if (cycleNum==recallCycleNumber)
  {
   recallTime=givenRecallTime;
   previousRecallTime=recallTime;
  }
  else recallTime=0;
  if (cycleNum==recallCycleNumber+1)
  {
   offsetTime=offsetTime+previousRecallTime;
   recallCycleNumber=-1;
  }

  if (cycleNum==extendCycleNumber)
  {
   extensionTime=givenExtensionTime;
   previousExtensionTime=extensionTime;
  }
  else extensionTime=0;
  if (cycleNum==extendCycleNumber+1)
  {
   offsetTime=offsetTime-previousExtensionTime;
   extendCycleNumber=-1;
  }

  if(extensionTime<0)      //side road -ve extension by using offset to
reduce green and cycle time
  {
   if (cycleNum==extendCycleNumber)
   {
    offsetTime=offsetTime-previousExtensionTime;
```

```
     extendCycleNumber=-1;
     extensionTime=0;
    }
  }
  if(recallTime<0)extensionTime=givenRecallTime;  //For side signal
 shortening of green time page 34

  int timing = times+offsetTime;
  if (recallTime>0) timing=timing%cycleTime;      //Very important (page 28)
  if (recallTime<0)
  {
   timing=timing%cycleTime+cycleTime;  //Very important (page 51) for
 comparing with extended cycle time > common cycle time
   //extensionTime=recallTime;      //This is to decrease Green period in
 case of -ve Recall
   extensionTime=0;
  }
  int checkTime = timing%(cycleTime-recallTime); //Recalling means shortening
 of CycleTime

  //if(extensionTime<0)extensionTime=0;   //not to change the length of side
 road green time

  if(checkTime<greenTime+extensionTime)
  {
   stageNumber=1;    //Green stage
   totalGreenTime++;
  }
  else if(checkTime<(greenTime+extensionTime+amberTime))
  {
    stageNumber = 2;     //Amber stage
  }
  else
  {
    stageNumber=3;      //Red stage
  }
  return stageNumber;
}

//---------------------------------------------------------------------------
int Signal::SignalStageState()
{
  return stageNumber;
}
//---------------------------------------------------------------------------
int Signal::TotalGreenTime()
{
  return totalGreenTime;         //this is for checking change in Green due
to priority
}
//---------------------------------------------------------------------------
void Signal::CalculateCarNumber(int times)
{
  int addCar=0;
  delCar=0;
  float carGenerateTime=300.0/flowArms;
  float carDischargeTime=3600.0/satFlows;

  if (times>=carGenerateFactor*carGenerateTime)
  {
    addCar=1;
    carGenerateFactor++;
  }

  if(stageNumber!=1)
  {
   carDischargeFactor=1;
   greenTimes=0;
  }
  else if(stageNumber==1)
  {
    greenTimes++;
```

```
      if (greenTimes>=carDischargeFactor*carDischargeTime)
      {
        delCar=(greenTimes/carDischargeTime)-(carDischargeFactor-1);
        carDischargeFactor+=delCar;
      }
    }
    numberCar2 = numberCar1+addCar-delCar;
    if(numberCar2<1) numberCar2 =0 ;    //there is a problem if I don't use this
 statement
    //if(checkTime==greenTime)numberCar2=0; //this is for deleting the last
 vehicle appearing just befor red time
    delayToCars=delayToCars+numberCar2*1;
    if(numberCar2>numberCar1) maxCarNumber=numberCar2;
    totalDelayCarNumber+=numberCar2;        //this is for calculation of Delays
 to cars at signal
    numberCar1=numberCar2;
 }
 //-----------------------------------------------------------------------
 int Signal::newCarNumber()
 {
    return numberCar2;
 }
 //-----------------------------------------------------------------------
 int Signal::frontCarNumber(int busNumber2)  //to discharge bus if no car
 infront
 {
    if (busNumber2!=busNum1)
    {
      busNum1=busNumber2;
      numberFrontCar1 = numberCar2;
      frontCarAtFirst = numberCar2;
    }
    else {numberFrontCar1-= delCar;}
    if (numberFrontCar1<0) numberFrontCar1=0;
    return numberFrontCar1;
 }
 //-----------------------------------------------------------------------
 int Signal::frontCarFirst()     {return frontCarAtFirst;}   //for output
 //-----------------------------------------------------------------------
 int Signal::carPositionNos(int busNumber2)
 {
    if (busNumber2!=busCarNum1)
    {
      busCarNum1=busNumber2;
      numberFrontCar111 = numberCar2;
      lockBusNumber=busNumber2;
    }
    else {numberFrontCar111-= delCar;}
    if (numberFrontCar111<0) numberFrontCar111=0;
    return numberFrontCar111;
 }
 //-----------------------------------------------------------------------
 int Signal::carPosition(int nums)//int times, int busNumber2)
 {
    int positionCar;
    //if(numberFrontCar111<1) numberFrontCar11=numberCar2;
    if(nums==0) numberFrontCar11=numberCar2;
    if(nums>=1) numberFrontCar11=numberFrontCar111;
    //if(numberFrontCar111>=1) numberFrontCar11=numberFrontCar111;
    if(numberFrontCar11<0) numberFrontCar11 =0 ;

    if(numberFrontCar11>=1) positionCar = signalPosi-
 numberFrontCar11*4.0;//2.5;
    else positionCar = 5000;
    return positionCar;
 }
 //-----------------------------------------------------------------------
 void Signal::changeFrontCarNos()    {numberFrontCar111=0;}
 int Signal::LockBusNos()            {return lockBusNumber;}
 int Signal::demoFrontCarNumber()    {return numberFrontCar11;}

 int Signal::DelayCarNumber()         {return numberCar2;}
```

```
int Signal::TotalDelayCarNumber()   {return totalDelayCarNumber;}
//---------------------------------------------------------------------------
float Signal::AverageCarDelay(int totalTime)
{
  float averageCarDelay=totalDelayCarNumber/(totalTime*flowArms/300);
  return averageCarDelay;
}
//---------------------------------------------------------------------------
void Signal::GetBusDelay(int busDelay)
{
 numberOfBuses++;
 cumuBusDelay+=busDelay;
 averageBusDelay=cumuBusDelay/numberOfBuses;
}
//---------------------------------------------------------------------------
float Signal::AverageBusDelay()  {return averageBusDelay;}
//---------------------------------------------------------------------------
//This is Bus priority section
//---------------------------------------------------------------------------
void Signal::PriorityOption(int logicNo, int lateTime, int scheduleTime)
{
  //scheduleTime is used in case of ratio which is scheduled headway 600 secs
  switch (logicNo)
  {
    case 1:   //for No priority
    {
     extensionTimeAllowed=0;
     recallTimeAllowed=0;
     break;
    }
    case 2:     //for SVD priority with Normal DoS/ High DoS
    {
      extensionTimeAllowed=maxExtension1;
      recallTimeAllowed=maxRecall1;
      break;
    }
    case 3:   //Differential with Normal DoS
    {
      //if(lateTime/scheduleTime>=0.0)    //based on ratio
      if(lateTime>0)                     //based on difference
      {
       extensionTimeAllowed=maxExtension1;
       recallTimeAllowed=maxRecall1;
      }
      else
      {
        extensionTimeAllowed=0;
        recallTimeAllowed=0;
      }
      break;
    }
    case 4:   //Differential with High DoS
    {
      if(lateTime>0)                     //based on difference
      {
       extensionTimeAllowed=maxExtension2;
       recallTimeAllowed=maxRecall2;
      }
      else
      {
        extensionTimeAllowed=0;
        recallTimeAllowed=0;
      }
      break;
    }
    case 5: //Differential with HighDoS for Lateness>60 and Low for >0
    {
      //if(lateTime/scheduleTime>=0.0)    //based on ratio
      if(lateTime>60)                    //based on difference
      {
       extensionTimeAllowed=maxExtension2;
       recallTimeAllowed=maxRecall2;
       }
       else if(lateTime>0)                    //based on difference
       {
        extensionTimeAllowed=maxExtension1;
        recallTimeAllowed=maxRecall1;
       }
       else
       {
        extensionTimeAllowed=0;
        recallTimeAllowed=0;
       }
       break;
     }
     case 6: //Differential and SVD
     {
       //if(lateTime/scheduleTime>=0.0)    //based on ratio
       if(signalID<8)                     //based on difference
       {
        if(lateTime>0)                     //based on difference
        {
         extensionTimeAllowed=maxExtension1;
         recallTimeAllowed=maxRecall1;
        }
        else
        {
         extensionTimeAllowed=0;
         recallTimeAllowed=0;
        }
       }
       else                    //based on difference
       {
        extensionTimeAllowed=maxExtension1;
        recallTimeAllowed=maxRecall1;
       }
       break;
     }
     case 7: //Differential and SVD
     {
       if(signalID<8)                     //based on difference
       {
        if(lateTime>60)                     //based on difference
        {
         extensionTimeAllowed=maxExtension2;
         recallTimeAllowed=maxRecall2;
        }
        else if(lateTime>0)                    //based on difference
        {
         extensionTimeAllowed=maxExtension1;
         recallTimeAllowed=maxRecall1;
        }
        else
        {
         extensionTimeAllowed=0;
         recallTimeAllowed=0;
        }
       }
       else                    //based on difference
       {
        extensionTimeAllowed=maxExtension1;
        recallTimeAllowed=maxRecall1;
       }
       break;
     }
   }
}
//-----------------------------------------------------------------------------
int Signal::MaxExtensionTime()        {return extensionTimeAllowed;}
int Signal::MaxRecallTime()           {return recallTimeAllowed;}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_VIRTUALDETECTOR)
#define EXAMPLE_BUS_VIRTUALDETECTOR

#include <stdlib.h>
#include <iostream.h>


class VirtualDetector
{
  public:
    VirtualDetector(int vDetectId,int sigId,int vDetectP,int vDetectDist);
    ~VirtualDetector();
    int VirtualDetectorIdentity();
    int SignalIdentity();
    int VirtualDetectorPosition();
    int VirtualDetectorDistance();

  private:
    int virtualDetectorID;
    int signalID;
    float virtualDetectorPosi;
    int virtualDetectorDist;
};

#endif // end of !defined(EXAMPLE_BUS_VirtualDetector) check
```

```
#include "virtualDetector.h"


VirtualDetector::VirtualDetector(int vDetectId,int sigId,int vDetectP,int
vDetectDist)
{
  virtualDetectorID = vDetectId;
  signalID = sigId;
  virtualDetectorPosi = vDetectP;
  virtualDetectorDist = vDetectDist;
}
//------------------------------------------------------------------------
VirtualDetector::~VirtualDetector()    {}
int VirtualDetector::VirtualDetectorIdentity()  {return virtualDetectorID;}
int VirtualDetector::SignalIdentity()           {return signalID;}
int VirtualDetector::VirtualDetectorPosition()  {return virtualDetectorPosi;}
int VirtualDetector::VirtualDetectorDistance()  {return virtualDetectorDist;}
//------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_Beacon)
#define EXAMPLE_BUS_Beacon

#include <stdlib.h>
#include <iostream.h>


class Beacon
{
  public:
    Beacon(int beaconId,int linkNo,int beaconP);
    ~Beacon();
    int BeaconIdentity();
    int BeaconPosition();
    void GetBeaconColor();
    int BeaconColor();


  private:
    int beaconID;
    float beaconPosi;
    int beaconColor;

};

#endif // end of !defined(EXAMPLE_BUS_Beacon) check
```

```
#include "Beacon.h"


Beacon::Beacon(int beaconId,int linkNo,int beaconP)
{
  beaconID = beaconId;
  beaconPosi = beaconP;
  beaconColor=0;
}
//-----------------------------------------------------------------------------
Beacon::~Beacon()             {}
int Beacon::BeaconIdentity()   {return beaconID;}
int Beacon::BeaconPosition()   {return beaconPosi;}
void Beacon::GetBeaconColor()  {beaconColor=3;}
//-----------------------------------------------------------------------------
int Beacon::BeaconColor()
{
  if (beaconColor>0)beaconColor--;
  return beaconColor;
}
//-----------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_BUSLIST)
#define EXAMPLE_BUS_BUSLIST

#include <iostream.h>
#include "bus.h"
#include "busitem.h"

class BusList
{
  public:
    BusList();
    virtual ~BusList();
    void addBus(Bus* abus);
    int getLength();
    Bus* getBus(int index);
    Bus* exitBus(int index);

  protected:
    BusListItem* head;
    BusListItem* stop;
};

#endif
```

```
#include "buslist.h"

//--------------------------------------------------------------------------
BusList::BusList()  {head = NULL;}
BusList::~BusList() {if(head != NULL) delete head;}
//--------------------------------------------------------------------------
void BusList::addBus(Bus* abus)
{
  if(head == NULL) head = new BusListItem(abus);
  else head->addBus(abus);
}
//--------------------------------------------------------------------------
int BusList::getLength()
{
  if(head == NULL) return 0;
  else
  {
    int count = 0;
    BusListItem* current = head;
    while(current != NULL)
    {
      count++;
      current = current->next;
    }
    return count;
  }
}
//--------------------------------------------------------------------------
Bus* BusList::getBus(int index)
{
  if(index > getLength()) return NULL;
  else
  {
    BusListItem* current = head;
    while(current != NULL && index > 0)
    {
      current = current->next;
      index--;
    }
    return current->abus;
  }
}
//--------------------------------------------------------------------------
Bus* BusList::exitBus(int index)
{
  if(index >= getLength()) return 0;
  else
  {
    BusListItem* current = head;
    BusListItem* previous = NULL;
    while(current != NULL && index > 0)
    {
      previous = current;
      current = current->next;
      index--;
    }
    if(previous == NULL)
    {
      head = current->next;             //removing the first element
      stop = current;
    }
    else
    {
      previous->next = current->next;
      stop = current;
    }
    return 0;
  }
}
//--------------------------------------------------------------------------
```

```
#if !defined(EXAMPLE_BUS_BUSLISTITEM)
#define EXAMPLE_BUS_BUSLISTITEM


#include <iostream.h>

class BusListItem;
#include "buslist.h"

class BusListItem
{
  friend class BusList;

  public:
    BusListItem(Bus* a);
    virtual ~BusListItem();
    Bus* getBus();

  protected:
    BusListItem* next;
    Bus* abus;
    void addBus(Bus* toadd);
};

#endif
```

```
#include "busitem.h"

//---------------------------------------------------------------------------
BusListItem::BusListItem(Bus* a)
{
  abus = a;
  next = NULL;
}
//---------------------------------------------------------------------------
BusListItem::~BusListItem()
{
  if (next != NULL) delete next;
}
//---------------------------------------------------------------------------
Bus* BusListItem::getBus()
{
  return abus;
}
//---------------------------------------------------------------------------
void BusListItem::addBus(Bus* toadd)
{
  if(next != NULL) next->addBus(toadd);
  else next = new BusListItem(toadd);
}
//---------------------------------------------------------------------------
```