National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

Study of Bus Control Strategies By Computer Simulation

by

Lule Chen

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree
of Master of Science (Systems Science)

Ottawa, Ontario, 1994

ISBN 0-612-00453-8

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Lule Chen

I further authorize the University of Ottawa to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Lule Chen

ABSTRACT

In this thesis, a simulation model used to simulate the running of buses on a single route, OC Transpo route 95, is studied. The model is implemented on a platform different from its original design at the University of Ottawa, and validation experiments are performed. For the most part, the model seems valid, although some additional calibration is recommended.

New headway-based bus control strategies are presented and compared with other known control strategies. The statistical analysis and comparisons of those control strategies are based on the testing results obtained from the simulation model. The new headway control policies (called "Message Board" and "hybrid" policies) are shown to provide better service reliability in terms of less variable headway for the high-frequency route 95.

Finally, a prototype of a knowledge-based system for real-time bus control purposes is developed and tested on the simulation model. The prototype is shown to be an interesting alternative for real-time bus control: it can effectively interface with the simulation model and perform real-time service control. This prototype will be a building block in the development of a more complete knowledge-based system capable of implementing more complex real-time service control policies in the hope of providing assistance to human controllers.

ACKNOWLEDGEMENTS

I wish to express my most sincere gratitude to Dr. François M. Julien, my supervisor, for his advice, valuable guidance and support, encouragement in this research work and great effort in helping me to prepare this thesis.

I would like to give special thanks to Prof. Jeffrey Sidney, my co-supervisor, for much appreciated suggestions, valuable support and guidance in this thesis work.

I would like to take this opportunity to express my most sincere thanks to Dr. Helen Gault and Mr. Joel Koffman, from the Systems Planning Department at OC Transpo, for providing me such an interesting project and much needed support.

Profound thanks are also extended to Prof. Dan Lane for his advice and encouragement in this thesis work, to Mr. Harry Boumeester of the Canadian Institute of Guided Ground Transportation, Queens University, for his valuable time and effort in answering my questions in this thesis work, and to Dr. Tony Quon for providing assistance regarding the statistical analysis performed in this research.

DEDICATION

To my parents Mr. & Mrs Chen and my wife Zeying

Table of Contents

1. Introduction and Literature Review

The goals of a transit agency are to provide good service to the public and to do so at the lowest possible cost. Good service means, among other things, a minimum of waiting time at stops and transfer points, reliable schedules, short travel times, uncrowded vehicles and safe, courteous service. To achieve those goals, the running of the vehicles should be controlled according to some useful targets. Traditionally, on-street supervisors have been used to check schedule adherence of vehicles. The effectiveness of this method is restricted to some extent because of its labour intensive nature. In recent years, the advent of new technologies have made it possible for public transit agencies to control the vehicles remotely in real time by means of an *Automatic Vehicle Location and Control* (AVLC) system, a much more efficient and powerful tool for on-street vehicle control. Furthermore, buses may be equipped with an *Automatic Passenger Counting* (APC) system, which can provide management with useful information about passenger boarding and alighting activities, among other things, which may be used to facilitate planning and improve schedules.

With the installation and the operation of AVLC and APC systems in transit agencies, the important challenges of optimizing service reliability and minimizing the cost of operations of the systems arise. The first task in this endeavour is to examine and analyze ways to improve the control of vehicles on the street. Since both the AVLC and APC systems produce large amounts of information, it is impossible for a human controller to process this much information and use it effectively in a very short period of time (in an on-line environment reaction to incidents should be almost instantaneous). Hence a knowledge-based expert system that can assist controllers in their decision making process could provide an interesting approach to bus control. Such a knowledge-based system could be used to help controllers in taking on-line control actions, could automatically perform dynamic re-scheduling, and could electronically communicate appropriate control actions to all drivers along the route.

The purpose of this thesis is to provide the first step towards the development of such a knowledge-based expert system that can provide on-line vehicle control assistance. In order to develop such a system, the necessary *knowledge* should be acquired from all available sources. We decided to work with a transit agency, the Ottawa-Carleton Regional Transit Commission

(OC Transpo), to get better insight on the relevant issues and to base our analysis on their expertise and real data.

With AVLC systems, which allow controllers to determine the relative position of all buses along the route and to communicate control actions to drivers, typically through radio channels, the implementation of more sophisticated real-time control strategies becomes more applicable in practice. Furthermore, the new technology motivates the study of new control strategies that would otherwise be difficult, if not impossible to implement in practice. In this thesis, we also propose new real-time control strategies and evaluate the effectiveness of these and of other known real-time control strategies on providing quality service to users. As these are new control strategies, no knowledge as to their effectiveness can be acquired from past experience. So it is necessary to acquire knowledge through simulation experimentation. Our analysis is made with the help of a sophisticated simulation model, which was developed by CIGGT (Canadian Institute of Guided Ground Transportation), designed to accurately simulate the movement of buses along a single route. The ultimate goal is to acquire knowledge, understanding and insight into the effectiveness of various control strategies and actions which can eventually be incorporated in a knowledge-based system that could be connected to the OC Transpo AVLC system.

The thesis is composed of six chapters. In the second chapter, we provide a brief description of the simulation model. In the third chapter of this thesis, we report some preliminary validation testing of the simulation model. In the fourth chapter, we present a new real-time control policy whose purpose is to improve bus regularity on a high frequency route (more than 15 buses per hour) by reducing headway variability. The headway represents the time difference between the departure of two consecutive buses from a given stop. The effectiveness of the new policy is compared to that of other known real-time control policies using the simulation model, as part of the knowledge acquisition process. In the fifth chapter, we present a prototype of a Knowledge-Based System for bus control purposes, and some testing results of the prototype on the simulation model. In the last chapter we present some general conclusions and suggest directions for future research.

1.1 Description of OC Transpo

OC Transpo provides bus services to the public in the Ottawa - Carleton region. In 1992, 2,203 employees were employed by OC Transpo, 820 buses were in service, the service area covered 138 routes and 5,000 stops and served a population of 603,800.

OC Transpo has implemented an AVLC system which provides real-time bus monitoring and service control capabilities. The main functions of the system are:

- to monitor and track the location of all buses at strategic points throughout the service region;
- to display to controllers exceptions or incidents
  1) schedule violation on infrequent routes,
  2) schedule and/or spacing violations on frequent routes;
- to assist in two-way communication between the bus operator and central control.

The APC system, which provides basic information on passenger loads, passenger boarding and alighting, and travel times between stops, has been installed on approximately 10% of OC Transpo's fleet. Both the AVLC and APC data have been used for validation of the simulation model, comparisons of the different control strategies and tests of the prototype of the knowledge-based system for on-line bus control on the simulation model in this thesis.

1.2 Literature Review

The urban transit service literature is examined below; also, some applications of artificial intelligence relevant to the proposed work have been reviewed. Specifically, the areas of research reviewed include:

1) - measurements of service performance,
2) - new technologies,
3) - control strategies,
4) - applications of artificial intelligence/expert systems.

1.2.1. Measurements of Service Performance

Abkowitz et al [2] presents an overview of the subject of transit service reliability, and proposes a formal definition of service reliability as the invariability of service attributes which influence the decisions of travellers and transportation providers. Shields [22] describes a system

set up by London Transport to monitor the quality of its bus services. The service quality is mainly measured by two types of indicators: indicators of regularity and indicators of punctuality. Some of the results in [22] which are relevant to the current work are now described.

Service frequency is usually defined using headway: for example, low-frequency services will be those with planned headway greater than 10 minutes (less than six buses per hour), mid-frequency services will have planned headway between 4 and 10 minutes, while high-frequency services will have planned headway less than 4 minutes (more than 15 buses per hour). For a frequent service, with planned headway less than 4 minutes, passengers are less concerned with schedule adherence than with regularity of service. Assuming a Poisson arrival process for passengers at each stop (which is reasonable for high frequency services), the passenger waiting time is a function of the average and variance of the actual headway and is given by the following equation derived by Welding [28] and Kulash [15]

$$AWT = \frac{\sum h^2}{2\sum h} = \frac{\bar{h}}{2}\left(1+\left(\frac{S.D.(h)}{\bar{h}}\right)^2\right) \qquad (1.1)$$

where $\bar{h}$ and $S.D.(h)$ are the mean and standard deviation of the observed headway. The coefficient of variation of the headway is given by:

$$\textit{Coefficient of Variation } (CV) = \frac{S.D.(h)}{\bar{h}} \qquad (1.2)$$

The regularity index is defined as

$$\textit{Regularity Index } (RI) = \frac{AWT}{\bar{h}} \qquad (1.3)$$

Sheilds [22] also mentions that for an infrequent service, the punctuality of a bus is more important than its regularity. To indicate the punctuality an Excess Waiting Time (EWT) function needs to be developed. OC Transpo uses the percentages of which buses arrive early, on time and late at each stop to indicate the punctuality of its services. If a bus arrives at a stop earlier than

its scheduled time, it is defined to be early. If a bus arrives at a stop three minutes or more later than its scheduled time, it is defined to be late; otherwise it is defined to be on time.

1.2.2 New Technologies

Heti [11] provides an overview of the development of Automatic Vehicle Location and Control (AVLC) systems in Europe and North America. The overview gives a clear indication that hardware and software related problems of AVLC systems have largely been solved. The problem is no longer the implementation of such a system, but rather with using it to maximize the effectiveness of the transit system. Problems appeared to centre around control strategies, and the repertoire of control strategies used seemed very limited, with only one city, Friedrichshafen, Germany, varying strategies by time of day and location. The author also raised the question as how one can best test alternative strategies.

The systems planning department of OC Transpo [23] reported the development and implementation of an APC system. Koffman and Gault [16] discussed possible ways of using the AVLC and APC data more effectively. Owing to their careful planning, a powerful data processing system has been developed to produce usable information from AVLC and APC "raw" data, which facilitate the usage of those data in this thesis work. Gault [8] summarized both the technical and management aspects of international experiences with AVLC. Furthermore, Gault [9] discussed the research directions related to AVLC systems. Future research includes the exploration of the possible application of artificial intelligence to enhance the service control function.

The Canadian Institute of Guided Ground Transportation [7] described the main features of the simulation model developed for OC Transpo, which is heavily used in this thesis.

1.2.3 Control Strategies

Levinson [20] reviewed the various control strategies that have been studied in the literature. There are: Schedule-Related strategies used to adjust the schedule to reflect the service and to adjust the service to the meet the schedule; service restoration strategies, whose purpose is to bring bus service back to normal after it has been disrupted; schedule control strategies which attempt to return buses to their original schedule; headway control strategies which attempt to obtain uniform headway; load control strategies whose purposes are to keep buses from getting

overcrowded and to reduce dwell times at busy stops; etc. These strategies can be adapted to improve the reliability of bus service, and to achieve uniform spacing.

Strategies suggested in the literature to improve bus reliability may be separated into two classes: planning control strategies and real-time control strategies (Turnquist and Blume [25]). Planning control strategies involve changes of a persistent nature to the bus route. Examples of such strategies include restructuring the routes and the schedules, making changes to the number and location of stops along a route, making provisions for exclusive rights of way, and arranging for traffic signal light preemption. In practice, schedule adjustment problems are important and require complex analysis; the feasibility of bus priority lanes depends upon many factors other than bus volume; furthermore, traffic light preemption may be difficult to obtain and has a very selective application (Levinson [20]). Further discussion of planning strategies exceeds the scope of this thesis.

Real-time control strategies are designed to act quickly to remedy specific problems as they occur: their effect is immediate, and rarely has any influence on the general nature of operations in the medium or long run. Examples of such strategies include adjusting with an extra bus, which consists in adding an extra bus on the route when a vehicle is significantly delayed or halted because of traffic congestion or a breakdown; skipping stops, used if no passenger wants to alight at a stop, in order to make up for lost time and try to reduce the possibility of bus bunching, with the understanding that another bus will follow shortly after to pick up any passenger waiting and that stop; short-turning, (for example, a vehicle may be asked to turn around before the end of the trip to catch up with the schedule); and vehicle-holding which entails holding selected vehicles at pre-specified control points (stops) along a route for purposes of schedule adherence or regularity.

In this thesis we focus mainly on real-time control strategies which may be divided in two groups: schedule adherence (or schedule control) strategies, and headway control strategies. The purpose of schedule adherence strategies is to maintain a service which is as close as possible to being "on-time", while the goal of headway control strategies is to maintain service regularity by "spacing" buses to obtain uniform headway. The effectiveness of these strategies on a given route will undoubtedly depend on the service frequency of that route, as passengers will have different expectations regarding the quality of the service provided. For low-frequency routes with

headway $h > 10$ minutes, schedule adherence strategies seem most appropriate. Passenger arrivals will typically occur very close to the bus scheduled departure time as passengers will want to minimize their wait time: passengers expect buses to arrive on time. The consequences of buses not being on time, especially if the bus is too early, may be significant for the passenger. On the other hand, for high-frequency service routes with headway $h < 4$ minutes, passengers typically are not concerned about schedule adherence, but will usually expect to see a bus every h minutes so that their wait time should be at most h minutes. Passenger arrivals will normally be random (and modelled as a Poisson process), and headway control strategies seem most appropriate for these routes.

In practice, schedule adherence strategies are easier to implement as the driver of the bus need not know or be concerned about the relative position of other buses along the route in order to adhere to his schedule. This is due to the static nature of a schedule. On the other hand, headway is dynamic, and, as the relative position of buses is of essence, some means of communication with the drivers must be available in order to implement headway control actions. The result is that in most cases, transit operators will either follow schedule control strategies, even for high-frequency routes, or implement a very simple headway control holding strategy known as the "threshold" policy, where a supervisor located at a control stop will monitor the headway and delay an arriving bus only if the headway between this bus and the previous one is less than the threshold value. This value is often set to be the planned headway. We have found only one real application of this policy in the literature, and the success of this experiment seemed limited: this could be explained by the fact that there was only one control point on the route, and that the reported effect of this policy may have been understated because street supervisors did not appear to have held all vehicles where such action was warranted (Abkowitz and Lepofsky [1]). This latter point could be explained by the fact that the underlying way of thinking and training of both supervisors and drivers was in terms of schedule adherence, and that deliberately delaying a bus past its scheduled departure time might have seemed counter-intuitive.

There exists other headway-based holding policies suggested by Jackson [13] and Turnquist and Bowman [26]. One is the so-called "prefol" policy: at each control stop, the departure time of an arriving bus is based on the headway with the previous bus and that with the following one. The headway with the previous bus is easy to obtain; in order to estimate

headway with the following bus, one must calculate the expected arrival time of the following bus at that stop, based on information regarding the current (approximate) position of that bus and a projection of the time needed to reach the control point from that position. This implies the use of technology such as an AVLC system. The departure time of the current bus is set so that the previous and following headway are equal, and the current bus is delayed accordingly, if needed.

The second policy, known as the "single-headway policy" is similar to the prefol policy except that the estimated arrival time of the following bus is based solely on statistical expectation and not on actual information about the current position of the bus on the route and its current speed. Simulation studies (Jackson [13], Turnquist and Bowman [26]) have shown the single-headway policy to be less reliable than the prefol policy, which however requires data which is more difficult and expensive to obtain in practice.

Based on this review of the literature, we conclude that few researchers have studied the effects of various control actions on bus performance, but that new technologies, such as AVLC and APC systems now motivate such an analysis to be made.

1.2.4 Applications of Artificial Intelligence / Expert Systems

Lévine and Poverol [21] indicate that, to solve the problem of dispatching the cars in a railway system, "there are two options - to produce a purely operational research optimization model or a knowledge based decision support system." Comparing these two options, the authors note that "the first approach has several limitations. First, modelling according to these traditional techniques raise many theoretical questions. Second, the length of computation time often prohibits on-line processing. Third, the expert's experience can't be used. The last but not least difficulty is the question of what function to optimize. The problem intrinsically involves multiple criteria. Dispatching involves various reasoning processes that combine three main criteria of equal importance - minimizing running costs, complying with schedule and satisfying customers." They built a knowledge based decision support system for a subset of the railroad network, and, based on this experience, conclude that "a knowledge-based system for a large, complex allocation problem" is feasible, and that their system "more closely mimics distributors' actual practice than a pure operational-research system".

Louis and Roswell made a synthesis of highway practice on "Knowledge Based Expert Systems in Transportation" [19]. In the synthesis, it indicates that "Advances in computing have dramatically changed the transportation field in the last twenty years. The next twenty years may see even more changes as computing technology continues to evolve. The widespread application of expert systems may be the key to these changes. Expert systems can capture knowledge currently residing in the transportation work force and make it available to others through knowledge based tutorials. They can automate mundane and repetitious tasks, and provide ready access to information in manuals and codes. Knowledge based expert systems (or Knowledge-Based Systems) can emulate expert colleagues to advise engineers in solving difficult problems, and even assist in designing facilities." Applications of knowledge based expert systems to transportation issues are discussed, and two current expert systems applications are described in detail.

Lamma and Mello [18] presented the basic design guidelines of a system able to assist a station-master in real-time planning and recovery of railway signalling systems. The system uses the knowledge-based system technology heavily. Some of the features presented in the paper had been implemented in a prototype way. Horellou C., Rossi C. and Sissa G. [10] reported a knowledge-based system for scheduling and controlling underground rail traffic, which is the implementation of a knowledge-based system most closely related to our research and approach. The system uses the configuration of the knowledge base describing the network of the underground railway to interpret the situation and then goes on to reproduce the expert's reasoning and provide one or more action plans to rectify the situation. The system is expected to be connected to the Rome underground installation in the future. Unfortunately, neither of the papers present test results to show the effectiveness of the systems or its potential problems.

From the literature overview, it appears that our research represents the first attempt at developing a knowledge-based system for the purpose of on-line bus control assistance.

2. Description of the Simulation Model

As the research contained in this thesis is heavily dependent upon the simulation model designed by CIGGT, we now provide a general presentation of the capabilities and limitations of the model.

The current version of the simulation model is designed to replicate the movement of buses on route 95. The simulation model is flexible enough to allow for any route to be simulated. Route 95, operated by OC Transpo in Ottawa, has been chosen to be simulated because it is the backbone of the transit network. Route 95 crosses the Ottawa region from East to West and back, going through the downtown area. It operates largely on OC Transpo's Rapid Transitway, which is a roadway dedicated to the exclusive use of OC Transpo buses. Six strategically located stops along the Transitway[1] are equipped with AVLC detectors (one in both direction) to monitor the movement of buses, see Figure 2.1. Most routes are designed to have at least one stop on the Transitway: only route 95 covers it entirely. Since almost all local routes connect at some point to the Transitway, route 95 links almost all local routes together.

Service on route 95 is from 4:00 am to 1:00 am the following day during the week. The service frequency of the 95 route varies with time of day: in the morning and afternoon peak hours, the route operates on a three-minute (3) headway. The rest of the day, the planned headway is five (5) minutes, except after 7:00 pm when the service drops to fifteen-minute (15) headway. A peculiarity of the route is that it runs two different patterns in each direction. The westbound patterns either start at Place d'Orléans (the easternmost stop of the route) or at Blair Station, and both end at Baseline Station, (the westernmost stop of the route). The two eastbound patterns start at Baseline: the first pattern ends at Blair, and the second goes all the way to Place d'Orléans. Every other bus runs the longer "Orléans" pattern, and the other half runs the shorter "Blair" pattern. This implies that the planned headway for stops between Blair and Place d'Orléans is always twice the planned headway for the rest of the route (for example, six (6) minutes for the peak hours). These patterns were also incorporated in the simulation.

---

[1] There are twenty-five stops in each direction along the Transitway.

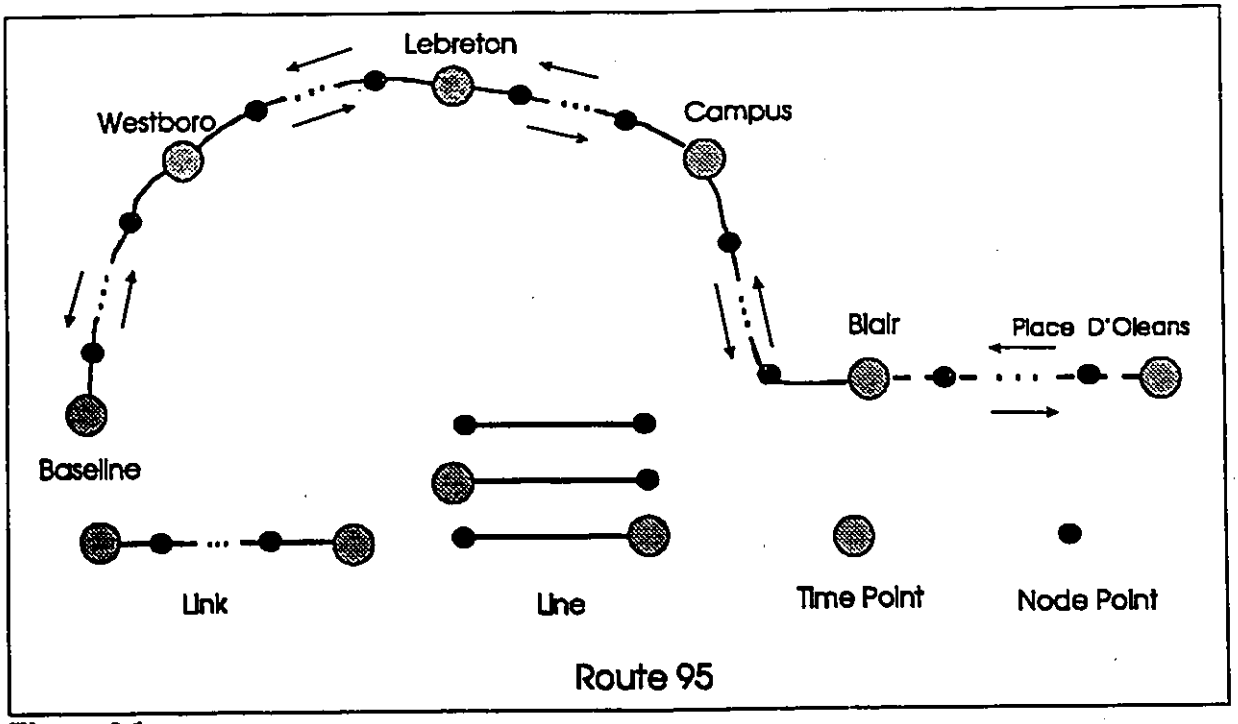


Figure 2.1

Before proceeding any further, it is important to introduce certain definitions:

- trip: the specified section of the route to be travelled in one direction by a single run of the bus. There are 4 types of trip (also called as trip patterns) for route 95: Baseline to Blair, Baseline to Place d'Orléans, Blair to Baseline and Place d'Orléans to Baseline.
- Time Point: a stop equipped with an AVLC detector to monitor the movement of buses. There are scheduled departure times at the time point for each bus for each trip.
- Link: is the portion of the route between two consecutive time points.
- Node: is any bus stop. The first node of each link must be a time point, and is called the start node; the last node of the link must also be a time point, and is called the end node.
- Line: is the portion of the route between two consecutive nodes. Therefore the link is composed of at least one line.

2.1 Capabilities and Requirements of the Simulation Model

2.1.1 Capabilities of the Model

The simulation model was designed to simulate the flow of buses and passengers' activities on a single route. It simulates the movement of buses to reflect the on-street conditions and environment within which buses operate. An important feature of this model is that it simulates a real bus route based on real empirical data. The model uses information collected by buses equipped with an Automatic Passenger Counting (APC) system. As discussed earlier, the APC equipped buses are capable of collecting data on the boarding and alighting activities (how many people get on and off) at stops along any given route. Data collected by the APC system may also be used to determine trip start time, arrival and departure times at stops, the total time spent at each stop, which includes the time spent boarding and alighting passengers, and dwell time, (which is the time during which the vehicle is stopped but no boarding or alighting activity is taking place), and travel times and distances between stops.

The model also uses information contained in OC Transpo's AVLC system. Such information includes the actual scheduled start time of every trip for every day of the week. The scheduled departure times are also available for the time points. The AVLC system provides information about the bus assignment to specific "blocks" which are collections of trips that must be performed by the vehicle. Information is also available on the probability of breakdown of buses and on traffic congestion in the various sectors of the route by time of day. All of this information has been used to simulate the movement of buses along the route, the arrival of passengers at stops, the boarding and alighting activities, etc., in order to replicate as closely as possible the real operation of the route.

The simulation model is flexible: provided the data is available, it can be used to simulate any route operated by the transit property (At present, only the data related to the route 95 has been linked to the model). A future version of the model is being designed to be connected to OC Transpo's AVLC system, so that any change in schedule, for example, can immediately be integrated in the simulation model. The current version does not yet have this capability, but can be run on a stand-alone mode. The user interface was designed to replicate the AVLC screens that the controller would actually use in controlling the real system. This is so as to meet OC Transpo's first objective that the model be used for the training of controllers.

A second no less important use for this simulation model, is to provide a means of testing rapidly the effects of various planning and real-time control strategies, which were discussed in the first chapter.

The simulation has been designed to run buses according to one of three distinct real-time holding control policies. The user may select the "No Control" (NC) scenario, the "Schedule Control" (SC) scenario, or the "Headway Control" (HC) scenario. Under the NC and the SC scenarios, the trip departure time of a bus is based on the schedule provided by the AVLC data. If the bus is ready for departure prior to the scheduled departure time, it will leave exactly on schedule. Otherwise, it will leave as soon as it is ready (i.e. buses never start a trip early). For the other stops, no control action is taken under the NC scenario: each bus leaves the stop as soon as the boarding/alighting activities are done. For the SC scenario, the user may select any combination of stops as control points. At these control points, the bus will either leave the stop on time, if it is ready to leave before or at the scheduled departure time, or it will leave as soon as possible if it is late. No control action is taken at non-control stops. If no stop is selected as control point, the SC scenario is equivalent to the NC scenario. Finally, under the HC scenario, the user may again select any combination of stops as control points. At the chosen control points, bus departure time is based on headway using a threshold policy. When a bus is ready to leave a control stop, the system checks the departure time of the previous bus. The threshold value t is often set to be equal to the planned headway, h, in seconds. If the previous bus has left the stop less than t seconds prior to the current time, the bus is held long enough for the actual headway between these two buses to be h at that stop. If the previous bus left more than t seconds before, no delay action is taken.

The states (queue length) and flows (boarding and departure) of passengers at each of the stops on the route can be simulated. The states (loading, malfunctions) and flows of buses along with the effects of the conditions of route segments (congestion, closure) on bus transit times can also be simulated.

There are two modes of operation of the simulation. They are:

- Simulation from the beginning of the day (No bus starts its scheduled runs prior to the start of the simulation run).

• Simulation from the middle of a day. In this mode the user selects a simulation start time. Any bus which starts its trips after that start time will be simulated. Any bus which ends its trips before that start time will not be simulated. Buses starting their trips before that start time and completing the trips during the simulation are called the *phantom buses* and are simulated as well. Statistics are collected for the phantom buses but it is assumed that the phantom buses run exactly on schedule.

Both simulation modes accept mid-simulation interruptions by the user, thus allowing him/her to change parameters or to take control actions (rescheduling the buses, cancelling trips, adding an extra bus, etc.). The simulation resumes with the updated information (if any) whenever the user is ready.

There are two types of statistical output provided by the simulation model: run-time screen displays and the final output reports.

Run Time Screen Displays

The typical run time screen display is shown in Figure 2.1.1. The **scenario** reflects the control environment in which the simulation is running. There are three basic scenarios, namely, the NC scenario, the SC scenario, and the HC scenario described previously.

The **simulation time** is updated in one minute simulation time intervals according to the simulation clock rate. The **clock rate** displays the current multiple of real time at which the simulation is running. This **clock rate** ranges from 1 to 20, or may be set to 99 which indicates a batch run. In batch mode all events will be processed in time sequence with no real time elapsing except for program execution time.

The **incident section** displays a rolling log of the five most recent incidents(eg. a link delay) with the start time and a brief (25 characters) description.

The **radio message area** is for reports of equipment failures which normally come to the controller's attention via voice radio.

The **stop** performance area shows the queue lengths at bus arrival time and the passenger wait time averaged over the previous quarter hour interval. The left side presents the worst, second worst and average of the worst 50% of stops on the route in relation to wait time or queue length as specified by the user. The right side presents the wait and queue states for

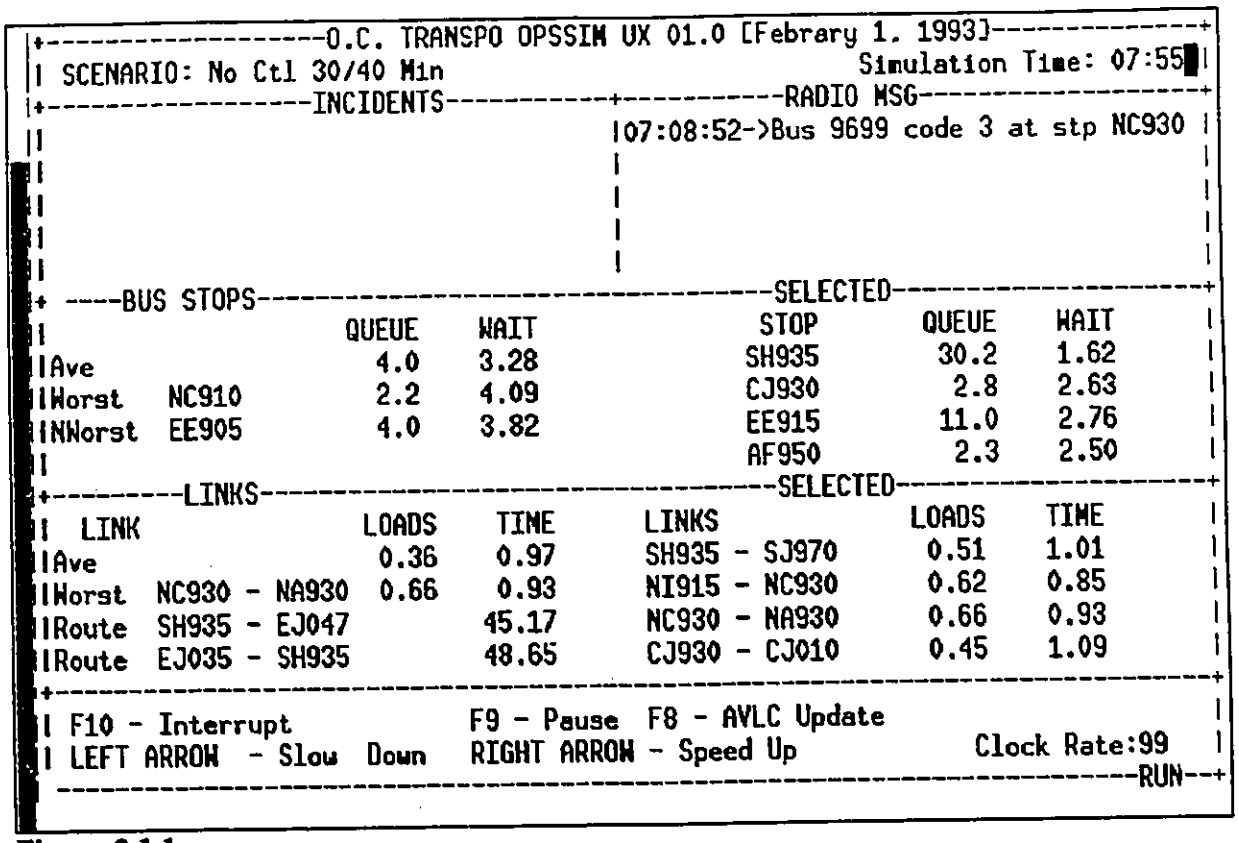


Figure 2.1.1

preselected stops of interest as specified by the user.

Similarly, the **link** section presents loadings (% capacity of the bus) and transit times (% of scheduled transit time) for the worst (has the maximum load) link and the total transit times (in minutes) for the route in both directions (left side), and loadings (% capacity of the bus) and the transit times (% of scheduled transit time) for selected links (right side).

The soft keys are displayed to show which keys are recognized during the simulation. For an example, if F10 is pressed, the simulation clock and current state are frozen and the input menu screen is displayed for the user. The LEFT ARROW key slows down the clock rate and the RIGHT ARROW key speeds it up.

The screen running display is refreshed every 15 minutes of the simulation time, but the messages in the "incidents" section and the "radio msg" section is displayed at the time of occurrence.

Output Reports

The simulation produces a standard report to a file at the end of the simulation time. The standard report includes a log of the same information displayed on the screen during the run. In addition there are summary performance tables included in the standard report file. These reports include:

- Route Transit Performance, which provides the average total travel times in both directions;
- Bus Schedule Performance, which displays the proportion of buses which left early, on time or late from each stop;
- Stop Headway Performance, which gives the average headway and the wait factor at each stop along with the minimum, the median and the maximum headway, and the 10th and 90th percentile of headway. Stop Queue Length, which shows the average, minimum and maximum queue length at each stop along with the total number of the passengers who arrived.
- Stop Load on Departure, which provides the minimum, average and maximum bus load on departure at each stop.

For a more detailed description of those reports, refer to [7].

2.1.2 Requirements of the Model

Input State Parameters

In order to simulate the operation of a route, a number of "input state parameters"[2] must be entered to the simulation model to describe each of the main influencing factors. The major state variables used by the simulation model, which fully describe the simulated operating environment, are as follows:

1) Bus State
2) Link State
3) Stop State (for the applicable bus route number)
4) Flow State
5) Regulation State
6) Detector/communication State
7) Bus Trip Patterns
8) Block Assignments
9) Trip Assignments

---

[2] This terminology was introduced in [7]. We use it throughout this thesis for consistency.

Requirements from 1) through 5) were obtained from the APC data, and the associated parameters can be modified by the user via the user interface module of the simulation model. The last four requirements were derived from the AVLC data. For detailed description of those state variables and the associated parameters, refer to [7].

Running Environment

In order to compare and evaluate different control actions tested on the simulation model, a multi-tasking environment must be provided to allow the interruption of the simulation, the taking of a control action, and the resuming of the simulation run. More than 4 Megabytes of working memory is needed to run the simulation. For example, we have installed the simulation model on a Sun work station at the University of Ottawa. The operating system of the work station is UNIX which provides a multi-tasking environment and dozens of gigabytes of working memory.

2.2 Detailed Description of the Simulation Model

2.2.1 Structure of the Program

Data Structures

Dynamic double linked lists are heavily used to store the data for running the simulation and collecting the results. These fall into the following categories:

1) Lists for storing AVLC data, such as sRoute, sBlock, etc.

2) Lists for storing scenario data which drives the simulation, such as sRoutePoints, sRunInfo, sRegulation, sBusType, sStop, etc.

3) List for storing the event table, sNextEvent.

4) List for storing statistical data, sStats.

5) Lists for storing menu screen display messages, sScreenMenu.

Most of the lists are linked together by means of pointers between fields. A record in a list may be able to access the data from other related lists. For example, the data record associated with a single bus may access the data from the lists of Stops, Blocks, Trips, etc., as shown in Figure 2.2.1.

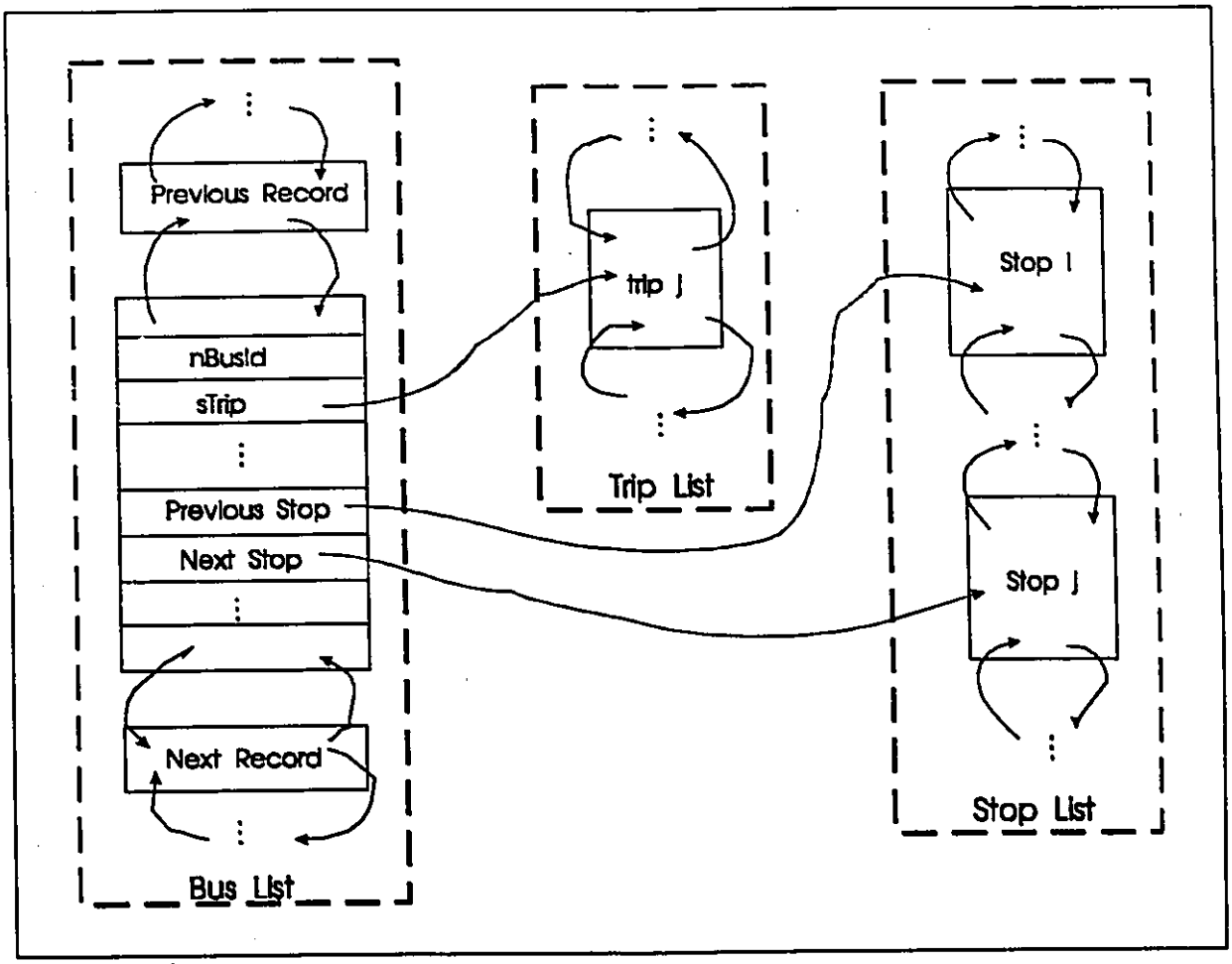


Figure 2.2.1

Logic Control of the Program

1) A scenario to be simulated must be selected, and the appropriate data files such as "R95_DAY*.DAT", ... (where "*" is either N, S or H for the NC, SC and HC scenario) will be loaded into memory to generate the scenario link lists.

2) The model waits for the user to modify the parameters associated with the selected scenario.

3) The user starts the running of the simulation

- AVLC data files such as "block.dat", "trip.dat", ... are loaded to generate the AVLC link lists.

- If it is a middle day run, the system sets up a phantom bus list and runs the phantom buses until the simulation start time.
- The bus entry events, which are associated with the buses with scheduled start times greater than the simulation start time are added into the simulation event table.
- Other events are then added to the event table, such as End Simulation event, Link Delay event, Bus Break Down event, Clock event, etc.
- The running of the simulation is started.

The order of events in the event table is sorted according to the event time and the priority of the event. Events are sorted based on earliest event time first. If there is a tie, the event with highest priority gets processed first. A list of events ranked from highest to lowest priority is given below.

Clock event

Bus event

- bus entry
- bus arrival
- bus waiting
- bus unloading
- bus loading
- bus dwelling
- bus adherence to the regulation rules
- bus departing
- bus exit

Bus break down event

Extraordinary Link delay event

- delay start
- delay end
- delay message to display

Statistic data output event

End simulation event

4) Running of the simulation

The system constantly checks if a command key has been pressed. If so, the appropriate command will be executed; otherwise the system always processes the next event in the event table. The command keys and the respective commands are as follows:

- F10, interrupt the running of the simulation and return to the main menu;
- F9, interrupt the running of the simulation and let the user check the current screen display of the simulation intermediate results;
- F8, interrupt the running of the simulation and let the user update the AVLC data files resume the running of the simulation;
- Cursor_Right key (or key "+"), speed up the simulation clock rate;
- Cursor_Left key (or key "-"), slow down the simulation clock rate.

When the F8 key is pressed, the simulation time will be "frozen" until the user resumes running of the simulation. When it resumes, the AVLC link lists and the simulation event table will be updated to reflect the updated environment in which the events are simulated. The logic relations are shown in Figure 2.2.2.

2.2.2 Passengers' Activities

Passenger Arrival Rate

Passenger arrival rates at bus stops are derived as follows. The average passenger arrival rates (number of passengers per minute) for each stop at discrete time points which cover the time period starting from 5:00 AM to 3:00 AM the following day have been derived by the model developer based on the APC data from OC Transpo for weekdays and weekends. These actual rates R_1, R_2, ..., R_n are only specified for certain points in time T_1, T_2, ..., T_n. For any point in time t, such that $T_i < t < T_{i+1}$, the arrival rate at time t is approximated by fitting a S-shaped curve which links the specified arrival rates R_i and R_{i+1} for times T_i and T_{i+1} respectively, as shown in Figure 2.2.3. The following equation is defined:

$$f_m(t) = C_1 + e^{[C_2 - \frac{C_3(t-T_i)}{T_{i+1}-T_i}]} \qquad (2.1)$$

where the parameters of the function C_1, C_2 and C_3 are estimated with the APC data as 1.0, 4.0 and 8.0 respectively, and $T_i \le t \le T_{i+1}$.

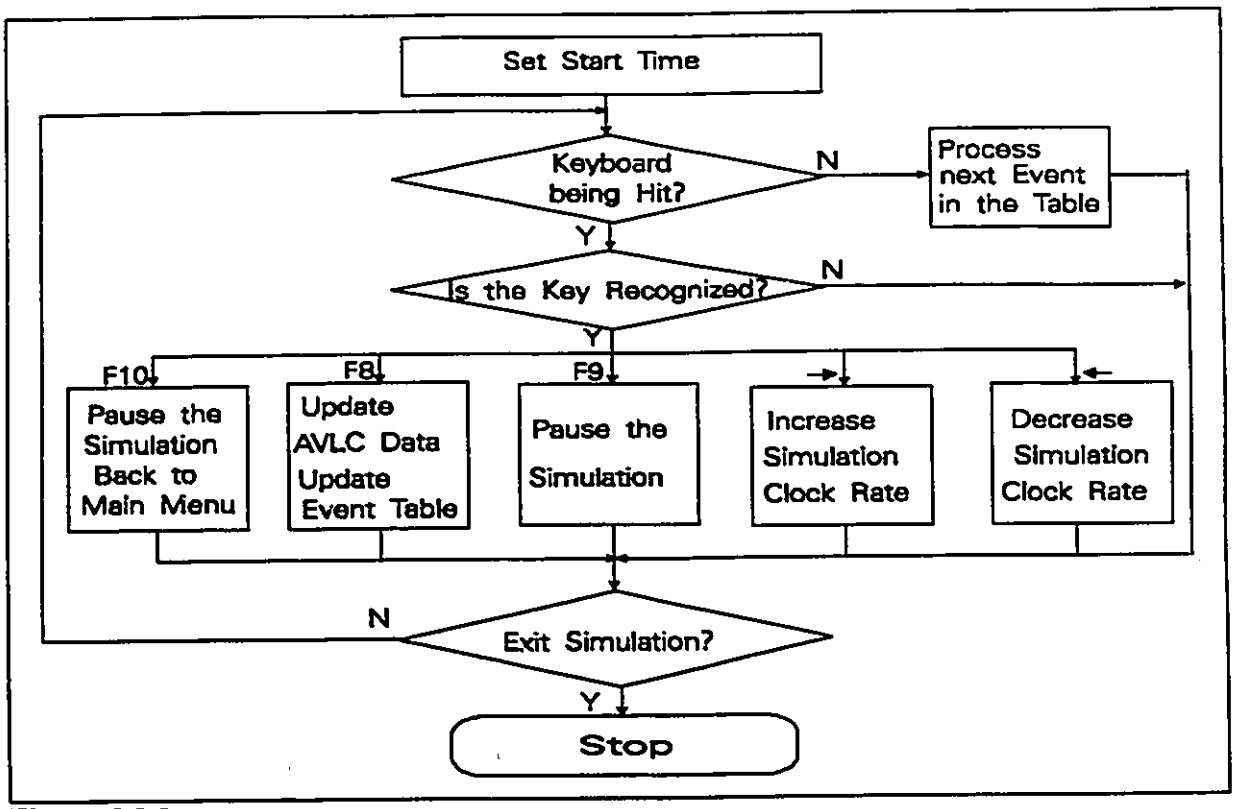


Figure 2.2.2

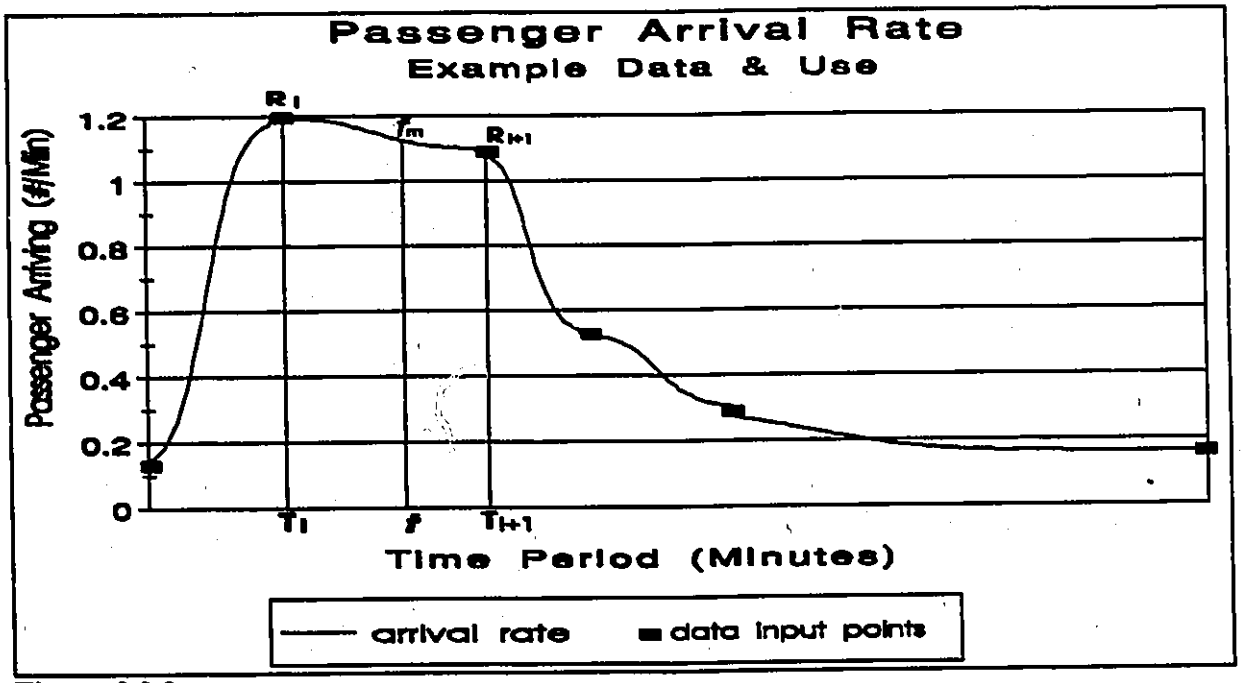


Figure 2.2.3

Next R_i and R_{i+1} (the discrete passenger arrival rate associated with the time T_1 and T_2 respectively) are adjusted by multiplying the "multiplier" set by the user to give $\overline{R}_i$, $\overline{R}_{i+1}$. The default value of the modifier is 1.0 and can be changed by the user for the stops selected by the user for the purpose of increasing or decreasing the passenger arrival rate at those stops. In Chapter 3 where we did validation tests on the model, we changed the passenger arrival rate by setting the modifier to a different value. The estimated arrival rate at time t $\overline{f}_m(t)$ is an S-shaped interpolation between the two data points $\overline{R}_i$ and $\overline{R}_{i+1}$ as given by

$$\overline{f}_m(t) = \overline{R}_i + \frac{(\overline{R}_{i+1} - \overline{R}_i)}{f_m(t)} \qquad (2.2)$$

A standard deviation, σ, is derived to take into account daily variability and variations specific to the time period $[T_i, T_{i+1}]$. The final passenger arrival rate at time t for the stop is equal to ($\overline{f}_m$ + N_σ), in which N_σ is a random number drawn from a normal distribution with mean zero and standard deviation σ.

Number of Passengers Getting off at Each Stop

Boarding passengers will be assigned a destination on the basis of APC stop unloading distributions for the relevant time of day. Stops are divided into the *specified* stops and the *unspecified* stops. Suppose there are *N* stops, with *K specified* stops and *L unspecified* stops. Assuming that *p* percent of passengers go to *specified* stops and (1-*p*) percent of passengers go to *unspecified* stops.

The proportion of on-board passengers to alight at the stop will be determined as follows:

• For each specified stop *i*, there is a fixed unloading percentage for the various times of the day (AM peak, PM peak and day time off peak hours), denoted by

$$P_{AM}(i),\ P_{PM}(i),\ P_{OFF}(i), \qquad 1 \le i \le K$$

• For each unspecified stop j, suppose the remaining number of unspecified stops from j to the end of the trip is T, the sum of the alighting percentages at the remaining specified stops is P_R, and the sum of the alighting percentages at the specified stops of the whole route is P_S, then the proportion of the on-board passengers to alight, denoted by $P_{unspecified}$, is given by

$$P_{unspecified} = \frac{1.0}{T^{0.8}} \left[1.0 - p \frac{P_R}{P_S} \right] \qquad (2.3)$$

where

$$P_R = \sum_{\substack{i>j \\ i \in Specified}} P_*(i), \quad and \quad P_S = \sum_{i \in Specified} P_*(i)$$

in which * = AM, PM or OFF depending the time of day.

The number of passengers to alight at the stop is determined by the proportion derived above times the current load of the bus.

Number of Passengers Getting on at Each Stop

The number of passengers boarding at a given stop is determined by multiplying the passenger arrival rate, which is described above, by the total time spent between the departure of the previous bus and the departure of the current bus. This time includes the time between the departure of the previous bus and the arrival of the current bus, the time needed for passengers to board the bus, and if any, the time spent waiting for the proper departure time as given by the schedule adherence or headway control scenario.

The total queue length is the total number of passengers waiting to board the bus. If that length is less than or equal to the difference between the maximum capacity of the bus and the current load of the bus, then all the passengers get on the bus. Otherwise only a number of passengers equal to the difference between the capacity of the bus and the current load of the bus will get on the bus, leaving the rest of the queue to wait for the next bus.

Passenger Boarding Time

Boarding times will include an initial dead time and a per-passenger boarding time (boarding rate) for front door users. The percentage of the passengers using the front door , the

initial dead time, and the mean and variance of the boarding rate may be defined by the user for peak and off-peak time periods. Default values are derived from the APC data, which can be changed by the user.

Passenger Alighting Time

Passenger alighting times may be defined in a similar way to passenger boarding times, with the exception that the rear door usage pattern (% of alighting passengers using the door) will be considered instead of the front door usage pattern. The underlying assumption is that most passengers board using the front door, while most exit through the rear door.

Stop Times

Stop times will include time allocations for alighting, loading and excess idle time. The excess idle time will be determined based on schedule/headway targets. There will be no excess idle time for the NC scenario: the bus departs the stop when it finishes loading and unloading the passengers. For the SC scenario, if the time at which the bus finishes the boarding and alighting of the passengers at the stop is greater than or equal to the scheduled departure time, the bus leaves the stop right away, otherwise it will stay idle at the stop until the scheduled departure time. For the HC scenario, if the difference between the previous bus departure time and the current time when the bus finishes loading and unloading the passengers is equal to or greater than the planned headway, the bus leaves the stop right away, otherwise it remains idle at the stop until the actual headway equals the planned headway.

Transit Time across a Line

The calculation of the transit time across a line will take the following things into account:

- Daily road conditions,
- Extraordinary Delays,
- Bypass a Stop,
- Running a phantom bus.

Given the scheduled link transit times, a linear interpolation is used to determine the average transit times of each line of the link. As Traffic conditions may vary in the real system, the model computes a congestion delay factor for each line, except each line immediately following a time point. The mean congestion delay factor and its standard deviation (either σ_1

or σ_2, depending on the condition as the following) may be adjusted by the user. The default values are derived from the APC data for different time periods. The actual congestion delay is simulated by drawing a random number from a uniform distribution between 0 and 1. If the random number is less than or equal to 0.5, another random number will be drawn from a normal distribution with mean 0 and standard deviation σ_1. The absolute value of this random number is subtracted from the mean congestion delay factor. Otherwise, the second random number to be generated will be drawn from a normal distribution with mean 0 and standard deviation σ_2. In this case, the absolute value of this random number will be added to the mean congestion delay factor. This procedure was designed to simulate a 50-50 chance of congestion being lighter or heavier than average. The average line transit time will be multiplied by this mean congestion delay factor to simulate the effect of the traffic congestion on the transit time. The congestion delays are not applied to the lines immediately following the time points.

If the bus is late, the mean congestion factor will be reduced by a certain percentage (20%), in an attempt to simulate the normal speeding up done by drivers when they realize they are behind schedule.

Extraordinary delays are not generated randomly by the system. To simulate extraordinary delays, the mean delay time and the associated standard deviation as well as the delay start time, delay end time and the link(s) where the delay will happen should be defined by the user before the running of the simulation. If the actual simulation time is between the delay start and end times for such a link, the average transit time will be increased by the mean delay time plus a random number generated from a normal distribution with mean zero and the user defined standard deviation. A construction delay, for example, could be so modeled.

If the bus bypasses a stop, the current link will be joined to the next link to keep track of the total distance, various transit times and detectors crossed. The average transit time will be reduced by the time saved by not stopping, which is one of the bus regulation parameters which can be changed by the user.

For phantom buses, the actual transit time is equal to the scheduled one, i.e., it is assumed that all the buses running before the simulation start time are on schedule.

2.2.3 Movement of Buses

The simulated movement of buses along the route is done on a point to point basis: the model does not perform a continuous simulation of each bus. The point to point times and congestion delays will be based on the data available from the APC system. The movements of buses includes the following activities:

- Bus Entry into the System,
- Bus Exiting from the System,
- Bus Arrival at and Departure from a Stop,
- Bus Incident Initiated.

Bus Entry into the System

Buses which have a scheduled start time greater than the simulation start time will be put into the event table under the event "Bus Entry".

Bus Exiting the System

In either one of the following two cases a bus will exit the system:

1) the bus finishes all the scheduled trips and has no more trip to run,

2) the bus breaks down at a stop.

Bus Arrival at and Departure from a Bus Stop

There are five major events associated with the stopping of each bus at each stop: Bus Arrival at the stop, Bus Unloading the Passengers at the Stop, Bus Loading the Passengers at the Stop, Bus Dwelling at the Stop (staying excess time at the stop due to certain bus holding policy), and Bus Departure from the Stop. The major activities associated with each of those five events are shown in Figure 2.2.4 to Figure 2.2.8.

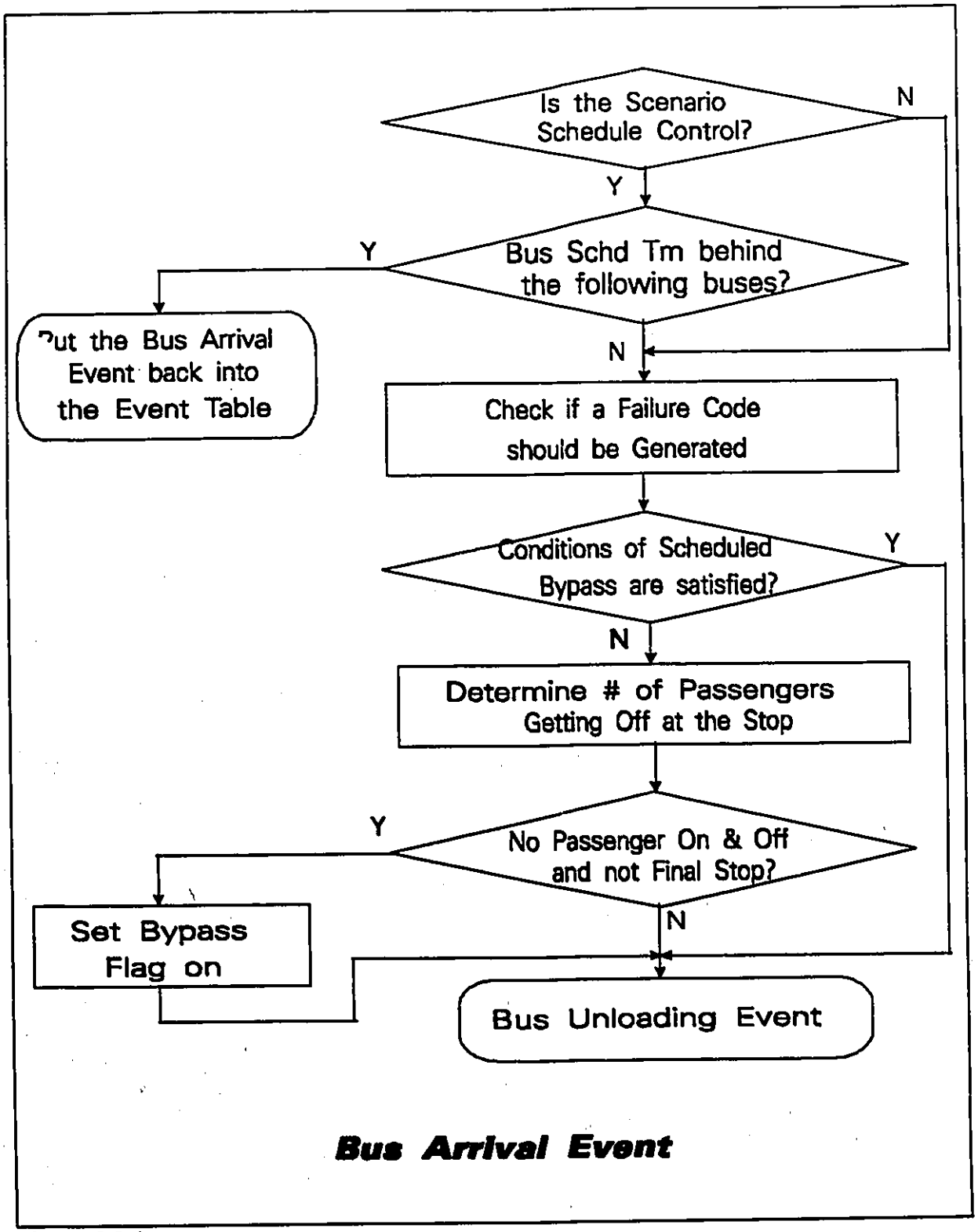

Is the Scenario Schedule Control?
N
Y
Bus Schd Tm behind the following buses?
Y
Put the Bus Arrival Event back into the Event Table
N
Check if a Failure Code should be Generated
Conditions of Scheduled Bypass are satisfied?
Y
N
Determine # of Passengers Getting Off at the Stop
No Passenger On & Off and not Final Stop?
Y
N
Set Bypass Flag on
Bus Unloading Event
Bus Arrival Event

Figure 2.2.4

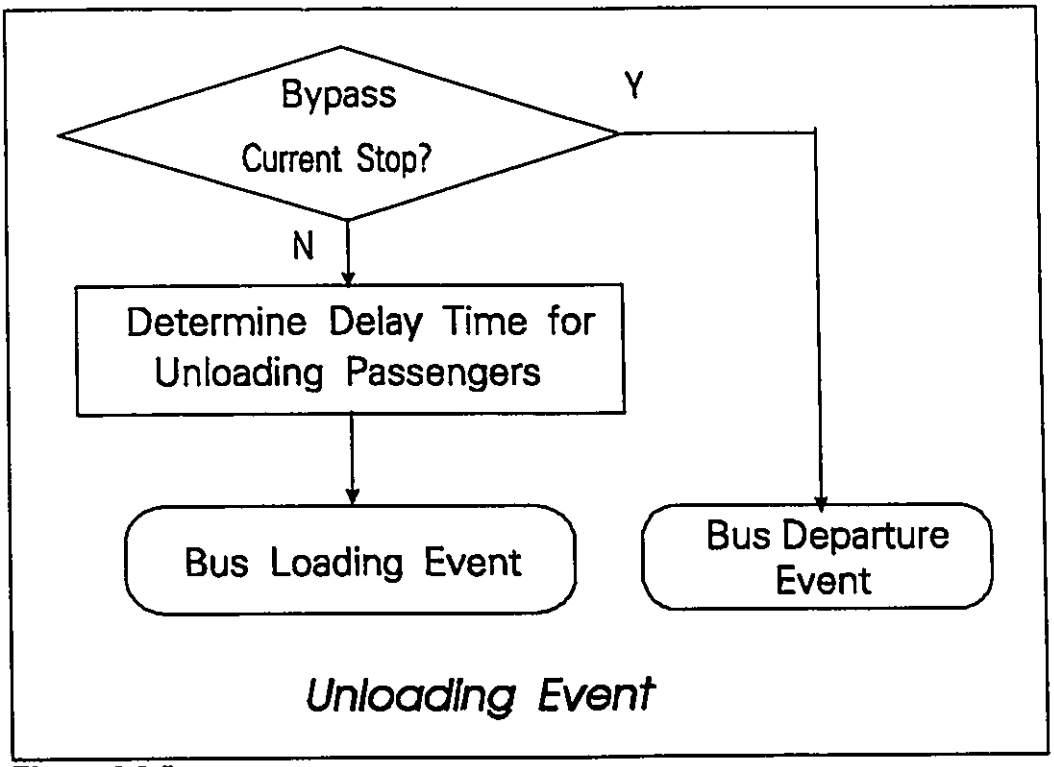

Bypass
Current Stop?
Y
N
Determine Delay Time for Unloading Passengers
Bus Loading Event
Bus Departure Event
Unloading Event

Figure 2.2.5

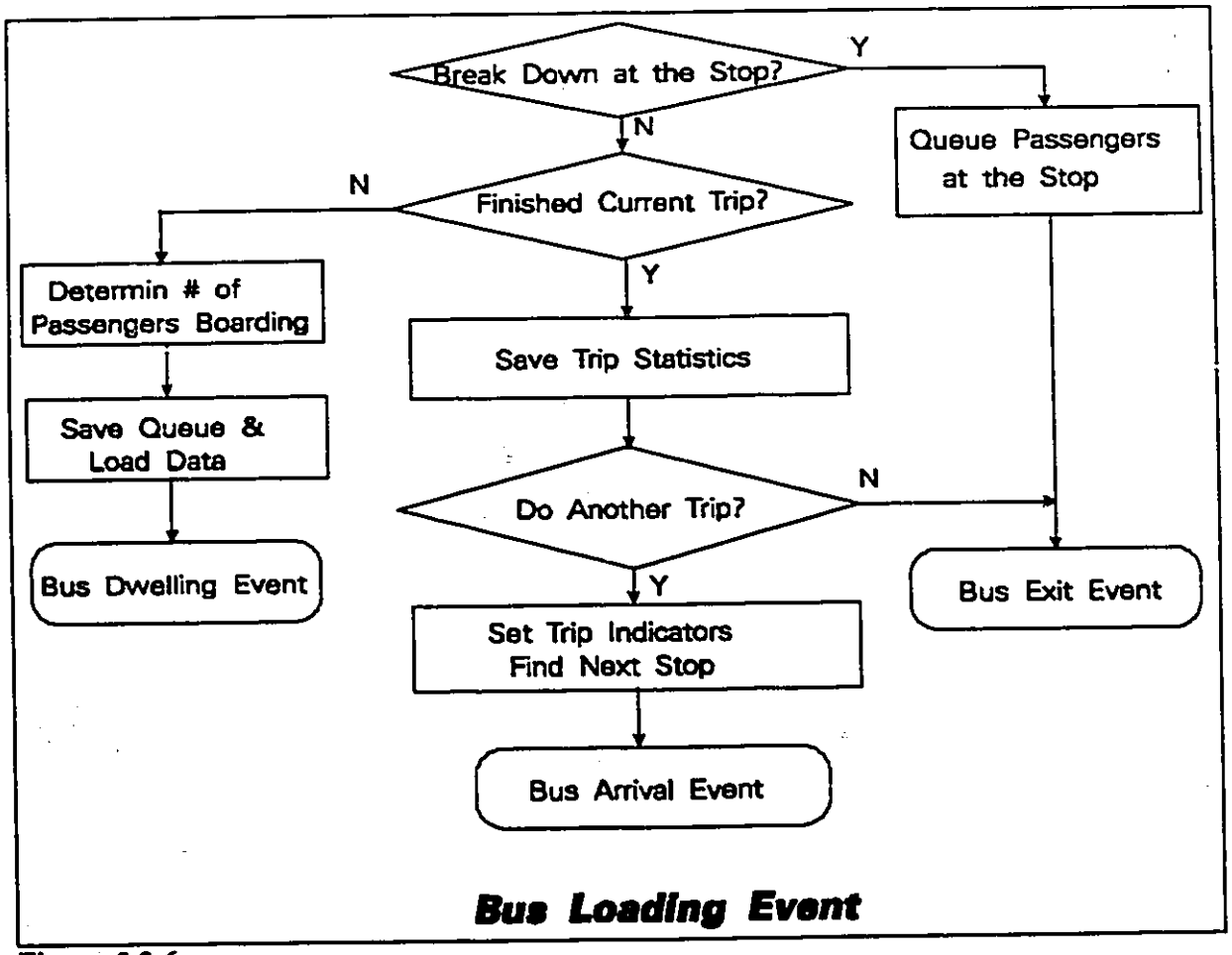

Break Down at the Stop?
Y
N
Queue Passengers at the Stop
N
Finished Current Trip?
Y
Determin # of Passengers Boarding
Save Trip Statistics
Save Queue & Load Data
Do Another Trip?
N
Bus Dwelling Event
Y
Bus Exit Event
Set Trip Indicators Find Next Stop
Bus Arrival Event
Bus Loading Event

Figure 2.2.6

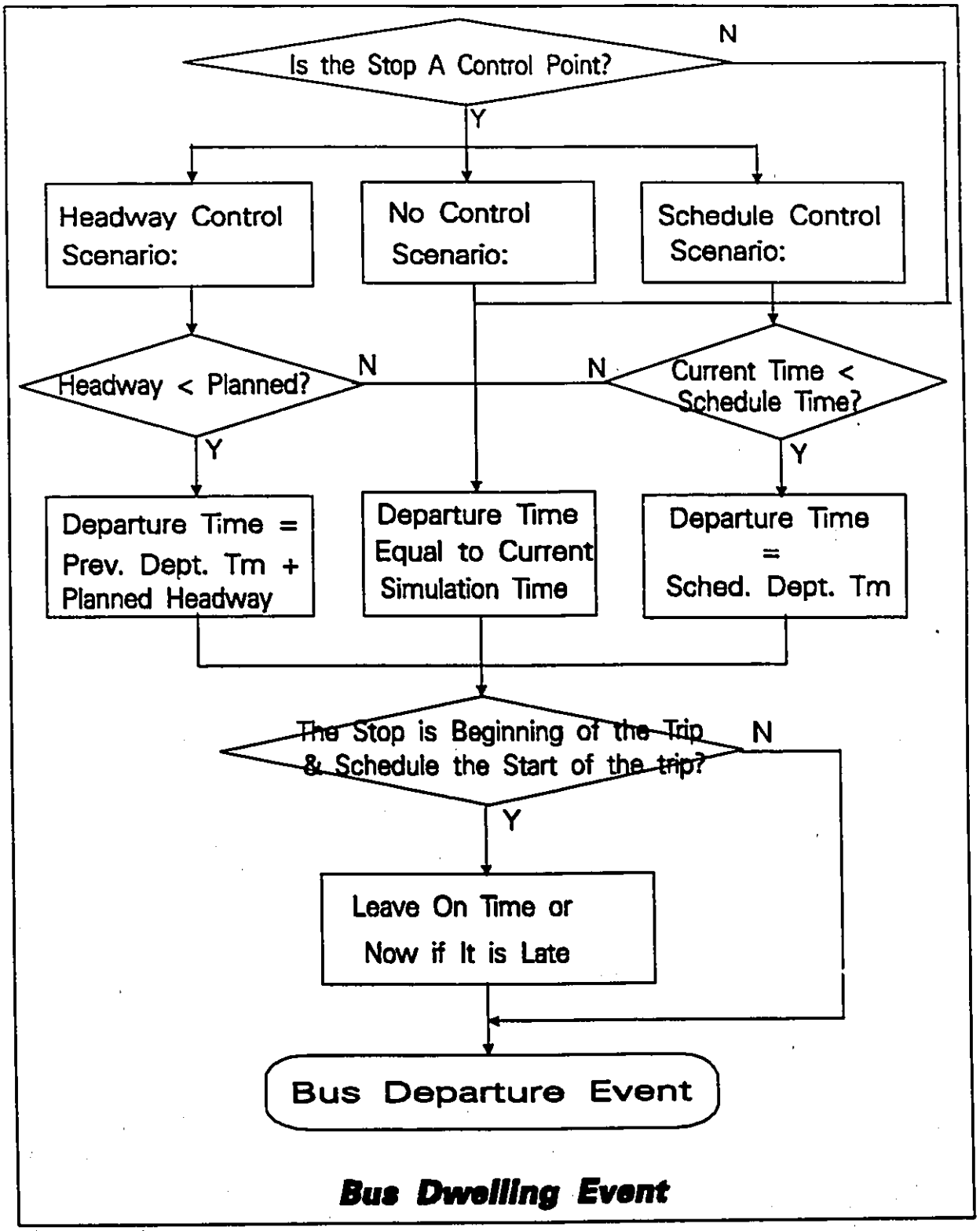

Is the Stop A Control Point?
N
Y
Headway Control Scenario:
No Control Scenario:
Schedule Control Scenario:
Headway < Planned?
N
Y
N
Current Time < Schedule Time?
Y
Departure Time = Prev. Dept. Tm + Planned Headway
Departure Time Equal to Current Simulation Time
Departure Time = Sched. Dept. Tm
The Stop is Beginning of the Trip & Schedule the Start of the trip?
N
Y
Leave On Time or Now if It is Late
Bus Departure Event
Bus Dwelling Event

Figure 2.2.7

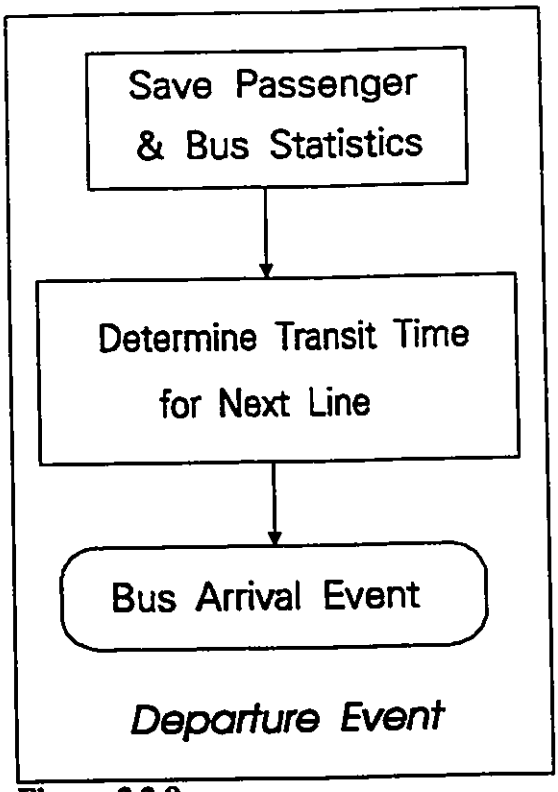


Figure 2.2.8

Bus Incident Initiation

There are two types of failure code associated with a certain type of equipment failure of the bus:

- code 1, 2, 5 and 39 which will have no effect on the simulation of the bus,
- code 3 which will lead to an inoperative state, accompanied by unloading of passengers to join the queue at the stop and exit of the bus from the system.

One of the failure codes will be generated as following:

i) Generate a random number from a uniform distribution between 0 and 1

ii) Define

$$fCalc = \frac{(S_t - L_t) \times P_b}{T_f} \qquad (2.4)$$

where S_t is the current simulation time, L_t is the last time when the bus had a failure, P_b is the probability of having a break down (for a given type of bus) T_f is the probability time frame, which is a constant and equal to 6 hours. Given a breakdown occurs, let P_i represent the

probability that the breakdown is associated to code i, then $\sum P_i = 1$. The P_b and P_i may be changed by the user.

iii) Check if the bus is a phantom bus. If it is, no failure code will be generated. If it is not, and "fCalc" is less than or equal to the random number generated, no failure code will be generated. If it is not a phantom bus and "fCalc" is greater than the random number, then a failure code will be generated. In this case , a second random number is drawn from the same distribution to determine which failure code will be generated. If the second random number is less than or equal to P_1, the probability associated code 1, then code 1 will be generated. If the random number is less than or equal to $P_1 + P_2$ but greater than P_1, then code 2 will be generated, and so on. If the failure is code 3, then the bus breakdown flag will be set, which means the bus will breakdown at the current stop.

There is a major short coming of this method of generating bus break downs. The method is sensitive to the control actions, since the generation of the failure code depends on the time at which the random number is generated. The simulation uses a single random seed for each simulation run so that only one string of random number is used for all simulation events. If control actions are taken, different additional dwelling times are added to the running buses and the order of the event table are changed (compared to the simulation run with no control action). Therefore, a given random number will not necessarily be used for the same purpose in the two runs. In particular, the random number which will be compared to "fCalc" to determine whether or not a second random number should be generated to determine the nature of the failure code may not be the same in the "control action" run as in the "no action" run. Therefore, more or less bus breakdowns may occur in the "control action" run than in the "no control" run. A change in control policy should have no effect on the probability of bus breakdown: the use of a dedicated random number string associated with bus breakdowns should be considered. Further modifications of the program to insure that bus breakdown is independent of the control action is desirable and recommended but it may be difficult to achieve. In order to go around this difficulty, we set the probability of code 3 to zero when we tested the different control actions.

2.3 Porting the Simulation Program to the Work Station

The original simulation model was written in VAX C and developed on a VAX machine. Some functions of the program were machine dependent, especially the functions dealing with user interface (screen displays, keyboard controls, reading and writing the data files, etc.). Given the unavailability of a compatible VAX machine at the University of Ottawa and the delayed installation of the simulation program at OC Transpo, our first task was to port the program to another platform. There were two options available for us: either porting the program to a Personal Computer such as a IBM PC/486 running MS/DOS, or porting the program to a work station running UNIX. Given the simpler structures and a mono-user operating system of MS/DOS, it is easier to manage the screen display, to get the control of a keyboard and to manage the files than on a work station, making the porting of the user interface functions of the program easier. Porting to PC allowed us to get familiar with the feature of the simulation program and to make some preliminary tests to identify any potential problem with the model more quickly. We tried porting the program to a IBM PC/486, but found that the working memory of the system was too limited for the given size of the simulation program. Furthermore, as the DOS system is a mono-tasking system, we realized that it would not be able to run the simulation and to change the AVLC data files (to simulate a control action) at the same time. Therefore, the only option left for us was to port the program to a work station running UNIX. We thus ported the program to the Sun work station "ELCSparc Station" running UNIX V. The porting activity involved two full weeks of very technical work. For reason of readability, we differ the discussion of the porting of the program to Appendix A.

3. Preliminary Validation of the Simulation Model

The simulation model has been designed to perform several tasks described in the first chapter: of particularly interest for this thesis, is the testing of various on-line control strategies. Before using the model to perform hypothesis testing and to draw conclusions on the operations of the real system, it is important to perform a certain level of model validation in order to eliminate any system malfunction which may exist, and which could bias the results, and also to provide some degree of confidence in the repeated outcomes of the simulation runs. Before extensive validation is preformed, it is important to note the following:

- "A simulation model of a complex, real world system is always only an *approximation* to the actual system, regardless of how much effort is put into developing the model. Thus, one should not speak of the absolute validity or invalidity of a model, but rather of the degree to which the model agrees with the system. The more time (and hence money) that is spent on validation, the closer the agreement of the model with the real system should be" [17].

- Simulation models are generally better at comparing alternatives than at determining absolute answers. The model must be sufficiently valid to allow comparison, in a relative sense, of different control strategies, even though it may not be possible to calibrate it accurately enough to predict precisely the actual bus performance measures under the various control policies. The trends observed from the simulation results should be valid even if the actual numbers may not.

- A simulation model should be validated relative to a specified set of criteria, namely, the criteria that will actually be used for decision making, such as bus headway performance and schedule performance in our case.

- The use of formal statistical procedures is only one part of the validation process; at present most of the "validation" done in practice seems to be subjective. One reason for this is that most classical statistical techniques can not be directly applied in the context of simulation model validation [17].

- The main focus of this thesis is not to perform a thorough validation and calibration of the simulation model, but rather a limited validation to obtain a certain degree of confidence in the model so that a valid comparison between various control strategies can be performed using

the model. Furthermore the validation should be sufficient to allow a prototype of a knowledge-based bus control system to be tested using the model.

Keeping all of these factors in mind and given the time constraints for this research, limited validation of the model has been performed, and reported in this chapter.

There is a so called "three-step approach" for validating a simulation model, widely used in practice [17]: Face Validating, Assumption Validating and Validating of Input-Output Transformations. The first step of validation is to develop a model with high *face validity*, i.e. a model which, on the surface, seems reasonable to people who are *knowledgable* about the real system under study. The goal of the second step is to test quantitatively the assumptions made during the initial stages of the model development. The last step of the validation is to establish that the model output data closely resembles the output data that would be expected from the real system. The first step, *face validation*, is best performed by the experts who have commissioned the system and who will eventually use the model. As we do not claim to have this level of expertise or experience, the validation performed here is limited to the last two steps.

3.1 Model Assumptions Validating

Sensitivity analysis has been used to check the validity of the model's assumptions. The model should behave in an "expected" way if one or more of the main influencing factors (or the input state parameters of the model, such as bus state parameters, flow state parameters, etc.) are changed. The set of all such input parameters are listed in Table 3.1.1. Sensitivity analysis could not realistically be performed on all input state parameters given the time allowed and the scope of this thesis. Hence, only the most critical input state parameters have been selected for testing (Table 3.1.1 lists the parameters chosen for sensitivity analysis testing).

The running time of a bus will be governed by the flow of the passengers (i.e. the number of passengers boarding and alighting determines the time needed for those activities) and by the states of the links and the stops. With this in mind, it follows that the flow state parameters, the stop state parameters and the link state parameters are the most important input state parameters for the simulation. Within the stop state parameters, the passenger arrival rate is the most important as it determines the number of passengers who will arrive at the stop, which will affect the time the bus needs to load and unload those passengers. The "% Rejection on 2nd Arrival"

State	Parameter	Selected for Sensitivity Analysis
Flow	Passenger Boarding Times (Secs)	
	- Mean Dead Time	✓
	- Marginal Rate	✓
	- Standard Deviation	✓
	Passenger Alighting Times (Secs)	
	- Mean Dead Time	
	- Marginal Rate	
	- Standard Deviation	
	Front Door Usage Patterns (%)	
	- Mean	
	- Standard Deviation	
Stop	Passenger Arrival Rate	✓
	% Rejection on 2nd Arrival	
	- when Queue length > #	
	- when L2 / L1 < #	
Link	Congestion Factor (Delay Multiplier)	✓
	Probability of Extraordinary Delay	
	- Average Increased Delay Time	
	- Standard Deviation	
	- Delay Start Time, End Time	
	Predefined Bus Breakdown Incident	
Regulation	Mean of the Bus Dwell at each Stop (Secs)	
	Standard Deviation of the Dwell (Secs)	
	Bypass Stop Conditions	
	Control Actions (Sch/Hdw/None, etc.)	✓
Bus	Passenger Capacity	
	Probability of each Type of Breakdown	
	Breakdown Probability in ID Tag	

Table 3.1.1

parameter represents the percentage of passengers in the queue who move to the second bus when two busses arrive at a stop simultaneously. The "% of Rejection on 2nd Arrival" will be zero unless two conditions are met (namely, the queue length for the first bus exceeds a user defined number, and the ratio of the load of the second bus to that of the first bus is greater than a user defined number). The "% Rejection on 2nd Arrival" was not viewed as a critical factor, and was therefore not chosen for testing. OC Transpo analysts advise that the parameters associated with the passenger boarding times (Mean Dead Time, Marginal Rate and Standard Deviation) are the most important flow state parameters, based upon these considerations: (i) for all stops (except perhaps a few stops at the end of each trip), passenger boarding times account for the major portion of the time buses spend at each stop; (ii) boarding and alighting occur simultaneously; however, boarding is more time consuming than alighting because time is needed to pay the fees or show a pass, get a transfer, get directions from the driver, etc. For the link state, the congestion factor is the most important input parameter: it determines road conditions and will affect the time the bus will spend on the line, whereas the parameters associated with extraordinary delays deal with infrequent situations (e.g., road construction) and are thus of secondary importance. For the regulation state, the control actions are the most important parameters: they determine the extra time a bus will spend at a stop to meet control targets (e.g., schedule adherence or headway control). At the moment, there is no need to test different bus types with the model; therefore, the bus state will not be of concern for the validation. For the selected input state parameters the assumptions are:

- If we change the parameters associated with the passenger arrival rate by a certain percentage, the total number of passengers arriving during the simulation time period is expected to change by about the same percentage. Due to the change in the passenger arrival rate, we expect changes in the times needed to board and alight those passengers and that will result in a change in the total transit times. Changes to the schedule performance measures (the percentages of earliness and lateness), as well as changes in the headway performance measures (such as the wait factors) are also expected.

- If we change the parameters associated with the passenger boarding times, we expect changes in the same direction for transit performance, schedule performance and headway performance.

• If we change the link congestion factors, we expect changes in link transit times which in turn will affect schedule performance.

• For the "control actions" parameter, we want to test whether or not buses are controllable at the control points. If we select the SC "Schedule Control" scenario, i.e. the control action is to hold early buses until the scheduled departure time, we expect to obtain the best schedule performance among the three default scenarios, NC, SC and HC. As buses adhere more closely to the schedule, we expect improvements in headway performance (as compared with the no control case). If we select the HC "Headway Control" scenario, i.e. the control action is to hold buses which are too close to the previous bus in order to achieve the planned headway, we expect to obtain the best headway performance of the three default scenarios, NC, SC and HC.

3.1.1 Testing Methods

To test the selected state parameters (except the regulation state parameters), the NC "No Control" scenario has been chosen in all cases, based on the opinions of OC Transpo experts. Furthermore we did not want the effects of the internal control actions related to either the SC scenario or the HC scenario to interfere with and hide the true variations resulting from the changes in the input parameters.

As described in the first chapter, there are different planned headway for different periods of day for route 95. For peak hours, the planned headway is 3 minutes; this makes testing different headway control strategies appealing, since it is believed the headway control strategies will be more effective than the traditional schedule control strategy for high frequency services. There are two peak periods: the AM peak (07:00 - 09:00) and PM peak (15:00 - 17:30). We chose the AM peak hour to perform all the tests. This choice was motivated by preliminary tests which showed that simulation results for the AM peak period are more consistent with the real (APC) data than those from the PM peak period.

Sensitivity analysis performed on the flow, stop and link parameters was performed in the following manner. For each parameter chosen, three settings were tested: the default setting which is the original setting of the parameter as defined by the model developer, the "increased setting", where we increased the value of the default setting by 50 percent, and the "decreased setting", where we decreased the default value of the parameter by 30 percent. For each of the three parameter settings, 10 simulation runs have been performed for statistical analysis purposes,

using 10 different random seeds. For each of the 10 runs for a given setting, we obtained the average and the standard error (the standard deviation divided by the square root of 10) of the 10 simulation results. For example, for the stop state parameter "Passenger Arrival Rate", we ran 10 replications for the default setting, 10 replications for the upper setting (+50%) and 10 replications for the lower setting (-30%). Table 3.1.2 summarizes the various experiments that were performed.

Test	Selected Parameter & Scenario	Setting	runs
1	None, NC	Default	10
2	Passenger Arrival Rate, NC	Increased	10
3	"	Decreased	10
4	Passenger Boarding Times, NC	Increased	10
5	"	Decreased	10
6	Congestion Factor, NC	Increased	10
7	"	Decreased	10
8	None, SC	Default	10
9	None, HC	Default	10

Table 3.1.2 Performed Tests

Lastly, for the regulation state parameter, control actions, we use the default settings for all the input parameters except the selection of the running scenario. We first select the SC scenario, run the simulation 10 times for the morning peak (07:00 - 09:00) with the control points set at all the time points, using 10 different random seeds. The procedure is then repeated for the HC procedure.

3.1.2 Statistical Techniques Applied in the Analysis of the Results

In [17], the following procedure is suggested for comparing two systems on the basis of a mean (or expected) response (performance measure). The approach is to construct a confidence

interval for the *difference* in the two expectations, as well as performing a hypothesis test to see whether the observed difference can be distinguished from zero. The confidence interval gives us this information (according to whether or not the confidence interval contains zero) as well as a measure of the *degree* to which the system responses are likely to differ, if at all. For $i = 1, 2$, let $X_{i1}, X_{i2}, ..., X_{in}$ be a sample of n IID observations (from simulation runs) from system i, and let $\mu_i = E(X_{ij})$ be the mean response of interest; we want to construct a confidence interval for $\xi = \mu_1 - \mu_2$. We can *pair* X_{1j} with X_{2j} to define $Z_j = X_{1j} - X_{2j}$ for $j = 1, 2, ..., n$. Then the Z_j's are IID random variables and $E(Z_j) = \xi$, the quantity for which we want to construct a confidence interval. Thus we can let

$$\bar{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}, \quad \text{and} \quad \hat{\sigma}^2[\bar{Z}(n)] = \frac{\sum_{j=1}^{n} [Z_j - \bar{Z}(n)]^2}{n(n-1)} \qquad (3.1)$$

and form the (approximate) 100(1-α) percent confidence interval

$$\bar{Z}(n) \pm t_{1-\alpha, n-1}\sqrt{\hat{\sigma}^2[\bar{Z}(n)]} \qquad (3.2)$$

If the Z_j's are normally distributed, this confidence interval is exact; i.e., it covers ξ with probability 1 - α; otherwise, we rely on the central limit theorem, which implies that this coverage probability is *near* 1 - α for large *n, or for distributions which are normal.* An important point here is that we did *not* have to assume that X_{1j} and X_{2j} are independent; nor did we have to assume that Var (X_{1j}) and Var (X_{2j}) are equal. [17]

3.1.3 Analysis of the Results of the Testing

• ***Results of Changing Passenger Boarding Times***

The simulation results show that for the transit performance (i.e., average travel time for each one-way trip, eastbound or westbound) the effects of changing passenger boarding times on total trip travel times are not as we expected. We only obtain relatively small changes for the total transit times. This may be explained by the underlying adjustment procedure found in the model, which we have discussed in Chapter 2, and which dictates that if a bus is behind schedule at a stop, the program will reduce the transit time for the line following the stop by a certain amount, thus simulating the accelerations made by the drivers. Since the passenger boarding times only represent a small proportion of the total transit times for the route, it is possible that

changes associated with the parameter(s) will be offset by the reductions of the transit times for the lines resulting from lateness. The comparison of the results is shown in Table 3.1.2. We performed the following statistical tests to see whether or not the changes in the passenger boarding times affect significantly the total transit times.

Case 1: Comparison a 50% increase of the initial dead time, the marginal rate and the standard deviation of the passenger boarding times with the default case. Let X_{ij} be the average total transit time from east to west, and Y_{ij} is the average total transit time from west to east, of the parameter set i on the jth replication. Here i =1 represents the increase in the parameters, and i = 2 represent the default case. We made n = 10 independent replications for each setting of the input parameters. Table 3.1.3 contains the results.

	East Bound			West Bound		
j	X_{1j}	X_{2j}	Z_{Xj}	Y_{1j}	Y_{2j}	Z_{Yj}
1	49.52	48.10	1.42	53.43	51.34	2.09
2	49.69	48.09	1.60	52.70	51.09	1.61
3	48.75	48.07	0.68	52.26	51.43	0.83
4	49.19	47.81	1.38	52.89	51.21	1.68
5	51.22	48.06	3.16	55.73	51.56	4.17
6	50.25	48.05	2.20	53.53	51.77	1.76
7	51.91	48.77	3.14	54.28	52.58	1.70
8	50.78	48.50	2.28	53.47	51.66	1.81
9	51.15	48.37	2.78	53.32	52.44	0.88
10	51.66	48.93	2.73	53.73	52.64	1.09

Table 3.1.3

Formally, we performed a statistical test of the null hypothesis

H_0: $E(Z) = 0.0$

versus

H_1: $E(Z) > 0.0$, for $Z = Z_X, Z_Y$

Step 1. Choose a level of significance α and a sample size n. Here we choose

$$\alpha = 0.05, \qquad n = 10$$

Step 2. Compute the sample means $\overline{Z_X}$, $\overline{Z_Y}$ and the sample standard deviation S_X, S_Y over the n replications

$\overline{Z_X}$ = 2.137 S_X = 0.839 (East Bound)

$\overline{Z_Y}$ = 1.762 S_Y = 0.946 (West Bound)

Step 3. Get critical value of t from the table of t-distribution. We have $t_{0.05,9} = 1.833$

Step 4. Compute the test statistic for Z = Z_X or Z_Y, and S = S_X or S_Y :

$$t_0 = \frac{\overline{Z} - \mu_0}{S/\sqrt{n}} ,$$

where $\mu_o = 0.0$, so that

$$t_{ox} = \frac{2.137}{0.796/\sqrt{10}} = 8.490 \quad (East\ Bound), \qquad t_{oy} = \frac{1.762}{0.897/\sqrt{10}} = 6.212 \quad (West\ Bound)$$

Step 5. For the one-sided test with H_1: $E(Z) > \mu_0$, reject H_0 if $t_0 > t_{\alpha,n-1}$. H_0 is clearly rejected for both the east bound and west bound. Therefore, we can make the conclusion that by increasing the passenger boarding times 50 percent the total transit times will be increased. With $\overline{\sigma}(\overline{Z_X}) = 0.252$ (east bound) and $\overline{\sigma}(\overline{Z_Y}) = .284$ (west bound), using Equation 3.1, we obtain the 95 percent confidence intervals of the increment of the total transit times:

[1.68, 2.60] (East Bound), [1.24, 2.28] (West Bound)

One approach for verifying whether the increases are reasonable in practice might be comparing 2.6 (upper confidence interval endpoint for eastbound transit time) with 48.28 (average eastbound transit time), which results in the ratio 2.6/48.28 ≈ 0.05, and comparing 2.28 (upper confidence interval endpoint for westbound transit time) with 51.77 (average westbound transit

time), which results in the ratio 2.28/51.77 ≈ 0.04. Thus, loosely speaking, the model yields at most a 5 percent increase (at the 95% confidence level) in the total transit times from a 50 percent increase in the passenger boarding times. To check whether these increases are reasonable, we made the following estimations.

• *Estimate the average of the total boarding times for each trip.* From the queue length report of the simulation, we find that the average queue length for the route (sum of the average queue length of each stop) is about 219. Using the default passenger boarding rate, 2.7 seconds per passenger, and noting that 67 percent of the passengers use the front door, the average total boarding time is about 219 × 2.7 x 0.67 / 60 ≈ 6.6 minutes. The 50% increase should be about 3.3 minutes.

• *Estimate the effects of the reduction of the congestion factor when the bus is late.* The reduction of the congestion factor happens at each line that is not proceeding one of the time points (as described in Chapter 2). The average total route transit time is about 50 minutes, and the average total lines transit time is estimated at 50 - 6.6 ≈ 43 minutes. Since there are 24 lines and 6 time points for each direction, the reduction of the congestion factor will happen at 24 - 6 = 18 lines at most for each trip, which will roughly affect 18/24 = 3/4 of the total line transit time. From the data used by the simulation model, we find the average congestion factor is about 1.72, whereas the effect of the reduction of the congestion factor would be around 0.12 times the line transit time. From the simulation results (see Table 3.1.4), the average lateness would be about 29 percent. Therefore a rough estimation of the effects of the reduction of the congestion factor would be 0.75 x 0.12 x 0.29 x 43 ≈ 1.1 minutes for the total line transit times.

Combining the above estimates, we estimate that the increment in the total transit time should be around 3.3 - 1.1 = 2.2 minutes. The simulation results show that the average increment is (2.137 (east bound) + 1.762 (west bound)) /2 ≈ 2 minutes, which agrees with the estimate. From these rough estimates it appears that the increase in the total transit time due to the 50% increase in the passenger boarding times is reasonable.

Next we would like to check whether the sample size of n = 10 is large enough for our purposes. Suppose we would like to accept a type II error of at most 10 percent. We make the following test, as described by Kelton and Law [17]. Let δ be given by:

$$\delta = \frac{|E(Z_X) - \mu_c|}{S_X} = \frac{2.137}{0.796} = 2.68 \qquad \textit{(East Bound)}$$

$$\delta = \frac{|E(Z_Y) - \mu_c|}{S_Y} = \frac{1.762}{0.897} = 1.96 \qquad \textit{(West Bound)}$$

For the one-sided t test for a level of significance $\alpha = 0.05$, from the chart of the Operating Characteristic Curves [17] we obtain

$$\beta(\delta) = \beta(2.68) = 0.05 \qquad \textit{for } n = 4 \qquad \textit{(East Bound)}$$

$$\beta(\delta) = \beta(1.68) = 0.05 \qquad \textit{for } n = 6 \qquad \textit{(West Bound)}$$

which reveals that a sample size of n = 10 is sufficient to guarantee the type II error be less than 0.10 as intended.

Case 2) Comparing the default case with a decrease in the initial dead time, the marginal rate and the standard deviation of the passenger boarding times. Let X_{ij} and Y_{ij} be the average total westbound and eastbound transit times respectively. Here $i = 1$ represents the default case, *and* $i = 2$ represents the case of decreased parameters. The simulation results are shown in Table 3.1.4.

As in the previous case, the null hypothesis is

$$H_0: \quad E(Z) = 0.0$$

but now we use

$$H_1: \quad E(Z) < 0.0, \quad \text{for } Z = Z_X, Z_Y$$

For $\alpha = 0.05$, and using the same steps described in Case 1), we obtain $t_0 = -3.105$ (East Bound) and $t_0 = -5.010$ (West Bound). Since they are both less than $-t_{\alpha,n-1} = -1.833$, we reject H_0. Therefore, we reach the conclusion that by decreasing the passenger boarding times 30 percent the total transit times will be decreased. The 90 percent confidence intervals of the decrease of the total transit times are:

[0.19, 0.74] (East Bound), [0.55, 1.19] (West Bound)

	East Bound			West Bound		
j	X_{1j}	X_{2j}	Z_{xj}	Y_{1j}	Y_{2j}	Z_{yj}
1	47.32	48.10	-0.78	50.15	51.34	-1.19
2	47.65	48.09	-0.44	50.16	51.09	-0.93
3	47.69	48.07	-0.38	50.89	51.43	-0.54
4	47.70	47.81	-0.11	51.39	51.21	0.18
5	48.29	48.06	0.23	51.39	51.56	-0.17
6	47.84	48.05	-0.21	51.04	51.77	-0.73
7	48.13	48.77	-0.64	50.86	52.58	-1.72
8	47.80	48.50	-0.70	50.72	51.66	-0.94
9	48.31	48.37	-0.06	50.97	52.44	-1.47
10	47.36	48.93	-1.57	51.42	52.64	-1.22

Table 3.1.4

Using the same heuristic argument as Case 1), we reach the conclusion that the decrease in total transit time due to the 30 percent decrease in passenger boarding times is reasonable.

The results show that the effects of changing the passenger boarding times are on average reasonable for bus schedule performance and headway performance. By increasing the mean dead time and boarding time, lateness increases and on-time and earliness decreases (as expected), since each bus will spend more time at each stop. The wait factor is also larger due to the larger variations of the passenger boarding times. By decreasing the passenger boarding times, we get more earliness and less lateness (as expected), since each bus will spend less time at each stop to load the passengers. The wait factor is also smaller due to the smaller variations of the passenger boarding times. The summarized comparison of the schedule performance is shown in Table 3.1.6, which gives the average percentage of earliness, on time and lateness over all the stops. The summarised comparison of the headway performance is shown in Table 3.1.7, which gives the average passenger wait factor over all the stops.

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
E Bound	48.28	0.108	50.41	0.328	47.81	0.104
W Bound	51.77	0.173	53.53	0.287	50.90	0.139

*Vary the passenger boarding times

Table 3.1.5 Comparison of the Transit Times (min.)

Setting	Earliness %		On Time %		Lateness %	
	Average	Std Err	Average	Std Err	Average	Std Err
Default	50.96	1.979	44.61	1.994	4.51	1.654
Increased	30.04	2.863	41.07	2.776	28.92	3.774
Decreased	60.75	1.610	38.33	1.604	0.95	0.359

*Vary the passenger boarding times

Table 3.1.6 Comparison of the Schedule Performance

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
Wait Factor%	160.47	4.984	237.02	13.616	146.82	2.786

*Vary the passenger boarding times

Table 3.1.7 Comparison of the Wait Factor

• ***Results of Changing the Passenger Arrival Rate***

For the same reasons, the effects on transit performance of changing the passenger arrival rate appears to be small. The comparison of the average total transit times and the standard errors for the tests are shown in Table 3.1.8. The effects of the change of the passenger arrival rate are quite reasonable for schedule performance, for headway performance and for the passenger queue length report. By increasing the passenger arrival rate, we get more lateness and less earliness, as expected, since each bus will spend more time at each stop to load and unload more

passengers. Also, the maximum passenger wait times and the wait factor are increased in this case because of the longer boarding and alighting times, and the larger standard deviations of passenger boarding and alighting times. Conversely, by decreasing the passenger arrival rates, we get less lateness and more earliness since each bus will spend less time at each stop. Maximum passenger wait times and wait factors are also decreased because of the smaller standard deviations of the passenger boarding and alighting times. The comparison of average schedule performance over all the stops and the standard errors of the tests are shown in Table 3.1.9. The comparison of the average wait factors over all the stops as well as the standard errors of the tests are shown in Table 3.1.10. By increasing the passenger arrival rate by 50%, we get approximately 50% more passengers arriving over all the stops. By decreasing the passenger arrival rate by 30%, we get approximately 30% fewer passengers arriving over all the stops. The comparison of total passengers arriving at all stops as well as the standard errors of the tests are shown in Table 3.1.11. More detailed comparisons based on each stop are shown in Figures 3.1.1 to Figure 3.1.4.

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
E Bound	48.28	0.108	49.08	0.116	47.79	0.046
W Bound	51.77	0.173	53.37	0.255	51.07	0.114

*Vary the passenger arrival rate

Table 3.1.8 Comparison of the Transit times (min.)

Setting	Earliness %		On Time %		Lateness %	
	Average	Std Err	Average	Std Err	Average	Std Err
Default	50.96	1.979	44.61	1.994	4.51	1.065
Increased	33.45	2.728	44.25	3.322	22.32	4.081
Decreased	60.26	1.542	38.80	1.516	1.00	0.322

*Vary the passenger arrival rate

Table 3.1.9 Comparison of the Schedule Performances

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
Wait Factor%	160.5	5.00	200.4	9.77	149.6	3.51

*Vary the passenger arrival rate

Table 3.1.10 Comparison of the Wait Factors

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
# Passengers	7822.5	342.9	12526.2	713.8	5424.8	223.8

*Vary the passenger arrival rate

Table 3.1.11 Comparison of the Total Number of Arrived Passengers

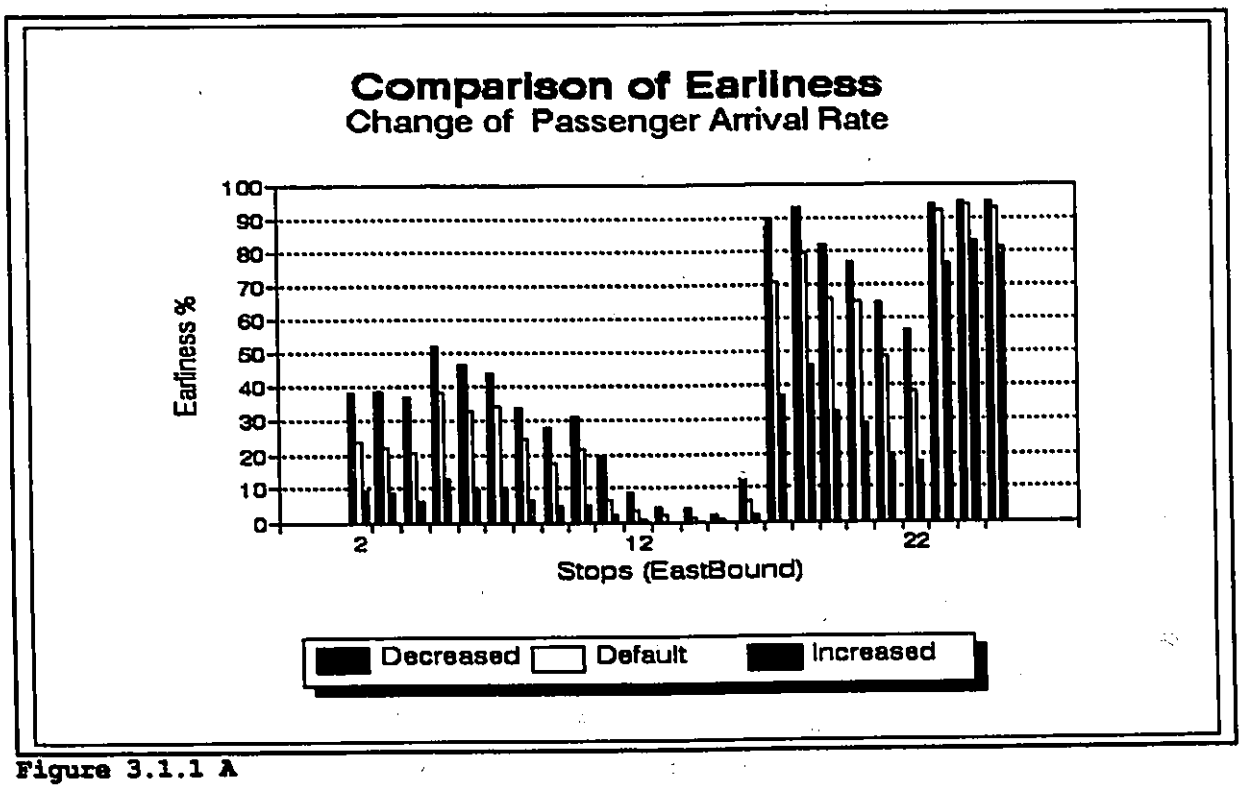


Figure 3.1.1 A

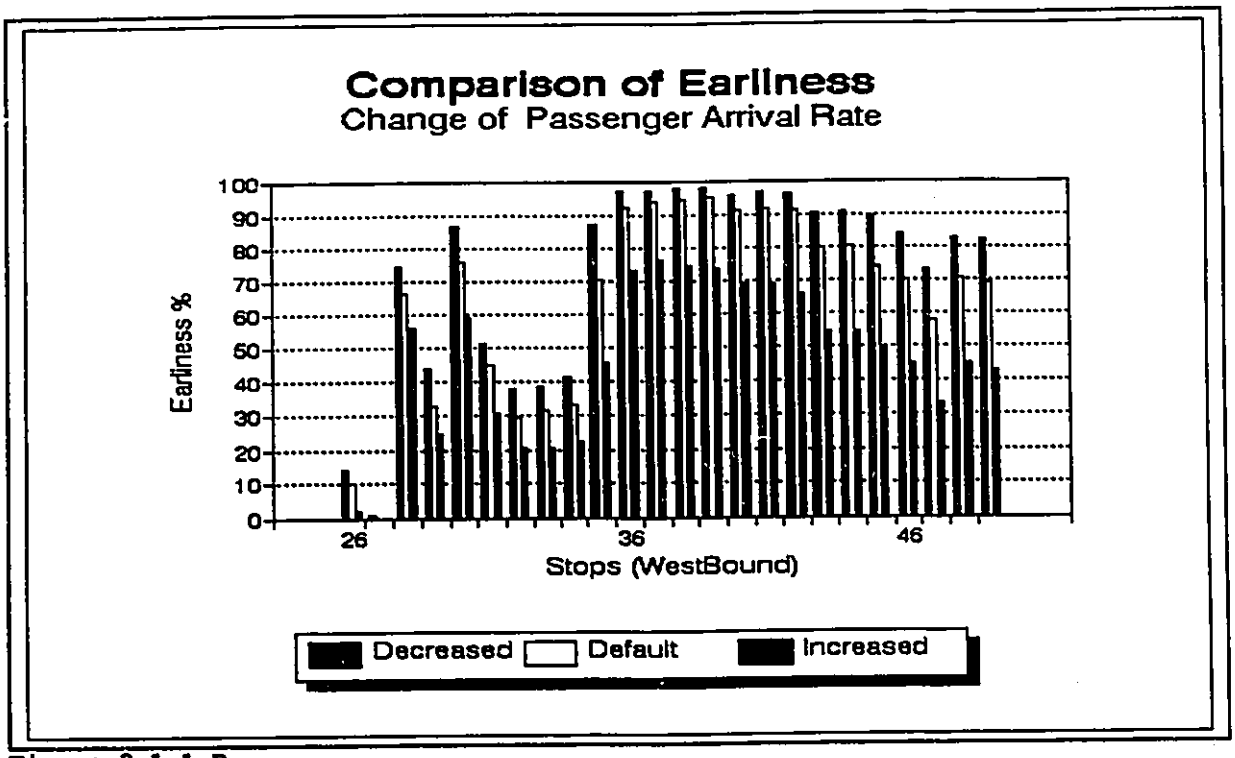


Figure 3.1.1 B

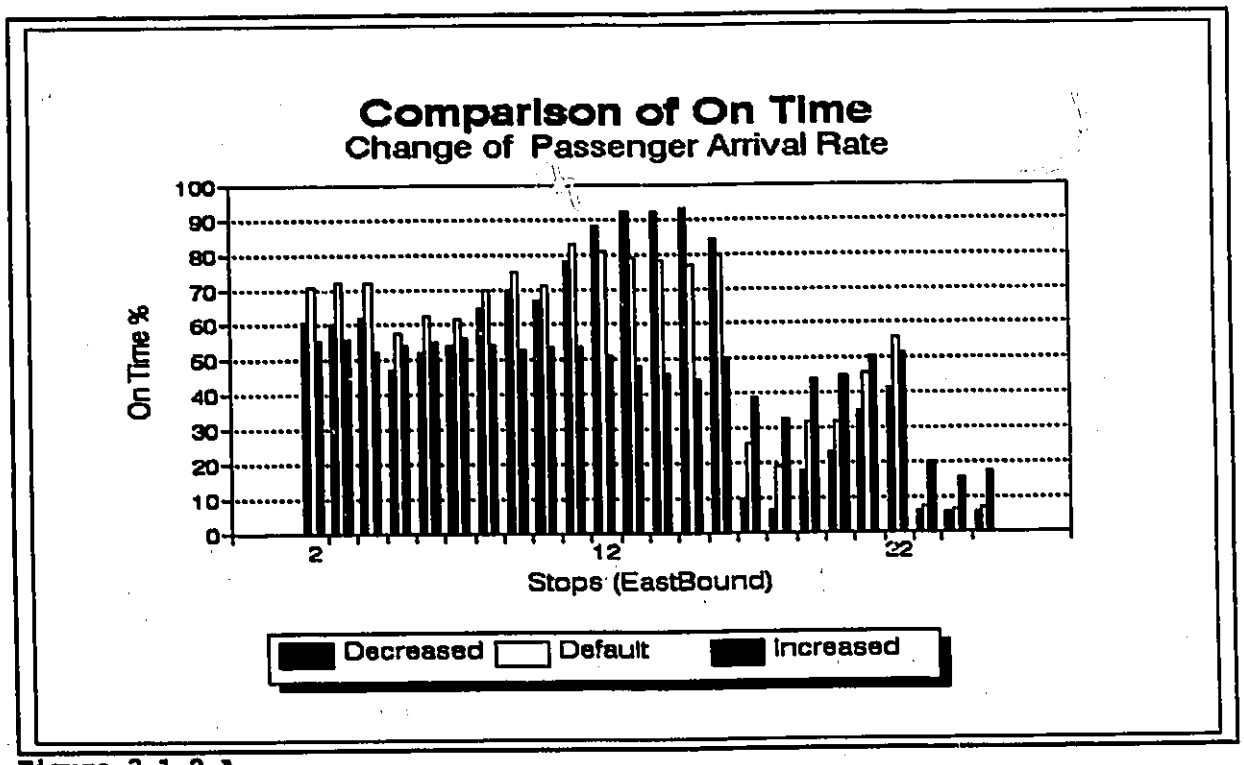


Figure 3.1.2 A

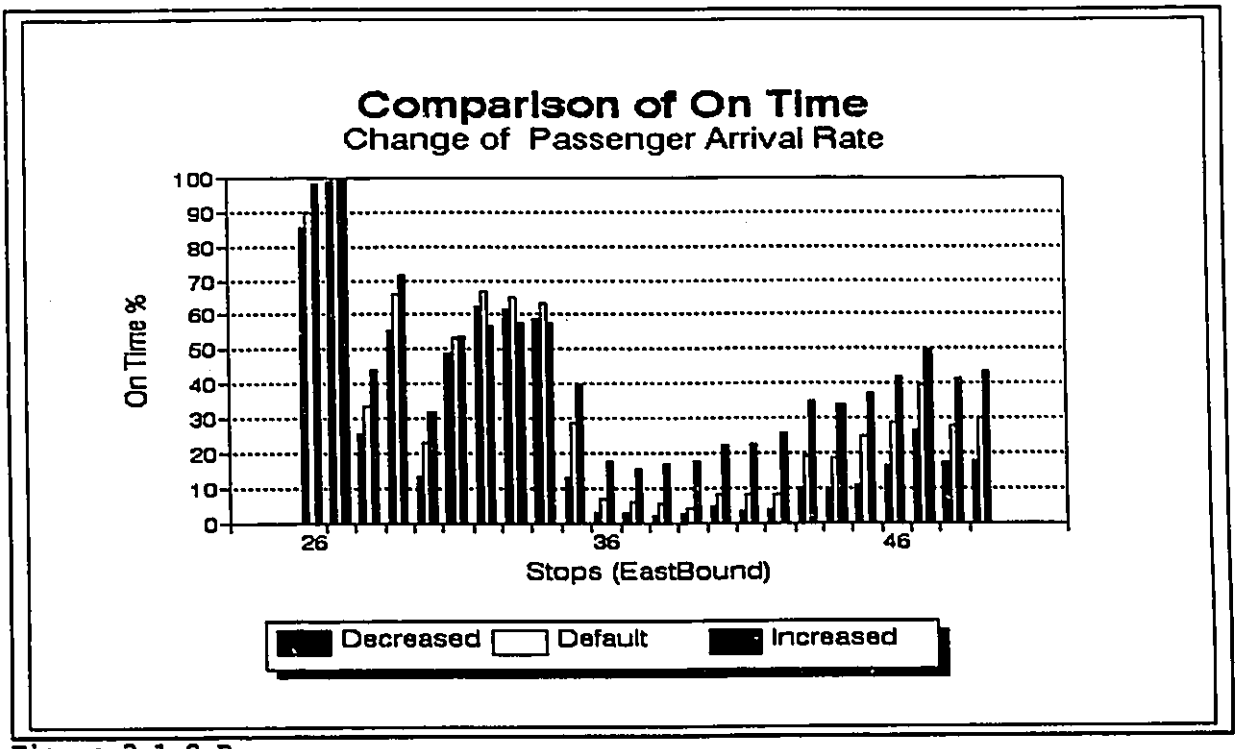


Figure 3.1.2 B

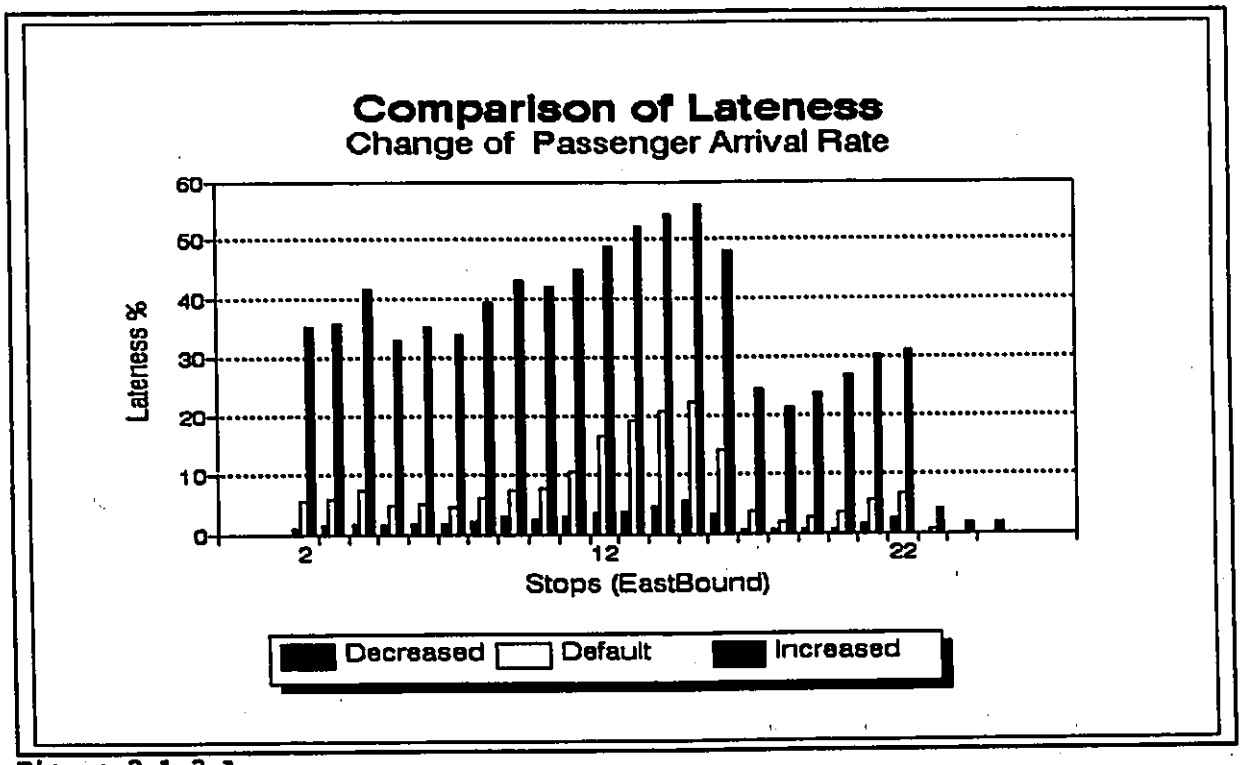


Figure 3.1.3 A

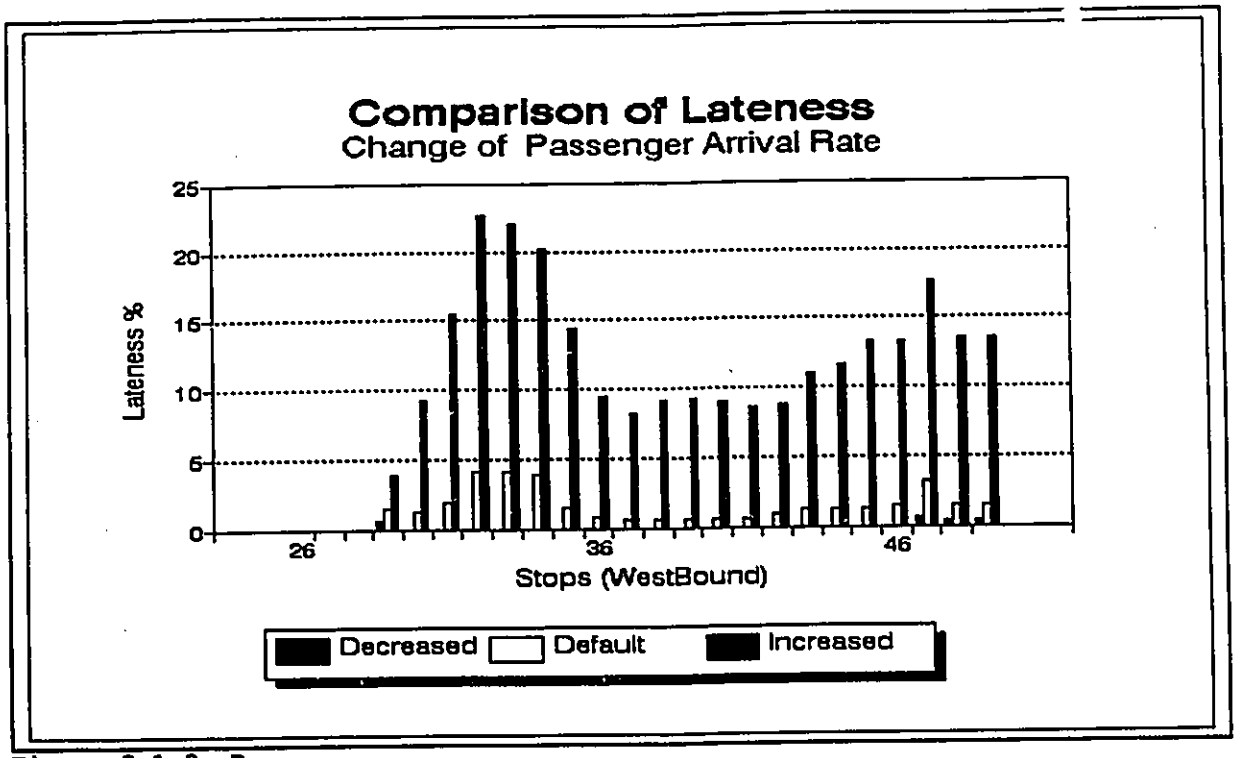


Figure 3.1.3. B

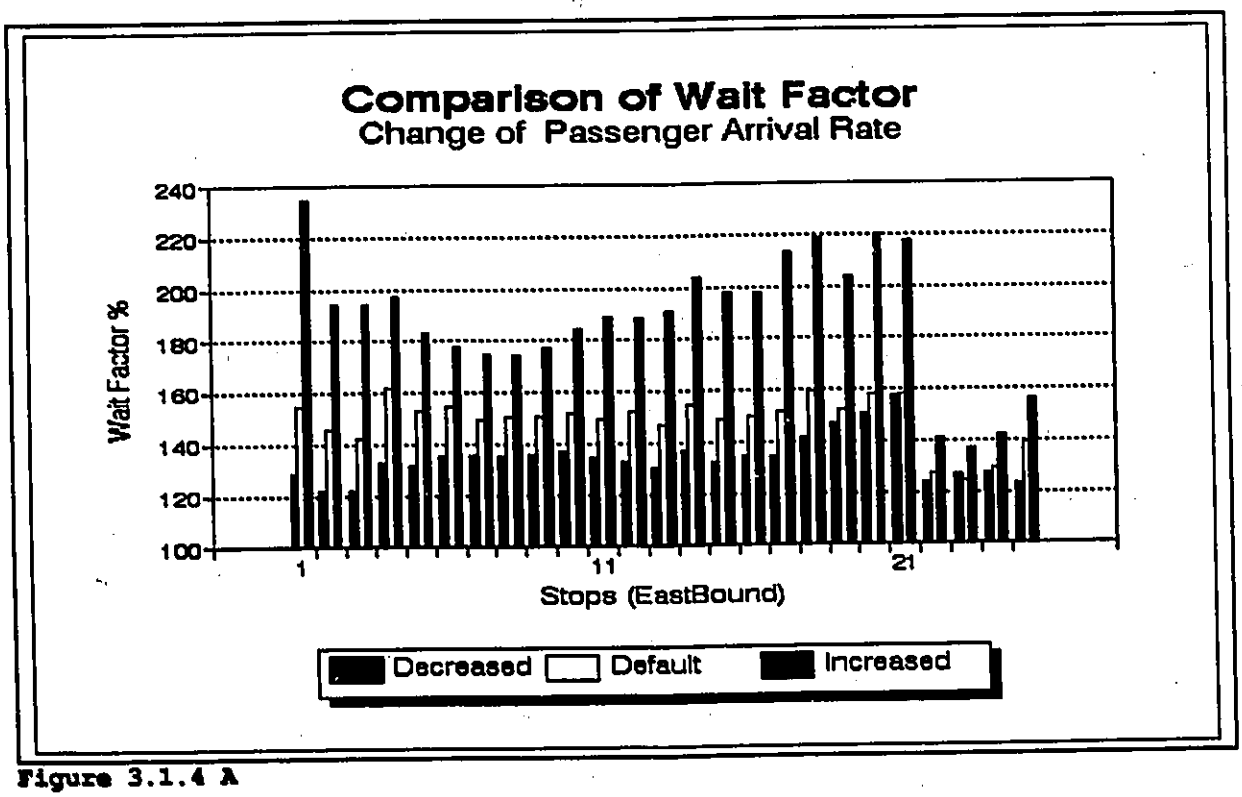


Figure 3.1.4 A

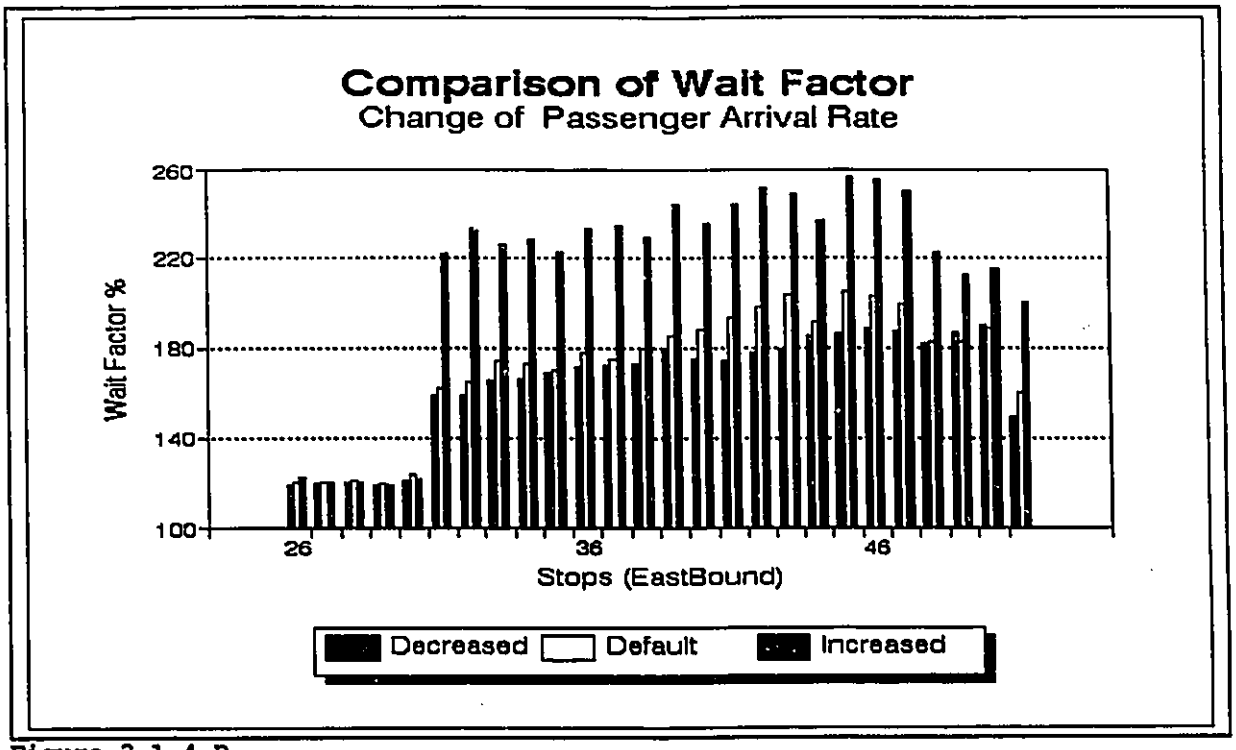


Figure 3.1.4 B

• ***Results of Changing the Link Congestion Factors***

The effects of the change of the congestion factors are quite reasonable. By increasing the link delay caused by congestion, we get longer travel times to complete the trip, more lateness and less earliness, as expected, since each bus will take more time to travel each link. Conversely by decreasing the congestion factor modifier, we get shorter times to complete the route, less lateness and more earliness, since each bus will take less time to cover each line. The summarised comparisons are shown in Table 3.1.12 to Table 3.1.13.

	Default Setting		Increased Setting		Decreased Setting	
	Average	Std Err	Average	Std Err	Average	Std Err
E Bound	48.28	0.108	58.54	0.170	39.81	0.171
W Bound	51.77	0.173	61.56	0.228	43.93	0.167

*Vary the congestion factor

Table 3.1.12 Comparison of the Transit Times

Setting	Earliness %		On Time %		Lateness %	
	Average	Std Err	Average	Std Err	Average	Std Err
Default	51.0	1.98	44.6	1.99	4.5	1.07
Increased	1.1	0.21	21.6	1.16	77.4	1.11
Decreased	90.5	0.93	9.5	0.93	0.0	0.03

*Vary the congestion factor

Table 3.1.13 Comparison of the Schedule Performances

• **Results of Changing** • ***Control Actions***

We compared the NC, HC and SC scenarios with the control points set at the time points for schedule performance (measured by the percentage of earliness, on time and lateness) and for headway performance (measured by the wait factor). Results show that the SC scenario yields the best schedule performance (the largest percentage of the buses on time) and the HC scenario yields the best headway performance (the lowest wait factor) as expected. Furthermore, with the SC scenario, the percentage of earliness is 0 (i.e. no busses leave early) for the control points; with the HC scenario, the minimum headway is equal to or greater than the planned headway at the control points. The results show that the model has the capability or effectively holding busses according to schedule adherence or to headway control policies. The summarised comparisons are shown in Table 3.1.14 - Table 3.1.15, which contain the average values over all runs and over all the stops, and the related standard errors.

	Earliness %		On Time %		Lateness %	
	Average	Std Err	Average	Std Err	Average	Std Err
NC Scenario	51.0	1.98	44.6	1.99	4.5	1.07
HC Scenario	42.9	2.16	38.1	3.50	19.1	2.72
SC Scenario	32.4	1.03	64.3	1.57	3.3	0.87

*Vary the control actions

Table 3.1.14 Comparison of Schedule Performances

	NC Scenario		HC Scenario		SC Scenario	
	Average	Std Err	Average	Std Err	Average	Std Err
Wait Factor%	183.0	5.56	146.0	2.17	152.0	3.17

*Vary the control actions

Table 3.1.15 Comparison of the Wait Factors

3.2 Validating Input-Output Transformations

"In this phase of the validation process, the model is viewed as an input-output - that is, the model accepts values of the input parameters and transforms these inputs into output measures of performance. It is this correspondence that is being validated." (Banks and Carson [6]). Banks and Carson also recommend that instead of validating the model input-output transformations by predicting the future, the past historical data can be used for validation purposes. Thus the accurate "prediction of the past" may replace predication of the future for the purpose of validating the model.

Banks and Carson indicate that to conduct a validation test using historical input data, it is important that all the input data and all the system response be collected over the same time period, otherwise, the comparison of model responses to system responses could be misleading. All the input state parameters of the model are derived from the APC data collected during the Spring of 1992 (January 1 to April 30). We expect that the simulation results will match the "real system" results, i.e. the simulation results will be reasonably close to the same set of APC data for the period of interest. Law and Kelton [17] question whether hypothesis testing is an appropriate statistical approach for this step of validation. Since the model is only an approximation to actual system, a null hypothesis that the system and model are the "same" is clearly false. They believe that it is more useful to ask whether or not the differences between the system and the model are significant enough to affect any conclusions derived from the model.

Comparisons of the Transit performance

The total transit performance obtained by simulation were compared to the available APC data. We were interested in determining the average travel time and its standard deviation, in both directions, for the whole route, and also for the downtown section of the route (stops between and including Lebreton and Campus). As the model could only report the average transit times for the whole route, we had to modify the program slightly so that the required statistics could be computed. Again the NC scenario was chosen as it was judged important not to bias the results with the underlying control actions found in the two other scenarios. 10 runs were generated for each of the time period: AM peak (07:00 - 09:00), daytime off-peak (11:30 -14:00) and PM peak (15:30 - 17:30). As shown in Table 3.2.1 to Table 3.2.3. the simulation results seem to be consistent with APC data.

07:00 - 09:00	Simulation Result		APC Data	
Route	Average	Std Dev	Average	Std Dev
EB Whole	48.91	2.37	47.00	2.50
WB Whole	52.67	2.62	57.00	4.63
EB Down Town	10.84	0.81	10.57	1.04
WB Down Town	9.72	1.14	11.03	1.28

Table 3.2.1 Comparison of the Transit Times (min.)

11:30 - 14:00	Simulation Result		APC Data	
Route	Average	Deviation	Average	Deviation
EB Whole	47.0	1.8	50.0	2.9
WB Whole	45.0	2.2	47.0	2.6
EB Down Town	10.2	1.1	11.0	1.3
WB Down Town	8.4	0.7	10.1	1.1

Table 3.2.2 Comparison of the Transit Times (min.)

15:30 - 17:30	Simulation Result		APC Data	
Route	Average	Deviation	Average	Deviation
EB Whole	52.1	2.4	57.0	2.7
WB Whole	52.2	4.5	53.0	2.3
EB Down Town	12.2	1.1	12.6	1.9
WB Down Town	10.6	1.7	12.3	1.4

Table 3.2.3 Comparison of the Transit Times (min.)

Comparisons of the Bus Load on Departure

We then compared the simulation results with the APC data for the bus load on departure of each stop. Again, the NC scenario has been chosen to run the simulation. The simulation was run 10 times by using 10 different random seeds for both the morning peak hour (07:30 to 08:30) and the afternoon peak hour (15:15 to 17:15), as the APC data was available for these time periods respectively. We took the average of the 10 simulation results to compare with the APC data. Since the route 95 has two running patterns for each direction as described in Chapter 1, we use the appropriate APC data for each of the running patterns. However the simulation only gives a single bus load on departure report, i.e. it mixes the data of the two running patterns. This makes the comparison with the APC data difficult. In order to make the comparison, the two sets of APC data (short and long running patterns) have been aggregated into a single set. The major items of the simulation load on departure report for each stop are: minimum load, maximum load and the average load (the APC data has the same type of information). We made following aggregation of the two sets of APC data: take the minimum of the minimum loads of the two running patterns at each stop; take the maximum of the maximum loads of the two patterns at each stop; and take the average of the average loads of the two running patterns at each stop. This approach is required for all of the 95 route except for the section between "Blair" and "Place d'Orleans", where aggregation is not necessary, since there is only one pattern which occurs over this segment of the 95 route. The comparison for the morning peak hour is shown in Figure 3.2.1. From the figures we can see that on average the simulation results follow the pattern of

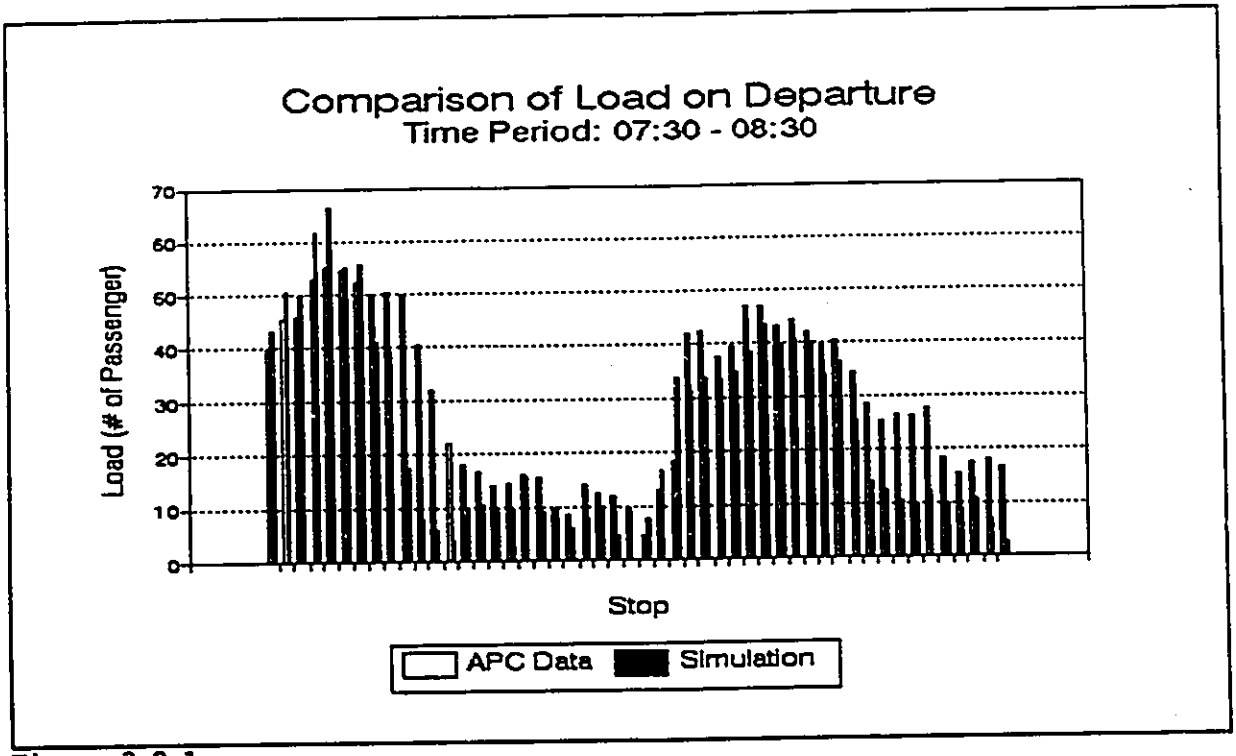


Figure 3.2.1

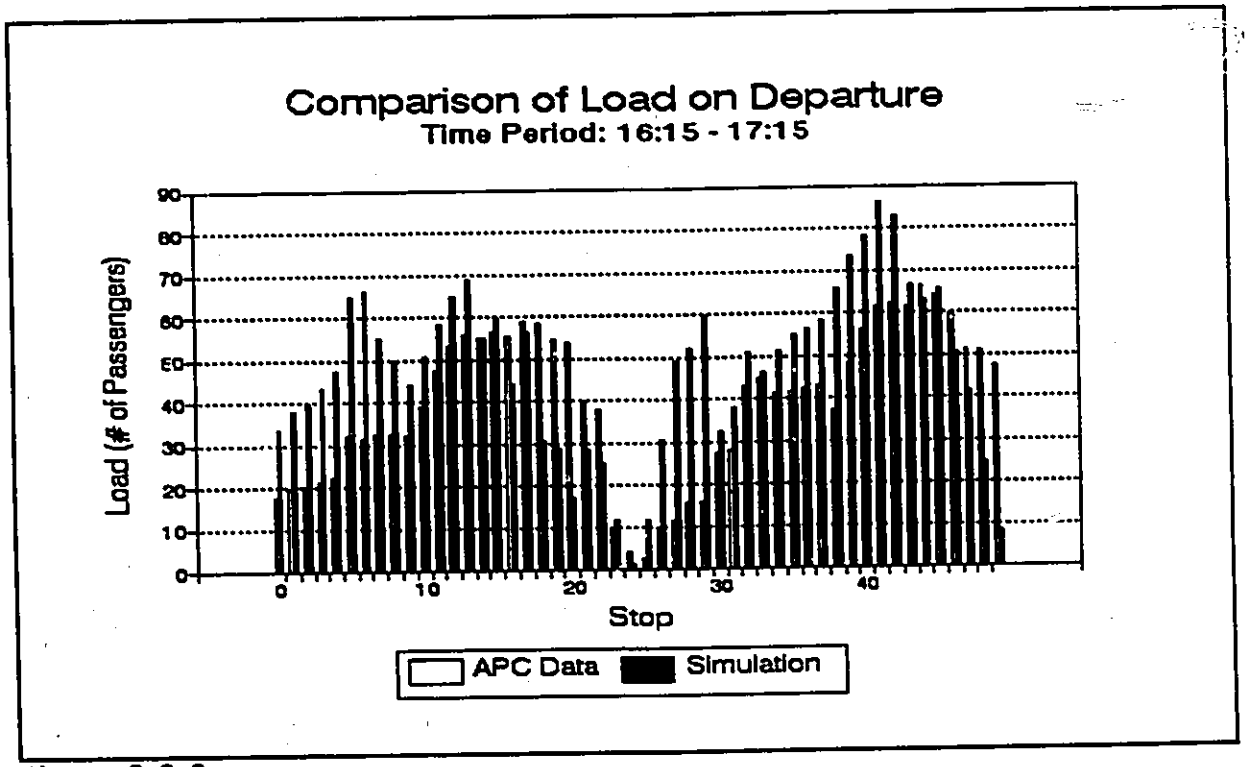


Figure 3.2.2

the APC data, with a difference ranging between -31 and 15. Using the APC data as the independent variable and the simulated average load on departure as the dependant variable, linear regression analysis gives a slope (x-coefficient) of about 1.0 and an *R Squared* is about 0.7, which shows the simulation result is moderately to highly correlated with the APC data. The regression slope was tested for significance, and yielded a p-value of 0.0. The comparison for the afternoon peak is shown in Figure 3.2.2, which reveals a relatively larger discrepancy between the simulation results and the APC data. The regression analysis gives an *R Squared* of about 0.2, which shows a weak correlation between the PM simulation results and the APC data, although the regression is significant at the 0.0 level (p-value). These results suggest that further calibration of the simulation model will be useful, especially for the afternoon peak time period.

3.3 Brief Conclusions of the Validation

From the above limited validation testing, we can see that the model in general yields the expected trends for sensitivity analysis, which shows that the model can be used to compare different control strategies in a relative sense. However, further calibration of the model is needed for the PM peak hours, before using this time of day for further experiments. Thus, further testing was restricted to the AM peak period only.

4. Testing of Real-Time Control Policies on the Simulation Model

The major goal of our experiment was to study the effectiveness of a variety of real-time control policies to improve service reliability, and to use the knowledge acquired as the basis for developing a knowledge-based system for the purpose of on-line bus control assistance. Until recently, most public transit agencies have been using schedule adherence control policies where drivers are given a timetable showing scheduled departure times for specific stops along the route and are asked to adhere to this schedule. Such a policy has traditionally been enforced by street supervisors. As discussed previously, schedule adherence control is reasonable for low frequency routes. However, for high frequency routes, headway control seems most appropriate (Barnett [5]). In particular, route 95 has the characteristics identified by Abkowitz and Tozzi [2] for headway-based control to be effective: it is a cross-town route with high-frequency service so that random passenger arrivals to bus stops are to be expected. As headway changes dynamically in a real system, it has been difficult in the past to implement such a policy without the help of electronic systems such as AVLC. For this reason, only a simple headway control policy has been implemented in practice: the threshold policy, described in Chapter 1 (for an example, see Abkowitz and Lepolsky [1]).

As part of the knowledge acquisition process, it is reasonable to study both the headway threshold policy and the schedule adherence policy, and to measure their effectiveness in improving service reliability. However, as new technology (such as AVLC and knowledge-based systems) are being developed and implemented, more sophisticated real-time control policies can now be considered and implemented. In section 4.1, we describe such a new real-time control policy, the "message board" policy which will also be evaluated in this chapter. A hybrid policy which is based on both the "message board" policy and on the threshold headway control policy will also be tested. For the sake of comparison, we also decided to model a policy which closely ressembles that recently used by OC Transpo controllers.

This chapter is composed of five sections. In section 4.1, we describe the new message board policy as well as an hybrid policy based on the concepts of the message board policy and of the threshold headway control policy. In section 4.2, we provide the experimental design of the tests we will perform on the various control policies. The technical modifications that needed to be

made in the simulation model to allow such experiments to be undertaken will be described in section 4.3. The results and analysis of the experiments are provided in section 4.4. A brief discussion and conclusions follow in Section 4.5.

4.1 New Headway-Based Control Policies

We now propose a new headway-based policy which we will refer to as the "message board" policy. This policy combines ideas from the **threshold** and **prefol** policies described in Chapter 1. When bus $i + 1$ is ready to depart the control stop, if the difference between the current time and the departure time of the previous bus, i, is less than a threshold value, t, bus $i + 1$ is delayed long enough so that the actual headway between these two buses at that stop will be equal to the planned headway h (in seconds). We let the threshold value be $t = h - 20$, for practical reasons: this provides a certain degree of flexibility in the application of the policy as it implies that headway between 160 and 180 seconds is acceptable: a bus will be delayed at the control stop only if the deviation from the planned headway is "significant", i.e. more than 20 seconds. However, if bus i left more than h seconds before the current time, bus $i + 1$ is released. Since the actual headway between these two buses is greater than h, the system next considers delaying bus i by sending a delay message. Delaying bus i would increase the gap between i and $i - 1$. Before sending the message, the system checks the headway between i and $i - 1$ when i left the control stop. If that headway was already larger than h, the system will not try to delay bus i since this would likely increase the already "large" gap between i and $i - 1$. However, if the gap between i and $i - 1$ was less than or equal to h when i left the control point, then a message will be sent to bus i asking it to delay its departure from the next stop it leaves. The delay time for bus i is set to be half the difference between the actual headway between buses $i + 1$ and i, and h, the planned headway (which is similar to the prefol idea of "splitting the difference"). More precisely, the procedure is in fact designed so that a delay message is only sent if the headway between i and $i + 1$ is more than $h + 20$ seconds, so that a bus is never delayed by less than 10 seconds: this makes sense from a practical point of view. Hence no control action is taken if the headway is judged "reasonable", i.e. between $h - 20$ seconds and $h + 20$ seconds. Furthermore, the procedure has been designed so that buses would never be delayed by more than 90 seconds

at a given stop. This is to ensure that the passengers on board will not be too much inconvenienced by long delays, and will have a better chance at finding their connecting bus at transfer points. This also prevents, to some extent, against bus bunching.

In practice, this method may be implemented by asking a street supervisor to take note of the headway at a given control stop, and to send the delay message by radio when needed. In the simulation model, the "radio message" was replaced by a "message board": each time a delay message - consisting of the number of the bus to which it is destined and the amount of time that bus had to be delayed - was sent, it was stored on the message board. Before leaving any given stop, a bus would first look at the message board: if there is no message, and that stop was not a control point, the bus would leave immediately. If there is a message, the bus would be delayed by the required amount of time, and the system would erase the message before the departure of the bus so that the same message is not executed twice.

Generalization of headway-based control policies may also be considered. The most immediate is to increase the number of control stops along the route. Typically, a single control stop would be chosen, primarily for practical reasons, as in the experiment described in Abkowitz and Lepofsky [1]. With simulation, it is easy to maintain many control points: at the limit, all stops could be selected as control points. Another way of implementing headway-based holding was suggested in Koffman [14]: the threshold policy was applied at every stop provided there was passenger activity: if no one was alighting or boarding, the bus would simply go on to the next stop without any control action taken. This approach seems less desirable however as one can imagine a case where an almost empty early bus catches up to its predecessor: it would find no passengers at the various stops, and, if no passengers alight, it would never stop so that no holding action would delay it.

One of the risks of having many control points is that the positive effect of an action taken at one stop may be reduced by an action taken at a future stop: making too many control actions could be a source of variability. For the message board policy, maintaining many control stops creates difficulties in managing the messages; for example, before a message is generated for a given bus, the system must first determine whether there already exits a message for that bus, (in which case, the new message should replace the old as it is based on a more recent evaluation of headway). However, if there are no previous message on the message board, the

system must be able to know when the last message was read so that the bus is not delayed excessively.

Finally, it is also possible to combine policies to form hybrid policies. We now wish to propose such a hybrid policy which combines ideas found in both the message board policy and the threshold headway control policy. In this hybrid policy, a limited number of control stops will be defined as "message board control stops", where the message board policy will be implemented. For practical implementation reasons, the number of message board control stops will be limited to two (one in each direction). It is expected that the positive effects resulting from the implementation of the message board policy will fade as buses move away from the message board control stops. As the threshold policy is easier to implement, several other stops strategically located along the route (in both directions) could be selected as threshold headway control stops, where the threshold policy would be implemented. This would permit better headway control all the way through the route. Thus, this hybrid policy, as well as our basic message board policy, and the more traditional threshold headway control and schedule adherence policies will all be tested.

4.2 Evaluating the Effectiveness of Certain Real-Time Holding Policies

One of the goals of the experiments is to compare the effectiveness of the various control policies, listed in the previous section, in particular the new "message board" policy, in providing good service reliability. It is expected that service reliability will increase if the number of control stops increases. Hence, a second goal of our experiments is to measure the extent to which service reliability improves with the number of control stops. For this reason, we decided to test the various control policies with different numbers of control points. At the minimum, one control point is needed in each direction. Furthermore, given AVLC detectors are present at six locations along the route (i.e., the so-called "time-points") making real-time control actions possible in practice, we decided to also test the policies when all time-points are chosen as control points. Finally, even if this may be very difficult to implement in practice, we decided to measure the effectiveness of the various control policies if all stops were used as control points.

One more question needed to be answered: for the experiments involving only two control points, one in each direction, which stops should be selected as control points? As we specifically wanted to gain insight into the new message board policy, we used this policy to identify which stops should be chosen as control points. Turnquist and Blume [26] have indicated that while the major incentive for making headway more regular was to reduce waiting time of passengers who board at or beyond the control point, the major cost of such a policy is borne by passengers on board since they are delayed when the bus is held up. Based on a simple analytical model, these authors suggest that headway control points should be located near the start of the bus route, at a stop where it is known that the variability of headway is significant. In order to clearly identify the impact of each candidate control point, for practical reasons we chose only one stop at a time for the whole route. Again for practical reasons we decided to concentrate only on stops equipped with AVLC detectors. Based on these observations, we tested three possible locations for the single control point: we considered placing the single control point at Baseline Station (first stop, eastbound), at Lebreton Station (eighth stop, eastbound), and at Blair Station (first stop, ("Blair pattern"), and sixth stop ("Orléans pattern"), westbound). Another experiment considered placing two control points, one at Baseline Station, and the other at Blair Station (hence, one in each direction).

From these experiments it became clear that if only one control point is to be implemented for the message board policy, it is most effective to choose Blair Station. Furthermore, maintaining two control points (at "Blair" and "Baseline" stations) was shown to be marginally better than having only one at Blair.

Given the AVLC system maintains six time points in each direction, (including Baseline, Lebreton and Blair stations), the SC scenario with the time points as control stops, a policy resembling that used by OC Transpo controllers, would have a total of twelve control points. Hence we considered creating an hybrid policy also consisting of twelve control points, for comparison purposes. Based on prior testing of the message board policy, our hybrid strategy was designed to have two of the time points (Baseline and Blair) to perform "message board" headway control, and the remaining ten time points as threshold headway control points. The message board policy was implemented by adding a sub-program in the simulation model to perform message board headway control on the chosen stops: the hybrid policy was created by

simply running the headway control (HC) scenario with Blair and Baseline as message board control stops, and the other time points as threshold control stops. The hybrid procedure and the SC scenario would then have the same control points (i.e. the twelve time points).

Our message board policy (with two control points) was also tested against both SC and HC scenarios with the same two control points, again to compare policies having the same number of control points.

Finally, since we decided to verify the effectiveness of SC and HC when all stops were control points the hybrid policy was also modified to maintain threshold headway control at all stops except at Baseline and Blair where message board headway control was performed.

Note that for the two hybrid policies discussed above, a message may be received by a bus while that bus is at a threshold control stop. The delay time contained in the message will simply be added to the delay computed by the threshold policy, if any. The possibility of overly long delays being generated in this way will have to be examined in future research.

In summary, a total of nine (9) holding policies have been tested. Under the schedule control strategy, the following policies were tested: S_2 (schedule control at two control points, Blair and Baseline), S_TP (schedule control at the time points), S_Al (schedule control at all stops). Under the headway control strategy, the following policies were tested: T_2 (threshold policy with two control points, Blair and Baseline), T_TP (threshold policy with the time points at control stops), T_Al (threshold policy where all stops are control points), M_2 (message board policy with control at Blair and Baseline), M_TP (message board control at Blair and Baseline, threshold control at all other time points), and M_Al (message board control at Blair and Baseline, threshold control at all other stops).

For each control policy tested, we ran the simulation ten times for the morning peak hours, from 7:00 am to 9:00 am where the planned headway is three (3) minutes (except between Blair Station and Place d'Orléans, where the planned headway is six (6) minutes), using ten different random seeds. The same seeds were used for each policy tested for statistical purposes. The measure of performance chosen, also used by OC Transpo, is called the Wait Factor (WF). It is based on the the regularity index and is defined as follows for each stop:

$$WF = 2\frac{AWT}{h} \times 100\% \tag{4.1}$$

where the *AWT* is the average wait time of passengers at the stop, and *h* is the planned headway for that stop. The *AWT* may be calculated using expression (1.1) (for the expected wait time), where the expected headway is replaced by the average simulation headway at that stop, and the headway variance is the sample headway variance of the experiment. The wait factor should normally have a value close to 100%. To see this, note that given random (Poisson) arrivals of passengers at the stop, if the variance of headway was 0, the expected waiting time would be half the planned headway, and WF = 100%. The value of WF is an indication of how much, in percentage, the actual waiting time deviates from the expected "ideal" waiting time (when headway variance is 0). We computed the average wait factor over the 10 runs as well as the wait factor variance for each stop.

The results of the simulation experiments are presented in Section 4.4.

4.3 Technical Modifications to the Simulation Program to Support the Testings

There is a routine called "Bus_Regulation" in the simulation program to determine if the bus should be held at the stop for extra time according to the "No Control" scenario, the "Headway Control" scenario or the "Schedule Control" scenario respectively. The routine will be used twice each time a bus stays at a stop. The first time, it is called at the arrival of the bus at the stop, and it is used to estimate the extra time the bus will dwell at the stop in order to calculate the number of passengers that will be arriving during this extra dwelling time. The second time this routine is called is after the boarding and alighting of the passengers at the stop. It is then used to determine the actual extra time the bus should dwell at the stop before it can depart from the stop, based on the chosen control policy. In order to test our "Message Board" control policy as well as other hybrid control policies, a procedure has been added to the end of the "Bus Regulation" routine to implement the "Message Board" control policy at selected control points. In this way, the pure "Message Board" control policy can be tested on the selected control point(s), if the "No Control" scenario is selected. There will be no control at any other stops, but there will be controls at the selected control point(s) using the added procedure. Hybrid control

policies can also be tested out, since for the control points, the bus will be regulated according to the selected scenario, i.e. either "Headway Control" or "Schedule Control"; for the control points where the "Message Board" control policy is applied, the added procedure will be used to activate the "Message Board" control policy. For the added procedure designed to implement the "Message Board" control policy, the detailed logic relations contained in the bus regulation routine are shown in Figure 4.3.1. Note that in this flow chart, there can be only one message at a time on the message board: if an unread message remains on the message board, no new message will be generated. We were constrained to implement the message board policy in this way because of major complications related to modifying the original simulation source code to allow more messages to be maintained simultaneously. However, as will be discussed in Chapter 5, we were able to implement a message board policy capable of maintaining as many messages as required on the message board through the use of our prototype knowledge-based system.

4.4 Results of the Experiments

As we mentioned in Chapter 2, there are 50 stops in both directions (25 stops in each direction) for route 95, and 12 of them are time points (equipped with AVLC detectors). There are also two running patterns in each direction (Pattern 1 is from Baseline to Blair, Pattern 2 from Baseline to Place D'Orléans, Pattern 3 from Blair to Baseline and Pattern 4 from Place d'Orléans to Baseline). To simplify the presentation of our results, we label the stops numerically. Table 4.4.1 gives the labels of the time points. Stops with numbers between 6 and 16 exclusively, and between 37 and 47 exclusively are in the downtown section (going east and west respectively). Every second bus which starts at "Baseline" (#1) will run Pattern 1, i.e. stop at "Blair EB" (#21) and go back to "Baseline"; the other one will run Pattern 2, i.e. go to "Place D Orléans" (#26, #27) and back through "Blair WB" (#32) to Baseline). Therefore, the headway for the stops #22 - #31 will be twice the planned headway. If the planned headway over the rest of the route is 3 minutes, the planned headway for those stops would be 6 minutes.

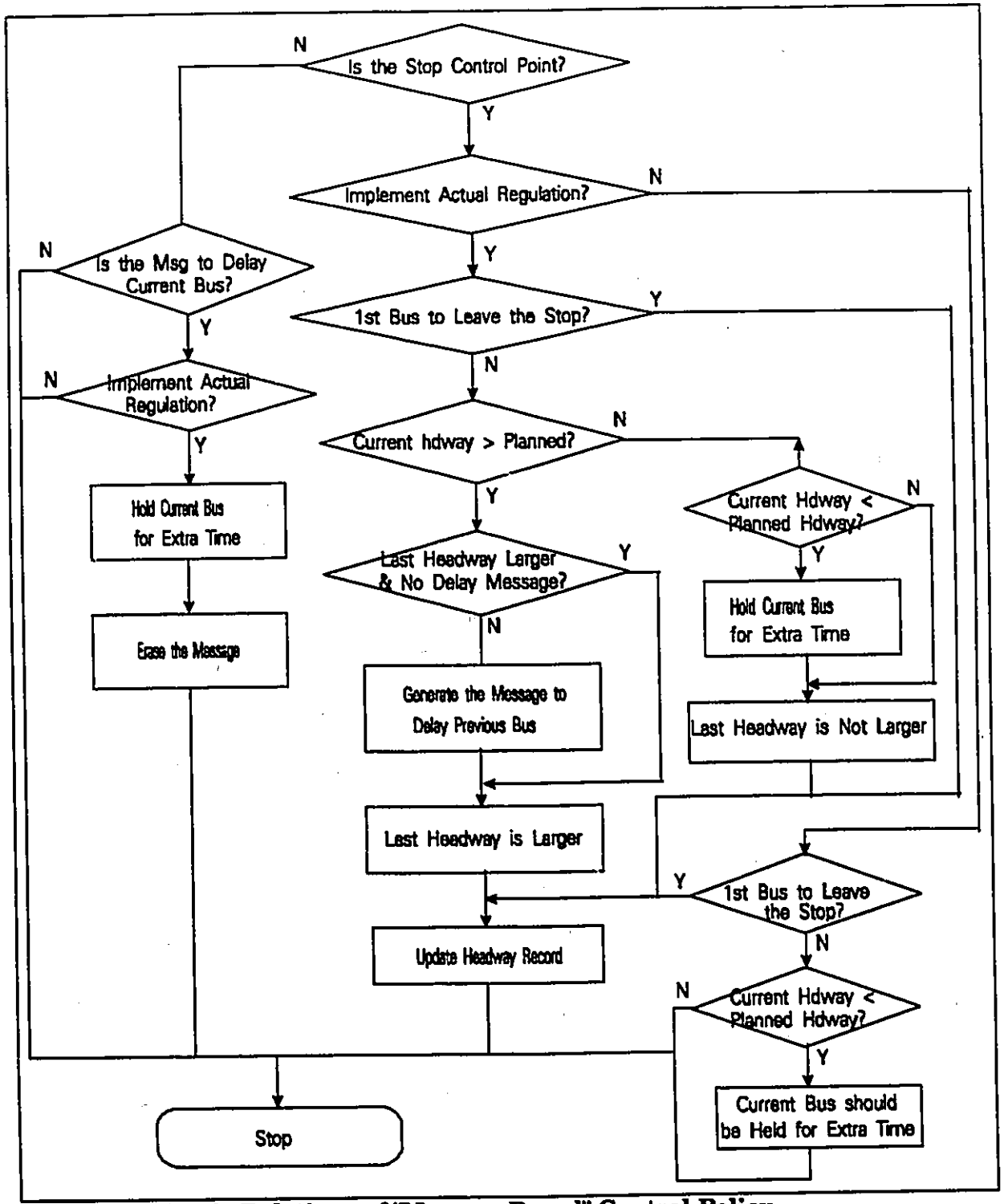


Figure 4.3.1 Logic Relations of "Message Board" Control Policy

Number	Stop	description
1	Baseline	Time Point, Start stop for pattern 1 and 2
5	Westboro EB	Time Point
7	Lebreton EB	Time Point, terminal stop of Down Town section
15	Campus EB	Time Point, terminal stop of Down Town section
21	Blair EB	Time Point for pattern 2, End stop for pattern 1
26	Place D Orleans EB	Time Point, end stop for pattern 2
27	Place D Orleans WB	Time Point, start stop for pattern 4
32	Blair WB	Time Point, start stop for pattern 3
38	Campus WB	Time Point, terminal stop of Down Town section
46	Lebreton WB	Time point, terminal stop of Down Town section
48	Westboro WB	Time point

Table 4.4.1 Descriptions of some of the stops

Our goal is to compare different control policies for all the stops of the route. In doing this, the same statistical techniques described in the previous chapter were employed for comparisons of those control policies. To compare one control policy with another, the differences of the average and the variation of the wait factors of the results were computed for each stop, from which 90 percent confidence intervals were derived and statistical hypothesis test were performed.

Another way used to compare two control policies is the range analysis. For each stop, the average wait factor over all simulation runs was computed. We partitioned these average wait factors into several groups according to the range of the wait factors. Since there is no special importance or weight given to any specific stop, i.e. all the stops are treated equally, we expect all the stops to have the same range of wait factor. In some cases, it is ambiguous to compare two control policies using only the average wait factors over all the stops. There may be very small wait factors for some stops and very large wait factors for other stops for one control policy while there are relatively stable wait factors among all the stops for the other

control policy, yet the average wait factor over all the stops could be almost the same for the two control policies. Turnquist [24] suggests that service reliability may be defined as the variability of a system performance measure over time: we argue that minimizing the variability of a performance measure over all stops, while keeping the mean close to target, is also an important element of reliable service. In those cases, the range analysis can often clearly show the difference of the two control policies. For the range analysis, policy A is said to be better than policy B if there are more wait factors in the smaller range group and fewer in the larger range group for policy A than those for policy B. For the example shown in Table 4.4.2, where f(i) represents the number of the stops in the range i and F(i) represents the cumulative number of stops with the wait factor below the upper limit of range i, for the 50 wait factors related to the 50 stops, policy A would be judged to be better than policy B since there are more stops with the wait factor in the smaller range for policy A.

Range	Interval	Policy A		Policy B	
i	min, max	f(i)	F(i)	f(i)	F(i)
1	0, 110	0	0	0	0
2	110, 120	30	30	10	10
3	120, 130	15	45	20	30
4	130, 140	5	50	10	40
5	140, 150	0	50	10	50

Table 4.4.2

As mentioned in the previous section, if only one control point is to be implemented for the message board policy, it is most effective to choose Blair Station. Lebreton as a single control point was shown to be least desirable, providing only marginal improvement over the no control scenario (NC). As shown in Figure 4.4.1, which represents the result of a typical simulation run, it is clear that using Blair (stop 31) as opposed to Baseline (stop 1) as the single control point for the message board policy has two main advantages: it provides more stable and smaller wait factors (on average) across all stops. As noted in Turnquist [25], the most effective location of

a control stop is near the start of a trip (explaining the marginal success of Lebreton), where there is a large variability in headway (this is where Blair had an hedge over Baseline). The larger variability at Blair comes probably from the fact that the distance between the various stops between Place d'Orléans and Blair are much larger than average, and because the headway of that area is six instead of three minutes (only every other bus goes to Orléans).

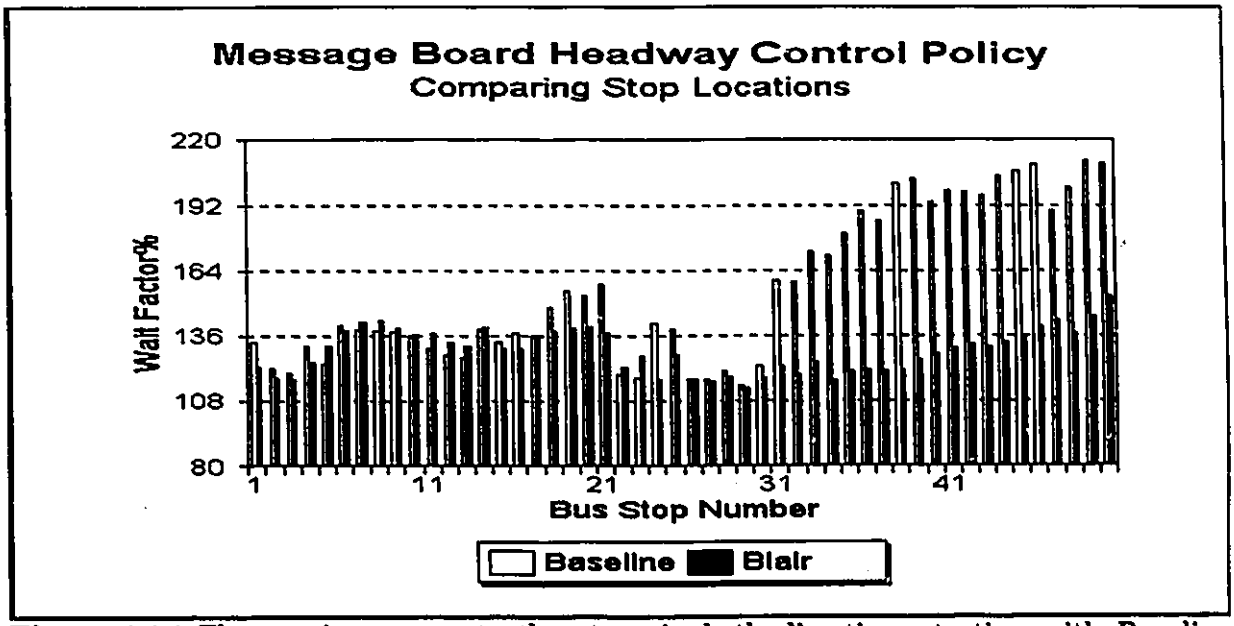


Figure 4.4.1 The x-axis represents the stops in both direction, starting with Baseline (stop 1) and going eastbound towards Place d'Orléans, (stop 25). Stops 26 to 50 represent the stops for the westbound trip.

Studying our message board policy with one or two control points allowed us to measure the impact of that policy and to obtain additional insight on the operation of route 95 (as represented by the simulation). Given or experiments covered a two hour am peak period with a planned headway of three minutes (for most stops), we simulated the departure of approximately 20 buses per hour at Blair and Baseline. Approximately 80 bus departures were subject to control. In a typical run, 25 buses would be delayed at one of the control points, and most of these delays occurred at Blair. At least 9 and as many as 14 delay messages were sent. Most originated from Blair, but the average delay was largest for the messages coming from Baseline. The large majority of messages sent from Baseline were "executed" four stops later at Westboro station: occasionally, the Baseline messages were executed at the third or fourth stop

eastbound (i.e. Queensway Station and Lincoln Fields Station). It is much more difficult to predict where a Blair message would be executed: in most cases, the messages were read at Train Station (stop 35), three stops west of Blair, but messages could be read as early as Cyrville Station (the stop following Blair), and as late as Laurier Station (7 stops after Blair). All these observations tend to point to a large degree of variability in running times between stops leading to Blair, as argued previously, but also between the first few stops following Blair. This may explain why controlling at Blair proved to be so effective in our experiments.

We have chosen to use the two control points policy, i.e. one control point for each direction, for further testing and comparison. We choose "Baseline" (Stop 1) as the control point for the east bound and "Blair" (Stop 32) for the west bound.

The effectiveness of our two control point message board policy, M_2 for brevity, was next tested against two other holding policies which also maintained only two control points at Blair and Baseline: a schedule control policy, S_2 for brevity, and a headway-based threshold policy, T_2. The two headway-based policies clearly outperform the schedule control policy. The message board policy seems to perform marginally better than the threshold policy. This may be explained by the fact that the message board policy is somewhat more sophisticated than the threshold policy: a control action is taking place not only at the control points, but also at the stops where the messages are executed. When a bus arrives at the stop where the message is going to be read and executed, the gap between that bus and the bus ahead may be subject to change. The gap may remain less than or equal to the planned headway, or become larger than the planned headway. If the gap is larger than the planned headway, the execution of the message (i.e. to delay the bus) will create a even bigger gap. The message board policy could be more effective, if the gap would be subject to fewer changes. This is one of the reasons which motivated the development of the hybrid control policy which combines the message board control policy applied at the two points and the threshold control policy applied at some of the remaining stops. The hope was to obtain a more constant headway at the stops where the message will be read and executed.

The comparisons of the hybrid policy M_TP and the pure time points threshold control policy, T_TP are shown in Table 4.4.3 to Table 4.4.5.

Stop	$\bar{Z}$	$\bar{\sigma}(Z)$	90% c. i.		Tcalc	Best
1	6.66	2.15	3.69	9.63	3.10	*B
2	2.36	3.35	-2.28	7.00	0.70	?
3	4.07	3.13	-0.25	8.39	1.30	?
4	4.57	2.10	1.66	7.47	2.17	B
5	12.83	2.85	8.88	16.77	4.50	B
6	5.47	1.89	2.86	8.09	2.89	B
7	4.76	1.72	2.38	7.14	2.77	B
8	4.05	1.78	1.59	6.52	2.27	B
9	3.49	1.87	0.91	6.07	1.87	B
10	3.86	1.93	1.20	6.53	2.00	B
11	3.44	2.00	0.67	6.21	1.72	B
12	4.58	2.54	1.08	8.09	1.81	B
13	3.70	3.11	-0.60	8.01	1.19	?
14	6.23	2.79	2.37	10.08	2.23	B
15	7.15	2.98	3.02	11.27	2.40	B
16	3.82	1.54	1.69	5.96	2.47	B
17	3.72	1.47	1.69	5.75	2.54	B
18	3.13	2.38	-0.16	6.41	1.32	?
19	3.39	1.16	1.78	4.99	2.91	B
20	6.67	1.50	4.60	8.75	4.45	B
21	8.37	1.64	6.10	10.64	5.10	B
22	-0.09	1.48	-2.14	1.96	-0.06	?
23	0.56	1.24	-1.16	2.28	0.45	?
24	5.46	6.05	-2.90	13.83	0.90	?
25	-0.95	6.43	-9.85	7.94	-0.15	?
26	-0.58	1.44	-2.57	1.42	-0.40	?

*A = Threshold Control Policy *B = "Message Board" Control Policy

Table 4.4.3_A Comparisons of Two Control Policy (Time Points)

Stop	$\bar{Z}$	$\bar{\sigma}(Z)$	90% c. i.		Tcalc	Best
27	-0.70	1.34	-2.55	1.16	-0.52	?
28	0.43	1.47	-1.60	2.47	0.30	?
29	1.19	1.31	-0.63	3.00	0.90	?
30	-1.01	1.27	-2.77	0.75	-0.79	?
31	-0.72	1.11	-2.25	0.81	-0.65	?
32	-1.20	1.34	-3.05	0.65	-0.90	?
33	-1.07	2.09	-3.96	1.83	-0.51	?
34	-1.68	3.94	-7.12	3.77	-0.43	?
35	2.48	2.17	-0.51	5.47	1.15	?
36	-1.01	4.14	-6.74	4.72	-0.24	?
37	-0.69	3.64	-5.72	4.35	-0.19	?
38	2.04	1.68	-0.29	4.37	1.21	?
39	2.83	1.68	0.50	5.16	1.68	B
40	4.79	2.18	1.78	7.80	2.20	B
41	7.12	2.04	4.30	9.94	3.49	B
42	7.65	2.41	4.32	10.97	3.18	B
43	8.14	2.02	5.34	10.94	4.02	B
44	7.67	2.74	3.87	11.46	2.80	B
45	1.53	1.94	-1.15	4.21	0.79	?
46	1.98	2.00	-0.79	4.74	0.99	?
47	2.35	1.78	-0.12	4.81	1.32	?
48	-0.01	1.22	-1.71	1.68	-0.01	?
49	0.83	1.55	-1.32	2.97	0.53	?
50	-0.30	1.34	-2.15	1.55	-0.22	?
Avg	3.07	0.69	2.11	4.02	4.45	B

*A = Threshold Control Policy *B = "Message Board" Control Policy

Table 4.4.3_B Comparisons of Two Control Policy (Time Points)

Range	Interval	Threshold Control		Message Board	
i	min, max	f(i)	F(i)	f(i)	F(i)
1	0, 110	0	0	0	0
2	110, 120	16	16	19	19
3	120, 130	22	38	28	47
4	130, 140	11	49	3	50
5	140, 150	1	50	0	50

Table 4.4.4 Range Analysis, all stops (Control at Time Points)

Table 4.4.3 shows the average and standard deviation of the difference of the wait factors for the two policies over 10 runs, the 90% confidence interval of the difference and the result of the statistical hypothesis testing (using the equation 3.1 of Chapter 3) for each stop. The M_TP policy provides better average wait factor for almost half of the stops compared to T_TP policy. For the rest of the stops, it is difficult to tell which policy is better. For the whole route, the M_TP policy gains an approximate 5% decrease in the wait factor at the 90% confidence level compared to T_TP policy and the statistical hypothesis testing indicates that M_TP policy is better than T_TP. Table 4.4.4 shows the range analysis for all the stops, which indicates that for the wait factors larger than 130 there are only 3 stops for M_TP policy while there are 12 stops for T_TP policy. Table 4.4.5 shows the range analysis for downtown.

Range	Interval	Threshold Control		Message Board	
i	min, max	f(i)	F(i)	f(i)	F(i)
1	100, 120	2	2	5	5
2	120, 125	6	8	6	11
3	125, 130	3	11	5	16
4	130, 135	5	16	2	18
5	135, 140	1	17	0	18
6	140, 145	1	18	0	18

Table 4.4.5 Range Analysis, downtown (Control at Time Points)

These two policies again clearly outperformed the schedule control policy with control at time points, S_TP for brevity. The comparisons of S_TP and M_TP are shown in Table 4.4.6 to Table 4.4.8.

Stop	$\bar{Z}$	$\bar{\sigma}(Z)$	90% c. i.		Tcalc	Best
1	15.06	2.33	11.83	18.28	6.46	*B
2	12.49	2.89	8.49	16.50	4.32	B
3	9.93	2.53	6.43	13.44	3.92	B
4	20.83	3.39	16.14	25.52	6.14	B
5	17.65	2.42	14.30	21.00	7.28	B
6	16.11	2.82	12.22	20.00	5.72	B
7	12.10	1.99	9.35	14.84	6.08	B
8	13.70	2.13	10.75	16.64	6.43	B
9	14.00	2.42	10.66	17.34	5.79	B
10	14.16	2.53	10.66	17.66	5.60	B
11	12.69	3.56	7.76	17.61	3.56	B
12	13.44	4.38	7.38	19.50	3.07	B
13	9.87	4.95	3.02	16.72	1.99	B
14	9.75	5.00	2.84	16.66	1.95	B
15	8.10	3.79	2.85	13.34	2.13	B
16	8.90	3.64	3.86	13.93	2.44	B
17	8.13	4.36	2.10	14.16	1.86	B
18	1.45	4.15	-4.29	7.19	0.35	?
19	0.52	1.66	-1.78	2.82	0.31	?
20	2.63	2.72	-1.13	6.39	0.97	?
21	-0.08	1.86	-2.66	2.50	-0.04	?
22	-2.38	1.24	-4.09	-0.67	-1.93	*A
23	-2.75	1.09	-4.26	-1.24	-2.52	A
24	-9.45	3.61	-14.44	-4.45	-2.61	A
25	-4.95	3.63	-9.97	0.08	-1.36	?
26	-1.65	1.00	-3.04	-0.27	-1.65	A

*A = Schedule Control Policy *B = "Message Board" Control Policy

Table 4.4.6_A Comparisons of Two Control Policies(Time Points)

Stop	$\bar{Z}$	$\bar{\sigma}(Z)$	90% c. i.		Tcalc	Best
27	-1.22	0.96	-2.56	0.11	-1.27	?
28	-0.09	1.14	-1.66	1.48	-0.08	?
29	1.36	0.88	0.15	2.58	1.55	*B
30	-1.42	1.28	-3.20	0.35	-1.11	?
31	1.19	1.86	-1.38	3.77	0.64	?
32	1.52	1.99	-1.24	4.27	0.76	?
33	3.26	2.82	-0.65	7.16	1.15	?
34	-0.48	4.92	-7.28	6.32	-0.10	?
35	7.62	2.34	4.38	10.85	3.25	B
36	4.64	4.69	-1.84	11.13	0.99	?
37	4.06	4.52	-2.18	10.31	0.90	?
38	7.55	1.75	5.12	9.97	4.31	B
39	9.75	2.31	6.57	12.94	4.23	B
40	12.61	2.56	9.07	16.15	4.93	B
41	14.03	4.06	8.42	19.64	3.46	B
42	16.19	4.53	9.92	22.46	3.57	B
43	13.00	4.15	7.25	18.74	3.13	B
44	12.98	3.34	8.35	17.60	3.88	B
45	4.64	2.36	1.37	7.90	1.97	B
46	4.18	2.24	1.09	7.28	1.87	B
47	6.90	1.08	5.40	8.40	6.36	B
48	15.89	2.22	12.82	18.97	7.15	B
49	17.58	2.35	14.33	20.82	7.49	B
50	14.57	1.76	12.13	17.00	8.27	B
Avg	7.41	1.22	5.72	9.10	6.08	B

*A = Schedule Control Policy *B = "Message Board" Control Policy

Table 4.4.6_B Comparisons of Two Control Policies(Time Points)

M_TP is better than S_TP for most of the stops except for the section where the headway is 6 minutes, twice the planned headway. Since headway control strategies are most effective for high frequency services (planned headway is less than 4 minutes), for a 6-minute headway service, the headway control strategy was expected to be less effective.

Range	Interval	Schedule Control Policy		Message Board Policy	
i	min, max	f(i)	F(i)	f(i)	F(i)
1	0, 110	0	0	0	0
2	110, 120	9	9	19	19
3	120, 130	15	24	28	47
4	130, 140	21	45	3	50
5	140, 150	5	50	0	50

Table 4.4.7 Range Analysis

Range	Interval	Schedule Control Policy		Message Board Policy	
i	min, max	f(i)	F(i)	f(i)	F(i)
1	100, 120	0	0	5	5
2	120, 125	2	2	6	11
3	125, 130	1	3	5	16
4	130, 135	8	11	2	18
5	135, 140	4	15	0	18
6	140, 145	2	17	0	18
	145, 150	1	18	0	18

Table 4.4.8 Range Analysis

We next wanted to determine how much improvement, if any, was achieved in increasing the number of control points for these three policies. They behaved similarly: as the number of

control points increased (from two (2), at Blair and Baseline, to the twelve (12) time points, to all fifty (50) stops, all the policies showed improvements in the average wait factor for most of the stops. However, most of the total improvement came as a result of increasing the number of control stops from two to twelve, thus suggesting that the investment associated with increasing the number of control points yields diminishing returns. Table 4.4.9 shows the aggregated statistical comparisons for each of the three control strategy: the mean of the related statistical calculations over all the stops. Each row of the table represents the comparison of using two different number of control points for the same control strategy. As a reminder, the symbol for each policy is made of 3 characters, the first one stands for the control strategy and last one represents the number of control points. For example, M_TP represents the "Message Board" control using time points as control points.

A vs B	$Z_A - Z_B$	σ	90% c. i.		Tcalc	Best
S_2 vs S_TP	24.64	1.30	22.84	26.44	18.92	B
S_TP vs S_Al	2.47	1.36	0.59	4.34	1.82	B
T_2 vs T_TP	12.07	0.67	11.15	13.00	17.98	B
T_TP vs T_Al	5.53	0.85	4.36	6.71	6.52	B
M_2 vs M_TP	12.95	1.38	11.05	14.85	9.41	B
M_TP vs M_Al	3.37	0.66	2.46	4.28	5.12	B

Table 4.4.9 Effects of the Number of Control Points

To conclude our analysis of the simulation experiments, we have computed aggregate statistics: for all stops, over all runs. The results appear in Table 4.4.10, in which each row represents the comparison for one pair, i.e. Policy A vs Policy B. It shows that "Message Board" control policy performed best in all the cases, either using the two control points or using the time points as the control points, while the two headway control policies perform better than the schedule control policies.

In future research, one may decide to give more importance to certain stops. For example, stops where important boarding occurs should be treated with more attention in the control

activities. These stops are likely to change with time of day. For instance, it is reasonable to assume that stops leading to downtown must be controlled tightly during the AM peak, whereas the stops in the downtown core show be closely monitored during the PM peak. This suggests the use of different control strategies, or even the choice of different control stops as function of time of day. More research is needed in this area.

A vs B	$Z_A - Z_B$	σ	90% c. i.		Tcalc	Best
S_2 vs T_2	16.91	1.55	14.76	19.05	10.90	B
S_2 vs M_2	19.10	2.24	16.00	22.20	8.53	B
T_2 vs M_2	2.19	1.23	0.49	3.89	1.78	B
S_TP vs T_TP	4.34	1.32	2.52	6.17	3.29	B
S_TP vs M_TP	7.41	1.22	5.72	9.10	6.08	B
T_TP vs M_TP	3.07	0.69	2.11	4.02	4.45	B
S_Al vs T_Al	7.41	1.88	4.82	10.00	3.95	B
S_Al vs M_Al	8.31	1.98	5.57	11.06	4.19	B
T_Al vs M_Al	0.90	0.50	0.21	1.59	1.80	B

Table 4.4.10 Effects of Control Policies

5. Real Time Bus Control Using Knowledge-Based Assistance

Given the overwhelming amount of information that will be generated and communicated in real-time by the AVLC system, it becomes almost impossible for human controllers to process, analyze and use the data effectively for on-line bus control. Hence, a knowledge-based decision support system becomes an interesting alternative for real-time bus control. In this chapter, we provide a general description of knowledge-based systems, describe a structure for a prototype system designed to assist in real-time control, and report on two experiments involving the prototype system, namely the automatic switching of control strategies as a function of the headway during different time periods and the testing of the message board control policy.

5.1 General Definitions and Description of Knowledge-Based Systems

The contents of this subsection is largely based on [27], [19]. A Knowledge-Based Systems (KBS) is a computer program that uses the knowledge and inference procedures of human experts to solve difficult problems. This is in contrast to a conventional computer program which is algorithmic in nature, using precisely defined, logical formulas and data. A Knowledge-Based Systems approach works well when the problem to be solved is complex or ill-defined, and when judgement and experience are useful tools in finding the solution. Real-time bus control is an area where problems are very difficult or even impossible to describe with precise mathematic models and must be solved in very short periods of time. Therefore, a Knowledge-Based System might be a good solution approach for those problems since it solves problems as a human expert would, based on the same logic, reasoning, and experiences, but more rapidly.

A typical KBS is made up of four primary components:

- **A knowledge base** containing the domain-specific facts and heuristics associated with a particular field;
- **A rule interpreter, or inference engine,** that can use the knowledge base to solve a domain-specific problem;
- **A system work area,** or a global database or work space that maintains the problem status, the input data;
- **A user interface,** which allows users to communicate with the system.

• ***The Knowledge Base***

The knowledge base contains the facts and rules that represent the expert's knowledge and experience. Facts and rules are used by the expert system to emulate the reasoning of an expert and make an intelligent decision. Knowledge can be gathered from a multitude of sources, such as interviews with experts, study of manuals or research reports, use of simulation, etc. Knowledge can be represented in several ways. The rule-based knowledge representation is the most popular method of representing domain-specific knowledge due to its simplicity. For the rule-based knowledge representation, the domain knowledge is represented as a set of rules. Each rule is made up of a condition part and an action part: IF (condition) THEN (action).

• ***Inference Engine***

The inference engine is the component of an expert system that manipulates the knowledge contained in the knowledge base to solve the problem at hand. Several different methods can be used for applying rules. The two methods commonly employed by the knowledge-based system are forward chaining (data driven) and backward chaining (goal driven).

In the first approach, the condition portions of the rules are checked against available facts to determine whether or not they are true. All the rules whose condition parts are proven to be true are fired (executed).

In backward chaining, the inference engine begins the search with a goal, or hypothesis, and works backward through the rules in an effort to evaluate whether or not all the data support the goal or hypothesis being considered.

• ***System Work Area***

The system work area is a space containing information describing the problem under study, such as user input data, the data derived through the application of rules, etc.

• ***User Interface***

The user interface acts as a vehicle through which the user and the knowledge-based system communicate with each other.

5.2 Description and Structure of the Prototype of the Knowledge-Based System

Our prototype is a "first cut" attempt at applying a knowledge-based system approach to the bus control problem. The prototype of a bus control system using a knowledge-based system approach was developed for testing on the simulation model for the following two purposes:

• As part of the knowledge acquisition process, to test different control actions and control strategies on the simulation model more effectively. We believe it will be more appropriate to separate all the bus control actions from the simulation model. We let the simulation model only simulate the bus running, road conditions and the passengers' activities, and let the knowledge-based system act as controller, guiding the running of the simulation model. Different control strategies and control actions can be implemented in the knowledge base. To test different control strategies or control actions, only the knowledge base needs to be modified not the simulation model. In this way, the integrity of the test results can be guaranteed.

• A feasibility study of such a system for on-line bus control.

The main structure of the prototype and its relations with the simulation model are shown in Figure 5.2.1. Quintus Prolog has been used as the development tool of the prototype: it provides the backward chaining mechanism, and capabilities to interface with external C programs.

We made two tests with the prototype: using different control strategies for different time periods and testing the message board control policy.

• ***Using Different Control Strategies for Different Time Periods***

Route 95 has different headway for different periods of time:

- 3 minutes headway for morning peak and afternoon peak,
- 5 minutes headway for day off peak and,
- 15 minutes headway for the evening.

As discussed earlier, it is appropriate to use different control strategies as a function of headway. This implies changing from a 5 minutes threshold headway control (between 5:00 and 7:00) to a 3 minutes threshold headway control (between 7:00 and 9:00), back to 5 minutes threshold headway control (between 9:00 and 15:00), followed by a 3 minutes threshold headway control for the PM peak (15:00 to 17:30). The control then switches to a 5 minutes threshold headway control until 19:00 and finally to schedule control from 19:00 to 0:30. As a first step, we built

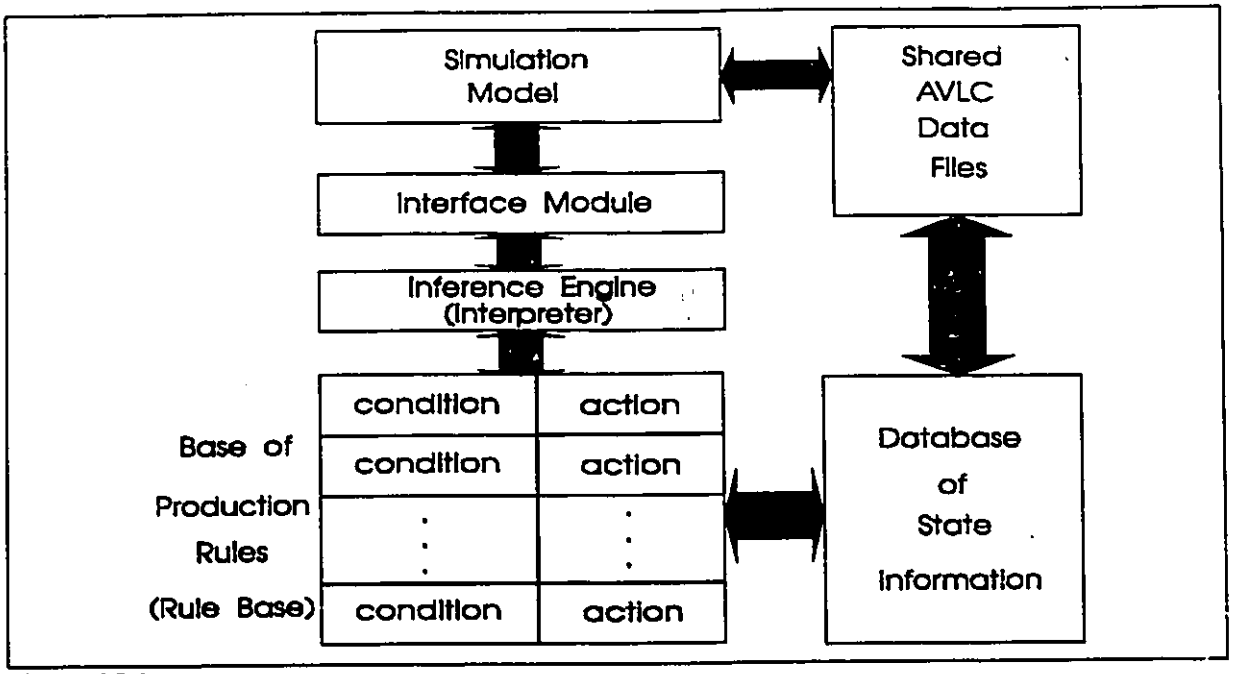


Figure 5.2.1

the prototype for controlling those switches to see whether or not the simulation program can communicate with and be controlled by the prototype.

The simulation program has been modified to interface with the prototype. Every 15 minutes, the simulation model will report to the prototype the current simulation time and the current control policy, and will receive the instruction from the prototype to maintain the current control policy or to change to a different policy. If the simulation receives an instruction to change the control policy, it will output the bus headway performance for the current control policy and reset the control parameters related to a different control policy. For example, if the instruction says to change the control policy from 3 minutes threshold headway control to 5 minutes threshold headway control, the variable that represents the threshold will be reset to 5; if the instruction specifies to change the control policy from the threshold headway control to schedule control, the flags of the control points will be set for schedule control (which means that whether or not the bus will be held at the control point will depend on the bus schedule).

The knowledge used to determine when to switch from one control strategy to another is implemented into a rule-base representation. For instance, the planned headway is a series of rules which checks the current simulation time (in seconds) then gives out the appropriate value

(the planned headway). In Prolog form, it looks like the following: (all times are expressed in seconds with time 0 being 5:00 am)

```
...
planned(Time, Headway) :-
        Time >= 25200, Time < 32400, Headway is 180, !.
planned(Time, Headway) :-
        Time >= 54000, Time < 63000, Headway is 180, !.
planned(Time, Headway) :-
        Time >= 18000, Time < 25200, Headway is 300, !.
...
switch(H, OldHdwy, OldPolicy) :-
        H < 600, OldHdwy is H, OldPolicy is 2,
        print('0\n'),told, Headway is integer(OldHdwy/60),
        format('~s~d~n',["Keep the ', Headway, " minutes headway control."]),!.
switch(H, _, _) :-
        H < 600, NewHdwy is H, Policy is 2, Headway is integer(NewHdwy/60),
        print('1\t'), print(NewHdwy), print('\t'), print(Policy), print('\n'), told,
        format('~s~d~n', ["Switch to ", Headway, " minutes headway control."]), !.
...
```

Once the prototype finds a planned headway for the current simulation time, then it checks if the planned headway is small enough to implement the headway control, etc., by doing that, the prototype will fire other related rules until it comes to a decision.

• ***Testing of the "Message Board Control" Policy***

The simulation program is modified to report the current stop and bus number, the departure time of the previous bus, and the time at which the current bus is ready to depart the stop. The prototype will operate the "message board control" policy which was described in Chapter 4, and communicate the bus departure time according to this policy to the simulation program.

The knowledge used for this control policy is implemented in two parts: the facts and the rules. The facts include the control points, the message that says to delay a bus for a certain amount of time and the records of the buses, which left the control points. In Prolog form, these facts are represented as follows:

```
...
control_point('SH935').
control_point('EE915').
...
message(BusId, DelayTm).
...
busrecd(Stop, BusId).
...
```

The "message" and the "busrecd" are defined as a *dynamic* clause in Prolog: for the "message", a new message needs to be inserted into the knowledge-base when it is generated, and an existing message needs to be erased from the knowledge-base when it has been read; for the "busrecd", every time a bus leaves one of the control points, the related "busrecd" needs to be updated. The rule part of the knowledge-base contains the rules used to decide what should be done if the current stop is a control point and the current headway is greater than the planned headway, etc. The rule-base looks like the following in Prolog form:

```
...
control(Stop, _, CurTime, PreTime) :-
        PreTime =< 0, control_point(Stop), print(CurTime), print('\n'), told, !.
control(Stop, Bus, CurTime, PreTime) :-
        control_point(Stop), clause(busrecd(Stop, PreBus), _,R),
        CurTime > PreTime + 200, X is integer((CurTime - PreTime - 180)/2),
        min(90, X, DelayTime), assert(message(PreBus, DelayTime)),
        print(CurTime), print('\n'), told,
```

```
        dispinfo(Stop, Bus, PreBus, CurTime, PreTime),
        print('Issued delay message.\n'), erase(R), assert(busrecd(Stop, Bus)),!.
control(Stop, Bus, CurTime, PreTime) :-
        control_point(Stop), CurTime > PreTime + 200,
        print(CurTime), print('\n'), told,
        dispinfo(Stop, Bus, _, CurTime, PreTime),
        print('The first bus leaving control point.\n'),
        assert(busrecd(Stop, Bus)),!.
...
```

5.3 Some Testing Results of the Prototype

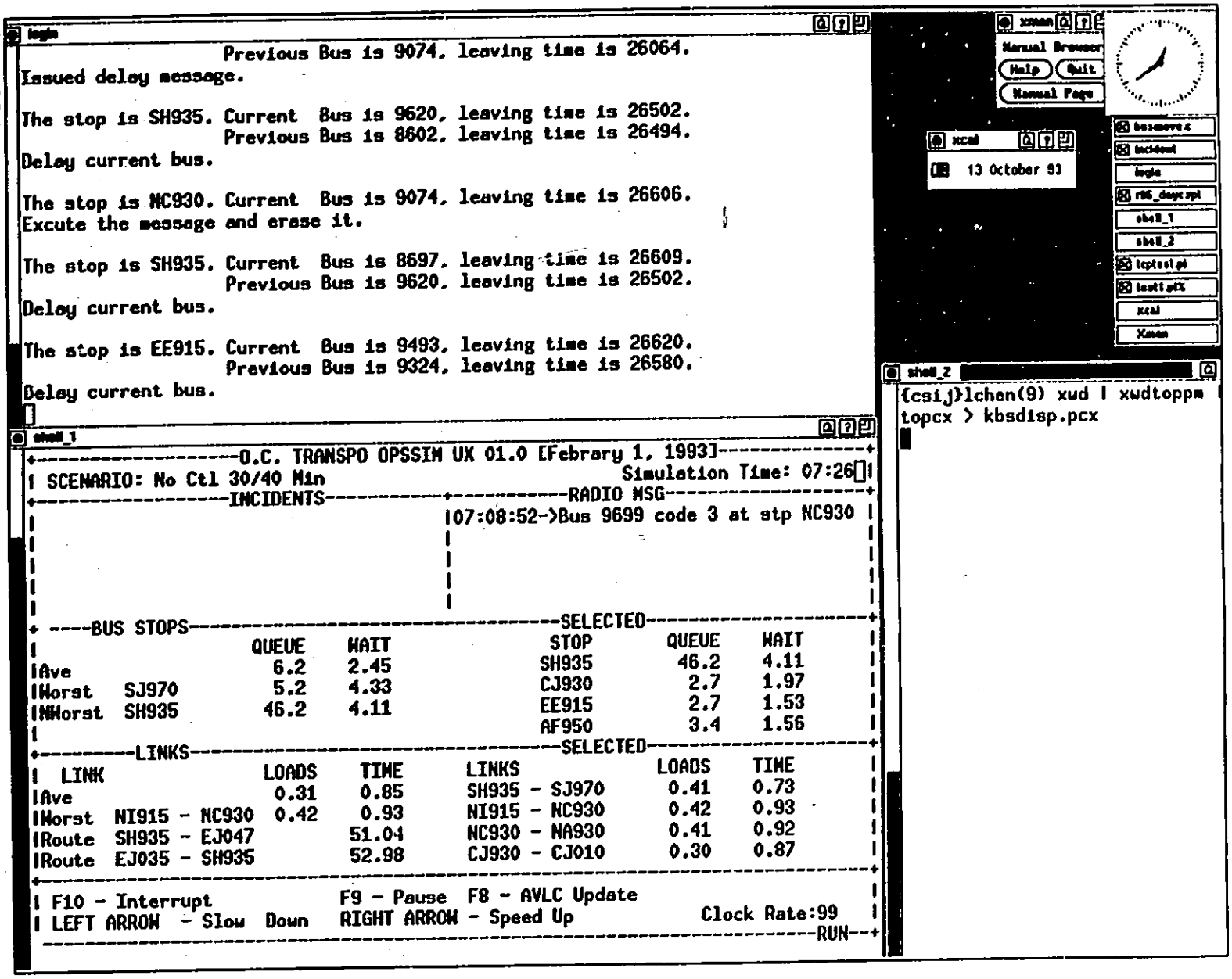
In X Window (an environment in UNIX), we first created one window to run the prototype: the prototype then waits to receive the call from the simulation. We then create another window to run the simulation. During the running of the simulation, the simulation program will issue a request to the prototype, pause the running and wait for the instructions from the prototype, receive the instructions from the prototype and then resume the running and modify the related parameters if the control policy has been changed.

We first tested dynamically switching control strategies for different time periods. The tests show that the prototype can work correctly with the simulation. However the results show that neither the schedule performance measures nor the headway performance measures improve by switching control strategy for different periods of time compared to either running the schedule control scenario or the headway control scenario for the whole day. One of the most important differences of implementing dynamic control switching occurs at transition time periods. For instance, a bus which is scheduled to run during the PM peak period and the evening period may be delayed, according to the 3-minute threshold headway control policy: that bus would then be late for its evening trips, under the schedule control policy; no holding action would be taken on that bus and bus bunching may occur. This would almost be similar to running buses under the no control scenario. Changes in the holding policies or in the real system may need to be implemented. For example, the holding policies may need to be adjusted so that there is a maximum amount of time that a bus which operates in two different control periods

is delayed. From a planning point of view, the schedule may need to be set to allow more slack (idle) time at the end of a control period for buses operating in two different control periods. Alternatively, the transit agency may consider the idea of dynamic scheduling of the buses.

We then tested the "message board control" with the prototype and compared the results with those obtained when that policy was implemented using a subroutine within the simulation program (in C code). Implementing the "Message Board" policy through our prototype provided no worse results than when this policy was implemented directly in the simulation code. Further testing will be performed to confirm how much better the results are using the prototype. However, using the prototype model as opposed to performing control actions by changing directly the simulation code has several advantages. First as suggested above, it is expected that results from external (the prototype) control will be better than the results obtained from internal (within the simulation) control, since the "message board" policy has been refined in the prototype: before executing any given message, the current headway between the bus to which the message is destined and its predecessor is re-evaluated to check if delay is still needed based on a more recent evaluation of actual headway at the time the message is read. *Second*, as described in the last Chapter, there is a restriction for implementing the control policy in the simulation program that there can be only one message at a time to be on the board, but for the prototype more than one message can be maintained. *Third*, many stops can be chosen as "message board" control points, while it is difficult to do so using the subroutine in the simulation program, simply because it will be too complex to maintain those "messages" using the subroutine. *Fourth*, with the prototype, it will be much easier to test different control strategies: this can simply be achieved by adding the related facts and rules into the prototype. The simulation program does not need to be modified and re-compiled again and again. Furthermore, more sophisticated control strategies may be experimented. For example, one can combine one of the headway control strategies with dynamic re-scheduling of the buses (where the scheduled start time for each trip is not predefined, but scheduled dynamically according to the planned headway and the current route situation) on the simulation for the peak hours. The benefits of such complex strategies may then be compared with the implementation difficulties for the real system. The snapshot of the running screen of the prototype and the simulation is shown as Figure 5.3.1.

Figure 5.3.1


Previous Bus is 9074, leaving time is 26064.
Issued delay message.
The stop is SH935. Current Bus is 9620, leaving time is 26502.
Previous Bus is 8602, leaving time is 26494.
Delay current bus.
The stop is NC930. Current Bus is 9074, leaving time is 26606.
Excute the message and erase it.
The stop is SH935. Current Bus is 8697, leaving time is 26609.
Previous Bus is 9620, leaving time is 26502.
Delay current bus.
The stop is EE915. Current Bus is 9493, leaving time is 26620.
Previous Bus is 9324, leaving time is 26580.
Delay current bus.
O.C. TRANSPO OPSSIM UX 01.0 [Febrary 1, 1993]
SCENARIO: No Ctl 30/40 Min
Simulation Time: 07:26
INCIDENTS
RADIO MSG
07:08:52->Bus 9699 code 3 at stp NC930
BUS STOPS
QUEUE WAIT
Ave 6.2 2.45
Worst SJ970 5.2 4.33
NWorst SH935 46.2 4.11
SELECTED
STOP QUEUE WAIT
SH935 46.2 4.11
CJ930 2.7 1.97
EE915 2.7 1.53
AF950 3.4 1.56
LINKS
LINK LOADS TIME
Ave 0.31 0.85
Worst NI915 - NC930 0.42 0.93
Route SH935 - EJ047 51.04
Route EJ035 - SH935 52.98
SELECTED
LINKS LOADS TIME
SH935 - SJ970 0.41 0.73
NI915 - NC930 0.42 0.93
NC930 - NA930 0.41 0.92
CJ930 - CJ010 0.30 0.87
F10 - Interrupt
F9 - Pause F8 - AVLC Update
LEFT ARROW - Slow Down
RIGHT ARROW - Speed Up
Clock Rate:99
RUN
13 October 93
{csij}lchen(9) xud | xwdtoppm
topcx > kbsdisp.pcx

6. General Conclusions and Future Research

In this thesis, we propose the use of knowledge-based systems to help service controllers to provide real-time control of buses in urban transit agencies. While the development of such a system exceeded the scope of the thesis, we nevertheless developed a prototype model to provide insight as to the usefulness and the effectiveness of such a system. In order to test this prototype, we have to rely on a simulation model used to represent the movement of buses along a specific route. A simulation model commissioned by OC Transpo and designed by CIGGT was heavily used in our experiments. The model was designed to simulate buses on OC Transpo route 95. Our first task was to validate the model. The validation results show that in most cases the model yields reasonable results compared to historical data especially for the morning peak period, but further calibrations of the model are needed. New headway control strategies were then presented and compared to other known control strategies. The goal was knowledge acquisition for future use in the knowledge-based system. The statistical tests show our new headway control policies ("Message Board" and hybrid policies) performed better than other known control strategies. Implementation issues related to the various control policies were discussed. A prototype of a knowledge-based system for on-line bus control was finally developed and tested on the simulation model. The results show the prototype to be a promising alternative for real-time bus control.

6.1 Future Research

As mentioned in Chapter 2, the simulation model which OC Transpo asked us to use in this research seems to behave at times in a manner that is somewhat counter-intuitive. Specifically, the implementation of a control action seems to affect the breakdown probabilities for busses. Our best judgment is that we can use the model to get general comparisons of the effects of alternative control policies by turning off the questionable function, i.e., bus breakdowns. Through such simulations, we believe we can establish the relative performance of various control strategies. However, the actual numbers (e.g., time, percentage of early busses, etc.) may not be too accurate. While this kind of behaviour typifies many useful simulation models, we recommend that the breakdown module of the model be modified appropriately. This

is particularly important since bus breakdowns occur on a daily basis. It is precisely for these reasons that we did not test intervention policies for breakdowns. The simulation model needs further calibration and validation to yield more accurate and reliable results compared to the real system.

Different control strategies should be examined, especially those that offer dynamic re-scheduling synchronization, and that take into account factors such as giving more importance (heavier weight) to certain stops at different times during the day.

More work is needed towards the development of the Knowledge-Based system that can provide on-line bus control assistance. More knowledge needs to be gained from both the human controllers and simulation experiments to identify which control action is most effective given certain conditions, and under what circumstances a given action is most appreciate. Models and algorithms are needed that will effectively integrate the facts (e.g. bus schedules), the decision rules (e.g. the various control actions and their merits and weaknesses) and real-time information gathered by AVLC to provide better recommendations as to the best course of action.

Rule-based reasoning is a method that can be used to represent the "common" knowledge dealing with general cases. Recently, more and more researchers have been studying the case-based reasoning mechanism as a new knowledge representation method dealing with each particular cases. Apparently, a major part of a human controller's knowledge is accumulated from his day to day experiences, i.e. accumulated from all the cases he experienced. Therefore, a case-based reasoning mechanism may provide a better way of representing a human controller's knowledge and the possible way of implementing it in the knowledge-based system needs to be explored.

References:

[1] Abkowitz, M., Lepofsky, M., "Implementing Headway-Based Reliability Control on Transit Routes", American Society of Civil Engineers, *Journal of Transportation Engineering*, Vol. 116, No. 7 (Jan./Feb. 1990), pp. 49-63.

[2] Abkowits, M., Siavin, H., Waksman, R., Englisher, L., Wilson, N., "Transit Service Reliability", TSC Urban & Regional Research Series, U.S. Dept. of Transportation, Dec. 1978

[3] Abkowitz, M., Tozzi, J., "Transit Route Characteristics and Headway-Based Reliability Control", *Transportation Research Record 1078*, Transportation Research Board, National Research Council, Washington, D.C. (1986), pp. 11-16.

[4] Andleigh, P. K., "Unix System Architecture", Prentice-Hall Inc., 1990

[5] Barnett, A., "On Controlling Randomness in Transit Operations", *Transportation Science*, Vol. 8, No. 2, (1974), pp. 102-116.

[6] Banks, J. and Carson, J. S., "Discrete-Event System Simulation", Prentice-Hall Inc., 1984

[7] Canadian Institute of Guided Ground Transportation, "Specifications for OC Transpo Transit Simulator OPSSIM version 1.0", 1992.

[8] Gault, H., "Summary of Main Conference Themes", Proc. of Intl. Conf. on Automatic Vehicle Location in Urban Transit System, Sep. 1988a

[9] Gault, H., "Looking to the Future; A panel Discussion", Proc. of Intl. Conf. on Automatic Vehicle Location in Urban Transit System, Sep. 1988b

[10] Horellow, C., Rossi, C. and Sissa, G., "An AI/Real-Time Solution for Expert Scheduling of underground Rail Traffic", Proc. Sefety and Reliability Society Symosium '89, G.B. Guy, ed., Elsevier, New York, 1989, pp. 47-57.

[11] Heti, G., "Operational Control Strategies Using AVLC; an International Survey", Proc. of Intl. Conf. on Automatic Vehicle Location in Urban Transit System, Sep. 1988

[12] Haviland, K., Salama, B., "Unix System Programming", Addison-Wesley Publishing Company, 1987

[13] Jackson, R. L., "Evaluation by Simulation of Control Strategies for a High Frequency Bus Service", U.K. Transport and Road Research Laboratory, Crowthorne, Berkshire, England, *TRRL Report 807* (1977).

[14] Koffman, D., "A Simulation of Alternative Real-Time Bus Headway Control Strategies", *Transportation Research Record 663*, Transportation Research Board, National Research Council, Washington, D.C. (1978), pp. 41-46.

[15] Kulash, D., "Routing and Scheduling in Public Transit Systems." (MIT PhD. Thesis, 1971)

[16] Koffman, J., Gault, H., "AVLC Data for Planning and Management; Information Overload? ", Proc. of Intl. Conf. on Automatic Vehicle Location in Urban Transit System, Sep. 1988

[17] Law, A. M. and Kelton, W. D., "Simulation Modeling and Analysis", McGraw Hill Inc., 1982

[18] Lamma, E. and Mello, P., "A knowledge-Based Assistant for Real-Time Planning and Recovery in Automatic Train Protection Systems," Proc. Safety and Reliability Society Symposium '89, G.B. Guy, ed., Elsevier, New York, 1989, pp. 78-91.

[19] Louis, F. C. and Roswell, A. H., "Knowledge Based Expert Systems in Transportation", *Transportation Research Record* Transportation Research Board, National Research Council, Washington, D.C. Sep. 1992

[20] Levinson, H. S., "15 Synthesis of Transit Practice; Supervision Strategies for Improved Reliability of Bus Routes", Transportation Research Board, National Research Council, Washington D.C., Sep. 1991

[21] Lévine, P., Poverol, J., "Railcar Distribution at the French Railways", IEEE Expert, vol. 5, No. 5, Oct. 1990

[22] Shields, L. R., "Quality of Service Indicators", HEE, 1978

[23] Systems Planning Dept., "APC at OC Transpo; A New Dimension", Working Paper #1, OC Transpo, Ottawa, 1986

[24] Turnquist, M. A., "Strategies for Improving Reliability of Bus Transit Service", *Transportation Research Record 818*, Transportation Research Board, National Research Council, Washington, D.C. (1981), pp. 7-13.

[25] Turnquist, M. A., Blume, S. W., "Evaluating Potential Effectiveness of Headway Control Strategies for Transit Systems", *Transportation Research Record 746*, Transportation Research Board, National Research Council, Washington, D.C. (1980), pp. 25-29.

[26] Turnquist, M. A., Bowman, L., "Control of Service Reliability in Transit Networks", Report DOT/RSPA/DPB-50/79/5, Office of University Research, U. S. Department of Transportation, (1979).

[27] Wolfgram, D. D., Dear, T. J. and Galbraith, C. S., "Expert Systems for the Technical Professional", 1987

[28] Welding, P. I., "The Instability of Close Interval Service", Operational Research Quarterly, Vol. 8, (1957), pp. 133-148.

Appendix A

The purpose of this Appendix is to provide more technical background behind the porting of the simulation from the Vax platform to a Sun work station running Unix. This will be most useful for future developpers of the simulation model. It is thus assumed that the reader of this chapter will have sufficient technical background to follow the discussion. For the interested readers of this chapter, the related information can be found in [4], [12].

1. Modifications of the functions dealing with the keyboard

For Sun "SPARC station ELC", the code sequence that represents the special keys on the related keyboard are as shown in Table A1.

Key	Code Sequence	Key	Code Sequence
F1	'\E' '[' '1' '1' '~'	F2	'\E' '[' '1' '2' '~'
F3	'\E' '[' '1' '3' '~'	F4	'\E' '[' '1' '4' '~'
F5	'\E' '[' '1' '5' '~'	F6	'\E' '[' '1' '7' '~'
F7	'\E' '[' '1' '8' '~'	F8	'\E' '[' '1' '9' '~'
F9	'\E' '[' '2' '0' '~'	F10	'\E' '[' '2' '1' '~'
Page Up	'\E' '[' '5' '~'	Page Down	'\E' '[' '6' '~'
Delete	'\127'	Return	'\10'
Up Arrow	'\E' '[' 'A' '~'	Down Arrow	'\E' '[' 'B' '~'
Right Arrow	'\E' '[' 'C' '~'	Left Arrow	'\E' '[' 'D' '~'

where '\E' means the Escape key: its ascii code is 27.

Table A1

The keyboard control actions are: check if a key is pressed, determine which key is pressed and determine if the key being pressed is a function key or not. Those types of things can be done only by a system call "ioctl(fd, request, arg)" that requests the Kernel to perform the desired actions. "ioctl()" performs a special function on the object referred to by the open

descriptor fd. Here fd is equal to 0, since for a default format, 0 means a standard input object (normally it refers to a keyboard).

The request codes for particular functions used in the program are as follows:

FIONREAD: Get the length of the keyboard buffer, if the length is equal to zero, it means there is no key been pressed. Otherwise, it gives out the number of characters that have been stored in the keyboard buffer.

FIOCGETP: To get the system data structure sgttyb, which is used for manipulation of the screen.

TIOCSETP: To reset the system data structure sgttyb.

The arg is a pointer to data to be used by the function or to be filled by the function.

The system head files <sgtty.h> and <ctype.h> should be added into the head file "machine.h" and some of the constants declared in "machine.h" referring to the definitions of the keys should be changed as follows:

```
#define     CURSOR_UP       65
#define     CURSOR_DOWN     66
#define     CURSOR_RIGHT    67
#define     CURSOR_LEFT     68
#define     PAGE_DOWN       14
#define     PAGE_UP         16
#define     F1              201
#define     F2              202
#define     F8              208
#define     F9              209
#define     F10             210
```

For different types of work stations the function-key codes are different especially for function keys F1 to F10. In the head file "machine.h" F1 to F10 are arbitrarily defined as constants 201 to 210. Once one of the function key F* (* is within 1 to 10) is recognized the program will assign the constant defined in the head file "machine.h" to the key.

In the function "BOOL KeyBoard_Hit(BOOL *IsCommand, INT2B *CharValue)", the program first uses the system call "ioctl(0, FIONREAD, &inpchar)" to check if there is no input

character in the keyboard buffer, in which case the function returns "false". If the keyboard buffer is not empty, the function reads in all the characters in the buffer consecutively and check if it is a special key such as Delete Key, Return Key, Function Key, etc. . If it is one of those special keys the function set the pointer *IsCommand to "true", otherwise set it to "false". In that case, an appropriate value will be assigned to the pointer *CharValue and the function returns "true". The detailed logic relations are shown in Figure A.1.

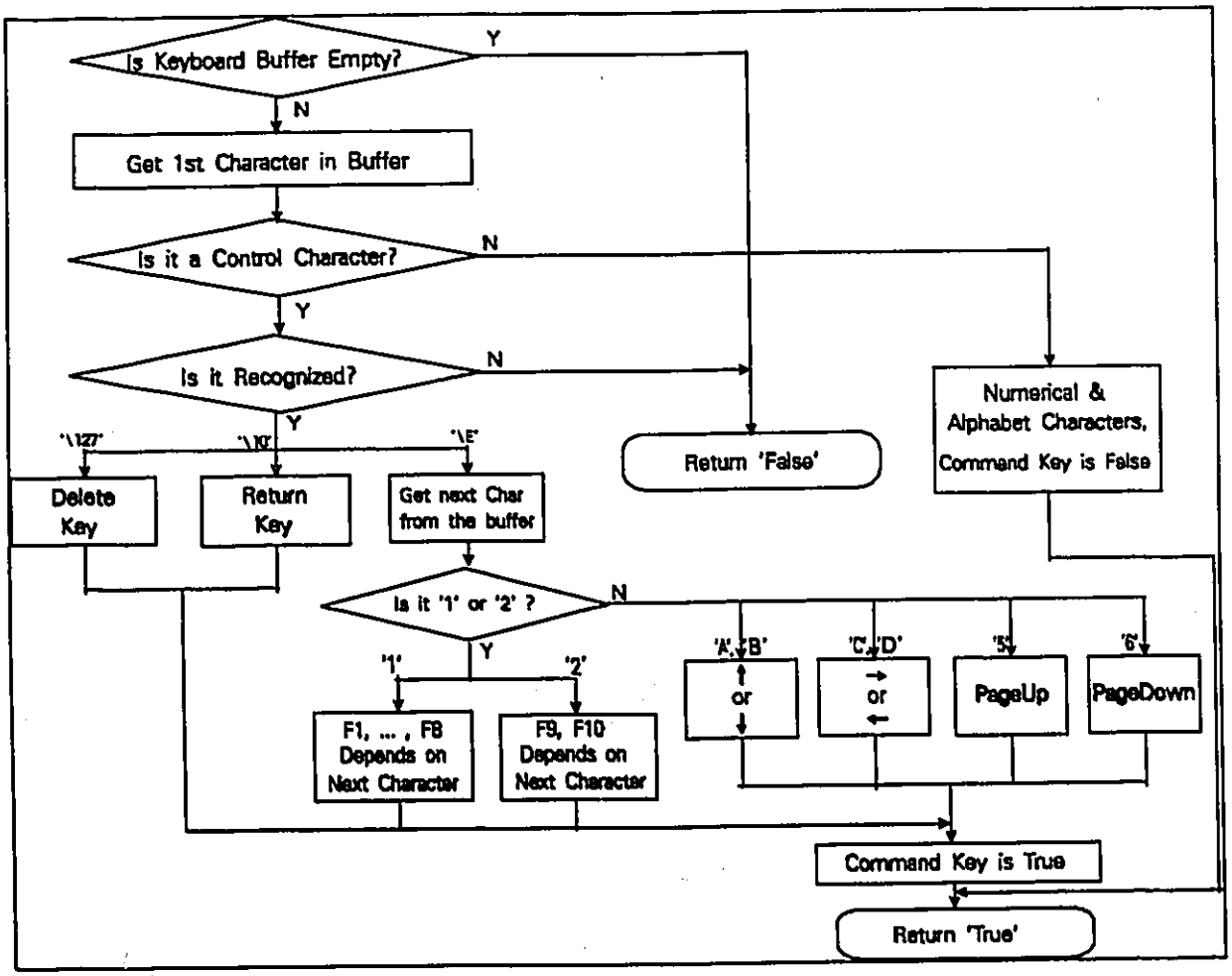


Figure A.1

2. Modifications of the functions dealing with the terminal

In the simulation model, a terminal is used for two purposes. One is to display different menu windows to allow the user to select an option as well as selections of various parameters

used to run the simulation. The other purpose is to display the summarized intermediate running results. The functions that need to be implemented are defining the screen, positioning the cursor on the screen, writting a character at the cursor position on the screen, writting a line of characters on the screen, erassing a character at the cursor position on the screen, erassing a line of characters on the screen, clearing the screen and refreshing the screen, etc.

The screen-handling tool *curses*, which comes in the form of UNIX C libraries, has been used to replace the screen-handling tool which comes in VAX libraries counterpart. "*curses*" allows the programmer to generate a data structure called a window, which corresponds to all or part of the physical screen. Characters can be 'written' into this window at any position. The system head file <curses.h>, which defines the data type related to the window, must be included in the main program "ops_main.c". The structure pointer *Win of type _win_st, where _win_st is defined in <curses.h>, must be declared in the head file "variable.h" and also be declared in the head file "external.h" as an external variable. The screen manipulations that are supported by the system and will be used in the simulation are the following:

Win *initscr() — The first routine to be called. This will determine the terminal type and initialize all *curses* data structures.

Win *endwin() — The last routine to be called before exiting the program. This routine will reset the terminal into the proper non-visual mode.

Win *newwin (nlines, ncols, begin y, begin x) — Create and return a pointer to a new window with the given member of lines (or rows) nlines, and columns, ncols. The upper left corner of the window is at line begin y, column begin x.

wrefresh(Win)0 — This routine must be called to write output to the terminal. It copies the named window to the physical terminal screen, taking into account what is already there in order to minimize the amount of information that's sent to the terminal.

wclear(Win) — This routine insures that the screen will be cleared completely on the next call to wrefresh() for that named window, and repainted from scratch.

wmove(Win, y, x) The cursor associated with the named window is moved to line (row) y, column x. This does not move the physical cursor of the terminal until wrefresh() is called. The position specified is relative to the upper left corner of the window, which is (0, 0).

waddstr(Win, str) This routine writes all the characters of the null-terminated character string str on the named window.

Initializing the Screen

The terminal screen is initialized and defined in the function "VOID OpenVirtDisp(VOID)" as follows:

```
initscr();
Win = newwin(24, 80, 0, 0);
```

where the size of the screen is defined as 80 (columns) X 24 (rows).

Writing characters and strings

Writing a character to a specified position on the screen is done by first calling the routine "wmove(Win, y, x)" to move the cursor to the position y (row), x (column), then writing the string (a string ended with a null character) to the screen, and finally calling the routine "wrefresh(Win)" to display the character on the physical terminal. To erase a character at a specified position on the screen involves the same procedure as writing a character on the screen but letting the character be a blank.

Mode Setting

The terminal can be put into and out of CBREAK mode. In CBREAK mode, characters typed by the user are immediately available to the program and erase/kill character processing is not performed. When in NOCBREAK mode, the tty driver will buffer characters typed until a NEWLINE or RETURN is typed. Interrupt and flow-control characters are unaffected by this mode.

At the beginning of the program, the terminal should be set to the "echo off" mode by calling "ioctl(0, TIOCSETP, &my_sgb)" in the function "VOID OpenKeyBoard(VOID)". That will enable the user's pressed keys to be buffered and processed separately. Before exiting the program, the terminal should be set back to echo on mode and the routine "endwin()" must be called to restore the original screen. Those functions will be realized in the subroutine "VOID

ErrorMsg(BYTE cModule[], BYTE cErrorDesc[], INT2B nErrorType)". One case of JOBFIN is added to the subroutine to handle the case where the simulation job is finished and the terminal should be restored. If a fatal error occurs in the simulation or the simulation run is completed, the main program will call this subroutine to finish the program properly.

Every time one of those screen manipulation routines is called, the terminal should be set to NOCBREAK mode to enable those screen manipulation routines to execute properly. After calling those routines, the terminal should be set back to CBREAK mode to enable the user's pressed keys to be accepted by the program immediately and processed accordingly.

3. Modifications of the file manipulations

In order to run the simulation, one of the scenario files, which defines the scenario to be simulated, must be loaded into the program. The original program displays the name of those scenario files one by one to let the user choose one of those listed scenario files to be loaded into the program by calling the VAX library functions to resolve the wild files that match the name "r95_day*.dat". Since there is no similar functions supported by the UNIX system, the names of the scenario files must be stored in a separate file "scenfile.name" before the running of the simulation, which will be read in the function "struct sFileList *SelectFile(BOOL bSaveFile)".