BusNet - A Prototype Implementation of an Intelligent Transportation System for Winnipeg Urban Transit

by
Ghavam Fayyazi

An M.Sc. Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg,
Manitoba, Canada

Thesis Advisor: R. D. McLeod, Ph.D., P.Eng.

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

BUSNET - A PROTOTYPE IMPLEMENTATION OF AN INTELLIGENT TRANSPORTATION SYSTEM FOR WINNIPEG URBAN TRANSIT

BY

GHAVAM FAYYAZI

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

To Winnipeg

ACKNOWLEDGMENTS

With special thanks:

To *Dr. R. D. McLeod,* for his professional advise as well as support that enabled me both to learn and to write what is presented herein.

To my wife *Firouzeh* for her sacrifices and understanding.

Table of Contents

- Abstract

Intelligent Transportation Systems (ITS) are being used in many cities of the world. In this project, as a subset of such a system for the city of Winnipeg, an instance of an Advanced Traveler Information System (ATIS) is introduced and the implementation of a prototype of it is described. Such a system could be used to access to the real-time information of the operative buses of the city transit system. Using low price GPS receivers, specific computer applications, and wireless communication/ telecommunication systems capabilities, as well as, the Bluetooth wireless technology a positioning system is designed and employed to transfer the real-time information of the locations of the operative buses to a database. The information gathered in the database is provided to the end users connected to the Internet. The information is also provided to be used by the transit system monitoring and management centre. Fuzzy logic concepts in reporting the real-time positioning information and Neural Network methods in creating transit system timetables are discussed. The system architecture, following Canadian and US national ITS architecture, is introduced and closely investigated. Such a scaleable system could be a part of an integrated Winnipeg Transportation Management Center (TMC) in future.

Section 1 - Introduction

1- Introduction

Recent advances in computer and communication technologies have greatly enhanced our ability to manage transportation systems on an area-wide basis. Typical aspects of this capability are the monitoring of traffic flow, remotely adjusting the timing of traffic signals, detecting incidents and dispatching emergency equipment, tracking the movement of transit vehicles and communicating with their operators, monitoring the capacity of parking facilities, and issuing advisories to the traveling public.

Providing people with timely and accurate traveling information, as well as, improving the quality of it have been prime objectives of Intelligent Transportation Systems (ITS). Real-time information about traffic conditions and travelling options help travelers make more informed decisions about routes, travel modes, and times of departures. Such information provides passengers with faster commutes and more efficient transportation systems.

In this project, an ITS system is introduced and implemented, as a prototype, in which the real-time positioning information of the buses of Winnipeg Transit System are gathered on a database and accessed by users connected to the Internet. The main idea behind of this project, at the time of proposing, was to use the capabilities of evolving GPS, communication, and telecommunication technologies to provide the required building blocks for developing such an ITS system for the city of Winnipeg.

Figure 1-1 illustrates the various components of the suggested system from a general point of view. The figure is not a precise technical depiction of the project, named BusNet, and neither does it reflect all components involved in defining and/or implementing of the system. A detailed depiction of the system components, and their interaction with each other, will be presented and discussed in a later section when the system architecture is investigated. The figure, however, provides a general idea about the project and supports a visual description of it in an easily

understandable way. The figure shows the system as it was thought of, and suggested at its very early steps of proposing, as sketched on a white board.

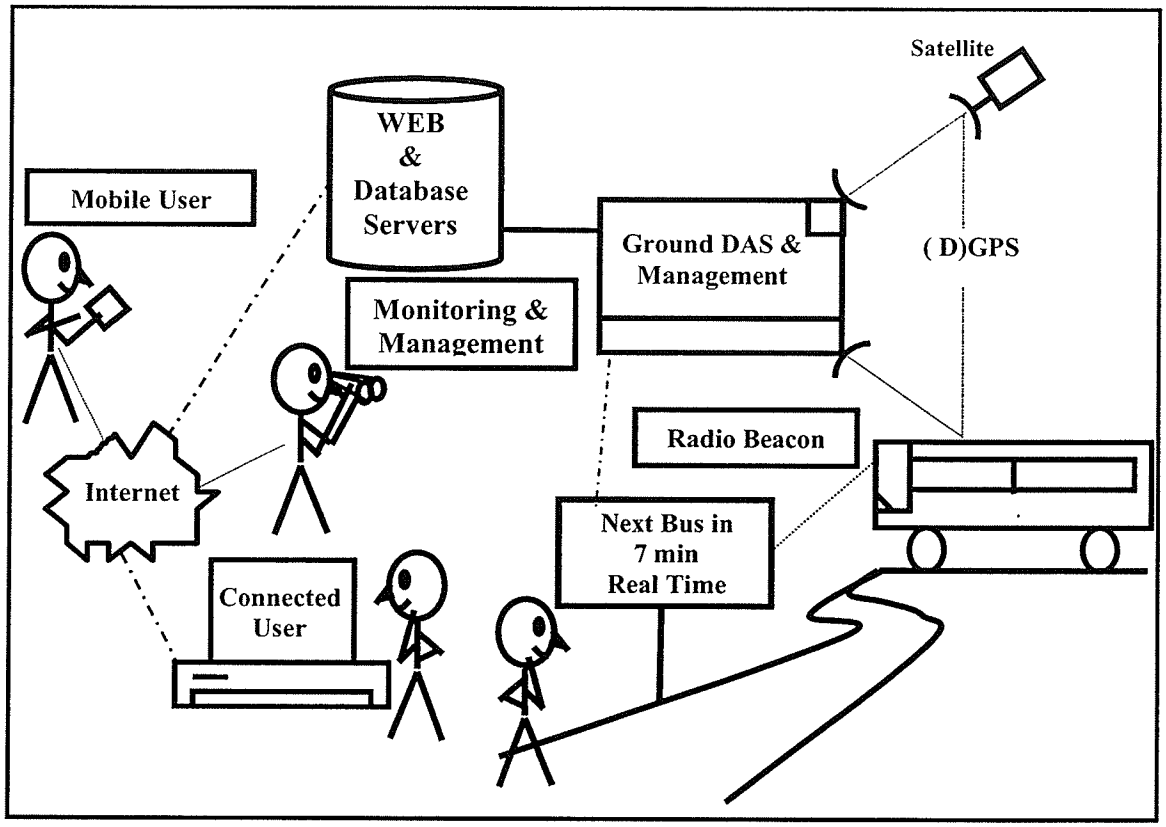


Figure 1-1 – General system layout of this project.

It is presented here, to support an introductory view over the system, as it historically did when the first part of the project was proposed.

The remaining part of this introduction overviews what this project provides for the system users and where more rigorous definition, explanation and implementation of its components can be found.

The next section, Section 2, reviews the present status and the future plans of the Winnipeg transit system, to provide a general background about a typical transit system as a model, based on which the present ITS system is designed. Section 3 discusses Intelligent Transportation Systems (ITS) in general, its present situation in Canada and around the world. It also describes the project

system architecture from the Canadian ITS Architecture point of view. In Section 4 an explanation about positioning systems employed in ITS systems is presented. Global Positioning System (GPS), Differential GPS (DGPS) are explained, and the type of DGPS used in this project are introduced. Section 5 discusses the mapping techniques employed in this project. In Section 6, the development of an application that simulates the behavior of a GPS receiver, installed in an operative transit bus, is explained. Section 7 is devoted to GPS receiver output analysis, various auxiliary-positioning methods deployed in this project, and fuzzy logic concepts implemented for making decisions about the reporting time of the detected out-of-schedule positioning events. A light prototype of a central monitoring unit is also introduced in the same section. Section 8 discusses how Bluetooth devices can be used as a part of a positioning system and the way that they could help in the interpretation of the on-board GPS receiver's output. Section 8 also explains how the Bluetooth scatternet topology might be employed to make a new positioning system, substituting GPS receivers in this project. Section 9 discusses accessing the system database over the Internet and developing the corresponding Internet applications. Section 10 describes, in detail, an application developed for PDA devices, equipped with Palm OS 3.5 or higher, which provides a mobile user of the transit system with the real time information of it. Using Kalman filters in anticipating bus arrival times to bus stops, employing Neural Networks in creating and/or modifying the system's timetables, and implementing the system's hardware are all briefly discussed in the Section 11 which also concludes the thesis. The application codes developed for the project are all available electronically.

Section 2 – Winnipeg Transit System

2-1 Winnipeg Transit System at a Glance

2-2 Recommended Improvements

2-3 Recommendations Followed in this Project

2-4 References

2– Winnipeg Transit System

Traditionally, a public transit system is well recognized for its many benefits such as reducing traffic congestion, noise, air pollution, usage of fossil fuels, and reducing the need to invest in a city's roadway infrastructure. Public transit systems are also space and cost efficient, and can carry many passengers at the same time, particularly important at rush hours in crowded areas of a downtown. A transit system is designed to accommodate peak travel demands, such as patterned trips to work places and/or educational institutes. A car carries an average of 1.2 persons in rush hours, a bus carries as many passengers as around fifty cars [1]. Public transit also enhances general public safety through encouraging more pedestrian activity, and providing safer public spaces. It improves downtown economical development through expanding the market for businesses. Transit systems help create more pedestrian friendly downtowns and therefore better urban environmental conditions. In general the life quality in a transit oriented downtown is higher than that of an automobile oriented one.

However, to realize all the mentioned benefits of a transit system, the level of ridership should be enhanced as much as possible. This goal is reached through creating more supportive and attractive conditions for the transit users. In fact, the quality of the transit service is among the most important factors affecting ridership. The following specifications may be enumerated as important parameters in the overall transit service quality:

- Coverage — proximity of bus stops to residents and major destinations
- Comfort — well maintained, clean bus stops and buses; safe operation of buses
- Travel Time — traveling as fast as cars; buses operated at desired frequencies
- Reliability — on time services; operating all scheduled trips
- Convenience — readily available route and schedule information; current and easy to understand posted information

In the following subsections, some facts about the Winnipeg public transit system are introduced, and recommendations for enhancing the level of ridership of the system, made by a task force employed by Winnipeg City Council, are looked at. Finally we investigate briefly how this project may help achieve some goals highlighted in the recommendations.

2-1 Winnipeg Transit System at a Glance

Working 365 days a year and more than 18 hours a day, Winnipeg Transit System operates 68 fixed routes throughout the city of Winnipeg - with the population of 630,000 (estimated for years 2001-2 [3]) [2].

In general 60% of Winnipeg population use the transit system. However, only 20% of users are heavily dependent on the system, making more than 6 trips per week. The other 80% of the users rank between moderate to occasional user groups, making from 3-6 trips to less than 1 trip per week. These facts show that there must be more effort to attract people in the group of moderate usage or less to the group of people with higher level of system usage. Further analysis of ridership also indicates that the most frequent users are among teenagers and younger adults rather than older ones. More than 50% to 60% of people between 15 to 24 years old are transit system users. One third of females of age 25 and higher are transit users. However, the rate of transit usage decreases by age in males of the same range of age.

Regarding the types of the trips made by the Winnipegers, the statistics show that about 14% of transit users are completely dependent on transit system for all their trips. About 11% of the passengers use the transit system for their work/study trips and 22% of them use it for their trips of other purposes.

The statistics also show that about one third of the employees of the down town area, and the same proportion of the students of the university of Manitoba and Red River College use transit system to reach to their destinations. At the university of Winnipeg 60% of the total number of students use the city transit systems.

In 1998, there were 39 million Revenue Passengers who paid for the trip in cash, ticket, or pass fairs when boarding the transit system. There were also 14 million boarding passengers who use transfer slips when boarding the transit system, indicating that almost 30% of passengers use more than one bus route to get their destination in a trip. Statistics also show that less than 50% of the city population are within 35 minutes time-distance from Winnipeg downtown, who travel by transit system. This makes the transit system a less favorable transportation option for the people far from the city center.

Winnipeg transit system spends almost 1.3 million bus hours per year on the area it covers.

During 1990's the cost of regular transit fare increased at a rate much higher than the inflation rate. Nevertheless, only 67% of the costs of Winnipeg transit system spent on regular operations.

The following major resources support the operation of transit systems of the city of Winnipeg:

- Passenger fares
- Advertising on buses, shelters and benches
- Province of Manitoba and City of Winnipeg Operating Grants

The Innovation Transit Program, started in the early 1980s, contributes $500,000 to $600,000, annually, to make capital improvements to the transit system. The main resource of the program is the City's Capital Estimates.

2-2 Recommended Improvements

In December 1998, a working group was established to recommend outlines of long-term frameworks for a better Winnipeg public transit system for consideration by the City Council. The working group reviewed the factors influencing the transit ridership and financial performance of the system.

Submitted in May 1999, the working group recognized the following objectives, among others, for the Winnipeg's fundamental urban transportation system;

- To provide affordable and accessible transportation system for effective and efficient mobility persons and goods,
- To reduce traffic congestion through providing sufficient capacity in the transportation infrastructure,
- To reduce air pollution related to urban transportation,
- To reduce energy use related to urban transportation.

To provide better public transit services, the committee suggested that the following supportive developments be accomplished in the system to ensure attracting more users to the system, and achieving a higher level of ridership;

- Make ongoing improvements to the service
- Make the service more affordable, reliable, comfortable, safe and accessible
- Provide the user with more transit information
- Make a commitment to high speed transit
- Make the service more productive
- Make the system easier to use

More specifically, as far as this project is concerned, the last two recommendations can be exemplified as follows:

- Make the service more productive
  - Upgrade the present two-way radio system, installed in 1982, to make Automatic Vehicle Location (AVL) possible – planned for realizing by 2005.
- Make the system easier to use
  - Timetable and map improvements
  - More information posted at bus stops
  - Automated trip planning service
  - Real-time schedule displays – planned for realizing after 2005

- Further develop of the transit system web site
- Next stop display onboard – planned for realizing by 2005

To learn more about the Winnipeg transit system and its future plans references [4,5] may be consulted.

2-3 Recommendations Followed in this Project

In this project some of the recommendations mentioned in the previous subsection have been prototyped. More specifically, this project presents methods to post the real-time location information of the transit buses to the end users connected to the Internet at bus stops or elsewhere. Therefore, the transit system scheduler information will be easily accessible and more reliable, as it is augmented and updated with real-time data. This project also suggests replacing the present two-way radio being used in the transit system, by a data network that would be improved in performance. This would not only make the AVL service possible but also provide a low cost infrastructure facilitating data communication to the operative buses of the transit system. It could provide passengers and drivers with the real-time data of road conditions, as well as, commercial and advertising announcements. It would also provide a better monitoring capability facilitating the application of more efficient management methods.

This project gives an underlying structure, on top of which more powerful and advance ITS components, such as an automated trip planning system, might be built in the future.

2-4 References

1. Winnipeg Transit system, "*Transit Facts and Reports*", Copyright 1999 City of Winnipeg Transit System, http://www.winnipegtransit.com/OtherPages/facts.html
2. Winnipeg Transit system, "*Public Transportation Services Provided*", Copyright 1999 City of Winnipeg Transit System, http://www.winnipegtransit.com/OtherPages/services.html

3. CAO Secretariat, Statistics Canada, "*Population for City of Winnipeg – Estimated Population in November 2001*", March 6, 2002, http://www.city.winnipeg.mb.ca/interhom/pdfs/about_winnipeg/profile/population.pdf
4. P. De Smedt, J. Eadie, J. Gerbasi, M. Lobusch, and R. Borland, "*The Guide to Better Transit for Winnipeg*", Final Ropert, January 7, 2000. http://www.winnipegtransit.com/OtherPages/transitReport.pdf
5. City of Winnipeg Transit System, Copyright 1999 City of Winnipeg Transit System, http://www.winnipegtransit.com/

Section 3 - System Architecture

3- The System Architecture

There are great efforts around the world to bring intelligence to transportation systems. The main goals are getting more out of the present transportation systems by applying system optimization methods offered by integration of emerging technologies in the fields such as microelectronics, communications, computer, and information processing. So-called Intelligent Transportation Systems (ITS) are designed to provide their end users with more efficiency and safety, and the environment with less pollution and waste of resources. These methods enhance the capacities of the transportation systems without requiring expanding their infrastructure. ITS systems bring the interaction of the three major transport systems' components: the user, vehicle and infrastructure intelligently in coordination with each other. The annual world market for ITS industry is estimated to be $90 billion by 2011 [1]. Interested readers may consult references [6-17] presented in the "Reference" sub-section, below, to learn more about the current ITS research and development programs around the world.

In this project, as a prototype model of an advance public transportation system, the real-time and scheduler locations of the buses of the Winnipeg Transit System are provided on the maps of desired routes to the end users connected to the Internet. In addition, the estimated arrival time of buses to selected bus stops would be calculated and announced upon a user's request.

The implementation of two examples of ITS sub-systems called Advance Traveler Information Systems (ATIS) and Advance Traveler Monitoring Systems (ATMS) are partly discussed. In a broad sense, the mentioned systems inform travelers about road conditions, probable congestion, best routes to take to get a particular destinations, etc. The prototype discussed here, however, is designed to give the traveler the real-time monitoring capability of the transit system through which the road conditions and its congestion or delay status can be inferred. The transit system management and control unit may also use the same information to gain higher level of operational and cost efficiency.

BusNet, as the chosen name for this project, will be used it in this discussion, wherever appropriate. It is a dynamic bus information system that as mentioned above gives its users the capability of monitoring the present locations of buses that are in service and/or the prediction of their arrival time to desired bus stops. Therefore, it relies mainly on the employed positioning system. Generally speaking, the positioning system is a major component of an Intelligent Transportation System (ITS). It is responsible for measuring the position of the vehicles (moving platforms) of any transportation system including cars, buses, trucks, trains, etc. The positioning system may be constituted of more than one subsystem. There might also be other components in an ITS system such as a time estimation unit, statistical and scheduler information database, communication facilities and various hardware or software units.

An ITS system designer should follow certain rules in designing the system architecture to co-ordinate of the various components in the system. Establishing and following widely accepted system architectures ensures compatibility among deployed ITS technologies created or used by private and public sectors. The latter has a major role in directing the engineering and management efforts toward creating more efficient designs in terms of system cost and unit price, among other goals. Depending on how an architecture describes the components and their relationship in an ITS system it may be categorized as logical or physical architecture. While a logical architecture determines the data flow and its processing between the system logical components, the physical architecture identifies the physical entities used in the system and their interconnection.

3-1 ITS Architecture

Among the ongoing efforts to define ITS architectures in various countries and regions of the world we select, develop, and discuss the BusNet system architecture in accordance with those suggested under the following titles:

- "ITS Architecture for Canada release 1.1" last updated in August 2002 [18]

- "The United States National Architecture US-ITS version 4" released in March 2002 [5].

The US-ITS architecture is among the first announced and most comprehensive architectures in the world that may be followed, more or less as a reference by the corresponding decision-maker institutes around the world. It is consistent with the ITS architecture in a Canadian context, mainly because the Canadian ITS architecture largely reflects U.S. standards [3]. Such a consistency guarantees cross border compatibility, and Canada's competitiveness in the ITS industry in the international marketplace. To be acquainted with Canadian ITS Architecture references [2-4] and [18-19] may be consulted.

This section of thesis was first developed based on US-ITS Architecture, simply because Canadian ITS Architecture was not available at the time of preparing this section. However, at the time that the section was nearing completion the Canadian ITS Architecture was released, and fortunately found to be very close to US-ITS Architecture, as far as this project is concerned. Therefore, in the following discussion we use term "ITS-Architecture" and by which we refer to "Canadian ITS Architecture" and "US-ITS Architecture" synonymously.

The ITS-Architecture is the result of a 5-year-old period contribution of a great number of ITS community members including system developers, technology specialists, consultants, and transportation practitioners. It gives general frameworks for planning, defining, and integrating Intelligent Transportation Systems. As such, it suggests a broad ITS system scope that will help the developers in their system design. As a general guidance, it also shows how a project may fit into a larger regional transportation system context. More specifically, the architecture defines:

- The functions and services that an ITS system is required to perform or provide, respectively, to do a specific job,
- The physical entities or subsystems capable of performing the functions or providing the services,

- The logical interconnections of the physical subsystems and the way that information flows between them in the integrated system.

Therefore, the following concepts and terms, as offered in the ITS-Architecture, could be of great use in every system under development.

- User Service and User Service Requirements
- Logical Architecture
- Physical Architecture
- Equipment Packages
- Market Packages

In the subsequent sections definitions of the above concepts and their application to the case of this project will be reviewed and closely investigated. Finally, the implementation of the components of this project will be discussed in detail.

3-1-1 User Services and User Service Requirements

User services represent the services that the system will provide for a system operator or the public, as the user.

The ITS-Architecture defines a total of 31 user services grouped in 7 major categories. Table 3-1 below shows user services as defined in the architecture [1,5].

The concept of user services lets a project or system definition start with identifying the services that the system should provide to address certain problems, or fulfil specified needs, from a high-level service point of view.

User Service Bundle	User Service
• Travel and Traffic Management	• Pre- trip Travel Information • En-route Driver Information • Route Guidance • Ride Matching and Reservation

	• Traveler Services Information • Traffic Control • Incident Management • Travel Demand Management • Emissions Testing and Mitigation • Highway-rail Intersection
• Public Transportation Management	• Public Transpiration Management • En-route Transit Information • Personalized public Transit • Public Travel Security
• Electronic Payment	• Electronic Payment Services
• Commercial Vehicle operations	• Commercial Vehicle Electronic Clearance • Automated Roadside Safety Inspection • On-board Safety monitoring • Commercial Vehicle Administrative Processes • Hazardous Material Incident Response • Commercial Fleet Management
• Emergency Management	• Emergency Notification and Personal Security • Emergency Vehicle Management
• Advanced Vehicle Safety Systems	• Longitudinal Collision Avoidance • Lateral Collision Avoidance • Intersection Collision Avoidance

	• Vision Enhancement for Crash avoidance • Safety Readiness • Pre-crash Restrain deployment • Automated Vehicle Operation
• Information Management	• Archived Data Function

Table 3-1 – User Services as defined by ITS-Architecture.

Among all user services represented in Table 3-1, this project is an ITS system prototype implementing the following user services:

- Pre-trip Travel Information
- Traveler Services Information
- En-route Transit Information

It also supports providing the following user services in future;

- Public Transportation Management
- Commercial Fleet Management

In the ITS-Architecture each user service is broken down to more detailed functional statements called user service requirements. User service requirements represent the functions that should be accomplished by a user service.

In fact, these requirements may be used as the starting points to assert descriptive statements about functional specifications of the system. Considering the User Service Requirements presented in ITS-Architecture the following functional blocks pertaining to the user services provided by this project can be introduced.

3-1-1-1 Pre–Trip Travel Information

This project provides a Pre-Trip Travel Information (PTTI) capability for the user to make pre trip decisions. The functions implemented in the system provide the user with the access to the current situation information. The information includes the arrival time of four buses to a desired

bus stop on a real as well as scheduled time basis. The user has the capability of accessing the information from wherever the Internet connectivity may be provided.

The PTTI also gives the mobile users who have connected PDAs, equipped by Palm OS 3.5 or higher operating systems, almost the same accessibility. As a matter of fact, the limited processing power of PDAs and their usage purposes make providing the same functionality as available through a PC, unnecessary.

3-1-1-2 Traveler Services Information

Since the information is provided to the user via Internet the appropriate links can easily be presented to the user with the latest conditions of the following items of interest:

- Weather
- Road accidents, incidents, construction
- Transit fare
- Availability of the present and alternative transit routes
- Changes in bus schedules
- Transit transfer suggestions and options
- Ride match services

Having the real-time data of the transit system available, and by using appropriate analysis and implementation methods, the following services could easily be provided for the system user:

- Current speeds on specific routes
- Parking conditions in key areas
- Trip planning services
- Calculated trip itinerary
- Traveler Services Information

Since the services of this project in bus stops are provided to the transit users via Internet, it is also possible to provide the users with more facilities such as area maps, closest shopping, food

and tourist centers, local library, police and hospital locations. As mentioned in the previous sub-section the real-time information is also available for mobile travelers having connected PDAs.

3-1-1-3 En-route Transit Information

In this project, the time difference between the real-time position of the bus and its scheduler location is calculated and reported to the bus driver. This service informs the driver about the amount of time that the bus is behind or ahead of its schedule. The driver can adjust the bus real location with where it is supposed to be, accordingly.

3-1-1-4 Other User Services

Since continuous monitoring of a system is an important part of its management and this project provides the real-time information about the moving platforms (buses) of the transit system, the provided information can also be used for the system management. In other words, this project provides the required structure on top of which the Public Transportation Management and Commercial Fleet Management can easily be built.

3-1-2 Logical Architecture

The logical architecture defines what should be done to support the user services and/or fulfill the user service requirements. It can be shown as diagrams consisting of the functional entities that gather and process information, as well as, provide the desired services. The architecture also shows the way that the information flows between functional blocks.

The logical architecture is independent from technology and/or the way that the functions should be implemented. That is why the logical architecture, or so-called the "Essential Model", supports a wide range of system designs and permits scalability from small size designs to large size regional systems. Figure 3-1 shows the general logical architecture of this project. In Figure 3-1 the central circle (called process) includes the set of functions or activities that the system has been designed for. "Manage ITS .0" is the main process defined in ITS Logical-Architecture (the name used for the logical part of the ITS-Architecture). All other processes are defined within this

process. The names of the processes and their numbers, as presented throughout in this thesis, reflect exactly the way that they are recognized in the ITS Logical-Architecture. The arrows illustrate the information flow within the diagram.

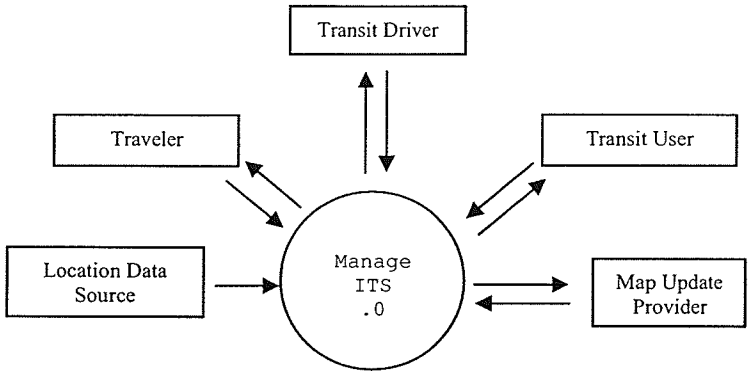


Figure 3-1 - High level general architecture of this project.

In this bubble (circle) chart of services each bubble decomposes further to other bubble charts conveying more detailed user services or functions. The decomposition process continues until the lowest functional blocks are reached which can not be decomposed any further. Finally, in Figure 3-1 the rectangles represent the terminators of the architecture. Terminators are boundaries of the architecture and represent people, systems or environment interfacing to the ITS system. As indicated in the figure, in a general logical architecture the processes communicate with each other and terminators using data flows.

In the following subsections we talk more about the inside of the process 0. After describing the logical architecture of this project, we will define and determine the corresponding physical architecture, and discuss the details of implementation of the hardware and software components used in this project, later, in this and the subsequent sections.

In the following discussion, wherever there is a reference to the architecture or its specifications, it is the architecture and the specifications used "in this project". Therefore, the phrase of "in this

project" is not required and can be dropped from the statements wherever there is not a fear of ambiguity and the subject is clearly distinguishable.

Adopting the same structure used in ITS-Architecture, the "Manage ITS .0" in Figure 3-1 is broken down to processes having higher level of specialty in terms of the functions and services that they provide. More specifically, the process can be decomposed further to three major processes "Manage Traffic 0.1", "Manage Transit 0.4" and "Provide Driver and Traveler services 0.6", As indicated in Figure 3-2.

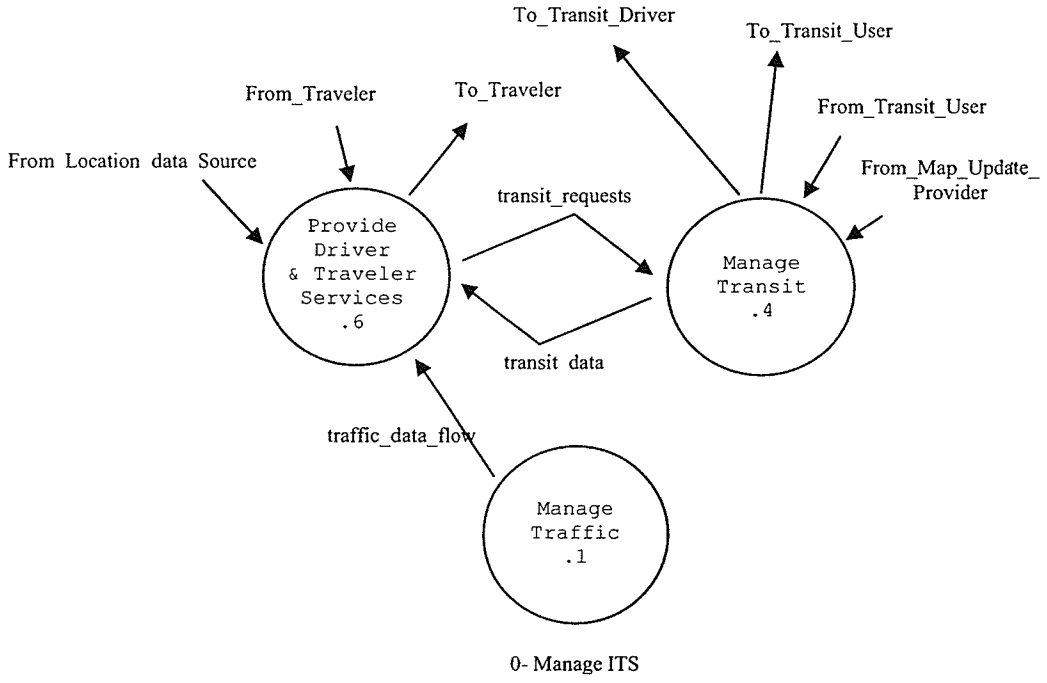


Figure 3-2 - Three major processes constituting the "Manage ITS .0" process, as implemented in this project.

"Manage Traffic 0.1" helps in determining the presence of transit vehicles in the vicinity of bus stops. It is used, as an auxiliary method, to locate buses of transit system in places where the output accuracy of the GPS receivers installed in buses are not promising. GPS is the main positioning device used in this prototype, however, its output might drift from the correct value because of the existences of many error sources detailed in the next section. Since the functions

provided by this process are connected to the operations of other major processes, a more detailed description of it is postponed to a later time when other processes are fully described. "Manage Transit .4" provides all services related to transit vehicle data gathering and processing. The functions performed by this process include scheduler consistency verification, time of arrival estimation, ultimate positioning determination, and information dispensation.

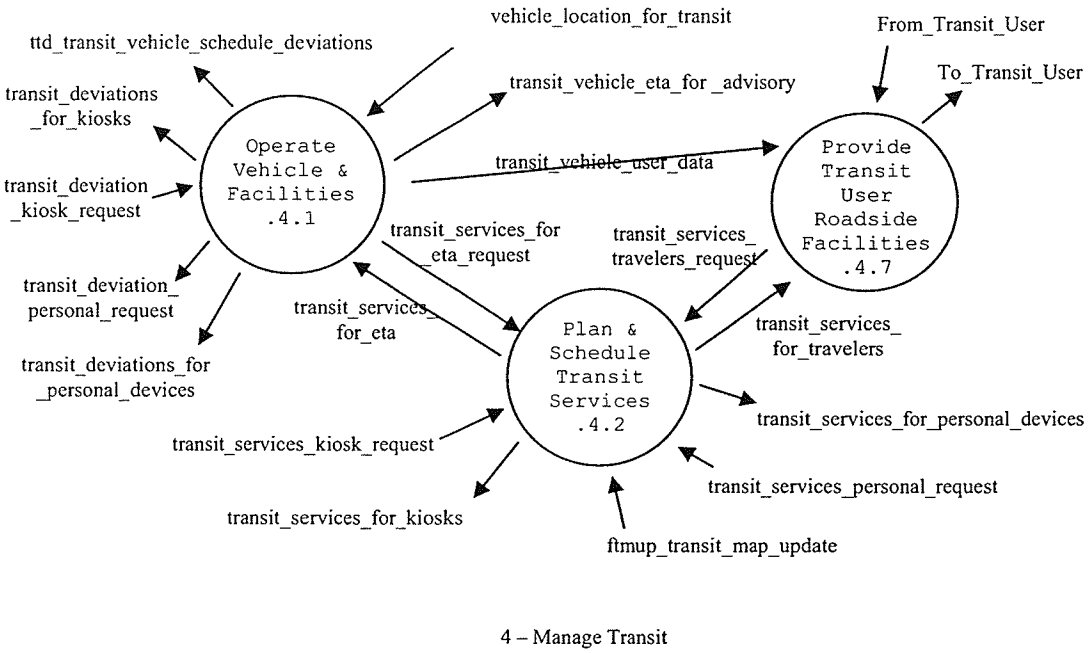


Figure 3-3 - The processes within the process "Manage Transit .4".

On the other hand, the "Provide Driver and Traveler Services .6", shown in Figure 3-2, is the process that provides drivers and travelers with the required facilities to access to the information gathered through the services of the process "Manage Transit 0.4".

The latter process can be decomposed further into more specific processes as indicated in Figure 3-3. The process "Operate Transit vehicles and Facilities .4.1", shown in the figure, provides services related to vehicle real-time positioning and time of arrival estimation. More specifically it can be decomposed to the processes shown in Figure 3-4.

In this figure, process .4.1.2 receives data flow input "transit_vehicle_locatioin_for_eta" from process "Provide Transit Vehicle Location Data .4.1.3". The latter process receives "vehicle_location_for_transit" data flow from "Provide Vehicle Location Data .6.7.2.2" process. The process .6.7.2.2 will be described in more details when the process .6, and its constituents, are considered later on.

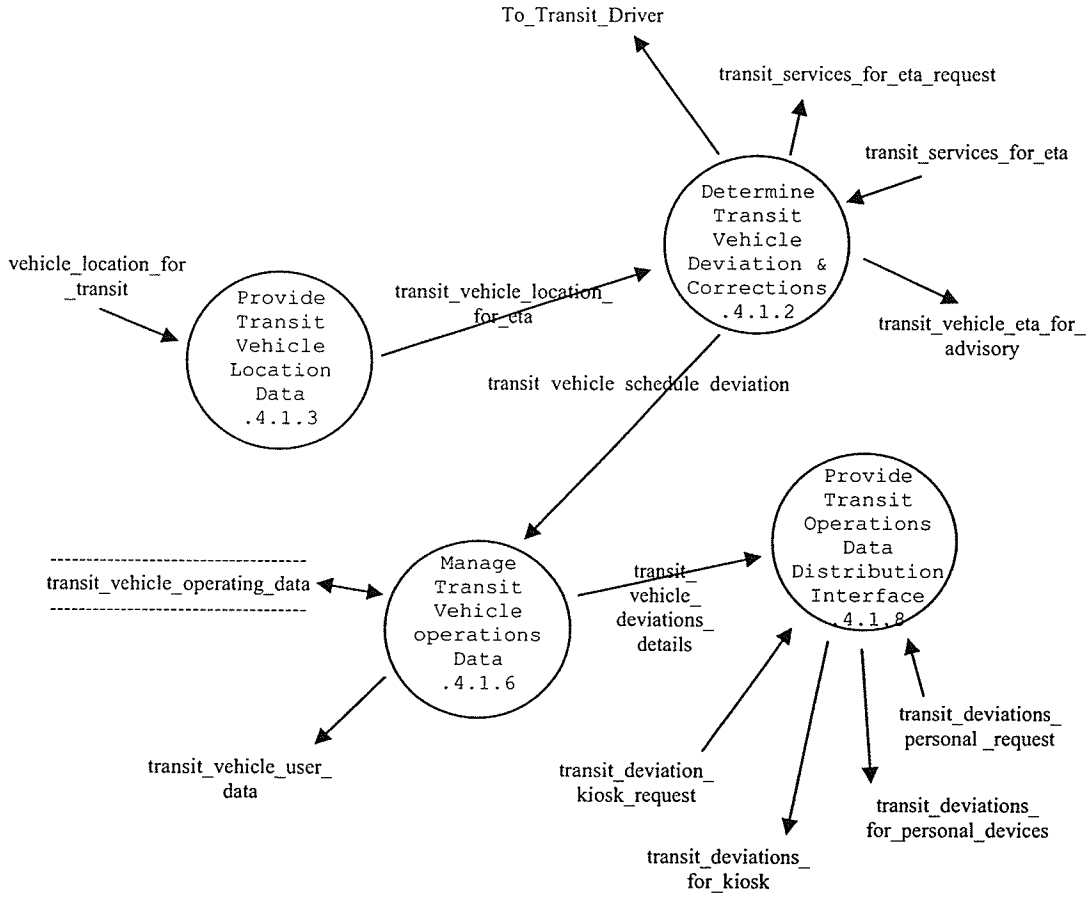


Figure 3-4 – The process "Operate Vehicle & Facilities .4.1" includes lower level and more specific processes shown above.

In Figure 3-4, "transit_vehicle_location_for_eta" conveys vehicle id and its location information to the process .4.1.2. This process also receives a data flow conveying information about bus route number, bus stop name, and bus schedule.

In turn, it provides the information regarding the transit vehicle deviations from schedule and estimates the buses arrival time to the selected bus stops on the desired bus routes. Using confirmatory information about the location of the transit vehicle, this process can generate more precise information regarding the segment number on which the transit vehicle is detected.

More specifically, the process .4.1.2 can be decomposed to two processes "Determine Transit Vehicle Deviation and ETA .4.1.2.1" and "Provide Transit Vehicle Driver Interface .4.1.2.3", as shown in Figure 3-5.

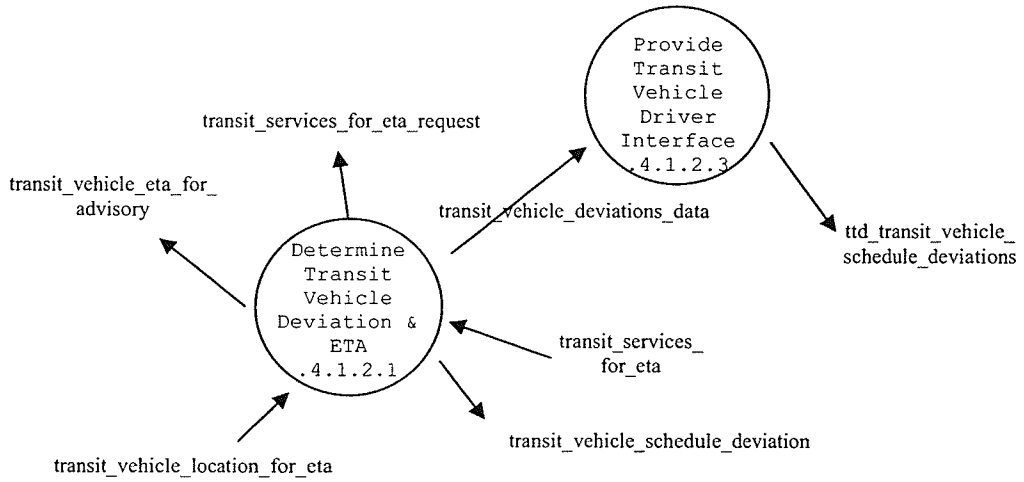


Figure 3-5 – The processes constituting the process .4.1.2 in this project.

The process .4.1.2.1 determines transit vehicle schedule deviation and estimates its arrival time to specific bus stops. This process sends "transit_vehicle_deviation_data" data flow to the process .4.1.2.3 to help the driver take required actions to make the deviations as small as possible.

The "ttd_transit_vehicle_schedule_deviations" data flow represents the data sent to the "Driver" terminator in Figure 3-1 and 3-2. The process .4.1.2.1 sends the request data flow "transit_services_for_eta_request" to receive the details of the route schedule being operated by the transit vehicle. The responsive data flow "transit_services_for_eta" would contain the

required data to calculate the deviation of the transit vehicle from its schedule. If for any reason the amount of deviation is beyond a determined threshold and the corrections become impossible - say due to the maximum speed restriction imposed by the road conditions - an advisory output data flow named "transit_vehicle_eta_for_advisory" is generated. The latter data flow is sent to the appropriate processes within the process .6 to start more a precise positioning inquiry procedure. This procedure is discussed later when position detection is considered in more detail. The result of the position detection procedure is used in the data flow named "transit_vehicle_schedule_deviation". The latter data flow, sent to the process .4.1.6, contains transit route number and vehicle identity, as well as route segment number on which the vehicle is detected. In addition, it indicates the estimated time of arrival is beyond a compensable threshold.

It should be noted that in this project the arrival time is estimated using two different methods for two distinct purposes. The first estimation occurs onboard using fuzzy logic methods in the process .4.1.2. The result of this estimation is provided to the bus driver to perform correcting activities, if possible and/or applicable. More detailed discussion about this estimation method will be presented in Section 7 when the implementations of the components related to the process .4.1.2 is explained. The second arrival time estimation method, using a Kalman filter algorithm, is applied to the data gathered by "Manage Transit Vehicle Operations Data .4.1.6" process where scheduled and real-time positioning database reside and are accessed. There is no direct statement with respect to providing such a service in the definition of the process .4.1.6. However, attributing the mentioned functionality is not in contradiction with its specifications. The details of applying off-board estimation methods to provide buses' arrival times will be discussed when the corresponding components are introduced in Section 9. The result of such estimation is sent to the end users that are interested in having the information about the arriving time of the next buses to their desired bus stops.

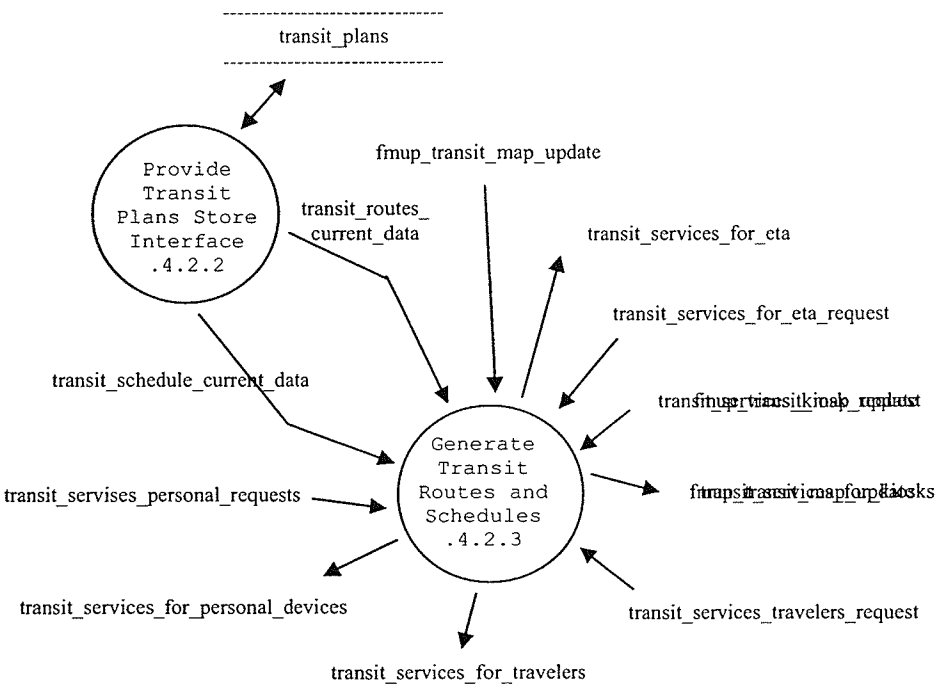
In Figure 3-4, the process .4.1.6 manages the transit vehicle operations data. The data is collected from the process .4.1.2, and through interaction with "transit_vehicle_operating_data" data collection. The data sent to the process .4.1.6 contains information about transit vehicle identity and route segment number (location), as well as the route number on which the vehicle is running. The mentioned pieces of information are particularly important when the transit vehicle shows a delay that can not be compensated.

The outputs from the process .4.1.6 go to the processes "Provide Transit Operations Data Distribution Interface .4.1.8" and "Provide User Roadside Facilities .4.7" through the data flow "transit_vehicle_deviations_details" and "transit_vehicle_user_data", respectively. The former data flow conveys a whole range of information about transit vehicle from its identity and route numbers, bus stop and location to the delay occurred from schedule. The "transit_vehicle_ user_data" data flow will be talked about in more detail when we investigate the function .4.7 in subsequent sub-sections.

The data flow from the process .4.1.6 to the process .4.1.8 provides end users with the real-time operational information of transit system. The end user may receive the information by sending the following inquiry data flows: "transit_deviations_personal_request", and "transit_ deviation_kiosk_request". The mentioned data flows come from different processes within the main process "Provide Driver and Traveler Services .6", the details of which will be explained when the implemented functions of it are considered in the next sub-sections. The data flows generated by the process .4.1.8, in response to the above requests, can be enumerated as follows: "transit_deviation_for_personal_devices", and "transit_deviations_for_kiosks".

The data flows from the process .4.1.8 sent to different functions in the process .6 are bundled in a parent data flow "transit_data". The detail specifications of each of the above mentioned data flows will be discussed later in the following sub-section that is about the process .6.

As shown in Figure 3-3 above, The process .4.1 communicates with the process "Plan and Schedule Transit Services .4.2" through the data flows "transit_services_for_eta_request" and "transit_services_for_eta". The data flows are used for requesting and receiving the data necessary to estimate the arrival time of the transit vehicle to a desired bus stop. Figure 3-6 below shows how the process .4.2 is decomposed further to more specific processes.

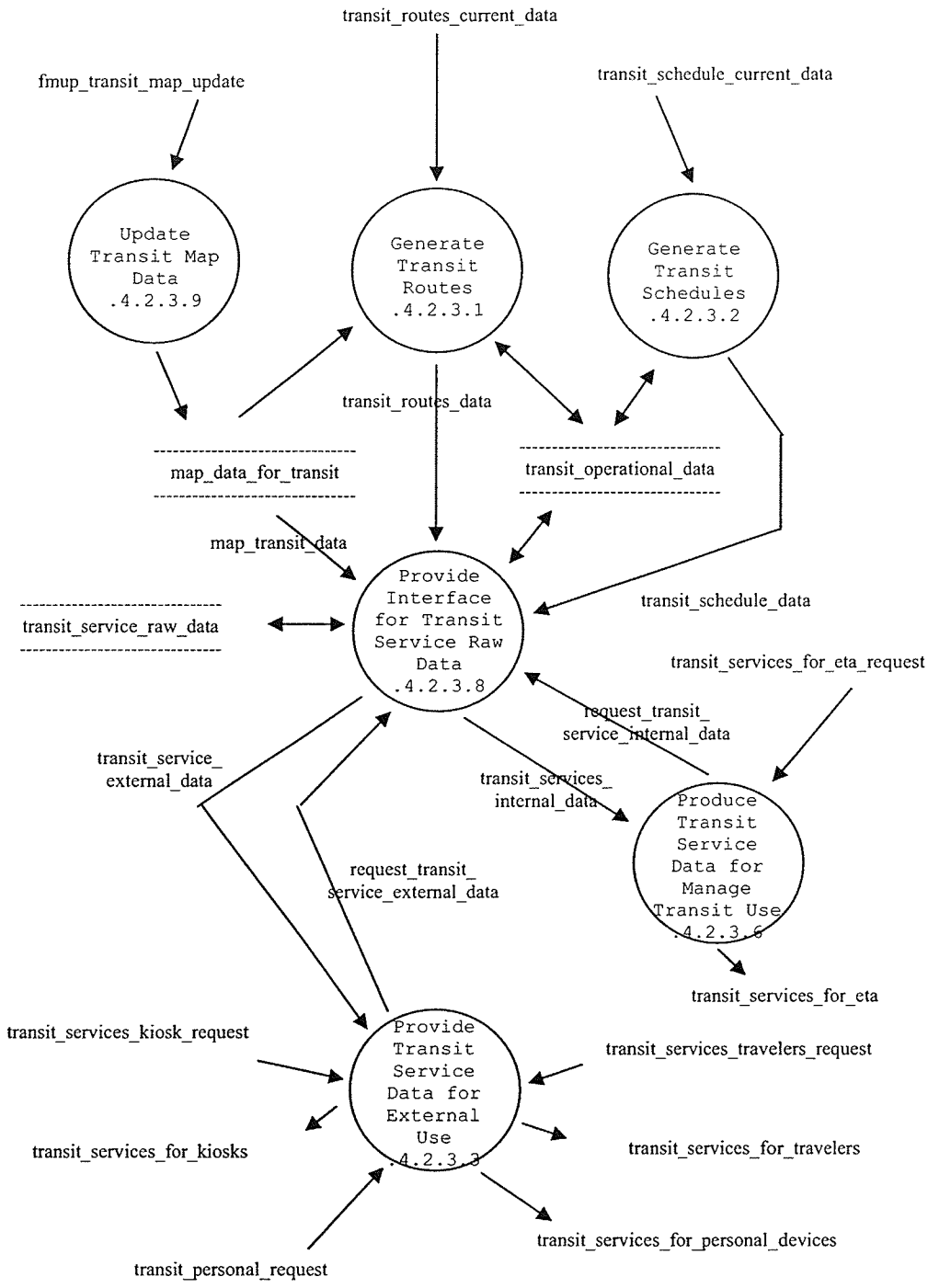


.4.2 - Plan and Schedule Transit Services

Figure 3-6 – The components of the process .4.2 used in this project.

The above mentioned data flows connect the process .4.1.2.1 within the process .4.1 to the process "Generate Transit Routes and Schedules .4.2.3" within the process .4.2. The process .4.2.3 receives two data flows "transit_schedule_current_data" and "transit_routes_current_data" from "Provide Transit Plans Store Interface .4.2.2" function.

It also receives "From_map_update_provider" data flow, originally coming from "Map Update Provider" terminator shown in Figure 3-1 and/ or 3-2. The process .4.2.2 functions as an interface



.4.2.3 – Generate Transit Routes and Schedules

Figure 3-7 – The processes within the process 4.2.3 that are of importance in implementation of this project.

to the store of current regular plans, shown in Figure 3-6 as the "transit_plans" data store. Transit system routes and schedule constitute the contents of the "transit_plans" data store.

The process .4.2.3 receives more requesting data flows from the process .6 and provides the corresponding processes within the process .6 with the data that ultimately make the end user able to get the latest information about the transit system services in a desired location. The process .4.2.3 shown in Figure 3-6 is decomposed to eight more specific functions. Figure 2-7 shows the functions within the process .4.2.3 that are of interest in this project. The process "Update transit Map Data 4.2.3.9" provides updates to the digitized maps used for displaying the desired routes of the transit system to the end users. It receives updated maps from "Map Update Provider" terminator shown in Figures 3-1 and 3-2, and provides them in the "map_data_for_transit" data store.

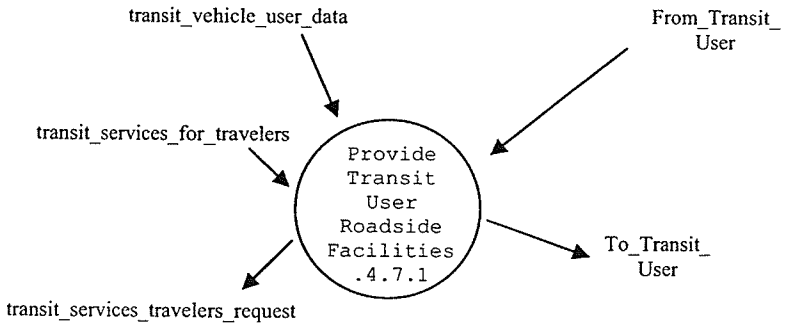
The process "Generate Transit Routes .4.2.3.1" uses the contents of the latter data store. Besides, it receives "transit_route_current_data" data flow to provide the "transit_operation_data" data store with the latest information regarding the route segments, their start and stop points, and their corresponding traveling time, as well as the bus stop locations of the transit system. The latter data store also receives a part of its information from "Generate Transit Schedules .4.2.3.2" process that, in turn, is updated by the "transit_schedule_current_data" data flow. This flow provides the information pertaining to the route number, its schedule, and the segment list.

In Figure 3-7 the process "Provide Interface for Transit Service Raw Data .4.2.3.8" receives input data flows "transit_routes_data", "transit_schedule_data", "request_transit_service_internal_ data", and "map_transit_data", from the processes .4.2.3.1, .4.2.3.2, "Produce Transit Service Data for Manage Transit Use .4.2.3.6", and the data store "map_data_for_transit", respectively.

The process .4.2.3.8 manages and provides an interface to the transit service raw data storage. It receives the data flow "request_transit_service_external_data" from the "Provide Transit Service Data for External Use .4.2.3.3" process. In response, it provides the "transit_service_external_

data" data flow containing transit maps, routes and schedule data. The process .4.2.3.3 is responsible for providing the transit service information for the processes outside of the transit centre, upon receiving their requests. In this project, the requests could be the following data flows: "transit_services_travelers_request", "transit_services_personal_request", and "transit_services_ kiosk_request", coming from corresponding processes within the processes .4.7 and .6. In response, the process .4.2.3.3 sends the requested information back to the requesting processes. There is also "request_transit_service_internal_data" data flow generated from the process .4.2.3.6 coming to the process 4.2.3.8. In response to the request, the process .4.2.3.6 receives "transit_services_internal_data" data flow and provides the process 4.1.2.1, shown in Figure 3-5, with the information about the schedules of the transit system required to calculate the estimated time of arrival of the transit vehicles. The data flow conveying the latter information is "transit_services_for_eta" that is generated upon receiving the data flow "trsnsit_services_ for_eta_request" by the process .4.2.3.6.

As shown in Figures 3-3, 3-4, and 3-7, within the process .4 there are three sub-process .4.1, 4.2, and .4.7 that are in connection with each other. The latter process is decomposed to two processes with one of them of importance in this project. As shown in Figures 3-8, the process "Provide



4.7 – Provide Transit User Roadside Facilities

Figure 3-8 – The important process within the process .4.7. implemented in this project.

Transit User Roadside Facilities .4.7.1" receives two input data flows "transit_vehicle_user_data" and "transit_services_for_travelers" from the processes .4.1.6 and .4.2.3.3, respectively. These data flows were mentioned earlier when the corresponding processes were being described. The latter data flow is received by the process .4.7.1 in response to the request "transit_services_travelers_request" issued originally by the traveler. More specifically, in this project the process .4.7.1 is responsible for providing the public transit information to transit users at roadside locations. These locations can be any place where general public transit information may be provided. The main concern would be the places like homes, work places, or elsewhere that access to the Internet is possible. In this case, the access points behave like roadside terminals providing users with the required interface processes that obtain their requests and reply by sending scheduler, as well as, real-time information about desired transit services. The processes update the provided information periodically, as long as, the user remains connected to obtain the information.

The above discussion terminates our study about the role of the process "Manage Transit .4" in the logical architecture of this project.

In the subsequent parts, those processes within the process .6 that are of importance in this project will be introduced in more detail, and the implementation of them will be discussed, as well.

Recall from Figure 3-2 that the two processes "Manage Transit .4" and "Provide Driver and Traveler Services .6" are connected together through two data flows "transit_requests" and "transit_ data". In this project, each of the above mentioned data flows are constituted of five sub data flows.

The "transit_requests" data flow contains the following data flows: "transit_deviation_kiosk_ request", "transit_deviations_personal_request", "transit_services_kiosk_request", "transit_ services_ personal_ request", and "vehicle_location_for_transit".

On the other hand the "transit_data" data flow is constituted of the following data flows: "transit_deviations_for_kiosks", "transit_deviations_for_personal_devices", "transit_services_for_ kiosks", "transit_services_for_personal_ devices", and "transit_vehicle_ location".

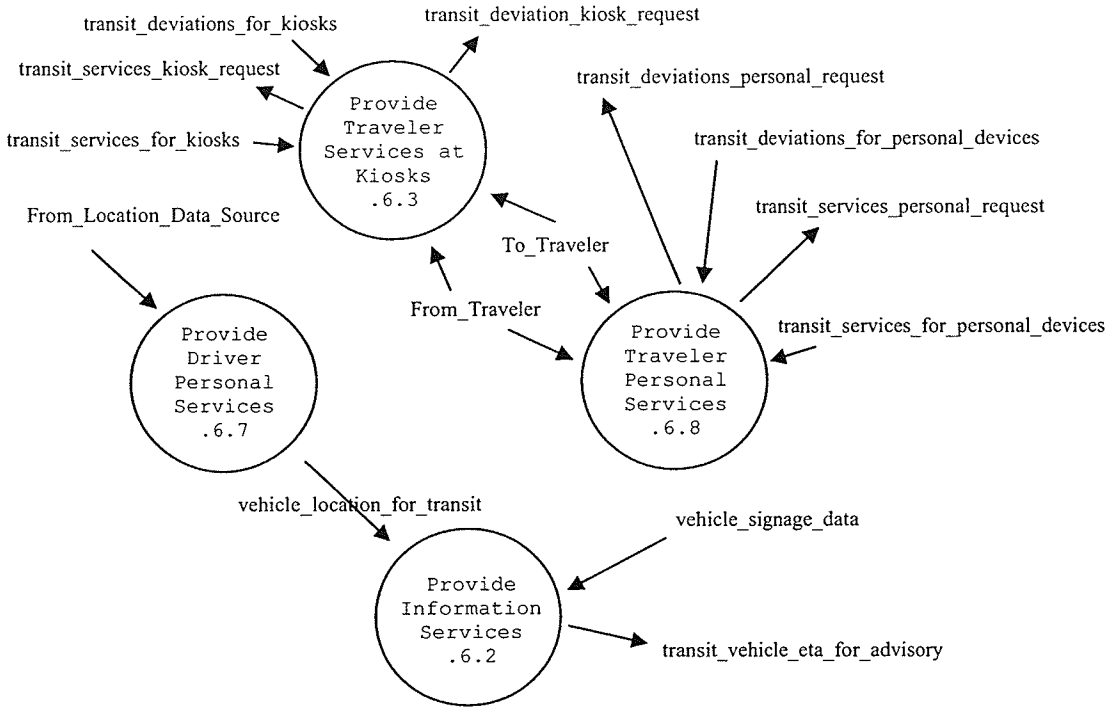


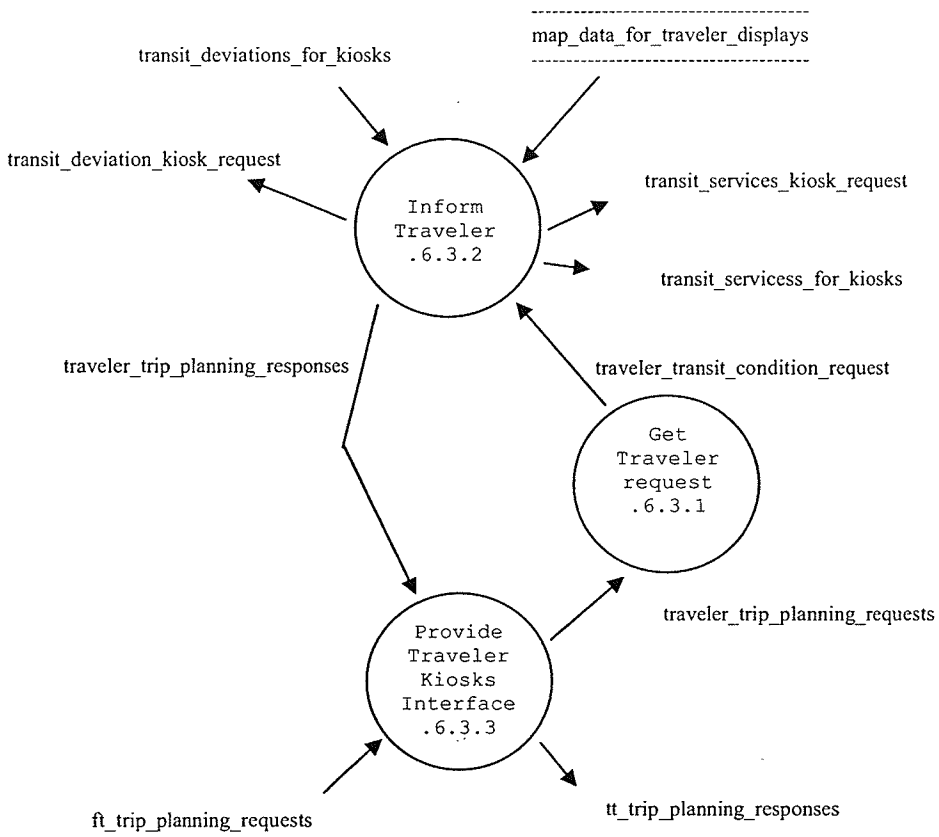
Figure 3-9 – The components of the process .6 implemented in this project.

As shown in Figure 3-9, above, in this project the process .6 is decomposed to three more specific processes, "Provide Traveler Services at Kiosks .6.3", "Provide Driver Personal Services .6.7" and "Provide Traveler Personal Services .6.8".

Figure 3-10 shows how the process .6.3 is decomposed further to three lower level processes "Inform Traveler .6.3.2", "Get Traveler request .6.3.1", and "Provide Traveler Kiosk Interface .6.3.3".

The process 6.3.2 provides the traveler with the transit system information. The data is sent to the interface process .6.3.3, through which the traveler can communicate with the information system set up at a kiosk. The information can be as simple as the time that buses arrive at the bus stop, or

it may display the real-time (and/or scheduler) location of the buses on a digitized map, depending on the facilities installed at the kiosk. The information presented by the process .6.3.2, in fact, originates from the processes .4.1.8 and .4.2.3.3. It also is connected to the "map_data_for_traveler_displays" data store.



.6.3 – Provide Traveler Services at Kiosks

Figure 3-10 – The functions inside of the process .6.3 used in this project.

The traveler makes the information request through the process .6.3.3 that acts as the information interface. The data flow "traveler_trip_planning_requests" takes the request to the process .6.3.1, which is responsible for receiving the traveler requests and directing them ultimately to the process .6.3.2. The requests are subsequently made to the process .4.1.8 and 4.2.3.3 through the

process .6.3.2. The responses are received by the process .6.3.2 and returned back to the traveler through the process .6.3.3.

Now let us consider the process .6.8 that is another major function in the process .6 implemented in this project. As can be seen in Figure 3-11, the process .6.8 is decomposed to a lower level process of "Provide Traveler Services at Personal Devices 6.8.3". The process .6.8.3 communicates the transit system information with the processes .4.1.8 and .4.2.3.3. It also provides the information directly to the traveler. More specifically, the process .6.8.3 is decomposed to lower level functions shown in Figure 3-12.

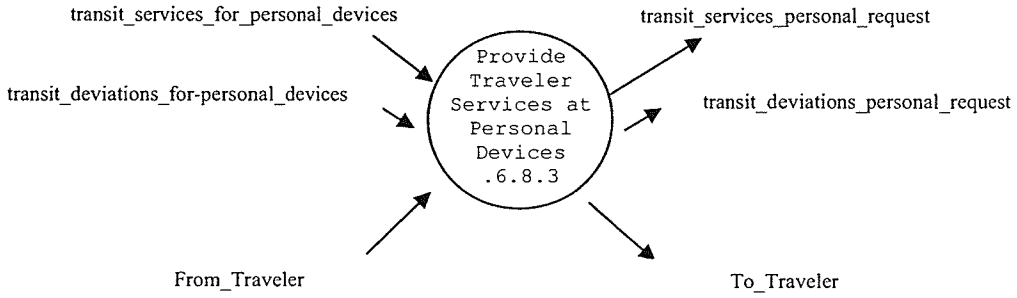
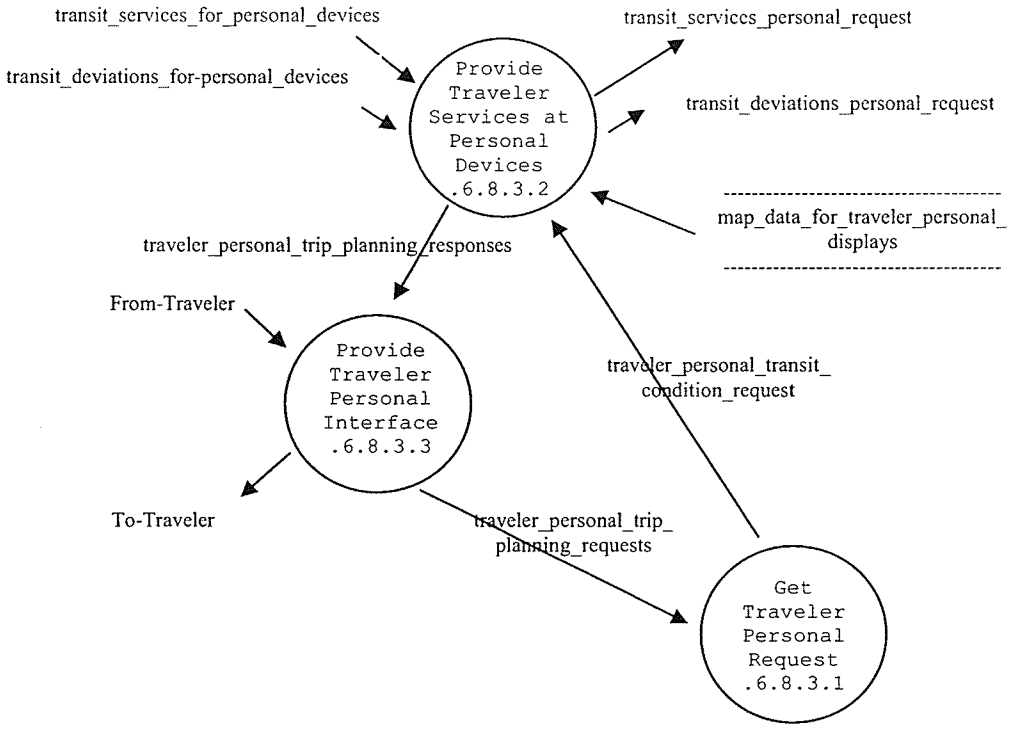


Figure 3-11 – Inside of process 6.8 as used in this project.

In Figure 3-12, the process "Provide Traveler with Personal Travel Information .6.8.3.2" receives the transit system information form the processes .4.1.8 and .4.2.3.3 through the "transit_deviations_for_personal_devices" and "transit_services_for_personal_devices" data flows, respectively. It receives the mentioned data flows through sending the following data flows requests; "transit_deviations_personal_request" and "transit_services_personal_request".

The process .6.8.3.2 also provides the traveler with the digital map displaying the bus route of request in "map_data_for_traveler_personal_displays" data store. The request originated from the traveler side is received by the process "Provide Traveler Personal Interface .6.8.3.3" implemented in a personal device by which transit system information can be obtained.

The traveler holding the personal device enters the request of obtaining the information through the "ft_trip_planning_request" data flow and receives the requested information through the "tt_trip_planning_responses".

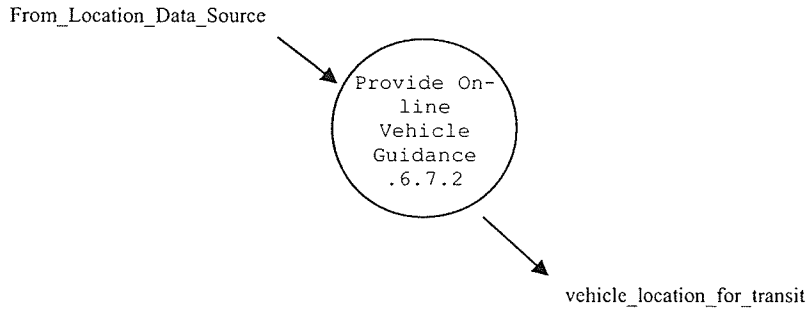


.6.8.3 – Provide Traveler Services at Personal Devices

Figure 3-12 – The components of the process .6.8.3 implemented in this project.

Receiving the former data flow, the process .6.8.3.3 sends the "traveler_personal_trip_planning_ requests" data flow to the process "Get Traveler Personal Requests .6.8.3.1". The latter process, in turn, directs another data flow "traveler_personal_ transit_condition_request" to the process .6.8.3.2 to inquire about the transit condition information. The latter process returns "traveler_personal_trip_ planning_responses" data flow to the process .6.8.3.3, which generates the "tt_trip_planning_ responses" data flow back to the traveler.

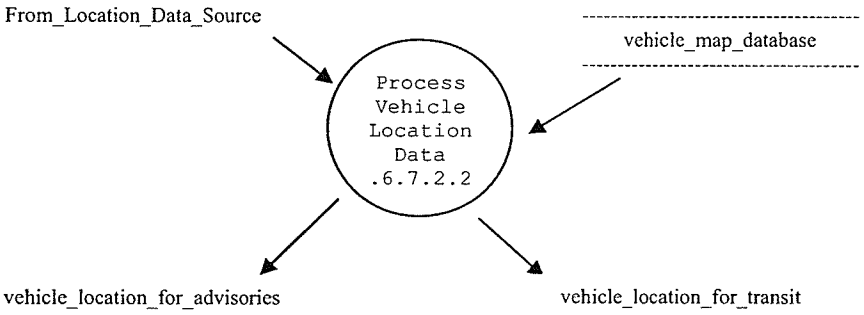
The process .6.7 presented in Figure 3-9 is decomposed to the more specific function of "Provide On-line Vehicle Guidance .6.7.2", as shown in Figure 3-13. In turn, the Process .6.7.2 is constituted of some more specific processes.



.6.7 – Provide Driver Personal Services

Figure 3-13 – The process 6.7.2 constituting the process 6.7 in this project.

Among them, Figure 3-14 shows the one used for the purposes of this project distinguished as "Process Vehicle Location Data 6.7.2.2". The latter process receives the real-time positioning data of the vehicle from a GPS receiver through the "From_Location_Data_Source" data flow.



.6.7.2 – Provide On-line Vehicle Guidance

Figure 3-14 – The components used in the process 6.7.2 as presented in this project.

As will be described later in the Mapping section the mentioned information is compared with the data already stored in the "vehicle_map_database" data store.

The comparison is the first step to change the format of the GPS output to the one useful for the purpose of displaying the vehicle location on the bus routes' digitized maps used in this project. The output of this process, used as a source of raw data of the vehicle location, is directed to the process 4.1.3 for further refinement. Ultimately, the latter process sends the location information

to the process 4.1.2 through the "transit_vehicle_location_for_eta" data flow - see Figure 3-5. It also sends the "vehicle_location_for_dvisories" data flow to process "Prepare and Output In-vehicle Displays .6.2.2". The function of the latter process will be discussed later in this sub-section.

Acting as a scheduler deviation detector, the process 4.1.2 compares the received positioning information with the location where the transit vehicle is supposed to be based on its schedule. If there is any discrepancy between the reported real-time location of the vehicle and its scheduler location the process seeks further confirmation of location. The process first invokes another process that helps reduce the probable error in the output of GPS receiver. In addition it activates a second process that checks for the consistency of the corrected GPS output with an auxiliary positioning data acquired in a different way.

Therefore, there are two methods, other than using ordinary GPS receivers, suggested in this project to determine the real-time location of the transit buses, more precisely, if the data received from .4.1.3 is found erroneous. The two methods work together with the information provided by the process .4.1.3 and will be discussed in detail in Sections 7 and 8.

As a brief note, however, consider this fact that, according to ITS Logical-Architecture, the information provided to the process .4.1.3. through the data flow "vehicle_location_transit" originates from an onboard positioning device like GPS or DGPS. In order to reduce the hardware costs used in this project it is suggested that low price GPS receivers be used in the vehicles. Regular GPS receivers are precise enough for the purposes of this project, however, they may not provide correct results when there are obstacles that prevent them from receiving signals from less than 3 satellites. The latter could be the case in down town and other crowded areas having tall buildings. Foliage, weather conditions, and daytime are among other sources of errors that will be closely investigated in Sections 4 and 5 when GPS and Mapping are subjects of discussion.

In order to reduce the errors affecting the output of the onboard GPS receivers, the output of a GPS receiver installed in a fixed location is continuously monitored and checked against its known position. A correcting signal is generated and becomes available to the transit vehicles when necessary. The mechanism of correcting the output of an ordinary mobile GPS receiver with the correcting signals generated by another GPS receiver located in a fix point is resemble to the scenario used for standard DGPS receivers. However, in the latter case the correction is performed on the pseudo range measurements made by the GPS receiver on the signals receiving from each satellite, separately. In addition, in the normal DGPS receivers the correcting signal is broadcasted through radio frequencies to the receivers equipped with special radio receivers for DGPS correcting signals. In the case of this project, however, the correcting signal is applied to the last step of the positioning procedure when the location is already calculated using all received GPS satellite signals together. Moreover, the correcting signals are available through a server via Internet connectivity for the buses that, as clients, request correcting information from them.

Figure 3-15, below, shows how the data flow "transit_vehicle_eta_ for_advisory" is used within the process .6.2 to invoke the procedure of GPS receiver output correction and confirmation.

In Figure 3-15, the process .6.2.1.2 which is the only inner process of the process .6.2.1 used in this project is shown, along with other processes of .6.2 involved in the positioning and confirmation functions, as applicable in this design. In Figure 3-15, the process "Provide Transit User Advisory Interface .6.2.3" receives the data flow "transit_vehicle_eta_for_advisory", and sends the data flow "transit_user_advisory_information_reques" to the process "Prepare and Output In-vehicle Displays .6.2.2".

The latter process, in turn, sends the request data flow "advisory_data_request" to process "Provide Advisory and Broadcast Data .6.2.1". The process .6.2.1 is decomposed of six more

specific processes. Among them, the only process of interest in this project is the process "Provide Traffic and Transit Advisory Messages .6.2.1.2".

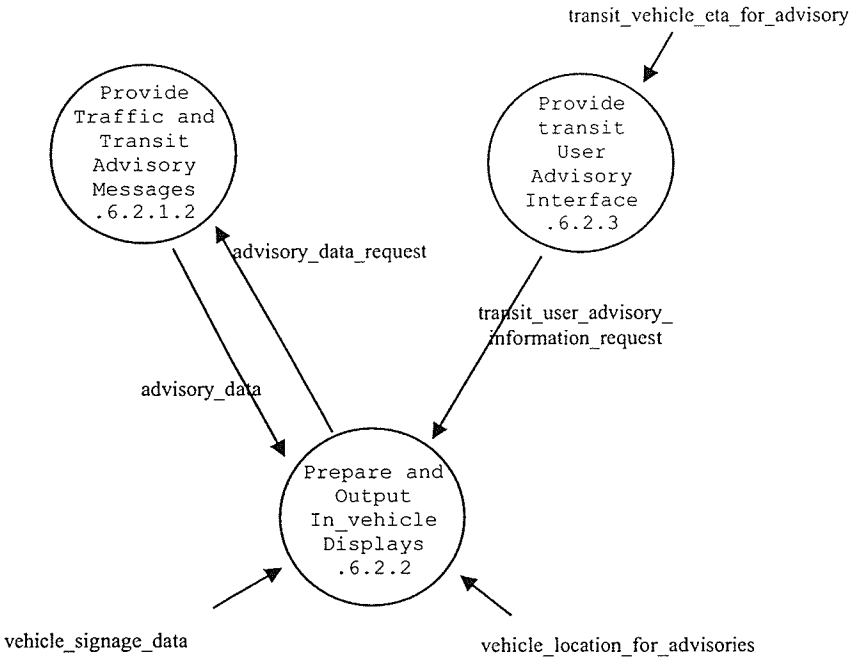


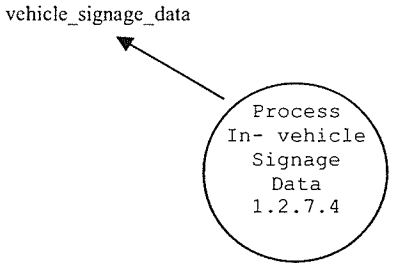
Figure 3-15 – The functions inside the process 6.2 involved in correction and confirmation of the transit vehicle location

This process supports a request/response type of data exchange with the user. The advisory transit information, in this case, is the GPS correction data that is sent to the process .6.2.2 through the data flow "advisory_data".

The process 6.2.2 also receives the data flow "vehicle_location_for_advisories", from the process .6.7.2.2, which is the main piece of information where correcting and confirmatory operations are performed.

As an auxiliary positioning method, this project suggests using positioning beacon signals generated at bus stops located on the roadways of transit routes. Since the bus stops are fixed

points and their location can be determined precisely, a bus in their vicinity can easily find about its location, too. The data flow "vehicle_signage_data", shown in Figure 3-15 above, conveys such an information to passing by transit vehicles. This data flow is generated by process "Process In-vehicle Signage Data 1.2.7.4".



1.2.7 – Provide Roadside Control Facilities

Figure 3-16 – The process used for confirming the transit vehicle's location

The latter process, shown in Figure 3-16, outputs bus stop location data for use in transit vehicles traveling along the road that are within a certain distance from a bus stop providing this function. The capabilities of Bluetooth technology to implement such a positioning system are discussed in detail in Sections 8.

It should be noticed that the ITS-Architecture followed by this thesis is in an evolving procedure and is still being completed to cover new scenarios suggested by emerging technologies.

The two methods suggested in this project to reduce the positioning error and adding the location confirmation attempt to fit within the processes introduced in the architecture. There are, however, minor modifications or additions that are required to be applied to the architecture to let it cover all implementation details performed in this project.

One of the required modifications is a data flow that is able to feed the process 4.1.2.1 with the last confirmed data location of the transit vehicle. In this project, however, we assume that the process 6.2.2 generating the "in-vehicle display" output provides such a signal. The signal is

ultimately used by Transit System Management as a data user – that, of course, would be different from the onboard transit traveler who is the genuine user for which the process 6.2 has been devised.

The origin of this difference is that in the logical architecture there is not any other way of determining the vehicle location other than what is provided through the "Location Data Source" terminator, as shown in Figures 3-1 and 3-2. The existence of such a terminator implies that the processes associated with developing an absolute positioning system in different steps are not supported by the architecture. In this project, however, it is suggested that the positioning correction or confirmation can be facilitated by the above suggested ITS architecture.

Specifically, the information provided by the absolute positioning sensors can be refined, reformatted and processed in different locations in the architecture. There are also other matters of deficiencies observed in the present version of ITS-Architecture summarized, along with the above mentioned one, as following;

1- In the ITS-Architecture, it is assumed that there is only one source of absolute positioning determination. Though the architecture supports relative positioning, fitting auxiliary positioning methods, like the ones suggested in this project, in the present architecture framework is quite a challenging task.

2- ITS-Architecture suggests that transit vehicle estimated time of arrival is performed on the board and the result is distributed to other processes as described earlier in the discussion of the process 4.1.2.1. In this project, however, the estimation of the arrival time is performed in two separate physical places; one onboard and the other in the transit system management unit. In case there is any time delay detected onboard, the real-time location data is posted to the management unit, through which the users access the transit system information. Therefore, there is no need to send the estimated arrival time calculated onboard through the communication channel. Having a schedule information database the estimation of buses

arrival time is performed on the server side and accessed by clients through the Internet connectivity.

3- Media terminals as presented in the ITS-Architecture mainly refer to the receiver of conventional broadcasting methods. Using the Internet as an interactive request/response (client/server) type of media does not fit into the processes suggested in the present version of the architecture. That is why, despite using the Internet as the major media resource in this project to provide real-time information of transit system, the processes leading to use the Media terminal has not been used. Instead, the ITS-Architecture assumes that the traveler uses the facilities, originally, designed for accessing the transit system information on roadsides.

4- The roadside and kiosk procedures suggested by ITS-Architecture, however, have close enough specifications, to be followed in scenarios adopted in this project. The latter, has adapted use for users in bus stops, while the former has been adapted for the case of people accessing the Internet elsewhere and also for location confirmation as described earlier.

The above mentioned discrepancies, however, do not remove the necessitation that one follow the frameworks suggested by ITS-Architecture until new and more complete versions of it are released. The suggested ITS-Architecture, however, works as a general common language providing the present project with a strong backbone.

At this point our discussion about the logical architecture of this project is complete. Tables 3-2 summarizes our discussion so far about all processes and data flows involved in the logical data flow. They make up the required base on top of which further discussion of the physical architecture presented in the following subsection lies.

3-1-3 Physical Architecture

The physical architecture defines physical entities around the processes and data flows presented in the logical architecture of an Intelligent Transportation System (ITS) like the one introduced in

the previous section for this project. Physical entities are constituted of subsystems and terminators, as well as architecture flows that connect the internal components of the physical entities together. Subsystems, in turn, may include internal deployment-size components named equipment packages. Table 3-2 summarizes all the data processing functions of the logical architecture deployed in this project, that were introduced in the previous section in detail, and their corresponding subsystems, as well as, the equipment packages. The parental processes that do not have any direct corresponding subsystem or equipment package equivalent are excluded. Using the general ITS architecture and communication, the subsystems connectivity configuration presented in Figure 3-17 shows how the subsystems of this project are related to each other.

Level	Name	Subsystem Reference	Equipment Packages
.4.1.2.1	Determine Transit Vehicle Deviations and ETA	Transit Vehicle (TVS)	On-board Fixed Route Schedule Management
.4.1.2.3	Provide Transit Vehicle Driver Interface	Transit Vehicle (TVS)	On-board Fixed Route Schedule Management
.4.1.3	Provide Transit Vehicle Location Data	Transit Vehicle (TVS)	On-board Transit Trip Monitoring
.4.1.6	Manage Transit Vehicle Operations Data	Transit Management (TRMS)	Transit Center Fixed-Route Operations
.4.1.8	Provide Transit Operations Data Distribution Interface	Information Service Provider (ISP)	Interactive Infrastructure Information
.4.2.2	Provide Transit Plans Store Interface	Transit Management (TRMS)	Transit Center Fixed-Route Operations
.4.2.3.1	Generate Transit Routes	Transit Management (TRMS)	Transit Center Fixed-Route Operations
.4.2.3.2	Generate Transit Schedules	Transit Management (TRMS)	Transit Center Fixed-Route Operations
.4.2.3.3	Produce Transit Service Data for External Use	Transit Management (TRMS)	Transit Center Information Services
			Transit Center Multi-Modal Coordination
.4.2.3.6	Produce Transit Service Data for Manage	Transit Management (TRMS)	Transit Center Fixed-Route Operations
.4.2.3.8	Provide Interface for Transit Service Raw Data	Transit Management (TRMS)	Transit Center Multi-Modal Coordination
.4.2.3.9	Update Transit Map Data	Transit Management (TRMS)	Transit Center Tracking and Dispatch
.4.7.1	Provide Transit User Roadside &	Remote Traveler	Remote Transit

	Vehicle Data Interface	Support (RTS)	Information Services
.6.3.1	Get Traveler Request	Remote Traveler Support (RTS)	Remote Interactive Information Reception
.6.3.2	Inform Traveler	Remote Traveler Support (RTS)	Remote Basic Information Reception
			Remote Interactive Information Reception
.6.3.3	Provide Traveler Kiosk Interface	Remote Traveler Support (RTS)	Remote Basic Information Reception
			Remote Interactive Information Reception
.6.7.2.2	Process Vehicle Location Data	Vehicle (VS)	Vehicle Location Determination
.6.8.3.1	Get Traveler Personal Request	Personal Information Access (PIAS)	Personal Interactive Information Reception
.6.8.3.2	Provide Traveler with Personal Travel Information	Personal Information Access (PIAS)	Personal Interactive Information Reception
.6.8.3.3	Provide Traveler Personal Interface	Personal Information Access (PIAS)	Provide Traveler Personal Interface
			Personal Provider-Based Route Guidance
6.2.1.2	Provide Traffic and Transit Advisory Messages	Information Service Provider (ISP)	Interactive Infrastructure Information
6.2.2	Prepare and Output In-vehicle Displays	Vehicle (VS)	In-Vehicle Signing System
6.2.3	Provide Transit User Advisory Interface	Transit Vehicle (TVS)	On-board Transit Information Services
1.2.7.4	Process In- vehicle Signage Data	Roadway (RS)	Roadway In-Vehicle Signing

Table 3-2 – The Subsystems and Equipment Packages corresponding to the processes introduces in the Logical Architecture of this project.

More specifically, Figure 3-18 shows subsystems and terminators, as the physical entities used in this project. In the figure, the right-corner rectangles represent the subsystem interfaces and the curved-corner rectangles show the terminator interfaces. The Vehicle Subsystem (VS) provides location sensory functions for the Transit Vehicle Subsystem (TRVS).

The TRVS provides the location information through the Location Data Source Terminator. It also takes the extra vehicle location signage of Bluetooth wireless technology from Roadway Subsystem (RS). The TRVS provides the Transit Driver Terminator with the display of the delay estimation. It is also connected to the Transit Management Subsystem (TRMS) to request and

receive information regarding the schedule performance and the corresponding driver instructions (if applicable).

TRMS takes the required map update information from the Map Update Provider Terminator.

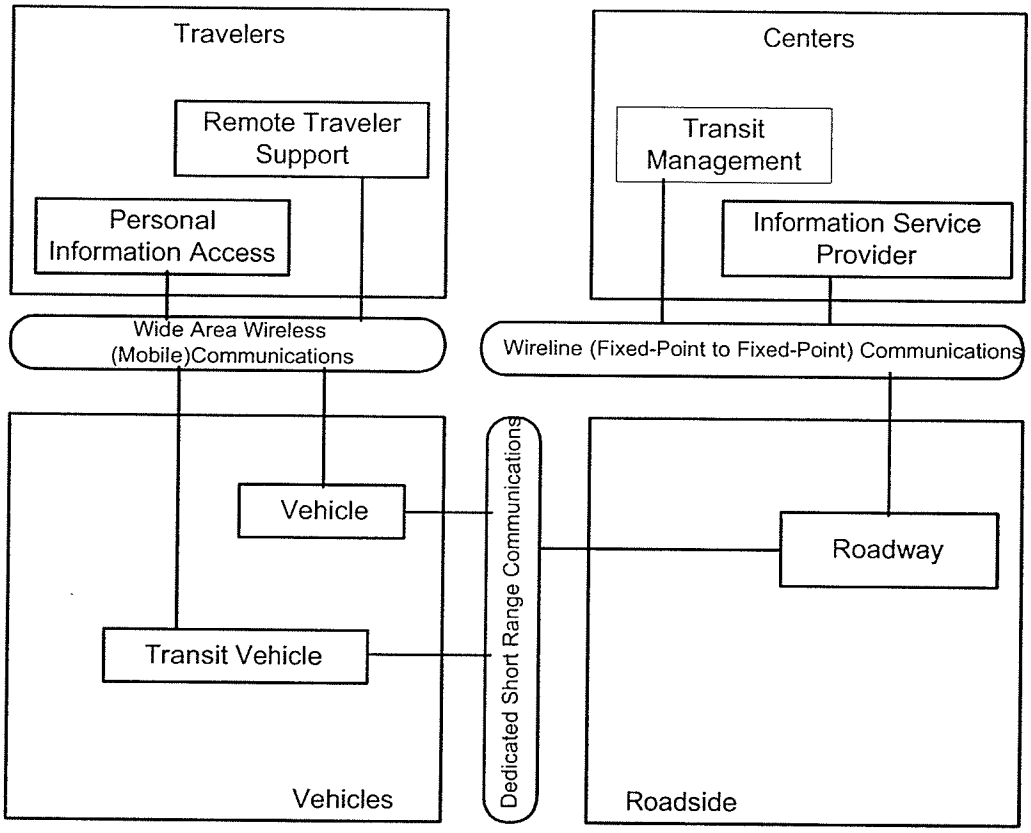


Figure 3-17 – The physical entities and their communications used in this project.

It also provides the information of interests to the Remote Traveler Support (RTS) and the Personal Information Access Subsystems (PIAS). The mentioned subsystems are both connected to the Information Service provider (ISP) subsystem to acquire traveling information based on the information request signal flow that either one could generate.

While the PIAS subsystem is only connected to the Traveler Terminator, the RTS subsystem provides the required information to the Traveler Terminator as well as the Transit User Terminator.

The above brief description of Physical Architecture terminates discussion of the general architecture of this project as a whole. In the following sections the details of implementation of each subsystem are introduced and investigated. In a general ITS system architecture discussion the corresponding equipment packages may also be investigated. However, in order to keep the size of the present section in proportion to the size of other ones, the equipment package specification of each physical subsystem is omitted. The concepts of equipment packages, however, are implicitly discussed in the subsequent sections wherever needed. Finally, Figure 3-19 shows the logical and physical architectures as well as subsystems and terminators of this project, in its entirety.

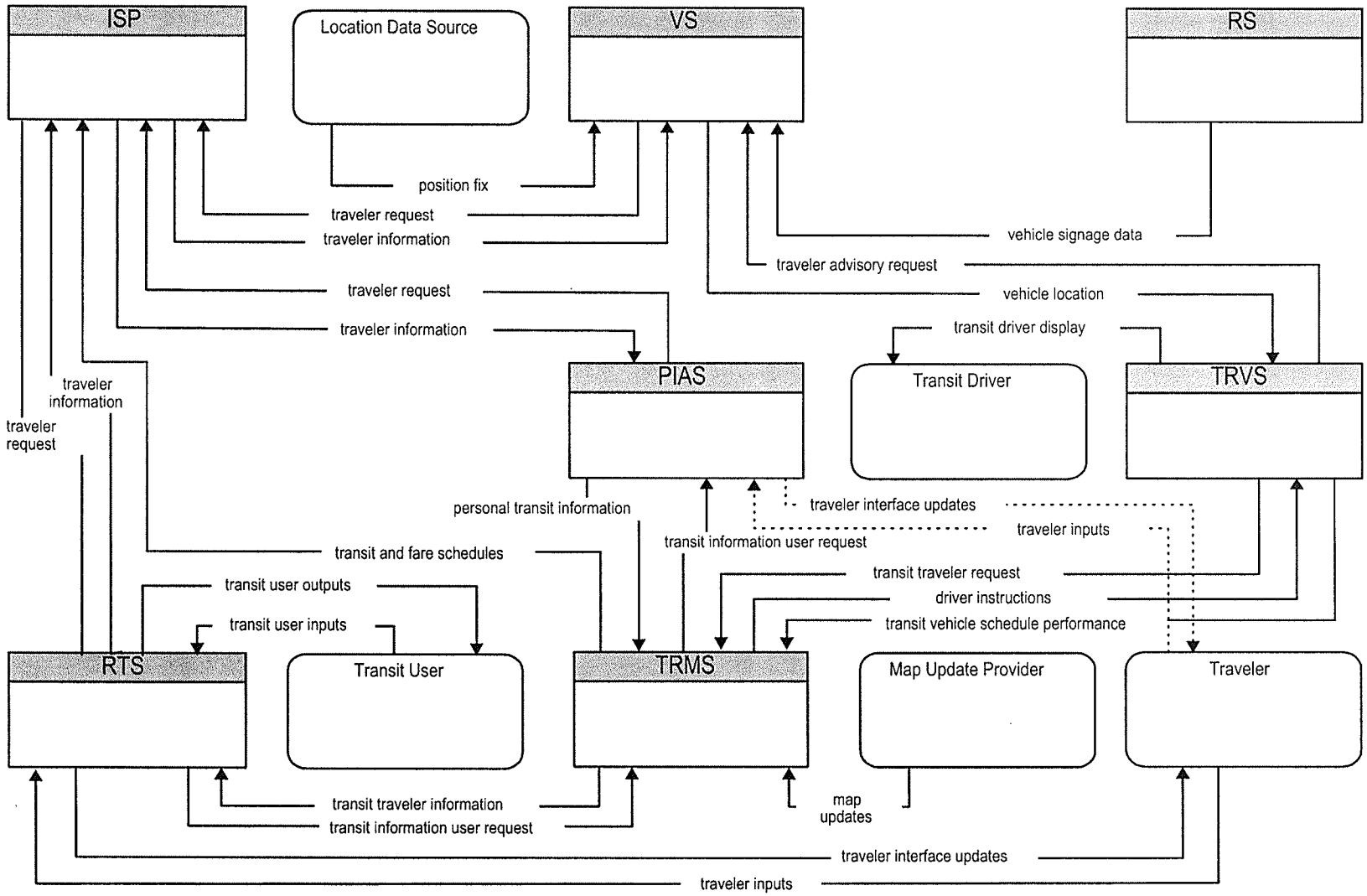


Figure 3-18 - The subsystem and terminator interfaces used in this project.

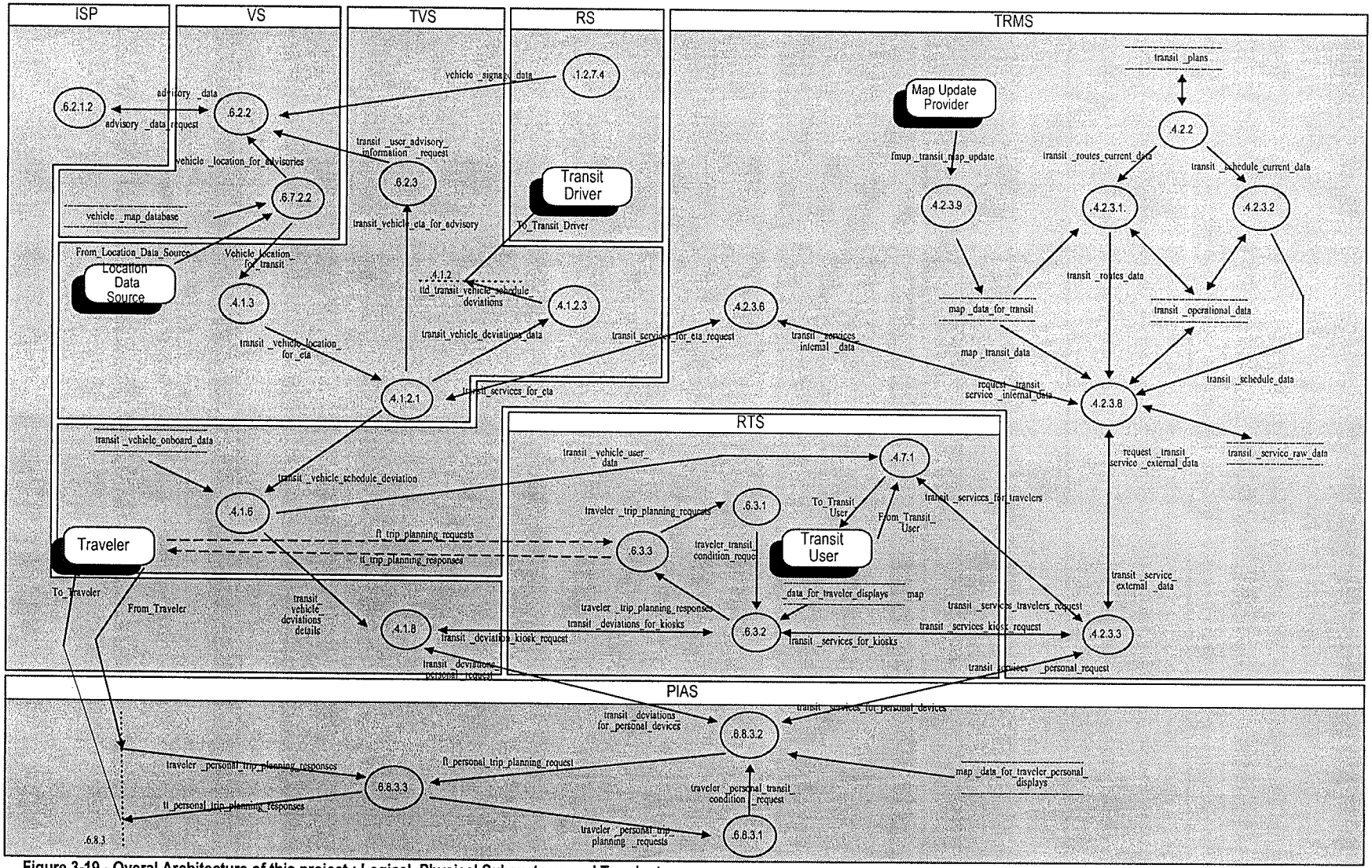


Figure 3-19 - Overal Architecture of this project ; Logical, Physical Subsystems and Terminators.

3-2 References

1. ITS Canada, "*Statistics*", http://www.itscanada.ca/html/statistics.html
2. Transport Canada, "*ITS architecture for Canada – issues study*", 2002-04-03, http://www.tc.gc.ca/tdc/projects/its/a/9618.htm
3. Brian Marshal, IBI Group, "*Review of ITS architecture within the Canadian context*", May 1999, http://www.tc.gc.ca/tdc/summary/13500/13544e.htm
4. Transport Canada, "*An ITS Plan for Canada -EN ROUTE TO INTELLIGENT MOBILITY*", November 1999, http://www.tc.gc.ca/tdc/projects/its/a/9618.htm
5. The National ITS Architecture, http://itsarch.iteris.com/itsarch/
6. ITS Canada, http://www.itscanada.ca
7. ITS America, ©2002 ITS America, http://www.itsa.org/
8. ITS Japan, http://www.iijnet.or.jp/vertis/e-frame.html
9. ITS Korea, http://www.itskorea.or.kr/eng/index.html
10. ITS United Kingdom, Copyright © Aug-02 ITS United Kingdom, http://www.its-focus.org.uk/
11. ERTICO (European ITS), http://www.ertico.com/
12. ITS Australia, http://www.its-australia.com.au/
13. Association for ITS India, http://www.itsindia.org/
14. Intelligent Transportation Society of Maryland, http://www.itsmd.org/
15. ITS – IEEE, Copyright © 2001- 2002 IEEE, http://grouper.ieee.org/groups/scc32/index.html
16. ITS International Magazine, ©Route One Publishing. All rights reserved, http://www.itsinternational.com/
17. ITSC (Intelligent Transportation Systems Council) - IEEE, http://www.ewh.ieee.org/tc/its/
18. Transport Canada, ITS Architecture for Canada, http://www.its-sti.gc.ca/Architecture/english/static/content.htm

19. Transport Canada, Canada's ITS Architecture, http://www.its-sti.gc.ca/en/architecture.htm

Section 4 – Global Positioning System (GPS)

4 – GPS

In this section some essential aspects of the Global Positioning System (GPS) operations and specifications that are of particular interest in this project are briefly reviewed. For a more comprehensive review the references [1] to [11], mentioned at the end of this section, may be consulted.

The Navstar Global Positioning System (GPS), funded and controlled by U.S. Department of Defense (DOD), is a world-wide radio navigation system that sends coded satellite signals to the users, equipped with GPS receivers, enabling them to compute their precise position, time and velocity. Used by 10 - 15 million users in civilian and military sectors around the world, GPS has a market of more than $12 billion a year [14-17].

GPS, in a sense, defines a coordinating system giving every point on the planet a unique address. The user needs to receive at least four satellite signals to be able to acquire position in three dimensions and time. The following sub-sections describe the major components of GPS and how they work together to provide the user the promised service.

4-1 GPS Components

The major segments of the GPS can be enumerated as follows;

- Space
- Control
- Receiver

4-1-1 Space Segment

This segment is an operational constellation of 24 satellites (with one or more satellites, in each orbit, often inactive as spares), orbiting 11000 nautical miles above the Earth, and sending GPS radio signals to the receivers on the Earth surface. The constellation consists of 6 orbits 60° apart from each other with the first orbit inclined 55° from the equatorial plane. The satellites are distributed evenly over the aforementioned orbits and travel around the Earth every 12 hours.

Given that all satellites of the constellation are functional, every user may find 5 to 8 satellites visible from any uncovered point on the Earth. To see illustrative diagrams about the Space Segment and satellite constellation of GPS, see references [3] and [4].

4-1-2 Control Segment

The control segment consists of five tracking stations located around the Earth. This segment monitors and controls the motion of the GPS satellites in their orbits. The stations analyze the signals being received from the satellites and send them back the ephemeris and clock corrections through the Master Control station located at Schriever Air Force Base (formerly Falcon AFB) in Colorado, USA.

4-1-3 Receiver Segment

This segment, consisting of GPS military and civil receiver devices, receives and processes the GPS positioning signals sent by the space segment and provides the user with the position, time, and velocity estimations.

4-2 How GPS Works

In a three-dimensional coordinating system, a desired point can be located on the intersection of three spheres, either of which is centered on a reference point having radius equal to the distance from the desired point. As a result of the errors encountered in measuring the distances between the desired and the reference points in GPS, one more reference point should be included to make precise three-dimensional positioning computation possible.

In order to find distance from a reference point, which is represented by a satellite in the GPS system, a receiver multiplies the velocity of the signal received from the satellite by the time taken for the signal to get to the receiver. It is assumed that the signal velocity is known and equal to the speed of light in free space.

The time taken by the signal to get the destination, however, should accurately be measurable. It is assumed that there are synchronous timers in both transmitter (satellite) and receiver and the

GPS signal is time-tagged when leaving the satellite. In a GPS receiver the difference between the time that the signal is sent from the satellite and when it is received is measured and used for the distance calculation.

Each satellite is equipped with four (two Rubidium and two Cesium) expensive and perfect atomic clocks. GPS receivers, however, can not accommodate atomic clocks to remain synchronous with the satellite transmitters. Ground control stations reset, maintain and apply required adjustments that keep the satellite clocks synchronous to the GPS time, within 0.1 μsec of error. Satellites convey the information of their offset from the GPS time to the receivers to let them apply the required corrections in the distance calculations. The more satellite and receiver clocks drift from each other the more error is encountered in measuring the distance in between.

There is a delay caused by the atmospheric physical conditions that is among the major sources of error in determining the location of the receiver. The time error appears as the fourth unknown independent variable in the set of the positioning equations that should be solved by the GPS receiver. The other three unknowns are location variables x, y, and z. Therefore, to find about its position, a receiver should receive signals from at least four satellites to have a complete set of four simultaneously independent linear equations of four unknowns [18].

As the result of eliminating the time error, caused by the atmospheric conditions, the need for implementing expensive and very precise atomic clocks in GPS receivers are removed. Therefore, GPS receivers can also be used as very good reference devices for the time measurements. Some receivers provide a 1 pulse per second output for this purpose [19-21].

4-2-1 GPS Satellite Signals

Figure 4-1 shows the signals generated and transmitted by the GPS satellites. Every satellite generates specific PRN (Pseudo Random Noise) code by which it can be recognized by the receivers. PRN codes are noise-like, predetermined, and unique for each satellite.

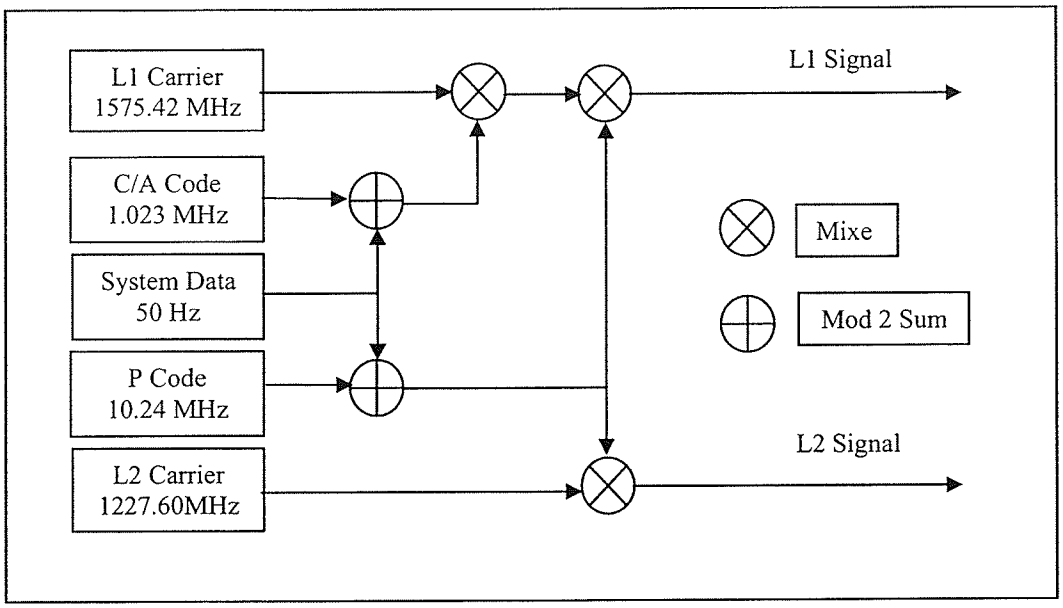


Figure 4-1- Satellite signal block diagram

Figure 4-1 shows a satellite's PRN code generation diagram. As shown in the figure, there are two PRN codes, generated by each satellite, named Coarse/Acquisition (CA) and Precise (P) data codes. Both of these codes convey 50 Hz ephemeral data, containing the information of clock correcting, satellite orbit, and other system parameters. To convey these binary codes to the receivers, every satellite uses two carrier signals L1 and L2 of the frequencies of 1575.42 MHz and 1227.6 MHz, respectively. The binary codes shift the phase of carrier signals by PSK (Phase Shift Keying) modulation methods.

Carrier L1 is modulated by both C/A and P codes. It is used by SPS (Standard Positioning System) receivers. Whereas, Carrier L2 is only modulated by P code and used by PPS (Precise Positioning System) receivers that are capable of decoding this code. In fact, in AS (Anti-Spoofing) mode of operation, PPS receivers decode Y codes that are the encrypted form of P codes. P codes are used for measuring the delay encountered by the satellite signals in ionosphere.

Having the frequency of 1.023 MHz, C/A code repeats itself every 1023 bits (1 milliseconds), and modulates L1 with bandwidths of 1 MHz as a PRN spread spectrum modulated signal. On the

other hand, the P code is a PRN code having frequency of 10.23 MHz, repeating itself every Seven days. Since the PRN codes are unique, GPS satellites are often referred to by their PRN numbers. The C/A code is used in the positioning calculations in the civil receivers employing SPS (Standard Positioning Systems) methods.

4-2-2 GPS Data Protocol

The 50Hz binary code conveying system data such as the satellite ephemeris, clock, ionosphere delay, almanac information, etc., consists of frames of 1500 bits. Each frame is divided further to five time-tagged sub-frames of 300 bits. Each sub-frame is sent during a 6-sec time interval. Therefore, It takes 30 seconds for a satellite to send a complete 1500 bit frame. Subframes also contain parity checks allowing error detection.

The first subframe, in the set of 5, conveys the information regarding the satellite clock correction. It helps the receiver stay synchronized to the UTC ("The language-independent international abbreviation, UTC, is neither English nor French. It means both "Coordinated Universal Time" and "Temps Universel Coordonné"."[32]) to within 100ns. The second and third subframes convey the ephemeris data representing short sections of the satellite orbits. The forth and fifth subframes contain other system parameters including ionosphere delay and almanac orbital information for all satellites. Receiver start up time can be significantly reduced, if almanac information is available.

4-2-3 GPS Receiver

A GPS receiver is capable of replicating 32 PRN codes corresponding to the PRN numbers of each GPS satellite (active or inactive) of the space segment constellation. It takes the correlation of the self-generated PRN codes with the signals it receives from the sky and recognizes the satellites sending the signals. When a match is detected, the time shift required to gain the maximum correlation is also measured. Since the satellite signals are time tagged it is known when a pattern has been sent to the receiver. At the receiver side it is also known when the same

pattern is received. Therefore, the time needed for the signal to reach to the receiver is known, and so is the corresponding distance. The distance measured between the receiver and a satellite is called the pseudorange. As described earlier to determine the position of a GPS receiver and eliminate the offset of its inexpensive clock, the receiver should measure at least four pseudoranges, simultaneously.

4–2-4 GPS Receiver Data Output and Connection

A GPS receiver provides its positioning information in the form of geodetic longitude, latitude, and height above ellipsoid. The geodetic parameters may be presented in different horizontal datum standards. A horizontal datum in fact defines where on the Earth the longitude and latitude lines are drawn. So, each system can be defined as a worldwide grid on the Earth. The most important earth datum systems are NAD27 (North of America Datum of 1927), and NAD83 (North of America Datum of 1983)/ WGS84 (World Geodetic System of 1984), and UTM (Universal Transverse Mercator). For all practical purposes, The NAD83 and WGS84 standards can be considered identical. WGS84, however, is the most commonly used datum in new maps. NASA uses it for earth orbital calculations. It is also used for locating the GPS monitoring stations on earth [22]. Using wrong datum in representing or interpreting the output of a GPS receiver may cause positioning error up to one hundred meters. Based on the datum selected for a GPS, the output may be displayed or delivered by the receiver in various ways. The output, however, corresponds to the map by which it is interpreted, and should always coincide with it. For example, a typical output of a GPS receiver, that is selected to work with a UTM datum map, may look like as follows [8]:

18 435500 E 4248500 N.

For a GPS receiver working with a Latitude/Longitude datum map, sending the output as ASCII code to its serial port, the output can be in the following formats [22,23]:

3606.2778 N, 11956.1321 W or 247591.31250 N, 3798694.25000 E

The above examples are parts of typical GPS outputs that represent the location information. A GPS receiver may provide many more pieces of information pertaining to the UTC time, velocity, direction of movement, etc., obtainable directly or indirectly, by further processing of the available data, from the receiver. The outputs also may have different formats and may be accompanied by various nomenclatures. In each case, it is usually the GPS receiver instruction manual that should be consulted to interpret the output. In this project, however, it is assumed that the output of the GPS is exactly the same or transformable to the positioning data available through ordinary digital maps. More specifically, to simulate the GPS output for the system prototyped in this project, digital maps presented by reference [24] have been used. Mapping, GPS simulation, and the programs implemented in this regard are further investigated in sections 5 and 6.

Most GPS receivers have NMEA-0183 output to send data to other instruments including autopilots. NMEA-0183, developed by the National Marine Electronics Association (NMEA), is an output standard for interfacing Marine electronic devices. The standard defines the data transmission protocol, timing, signal level, and ASCII output format for a 4800-baud serial data bus. There are approximately 50 pre-defined NMEA messages, each having 3-character identifier to provide many different types of navigation information. The signal levels of NMEA-0183, that are in fact EIA-422E, are not fully compatible with RS-232. For practical purposes, however, NMEA-0183 works well with RS-232. This makes connecting a GPS receiver equipped with NMEA-0183 to the serial port of a personal computer or other processing devices possible [27, 28].

4-3 GPS Accuracy and Error Budget

On May 1st, 2000, USA administration turned off an intentionally degradation of GPS signals' accuracy, called Selective Availability (SA). It was an added-in error to C/A signals to prevent "the enemy" making a full benefit of GPS accuracy. Therefore, non-military GPS receivers are

now capable of working 5 to 10 times more accurate than when SA was applying, without any further modification in hardware. They constitute 80% of all GPS receivers and are not equipped with the keys and decryption capabilities of Y codes (encrypted forms of P codes) [12,13].

A typical basic GPS receiver can provide the user with horizontal positioning accuracy of 10 – 20 meters, 95% of the time. The vertical (latitudinal) error is usually 1.5–2 times of the horizontal error, because in a typical GPS receiver the signals receiving from satellites that are spread over the whole angle of view and closer to the horizontal level are normally chosen to measure the horizontal locations. This method reduces the effect of Position Dilution of Precision (PDOP) that has an origin of error in position measurement due to the geographical distribution of satellites over the sky. The more distributed the satellites are over the sky the more precise are positioning measurements. On the other hand, performing the vertical positioning measurements based on the signals received from satellites that are in right angle position to each other with one over the receiver gives more accurate results. A typical GPS receiver, assigns higher priority to the signals received from the satellites distributed over the whole sky angle of the view which is a desirable situation for horizontal measurements with less error. The usage of GPS receivers for horizontal measurements is dominant in the receiver usage in comparison to their usage for vertical measurements [1, 24].

The following error budget is applicable for a commercial GPS receiver [25,26];

- Receiver errors 1.2 m – 1.5 m
- Satellite clock error 0.6 m – 2.1 m
- Tropospheric error 0.7 m
- Ionospheric error 3.6 m - 4.0 m
- Ephemeris error 0.6 m – 2.1 m
- Multipath error 1.4 m

Total rms value of the error 4.19 m – 5.43 m

The error values mentioned above are the results of the experiments reported from two different resources. Therefore, the calculated total rms values, shown above, should be treated differently. The lower end of "the total rms value of the error" range, shown above, should be multiplied by PDOP factor that, in turn, ranges from 2.5 to 5 [25]. The higher end of the range, however, should be multiplied by the Horizontal Dilution of Precision HDOP factor [26] that is estimated to be 2.0. In either case the GPS accuracy is concluded to be in the range of 10 to 20 meters which is consistent with other observations.

GPS receiver error changes with the time of the day, however, the pattern stays unchanged except that it appears 4 minutes sooner every day as oppose to the previous day. Objects that obscure the sky view of the receiver, such as high-rise buildings inside urban dense areas, can add to the error, or prevent fixing a point. Weather conditions that do not change troposphere specifications like rain, snow, or thunderstorms do not effect GPS receiver performance, as long as they do not cover the GPS antenna. Foliage also can affect GPS receiver measurement and increase the error [27].

4-4 Highly Accurate GPS Receivers and Methods

The following GPS measurement methods provide users with a few meters to sub-meter accuracy. The receivers used in these methods are much different than those used in regular single frequency measurements. Various important precise GPS measurement methods can be enumerated as follows;

- Carrier phase measurement
- Average measurement
- DGSP (Differential GSP)

In the case of carrier phase measurement, the GPS receiver uses carrier phase as opposed to the code by which they are modulated to measure time and location. The carrier frequencies are about 1000 times that of the C/A code frequencies, which are used to measure time (distance) in

ordinary GPS receivers. Detailed aspects of this kind of measurements are beyond the scope of this project and are not discussed further in this project. Interested, readers can find good materials in this regard in the literature [29-30].

Average measurement is another method, in which instead of performing one isolated reading at a time, a set of readings are made over some time interval and the average value of the set is reported as the position of the point of interest. Individual readings are performed in the time intervals not less than 15 min, and the overall process may be extended 2 to up to 48 hours. The reading could be as accurate as 3 meters or less, as oppose to 20 meter in normal readings [1-3].

Both of the methods mentioned so far need post processing operations and can not provide the results on a real-time basis.

In the following subsection DGPS receivers and the method employed in their measurements, as the selected highly precise methods in this project, are introduced briefly. Depending on the precision required and the desirable price for GPS receivers a combination of above-mentioned methods may be used.

While DGPS method provides results having enough precision for the purpose of this project, with some modifications, very low price receivers can be used for the positioning readings with almost the same amount of precision, all in a real-time basis, as described in the following sub-sections.

4-4-1 Differential Global Positioning Systems (DGPS)

In the DGPS method, a fixed reference point on the Earth with known positioning data is used to correct the outputs of other GPS receivers existing in the same area. Any difference between the known position of the reference point with the positioning data, measured by the GPS receiver located at the point, introduces calculated correcting values. The corrections could be applied to the measurements of other receivers to get precise positioning. Signals received from the same

satellites pass through the same atmospheric layers and are affected by their physical characteristics in the same way.

Applying the calculated corrections eliminate atmospheric error, which constitutes a major component in the GPS error budget list, as mentioned in the sub-section 4-4.

There are two strategies to calculate and apply the real-time DGPS corrections. The best method, called "measurement domain differential strategy", involves calculating the error in each and every measured pseudorange, and sending the results to other receivers to apply the corrections in the corresponding pseudoranges measured by themselves. In an alternative method, called "position domain differential strategy" which has more restrictions, the calculated errors pertain to the final longitude, latitude, and altitude values. They are obtained and applied after solving pseudorange equations that determine the final positioning values.

In either way, in order to perform real-time corrections, the reference point of a DGPS system should be capable of broadcasting the correcting signals to the receivers with a time delay not more than five seconds. In addition, DGPS receivers should be equipped with the capability of receiving the correcting signals. Depending on the method employed in the DGPS strategy the distance between the reference point and other receivers should be within the range of 20 - 100 km. DGPS systems typically employ radio signals broadcasting to establish the required communication [24,31]. Because of the higher processing capabilities and radio communication facilities employed in the DGPS receivers, they are much more expensive than basic receivers.

4-4-2 DGPS Model Used in this Project

In the case of this project, however, the receivers are not the expensive (DGPS) ones. Therefore, there is no correcting DGPS radio signal. Instead, the system is designed to employ low price basic receivers equipped with serial ports. Using wireless modem connectivity, the GPS receiver, installed in the moving vehicle, is connected to a web server, thereby, it can access the DGPS

correcting factors generated by the GPS receiver installed in the reference point. There are the following differences between a usual DGPS method and the one employed in this project.

- There is no correcting radio broadcasting signals involved.
- The communication link is established through wireless Internet connectivity.
- Instead of receiving correcting signals in determined time intervals, which is every 5 seconds in a typical DGPS method, the receiver on the moving vehicle receives corrections on its own. Therefore, the communication between the reference point and other GPS receivers is based on Client/Server paradigm. On the server side the correction parameters are refreshed every second.
- Instead of expensive real DGPS receivers, the project is designed to use low price basic receivers.

Since the project uses the "position domain differential strategy" it suffers from the limitation that the satellite constellation, simultaneously visible by the all receivers, should be the same. Otherwise, even applying the correcting parameters could not prevent the occurrence of large errors.

Since, in the case of this project, all receivers are within the same area (city), there is high probability that the visible satellite constellation at any given time are the same. However, the above requirement would not be fulfilled when a receiver is in an urban canyon, or where it is in adjacent to tall buildings occluding a view of the sky and/or causing multipath errors. Nevertheless, the mentioned limitation is not really an important obstacle in using the DGPS method as described above in this project.

As mentioned earlier the 20 meters accuracy of ordinary GPS is adequate for the purposes of this project. The DGPS method used is only a tool to double check the output of a receiver when it comes close to an obstacle that prevents receiving signal from satellite(s) existing in a wide portion of the sky. In the scenario described in the next sections, when occlusion occures, the

output of the receiver will be different from what it is supposed to be, based on the location predicted form the vehicle timetable. When the difference is detected to be more than a threshold value, the moving vehicle asks for the correcting parameters generated by the GPS receiver on the reference server. The rover GPS receiver – installed onboard the moving vehicle – applies the corrections and compares the output with the schedule once more. If there is still a discrepancy beyond a determined limitation, the moving vehicle asks for a final positioning fixation through a signpost method. The latter method employs the Bluetooth technology with a known amount of error to fix the vehicle position. The GPS output analysis will be described fully in Section 7 under the same title, after introducing the way that GPS output is simulated in this project in Section 6.

4-5 References

1-P. H. Dana, "*Global Positioning System Overview*", Copyright © 1999 Peter H. Dana, Revised: 05/01/2000 (first published in September, 1994), http://www.colorado.Edu/geography/gcraft/notes/gps/gps_f.html

2-K. Nakamura, "*The Global Positioning System FAQ*", version 9.001 – July 30, 1997, http://vancouver-webpages.com/peter/gpsfaq.txt

3-K. Nakamura, GPSy, "*The Global Positioning System (GPS) Resource Library*", Copyright ©1997-2001, http://www.gpsy.com/gpsinfo/index.html

4- Trimble Navigation Ltd., "*About GPS Technology*", © Trimble Navigation Limited 2002, http://www.trimble.com/gps/

5- Garmin International Inc.,"*What Is GPS?* ", © 1996-2002 Garmin Ltd. or its subsidiaries, http://www.garmin.com/aboutGPS/

6- Ashtech , "*GPS (Global Positioning System)* ", © 2002 Thales Navigation, Inc., http://www.ashtech.com/en/products/aboutgps/gps.asp

7- A. Kalinowski, "*The Good News about GPS*",

http://www.gpsnuts.com/myGPS/GPS/Tutorials/GPS/gps_guide_part_2.htm

8- R. Wilson, "*Maps, Compasses, & GPS's 101, A Basic Course* ",

http://www.gpsnuts.com/myGPS/GPS/Tutorials/Maps/maps.htm#datum

9- Courtesy of S. Wormley, "GPS SPS Signal Specification", main document, 2nd Edition, June 2, 1995, http://www.navcen.uscg.gov/pubs/gps/sigspec/default.htm

10- S. J. Wormley – "Global Positioning System (GPS) ", Educational Observatory Institute, © Copyright 2002, http://www.edu-observatory.org/gps/gps.html

11- S. J. Wormley – "GPS Status, DSPS Status, Errors ", Educational Observatory Institute, © Copyright 2002, http://www.edu-observatory.org/gps/gps.html

12- M. Goodman, S. Jacques," *Fact Sheet, Civilian Benefits of Discontinuing Selective Availability";* May1, 2000,

http://www.ngs.noaa.gov/FGCS/info/sans_SA/docs/GPS_SA_Factsheet.pdf

13- R. Fewing, Cranfield College of Aeronautics, *From Galileo to ...Galileo - European Community Strategies for the Development of a Satellite Navigation System,* ***AEROGRAM*** VOLUME 9 NUMBER 5June 2000, http://www.cranfield.ac.uk/coa/aerogram/june-00/aerog-32.htm

14- Governmental Fact Sheet, *Improving the Civilian Global Positioning System (GPS),* May 1, 2000, http://www.ngs.noaa.gov/FGCS/info/sans_SA/docs/GPS_SA_Factsheet.pdf

15- R. A. Kerr, "*GPS's 'Dress Rehearsal' for Year 2000 Problem*", Volume 285, Number 5429 Issue of 6 Aug 1999, p 816 , ©1999 by The American Association for the Advancement of Science http://www.geol.vt.edu/profs/jas/4104-web/S990806-GPS-Kerr.html

16- J. Lovel, " *A Dress Rehearsal for Y2K: Are You Ready?*", Directions Magazine, July 11, 1999, http://www.directionsmag.com/article.php?article_id=24

17- Engineering Times, *GPS Access Improves*, June 2000, http://www.nspe.org/etweb/16-00briefs.asp

18- C. Drane, C. Rizos, "*Positioning Systems in Intelligent Transportation Systems*", ©1998 Artech House, INC. , pp 169-177

19- SETI League Technical Manual, Tom Clark, "*Totally Accurate Clock*", July 18, 1998, © The SETI League, Inc, http://www.setileague.org/hardware/clock.htm

20- QST, the journal of the American radio Relay League, Tom Clark, "*Totally Accurate Clock*", July 1998, http://www.arrl.org/

21- R. Bean, "*The Clock Mini-Howto*", v 2.1 November 2000, http://www.tldp.org/HOWTO/mini/Clock-4.html#ss4.3

22- Garmin International Inc., "*Garmin GPS Receiver* ", July 15, 2002, http://www.atd.ucar.edu/rtf/facilities/isff/PAM_pdf/Systems/garmin_gps.pdf

23- ERDAS Geographic Imaging, "*Importing Trimble GPS Point Data into ERDAS IMAGINE* ", Copyright ©2002, http://support.erdas.com/procedures/imagine/gps/trimble.html

24- C. Drane, C. Rizos, "*Positioning Systems in Intelligent Transportation Systems*", ©1998 Artech House, INC. , pp 202-216

25- Trimble Navigation Ltd., "*A Guide to the next Utility*", © Trimble Navigation Limited 1989

26- B. W. Parkinson, J. J. Spilker, "*Global Positioning System: "Theory and Applications Volume II)* ", ©1996 American Institute of Aeronautics and Astronautics, pp 478-483

27- GPS Q&A, Earth Observation Magazin, EOM Archive Nov. 1995, "*Industry experts answer reader's GPS questions*", © 2002 by EOM and EOM Inc. http://www.eomonline.com/Common/Archives/Nov95/gps.htm

28- Two-way-Communication, "*NMEA Example* ", http://www.2wcom.com/alf-nmea.htm

29- K. Larson, J. Levine,"*Time transfer using the phase of the GPS Carrier* ", IEEE Trans. On Ultrasonics, Ferroelectrics and Frequency 23- Control, vol. 45, pp. 539-540, 1998 http://www.boulder.nist.gov/timefreq/general/pdf/1220.pdf

30- K. Larson, J. Levine, "*Carrier-Phase Time Transfer*", IEEE Trans. On Ultrasonics, Ferroelectrics and Frequency Control, vol. 46, pp. 1001-1012, 1999 http://www.boulder.nist.gov/timefreq/general/pdf/1291.pdf,

31- Canadian Coast Guard, "*Differential Global Positioning System (DGPS) Broadcast Standard For Marine Navigation*", Last updated: 2001-11-01, http://www.ccg-gcc.gc.ca/dgps/main_e.htm

32- D. Howe, "*Free On-Line Dictionary Of Computing*", Last modified: 2002-12-10 02:30, © 1993, http://foldoc.doc.ic.ac.uk/foldoc/index.html

Section 5 – Mapping

5-1 Digital Map

5-2 Maps of Route 60

5-3 References

5 Mapping

The GPS receiver, as the major positioning device used in this project, was briefly introduced in the previous section. As mentioned in sub-section 4-2-4, GPS output always corresponds with a geometry coordinating system and should be interpreted or displayed with a map created on the same basis.

While various projecting methods and coordinating systems are adopted in creating the maps, which are also used by GPS receivers to generate the positioning data, in this discussion we follow the standards adopted in Canadian National Topographic System (NTS). The National Topographic System provides topographic maps covering Canadian landmarks. They are available in two scales of 1/50,000 (1cm on map equals 0.5Km on the ground) and 1/250,000 (1cm on map equals 2.5Km on the ground). Two grid systems used on the maps are the geographic grid (longitude and latitude) and Universal Transverse Mercator (UTM). The former type of grid systems shows the longitudinal and latitudinal (angular) distances of a point of interest from a reference point on the Earth. The reference point was agreed to be the intersection of equator and the prime meridian running trough the Royal Observatory at Greenwich, England. To learn more about equator parallels, meridian, geographic grid, and coordinating systems references [1-4] may be consulted.

The second type of geographical grid systems, however, reflects the specific projection standard used for depicting the large spherical portions of the Earth like Canada on flat surfaces (sheets of paper).

To learn more about various types of maps available through Natural Resources Canada, reference [5] may be consulted. References [6,7] more specifically talk about the map projection methods, geodetic datum, and the related technical concepts, respectively. Reference [8] gives an overall information about the Canadian Map Libraries and Archives, available online.

5-1 Digital Map

A digital map, in essence, is a digitized format of a map that can be represented and processed by a digital device. Generally speaking, there are two methods to encode and present a map in a computer.

- Vector encoding – In this method a map is converted to a data structure by which it can be presented to the user based on the requested features and functions. Since a data structure creates relation between all the various elements of the map, this method is easier to manipulate and more flexible. It provides faster access time and requires less storage.
- Raster Encoding – In this method a paper map is digitized using a scanner and treated like a digitized image. They can be easily produced and may provide all the information contained in the original image. They can be exactly like the paper maps from which they are derived. They need a large amount of storage which is an important limitation in using them in memory-constraint devices.

In spite of its restriction the raster encoded (scanned) digital maps are used in applications designed for display purposes. Specifically, in the location tracking applications, where vehicle location is displayed by superposing its place over a map, scanned digital maps are of the most interest.

In this location tracking application, scanned digital maps are used. More specifically, and without lack of generality, we assume that the transit vehicles under this study belong to route 60 from University of Manitoba to the downtown Winnipeg and back. That is, we pick only one set of maps and investigate their specifications that are of importance in developing the applications used in this project. Though other routes of the transit system have different maps, the following discussion is independent of the shape of the bus route. Therefore, applications developed for one bus route and based on its own route map, and the conclusions derived, are applicable to each and every other bus route of the transit system in the same way.

5-2 Maps of Route 60

Figure 5-1 shows the area of the city of Winnipeg covered by route 60-Pembina, where buses commute between the city downtown and the University of Manitoba. In a typical application developed for tracking moving vehicles traces like that shown on Figure 5-1 are usually used to display vehicles locations [9]. The route, highlighted as bold black curved trace, could be shown on a map having much more detail including the location of bus stops, information about shopping centers, and/or the locations of public interests like libraries, governmental and educational institutes and so on. On the map shown in Figure 5-1 some of these centers have also been indicated as small-labeled squares. To create the map shown in Figure 5-1, free available online digital maps have been used [10]. Depending on the sources of the digital maps, they can be very different in terms of the amount of information they may provide for the user. The more complicated they are, however, the more difficult for a user to find the location of interest, and also the more computer memory the map occupies. The latter specification, as mentioned before, can be a serious limitation when this type of maps are used in memory constraint devices like handheld PDAs (Personal Digital Assistants).

Particularly, in the case of this project, which Palm Pilot devices are used as an example to show how the bus route's information can be accessed while the user is mobile, using the detailed digital maps do not bear any advantage. However, maps of the type shown in Figure 5-1 are linearly scaled which is a very important characteristic of them. That is, GPS receiver output installed in a moving vehicle can be displayed on these kinds of maps with minimum effort, as long as, both the GPS receiver and the map use consistent geodetic datum. Figure 5-2 shows another kind of map, called in this project as "cartoon-wise maps" or simply "cartoon maps", as issued by the Winnipeg Transit System.

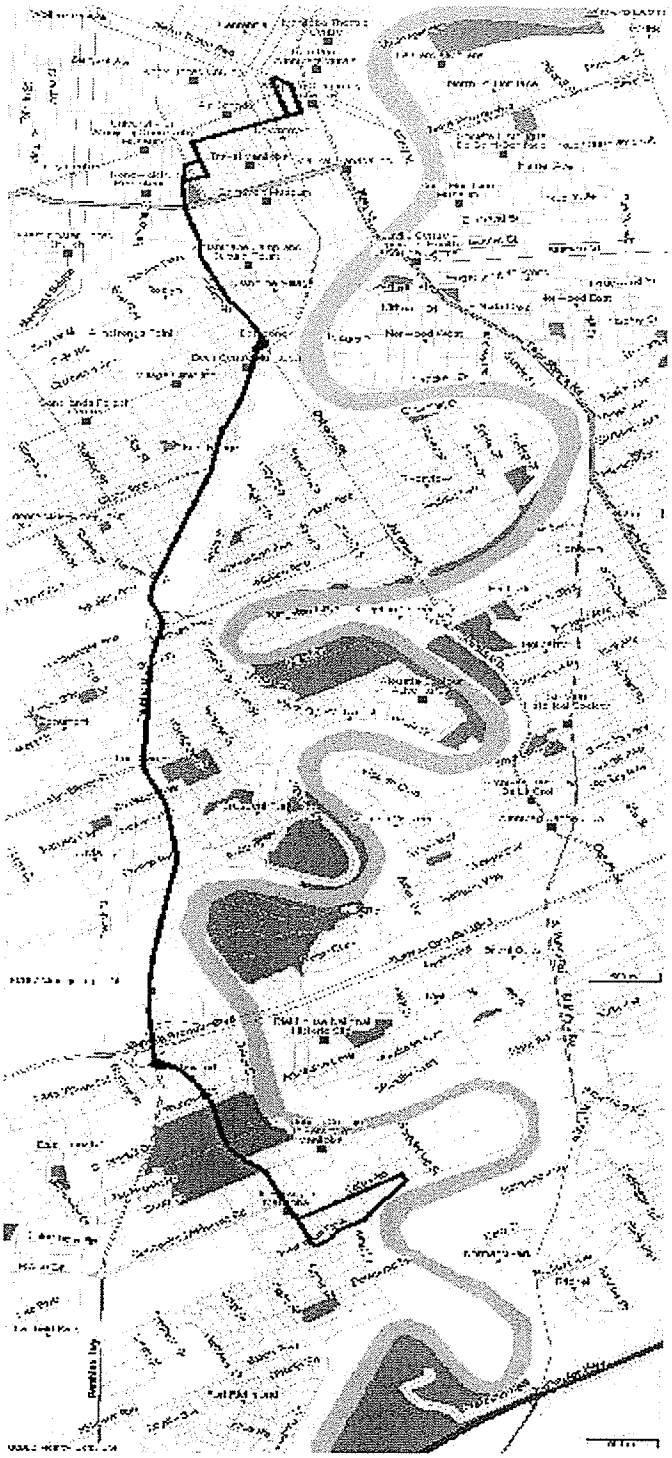


Figure 5-1 - – Route 60, shown on a real map as the heavily darkened trace.

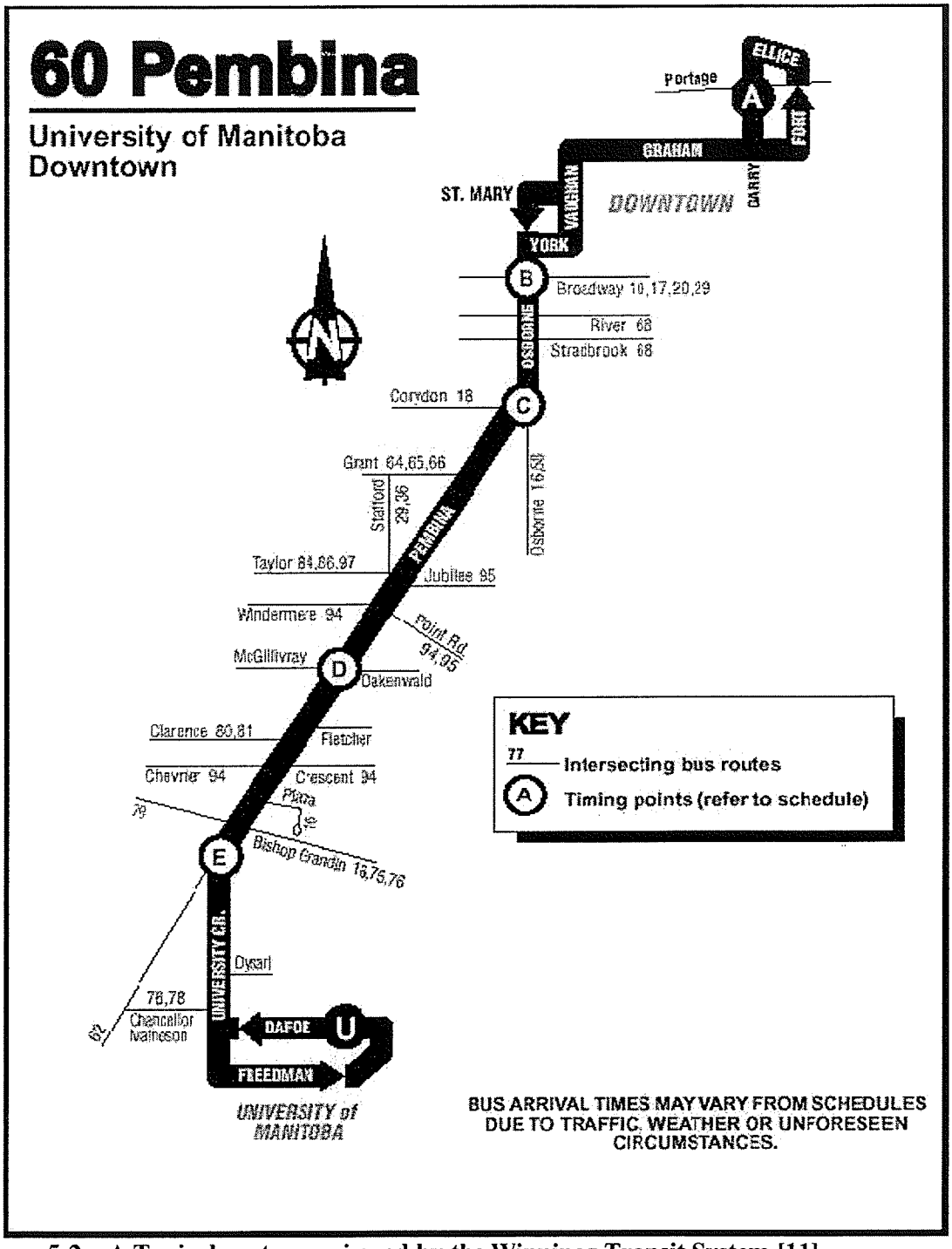


Figure 5-2 – A Typical route map issued by the Winnipeg Transit System [11].

Maps of the latter type are not linearly scaled and GPS receivers outputs are not directly applicable to them. However, they do have the following specifications that make the modified forms of them ideal choices for being used as display components in the user interfaces of the applications developed in this project:

- Cartoon maps have been in use for a long time, and the Winnipeg Transit System uses them along with the timetables of each route in printed leaflets distributed for the transit system users or at the location of its bus stops,
- Transit users are familiar with them and can use them easily and fast,
- They have just adequate information to display and do not need huge amount of memory to provide necessary details.

As cartoon maps, they are neither in scale nor in a predictable distorted form - in comparison to the linearly scaled or the orderly distorted maps available for various usage. Therefore, more effort should be made to use them in accompany with the output of a GPS receiver.

In order to take the most advantages and avoid the disadvantages of either regular or cartoon maps, in this project, the former maps are used as an interface between GPS receiver output and the cartoon maps. On the other hand, the latter maps are used to display the real-time location of transit system buses to end-users.

Figure 5-3 shows the relationship between "GPS receiver" – "real maps" in one side, and the "transit system timetable" – "cartoon maps", on the other side.

As shown in Figure 5-2 there are some fixed points, denoted by circled letters, for which the transit system announces the arriving time of the buses operating in the route. In Figure 5-3, the mentioned fixed points are indicated as oval black colored points on the cartoon and regular maps called A and A', respectively. Since point A represents a fixed bus stop, located at a known intersection along the bus route path shown on the cartoon map, there is no ambiguity in finding its corresponding point A' on the regular digital map. That is why the arrow representing their

direct correspondence on the two maps is shown by a straight line. On the other hand point B' corresponding to the output of the GPS receiver on the regular map does not have a direct correspondence on the cartoon-wise map. That is why its ultimate correspondence, point B, on the cartoon map is shown to be related to it by a fluctuated line arrow. Using different parallelograms to show the two maps is another way to illustrate the distorted relationship between them in Figure 5-3.

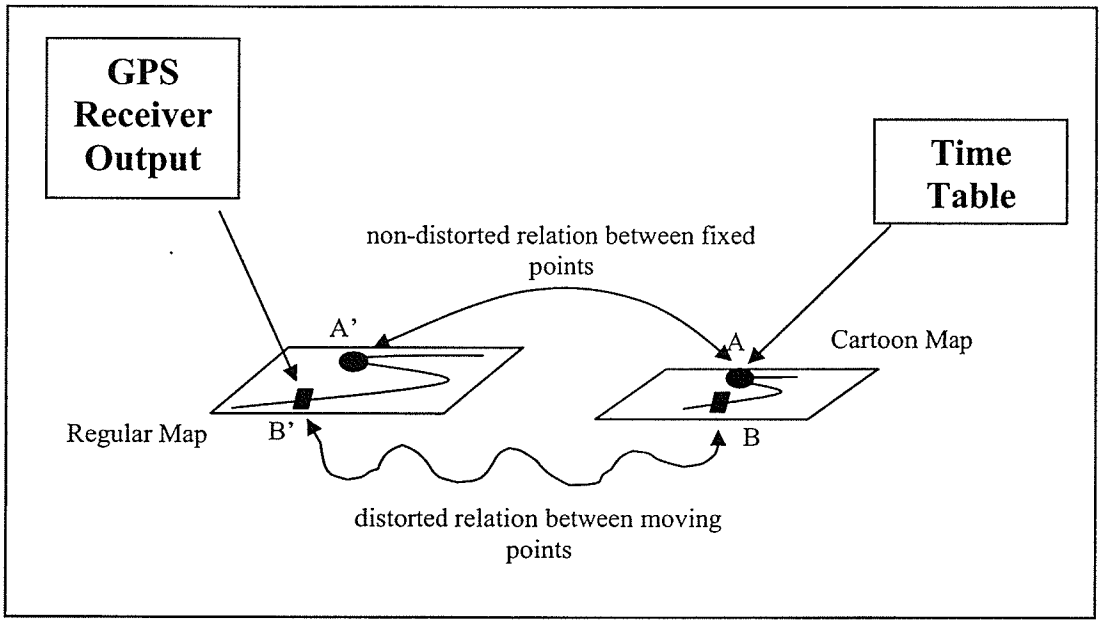


Figure 5-3 - The relationship between GPS receiver, regular map, cartoon map and the transit system timetable.

Materials presented in this section provide enough background to explain how the output of a hypothetical GPS receiver installed on an operative bus on route 60-Pembina, is simulated in this project. In the next section the simulation procedure will be discussed in details and the concepts presented in Figure 5-3 are employed in the simulator development, accordingly.

5-3 References

1- "*The Geographic Grid*", Millersville University, http://www.millersv.edu/~geograph/steve/230/grid/index.htm

2- V. Drake, "*Geographic grid and mapping the Earth*", Earth Science department, Santa Monica College, http://homepage.smc.edu/drake_vicki/GEOGRAPHIC%20GRID%20AND%20MAPPING%20THE%20EARTH.doc

3- Queensland Bushwalking, "*Latitude/Longitude and Universal Transverse Mercator*", October 06, 2000, http://www.qldwalking.org.au/bushwalking/utm.htm

4- S. J. Wormley – "*Maps & Mapping Agencies* ", Educational Observatory Institute, © Copyright 2002, http://www.edu-observatory.org/maps/maps.html

5– Centre for Topographic Information (Ottawa), "Natural Resources Canada", Copyright ©2001, http:// maps.nrcan.gc.ca/

6– P H. Dana, "*Map Projection Overview*", Copyright ©1999, http://www.colorado.Edu/geography/gcraft/notes/mapproj/mapproj. html

7– P. H. Dana, "*Geodetic Datum Overview*", Copyright ©1999, http://www.colorado.Edu/geography/gcraft/notes/datum/datum.html

8- The Association of Canadian Map Libraries and Archives (ACMLA), http://www.acmla.org

9- NextBus Information Systems, © 2002 NextBus Information Systems, Inc. All rights reserved http://www.nextbus.com/predictor/publicMap.shtml?a=sf-muni&r=J

10- Vicinity Corp., "*Map Blast*", http://www.mapblast.com/myblast/index.mb

11- Winnipeg Transit system, "*60 Pembina*", Copyright 1999 City of Winnipeg Transit System, http://www.winnipegtransit.com/TIMETABLE/MAPS/60MAPV6.gif

Section 6 - GPS Receiver Output Simulation

6 GPS Receiver Output Simulation

This project is planned to be a prototype of a complete ITS system to provide the real-time information of the locations of buses operating in an urban transit system, based on a determined schedule. The whole system is designed to work in a modular basis with each group of applications developed for one bus route, independent from the other groups. Therefore, in this project, we only consider developing the required set of applications for only one bus route. All applications developed in this way are fully useable for any other bus routes of the transit system, equivalently, and all results obtained are applicable to the cases of other routes, as well.

As an example and to start the first component of the group of applications required for one bus route, in this section the output of a GPS receiver, presumably installed in a bus, operating in route 60 – Pembina, is simulated.

In reality, instead of the simulator program, a real GPS receiver could be installed on a to works with other software components.

The simulator, however, helps reducing the cost and removes the dependency on any specific hardware. It resembles generating the output of a real GPS receiver, as long as, the latitude and longitude parts of its position coordination are concerned. Altitude, velocity, and time as other pieces of information, usually available through the output of a GPS receiver, are not used by other components of the project and are not required to be simulated here. However, by using the same methods applied for generating the surface coordination (latitude and longitude), directly or with minor modifications, the other pieces of information can be also generated, if required.

In developing this GPS receiver simulator, LabVIEW Graphical Programming for Instrumentation, v5.0 Student Edition has been used as the main core software. LabVIEW is an acronym for Laboratory Virtual Instrument Engineering Workbench. It is a software for developing "virtual instruments" (VIs) using its graphical programming language, called "G". It has been around for over 10 years and is being used by thousands of developers around the world

[1-4]. In addition, the main application and its components have been developed based on the materials presented in Sections 4 and 5.

Since the simulator is designed to resemble the output of a GPS receiver on an operative bus, and it is assumed that the bus is following the route schedule, in the following section the scheduler behavior of a bus is briefly described. Thereafter, simulating of such a behavior is investigated more closely.

6-1 Bus Scheduler Movement

As shown in the previous section the transit system only provides the time of arrival of its buses to a few major bus stops of a bus route as the route schedule [sub-section 5-2].

As can be seen in Figure 5-2, there are some encircled letters on the map showing the major bus stops of the route. Also seen in the figure, the whole path between the first bus stop, A, and the last one, U, is divided into shorter segments denoted by the letters like B, C, D, E, etc.

Figure 6-1 shows a part at the timetable, corresponding to the map shown in Figure 5-2, which indicates the arrival time of buses at the illustrated bus stops. The table content shows that there is only a limited amount of information about where a bus should be at any given time. It also shows the time a bus spends on each segment of the road in different times of the day.

Therefore, it can reflect the road conditions, the passenger demanding trends, and the transit system policies during the time interval that the timetable is announced to be valid and followed by the buses of the route. The validity could be as short as hours or days for special events, to months, seasons or years, in more stable conditions.

In the case where the passenger demand is high the transit system intends to respond to the demands as much as it can. To see how a timetable reflects the road and passenger conditions let's take route 60 south bound as an example and look at its timetable more closely.

A	B	C	D	E	U
5:25	5:32	5:36	5:44	5:50	5:55
5:38	5:45	5:49	5:57	6:03	6:08
5:51	5:58	6:02	6:10	6:16	6:21
6:07	6:14	6:18	6:25	6:31	6:36
6:19	6:27	6:31	6:40	6:46	6:51
6:31	6:39	6:43	6:52	6:58	7:03
6:42	6:50	6:54	7:03	7:10	7:15
6:52	7:00	7:04	7:13	7:20	7:26
7:02	7:11	7:15	7:24	7:31	7:38
7:12	7:21	7:25	7:34	7:41	7:48
7:22	7:31	7:35	7:44	7:51	7:59
7:27	7:36	7:40	7:49	7:56	8:04
7:32	7:41	7:45	7:54	8:01	8:09
-	-	7:50	7:59	8:06	8:14
7:42	7:51	7:55	8:04	8:11	8:19

Figure 6-1 – A part of the timetable corresponding to the route 60, south bond [5]

The information provided by the timetable of route 60, a part of which shown in Figure 6-1, has been extracted and shown in Figure 6-2. At any instant of time, the heights of the layers of different colors show the contribution of each segment of the road in the total time that a bus spends travelling between the first and the last bus stops.

For example, assume that the route condition at time 14:50 is of our particular interest. That is, we like to know how much time it takes for a bus to pass through segments AB, BC, CD, DE, and EU laid between points A and U.

Figure 6-2 shows that it takes 9mins, 5mins, 9mins, 9mins, and 7mins for a bus to pass through the mentioned segments, respectively, to complete its travel from the starting bus stop to the last one. Therefore, in this example, 39mins are needed to a bus to get to the destination U from the starting point A at time 14:50. It should be noted that in the timetable in Figure 6-1, buses move in 5 to 16 minutes time intervals from each other. This means that there are discrete time intervals for which we do have some information about the conditions of the various segments of the route.

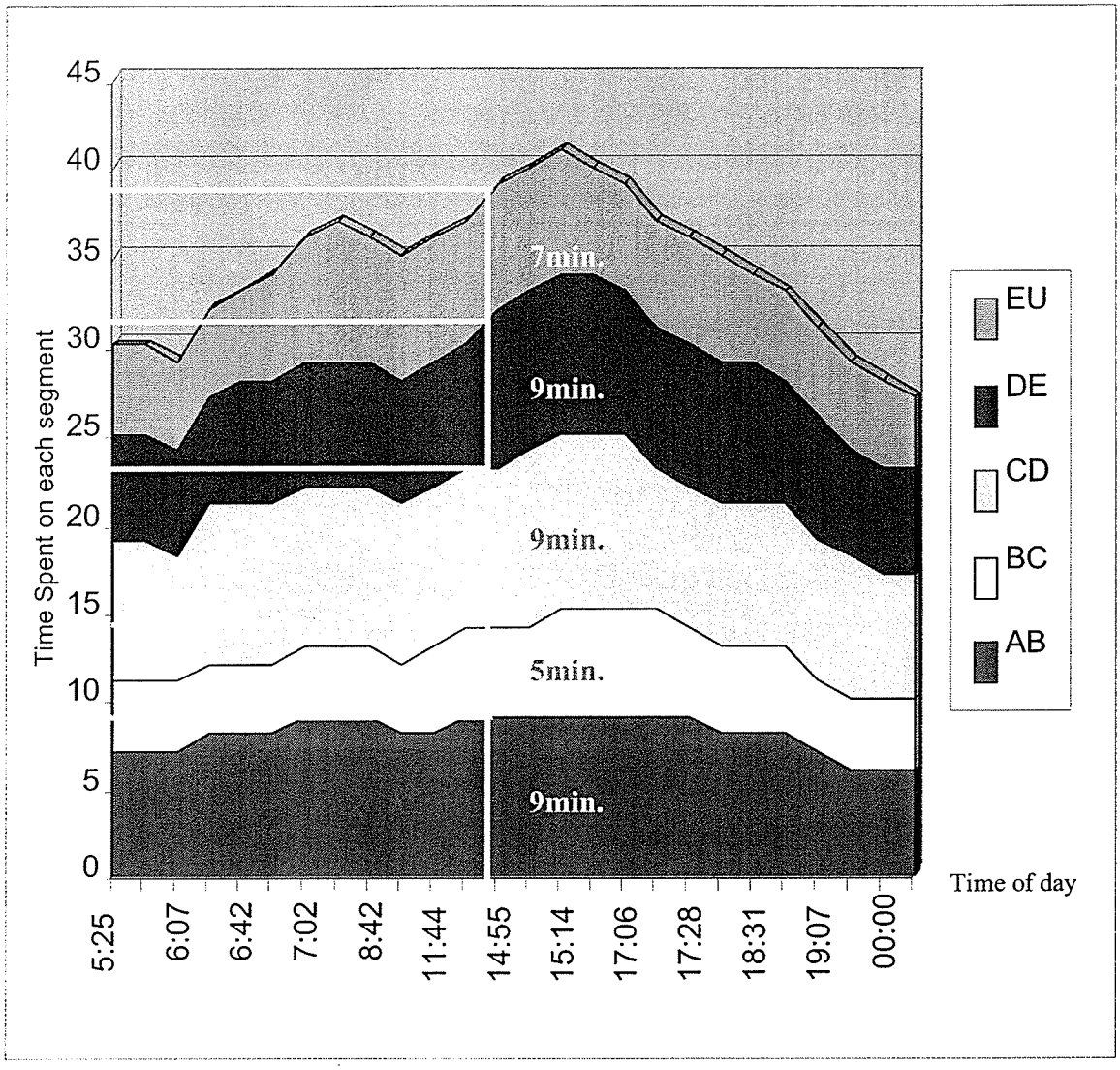


Figure 6-2 – The time that a bus of route 60 spends in each segment of the road.

In Figure 6-2, however, a continuous diagram of segment delays is presented versus time. The data presentation, therefore, has been made under the assumption that the road conditions between two consequent trips change more or less linearly.

In addition, it should be noted that the segmented time spent over each point of the "time of day" axis, shown in Figure 6-2, corresponds to the bus that started the trip at that time instance. Therefore, the time measurement for a bus starting at any specific point of the "time of day" axis

should not be shifted toward the right hand side of the time axis, when the bus arrives to the next segment along its trip and as time elapses.

As can be seen in Figure 6-1, when a bus is in any segment of the road (between two subsequent major bus stops), there is no explicit information about its exact location with respect to the time. However, as indicated in Figure 6-2, it is assumed that a bus generally spends its time linearly between major bus stops. That is, the bus travels between major bus stops with a constant speed. Such an assumption is maintained, particularly, when the bus schedule is being used as the reference timetable for developing the GPS receiver simulator.

In reality a typical GPS receiver outputs a new positioning fix every second. However, in this project there is no need to report a bus position in time intervals sooner than a minute, because the users are informed about the buses arrival time with minute resolution. For the purpose of simulating a GPS receiver knowing the position of buses in time intervals less than a minute is unnecessary, even if the new positioning information is available every second.

In the next section when the GPS receiver output analysis is investigated more closely, it is shown that keeping the frequency of the delay report to the positioning database to 1 report/min or less saves the communication bandwidth and lowers the running costs.

In reality an operating bus spends some time at almost every bus stop to let passengers leave the vehicle or get onboard, and also an extra optional time to become synchronized with its schedule. There are also uncertainties because of the traffic jams, incidents, and various delays behind traffic lights and so on. The more the scheduled path is segmented the more certain one can be about the behavior of an operative bus.

6-2 Application Development

Figure 6-3 shows the components used in implementing this application.

In the following sub-sections the details of each component is investigated, more closely.

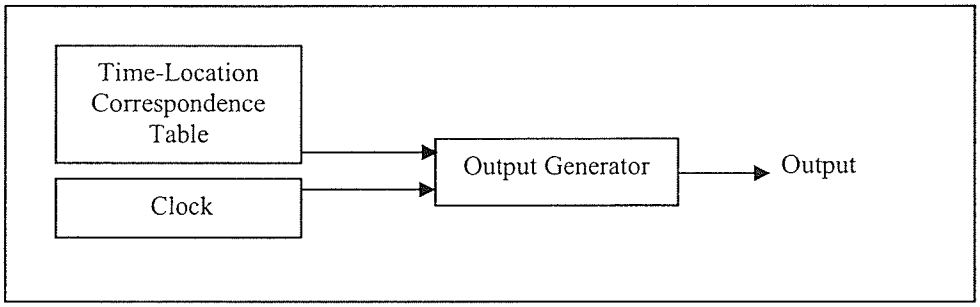


Figure 6-3 – The components of this simulator application.

6-2-1 Time-Location Correspondence Table

Based on the material mentioned in the previous sections the following assumptions have been made in developing the simulator application,

- The GPS output resembles that of a receiver installed in a bus following a timetable.
- It is assumed that the bus motion can be modeled by the motion of a moving platform having constant velocity over each segment of the road.
- Segment lengths and the velocity over it are determined by timetables issued by the transit system. The velocity over each segment may be changed by the time of day according to the timetable.
- GPS receiver positioning measurement error is assumed to be 20 meters 95% of the time. It, however, may change with the time of day (the number of satellites in view) and the location of the bus.
- There is always an output from the receiver, even if the value is totally incorrect. That is, there are no situations in which the receiver can not fix a positioning point. In the next section two alternative ways are introduced to find and discard the positioning values of the GPS receiver that are beyond a determined value of error.

Given the above-mentioned assumptions, the following procedure has been followed to obtain a time-tagged table of points on the cartoon map and the regular digital map corresponding to each

other. To implement the simulator application, such a table is used to generate the GPS receiver output, as will be shown in the next parts of this section.

1- The route shown in Figure 5-2 has been segmented into squares of 4 pixels on each side. That is, the path has been tiled with hidden mosaics with known coordination. The coordinates of each mosaic are attributed to its centre and are usually obtainable by pointing the cursor to it, using any software that works with image files. We represent the coordinates by the following notation "X x Y", in which "X" and "Y" represent the Cartesian coordinate of the square centre in pixel units and "x" is just a separator.

   The location of a bus on the cartoon map is represented by highlighting the corresponding mosaic of it, while other mosaics along the route on the map remain hidden. The cartoon map, that is used to show the position of the bus to the end user, is a slightly modified version of the map shown in Figure 5-2.

2- Using the route timetable the time in which the bus should be at every major bus stop is extracted.

3- The time between pairs of major bus stops is divided by the number of mosaics laid between the bus stops with minute resolution. Therefore, each mosaic of the cartoon map is time tagged with a minute resolution and for every minute of the day there is a location on the map where a bus is supposed to be. There might be more than one mosaic to represent every minute of the day as explained later.

4- The curved path of the route shown in Figure 5-1 is closely approximated by piece-wise straight lines.

5- Each piece generated in the previous step is matched by its corresponding piece of road in Figure 5-2, and divided by the number of mosaics laid down in that segment. Since each mosaic is already time-tagged as explained in step 3, so are the points on the map shown in Figure 5-1. It should be noted that the corresponding segments over the cartoon and regular

maps are determined by the fixed ends of them, usually recognizable by the intersections of the route with crossing streets.

In Figure 5-1, the coordination of the end points of each segment can easily be determined by its latitude and longitude on the digital map. The coordination of other points, in between, are calculated based on the number of points that existed on that segment, determined by the number of corresponding mosaics on the cartoon map.

Figure 6-4 shows two corresponding segments in cartoon and regular maps. The correspondence is indicated by the black mosaics at the ends of both segments.

The figure also demonstrates a special case in which a segment of the cartoon map includes three piece-wise straight-line approximations of the curved part of the corresponding regular map. As shown in the figure, in such cases the cartoon map is divided further to sub-segments, to provide a less distorted approximation. The sub sections are separated and shown by using gray color squares, as indicated in the figure.

In Figure 6-4, it can also be observed that while the mosaics cover all areas of the road on the cartoon map, they may only pin point isolated locations on the regular map. The reason is that the cartoon maps are used for display purposes, as will be shown in the next sections, and the path on them should be totally covered by squares that might represent the bus locations. The regular digital maps, however, are only used as auxiliary devices to help providing an interface between a GPS receiver output and the way that it can be represented to the end user of this project in a more understandable way.

In this respect they are used for only simulation purposes and work as hidden components of the system. Therefore, the regular maps are transparent to the end user. In addition, they are not used as major components of a real system if it were to be implemented. In this project, however, there is one possibility that regular digital maps might be of used as viewable components. This subject

will be discussed in the next sections when central monitoring and GPS output analysis are discussed in more details.

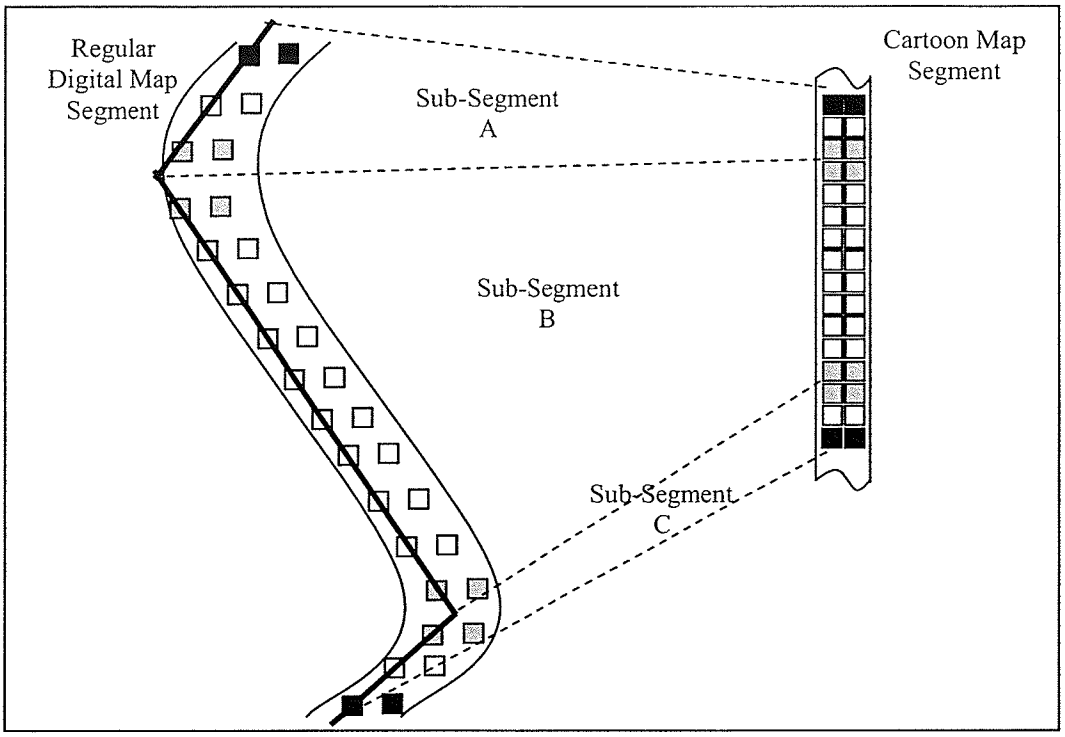


Figure 6-4 – Correspondence between segments between cartoon and regular digital maps.

In this project, the notation used for presenting the coordination of a point on the digital map is as follows "latitude: longitude". Both latitude and longitude figures are decimals with 8 significant digits in "degree".

Table 6-1 indicates a part of the table obtained through the method explained above. In Table 6-1, from right hand side, the column titled "Location Error Coefficient" has the values indicating how the output of the GPS receiver could be affected by the building occlusions along the road. It is arbitrarily set between 1 and 1.3 to apply a degradation in output accuracy up to 30%. The "Direction of Motion" column shows which direction the bus is bound. This is an important parameter to find about the location of a bus and will be discussed in the next section when GPS receiver output analysis is discussed.

The "Cartoon Map Coordination" and the "Regular Map Coordination" columns show the coordination of corresponding points on cartoon map and regular map, respectively. The other columns under the title "Time" indicate the times that a bus is in the location shown by the coordination columns according to the route schedule. Showing only a part of the time schedule of the bus route, the table indicates that there are always more than one point corresponding to the time units with minute resolution. The reason is that in one minute a bus can travel up to 1Km (having the maximum speed of 60Km/hour in the city and travelling non-stop). In the case of the cartoon map used for route 60, a mosaic may represent up to 20 meters x 50 meters of the road (that is equal to a block consisting of almost of 24 typical buses of a transit system [6]).

Time				Regular Map Coordinates	Cartoon Map Coordinates	Direction of Motion	Location Error Coefficient
21:13	22:31	23:37	1:04	**49.8956284:-97.141414**	**453x30**	ns	1.3
21:13	22:31	23:37	1:04	49.8953594:-97.141280	453x35	ns	1.3
21:13	22:31	23:37	1:04	49.8950894:-97.141145	453x40	ns	1.3
21:13	22:31	23:37	1:04	49.8948194:-97.141011	453x45	ns	1.3
21:13	22:31	23:37	1:04	49.8945494:-97.140876	453x50	ns	1.3
21:13	22:31	23:37	1:04	49.8942794:-97.140742	453x55	ns	1.3
21:13	22:31	23:37	1:04	49.8940104:-97.140607	453x60	ns	1.3
21:14	22:31	23:37	1:04	49.8937404:-97.140473	453x65	ns	1.3
21:14	22:31	23:37	1:04	49.8934704:-97.140338	453x70	ns	1.3
21:14	22:32	23:38	1:05	49.8932004:-97.140204	453x75	ns	1.3
21:14	22:32	23:38	1:05	49.8929304:-97.140070	453x80	ns	1.3
21:14	22:32	23:38	1:05	**49.8926604:-97.139935**	**453x84**	ns	1.3

Table 6-1 – A sample part of the table used for generating GPS receiver outputs by the simulator

Therefore, the maximum length traveled by a moving bus in every minute may cover up to 20 squares defined on cartoon map or their corresponding points on the regular map. In practice, the average speed of a bus may be half of the maximum permitted speed of the road or less, and depending on the distortion of the cartoon map the number of mosaics covered by a bus in a minute may be 10 or less.

Table 6-1 shows only a part of the time schedule of the bus route, in the columns under the title of "Time".

As the last step of the table investigation, let's look at the contents of the column "Regular Map Coordinates", and as an example, the last point presented in the table, more closely. The figure "49.8926604" is in units in degrees. Since the least significant digit is read as "4", in all GPS receiver reports, it is ignored and the figure is evaluated as "49.892660". It's equivalent in degree, minute, second format is as follows "49° 53' 33.58" ". In the latitude of the city of Winnipeg, each second is almost equivalent to 20 meters, The above figure of latitude gives resolution of 0.2 meters/second on the digital map shown in Figure 5-1. This is also the case for the longitude resolution in the area of interest represented by this map, GPS produces an output in either format, or can be easily convertible to them.

6-2-2 Clock

As mentioned earlier a regular GPS receiver fixes a new position every second. However, because the resolution of the positioning report time in this project is one minute, the reading frequency of one minute is used throughout in this project. One-minute delay in reporting a new position, however, could be a long time interval, particularly, when the system performance is investigated for the test or demonstration purposes. In Figure 6-3 the "Clock" block represents a system component that determines the time interval that a new output value appears in the simulator output. The new output time interval can be adjusted to any value that is suitable for test, demonstration, and/or real work conditions. The adjusted value, in fact, determines the scanning time of a text file that generates the string output with the format "HH: MM", resembling a clock output string. "HH" and "MM" stand for hour and minute of the time, respectively. For resembling real-time working conditions two alternative programs have been implemented. One that uses computer internal clock to generate a new read every time that it switches to a new minute. And another one that simulates switching to a new minute in any

adjustable time interval including a minute. While both methods basically work in the same way, the former implementation is not suitable for fast scanning, trouble shooting or demonstration purposes. However, it provides the impression of working of a real system, which is of course obtainable through the latter method too. In the remaining parts of this study the adjustable scanning time interval is used for the advantages mentioned above.

6-2-3 Output Generator

In Figure 6-3, the block "Output Generator" represents the program implemented for generating an output corresponding to a GPS receiver, presumably installed on a bus operating in the route 60 – Pembina and following the transit system schedule, as explain in the previous sections. The program takes the following steps to generate every location output:

1- The output is generated every time that the clock turns out to a new minute. The simulator may use the computer internal clock to measure one-minute time intervals or it may scan through a text file resembling the clock output in the time intervals adjusted desirably for different purposes, as explained in section 6-2-2.

2- The coordination of the points corresponding to the time presented at the clock output is read from "Time-Location correspondence Table". Since there might be more than one point corresponding to each time unit of one minute, the application selects one point out of the points' set corresponding to the time.

3- A standard error of 20 meters, multiplied by three error coefficients, is calculated as follows;

$$e_T = 20m \,.\, e_l \,.\, e_t \,.e_w \qquad (6\text{-}1)$$

In relation (6-1) e_T, e_l, e_t and e_w represent total error, location error coefficient, time of day error coefficient, and worsening error coefficient, respectively. Location error coefficient, e_l, results from a lack of receiving enough satellites signals due to the spatial occlusions in different locations along the road. Time of day error coefficient, e_t, originates from the horizon of the satellites constellation changing over sky during the day. Worsening error

coefficient, e_w, is used only for test and demonstration purposes and does not have a real equivalent.

4- The total error is randomly broken to two components e_{lat} and e_{lon} so that relation (6-2) is held;

$$e_T = (e_{lat}^2 + e_{lon}^2)^{1/2} \qquad (6\text{-}2)$$

In relation (6-2), e_{lat} and e_{lon} are latitudinal and longitudinal components of the total error e_T.

In fact, the above-mentioned procedure takes, randomly, a point from a circle around a fixed location on the regular digital map. The fixed point corresponds to where a bus should be, according to its schedule, at the given time.

6-2-4 Implemented Programs

The accompanying files of this section constitute the simulator application created in LabVIEW "G" language and can be run by LabVIEW version 5 or higher. They are .vi (Virtual Instrument) files that will be used in the next section with GPS receiver analyzer application. The other two text files are "Time-Location Correspondence" and "Clock" files, explained above in details in sub-sections 6-2-1, and 6-2-2, respectively.

As the last point it should be emphasized that in reality the GPS receiver is connected to a computer through a serial port (see Section 4). While it is possible to simply modify the present simulator to a form sending its output to a serial port, in the case of this implementation, the information delivery is performed internally. The latter method removes the need for two computers to take the roles of GPS receiver and GPS output analyzer connected together through their serial ports. In this way, both applications are run in one computer and the data is transferred internally.

6-2-5 Files Related to this Section

1. GPSdataGeneratorRoute60ScheduleFastScan5.vi, including:
   - ClockStringScheduledHoursUsingTextFile.vi

- GPSReceiverOutputSimulationRout60SchedulerMovement3.vi
- R60DMap_CMapCorres_Intrplt_N_S_Schdled2_LocationErrIncl_Rowed.txt
- ClockOutputSimulator24hoursFormatCoincideBusSchedule.txt

2. GPSdataGeneratorRoute60Schedule_InternalClockMinuteResolution_FastScan5_Drived.vi, including:

- ClockString24HoursUsingInternalComputerTime.vi
- GPSReceiverOutputSimulationRout60SchedulerMovement3.vi
- R60DMap_CMapCorres_Intrplt_N_S_Schdled2_LocationErrIncl_Rowed.txt

6-3 References

1- National Instrument Corp., "*LabVIEW Resources*", © 2002, http://amp.ni.com/niwc/labview/resource.jsp

2- M. T. Hall, M. L. Martin, "*LabVIEW for Automotive, Telecommunications, Semiconductor, Biomedical, and other applications*", Upper Saddle River, NJ : Prentice Hall PTR, c2000.

3- G. W. Johnson, "*LabVIEW Graphic Programming*", 2nd Edition, McGraw-Hill Professional Publishing, c1999

4- J. Essick, "*Advanced LabVIEW Labs*", 1st edition Prentice Hall, c1998

5- Winnipeg Transit System, "*60 Pembina- Timetable*", Copyright 1999 City of Winnipeg Transit System, http://www.winnipegtransit.com/TIMETABLE/TODAY/60-OB/bottom.html

6- Wright Bus, "*Manufacturers of advanced performance buses*", http://www.wrightbus.com/nonflash/history5.htm#

Section 7- GPS Receiver Output Analysis and Real-time Positioning Methods

7 GPS Receiver Real-time Analysis and Report

In this section we investigate how the output of a GPS receiver, installed in a bus operating on route 60 – Pembina, is analyzed and the real-time positioning information of the bus reported to various users. The users include: 1) the bus driver operating the bus, 2) transit system administrators, and 3) others who have access to the route information via the Internet. The users may be connected to the Internet through facilities installed in bus stops, or elsewhere. They may also carry handheld devices capable of making wireless connections to the Internet. The applications developed for the so-called "connected" users are discussed in Sections 9 and 10. This section covers the analyzer application, and reviews the algorithm for the GPS receiver output processing alone and also in connection to the other auxiliary methods used in the system. The application works like an interface between the positioning sensors (GPS, signpost, and direction of movement indicator), and the positioning real-time database.

7-1 Real-time Positioning along with Scheduler Positioning Information

Recalling the main aim of this project that is providing the real-time positioning information of buses operative in a typical scheduled bus route, the following two properties of a scheduled transit service may be considered to be of the most interest;

1. The arrival time of an operative bus to any location of its road can be directly or indirectly extracted from its schedule at any given time. Similarly, the exact or approximate location of a scheduled bus can be obtained form its timetable.
2. As long as a bus follows its schedule, there is no need for it to report the data of its location to the users, as the data is inferable from its timetable.

In this project, data stored in database tables has basically two different origins. The main part of it reflects the scheduled information of the transit system, which is not real-time. It has already been prepared and announced by the transit system to let the passengers know about the expected arrival times of the buses. The other part of the database, however, is the real-time information

about the locations of the operative buses when they are out of their schedule. The real-time information works as a complementary part to the scheduled information and needs to be reported (and/or updated) only when there is an out of the schedule event happening to an operative bus. When used along with each other, scheduled and real-time database tables provide the required information to precisely locate the operative buses at any point along the road at any given time. Thereby, using pre-determined scheduled data of bus routes saves the communication channel bandwidth from transmitting unnecessary data. On the other hand, the user does not need to relay, solely, on non-active information of the route timetables if for any reason the promised services can not be provided in the way already announced.

As will be described in section 9, the connected users of the transit system have access to the real-time positioning information database, as well as to the scheduled one. A server application retrieves the location information of the operative buses from the both database tables and sends them back to the clients upon their requests.

7-2 Positioning Data Processing Algorithm

From the above discussion it is concluded that the following algorithm can be used to provide the user with the accurate positioning information (and the time of arrival) of a bus operative in a scheduled bus route:

1. The positioning data of an operative bus is obtained in a real-time basis within the bus.
2. The information is compared with the position information inferred from the bus schedule.
3. If there is a discrepancy beyond a threshold, the real-time positioning information becomes available to the transit user. Otherwise, the route schedule reflects the bus whereabouts.

7-3 Positioning Auxiliary Methods

In this project, the main positioning probe is a GPS receiver. However, since its output is not reliable in downtowns that are crowded with high-rise buildings, other positioning techniques are

also employed to obtain more precise positioning information. There are two methods for correcting GPS receiver output, used in this project, that are enumerated as follows;

1. DGPS Method

   In this method a reference GPS receiver installed in a known location is used to determine GPS positioning measurement error. Since the reference GPS receiver is in the same area as the other rover GPS receivers are, they all experience the same amount of measurement error. The error is defined as any difference between the coordinates of the known position of the reference receiver with the position indicated by its output. The method, called "position domain differential strategy", is not the one typical in DGSP methods which involve broadcasting correcting radio signals. It is, however, precise enough for double checking the output of the ordinary GPS receivers, which might suffer from errors with magnitudes more than a threshold value in specific situations. Sub-sections 4-4-1 and 4-4-2 describe DGPS methods in general and the way it has been used in this project, in more details, respectively. In sub-section 7-5 the DGPS mechanism used in the present application is investigated in more details.

2. Bluetooth Signpost Method

   When the output of a GPS receiver is detected to be highly erroneous, or not available at all, even applying the corrections provided by the DGPS method may not be useful enough to remove the errors and provide correct positioning information. Such situations usually occur when a GPS receiver enters the shadow of high-rise buildings that occlude large parts of the sky. Therefore, the receiver is not able to receive signals from enough satellites. In such conditions, signpost-positioning methods can be used to find out where the vehicle is. Section 8 introduces and discusses the way that Bluetooth technology may be used to determine the locations of moving vehicles in the areas where GPS/DGPS methods fail to provide correct positioning information.

7-4 Decision Making about the Report Time of the Out-of-schedule Positioning

Depending on the road conditions even when a bus is out of its schedule, there might be a chance for the bus operator to compensate for the discrepancies between the bus real-time location and where it should be depending on its schedule. Since, ultimately, it is the time of arrival that is of interest (rather than the bus location, itself), when an out-of-schedule positioning status of a bus is detected, the event does not need to be reported immediately.

In other words, the real-time detected out-of-schedule positioning discrepancies are important only when:

- They lead to arrival times different from that reflected in the timetables,
- The amount of difference is more than a threshold value.

As mentioned earlier, applying such a consideration in categorizing the detected out-of-schedule bus locations saves the communication bandwidth.

Making decisions about the time that an out-of-schedule positioning status should be reported, however, is not straightforward. To address the inherent ambiguity that exists in the problem, Fuzzy Logic methods can be employed in this system design. The usage of this method is described more deeply, in sub-section 7-5-1 below. However, before reaching that step, in the following sub-section, the implementation of other components of this application is discussed a bit further to provide a better understanding of how the real-time positioning procedure is completed.

7-5 GPS Receiver Output Analyzer - Application Development

Based on the materials mentioned in the previous sub-sections, the application developed for the purpose of this part of the project is constituted of the components shown in Figure 7-1.

The block "GPS Receiver (Simulator)", shown in the figure, represents either a real GPS receiver installed onboard, or the simulator application described in the Section 6, generating the positioning information of the moving vehicle that follows the timetable of bus route 60-Pembina.

The output of the GPS receiver corresponds to a point on the bus path, in Figure 5-1, if there were no error involved in the GPS positioning system. Since the GPS output suffers from error originated from different sources [Section 4], at any given time, the point indicated by the GPS output is not exactly where the bus is on the road. On the other hand, for the reasons discussed in Section 5, in this project instead of a regular digital map a cartoon-wise map is used to represent the locations of the operative buses. As indicated in Table 6-1, a time-location table can be developed capable of determining the corresponding points on either map based on the information provided by the route schedule, at any given time.

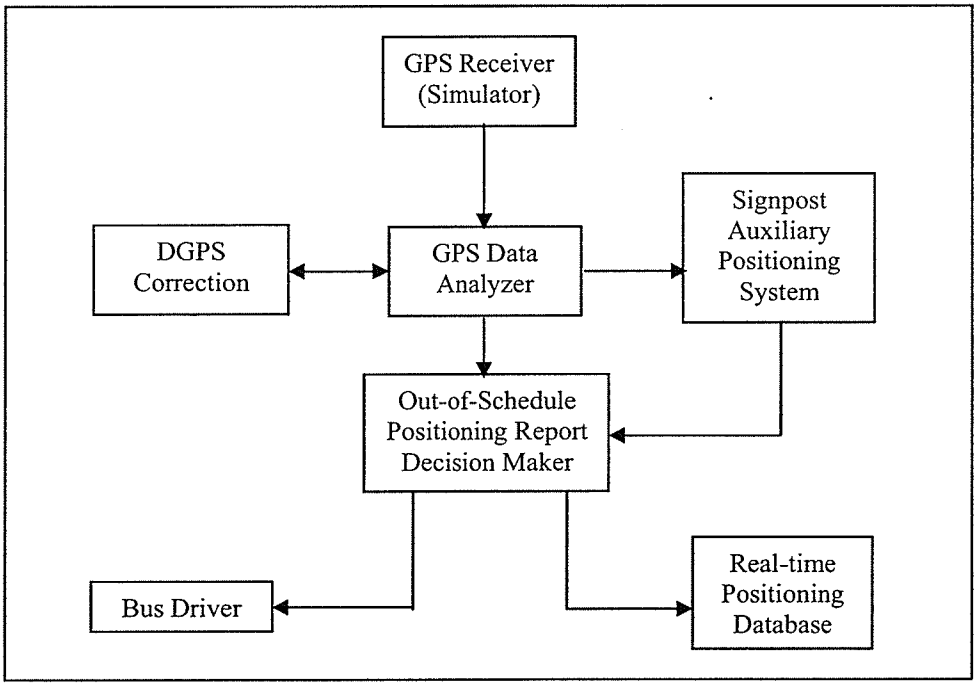


Figure 7-1- Functional components used in developing the GPS output analyzer implemented in this project.

In Figure 7-1, the block "GPS Data Analyzer" represents a software module that compares the "GPS Receiver (Simulator)" output with the corresponding scheduler points of the bus path. The module may use two blocks "DGPS Correction" and "Signpost Auxiliary Positioning System" to increase the accuracy of the comparison operation. The results of the block "GPS Data Analyzer"

is then used by the block "Out-of-Schedule Positioning Report-Decision Maker" to make discrepancy reports. There are two reports about the schedule discrepancy that should be made in this system. One report is made to the bus driver and is simply a message to indicate how well the bus is following its schedule.

This message can be renewed in time intervals as short as few seconds and let the driver take the required action to compensate for any discrepancy from the schedule. The other report, which can be made every minute, is the one received by the users reflecting the existence of discrepancy considered not compensable at the time that the report is made.

Since the point indicated by the GPS receiver output might not be exactly on the bus path, a point on the path having the minimum distance from it is selected as the representative of the bus location on the digital map. Figure 7-2, shows this situation, more clearly.

In this figure, point "O", shown as black square, corresponds to the output of the GPS receiver installed on the bus located in point "A". There is a distance "e" between points "A" and "O" that is equal to the error experienced by the receiver. If the error was known the intersections of the road with a circle centered at "O" and having radius "e" could lead to points "A" and "B", shown as gray squares in the figure. Based on the previous locations of the bus, the road conditions, the segment mean speed, and having more consistency with the bus timetable, one of the points of "A" or "B" could be selected as the real location of the bus and the other one would be rejected.

However, since the amount of the instant error is unknown, a direct leading from where the GPS receiver output shows (point "O") to where the vehicle is really located (point "A") is impossible. On the other hand, point "O'" estimates where the bus (located at point "A") is, and represents the closest square of the road to the point "O". As shown earlier in Figure 6-4, each square in Figure 7-2 is the digital map, which corresponds to a mosaic on the cartoon map, for which the timetable information is available through Table 6-1.

In reality the error "e" is much less than what shown in Figure 7-2 and the circle might cover fewer squares. In addition, the point of the road having the minimum distance from the point represented by the GPS output might be somewhere out of the predetermined squares of the digital map.

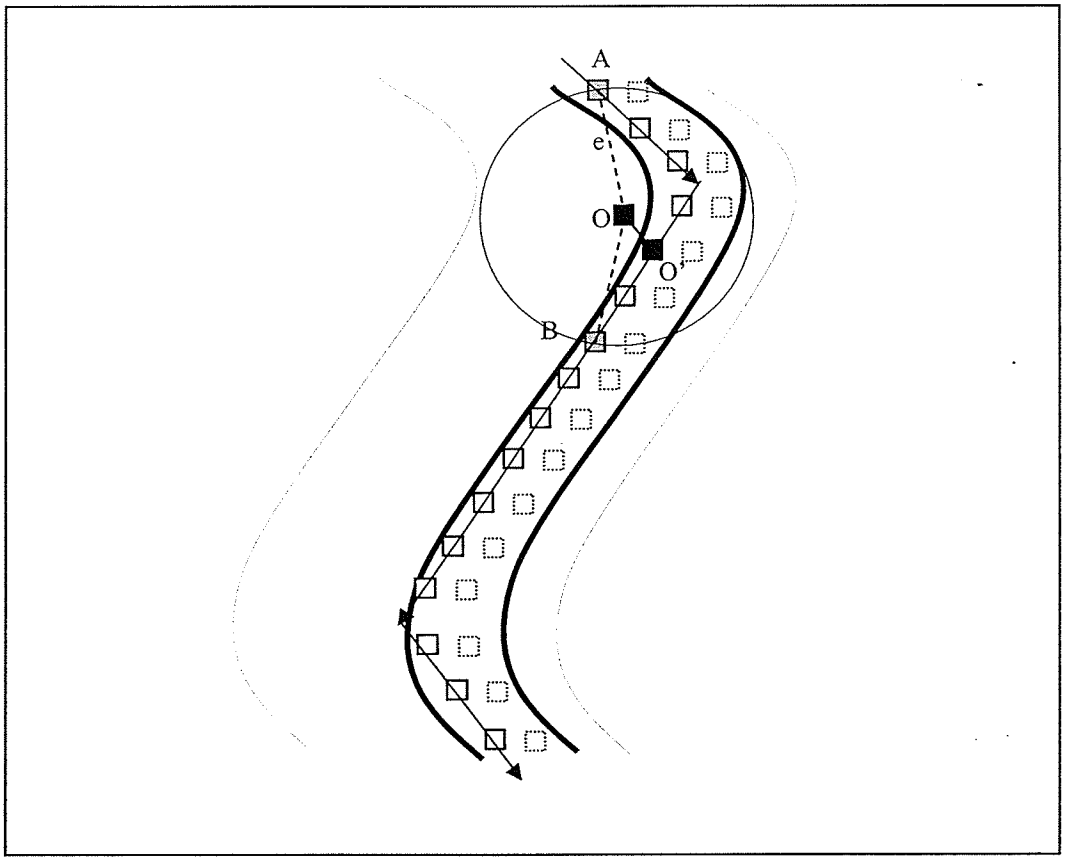


Figure 7-2 – The closest point of the road to the position measured by the GPS output is selected as the primary representative of the bus location.

However, since the squares are the only locations of the map for which the scheduler information is available in a more straightforward way, as presented in Table 6-1, the one having the minimum distance from the GPS receiver output is selected to be the on-road estimated location of the bus.

It should be noted that by using interpolating methods, tables can be created having more detail. Such tables could be used to locate the points of the map having the absolute minimum distance from the points represented by the GPS receiver output. However, since the system has minute resolution, finding the point of the road having the absolute minimum distance provides no further advantage, particularly, because the selected point in table 6-1 is only considered as a primary estimation.

In Figure 7-2, there are two dashed-lines, in parallel to the road bold lines, creating a margin area around the road where the points out of it have distance from the road more than a pre-determined threshold value equal to the width of the region. The GPS receiver outputs that represent points out of the mentioned regions are recognized as totally erroneous values and are not accepted for further processing. In such conditions, other auxiliary positioning methods, introduced later, can be used for acquiring the locations of the vehicles.

Figure 7-2, deliberately, shows a special case where a large curved part of the road is covered by the error circle, to exaggerate one of the worst scenarios of the GPS receiver output error with respect to the road topologies. As mentioned earlier, in reality the radius of the error circle (20 meter, in normal conditions) is much less than the road curves. Another unacceptable situation with respect to the road curve may be encountered when there is more than one point having the minimum distance.

Generally speaking, in this implementation, a point of the road is accepted as a primary estimation when all the three following requirements are fulfilled.

- The point on the road has the minimum distance from the GPS output,
- The point is at the next part of the road with respect to the bus's moving direction i.e. it is not in the region of the road that is already passed by or on opposite side. This condition is shown in Figure 7-2 by using the segmented arrows to emphasize the direction of motion, and also by illustrating the squares in the opposite direction with dashed-line borders.

- A new estimated point should be in an acceptable distance from the bus's previous location, reachable by the vehicle when it is driven with the maximum permitted (or expected) speed.

If the above conditions are not satisfied auxiliary positioning methods are used for re-evaluating GPS receiver output.

The first auxiliary method is DGPS as explained in sub-sections 7-3 above, 4-4-1, and 4-4-2. Working as a client (in the context of the client/server paradigm) this application inquires about the corrections it should apply to the GPS receiver installed on the bus. The server application, that works in accompaniment with a GPS receiver installed at a central fixed point, replies with the correcting values of latitude and longitude. In the central station, the output of the GPS receiver is compared with its known fixed latitude and longitude. The difference that should be within regular GPS receiver errors is sent to the mobile GPS receiver, installed onboard the buses in the same area (city), upon their request.

To implement the above mentioned scenario, in the simulated server application, the regular GPS receiver errors (20m) is randomly fragmented to two parts and then changed to their latitude and longitude equivalent. The values obtained are considered as errors measured by the central (fixed) GPS receiver. When a mobile GPS receiver's output is detected as being erroneous, as described earlier, the correcting latitude and longitude values are requested, obtained and to the mobile GPS receiver's measurements. The result, so obtained, is error free from the regular GPS receiver's point of view. If after this correction the output is still unacceptable, then it means that error factors other than atmospheric ones – like sky occlusion - are involved. In such conditions the GPS reading can not be of any more help and other positioning methods should be employed, if possible. Here is where the second auxiliary method, introduced in Section 8, which employs Bluetooth technology capabilities comes into action.

When the mentioned positioning operations, including applying the required corrections, are completed the result is considered to be as the best positioning estimation of the bus on the road.

This result is compared with the vehicle's location, as it should be, according to its schedule. If it is found that the difference between the two locations can not be compensated for, within a determined time interval, the present location of the bus is sent to the real-time positioning database.

Making decision about whether or not a discrepancy between the real-time and scheduled locations of a bus is compensable is discussed, more fully, in the next sub-section. The data from the real-time positioning operation is stored in its corresponding database table. Sections 9 and 10 describe how the mentioned data is processed further to become available for the transit users over the Internet.

7-5-1 Using Fuzzy Logic Methods in Making Real-time Positioning Reports

In this sub-section fuzzy logic methods are discussed that can be used in this application to decide about the suitable time that a real-time positioning report should be made to the corresponding database table. To save room for the other materials introduced in this thesis there is no explicit introductory provided for this topic. References [2-6] may be consulted in this regard.

So far, it has been shown that:

1. The bus route timetable can be used to reduce the number of times that the real-time positioning data should be reported to a central database, simply because the timetable may adequately provide the required information of where the buses are expected to be. Therefore, as long as the bus is following its timetable, the real-time positioning data provides no additional information.
2. The real-time positioning data is informative if and only if there is a difference between a bus real-time position and its schedule that leads to an off-schedule event observable by the users at bus stops.
3. The real-time positioning procedure is completed in approximately every second, however, the positioning report, when needed, is not required to be made more than once a minute.

4. The timetable has one-minute time resolution. Therefore even though the position discrepancies under a minute are measurable or estimated by this system, no report is required to be made to the users.

At this point an attempt is being made to establish and implement a method to save the communication channel from unnecessary usage. The discussion presented in this sub-section only introduces the way that the problem might be generally approached. Though the results of this discussion has led to implementation of a software module "Out-of-Schedule Positioning Report-Decision Maker" shown in Figure 7-1, the assumptions made here, specifically with respect to the fuzzy functions, should be refined in real cases.

Now let's look at the problem of reporting the real-time position of a bus to a central database via a communication line, more closely. Assume that in the system implemented so far the distance corresponding to one-minute time difference between real-time and scheduled positions is the selected threshold to make the real-time positioning report to the corresponding database table. The correspondence between time and distance is made using the mean speed, as the transform coefficient, of the transit buses on a selected route. Therefore, for a bus that is scheduled to be in a bus stop in 1 minute and also detected to be 1 minute behind its schedule, the system should show that the bus would arrive to the desired destination in 2 minutes. Since the system's time resolution from the user's point of view is one minute, it is not clear if a bus announced 1 minute behind its schedule has 1 minute and 1 second or 1 minute and 59 seconds latencies. The system treats both delays in the same way, and it doesn't help if the threshold value is changed, because the method of delay categorizing still remains the same. A similar problem exists when the bus is ahead of its schedule. The latter case is even more important because it is more likely that a transit user misses a bus that is ahead of its schedule than one behind schedule.

As can be seen here, the main problem is that in the system designed so far all the time instances within an interval are treated exactly the same. For example, all moments of delay beyond the

threshold value are considered to have the same weight of importance to be reported to the real-time database table. In reality, however, when a discrepancy is detected between the bus real-time and scheduled positions, there might be a chance for the bus driver to compensate for it, before it is needed to be reported to the database. In the above example, the one minute and one second delay may be compensated within a fraction of a minute. Therefore, there would not be any need to send the real-time positioning data to the database, because the bus can reach to the bus stop on schedule. The 1 minute and 59 seconds delay, however, may have much less chance to be compensated for in a 1-minute time interval. To solve this problem there needs to be developed a system that attributes distinct weights to different amount of delays within the same category of latencies. And here is exactly where fuzzy theory may be useful.

The rules and the attributed membership functions resulting from applying this theory to the experiment (or process) under investigation are found by measurement, observation, and, by subjective estimation based on (human) experience and intuition. The theory, announced for the first time in 1965 by L. A. Zadeh, provides a simple way to make definitive decisions from ambiguous information, as shown in the following paragraphs.

Figure 7-3 above shows the algorithm of detecting, processing, and reporting the discrepancies between real-time and scheduled positioning of a transit bus, as implemented in this project. It also shows where the fuzzy rules are applied in the processing steps.

As shown in the figure the real-time location of the bus is compared with its scheduled location, and if the difference is less than a threshold value (1 minute) the result is reported to the driver as "on schedule" message. If a discrepancy more than the threshold value is detected the system checks if the bus is ahead of its schedule. If this is the case, the system reports the status to the driver by the following message: "1min ahead of schedule". The "1min" part of the previous message may be replaced with other values, if it is found that the time difference is more than that. The time resolution, however, remains 1 minute as already emphasized. If none of the above

cases happens, the bus is considered to be behind schedule and the status is reported to the driver with a message like "1min behind schedule", again with the chance that "1min" is replaced by the real-time difference.

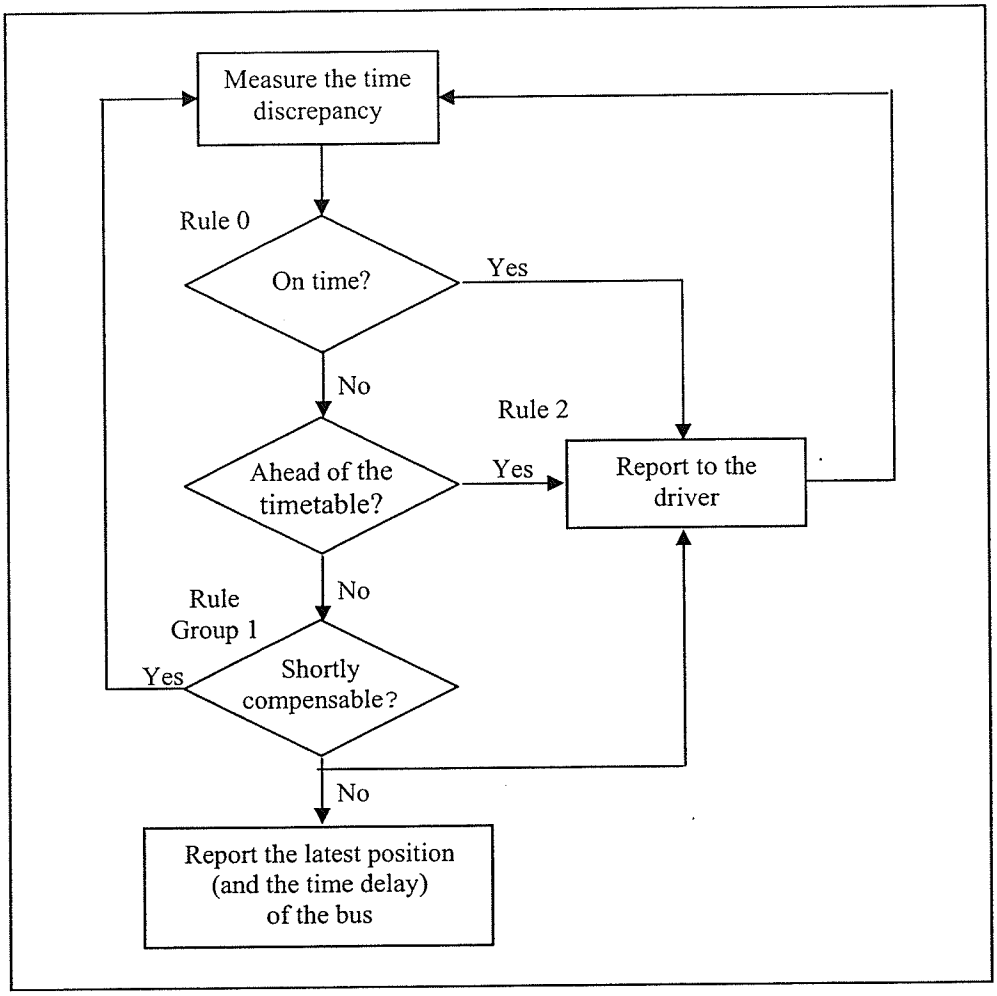


Figure 7-3 – The out-of-schedule report algorithm using fuzzy logic rules.

In Figure 7-3, there are three (groups of) rules representing the potential places where that fuzzy theory could be used for resolving the encountered ambiguity. Among all rules presented in the figure, this sub-section only discusses the implementation of. Rules (group) 1.

Rule 0 defines the threshold of the time discrepancy beyond which the bus may be recognized out of schedule. Though the threshold could be defined in a fuzzy way, in this implementation it is dealt with decisively. Instead, its fuzzy characteristics are shifted one step further and attributed

to the factors governing the delay compensations represented by Rule (group) 1, as described more fully in the next paragraphs. Rule (group) 2 is also almost governed by the same factors as Rule (group) 1. However, since adjusting the movement of a bus to its schedule, when it is ahead of its timetable, is usually much more easier for a driver than compensating for its delay, Rule (group) 2 is not talked about any further in this project. It should be emphasized that in the former case, depending on the amount of time discrepancy, the driver only needs to reduce speed or stop in the first possible bus stop until the required amount of time is elapsed. As a matter of fact, this part of the application is only designed to let the driver know about the situation and encourage her to take the required actions to compensate the time difference. Though the action of the driver involves inherent fuzziness, the following discussion about the delay case covers main aspects of it.

From the discussion presented earlier, it could be concluded that the two delay margins in the same category of latencies might be distinguishable in terms of their compensabilities. There could be at least two parameters, among others, involved in how fast a delay might be compensated: 1) the time of the day, and 2) the location of the road that delay is detected. Other parameters such as driver's response time, their attitude when they receive the delay message, cause of delay (such as traffic jams resulting from accidents), and so on, are also involved. However, since they are usually out of our control (to some extent), they are not considered major parameters in this discussion. Driver's manner is somehow controllable by defining protocols of response to time discrepancies reports. However, it adds a new function block to the architecture of this ITS system, beyond the scope of this thesis.

To see how the major parameters mentioned above might affect the compensability of a detected delay the following fuzzy rules are set forth their validity evaluated based on logic concepts by breaking it down to a group of rules. The rules in their preliminary form may be set as follows:

Rule-1: IF it is not a busy time of the day AND it is an area of the road where the bus is highly mobile, THEN the time discrepancy is compensable.

In Rule-1, the evaluation result of the statement "it is not a busy time of the day" may represent whether or not the discrepancy should be compensated within the rush hour time interval of the day. On the other hand, if the evaluation result of the statement "it is an area of the road where the bus is highly mobile" indicates that the bus may have enough mobility to compensate for the discrepancy. The conclusion part of Rule-1, which is the statement "time discrepancy is compensable", should also be evaluated on a fuzzy logic basis.

The second statement in the conditional part of Rule-1: "it is an area of the road where the bus is highly mobile" reflects the physical conditions of the road that affect the average velocity of the vehicle. The physical road conditions, as long as the delay compensation is concerned, may be determined by the factors like:

- The maximum permitted speed of the road,
- Number of intersections with other roads that lead to existence of frequent traffic lights or stop signs,
- Number of bus stops close to each other (usually the case in downtown areas),
- Number of available driving lines and the frequency of line changes encountered by the vehicle, and
- Road construction area.

The effect of such parameters on physical road conditions may also be time dependent. This time dependency is however different from the rush-hour effects on road conditions that are applicable all over the road in the same way at certain time intervals of the day. In the former case the road's physical conditions may be affected by activities that are more location oriented than being time oriented. For example a bus may experience lower mobility in the parts of the road that are in

vicinity of a cinema or a church at certain days of week and/or specific times of day, in time intervals other than the road's general rush hour times.

Now, let's break down Rule-1 to a group of rules having more specific conditional statements and corresponding results. Let's denote the time of the day by busy (B) and not busy (NB), and categorize various areas of the road as regions in which a transit bus has low mobility (LM) and high mobility (HM). The time discrepancy in the conclusion part of Rule-1 could also be categorized as compensable (C), somewhat compensable (SC), and not compensable (NC). Using the defined nomenclatures we may set forth the following group of rules, when a bus is detected to be at least one-minute later than its schedule:

Rule 1-1: IF B AND LM, THEN NC

Rule 1-2: IF NB AND LM THEN SC

Rule 1-3: IF B AND HM THEN SC

Rule 1-4: IF NB AND HM THEN C

Table 7-1 summarizes the above fuzzy rules in terms of the conditions and results.

	NB	B
LM	C	SC
HM	SC	NC

Table 7-1 – The fuzzy rules, conditions and results as used in this project.

Figure 7-4 below shows a member function for the fuzzy statement "busy time of the day" and reflects the road's rush hours during the operation time of the bus route.

Figure 7-5 below shows how the membership function of the fuzzy statement "an area of the road where the bus is highly mobile" may be illustrated in a quantitative way. For a two-dimensional variable like positioning coordination a three-dimensional fuzzy diagram may be considered more appropriate. However, as explained earlier the road's physical conditions are actually among location dependent specifications.

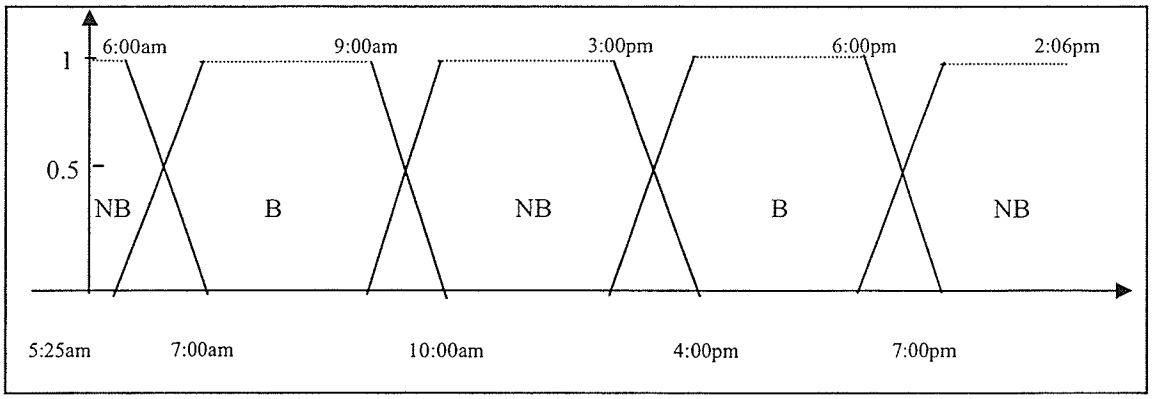


Figure 7-4 – The suggested fuzzy member function of the Busy and Not-Busy hours of the day during the operating time interval of the bus route.

Particularly, in the case of rout 60-Pembina only one dimension (the vertical component of pixel coordination of the cartoon map) is adequate to differentiate parts of lower mobility of the road from the parts of higher mobility.

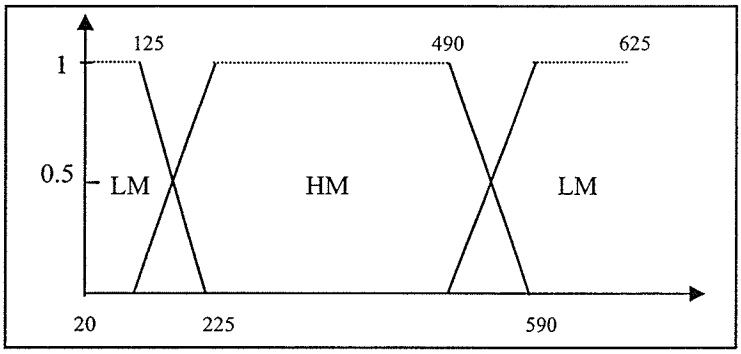


Figure 7-5 – The suggested fuzzy member function of the Low Mobility and High Mobility parts of the bus route, determined by only one component of the coordination system.

In this implementation, when a bus is detected that is 1 minute (or more) behind its schedule, depending on the time and the location of the measurement and by using the above mentioned fuzzy membership functions each member of Rule (group) 1 is evaluated. The compensability membership function, shown in Figure 7-6, it is checked whether or not the delay is 50% or more compensable (that, in this case means, the delay would be compensable in one minute or less). If this is the case the delay is not reported to the real-time database table and the communication channel is saved from unnecessary usage.

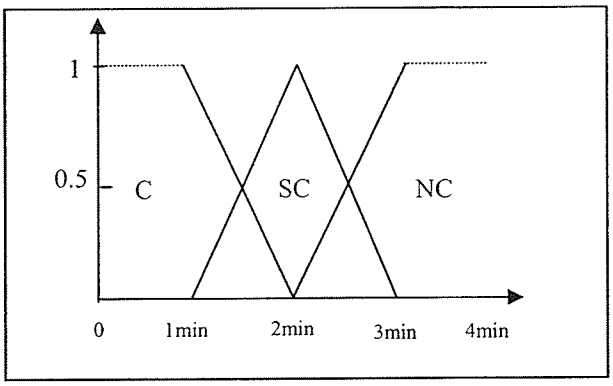


Figure 7-6 – Compensability member function as defined in this implementation.

It should be emphasized again that this implementation only suggests a way that the compensability problem might be solved or approached. The values used in the membership functions are supported by the information extracted from the route timetable and set up intuitively. The optimum values, however, should be obtained by running real experiments.

7-5-2 Implemented Programs

In this sub-section the software modules for performing various operations of the analyzer application are briefly reviewed and important specifications of them emphasized.

The analyzer uses the data generated by the GPS receiver simulator software described in the previous section, as its main input. For simulation (and/or demonstration) purposes an error-worsening factor has been added as an adjustable input that could affect the simulator output in a desired way. The application uses a text file to simulate clock output with minute resolution, as the main synchronizing device for comparing the GPS receiver simulator output with the bus route 60-Pembina timetable. To correspond the simulator's output with a point on the road, the software selects the point on the road that has the following specifications:

1. It is among the group of points of the digital map for which the times that the operative buses should reach them, based on the route timetable, are already known.
2. It has the minimum distance from the point indicated by the GPS receiver (simulator) output.

3. It is not 1000 meters further than the previous location of the bus on the road (assuming that the bus has 60Km/h speed, it can not go beyond that distance in a minute. Recall that the time resolution is 1min i.e. the simulator and the analyzer go through the steps of one minute in each assessment cycle. The one-minute hops can be adjusted to occur in seconds, simply by modifying the time interval that the simulator waits in each assessment loop before going to the next one.
4. It is not 300 meters further than where the GPS simulator output shows, otherwise, it is concluded that the output is more than erroneous than acceptable.
5. It is not selected from the points within the area already passed by the bus or on the opposite side of the road.

If a point can not be found satisfying the previous conditions the GPS receiver (simulator) output is to be corrected using the DGPS method. A DGPS server is contacted and the correcting data is received and applied to the GPS output. The above mentioned steps are performed again to find the point on the road.

If the criteria still can not be met, then the application uses a Bluetooth auxiliary method solution. The output of this method is not further evaluated simply because it reflects the closes signpost or bus stop to the bus.

When the real-time location of the bus is estimated, it is compared with where it should be according to its schedule. In fact, the location distance between two points is transformed to the time between them using the bus average speed (calculated as 251.67 meters/minute in the case of route 60-Pembine).

The point representing the GPS output, the closest road point to it, (the primary accepted estimation) and the point representing the best estimation of the bus real-time location are all presented on the digital and cartoon maps on the simulator's UI. The UI provides a user a better visual demonstration of the way the simulator works.

If the time distance is more than 1 minute the application goes through the steps explained in the previous sub-section to appraise whether or not the discrepancy is compensable. If it is not compensable in a minute or less, among other things, the real-time location of the bus is written in a text file (gps.txt). The text written in the file has the following format:

[X [Direction] Y] [Bus ID] [Time Delay].

In the [X [Direction] Y] part of the above statement, X and Y represent the coordination components of the real-time position of the bus over the cartoon map. [Direction] shows the direction of movement of the bus to distinguish its movement on opposite sides of its path used for later processing. [Bus ID] and [Time Delay] represent values corresponding to what they imply.

If the bus is not operative or the amount of time discrepancy is not beyond the thresholds the content of the text file will be changed to blank.

A Java application is used to read and transfer the contents of the gps.txt file to a database table devoted to the real-time positioning information of the bus route for further usage as will be described in Section 9. Figure 7-7 shows the applications and related files developed for this part of the application.

If the Java application, labeled in Figure 7-7 as "Real-Time Positioning Report - Java Application" finds the file non-blank the real-time positioning information in it is transferred to the database. Otherwise, it sends "null" to the database, and will not communicate with the database any further information until another non-blank line of information is found. This method saves the communication channel from unnecessary usage.

So far in this section it has been shown how the vehicle's positioning information is gathered, processed and transferred to a central database system. There might be many more real-time information of an operative bus that would be of interest to a central monitoring system. Among

them the number of passengers, drifting error from the schedule, the physical status of the bus and so on may be included.

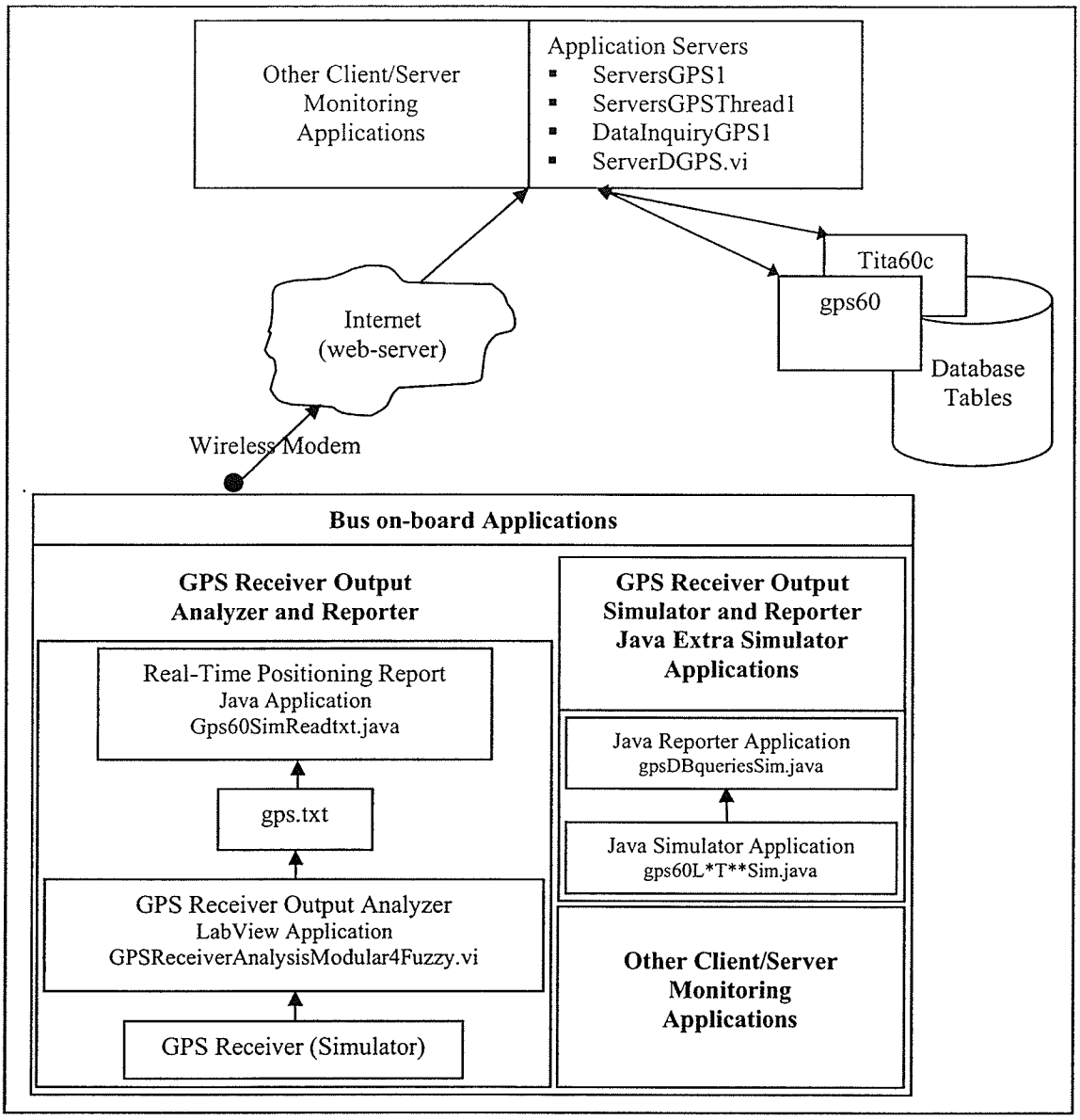


Figure 7-7 – The applications and related files developed for the "in-bus" part of this system.

In addition to the software discussed so far, for the GPS receiver output analysis and real-time positioning reporting, there is one simple monitoring simulator program (and its related files). There is no additional explanation about the work they do. They are presented here to emphasize

that this thesis can provide the basic required framework upon which more complicated central monitoring system of the transit buses may be developed.

7-5-3 Related Files

- GPSReceiverAnalysisModular4Fuzzy.vi
  - PresentTimeStartupforSimulator.vi
  - ClockStringScheduledHoursUsingTextFile.vi
  - Clock String file (ClockOutputSimulator24hoursFormatCoincideBusSchedule.txt)
  - GPSReceiverOutputSimulationRout60ScedulerMovement3.vi
  - LatLonExtractorDistanceMeterUsingRowNo1.vi
  - GPSdataanalyzer1.vi
  - Direction of Movement File (NorthtoSouth)
  - Direction of Movement File (SouthtoNorth)
  - ClientDGPS.vi
  - ServerDGPS.vi
  - GPSReceiverOutputSimulationRout60ScedulerMovement4.vi
  - Blutooth Time-Location table (Bluetooth_R60DMap_CMapCorres_N_S_Schdled2_Rowed1-2.txt)
  - FuzzyTimeoftheDayNBB.vi
  - FuzzyPartoftheRoadLMHM.vi
- R60DMap_CMapCorres_Intrplt_NtoS_Schdled2_LocationErrIncl_Rowed1.txt
- R60DMap_CMapCorres_Intrplt_StoN_Schdled2_LocationErrIncl_Rowed1.txt
- Gps.txt
- Gps60SimReadtxt.java
- BusInformation
  - ServerBusinf.vi

- Servergps.vi
- simulatorLocation.vi
- simulatorPassengers.vi
- simulatorSpeed.vi
- Clientbusinf.vi
- Clientgps.vi

7-6 References

1. J. Travis, "*Internet Applications in LabView*", 1st Edition, ,© April 15, 2000, Prentice Hall PTR
2. Y. Zhao, "*Intelligent Transportation Systems*", ©1997, Artech House Publishers, pp95-103, and 321-326.
3. H. T. Nguyen, Elbert A. Walker, "*A First Course in Fuzzy Logic*", 2nd Edition, ©1999, CRC Press
4. D. Teodorovic, KatarinaVulkadinovic, "*Traffic Control and Transport Planning – A Fuzzy sets and Neural Network Approach* ", ©1998, Kluwer Academic Publishers
5. S. D. Kaehler, "*Fuzzy Logic Tutorial*", ©1998, Seattle Robotics Society, http://www. seattlerobotics.org/encoder/mar98/fuz/flindex.html
6. L. A. Zadeh, "*Fuzzy Sets*", Information and Control, Vol.8, No3, 1965, pp. 338-353

8- Bluetooth Auxiliary Positioning Method

8- Auxiliary Positioning Method Using Bluetooth Wireless Technology

In Section 7, the Bluetooth wireless technology was introduced as the final auxiliary positioning method to provide the location of buses when GPS and DGPS methods are incapable of providing correct results. Particularly, adjacent to large buildings, where a large portion of the sky might be occluded, there would not be enough satellite signals reaching the GPS receivers. In such conditions GPS receivers would not be able to fix their location and/or the output might be totally erroneous.

To resolve the positioning problem, at the locations along the bus path where GPS receivers may fail to provide correct results, Bluetooth positioning signposts can be installed. They can provide passing buses with the precise location along the road where they are. Buses equipped with the Bluetooth transceivers can automatically acquire the positioning information when they come to the proximity of the signposts, and use the information in the same way that they use the positioning information provided through their GPS receivers.

In the following subsections, the specifications of the Bluetooth wireless technology, directly related to the purposes of this project, are briefly reviewed. Thereafter, employing Bluetooth signposts in accompany with other positioning systems used in this project is discussed. Finally, a new ITS system is introduced in which the Bluetooth signposts are employed as the sole devices of positioning.

8-1 Bluetooth

Bluetooth is a robust, low power, short range, and low complexity wireless technology that is intended to replace the cables between electronic devices to enable data and voice communication anywhere in the world. Its low cost makes it ideal to be used in the applications where mobility of electronic devices within a limited area is of major interest.

Bluetooth connectivity provides the devices and their users with the capability of forming ad hoc networks which makes the data exchange possible when the devices come into the vicinity of each other. The mentioned specification gives a location-oriented characteristic to Bluetooth

communication, which is the main subject of this discussion. Since such a specification is suggested to be used in moving vehicles, it is essential to show that the establishment of the communication channels between Bluetooth devices are fast and efficient enough to allow for data exchange, in short time intervals. In the following subsections the specifications of the Bluetooth wireless technology that are directly related to the purposes of this project, are briefly described.

8-1-1 The Bluetooth Protocol Stack

Figure 8-1 shows the components of Bluetooth protocol stack. In lower layers it centers around using the RF module, and in higher layers it reuses the transport protocols developed for other purposes and/or applications. Among all the layers shown in Figure 8-1, the Bluetooth Module and its corresponding components are of main concern in the following discussion. Specifically, we investigate the Bluetooth radio and baseband layers in more detail. We also talk briefly about other layers when they are directly related to the Bluetooth usage in this project. The reference list at the end of this section may be found to be useful for further consultation wherever this brief and specific overview is insufficient to clarify other interesting specifications of the Bluetooth technology.

8-1-2 Bluetooth Radio

Bluetooth radio is responsible for carrier generation, modulation and power adjustment. It is controlled by signals generated in higher layers and fed by a packetized data stream generated in the baseband layer. It operates in the unlicensed Industrial, Scientific and Medical (ISM) band at 2.4 GHZ.

Table 8-1 shows the Bluetooth channel frequencies used in the ISM band in the North of America. As shown in the table, 79 channels, each of which having 1 MHz bandwidth, divide the frequency spectrum. When communicating to each other, transceivers in Bluetooth devices use the whole 79MHz available frequency spectrum following a communication channel access

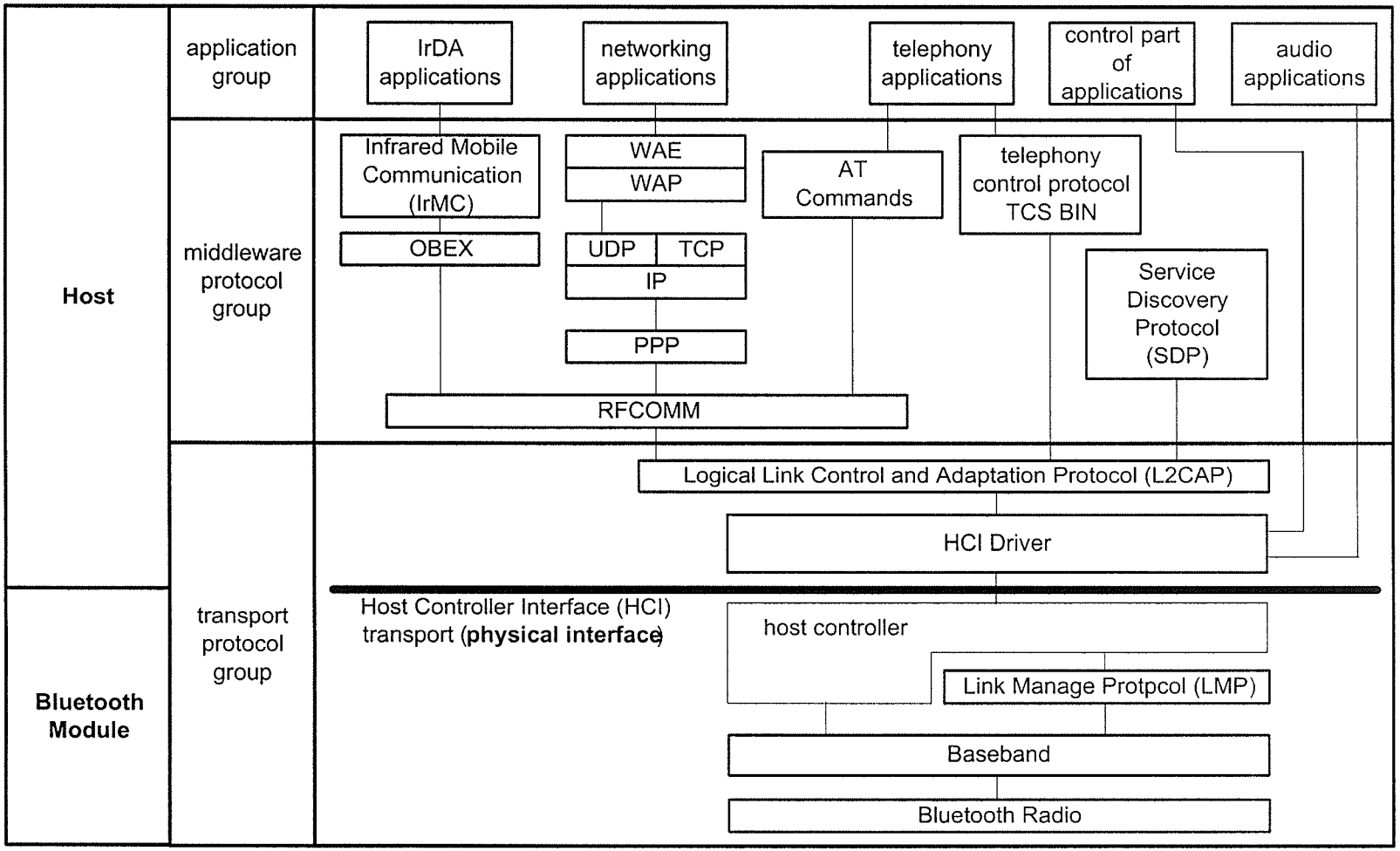


Figure 8 -1 - Host/Module Interface, Protocol Groups, and Protocol Stack in Bluetooth

scheme called Frequency Hopping Spread Spectrum (FHSS). To prevent interference the channel access scheme should be unique for the communicating devices.

Therefore, the receiver should know about the frequency-hopping pattern of the transmitter and also be completely synchronous with it to be able to maintain the communication link. In an established Bluetooth communication link, the device that governs the FHSS hopping pattern acts like a master and the device that follows the pattern and stays synchronous with the master acts like a slave.

Lower Guard Band (LGB)	2400MHz - 2MHz
Ch1	2402MHz - 1MHz
Ch2	2403MHz - 1MHz
Ch3	2404MHz - 1MHz
⋮	
Ch79	2480MHz - 1MHz
Upper Guard Band (UGB)	2481MHz - 3.5MHz

Table 8-1- License-free frequency allocation in the 2.4GHz ISM band in the North of America.

In higher level of communication, however, the devices are not aware of the role that they have already taken in lower levels and communicate to each other on a peer to peer basis. A communicating device can take either role of master or slave. It can be the master of one communication link while acting as the slave of other linkages.

The modulation used in the air link is binary Gaussian Frequency Shift Keying (GFSK) with the modulation index of 0.28-0.35, which is considered a non-complex design for the radio part of the technology.

Bluetooth transmitters are categorized to three classes of powers and/or coverage ranges, as follows:

- Class 1 Power: 0dBm(1mW) Range: 10m
- Class 2 Power: 4dBm(2.5mW) Range: 20m
- Class 3 Power: 20dBm(100mW) Range: 100m

In all classes the transmitter has the optional power control below –30dBm. The Bluetooth receiver has the sensitivity of 0.1% BER, with an input signal level of -70dBm or lower.

8-1-3 Baseband Layer

As shown in Figure 8-1, Bluetooth handles data and voice in two different ways. While data should pass through different layers of protocol stack, to be fragmented and packetized, the bit stream of digitized voice gets access to the baseband and radio layers more directly. In the baseband layer the bit stream of digitized voice should still be packetized. However, the packets carrying the voice information are completely different from the data packets. The mentioned packets are two major types of data units generated by the baseband layer. The data units called baseband packet data units (BB_PDUs) and have a maximum length of 2872 bits. Bluetooth radio is responsible for sending and/or receiving BB_PDUs through the air link. Each BB_PDU is sent or received over one of the 79 radio frequencies. Depending on the operational status of a Bluetooth device and the type of information - voice or data – the type of BB_PDUs may be different.

8-1-4 Operational States of a Bluetooth Device

Figure 8-2 shows operational states of a Bluetooth device. Two devices, that are connected and communicating their BB_PDUs, should agree on the frequency hopping pattern that they follow and be coherent to each other. To become connected one device should first know about the existence and collect the information of other devices in its proximity. Collecting information of a Bluetooth device, primarily, involves obtaining its fundamental elements that are its Bluetooth Device Address (BD_ADDR) and clock value.

8-1-4-1 Inquiry State

In the Inquiry State, a device that usually acts as a master in establishing the radio link broadcasts a short type BB_PDU to its vicinity to invite other devices to become connected to it.

The BB_PDU, called General (or Dedicated) Inquiry Access Code (GIAC, or DIAC), is a 68 bit packet conveying specific code of "0x9E8B3F" generated by the inquiring device. Each Bluetooth device has a unique 48-bit device address (BD_ADDR) that distinguishes it from other devices.

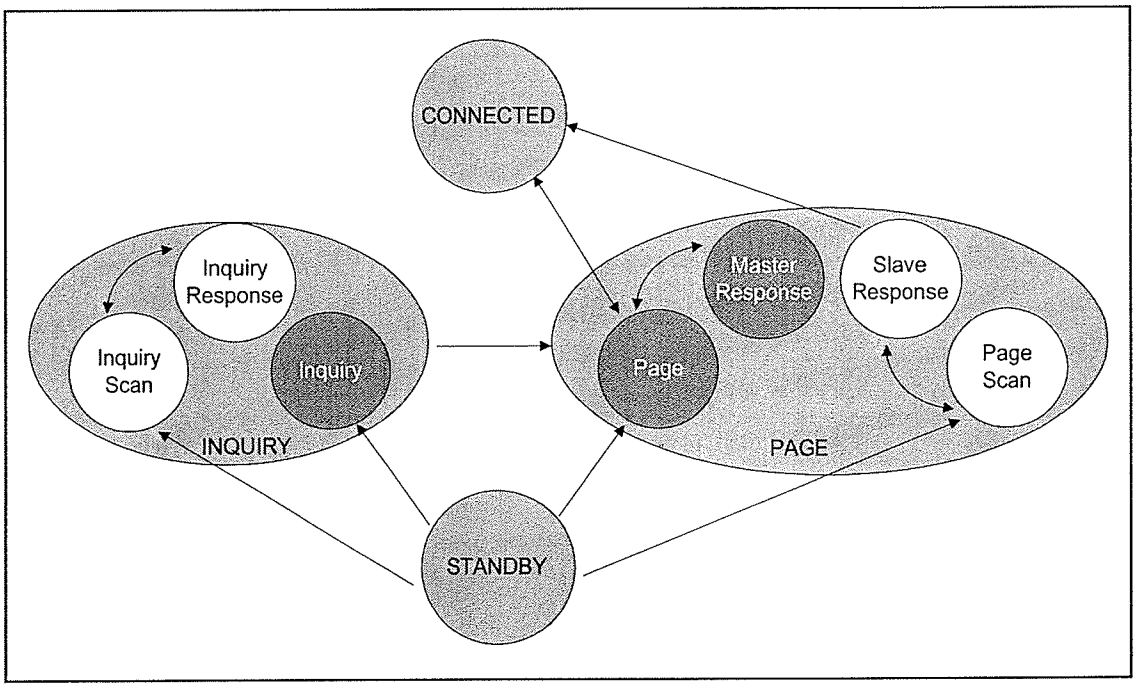


Figure 8-2- Operational states of a Bluetooth device.

The Lower Address Part (LAP) of the device - bits [23:0] – are generally used for generating the access code – bits [67:0] of BB_PDU. Therefore, the 68-bit access code can be used to identification distinct devices - or communication channels used by devices, depending on their operational status, as described later. In the Inquiry State, however, the access code does not reflect the LAP of the inquiring device, but its state of operation. That is why the GIAC of the inquiring device is also called the Inquiry ID.

In the Inquiry State, a device hops 3200 times per second – double the nominal hopping-frequency rate - on 32 channels distributed over the 79-channel frequency spectrum. Staying 312.5 μsec in each inquiry channel, the inquiring device uses two adjacent inquiry channels to send two 68-bit inquiry ID packets on each channel. Thereafter, it spends the next two 312.5 μsec time slots on the same frequencies, respectively, listening to the responses of its inquiries, coming back from the devices in its proximity. This means that the inquiring device sends an inquiry ID packet over a channel, and returns back to the same channel to listen to the response 625 μsec after the beginning of the time that the packet is sent.

A device should be in Inquiry Scanning sub-state to be able to receive and respond to an inquiry message. An inquired device spends 1.28 seconds in each inquiry channel to listen to the inquiring messages and upon receiving such a message enters Inquiry Responding sub-state to announce its readiness to be a slave of the inquiring device that takes the master role in establishing of the communication channel. Upon receiving an inquiry ID, to avoid BB_PDUs collisions, a slave remains non responsive for a time interval equal to the multiplication of a random number RN<1024 by the interval of a time slot of 312.5 μsec before responding the next occurrence of the same inquiry ID. When the mentioned time interval elapses the inquired device, which acts as the slave in this communication channel establishment, returns to the Inquiry Response sub-state. Upon receiving the next inquiry ID packet, the slave responds with a Frequency Hopping Sequence (FHS) packet. Having the length of 366 bits, among other information, the packet conveys two fundamental elements of the slave device: BD_ADDR and clock. The clock of a Bluetooth device is a 28-bit free running counter with the rate of 3.2 KHz that is non-stop and never adjusted. It determines the time slots and when the device should listen to or transmit the packets.

Considering that the gross data rate of the Bluetooth radio is 1 Mbps, it takes 68 μsec and 366 μsec for the inquiry ID and the FHS packet to be sent through the air link, respectively.

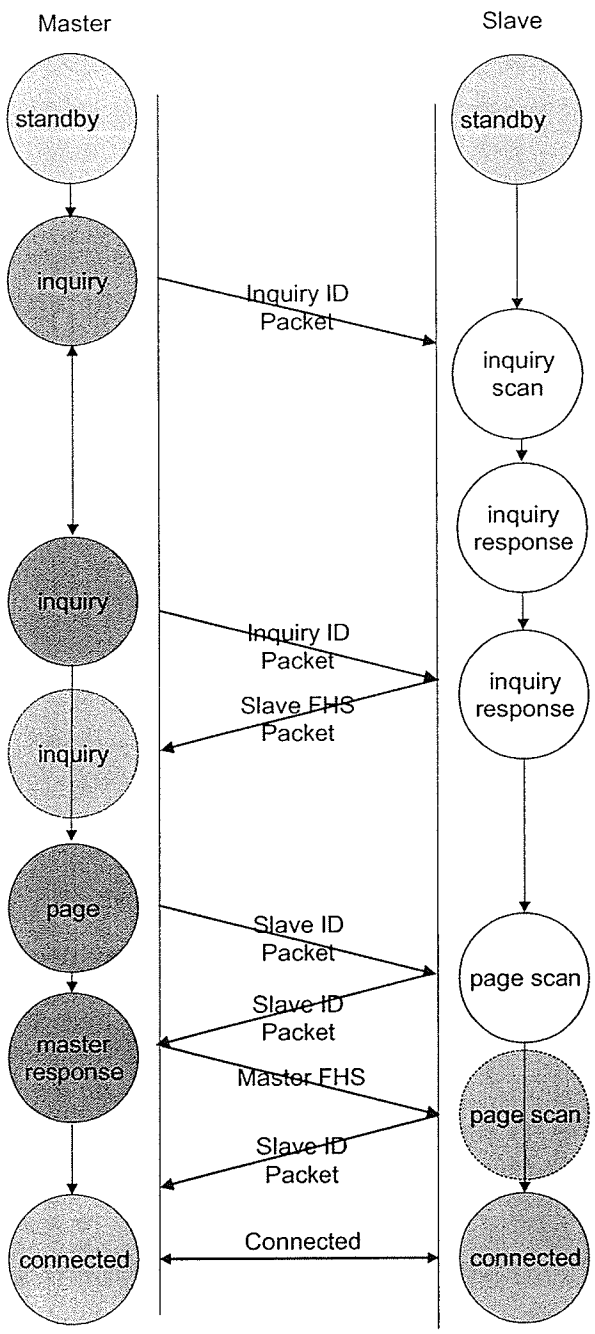


Figure 8-3- The procedure of establishing communication channel between two Bluetooth devices.

8-1-4-2 Inquiry Time Subtotal Calculation

Figure 8-3 summarizes the procedure of establishing communication channels between two Bluetooth devices, as partly described above and continued in the following subsection.

While the communication establishment scenario between two devices in the Inquiry State can not be altered, the channel scanning time intervals performed by the master may be different. For example in a 79-channel spectrum, a master typically sweeps just 16 out of the 32 inquiry channels over a 10-msec time interval 256 times, which takes 2.56 seconds. If it does not receive any response in the first round it repeats the sweep a second time before switching to the next group of 16 channels to perform the same procedure of inquiry ID packet broadcasting. The typical repetition value of 256, however, is controllable by the applications developed for controlling the Bluetooth device [1]. The ability to control the baseband or the radio layer behavior is shown in Figure 8-1 as the "control parts of the applications" categorized in the "application group" of the Bluetooth protocol stack.

Following the procedure mentioned in the previous sub-section, Table 8-2 shows that for a master and slave device to find each other in their proximity and exchange the primitive information necessary to transfer to the Page and Connected States, the time interval of 341 msec would be adequate.

Operation performed to establish the communication channel	Time $_{(max)}$
Master Sends inquiry ID packets and listening to the responses through the inquiry channels (32x2x0.3125msec)	20 msec
Slave assumes a random non-responding time interval to prevent collision of responses sent simultaneously (1023x0.3125msec)	320 msec
Master sends another inquiry ID packet to invoke the slave to resume responding and gets back to the listening status	0.625 msec
Slave sends back its FHS packet	0.625 msec
Subtotal	341.25 msec

Table 8-2 – The typical minimum time for master and slave devices to transfer from the Inquiry State to the Page State.

In Table 8-2, the follows two points should be emphasized,

1. The subtotal does not reflect the minimum time. The major component of the subtotal time, required for the master and slave devices to transfer to the Page State, may be even shorter than that represented in the table.
2. If for any reason the contact between master and slave devices can not be made in the first channel sweep performed by the master, extra rounds of channel scanning adds another 20-msec of time interval to the subtotal time shown in Table 8-2.

8-1-4-3 Page State

As mentioned above, when a slave receives an inquiry ID packet, it responds with the FHS packet conveying its fundamental elements: BD_ADDR and clock. The master uses theose values to send a page packet conveying the slave's Device Access Code (DAC) code. This BB_PDU is called slave ID packet. To be able to listen to the slave ID packet sent by the master, the slave should be in the Page Scan sub-state as shown in Figures 8-2 and 8-3. The purpose of sending the paging packets by a master is to invite the potential slave, called the paged device, to move to the connected status with the master, called the paging device. Having received the slave ID packet sent by the master, the slave sends back the slave ID packet simply to notify the master that it has received the master's paging packet. To send back the slave ID packet, the slave should be in Page Response sub-state. Upon receiving the mentioned confirmatory slave ID packet from the slave, the master sends an FHS packet conveying its own fundamental elements and a number from 1 to 7 named the Active Member Address (AM_ADDR). The AM_ADDR is generated by the master and attributed to its slaves to distinguish them from each other. Any master can have up to 7 actively connected slaves. The mentioned master FHS packet is sent when it is in Master Response sub-state. The slave responds with one more slave ID packet and comes to the Connected State with the master, as shown in Figure 8-3.

8-1-4-4 Page Time Subtotal Calculation

In the Page and Page Scan states a master and a slave typically follow the same time schedule explained in sub-section 8-1-4-1 and -2 about the Inquiry and Inquiry Scan states, respectively.

Moreover, the same rules governing the time calculations of the inquiry sub-states are applicable in the case of page sub-states. Following such procedures, however, is typical but not mandatory and can be changed by control parts of Bluetooth applications [1]. Specifically, some changes in the steps of channel scanning procedure could make the behavior of the baseband layer of the Bluetooth protocol stack more suitable for the purposes of this project. Following desired changes in channel scanning procedure, Table 8-3 shows the typical minimum time spent, by the master and slave, in the Page State until they come to the Connected State.

Operation performed to establish the communication channel	**Time (max)**
Master sends slave ID packets and listens to the responses through 32 paging channels (32x2x0.3125msec)	20 msec
Salves sends a salve ID packet to acknowledge receiving the previous packet sent by the master	0.3125 msec
Master sends its FHS packet	0.625 msec
Salves sends a salve ID packet to acknowledge receiving the previous packet sent by the master and gets ready to transfer to the connected state	0.625 msec
Subtotal	~ 22 msec

Table 8-3 – The time spent by master and slave devices in the Page State before transferring to the Connected State.

In Table 8-3, it can be seen that since there is no collision of simultaneous messages generated by slaves there is no need for dealing with the non-responding status. This makes the Master's channel hopping the most time consuming component of the Page State. The master needs to broadcast the slave ID packet on at most 32 page channels, until it receives a response from the slave.

8-1-4-5 Connected State

When the communication channel is established and the Bluetooth devices are connected, in the way explained above, they still need to pass through two more steps before being able to exchange applications data at highest layers of protocol stack. The following time intervals should be added up to the ~360msec sub-total time, shown in tables 8-2 and 8–3 above, for establishing the communication channel through which application data could be exchanged:

- The time required for authenticity checking, and connection set up procedure, performed in the Link Manager Protocol (LMP) layer,
- The time required for L2CAP signaling and Connection Oriented (CO) channel establishment procedure performed in the Logical Link Control and Adaptation Protocol (L2CAP) layer.

In both cases the total number of -PDUs exchanged between transceivers could be adjusted to be only 10 packets occupying almost 6 msec more than the above mentioned sub-total.

The calculation shows that appropriate control applications might set the transceiver radios such that the communication channel establishment time between them could be as low as a fraction of a second.

While the gross data rate is 1 Mbps, the data rate of half-duplex packetized asynchronous connection less (ACL) data communication is 723.2 Kbps, at most, and 57.2 Kbps in the return direction. In the full-duplex case the data rate is 432.6 Kbps. Bluetooth supports 64 Kbps SCO (synchronous Connection Oriented) channel voice data rate [2].

8-1-5 Bluetooth Network Topology

As described in the previous sub-sections in establishing a communication channel between two Bluetooth devices one behaves as the master while the other takes the role of a slave. A master can have more than one slave. In Bluetooth the set of a master and its slaves is called "piconet".

The primary role of a master is defining the following specifications of its piconet:

- current frequency of communication,
- channel frequency hopping sequence,
- frequency hopping time,
- sequence of polling of slaves (determining which slave will be permitted to transmit),

The first three specifications determine how a piconet is formed or maintained. The fourth one shows how transmissions occur in a piconet.

A master device can have up to 7 active slaves and 256 parked slaves. Depending on the required amount of activity, connected slaves can also be in sniff and hold modes of operation. The details

of different operational modes of activity are not of main concern of this project and can be investigated further using Bluetooth references.

When two or more piconets partially overlap, a new Bluetooth network topology is formed and called a "Scatternet". In such cases a device belongs to more than one piconet. In the case of the scatternet a device may act as slave in more than one piconets or be a master in one while acting as a slave in others.

The specifications of the Bluetooth wireless technology considered so far gives adequate background to discuss its deployment in the present project as introduced in the following sub-sections.

8-2 Isolated Bluetooth Positioning Signposts

As explained in section 8-1-2, depending on the class of power used for transceivers radios, two Bluetooth devices may communicate with each other when they come to 10m, 20m or 100m of distance from each other complying with class 1, 2, or 3 regulations, respectively.

Considering route 60-Pembina as a specific example for this project, it can be seen that there are some regions at the downtown end of the bus path where the chance of obtaining correct positioning data from the GPS receiver installed onboard could become very low. In such a case two Bluetooth transceivers (one installed onboard and the other installed on a bus stop located in the area that is not covered by GPS signals, appropriately) can be used to acquire the positing data by the bus.

The timetable analysis of route 60 shows that an operative bus in this route has an average velocity of 15.1 Km/h (or 251.67 meter/minute). Bus stops located in crowded areas of downtowns that are not able to receive GPS signals are usually considered the major ones where buses have to stop for a while to let passengers leave out or get in the buses. However, to consider the worst case scenario it is assumed that a bus has the maximum permitted speed in the downtown (that could be as high as 60Km/h or 1000 meter/minute) when passing by a Bluetooth signpost. As shown in tables 8-2 and 8-3, as well as, based on sub-section 8-1-4-5, there is almost

400 msec according to the Bluetooth specification needed for two Bluetooth devices establish a communication channel and exchange regular positioning information. The above calculation shows that even in the worst case using class 2 transceivers that cover a 20 meter range, communication can be adequate to establish a system of isolated Bluetooth positioning signposts. In real conditions signposts are usually required to be located in main bus stops in which buses spend enough time in a 10 or 20-meter distance proximity of bus stops to establish Bluetooth communication channels with the devices installed in them.

Buses in the downtown area either need to stop at almost all bus stops to respond passengers' demands or they have to highly reduce their speed due to the traffic in the downtown. In addition, they typically spend at least a couple of seconds at the location of the main bus stops that are usually adjacent to main intersections. In such conditions even using class 1 transceivers would fulfill the communication channel requirements.

In addition to the proximity-wise communication capability of the Bluetooth technology, which makes it very suitable for isolated positioning applications as explained above, there are two other important specifications of this technology that make use of it even more interesting in establishing a networked positioning system. Such a system not only provides a positioning system with the precision satisfying the requirements of the present application but also removes the need of using GPS receiver and cellular phones in each bus. The trade off, however, is the need for equipping the buses with Bluetooth devices and establishing networked signposts along the road as described in the following sub-section.

8-3 Networked Bluetooth Positioning Signposts

The following three specifications of the Bluetooth technology are particularly important in this project;

- It is proximity-wise communication technology that gives it location-oriented capability used for establishing isolated positioning probes (as described in the previous sub-section).
- It is capable of establishing data network between more than two Bluetooth devices.

- It is capable of providing ad hoc data networks.

Figure 8-4 shows a part of an imaginary bus path covered by a network of adjacent Bluetooth devices installed in bus stops, and/or points in between, along the road. It is assumed that in this case the Bluetooth devices are of class 3 having the maximum permitted power providing 100m of communication range. For the cases that adjacent bus stops are more than 100 meters apart (in case of route 60-pembina the distance is usually around 150-200 meters) extra signposts should be added between bus stops. In addition, depending on the road physical geometry the signposts could be established in a zigzag format in both sides of the road.

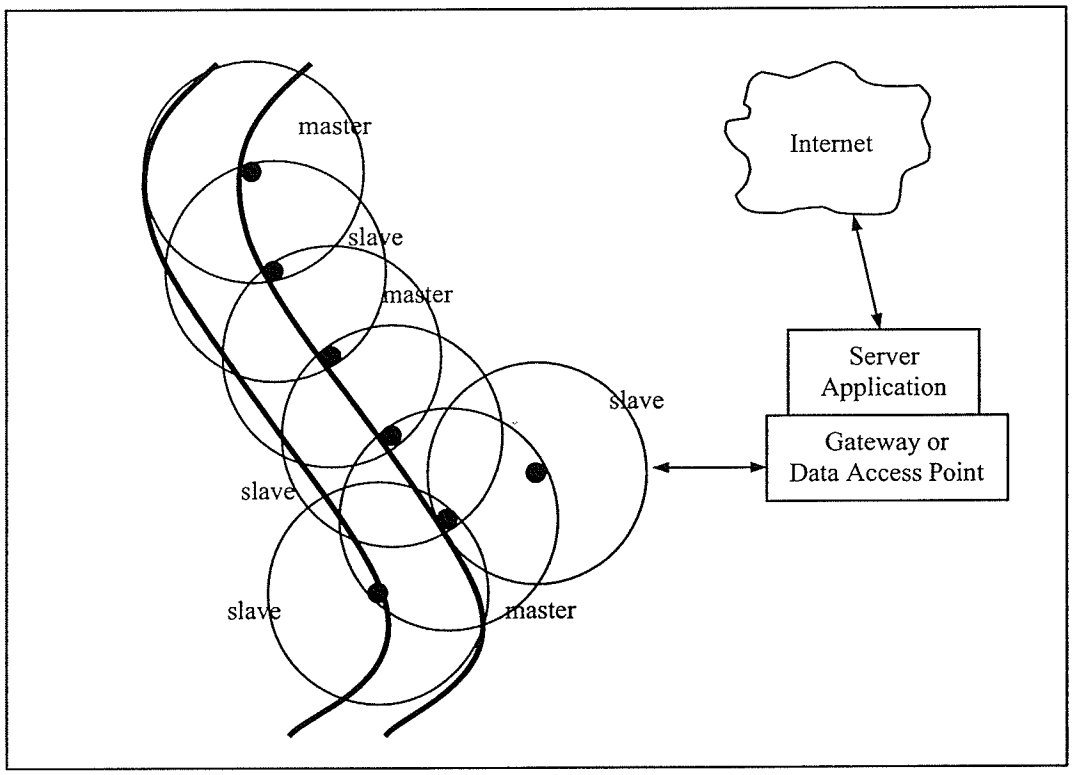


Figure 8-4 - Bluetooth signposts located in selected bus stops, and points in between, can cover the entire bus path.

Bluetooth devices located in adjacent signpost can be forced to take master or slave role in order to establish communication channels between each other and also be programmed to spread a scatternet along the road.

Figure 8-4 shows such a network configuration, as an example, in which adjacent signpost are assigned to be master and slave alternatively. Other kinds of network configurations can also be employed.

In Figure 8-4, there is also illustrated an extra node that works as the gateway or data access point that connects the network to the Internet. A server application in that node collects, and analyzes the bus route data gathered by each Bluetooth device and available to the scatternet as a whole and provides it to the transit system users through the Internet.

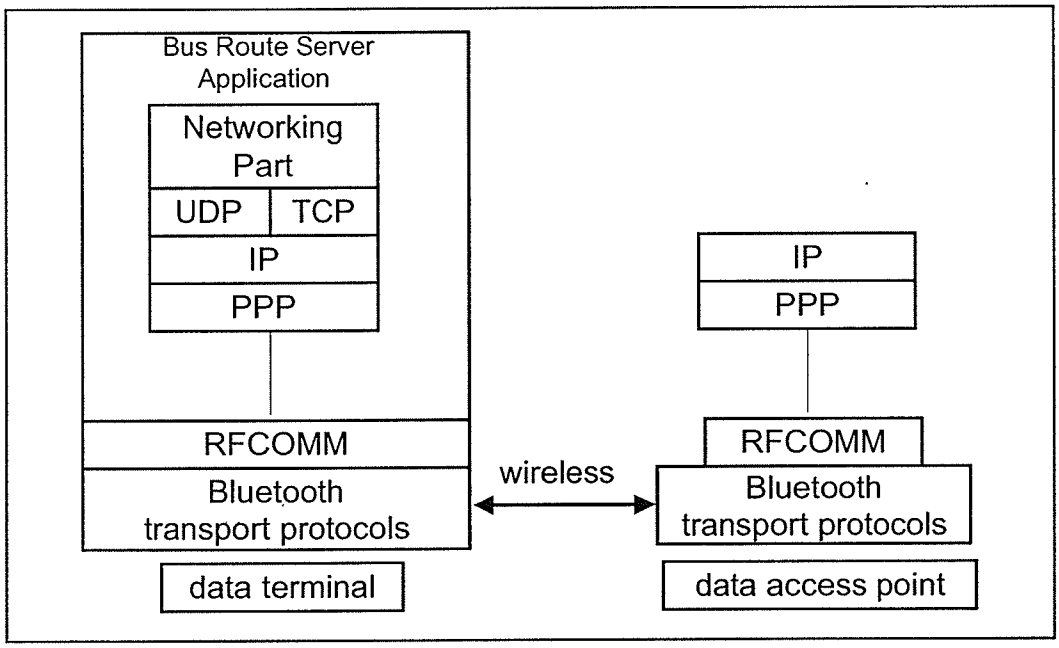


Figure 8-5- The protocol layers involved in developing applications capable of connecting the Bluetooth scatternet to the Internet.

Figure 8-5 indicates the network protocol layers involving in establishing such a system when a data access point (of say a LAN) is used to connect the server application to the Internet. A similar diagram can be used for illustrating the case when a dial up networking is used to connect the Bluetooth network to the Internet.

There are still many questions about establishing the scatternet that should be answered more elaborately. Some of the questions could be enumerated as follows;

- What are the scatternet restrictions in transferring data between different nodes along the road?
- How can suitable schemes for the protocols be defined to transfer the data?
- How would the number of nodes or the length of network affect the network performance?
- How long does it take for the nodes along the road to become updated about new events?
- What happens to the network if one of the devices along the road does not work appropriately?
- How is the information of a bus handled when it is in the area covered by more than one signpost?

In addition to the above questions that pertain to the network configuration and topology or the protocols involved, there might be many more questions about other subjects from the way that the signpost should be installed to the way that the server should be implemented.

Answering all the above questions is beyond the scope of this thesis and probably needs another thesis but would be required to be worked out.

The Bluetooth scatternet system, as described above, could have the following obvious advantageous in comparison to the GPS positioning system implemented in this project.

- The system replaces the need for GPS receiver and cellular wireless connectivity with the need for Bluetooth wireless connectivity. It is a cheaper approach in terms of the hardware operating costs, simply because the bandwidth does not need to be purchased, as the ISM bandwidth of 2.4 MHz is free to use.
- The scatternet system can be used by pedestrians in the proximity of the bus path, as well as by cars, in addition to its use for positioning information. Information about the transit system can be available for all the users, equipped with Bluetooth devices, who are in communication range of the network. On the other hand the information provided by the system could be more than just the bus route information. Advertising data, short data

messages, weather, traffic and commercial news can be delivered to users through this system, as well.

- Once the system is established it can be used for any vehicle equipped with appropriate hardware and software components.
- The system could work with WLAN, cellular phones, GPS and other installed infrastructures to communicate information of interests.

The system, however, may suffer from the following disadvantage:

- It needs extra investment for establishing the signpost network. That is, instead of using the investment already made in cellular phone and GPS systems infrastructure, new resources could be required to be spent in developing a new infrastructure. The price, however, is much less (or negligible) in comparison to that of a cellular or GPS system, simply because such a system is in fact a local network rather than wide area or global one.

As mentioned earlier, this section only discusses to the idea behind Bluetooth used in this project as was introduced in section 7. Using the Bluetooth scatternet is the key point associated with substituting the GPS receiver and cellular wireless connectivity with the Bluetooth network in this project.

The ideas that have been described emphasize how minor changes in the present system can result in developing new systems having more interesting capabilities. The main focus of this project, however, is in using GPS and cellular wireless systems. Therefore, in this project, it is assumed that there are isolated (not networked) Bluetooth signposts installed in approximately 100 meters from each other along the road including downtown locations where GPS receivers can not fix a point, appropriately, as shown in Figure 8-4.

To see the effect of the error of such a system when used instead of the GPS system on the total performance of this project, denote the errors of system when either GPS or Bluetooth is used for positioning operation as e_{20} and e_{100}, respectively. The numbers used as subscripts in e_{20} and e_{100}, represent the distance errors in the corresponding positioning systems.

Considering the fact that for the users of a transit system knowing about the time difference is more important and informative than knowing about the positioning difference, the distance errors mentioned above are better converted to time error to obtain more meaningful results.

The time errors of the GPS and Bluetooth systems can be represented as e_{20}/v and e_{100}/v, respectively, in which "v" represents the mean speed of the buses in their route. Since time resolution of the bus schedule and the monitoring system is 1 minute, it is concluded that timing errors are considered significant when either of e_{20}/v and e_{100}/v values are equal to 1 minute. Therefore, in GPS system with 20-meter distance resolution an average speed of as low as 20 meter/minute (1.2 Km/h) can be detected in every minute positioning re-evaluation procedure. In case of the Bluetooth positioning system with 100-meter distance resolution, as described above, the minimum detectable speed is 100 meter/minute (6 Km/h - which is equal to the average speed of a fast moving pedestrian).

The above discussion shows that for the transit system considered in this project with 1-minute time resolution, the resolution of the GPS positioning system is unnecessary. The average speed of 1.2 or even 6 Km/h occurs in the case of very dense traffic jams and/or closed road conditions. Delay detection regarding these cases can be easily made by other means (like radio broadcasting) and sending additional data related to the event may not be informative.

In normal cases the operative buses of a transit route either drive in speeds close to the maximum permitted value of the road or are stopped in bus stops to respond to the passengers' demands. There are also time intervals between these cases that the bus driver needs to spend to bring a bus to either extreme. In the case of the example taken for this project, route 60 – Pembina with the average scheduler speed of 15.1 Km/h (251.67 meters/minute), the distance resolution of the positioning system could even be as high as 250 meters.

To compare two systems from a different point of view, let's assume that an operative bus is traveling on the route with maximum permitted speed that is 60 Km/h (1000 meters/minute). A GPS positioning system can determine its arrival time to a destination with the time error of

20/1000 minute which is 1.2 sec. However, the Bluetooth positioning system anticipates the arriving time with 6-sec error. As the average speed decreases, the timing error increases. In the case of the example used in this project, route 60-Pembina, with the average speed is 251.67 meter/minute, the timing error provided by GPS and the Bluetooth systems are 4.8 sec and 24.8 sec, respectively.

As a matter of fact, in the case that a Bluetooth positioning system with less timing error is desired Bluetooth transceivers may be used dissipating less energy, at signposts installed closer to each other.

It can be seen that in either case the error value is less than the resolution of the reporting arrival time that is 1 minute. Therefore, in the normal cases the difference between the timing errors of the two systems is not significant. In fact, for the case of this project, it is even better to have a Bluetooth system with around 200 meters of distance error (distance between signposts) to save hardware resources. No class of power of Bluetooth technology, currently supports such a distance for establishing networked signposts.

8-4 References

1. B. A. Miller, C. Bisdikian, "*Bluetooth Revealed*", Copyright 2001 Prentice Hall PTR.
2. D. Sims, T. De Silva, K.Bebenek, "*Review of ITS Architecture within the Canadian Context*", IBI Group, May 1999, pp 74-76.

Section 9-Transit System Information Access over the Internet

9 Information Access over the Internet

In Section 7, it was shown how the real-time positioning information of an operative bus is gathered and becomes available for a central database unit. This section shows how the central database engine receives the real-time data, what operations it performs on real-time and scheduler data stored in it, and how it makes the required data available for users of the transit system over the Internet.

As a database application over the Internet, this part of the thesis present a prototype client/server model in the Java programming language, and database connectivity to provide the user with the requested information. In the following subsections the details of the role of each component used in this part of the system are investigated and the implementation of them are explained.

9-1 Client/Server Model

Clients and servers can be defined as separate entities that work together over a network to accomplish a task. Particularly, they have the following characteristics:

- As two processes usually running on separate machines their relationship is determined by the role they take to provide or receive (use or consume) services. As the names imply the services are provided by the servers and consumed by the clients.
- A server also acts as a regulator of the access of more than one client to the same service (resource) at the same time. Therefore, there is a many-to-one relationship between clients and servers.
- Clients always initiate requests while servers are passively awaiting them. The role of two processes may change in some cases (e.g. using callback objects of servers by the clients in which the servers call back the clients receiving a service). A program can be client, server or both.
- The location of a server is usually transparent to the clients. It may reside anywhere on the network or even on the same machine where the client is.

- Ideal client/server software is independent of the platform or operating system that either process may be running on.
- Clients and servers are loosely coupled systems interacting through a message-passing mechanism conveying service requests and replies.
- Client/Server systems are scalable, adding or removing clients with limited of impact in performance (horizontal scalability), and migrating to faster, larger, or distributed servers (vertical scalability).
- The servers are centrally managed which helps to reduce maintenance and better guard shared data integrity. The clients, however, remain personal and independent.

9-1-1 Client/Server Multi-tier Architecture

A client/server-modeled application may be split into smaller architectural units each of which functioning as client and one or more server parts. As such units are tied together, they constitute multi-tire client/server applications. In a 2-tier architecture the client side usually provides the Graphical User Interface (GUI) and the server side provides the logic (main process) of the application. For example in the case of a database client/server application, a user may make its inquires using the GUI provided by the client side, send the inquiry to the server, and wait to receive the result. Upon receiving the inquiry, the database engine, residing in the sever side, processes the inquiry and provides the results from the stored data in the corresponding data tables. There also might be cases of 2-tier applications in which the main process could reside in the application's client as in the case of fat client applications.

On the other hand, in a 3-tier client/server architecture – implemented in this prototype - instead of being directly coupled to the server, the client is coupled to a middle tier where the main process of the application resides. The middle tier is, in turn, coupled to the third tier of the application. Therefore, in the 3-tier case there are two servers connected together. Generally speaking, in N-tier (N>2) client/server application architecture, there are N-1 servers connected together. There is no limit in the scalability of this architectural model.

9-1-1-1 3-tier Versus 2-tier Applications

3-tier applications are easier to use and deploy on the network in comparison with the 2-tier applications. In addition, in the case of small footprint technologies like Java Applets and Beans most of the code runs on the servers. Table 9-1 shows the 3-tier versus 2-tier applications from a broader perspective and compares their more critical issues.

	3-tier	2-tier
System Administration	Less Complex (manageable through server application)	Complex (more logic in the client to manage)
Encapsulation of Data	High (the client invokes services or methods)	Low (data are exposed to the client)
Application Reuse	Excellent (services and objects can be reused)	Poor (client is monolithic)
Performance	Good (only service request and results are exchanged between client and server)	Poor (selected data must be downloaded before client can process them)
Security	High (object, service, or method level)	Low (data type level)
Scalability	Excellent (can distribute loads over multiple severs)	Poor (minimum management on client side)
Ease of Development	Getting Better (more tools are becoming available)	High
Availability	Excellent (middle tier components can be moved to other servers)	Poor
Hardware Architecture Flexibility	Excellent (the middle-tier servers can be distributed across several machines)	Limited (server resides on one machine)
Communication Choices	Rich (supports RPC-like, connectionless messaging, queued delivery, and broadcast)	Poor (only connection-oriented RPC-like calls)
Legacy Application Integration	Yes (via gateways encapsulated by services or objects)	No
Internet Support	Excellent (because of thin clients can be implemented and/or downloaded)	Poor (Bandwidth limitations exacerbates the poor characteristics of fat clients)
Heterogeneous Database Support	Yes (multiple databases can be used within the same business transaction)	No

Table 9-1 – Comparing 3-tier versus 2-tier

It is recommended that 3-tier architecture be used instead of the 2-tier one if the application has any of the following characteristics:

- More than 50 services should be provided by the sever application,

- The application is prepared by different sources using different computer programming languages.
- Heterogeneous data sources are involved.
- The application will live more than 3 years.
- The data is being accessed by more than 50000 transactions a day, or 300 users simultaneously.
- There is a significant inter-application communications.
- There is a possibility that the application will grow over time so that one of the above mentioned conditions will apply.

As will be explained in the following sub-sections, in this application a 3-tier client/server application has been implemented.

9-1-2 Client/Server Building Blocks

Figure 9-1 shows the general client/server block diagram and the particular structure of the software used in this application.

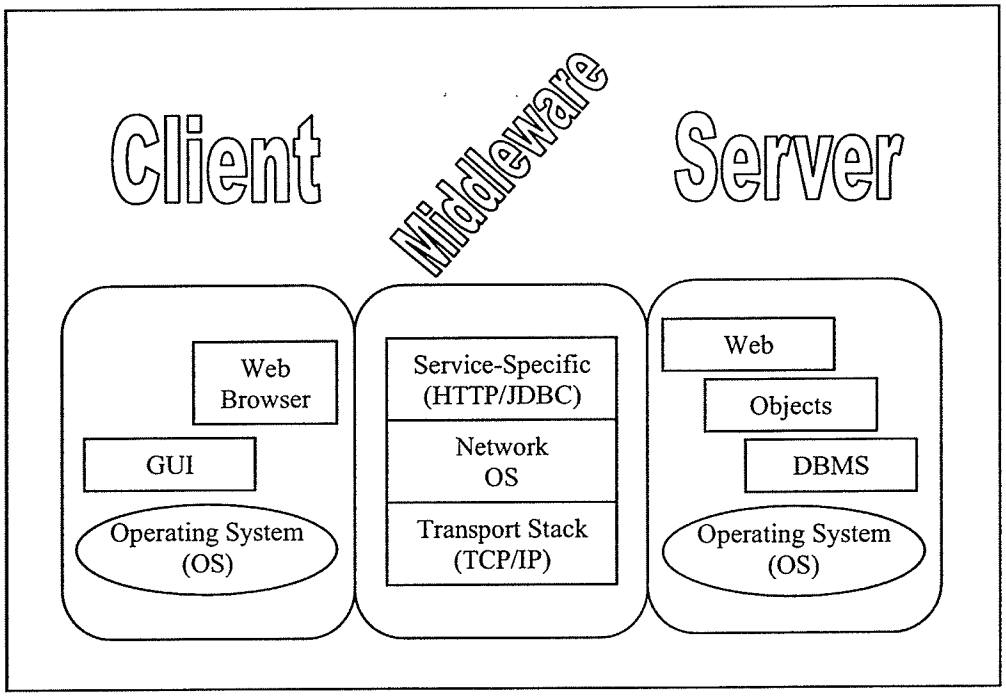


Figure 9-1 – The client/server structure of the application developed in this project.

In the following subsections the role of the components presented in each block of Figure 9-1are investigated in general, and the implementations of them as performed in this application are discussed in detail.

9-2 Middleware

The middleware part of the client/server architecture (the slash in the model!) is the glue that sticks the client and the server ends of the application together and lets the client obtain the requested service from the server. It runs on both client and server sides of the application, and is responsible for transmission of a service request, that is initiated from the client to the server over the network, and returning back the results from the server to the client. In 3-tier or N-tier applications middleware is responsible for making the middle-tiers bond together. Therefore, it is used to coordinate inter-server interactions, too.

In terms of the functions performed by the middleware, it may be attributed to have two functional blocks of pipes and platforms. Pipes provide the communication services between the various components of the client/server application. Platforms are server applications that provide the running environment for server side components.

The pipes may further be divided into the two following broad classes:

- General pipes, such as TCP/IP used in this project, which provide the substrate for most client/server interactions.
- Service-Specific pipes, such as JDBC and HTTP used in this project, which accomplish a particular type of service. JDBC is a database-specific middleware pipe, while HTTP is categorized as an internet-specific one.

Figure 9-1 shows how the middleware block of this application may be broken further to the layers based upon the categorization explained above.

TCP/IP is the telecommunication protocol stack that is used for message transmission between clients and servers. Since TCP/IP are classified as the lower-layer protocols in the network telecommunication modeling and are not directly used in developing of the application. The only

related point with regard to the TCP/IP protocol stack, used in this application, which should be mentioned here is that in part of this implementation Java sockets have been used. Sockets are one of the major peer-to-peer protocols. A socket address on the Internet TCP/IP consists of an IP address and a port number. Examples of these kinds of addressing are shown in the following sub-sections.

In Figure 9-1, Network Operating System (NOS) is the part of the middleware that is used to create the single-system-illusion. It provides the network users with location, Namespace, local/remote, and distributed time transparencies. Once again it is provided by the network administrative system and its specification is not directly related to this implementation and will not discussed further, either.

JDBC and HTTP middleware pipes are briefly explained as follows:

9-2-1 JDBC

Based on X/Open SQL Call Level Interface (CLI), Java Database Connectivity (JDBC) defines how client/server interactions are implemented for database systems. It is entirely written in Java and lets a programmer write a Database Management System (DBMS)- independent Java codes in the form of applets and applications and access any data source without the need to change the code. It is also an acronym to refer to an API for using low-level JDBC drivers or creating them to do the actual connecting and transacting operations with the database engines [1-3].

JDBC is a Java package and a part of Java Developer's Kit (JDK) distribution. Therefore when used in an applet, as in the case of this implementation, the driver is downloaded along with the applet's other classes (packages). Figure 9-2 shows the position of the JDBC layer within other layers of the database connectivity structure when more than one Database Management System (DBMS) is used. In this application, however, there is only one DBMS called mSQL. More about the databases will be provided in a later sub-section where the components of the server side of the client/server architecture are discussed.

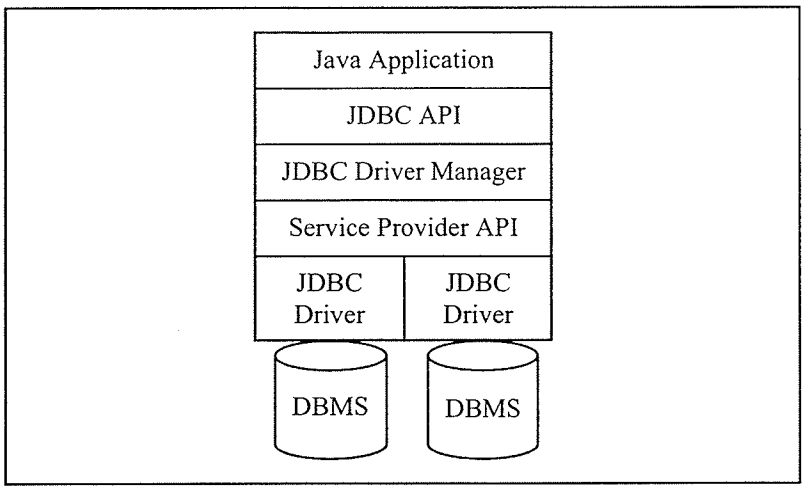


Figure 9-2 – Java Database connectivity structure using JDBC.

JDBC uses the following naming scheme to find and communicate with its database:

jdbc:<subprotocol>><domain name>

In this application the naming scheme is as follows:

Jdbc:msql://www.ee.umanitoba.ca/mydatabase.

The "mydatabase" part of the name scheme refers to different databases, used in this application, and is replaced with the corresponding names, when needed, as will be discussed further in the client subsection.

In this implementation the following JDBC driver has been used: Java Database Connectivity (JDBC) – msql-jdbc 2.0b5 [4].

9-2-2 HTTP

As show in Figure 9-1, HTTP (Hypertext Transfer Protocol), known as an Internet middleware pipe, is a service specific protocol that is used by the Web browser applications. A web browser is a minimalist universal client for a web server that interprets the information received from the server and displays it graphically to a user. It executes server's commands, received in the HTML (Hyper Text Markup Language) tag formats, to display text and images on a specific GUI platform.

Using the HTTP protocol Web browsers send HTTP servers their requests for accessing a web resource. The server replies with a platform-independent content that can be used by any browser. In fact, HTTP is the Web's Remote Procedure Call (RPC) protocol created on top of TCP/IP. It borrows many of its specifications from the Internet Mail, and Multipurpose Internet Mail Extension (MIME) protocols, which provide extensible mechanisms for transmitting multimedia email.

HTTP is used to call and retrieve Unified Resource Locator (URL)-named resources over the Web. A typical URL consists of the four following parts: 1) the protocol scheme, 2) the server name, 3) the port number, 4) the path to a target resource. Figure 9-3 shows a typical URL or Web address having all parts mentioned above.

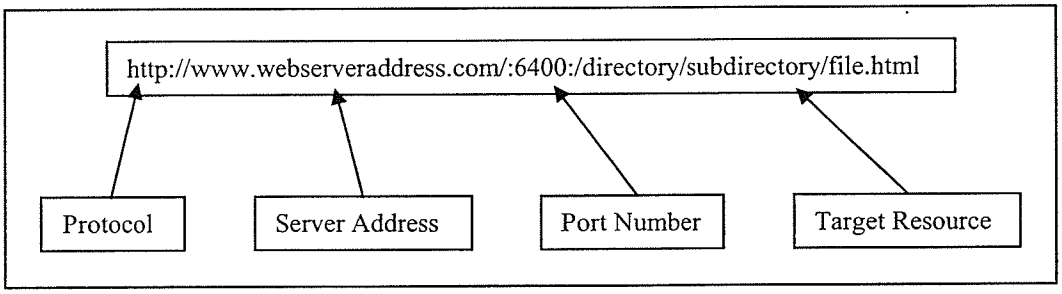


Figure 9-3 – The URL Structure

9-3 Client

Client/server applications are client-centric i.e. it is the client side that provides the look-and-feel specification of the application. In the case of this application, the thin client requires a Web browser to download Java applets to create the required GUI. Therefore, the browser should be Java enabled and the machine on which the client is running should provide the Java Virtual Machine (VM).

Generally speaking, the steps taken by a Web browser to request a Web page, with an embedded Java applet, from a Web server can be listed as follows. The following list also indicates the operations that the applet, by itself, performs to run in the environment provided by the Web browser (Figure 9-4 may be helpful in following the general steps described below. It also illustrates the 3 tier client/server architecture used in this application.):

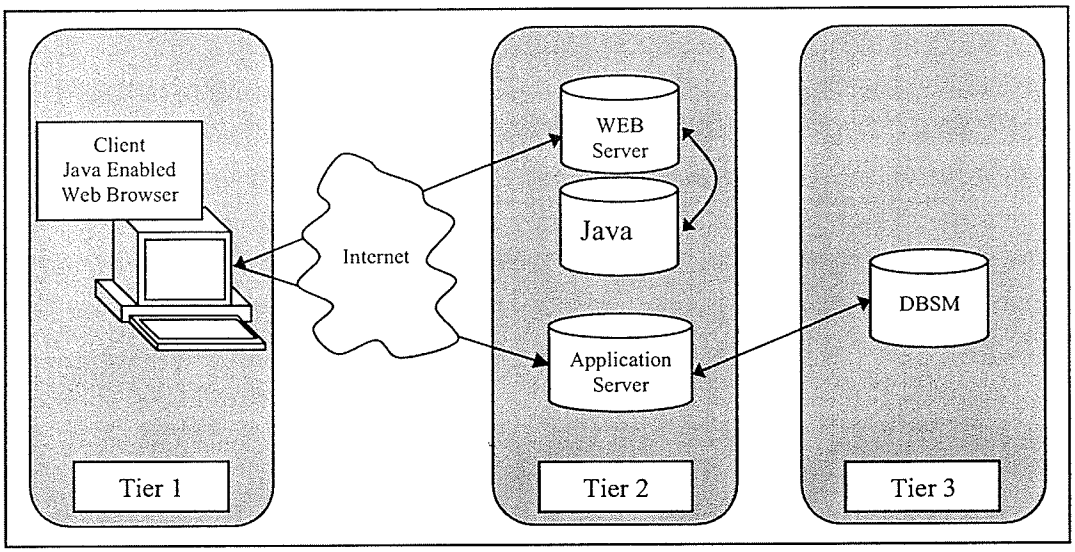


Figure 9-4 – The client interaction with other components of the client/server architecture as implemented in this application as an example.

1. The Web browser starts the Web client/server interaction with sending a URL, embedded inside an HTTP request, to the target server.
2. The HTTP server that is listening to its typical HTTP port (no. 80), or the one determined explicitly in the URL, initiates establishing a socket connection with the client. The server finds the requested HTML file, sends back a copy of it to the client, and typically closes the connection.
3. The Web browser receives the HTML commands, present in the HTML file sent by the server, interprets them and displays the page contents in its window. If it encounters an applet <OBJECT> tag in the HTML file it requests the corresponding Java applet, which its class filename is presented as the tag attribute. For the security purposes Java applets should usually reside on the same server machine where their HTML page originates.
4. The browser downloads each applet that it encounters with its name within the HTML page.
5. The browser loads the applet into the client's memory and then executes it. Typically, the applet displays its contents within the area of the browser window assigned to it by the

browser. The attributes of the <OBJECT> tag let the browser know about the area that should be assigned to the applet.

6. The Applet paints the contents of its region, applies the background (and foreground) colors, determines font type and size, and handles the keyboard and mouse events.
7. When the applet exits the web page, the browser deletes it from the memory of the client machine.

9-3-1 Java Applet's Portability, Safety, and Life Cycle

A Java applet is a mobile code (object). Like traditional software, it consists of a sequence of executable instructions. However, unlike traditional software, it is distributed across the network, and can be dynamically loaded and executed by standalone programs such as Web browsers. Therefore, an applet should be portably safe. The host system provides a run-time environment for loading, executing and unloading the applets. Also, the browser precisely controls the applet's access to memory, system calls and server function calls.

Java provides the safety and portability by compiling applets for the Java Virtual Machine that is modeled after a Reduced Instruction Set Computer (RISC) processor's instructions. The result of an applet compilation is a set of primitive instructions called bytecodes. Bytecodes are the lowest possible compiled instructions that are still machine-independent. Bytecodes make Java a partially compiled language, because creating of bytecodes is about 80% of the compilation of the Java code. Java run time interpreter is responsible for completing the remaining 20% of the interpreting of the bytecode to the host machine code. Therefore, Java applets can be thought 80% compiled and 20% interpreted. The following steps should be taken to create and execute a Java applet (Figure 9-5).

- The Java applet code goes through a Java compiler to create the bytecodes that are stored on a server.
- The bytecodes are copied to a target client machine when a browser requests the applet.

- Upon reaching the target machine, the bytecodes are run through a Java verifier that tests the codes for forged pointers, access violation, stack overflow, and parameter type matches; it ensures that the received code is safe.
- The bytecodes are handed over to the Java class loader after being verified that are safe.
- The class loader typically hands the bytecodes to an interpreter which is the run-time element executing the Java instructions on the target machine.

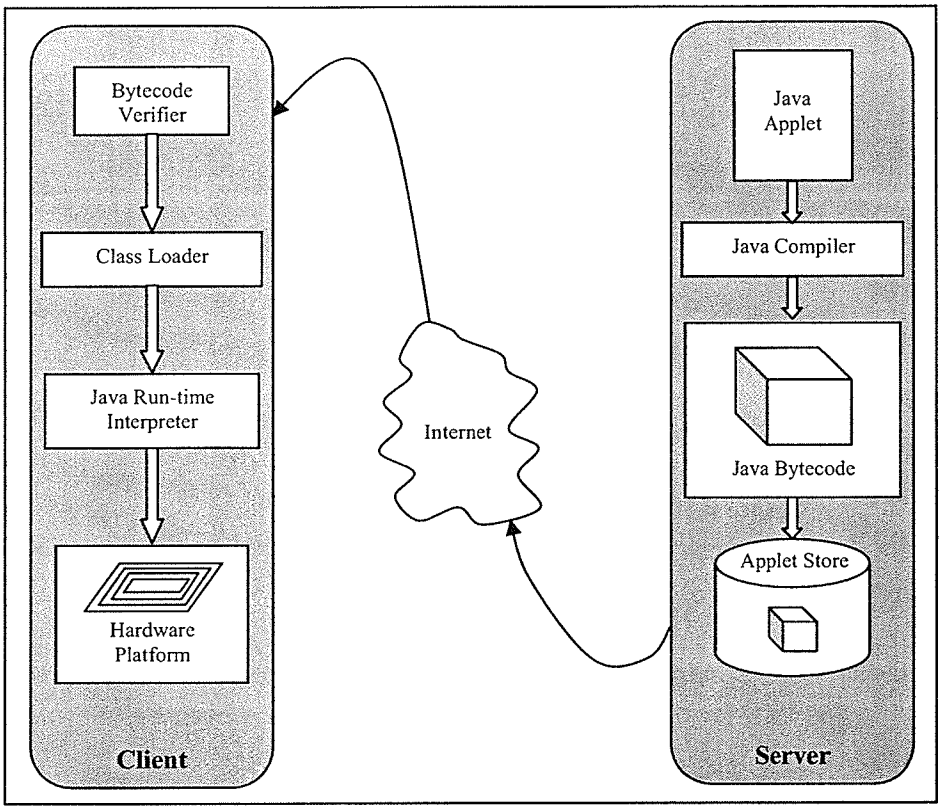


Figure 9-5 – The steps of creating and executing of a Java applet.

Most browsers impose the following restricting on untrusted applets [5], i.e. they can't:

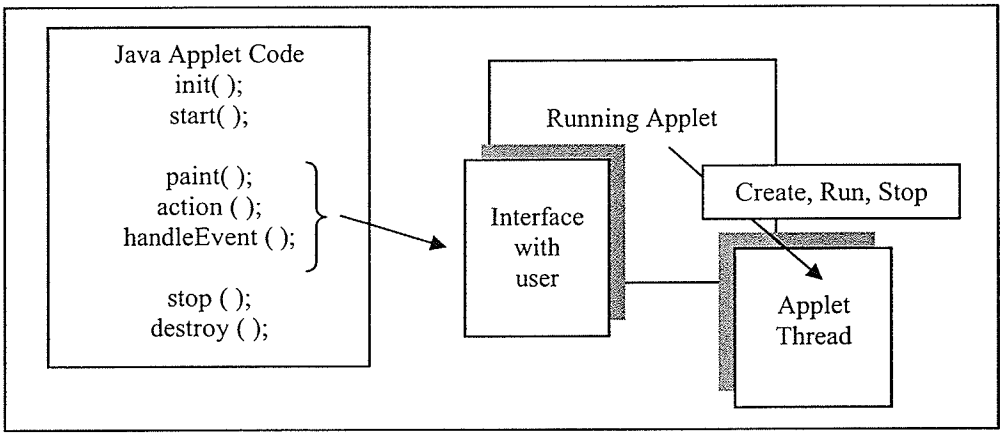
- Read or write files on the local host,
- Start other processes on the local host,
- Open network connections (sockets) to servers other than where they originated,
- Have native methods.

As a framework for running a Java applet, a browser provides three services for it. First, the browser controls the applet's life cycle. Second, it provides the applet with attribute information

from the <OBJECT> tag, including the browser's window real estate that the applet needs to use. Third, it serves as the main program or process within which the applet executes.

An applet is a Java class that inherits its behavior from the Java Applet class and extends its capabilities by providing more new functions. Therefore, an applet has the same life cycle, introduced in the Java Applet class, that its phase is fully controlled by Web browsers.

When a browser downloads and runs an applet in its environment, it is also informed about all events such as key or mouse strokes that happen during its lifecycle. Java applets should implement codes of the methods defined in the Java Applet Class and the Java Runnable interface in order to be invoked by a browser call.



Fugure 9-6 – An applet lifcycle.

Figure 9-6 shows the methods that a browser invokes to control an applet's lifecycle. When an applet is loaded for the first time by the borwser and becomes ready to run, its init( ) method is invoked once. As its name implies, init( ) is the method in which all initialization and/or one time set up tasks should take place such as loading resouces, building the user interface, and parsing parameters. The bowser calls the start ( ) method whenever user enters or retuns back to the Web page contining the applet i.e. whenever that it becomes visible. The stop ( ) method is called whenever the user moves off the page and/or the applet becomes invisible. Finally, the browser calls the destroy method ( ) before it shuts down. The method causes the applet to stop executing, kill all its threads, and release all resources.

9-3-2 The Client Side of this Implementation

Given the information about an applets' specifications in general, the important execution steps of the applets developed for the client side of this application is investigated. The corresponding codes can be found in the form of electronic attachments or the appendices of thesis. Figure 9-7 shows the client web page in which the main applet of this application is running. Figure 9-8 shows the zoomed-in area of Figure 9-7 enclosed in the gray-border square. The specifications and the lifecycle steps of the applet can be described as follows:

1. The applet shown in Figures 9-7 and 9-8 implements Java Interfaces MouseMotionListeners, MouseListener, and Runable.
2. In its init( ) method all the variables used in various methods of the applet are loaded with their corresponding initial values and the applet acquires the ability to receive and respond to mouse motion and click events.
3. The applet then creates and starts a thread that is responsible for providing the User Interface (UI).
4. The thread creates the UI displaying the cartoon map distributed by the transit system for the users. The map gives a familiar appearance to the UI and can be easily used by the user [Section 5]. The thread also displays some selected bus stops along the road. It also shows the location of the buses on the road based on the transit system timetable (to show where they are supposed to be) and their real-time locations (if there is a discrepancy between where they are and where they should be).
5. To find out about the points representing the bus locations on the map, the UI thread contacts two application servers at the following address: www.ee.umanitoba.ca, at ports: 6400 and 6100. When the connection between the UI thread and the servers are established the thread makes its queries to tables "tiTa60c" and "gps60" of the database server. In case of table "tiTa60c", which has the scheduled information of the bus route, the queries are made based on the server's local time.

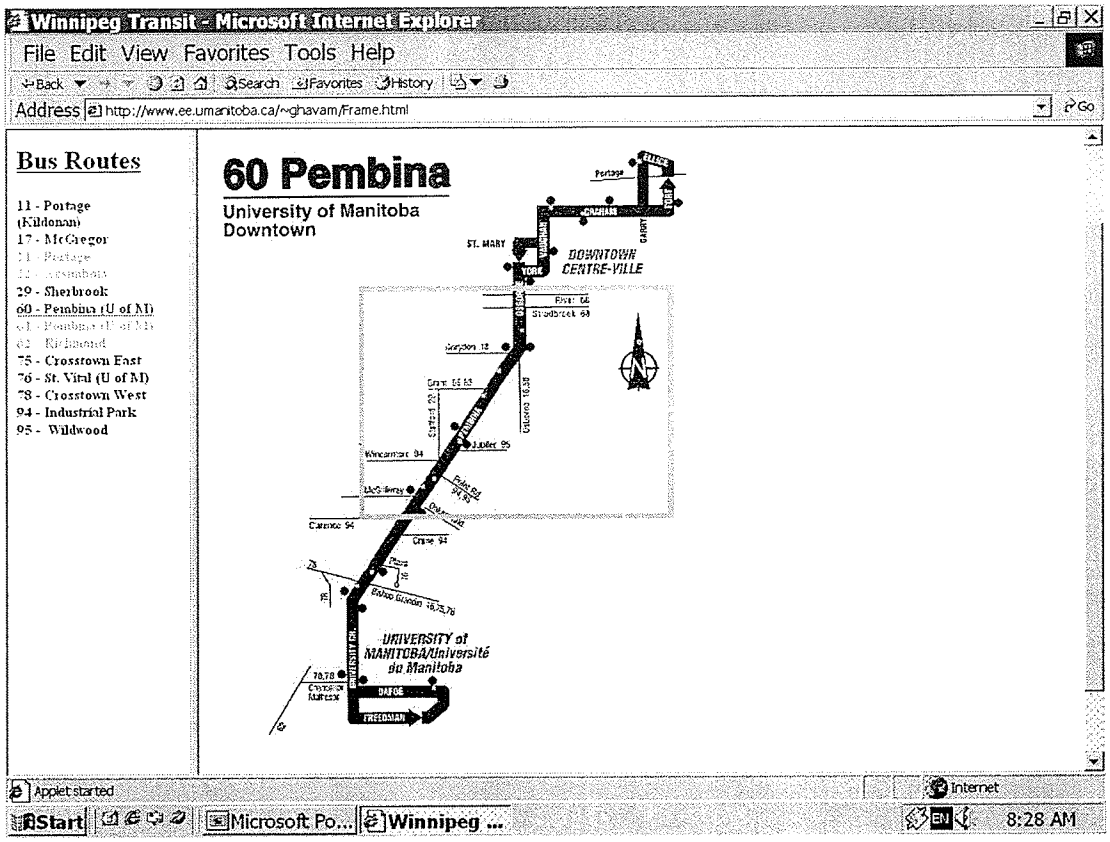


Figure 9-7 – The web page running UI Java applet, developed for displaying the real-time locations of the operative buses on route 60 – Pembina, on a cartoon map, distributed by the Winnipeg Transit System.

6. The servers receive the UI thread's queries and provide them with the information regarding scheduled and real-time location of the buses. If the connection with either server can not be established the following error message appears at the browser's Java consul: "Couldn't get I/O for connection to www.ee..." indicating that the server is not responding to the applet's requests for services. The activity of currently running Java virtual machine is terminated, and the applet execution is ceased.
7. The scheduled locations of the buses are shown by red-orange colored squars. The real-time locations of the out-of-schedule buses are shown by green color square.
8. The UI thread establishes socket connection to the servers, sends the queries, receives the responses, closes the connections, and represents the results on the map.

9. The UI thread makes the socket connections and queries in time intervals that may be informative for the user. In this application the applet asks updating inquiries every minute. The screen displays of the information and mouse event handling are performed in a continuous manner.

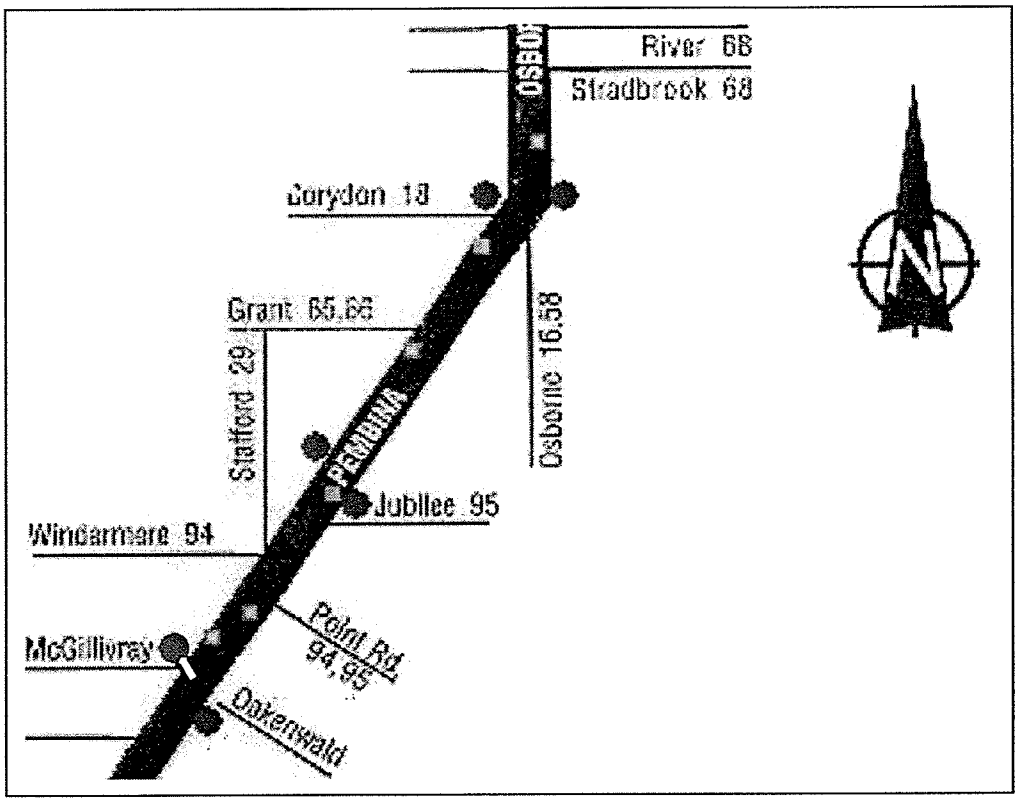


Figure 9-8 -The UI applet from a closer look, indicating the scheduler and real-time locations of the operative buses.

10. Since, there is no updating regarding the positions of the operative buses between two subsequent queries, the map looks like still. However, changes of the colors of the bus stop representative points resulting from the mouse movements over them and/or calling new applets showing the operative buses' arrival times to selected bus stops provides the interactive nature of the UI applet.

11. The map is sensitive to the motion of the mouse pointer at the location of the selected bus stops. When the pointer is not on a point representing a bus stop on the map, the point is shown in blue color. When the pointer is on a point representing a bus stop it is displayed in red color. The details of the UI can be seen in Figure 9-8.

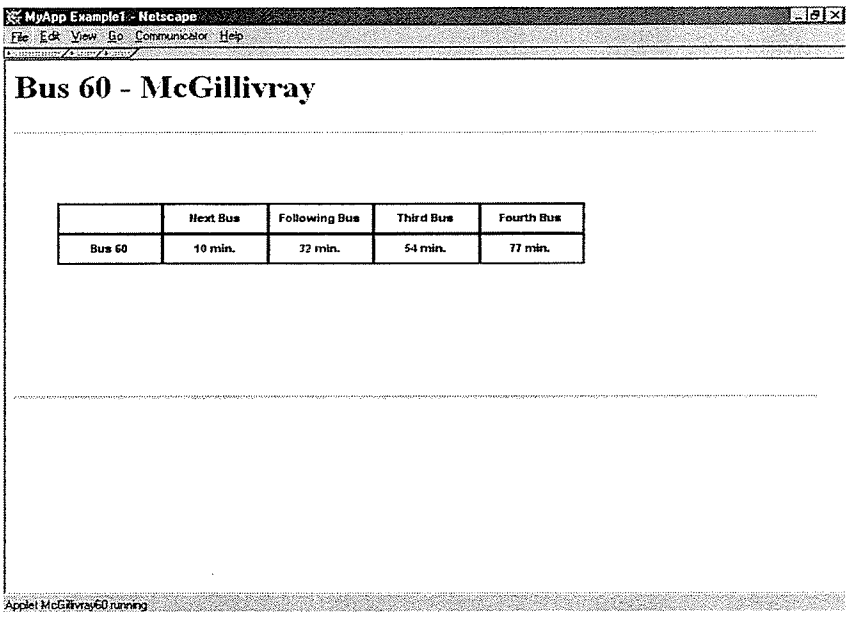


Figure 9-9 – The table showing the arrival time of the four next buses to a selected bus stop.

12. When the mouse is clicked over a selected bus stop, a new applet is called and displayed on a blank (new) window of the web browser. The applet displays the scheduled time of arrival of up to four next buses to the bus stop. Figure 9-9 shows a table displaying such information.
13. The applet also searches for any out-of-schedule buses arriving at the bus stop within the next 2 mins. If it finds any the following statement is also displayed below the table shown in Figure 9-7: "Out-of-schedule bus arrives in 2 min.". Predicting that an out-of-schedule bus arrival at a bus stop with a certain delay is another source of ambiguity that could be addressed by techniques such as Kalman filters.

    The information contained in this applet can be used in displaying bus arrival times to desired bus stops using various methods.

Figure 9-10 shows how this part of the project implements the general client/server architecture presented in Figure 9-4. The following sub-sections describe the remaining components of the client/server architecture as deployed in this part of the project.

9-4 Application Servers

The server typically runs on top of some shrink-wrapped server software package. In the case of this application the latter package is the mSQL database engine. The server depends on the OS, on which it resides, to interface with the middleware block to receive the requests for services.

9-4-1 The Application Servers of this Implementation

A server program, as its name implies, serves multiple clients with the resources that are of interest. The application servers developed for this project have the following working specifications:

- The server programs are started before clients can make their requests and keep running all the time.
- They spend most of their time passively waiting for client-initiated requests. Upon receiving client requests, they assign dedicated sessions to every client.
- While protecting the shared resources, the servers execute the requests of many clients at the same time. Since it is more efficient that tasks are allocated to parts of the same program rather than to separate programs, the servers are designed as multi-treaded (the tasks are called threads). These threads are faster to create and have easier access to the shared information. The multi-threaded server processes several clients programs, concurrently, and since they run over multitasking OS, they do not let one client access all of the system's resources and/or prevent the other clients have their requests proceed. Concurrent threads can use CPU's cycles productively to perform useful work. It is the operating system's responsibility to provide basic services for concurrent purposes, such as task preemptive, priority, synchronization mechanisms, inter-process communication, inter-task protection, efficient memory management, and so on.

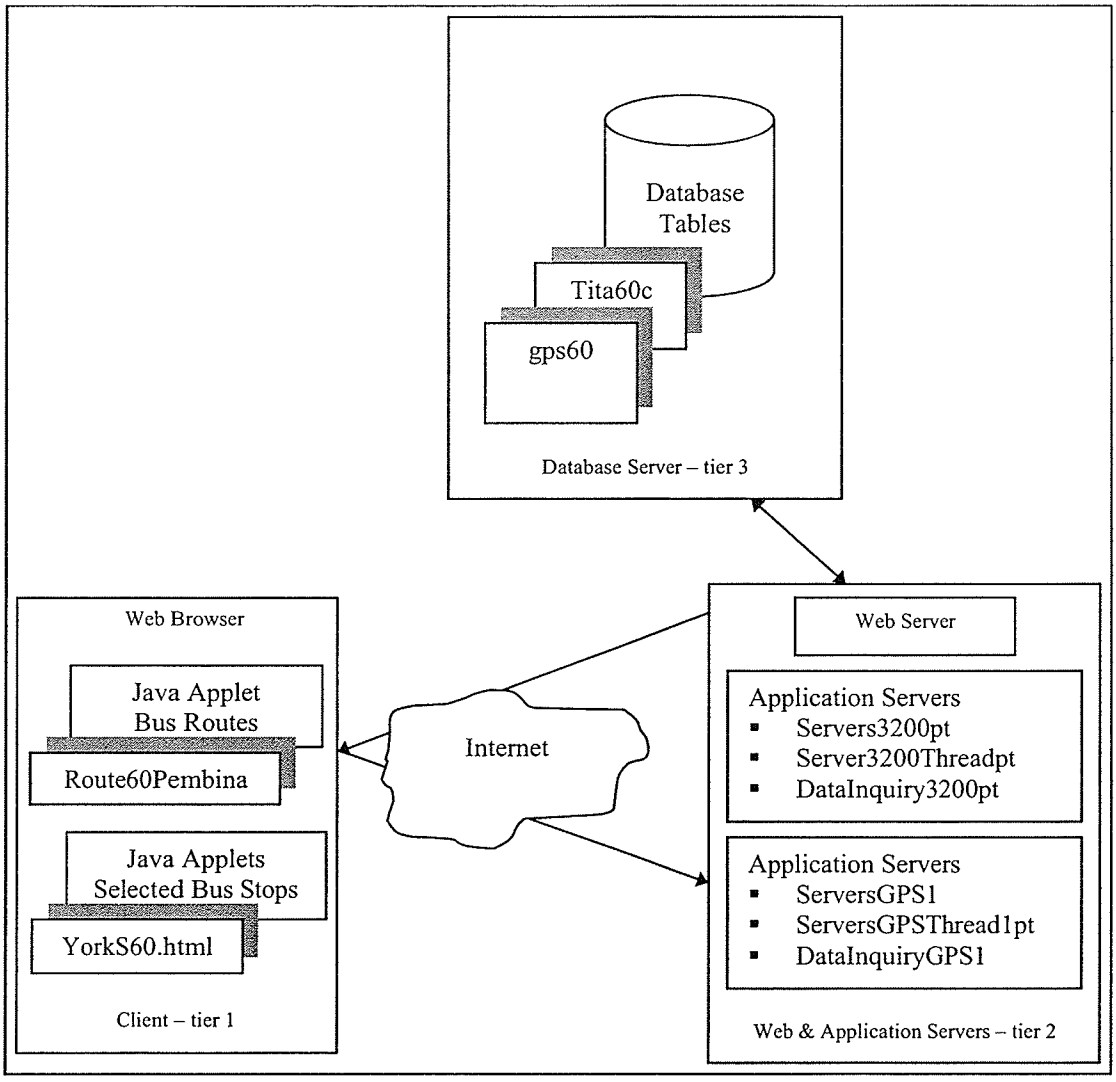


Figure 9-10 – The client/server implementation of this project.

- Due to the fact that the servers have been developed using Java's (object-oriented) capabilities, in essence, they are modular and can easily be scalable when new and more requirements need to be fulfilled in future.

9-5 Database Server

The database server, in this project, is a server application that receives Structured Query Language (SQL) command messages from the clients over the network, processes the request using the processing power of its own machine, and executes them over the data residing on the

same machine. The result is then sent back to the client. With this approach the Database Management Systems (DBMS) server codes are shrink-wrapped and provided by the vendor. The user, however, should create the SQL tables and load them with the corresponding data. The database application code residing on the client machine should be written by the user or can be bought as a shrink-wrapped query tool from other vendors.

9-5-1 Sequential Query Language (SQL)

SQL (Structured Query Language pronounced "sequel") is a universal query language to access relational (and non-relational, used in the case of IBM data warehouse to access IMS and Indexed files [1]) database systems. It consists a short list of commands that are English-like, powerful, flexible, and firmly rooted in mathematical foundation of set theory. The commands let information collected in relational database tables are manipulated in various desired ways. As an interactive ad hoc query language for database programming, SQL defines, administrates, and protects data in multi user networked database server environments. It can be embedded in many computer languages including Java or it can be called using SQL/ Call Level Interface (CLI) callable interfaces.

9-5-2 What a Database Server Does

In database-centric client/server architecture a client requests data services from the server that is also known as database engine, or SQL engine. The server responds to the client by providing it secured access to the shared data. It manages the control and execution of the SQL commands received from the client. In addition, it provides the required system administration features and utilities for managing the data in the database tables.

9-5-3 mSQL Database Engine

Mini SQL or mSQL is a lightweight relational DBMS, capable of handling simple tasks, and developed by Hughes Technologies [6]. It uses a subset of SQL commands and is designed to provide rapid access to data tables with little system overhead. It needs only small hardware (PC class hardware) resources to provide its full functionality for the user. Though it is available over

the Internet, it is not Freeware. However, its use as a free resource is permitted for developing non-commercial applications like the present one. Because of its specifications version "2.0.11" of it has been used as the database engine of this application [Figure 9-10]. This version has been designed for rapid access to large data sets in the million record sizes, and can reconfigure itself to handle over 200 simultaneous client connections.

The server has been installed in same Unix machine, where application servers are installed, at www.ee.umanitoba.ca at the port number 1116. The application servers use TCP/IP sockets to communicate with the servers (provided by the database API). Though TCP/IP sockets are not as efficient as Unix sockets in performance in the case of this database engine, they provide easier implementation and also chance of relocating the tiers 2 and 3 of this application on different machines in the future, if needed.

In the case of this application, mSQL has been used as a shrink-wrapped database engine and its performance has not been independently investigated. As far as this application is concerned the database engine can be replaced with more complicated ones when the load is higher than the capacity that the system is designed for. Specifically, the Java implementations of the application servers give the flexibility of using more than one database engine, when needed, given that the corresponding JDBC is provided [sub-section 9-2-1].

9-5-4 Database Tables of this Implementation

There are two main database tables created for this application shown in Figure 9-10 as "Tita60c" and "gps60".

Database table "Tita60c" stores the scheduled information of the bus route 60. It has been created based on the timetable of the bus route provided by the transit system. Since the timetable only shows the arrival times of the buses at major bus stops, the time of arrivals for other bus stops along the road have been linearly interpolated. It should be noted that such a linear interpolation may be another source of error.

The "gps60" database table, on the other hand, is essentially an empty table for accepting and providing the locations of the out-of-schedule buses.

To learn more about the queries made to each table, for the different purposes of this application, the clients' and servers' codes provided as the attachments to this thesis can be referred to. The codes are self-descriptive and well documented.

9-6 Implemented Programs and Related Files

The following list enumerates the name of the Java files related to this part of the application.

1. Clients:
   - Bus Routes (Route60Pembina.java)
   - Selected Bus Stops (YoekS60.java)
2. Application servers
   - Server3200pt.java, Server3200Threadpt.java, DataInquuiries3200pt.java
   - ServerGPS1, ServerGPSThread1.java, DataInquiriesGPS1.java
3. Utilities for table "gps60"
   - SrverGPS1NullAll
   - ServerGPS1NullColumn

9-7 References

1- R. Orfali, D. Harkey, J. Edwards, "Client/Server Survival Guide", 3rd Edition, © 1999, John Wiley and Sons, Inc.

2- P. Patel, K. Moss, "Java Database Programming with JDBC", 2nd Edition, © 1997, Coriolis Group Books.

3- G. Reese, "Database Programming with JDBC and Java", 1st Edition, © 1998, O'Reilly & Associates, Inc.

4- G. Reese, "mSQL-JDBC, The Imaginary JDBC Driver for MiniSQL", Copyright © 1999, http://www.dasein.org/soul/

5- P. Niemeyer, J. peck, "Exploring Java", 2nd Edition, © Sept. 1997, O'Reilly & Associates, Inc.

6- Hughes Technologies, Copyright © 2001, Hughes Technologies Pty Ltd. http://www.hughes.com.au

Section 10 – Access to the Transit Information Using Palm (OS 3.5 or Higher) PDAs

10 Access to the Transit Information Using PDAs Equipped with Palm OS 3.5 (and higher)

In the previous sections it has been shown how the information of the operating buses of the Winnipeg Transit System can be made accessible to the users connected to the Internet in a real-time basis. In this section providing the same information for the Palm Pilot devices equipped with Palm Operating System (OS) 3.5 or higher is discussed. Palm OS has been recognized as the world's most popular operating system for handheld computers and smart phones [1]. In this project, OS version 3.5 is used to show how mobile users can retrieve their desired information regarding the real-time status of the transit buses of a selected transit route. Such a specific example is intended to show how other devices capable of providing access to the pertaining database servers via wireless connectivity may be used to inform users about ongoing events in an ITS system.

In the subsequent sub-sections developing an ITS application for the purposes of this project on Palm Pilot OS 3.5 is fully described. In addition the hardware and wireless connectivity specifications of the device, as far as it is related to the performance of the present application is discussed.

10-1 Palm Computing Platform

Since its first use in the Palm 1000, released in 1996, the Palm OS has dominated the Personal Digital Assistant (PDAs) market. Before talking about the specifications of the Palm OS and taking advantage of its capabilities in developing an application, the Palm Computing platform (hardware and operating system) is investigated briefly.

Generally speaking, Palm handheld devices are designed to provide only the necessary data. Therefore, they have been optimized to view and enter small amounts of data in comparison to what is usually performed in the case of PCs. They are not miniaturized versions of desktop computing. Instead, they are designed as mobile computing platforms focusing on mobility as a user's unique experience.

They are physically small in size, have an ergonomic interface, and may be integrated easily to desktop computers.

In comparison with the corresponding specifications of PC systems there are the following significant peculiarities of Palm handhelds that should be considered when developing applications for them;

- Limited number of inputs, small screen size output, and small amount of memory
- Limited battery and processor power
- RAM as permanent data storage
- Require efficient code and user interface

10-2 Understanding the Palm OS

In this sub-section major features of the Palm devices, of interest to this project, are briefly introduced as follows:

10-2-1 Power Usage

Most Palm devices run on a pair of AAA alkaline batteries for weeks of normal operation. Since the real-time clock, interrupt generation circuitry, and memory constantly require a small amount of power to operate properly. The power must not be completely turned off. In the case of memory, since the applications and permanent data are all stored in RAM, loosing power means loosing all data. In Palm OS, in terms of power consumption, there are three modes of operation:

- Sleep mode – in this mode the device is turned off, recognizable by shut down display, digitizer and main system clock. However, the essential systems such as interrupt-generation circuitry, real-time clock, and RAM data holding mechanism remain operative. In such a condition pressing either of the hardware buttons, or receiving input from the serial port will wake up the device. If the device is not used for a time interval, which is user-customizable, between 1 to 3 minutes it goes to sleep mode.
- Doze mode – in this mode the main clock, LCD screen and digitizer are turned on. The processing clock is running, however, it is not executing instructions. When there is no input

from the user, a device enters doze mode. In this mode, the device appears to be "on". In this mode any interrupt received by the processor brings it out of the doze mode and leads it to the most active mode.

- Running mode – in this mode the processor actively executes instructions. The device remains in the active mode for the time that it is executing the user's input (usually for one second, at most). After the instruction execution period, the processor goes back to the doze mode. Most applications keep the processor in the running mode for only 5 percent of the time.

10-2-2 Running Palm OS Applications

The Kernel of the Palm OS is preemptive and multitasking. However, the User Interface Application Shell (UIAS), the part of the OS responsible for managing the applications, runs only one application at a time. Usually UIAS is the only task running, which calls application code as subroutines. It relinquishes control until the presently running application quits. At this point the UIAS runs the next application as another subroutine, right away. Since applications are run within the single thread of the UIAS, they can not be multithreaded. Using certain calls an application can cause the OS to launch a new task. As already mentioned, the system tasks are prioritized and can be performed simultaneously. However, it is only the system software that can launch a new task, and an application code does not have access to the Palm OS multitasking APIs.

When the OS launches an application, it calls a function named PilotMain and passes it a launch code. When an application is launched normally, it displays its user interface, starts up a loop called an event loop and starts processing an event queue. Depending on the launch code passed to the application, it may be launched normally as mentioned above, or be asked to perform some small tasks. The application may be asked to search its own database for a string, receive specific notifications (e.g. HotSync has just been completed), or open to a specific record, without displaying its User Interface, and then exit. When an application is passed a launch code other

than normal launch, its event loop is not started and control is passed to another function of the application that is responsible for performing the job (outside of the event loop).

The Palm OS application is event driven, and receives events from the OS. During a normal launch, the execution control is passed to the application's event loop. This loop retrieves the events from the event queue and dispatches them according to their type. Events are data structures conveying the type (for example, a stylus tap on an on-screen button) and the related information (such as the screen coordinate of a stylus tap). The application's event handler may handle an event, if the application itself is interested in the event, or it may be passed back to the event loop. Most common events such as displaying menus or determining what button on the screen was tapped, however, are passed back to the OS, because the system has facilities to deal with them.

When a Palm OS application receives a "closing application" event, it takes the control from event loop and passes it to another function responsible for cleaning up operations and prepares to shut down.

Passing the events back to the OS functions, which is a typical procedure in normal Palm OS applications, guarantees appropriate management of the device power. That is, it causes the processor go into doze mode when there is no further event to process or the operating system's auto-off feature is activated and puts the device into sleep mode when the device is left on for a few minutes.

10-2-3 Memory Management

Palm OS uses word "card", as a logical abstraction, to describe a memory area containing ROM and RAM. In the Palm OS 3.5 there is only one card available (card 0). Each word of memory has 32-bit length with the address register of the same length. Therefore, memory of up to 4G word-space could be addressed through the address register to store data and code. The data types, however, could be 8, 16, or 32 bits long. The Palm OS reserves 256 MB of the address space for each card. The RAM part of each card is divided into separate areas: dynamic RAM and storage

RAM. Both areas are used to implement memory heaps that are contiguous areas of memory containing smaller units called chunks. Each chunk is another contiguous area of memory having size of 1B to 64MB. All data in the Palm OS environment are stored in chunks.

Applications use the Palm OS memory manager to allocate, manipulate and free memory in the dynamic heap. Dynamic RAM consists of one heap.

The Palm OS dynamic heap provides memory space for the following purposes:

- System and application global variables
- System dynamic stack allocations (for example in the case of TCP/IP and IrDA)
- Application stack allocations
- Application temporary memory allocations
- Application dynamic allocations

Figure 10-1 illustrates the memory map in Palm OS 3.5.

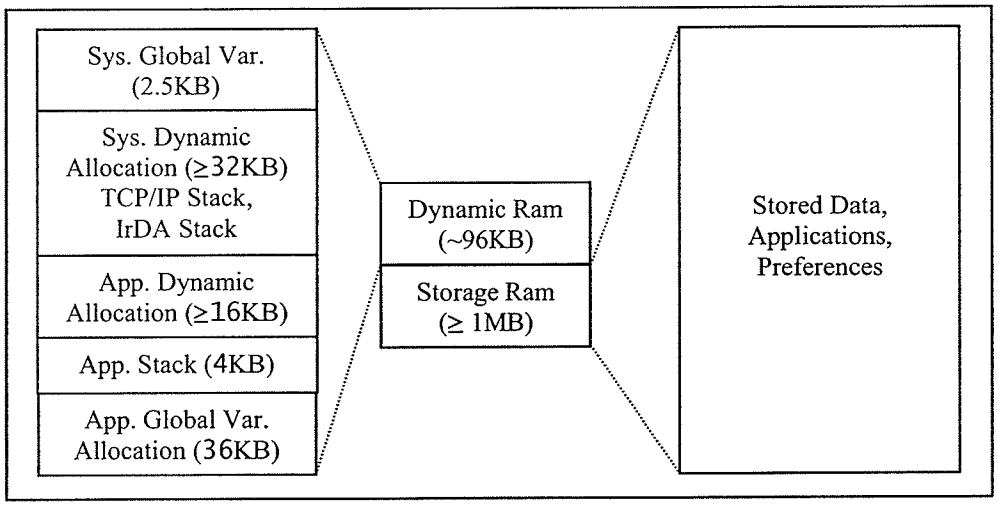


Figure 10-1 – RAM memory map in Palm OS 3.5.

In the Palm OS version 3.x storage RAM consists of one heap in which each memory chunk is called a record. Each record is part of a database that, in turn, is a list of memory records and some database header information. The Palm OS uses its data manager to implement and manage records. Most of the time, records in a database share some associations, such as they represent an appointment in the Date Book. The data manager provides functions for creating, opening,

closing and deleting databases. It also provides functions to manipulate records within databases. Depending on the content of its records, a database may represent an application, the stored data of an application or a shared library. A database in the Palm OS is very similar, in concept, to a file in a desktop computer.

In the Palm OS, applications do not copy data from a storage heap to the dynamic heap for modification. The Palm OS uses data and memory managers to lock individual chunks of memory and edit them in place - without relocating them to other places. The only restriction that exists for the locations of the individual records of a database is that they should reside on the same memory card. Other than that, records of different databases may be interspersed with each other. They may also be located in ROM as part of the applications shipped with the OS.

10-2-4 Application Resources

In the Palm OS, an application is a modular entity composed of executable code, data, and user interface elements, called application resources. Application resources can be categorized as follows:

- System resources, which are used by the system and include the application code itself, startup information required for launching the application, and data structures of the application's global variables that should be initialized. These resources are usually created automatically by the development environment using the source code written by the developer.
- Catalog resources, including various user interface elements, from labels to buttons, used by the application. They are identifiers created and supplied by the system developer to be used by the application code during execution.
- Project resources, including all application modules that are referenced throughout the application, such as forms, alert dialogs and menus.

A Palm OS application can be known as a resource database of its code, user interface, and other necessary resources that let it work properly. In the application developing procedure these

database files are referred to as PRC files. Having the resources of an application in separate modules helps the main code stay untouched not needing to be recompiled when some of modules are to be changed.

10-3 Development Environment

Among many tools available for developing Palm OS applications, Metrowerks CodeWarrior (version 7.0 - as used in this project) is the official Integrated Development Environment (IDE) supported by Palm computing. It uses C/C++ languages for developing source code, and contains the following components in its package:

- Constructor, which is a resource editor with a graphical interface. It is used to create user interface elements of the application that are combined with the source codes (by other tools of IDE) to create a finished program.
- Compiler and linker, which turn source codes into object codes for Motorola 68000 series (DragonBall or DragonBall EZ) processors, and link them, respectively.
- PalmRez, which changes the linked object codes generated by other components into a .prc file suitable for installation and execution on a Palm device or Palm OS emulator.
- Palm OS Emulator, which imitates most of the hardware and software functions of an actual Palm OS handheld.

10-4 Palm OS Emulation

As a debugging tool, the Palm OS Emulator (POSE) emulates a handheld at the hardware level and provides a much faster way to evaluate a Palm application program than running it over a real handheld. POSE can do almost anything that an actual handheld device is capable of doing, with only a few exceptions which originates from the differences between the desktop and handheld systems. In this implementation Pose has been used for all debugging and demonstration purposes. Among its other features, the emulator can be set to redirect the network library calls in POSE to the host system's TCP/IP stack, which is very useful when emulating a handheld device's performance, equipped with a wireless connection to the Internet, is of interest. In this

application this capability of the emulator is used for testing the application performance in retrieving the desired data from a database over the Internet.

POSE only emulates the hardware of the handheld device. To work properly, it also needs a ROM image containing the Palm OS system software. The latter is obtainable through either the Palm Inc. web site or directly from an actual Palm OS handheld.

10-5 Palm Programming – Application Development

In this sub-section the important aspects of the features of the Palm OS application developed for this thesis are reviewed. A comprehensive understanding of the way a handheld is programmed, the general features of application programming and other related details may be studied in the references [7-9]. In the figures below, the POSE has been used to simulate the appearance of a real handheld when the application executes.

Figure 10-2 shows how the BusNet (the chosen name for this application) application icon appeares among other icons in the main application display of the handheld.

When the BusNet icon is clicked the main user interface of the application is displayed. As indicated in the figure the "bus routes" form of this application corresponded to the bus routes available. The route numbers not used by the transit system are indicated by "*" sign. It has been suggested that the user interface of this application benefits the user in the following ways:

- Reduces the number of taps required to access various program functions,
- Provides various functions.
- Provides the most frequent functions in easily accessible manners.
- Provides the transfer between forms fast and infrequently as possible.
- Uses on screen buttons as quick user interface elements rather than using other elements such as long pop-up lists.

If an on-screen button is tapped by the stylus pen and a bus route is already allocated to the button a new from is displayed on the screen as shown in Figure 10-5. However if the tapped button is a reserved one labeled by "*" to which there is no bus allocated, a dialog box is displayed on the

Figure 10-2 – shows the result of loading BusNet application in the application group of the handheld.

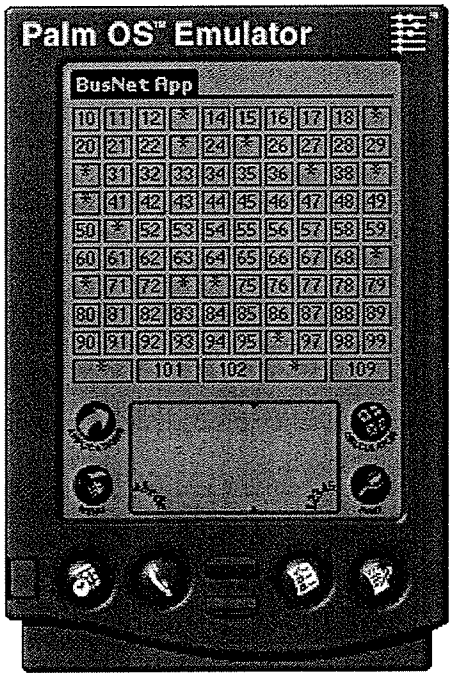


Figure 10-3 – The bus routes form of the BusNet application.

screen as indicated in Figure 10-4 to show there is no bus route allocated to the button. Clicking the "OK" button of the dialog box makes it disappear. On the other hand, there might be buttons on the screen that are not used frequently. For instance, a user may normally

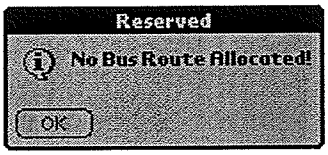


Figure 10-4 – The dialog box appeared on the screen when a non-allocated "*" button is tapped.

like to have access to the information of only a few bus routes at a time. Therefore, there is no need to load the memory with the information of the bus routes that are not of interest. When such information becomes necessary Palm OS conduit software can be designed to synchronize the user's needs with the bus routes information carried on the handheld. This approach eliminates the possibility of wasting the device resources with the unnecessary information. The program that is being presented here occupies almost 30KB of the memory space with the information of only one bus route. A new route information may also require 20KB - 25KB of memory. When an allocated, but unloaded, button is tapped, the form shown in Figure 10-5 is displayed on the screen. The information in it denotes the name of the bus routes with data accessible through the handheld. In the case of the Figure 10-5, route 60 is shown that having data available through this device.

In Figure 10-5, user may tap the "Back" button to return to the display shown in Figure 10-3. By tapping button "60" in the Figure 10-3 a new form containing the information corresponding to the bus route number represented by the button is displayed on the screen.

Figure 10-6 shows the device display corresponding to route 60- Pembina. As can be seen, the displayed screen contains a bit-map that is similar to the cartoon map already used in other parts of this project. As emphasized in Section 5, using such a map provides simplicity,

Figure 10-5 – The form displayed on the screen resulted from tapping the buttons that are allocated but not loaded with the corresponding route information.

ease of use and consistency for the end user.

Along the route shown on the map, there are star like signs that represent the major bus stops of the route. Tapping each mentioned point brings up the name of the bus stop in the line titled "Bus Stop". When the desired bus stop is selected a tap on the "Update" button, shown in the Figure 10-5, starts a process of loading the desired information. The device uses its wireless connection capability to contact a server to retrieve the information related to the arriving time of the next buses to the selected bus stop. The server accesses the required database tables to obtain scheduled as well as real-time information in a way very similar to that already explained in Section 9, as in the case of providing the same information to end users through Java applets and web browsers.

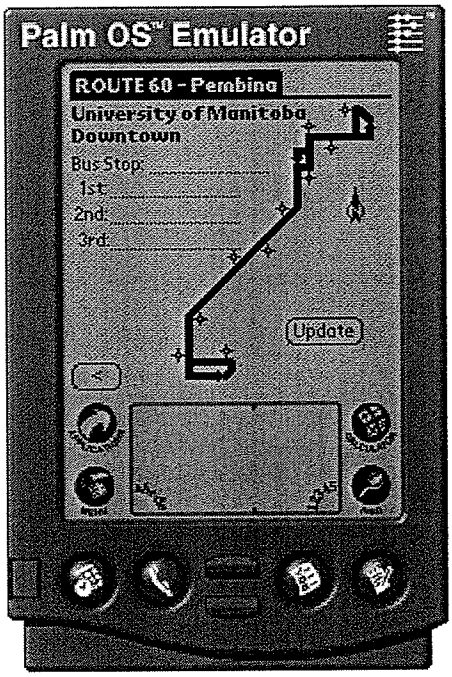


Figure 10-5 – The form displayed on the screen containing the information corresponding to Route 60 – Pembina.

Tapping the button represented by the arrow sign " < " takes the user back to the form displayed in Figure 10-3. Tapping the "BusNet App" form of Figure 10-3 brings up an "Options" menu bar that provides an "About BusNet App" menu item. Tapping the menu item leads the user to the "About BusNet" form containing the information about the developers, as shown in Figure 10-6. Touching the "OK" button in the latter screen by stylus pen leads the user back to the main application's form shown in Figure 10-3.

To let the Palm application described above retrieve the desired information there needs to be applications on the server side that read the latest information every minute, continuously, and write them in the files accessible by the handheld to obtain the updated information. Figure 10-7 shows the programs developed in this application that work together to provide the desired information for display over the handheld device. The servers applications retrieve the desired information from the corresponding database tables based upon the requests received from the

Java client applications. These applications are responsible for making database inquires in 1-minute time

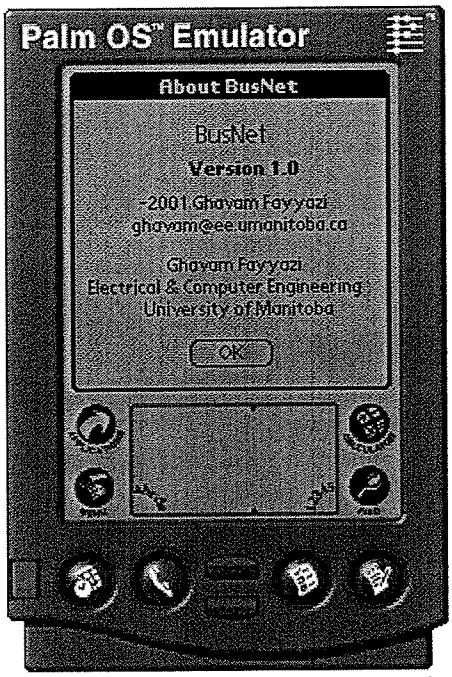


Figure 10-6 – The "About BusNet" form indicating the software and developers' general information.

intervals and writing the results in the html files corresponding to each major bus stop along the road. The Palm application, as explained above, is responsible for retrieving and displaying the information written in the html file corresponding to the desired bus stop.

The Java files corresponding to the applications developed for this section are presented in the appendix, and/or available as the softcopy. The complied codes of the Palm application, as well as, the main C files of it are also included in the same way. The following list enumerates the name of the files developed for the Palm application of this thesis.

- BusNet.prc, including; BusNet.c, and BusNet.rsrc among other library and header files generated automatically by the IDE software or publicly available through various resources.
- Java files, including: ServerPalm60.java, ServerPalm60Thread.java, DataInquiryPalm60.java, PalmDBReaderFWriter.java, PalmFilesWriter.java.

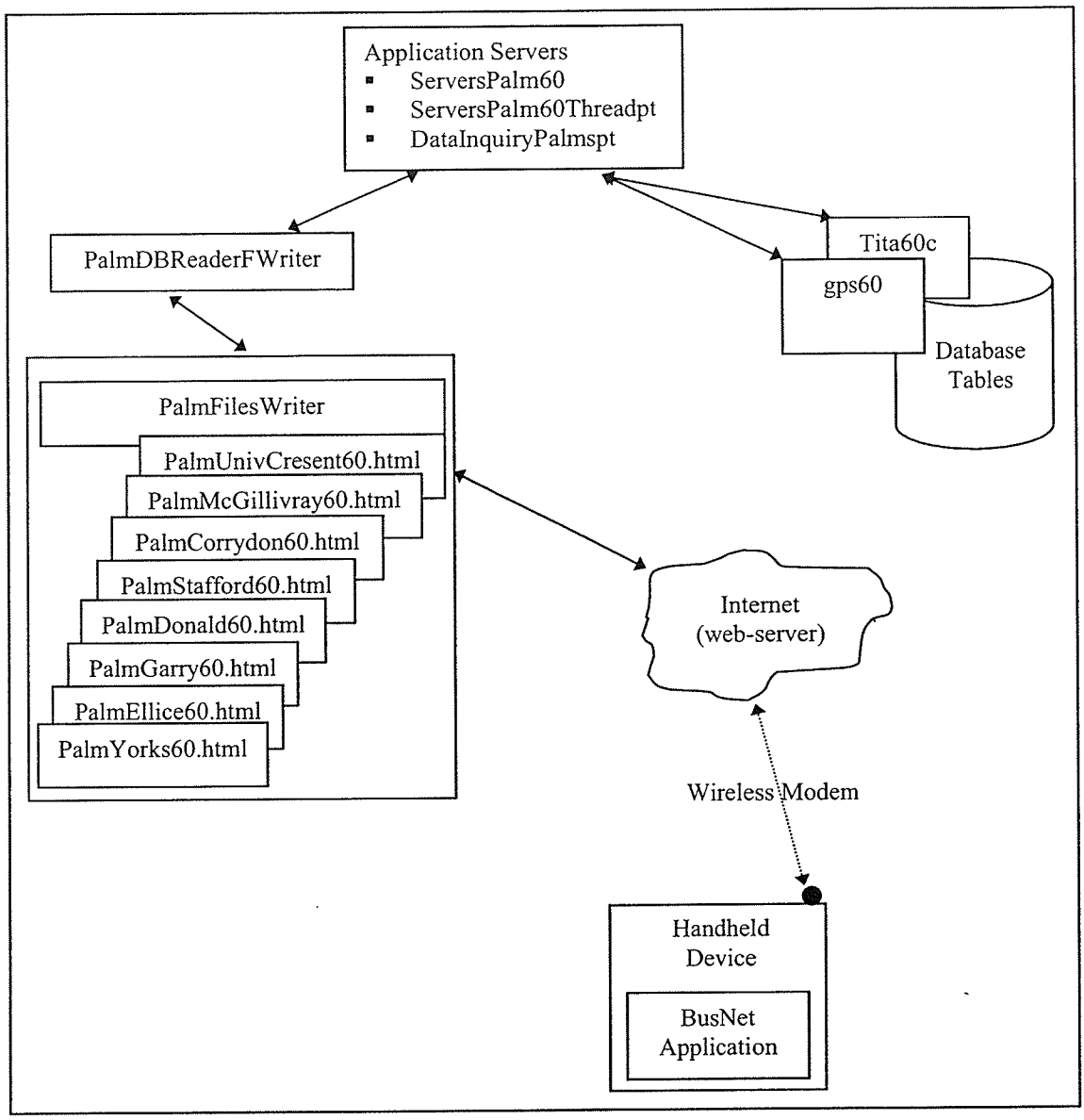


Figure 10-7- Files in the server side used for acquiring the desired information with respect to the arrival times of the next buses to the selected bus stops.

- Html files, including; PalmUnivCresent60.html, PalmMcGillivray60, PalmCorrydon60.html, PalmStafford60.html, PalmDonald60.html, PalmGarry60.html, PalmEllice60.html, PalmYorks60.html.

The Palm application development as performed in this thesis is terminated at this sub-section. The remaining question, however, is "How is the wireless connectivity implemented in the handheld?". The answer to this question needs investigation from a hardware point of view

beyond the scope of this thesis. It is assumed here that the hardware blocks are employed as black boxes controlled through corresponding drivers. Nevertheless, as will be discussed briefly in the next section, a Palm handheld may be considered as a device having characteristics that make it capable of assuming roles more than that described in this project. In the next sub-section the capabilities of the handheld devices equipped with Palm OS 3.5 is very briefly reviewed.

10-6 Communication with Other Devices

In this sub-section communication of the Palm OS with other devices, concerned in the present project, is investigated as follows:

10-6-1 Palm and Serial Port

Serial port communication capability of Palm OS is attributed to be a part of the platform's success and popularity, because when used with the right cable and/or third party hardware it lets the PDA device talk to any thing from modems to GPS receivers. The serial communication protocol is also used to synchronize the device through a cradle with a desktop computer. The Palm OS serial communication supports several layers of complexity including byte-level serial I/O and high level error-correcting protocols.

In this project there is no implementation of this protocol in the developed application, as it can be found as third party products when the PDA is required to be connected to real hardware. In this project, however, there are three cases where the device should communicate with other devices through serial port. These include:

- Communicating with the desktop computer to load the developed application, or add new components (required bus routes information as described above),
- Communicating with a wireless modem to acquire real-time data
- Communicating with Bluetooth devices in the case where the PDA is used in conjunction with the Bluetooth auxiliary systems described in Sections 7 and 8.

The serial port in Palm OS devices are stripped-down version of what might be found in the serial port of a desktop computer. The following external signals are used in the Universal Asynchronous Receiver and Transmitter (UART) chip of the Palm OS hardware:

- Clear to Send (CTS)
- Request to Send (RTS)
- Transmit Data (TD)
- Receive Data (RD)
- Signal Ground (SG)

The serial communication may occur in the speed range of 300bps to 115200bps, arranged in Motorola's big-endian byte order. This is an important difference between Palm OS and an Intel-based machine that uses little-endian byte order when two machines are to be connected through their serial ports. In this case, when two machines have to communicate multi-byte data, the byte order in either end (preferably the desktop end) should be reversed.

From the software point of view, the serial communication of the Palm OS has several layers. Figure 10 -8 shows the Palm OS serial communication stack.

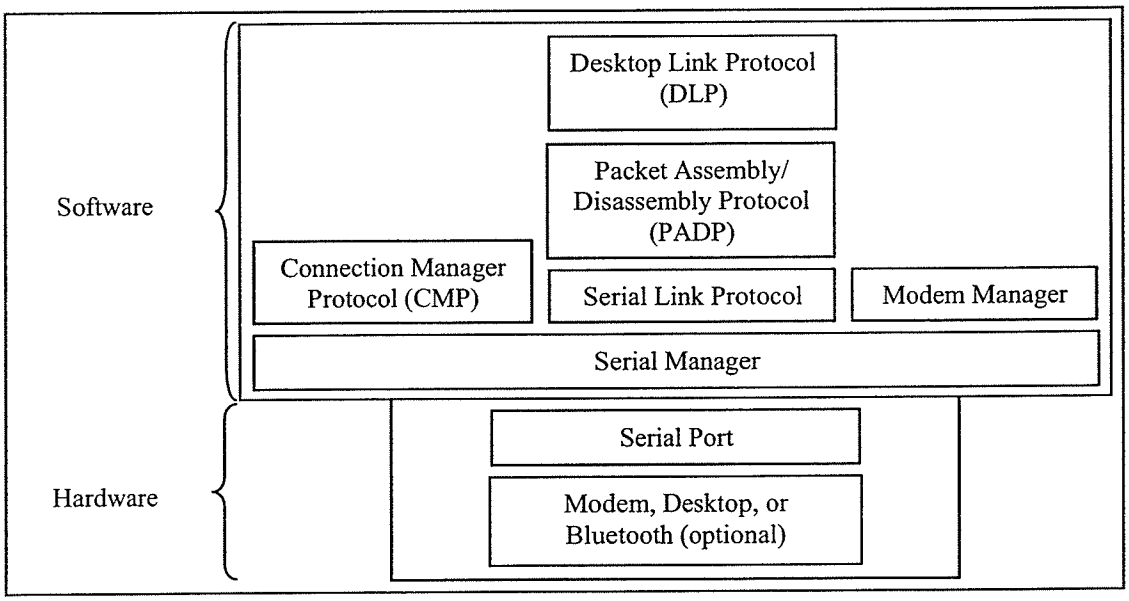


Figure 10-8 – Layers of the Palm OS serial communication stack.

In Figure 10-8 the following layers of software are as follows:

- Serial Manager. At the lowest level it provides direct control of the port signals. In the byte level communication this makes it the most flexible layer to use in custom applications.
- Modem Manager. It provides the necessary API for modem dialing and control. The modem might be directly or via a cable connected to the Palm OS device.
- Serial Link Protocol. Packet Assembly/Disassembly Protocol, and Desktop Link Protocol. These three protocols together provide remote access to various Palm OS subsystems including data storage specifically for HotSync conduit applications.
- Connection Manager Protocol. This protocol is used by the OS to negotiate the connection specifications with outside communication software and alter connection profiles used by the OS to connect applications via serial, IR, or network communications.

Among all layers of the serial communication stack, the serial manager is the most flexible way to connect a Palm OS application to another device.

10-6-2 Palm and Wireless Connectivity

Generally speaking wireless technology can be thought of as replacement for cables in either of the following network categories:

- Personal Area Networks (PAN) that link the devices in close proximity (10-20 meters), formed spontaneously (ad-hoc), not requiring extension infrastructure.
- Wireless Local Area Networks (WLAN) that replaces the Ethernet cables accessed through fixed based stations or "access points" with a range of 150 – 200 meters within the existing infrastructure. To learn more about Palm support for 802.11b, which has become the indisputable standard for WLANs, and add-on hardware modules for Palm devices see reference [6].
- Wide Area Networks that are the best known of the three categories. These technologies include cell phone technologies like TDMA, CDMA, and GSM [4]. A Palm handheld device may be wirelessly enabled in this mode in one of the following three ways:
  - Using Palm.Net® wireless service as in the case of Palm VII, VIIx, i705 handhelds,

- Using a Mobile Connectivity Kit (software shipped in the handhelds), a wireless modem (or compatible data-enabled mobile phone), Internet service Provider (ISP), and data services from mobile carrier as in the case of Palm m125, m130, m500, m505, or m515 handhelds,
- Using third party developed applications that use wireless modes (or data enabled mobile phones) along with other required services mentioned in the previous case.[3]

In any event a palm device is set to connect to the Internet. Palm Computing uses Web Clipping and Palm Query Application (PQA) methods in the server and client sides, to access web pages. To learn more about the mentioned methods reference [9] may be consulted.

10-6-2-1 Palm and Bluetooth

Palm OS with Bluetooth wireless capabilities makes other interesting handheld features possible [5]. In this project, such integration could be of great importance if we consider Bluetooth signposts (isolated or networked) as suggested in Sections 7 and 8. This system would provide the Bluetooth-enabled PDA holders who are in the signposts proximities (the pedestrians and users at bus stops) with the real-time information of the routes of interest. The PDAs do not need to have a wireless modem (embedded or through cellular phone) to be able to acquire the transit system real-time information.

An application developed for such a purpose could be written using either of the following capabilities:

- The Bluetooth API (which results developing Bluetooth-aware applications),
- The Bluetooth virtual serial driver (which is more appropriate when a serial port oriented application is being developed),
- The Exchange Manager/ Bluetooth Exchange Library (which uses the easiest mechanism to exchange data),
- TCP/IP (which requires no modification at all, but needs an available Bluetooth-enabled LAN access point (LAP or AP) in proximity using PPP over RFCOMM).

The application developed for this project is capable of being connected to the Internet using TCP/IP capabilities of the Palm OS 3.5. In the main bus stops where real-time information of the system is provided to the end users through the Internet, a Bluetooth Internet access point can easily be installed. In the case of the a networked Bluetooth signposts, as suggested in Figure 8-4, a Bluetooth PDA that is in proximity of the network could access to the Internet, automatically. Figure 10 – 9 shows the components in the Palm OS stack that is related to the Bluetooth layers. Generally speaking Palm OS software supports the following Bluetooth profiles [2]:

- Generic Access
- Service Discovery
- Serial Port
- Dialup Networking
- LAN Access point
- Generic Object Exchange Profile
- Object Push

Major specifications of the Palm Bluetooth software, as far as this project is concerned, can be summarized as follows:

- The Palm Bluetooth software is an add-on software for Palm OS 4.x (Note that in this section developing the Palm application, using OS 3.5 was investigated, as such, developing Bluetooth-aware applications is not discussed further than what is being presented here.)
- The add-on is a certified Bluetooth-1 compatible stack.
- The software provides cell phone, and serial connections, Internet/ network, and HotSynch access.

To learn more about the profiles, reference [2] or other references mentioned in Section 8 may be consulted.

As described in sections 7 and 8, the Bluetooth signposts are not considered as the main components of this project and have been introduced as auxiliary positioning devices. In fact,

using Bluetooth technology in accompaniment with isolated and/or networked signposts to develop a new positioning system needs a separate investigation. Nevertheless, as mentioned earlier, since the application developed in this project uses the TCP/IP library of Palm OS, it could be run with no problems on the devices that have the connectivity through Bluetooth Internet access points.

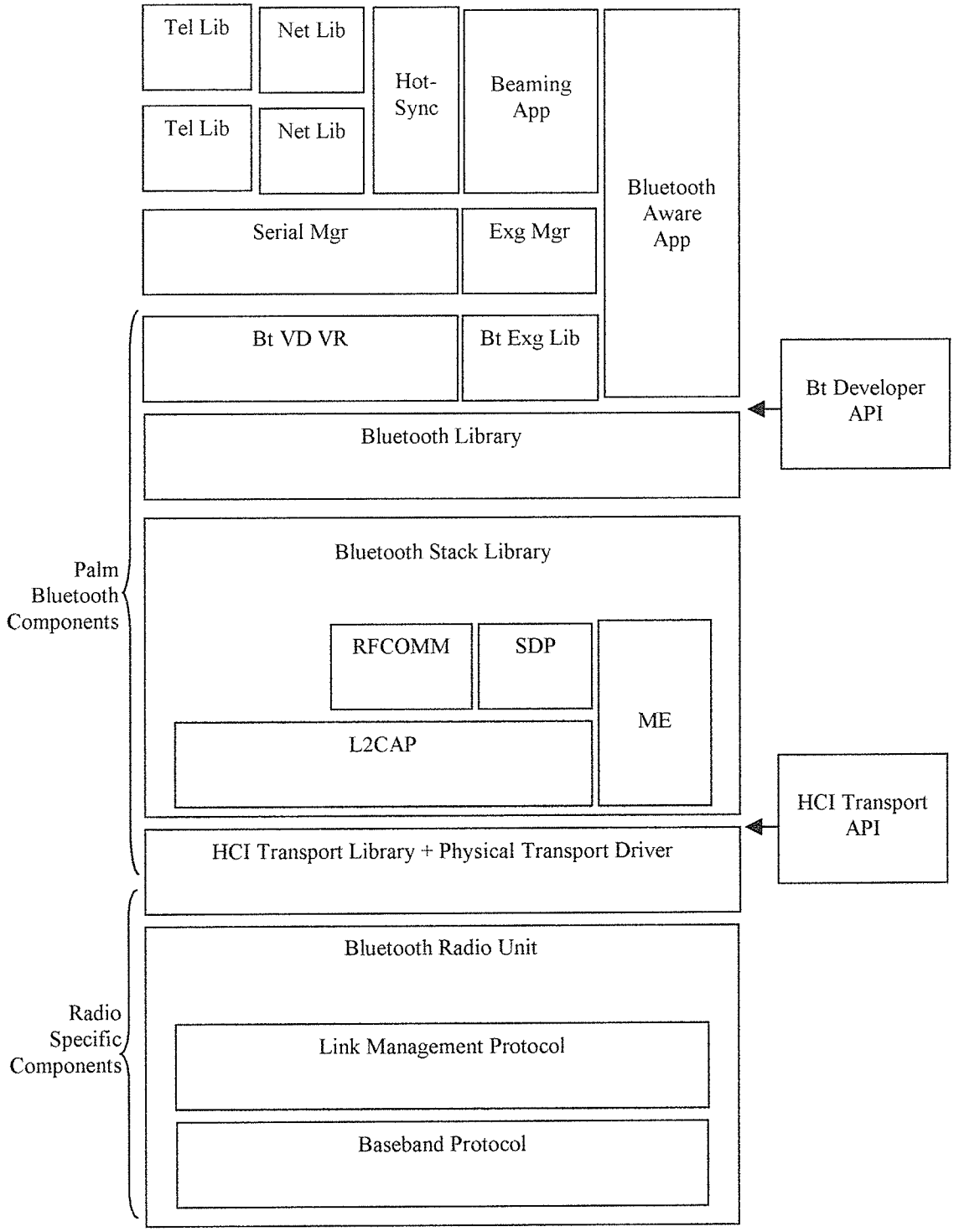


Figure 10-9- Bluetooth stack and its relationship with the components within Palm OS [2].

10-7 References

1- Palm Inc., http://www.palmsource.com/palmos/

2- Palm Inc., "*Bluetooth White Paper*", Copyright © 2001, October 22, 2001, http://www.palmos.com/bluetooth, and http://www.palmos.com/devzone

3- Palm Inc., "*Wireless Email Solutions*", Copyright © 2002, http://www.palm.com/wireless/email/others.html.

4- Palm Inc., "*White Paper: Providing Fluid Connectivity in the Wireless World*", Copyright © 2002, http://www.palm.com/products/accessories/expansioncards/bluetooth/

5- Palm Inc., "*Palm Bluetooth card*", Copyright © 2002, http://www.palm.com/products/accessories/expansioncards/bluetooth/

6- Intel Network Connectivity, "*Xircom® Wireless LAN Module for Palm* Handhelds*", ©2002 Intel Corporation, http://www.intel.com/network/connectivity/products/xirpwe1130.htmtel

7- S. Mann, R. Rischpater, *"Advanced Palm Programming"*, Copyright © 2001, John Willey and Sons, Inc.

8- N. Rhodes, J. McKeeban, *"Palm Programming, The Developer's Guide"*, Copyright © 1999, O'Reilly and Associates, Inc.

9- L. R. Foster, *"Palm OS Programming Bible"*, Copyright © 2000, IDG Books Worldwide, Inc.

Section 11 – Discussions and Conclusions

11 Discussions and Conclusions

In the previous sections the details of the project, technology integration, and the implementation of the various components used within this prototype were explained to the extent suggested by the scope of this thesis.

To complete this work, however, it is required to re-consider at least two sources of ambiguities in the methods presented for: 1) positioning of an operative bus, and 2) anticipating its arrival time to the next bus stop when it is detected to be out-of -schedule.

Also, this section briefly overviews the work required to change the present prototype to a real operative system, working on the scale of a city.

11-1 Timetable Creation, Interpolation, and Adaptation

So far, this project has been about providing real-time information of a transit bus operating on a scheduled basis. The schedule is important because it saves communication channels from being over used by not sending data that conveys no information.

A typical bus schedule, however, does not reflect the actual positions of the bus along the road at any given time. Instead, it usually provides only the information about the time that the bus is in known locations of the road (major bus stops). Therefore, between major bus stops there is no decisive information about where a bus might really be.

On the other hand, bus movement may be estimated to be linearly interpolated between major bus stops. A linearly interpolated approximation means that the bus keeps almost constant speed for a reasonable long time or distance along the road (between subsequent major bus stops of the road). In this way the short stops or reductions in speed can be compensated for (in a minute or less) when the bus moves with a speed higher than its average.

More specifically, recall that in creation table 6-1 there was one assumption made and one observation as follows (the table is the main source of scheduled information based on which the real-time positioning information of an operative bus is assessed and potentially reported to the corresponding servers):

- Assumption: buses move with constant speeds between major bus stops,
- Observation: There are segments (not just single points) of the road where buses might be in any given minute of the time

The assumption is valid for the primary treatment of the problem as explained in Section 6, as long as a moving bus can reach the next segment of the road in one minute. In such a condition it doesn't matter if the movement within each segment is linear or not. Figure 11-1 compares two imaginary cases more closely.

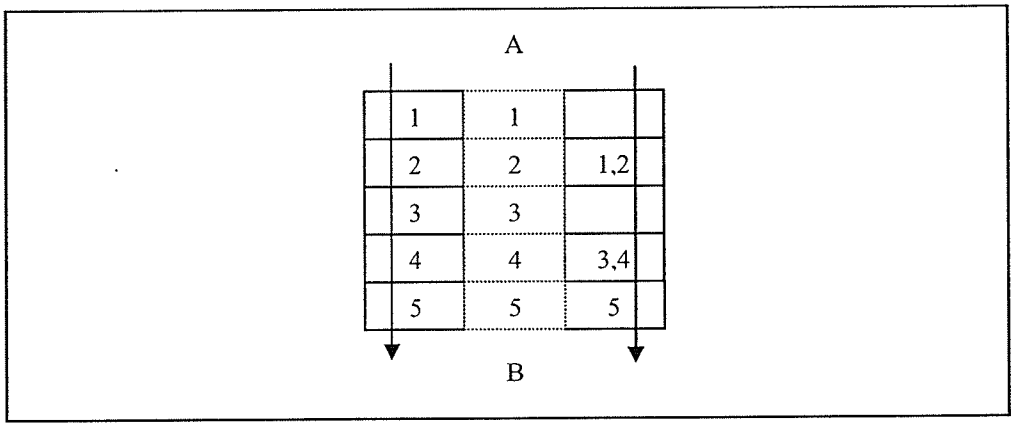


Figure 11-1 – Comparing linear (left arrow) and non-linear (right arrow) movements of a bus travelling along the part of the road between points A and B.

In Figure 1, the two columns at the both sides of the big square represent the part of the road that is between two major bus stops A and B. The middle column is used as index to represent the time unit in minutes, segmenting the road between points A and B to the parts corresponding to each minute. The bus moving with the pattern shown by the column at the left-hand side has (may be attributed to have) a constant speed. In another words, it can be assumed that the bus stays in each segment of the road corresponding that minute of the time. On the other hand, the bus having the moving pattern shown in the right hand side column, stays in the segment 2^{nd} by the end of minutes 1&2 and in the segment 4^{th} by the end of minutes 3&4, respectively. In the latter case the bus movement may not be attributed to be linear on minute time scale. Therefore, in this case, the validity of the above-mentioned assumption about the linear movement would not hold.

The latter case may happen to a bus travelling on a non-busy road that has intersections with busy roads, where the bus may have to wait for more than a minute to receive green light permission to continue its trip. Such a non-linear behavior may also be time variant, as well as, location dependant. The pattern of this non-linearity, however, may be repetitive during a long time interval like a year, half a year, or a season. Therefore, if the pattern is recognized it can be used for the time interval that it is valid. When the pattern is changed for any reason, the corresponding part of a detailed timetable should reflect the changes. Having precise pattern of the movement of the buses on any segments of the road could help prevent making unnecessary reports to the servers, because there would not be any discrepancy detected between the bus movement and its schedule.

The capability of the real-time monitoring of the buses' movements along the road, provided by the present system, makes extracting the statistical information of the scheduled behavior of the buses, possible. In order to extracting the movement pattern, however, appropriate methods of pattern recognition should be employed. For a non-linear system like the one studied here employing Artificial Neural Networks (ANN) is strongly recommended.

A neural network is a network of very simple processing units highly interconnected by unidirectional connections carrying numeric data. The units operate only on the local data they receive via the connections. The architecture of the processing unites (and the network) was inspired by the design and functioning of human brain. Neural networks can learn (adjust the weights of the connections) from examples and exhibit structural capability for generalization. As universal approximators, they are also able to represent any linear, non-linear, simple or complicated function (Kolmogorov's Theorem, 1957). The high degree of interconnection of neural networks allow for a high degree of parallelism. They do not have any idle memory containing programs and data, instead, each processing unit is programmed and continuously active.

Generally speaking, a neural network can be characterized by the following parameters:

- Processing units.
- The way that the units are connected.
- Transfer functions of the units, and the rule of signal propagation through the network.
- Training algorithms.
- Environment in which the network operates.

To learn more about neural networks, their implementations and usage in transportation systems the references [1-4] may be consulted.

Unlike pure statistical methods, neural networks are not as restricted by the number of input data, and do not assume that there are relatively simple dependency functions (linear or logarithmic) over the data set (as is the case in applying the regression methods). More specifically, as Professor P. D.M. MacDonald, of McMasterUniversity, Ontario, Canada says "Traditional approaches to statistical inference fail with large databases, however, because with thousands or millions of cases and hundreds or thousands of variables there will be a high level of redundancy among the variables, and there will be spurious relationships, and even the weakest relationships will be highly significant by any statistical test. The objective is to build a model with significant predictive power. It is not enough just to find which relationships are statistically significant.". As a matter of fact, it has been shown that conventional regression methods are particular cases of neural nets (with one layer and linear threshold function)[5].

Depending on the purpose of the usage, a neural network may be designed to use supervised or unsupervised training schemes. In the former case there is a set of data vectors that the responses of the network to them are already known. Therefore, the network output is used to calculate error functions used for adjusting the connection weights of the network in order to generate the output with minimum errors. In unsupervised training schemes there is no a priori knowledge of what the output should be for any given input. The network acts like a regulatory detector and tries to discover structures in the patterns presented to it. Such an approach could be used for deriving

the transit system schedule when the bus movements are constrained by the traffic conditions of the road rather than the time limitations introduced by the route time table.

When the schedule is created based on the road conditions, passengers' demand levels, and/or the transit system policies, a supervised NN can also be used for modification and/or interpolation of the route's time-table for the road segments where a detailed pattern of the buses' movements are not available.

A suggested architecture for neural network modules deployed in transportation systems (as well as, many other systems) that involve pattern recognition is the Bayesian-based Probabilistic Neural Networks (PNN) [6]. To learn more about the topic references [7,8] might be of interest.

Once a bus's movement pattern in every segment of the road at different times of the day are fully recognized the transparent part of the route timetable (from the user's point of view) can be modified in the corresponding database tables. Such patterns may not be linear, as adopted in the primary analysis employed in this system. Applying such modifications to the timetable can save the system from making unnecessary reports to the servers.

11-2 Arrival Time Anticipation

The second source of ambiguity in the present system, first introduced in Section 9, appears when the arrival time estimation of an out-of-schedule bus to the next bus stop is accomplished. Recall that the program takes the bus average speed to calculate the area that should be searched for existence of any out-of-schedule bus in a two-minute time distance from the bus stop. The assumption that the operative buses can be attributed to have constant speeds for a two-minute traveling time may be acceptable in a primitive system implementation. The assumption, however, can not be justified or the performance of a real system based on such an assumption may not be acceptable for the users, if they find that the system is not able to provide reliable anticipation of the buses' arrival time when they are out of schedule. One powerful tool that is shown to be very reliable in arrival time anticipation (even when compared with NN employed in real transit systems) is Kalman filters [9].

The Kalman filter as a set of equations that provide an efficient recursive computational solutions of the least-squares method. The filter is very powerful in using past and present states of a system from which the future states of it may be estimated [10]. More comprehensive explanation about the subject may be found in reference [11]. A good source of information, tools and working knowledge about the filter can be found in reference [12].

Both the mentioned adaptive methods are strongly dependent on the probabilistic and statistical information of the real-time performance of the system. Therefore, in the present implementation they are not being used as operative components of the system, as there is no real information available at the stage of the prototype development.

11-3 Real System Implementation

The system presented so far is a prototype based on which a real system can be developed as briefly described as follows:

11-3-1 Software Implementation

As explained in previous sections the various components developed for the software part of this prototype are completely expandable and could be used in a real system. The following features envisioned in the prototype implementation guarantee such scalabilities of this system:

- What has been done for a typical route of the transit system can be performed for any other route of the system requiring no change to other parts of the system. In essence, the system takes advantage of being completely modular in design.
- The GPS receiver simulator needs the minimum effort to be substituted with a real GPS receiver. The analyzer software can work with a real GPS receiver by simply adding the capability of a serial port. The present form of the software, however, is helpful for use in a single computer system for demonstrating or developing purposes.
- Though the various servers are very similar in functionality, they are intentionally implemented separately to provide the possibility of changing the application protocols in future when new functions are found that should be added to the system.

- The web clients follow certain templates when developed. Therefore, adding new routes to the system is a matter of replacing the new route's specific components to pre-defined and implemented functions of the previously created clients.
- Java object orientation capability is another desirable feature for the horizontal and vertical scalability of the system.
- The Palm part of the application provides the facility to add or eliminate the information of new routes to the hand held device without requiring any change to the other parts of software including its GUI.

11-3-2 Hardware Implementation

There are two main options in hardware implementation, installed in the buses, described as follows:

- The hardware can be a PC motherboard with required peripherals to accommodate a Windows OS. Such an approach can minimize the changes needed for the present prototype, as it is, to be used as a real application. The changes include creating executable versions of the present LabView programs runnable directly in windows environment, instead of using the National Instrument LabView software for execution. This is modification helps to save some hard drive as well as memory space.

  Provided appropriate hardware display are available, the information available through the applications can be provided for the driver and passengers in various formats.
- As an alternative method, a Palm OS hand held device may also be used as the main processing unit for managing and orchestrating the various hardware components used onboard. New Palm OS applications, however, should be developed for this purpose. Some extra information in Section 10 about Palm OS protocol stacks, indeed, provide support for such an idea.
- Any other minimum system may also be used as the core hardware device.

In either case, the central hardware unit should provide (serial) connection ports for a GPS receiver, a data modem or cellular phone, and a Bluetooth transceiver (in case that it is used as the auxiliary positioning system).

In bus stops the same combination of hardware components are required with this exception that, since it is not mobile there is no need for the wireless Internet connectivity, and also there is no need for Positioning sensors.

The last official hardware component of the system is the GPS receiver that should be installed in a central location to provide the DGPS functionality.

Some other optional pieces of hardware may also be added optionally to the system. These include different sensors to measure the vehicles parameters of interest to the central monitoring unit, as explained in Section 7.

11-4 Conclusions

The present prototype has tried to introduce and implement the major components required to develop an intelligent city transit system. More specifically, it provided:

- Standard architecture of the system,
- Real-time monitoring capability of the system over the Internet, which makes the required information widely accessible,
- Mobile access to some of the transit system information of interest to the end users.

It also introduced a system on top of which the following features can be added:

- Capability of creating and modifying transit system timetable on a real-time basis, or at least in a more interactive way,
- Capability of creating a central unit for the transit system monitoring and managing on a real-time basis.

The system also has introduced two innovative features, are not found to be reported previously.

- Reporting DGPS correction parameter over the Internet, which made using ordinary low price GPS receivers in highly accurate positioning measurements, possible.

- Introducing Bluetooth technology for creating (main or auxiliary) positioning devices.

11-5 References

1- S. Haykin, "*Neural Networks, a comprehensive foundation*", 2nd edition ©1999, Printice Hall.

2- J. A. Freeman, "*Neural Networks: algorithms, and programming techniques*", ©1991, Addison-Wesely.

3- V. Himanen, "*Neural Networks in transport applications*", ©1998, Ashgate.

4- D. Teodrovic, K. Vukadinovic, "*Traffic Control and Transport Planning*", ©1998, Kluwer Academic Publishes. Chapter 4.

5- PMSI, "Neural Networks: Advanced tutorial", ©1991-2002 pmsi, http://www.pmsi.fr/ neurin2a.htm

6- B. Abdulhai, S. G. Ritchie, "*Enhancing the Universality and Transferability of Freeway Incident Detection Using a Bayesian-Based Neural Network*", Journal of Transportation Research - Part C, Emerging Technologies, Vol. 7, pp 261-280, 1999. http://www.civ. utoronto.ca/sect/traeng/people/abdulhai.htm

7- Ch. M. Bishop, "*Neural Network for Pattern Recognition*", ©2000 Oxford University Press.

8- C.H. Chen, "*Fuzzy Logic and Neural Network Handbook*", ©1996 McGraw Hill Text.

9- A. Shalabi, C. Lyon, T. Sayed, "*Real-Time Bus Location, Passenger Information and Scheduling for Public Transportation*", 8th World Congress on Intelligent Transportation Systems, Sydney, Australia, 2001.

10- G. Welsh, G. Bishop, "*An Introduction to Kalman Filter*", Department of Computer Science, University of North Carolina, http://www.cs.unc.edu/ ~welch/ kalman/ kalmanIntro.html

11- M. S. Grewal, "*Kalman Filtering, theory and practice*", © 1993 Prentice Hall.

12- G. Welsh, G. Bishop, "*The Kalman Filter*", Department of Computer Science, University of North Carolina, http://www.cs.unc.edu/ ~welch/ kalman/

Appendixes

Appendix A – List of Figures and Tables

- Figures

- Tables

Appendix B – Softcopy of Source Code

Appendix A – List of Figures and Tables

- Figures

Appendix B – Softcopy of Source Code

The accompanying disk contains the source code of the programs developed in this thesis. The list of the files pertaining to each software component of this prototype can be found in the corresponding sections.

Index