Dublin Bus Tracking Service

Design and implementation of a device independent passenger information system

Eamonn Fallon

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

September 2000

DECLARATION

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: ____________________

Eamonn Fallon

September 2000

PERMISSION TO LEND AND/OR COPY

ACKNOWLEGEMENTS

I would like to thank my supervisor Alexis Donnelly for all his advice, support and patience throughout the year

I would also like to thank my family and friends for all their support throughout the year

SUMMARY

Traditionally public transport has been perceived as the less desirable alternative to the car. The environmental argument of using public transport has not led to a significant increase in the usage of public transport. In order to convince people to leave their car at home, public transport must be at least as, if not more desirable than driving a car. There are a number of ways of achieving this, among them reducing fares, introducing quality bus corridors etc. However the most effective strategy to do this must be to reduce the perceived unreliability of public transport. The most effective way of doing this is through a real-time passenger information system

Passenger information systems are large-scale capital intensive projects, which means only large organisations and governments can implement them. The most expensive component of the system is the automatic vehicle location (AVL) system. This thesis proposes a new type of AVL system based on third generation mobile positioning technology that significantly reduces the cost of implementing a passenger information system.

A best-of-breed architecture for implementing passenger information systems is described. It uses a service chain architecture to aggregate content and services from multiple sources to provide a coherent interface to the end user. The interface can be delivered to multiple devices with different display characteristics. A prototype implementation of this architecture is implemented in Java.

CONTENTS

TABLE OF FIGURES

CHAPTER 1

Introduction

A typical scenario in any city; a group of people waiting at a bus stop, many of them are noticeably impatient, some are pacing. Everyone is staring into the distance anticipating the bus that will eventually turn the corner.

Public transport networks involving buses are inherently unreliable due to traffic. The only way to make them predictable would be to give the buses dedicated roads with no traffic on them. This unreliability makes using public transport frustrating, illustrated by the scenario above. However if you can't increase the reliability, you can at least inform the passengers how unreliable the bus is. If passengers knew exactly where the next bus was at all times, waiting at a bus stop would not be frustrating. This is the problem that this thesis attempts to solve.

This thesis develops a framework for building cost-effective automatic vehicle location (AVL) tracking systems using third generation mobile technology. A generic systems architecture is proposed which incorporates the AVL framework. The architecture proposes a service chain model, to aggregate services and content from multiple providers into one coherent user interface.

A prototype implementation of a passenger information system is developed based on the service chain architecture. The prototype provides device independent access to a simulation of the Dublin bus transport network. The system can be accessed via web browser or WAP-enabled mobile phone.

1.1 Thesis Contribution

This thesis makes a contribution to the state of the art in passenger information systems, by tackling the problem from a different angle. Although none of the technologies described in this thesis are new, the mix of technologies used and architecture proposed are unique. Traditionally passenger information systems have been large-scale capital intensive projects. The most expensive component of the system is the AVL system. By using the M-AVL system proposed by this thesis, public transport operators and other transport companies can significantly reduce the capital costs involved in implementing passenger information systems.

This thesis also proposes a best-of-breed architecture for implementing passenger information systems. This builds upon the research covered by the PROMISE project and also incorporates the M-AVL system.

1.2 Thesis Roadmap

Chapter 2 reviews the state of the art in mobile positioning techniques. The architectures of traditional automatic vehicle location (AVL) systems are presented

Chapter 3 reviews the state of the art in passenger information systems. Two European research projects – PROMISE and Infopolis 2 - are examined, the major findings and recommendations of these projects are presented. Two existing passenger information systems are presented, and contrasted with the proposed AVL systems framework (M-AVL)

Chapter 4 details the architecture of the Dublin bus tracking service. It is explained using a number of different architectures: service chain architecture, functional architecture and communications architecture.

Chapter 5 presents the implementation of the Dublin bus tracking service and highlights the major characteristics.

Chapter 6 evaluates the architecture and implementation of the Dublin Bus Tracking Service and contrasts them with the two passenger information systems in chapter 3. A modified architecture is introduced that increases the scalability of the system.

CHAPTER 2

Positioning & Communications Infrastructure

The unique differentiating factor between mobile access to the Internet and fixed access is the ability to provide services to the mobile user based on their physical location. The mobile network operator can - with varying degrees of accuracy depending on the technology employed - pinpoint the location of a user terminal. This ability - while not widely exploited today – presents a myriad of possible applications and services that were not possible in the domain of the fixed Internet.

Standardisation of positioning solutions is being carried out by the Third Generation Partnership Project (3GPP). The 3GPP standards shall support location service features, to allow new and innovative location based services to be developed. It shall be possible to identify and report in a standard format (e.g. geographical co-ordinates) the current location of the user's terminal and to make the information available to the user, mobile equipment (ME), network operator, service provider, value added service providers and for PLMN[1] internal operations [1]. Mobile equipment – in this case - refers not only to mobile phones, but any type of device/equipment that can access the mobile network.

[1] Public Land Mobile Network

In Europe, consumer and corporate business opportunities are driving the adoption of mobile positioning technologies. In the United States however, mobile positioning is being driven by legal requirements. The Federal Communications Commission (FCC) issued the Enhanced 911 (E-911) mandate on June 12, 1996. It states that by October 1, 2001, mobile network providers must be able to pinpoint the location of the calling telephone to within 125 meters of the actual longitude, latitude location of the user. This location must be accurate at least 67% of the time. The mandate as since been updated (August 24, 2000) to provide certain deadline extensions [2].

The 3GPP is entrusted with the production of globally applicable technical specifications for a 3rd Generation Mobile System based on the evolved GSM core networks and the radio access technologies that the Organisational Partners[2] support [3]. The 3GPP is developing standards for GSM-based technologies - General Packet Radio System (GPRS) & EDGE - and third generation UMTS (WCDMA) networks. The 3GPP standardisation process is on going, and is far from complete. This is due in no small part to the large number of different technologies that the 3GPP must support and the fact that its scope covers the entire range of next generation mobile telephony services – of which positioning technologies are a small part.

Unlike the 3GPP, companies do not have to wait for standards to be finalised. Already, a number of companies have developed position determination technologies. These companies include:

- Cambridge Positioning Systems
- Cell-Loc
- Ericsson
- Grayson Wireless
- SigmaOne
- Qualcomm-owned SnapTrack
- True Position
- U S Wireless

Each of these systems uses a number of similar techniques and support different underlying network standards. However the Ericsson Mobile Location Solution (MLS) looks the most

[2] The 3GPP is a global collaborative initiative, consisting of 6 organisational partners: Japan-based Association of Radio Industries and Businesses (ARIB), China Wireless Telecommunication Standard group (CWTS), European Telecommunications Standards Institute (ETSI), US-based Committee T1, Korean-based Telecommunications Technology Assocation (TTA) and Japan-based Telcommunication Technology Committee (TTC).

promising solution. It provides a standard API that shields the systems developer from the underlying mobile positioning technology. This allows the systems developer create applications that are future proof and standards compliant. When standards are ratified they can be integrated into the MLS system without requiring modifications to the third-party application. The major benefit of using API's such as Ericsson's is that location dependent applications developed for GSM networks will also work on GPRS and UMTS networks. A company called SignalSoft also provides a similar API that integrates with all the vendors mentioned above, while still shielding the developer from the underlying positioning technique.

All the companies mentioned previously, base their solutions on a number of different underlying positioning technologies to track user handsets. These technologies fall into two main categories:

1) Terminal based positioning: This solution requires modifications to be made to the user terminal (user handset). There are 3 main terminal based positioning techniques:
    a. Global Positioning System (GPS)
    b. Network assisted GPS (AGPS)
    c. Enhanced Observed Time Difference (E-OTD)

2) Network based positioning: This requires no modifications to be made to the user terminal. This means that there is 100% market penetration as soon as the system is deployed. There are 3 main network based positioning techniques:
    a. Cell Global Identity & Timing Advance (CGI +TA)
    b. Time Difference of Arrival (TDOA)
    c. Angle of Arrival (AOA)

The following sections discuss and contrast the strengths and weaknesses of each of the positioning techniques mentioned above.

2.1 GPS Positioning

2.1.1 GPS

The Global Positioning System (GPS) is a satellite-based radio-navigation system developed and operated by the U.S. Department of Defence (DOD) [4]. GPS provides the user with their 3-dimensional position, their current velocity and the exact time. The GPS system consists of 24 satellites orbiting the earth in 6 circular orbits. The satellites are arranged, so that at any one time there are 6 satellites within range of a GPS receiver.

The control segment of the GPS system consists of one master control station in Colorado USA, with five ground control stations and three ground antennas located around the world. The monitor stations passively track all the satellites in view. A sample of each of the satellites broadcast signal is continuously taken. These samples are then forwarded on to the master control station, which calculates extremely precise satellite orbits. The orbit calculations are formatted into navigation instructions, which are uploaded to the individual satellites via the ground antennae.

At the same time each of the satellites is continuously broadcasting an exact position and time signal. The GPS receiver receives messages from at least 4 satellites and measures the time delay for each signal. From these values the GPS unit can calculate the user position and velocity. On May 1 2000 [5], the U.S. Government disabled the international degradation (Selective Availability) of the GPS signal. This increases the location accuracy of GPS receivers from approx. 100 metres to less than 20 metres.

2.1.2 Differential GPS

Differential GPS is a technique used to increase the accuracy of GPS receivers to between 1 to 3 metres. The technique involves placing a GPS receiver (a reference receiver) in a known physical location. The reference receiver collects data from all the satellites in view and performs error corrections on the signals checking the *actual* location against the *broadcast* location. These corrections can be either recorded (used for post-processing of signals) or broadcast in real-time via radio. In order to benefit from the broadcast of a DGPS signal, a GPS receiver needs to be equipped with a data port connected to a radio receiver. Furthermore the GPS unit must be within approx.150km of a DGPS signal transmitter. In Ireland we have 3 DGPS signal transmitters, operated as a free service by the Commissioner of Irish Lights[3].

2.2 Terminal Based Positioning Techniques

2.2.1 Network assisted GPS - AGPS

A GPS receiver can be integrated into the circuitry of a mobile phone/device with minimal price impact on the consumer. However there are problems with GPS that make it unsuitable for mobile positioning:

- A high *time-to-first-fix*. A GPS unit can take between 30 seconds and several minutes to initially acquire and track satellites.

- Low sensitivity to signal attenuation, blockage and multipath interference: A GPS unit will not accurate results - if any - in many difficult environments when the GPS signal is weak, e.g. In an urban canyon, inside a building or under dense foliage. Multipath interference occurs when signals get refracted for some reason (e.g. atmospheric layers, clouds, and buildings) and arrive out of phase with the original signal, thus cancelling it [6]. Many GPS units have up to 12 parallel GPS signal receivers to minimise these effects.

- Power inefficient: GPS keep a continuous track on the viewable satellites once a first fix is got. There is also a high overhead - in terms of power - of running up to 12 parallel receivers. For many mobile phone/device users, this power drain may outweigh the benefit of having terminal-based positioning.

- Accuracy: As mentioned before GPS is only accurate to 20 meters. Since a mobile is always connected to the mobile network, there is no need for a separate DGPS receiver. The DGPS signal can be sent via the mobile network.

[3] For more details on the Irish DGPS service see http://www.cil.ie

- Network Assisted GPS (AGPS) attempts to solve these problems. Most AGPS solutions do this by using the distributed architecture[4] in Figure 2.1. Essentially the GPS processing is distributed between the reference receiver, the location server and the GPS enabled user terminal.

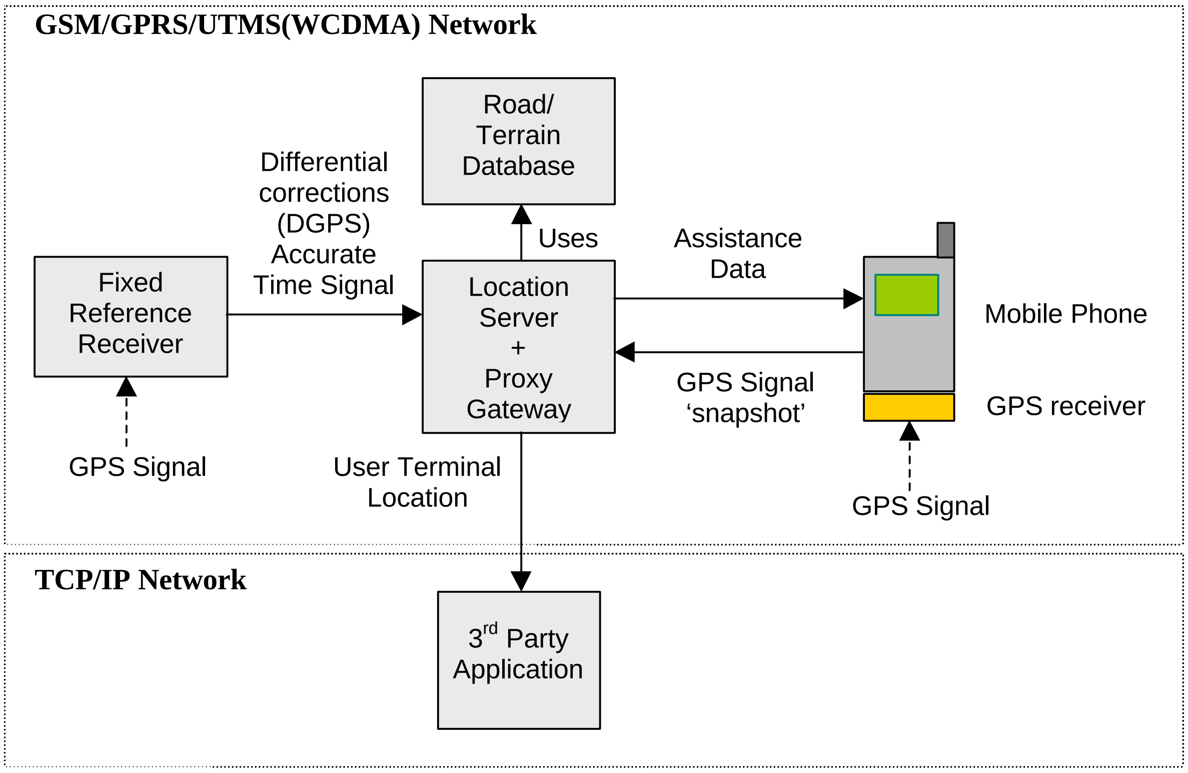


Figure 2.1: Network Assisted GPS Positioning

The architecture shown in Figure 2.1 consists of 3 main components – the Fixed Reference Receiver, the User Terminal and the Location Server:

<u>Fixed Reference Receiver</u>

The fixed reference receiver acts like a DGPS beacon. It is a normal GPS receiver, but because its physical location is known, it can calculate the differential corrections that need to be applied to the signal sent from the user terminal to the location server. These corrections along with an accurate time signal are continuously sent to the Location Server.

[4] This system is a hypothetical architecture based heavily on the SnapTrack network assisted mobile positioning solution. It is adapted to apply to digital networks and the existence of WAP.

User Terminal

The user terminal is a mobile device connected to a digital mobile network. The phone contains a GPS receiver either integrated into its circuitry or as an external plug-in. When a user terminal on the network needs to be located, the Location Server sends it assistance data. The request for a user terminal location will come from either the user terminal itself or a 3rd party application.

The assistance data consists of satellite information on the all the satellites in view in the terminals approximate location and the Doppler offsets of each of the satellites. The Location Sever uses the location of the base station as the approximate location of the user terminal. The user terminal then takes a snapshot of the GPS signal, pre-processes it and returns basic GPS measurements along with statistical values characterising the signal environment. This GPS Signal 'snapshot' is then returned to the Location Server.

The Location Server

This snapshot signal received by the Location Server from the User Terminal is then further processed to remove errors such as multipath interference and atmospheric delays. The DGPS data from the fixed reference receiver is also applied at this point. The mobile terminal's precise latitude, longitude, altitude, speed, and bearing are returned either to the user terminal or the application that requested it.

The Location Server can use a terrain database to further refine the position data. Possible uses of the terrain database would be:

- Snapping location (longitude, latitude and altitude) to the nearest road. This would be used for the bus tracking system implemented for this thesis.
- Snapping altitude to ground level if there are no building at the reported longitude and latitude.

Arguably, in order to maintain modularly in the system, and to support the widest range of 3rd party applications, the terrain database functionality should be a separate module. The module may even reside on an IP network outside the domain of mobile network provider. However these issues will be left to the Companies who implement network assisted GPS solutions.

The architecture described is based on a system developed by Qualcomm-owned SnapTrack Inc. It is developed for the analogue-based AMPS (Advanced Mobile Phone System) mobile network. In this system, a wireless modem must be built into the user terminal in order to communicate with the Location Server. In Europe, we have digital based GSM, and will soon have GPRS and UMTS to choose from. Since these networks are digital based, the handshaking with Location Server can be integrated into the network protocol (possibly using SMS for messaging). Currently AGPS has not yet been standardised by the 3GPP, which means any solutions implemented will be proprietary. The 3GPP plan to agree on a standard by the end of 2000. The accuracy of AGPS - while not quite as good as DGPS – is approx. 10 metres. For the moment while GSM is still the dominant mobile network this is the best (most accurate) mobile positioning solution.

Ericsson is the only company to have implemented an AGPS system for GSM. They have modified their GSM system software to report user location using either AGPS or CGI + TA. If a mobile network operator is to provide AGPS mobile positioning internally and to 3rd party developers they must upgrade the Ericsson software on all their base stations. They must also place reference receivers every 300 kilometres of network. Third party application developers wanting to access user terminal locations through AGPS, must communicate with Ericsson's Mobile Positioning Centre (MPC) server via the propriety Mobile Positioning Protocol (MPP).

2.2.2 Enhanced Observed Time Difference (E-OTD)

E-OTD requires only a software modification to the user terminal. However in order to run the E-OTD algorithms, E-OTD enabled terminals will need additional processing and memory capacity. As shown in Figure 2.2 E-OTD consists of the following procedures:

- The user terminal measures control signals from at least 3 surrounding Base Transceiver Stations (BTS).
- The user terminal measures the observed time difference (OTD) between pairs of incoming control signals. This data is put in a message and sent to a central server (called the Mobile Location Centre or MLC).
- When the MLC receives the message, it contacts the relevant location measurement unit (LMU) and requests the OTDs of the control signal from all the BTSs monitored by the LMU. An LMU is essentially a receive-only GSM antennae and must be deployed for every 4 BTSs.

- The MLC knows the physical location of the BTSs and it’s the physical location of the LMU. Using the calculating the difference in the OTDs of LMU and the User Terminal, the MLC can calculate the physical location of the user terminal. As in AGPS this location can be either sent back to the user terminal or to a 3rd party application.

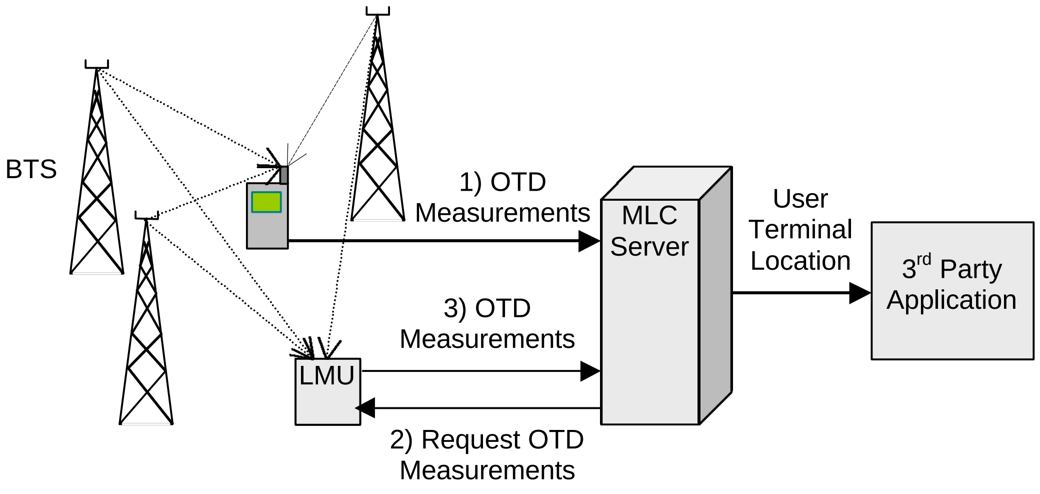


Figure 2.2: Enhanced Observed Time Difference Positioning

The accuracy of this system in GSM networks is between 60 metres in rural areas and 200 metres in bad urban areas (weak signal reception due to blockage and interference). However, it is a mobile positioning technology with huge potential for 3G networks. Cambridge Positioning Systems have managed to position the user terminal with an accuracy of 5 metres using E-OTD in a UMTS network.

2.3 Network Based Positioning Techniques

Network based positioning techniques, require no changes to be made to the user terminal in order to find its location. Therefore the major benefit of network-based positioning systems is backward compatibility. It may take up to five years to get everyone to upgrade his or her phone. With network based positioning, all modifications are made to the network, with no impact on the end user. However this is also a major disadvantage – the user has no control over when their terminals are being positioned and by whom. With terminal based positioning, the user will always have the option of switching it off.

2.3.1 Cell Global Identity & Timing Advance (CGI +TA)

The CGI + TA technique works by measuring two existing parameters in the GSM system. The Cell Global Identifier (CGI) is a unique identifier for each cell sector in the network. The Timing Advance (TA) parameter is the based on the access delay between the beginning of a time slot and the arrival of bursts from the mobile terminal. Since the access delay is proportional to the distance from the Base Transceiver Station (BTS) an estimate of the user radius about the BTS can be calculated. The TA value can only be calculated in increments of 550 metres. By combining the CGI and TA parameters a user terminal position can be estimated as an arc 550 metres wide within a particular cell sector.

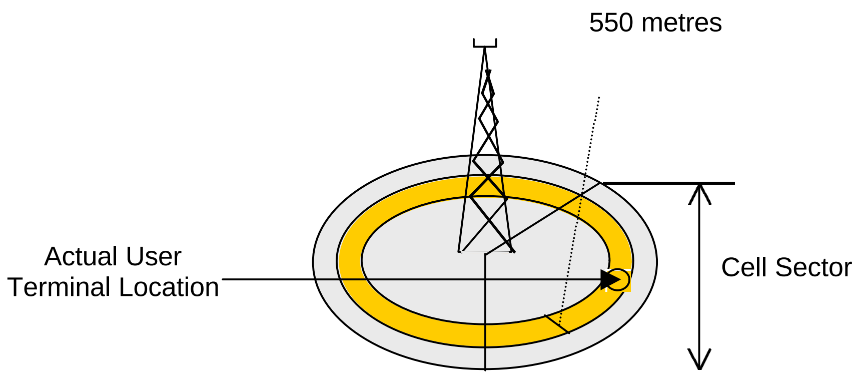


Figure 2.3: Cell Global Identity & Timing Advance Positioning

As can be seen in figure 2.3 the phone can be located to an arc 550 metres wide. The accuracy for this technique depends on three factors:

1) The distance the user terminal is from the BTS: When the user terminal the radius of the arc is smaller, thus more accurate estimate is given
2) The radius of the cell
3) The size of the sector

In a typical GSM system cell radii vary from 100 metres to 30 km. This technique is the least accurate of all the positioning techniques, but requires no modifications to the systems hardware or software.

2.3.2 Time Difference of Arrival (TDOA)

The Time Difference of Arrival (TDOA) technique firstly developed by Turin in 1972 [7] is essentially the same technique as E-OTD except it is done in the opposite direction. That is, the observed time difference (OTD) calculations are done at least 4 different base stations instead of at the user terminal. Each LMU is equipped with a GPS receiver in order to receive the time signal, which is broadcast continuously by the GPS satellites. Alternative timing references such as rubidium oscillators could be used, clock drift will arise but can be solved using a distributed time synchronisation algorithm [8]. Each LMU measures the Uplink Time of Arrival (UL-TOA) of a continuous stream of data from the user terminal. This value along with a precise timestamp is sent to the MLC server. The MLC calculates the time difference of arrival between pairs of UL-TOA values, taking into account the timestamps of each of the UL-TOA values. Since the geographical location of each Base Transceiver Station is known, an estimate of the location of user terminal can be calculated.

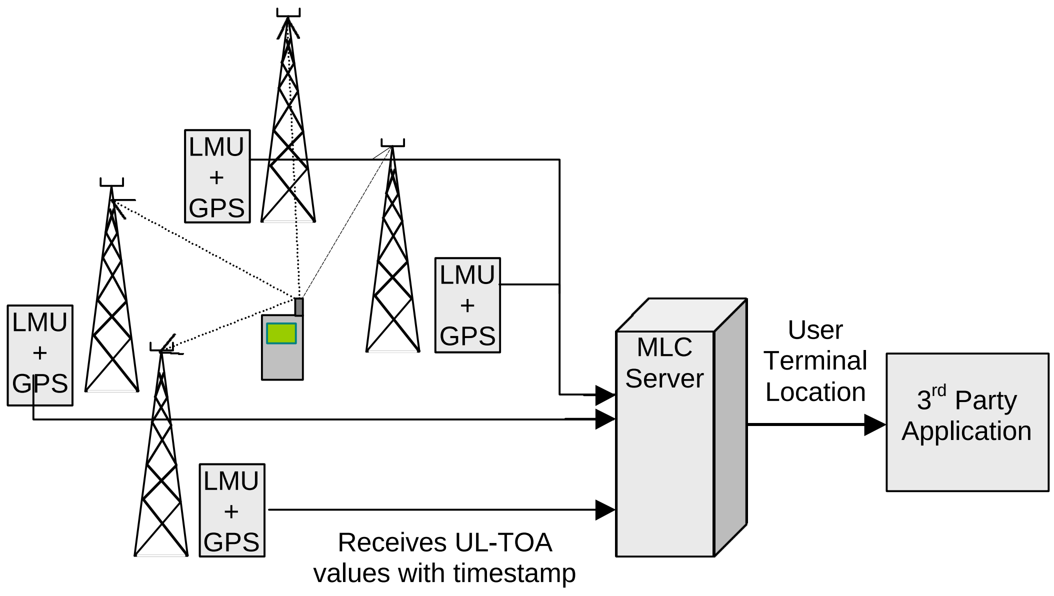


Figure 2.4: Time Difference of Arrival Positioning

The TDOA technique is quite costly from the network provider's point of view, since an LMU + GPS receiver must be installed in practically every base transceiver station. Each of the LMUs must also be connected via a landline to the MLC server. The system will not be able to give an estimate – or at least give an inaccurate estimate - if the user terminal cannot be tracked by at least four individual base transceiver stations. However for a network based solution the accuracy is relatively quite good – approx. 50 metres in rural location and 150 metres in bad urban areas. In the U.S., companies such as True Position, SigmaOne and Grayson Wireless have developed TDOA systems. They support the analogue AMPS network and the various digital based cellular networks: GSM 1900, CDMA IS-95, and TDMA IS-136.

2.3.3 Angle of Arrival (AOA)

The Angle of Arrival positioning technique requires only two base transceiver stations with antennae arrays installed to position a user terminal. An antennae array is a arrangement of antennae in a precise, fixed pattern. Given a known operating frequency, and known antenna spacing, by measuring the phase or phase difference on a number of antennas, the angle of arrival of a plane wave can be deduced [9]. By measuring the angle of arrival at a two or more BTSs, the intersection point of the bearing vectors can be calculated. This intersection is the estimated location of the user terminal.

AOA is very sensitive to multipath interference and accuracy reduces the further the user terminal from the BTS. Typically AOA is used in conjunction with the TDOA technique to improve accuracy, but due to its sensitivity to multipath interference it is not robust enough to use on its own. SigmaOne have reported an accuracy of 125 metres 90% of the time using TDOA and AOA techniques.

2.4 Automatic Vehicle Location (AVL) Systems

AVL or Fleet Management systems – of which bus tracking is a part – traditionally solve the problem of real-time vehicle tracking using a variety of techniques involving proprietary radio networks. In general the systems consist of two components:

- Measurement of the real-time location of the vehicle
- Relay of the position data to a central source

The actual real-time location of the vehicle is measured using one of the following techniques:

- Signpost and odometer
- Global Positioning System (GPS)
- Radio navigation/location
- Dead-reckoning

This location data is stored on board the vehicle for periods ranging from seconds to minutes. The amount of time the position information is stored on-board can depend on external factors, such as being polled or passing a specific location.

Once the real-time position is measured, the location data must be relayed back to a central point. The two most common methods of relaying are polling and exception reporting. With polling, the central server polls each of the vehicles in turn asking for its location. Polling is done over a proprietary radio network, that is, a portion of the available radio spectrum is reserved exclusively for tracking the network of vehicles in the system. This solution requires each of the vehicles to have a radio modem installed.
Polling is usually a continuous process – all the vehicles in the network are polled starting at the first vehicle, ending at the last and then beginning at the first again. The time taken to poll

the vehicles is proportional to the number of vehicles in the network, and the number of parallel radio channels available to poll on.

Since many cities have limited radio spectrum available, and long polling times add unacceptable errors to the system, many AVL systems use exception reporting as a relay technique. With exception reporting, the bus only reports its location at specified points or when bus is running off schedule beyond a specified tolerance (i.e. 1 min early, 15 minutes late). If the central server receives no report, it assumes the vehicle is on schedule. The reports are sent via radio modem to a central server. This technique reduces radio traffic, however it has disadvantages:

- The vehicle must have knowledge of its schedule, which can be complicated by changing traffic flows. This requires extra hardware/software on vehicle, which is not only more expensive, but difficult to maintain.
- Transmission errors may result in the central system inferring that the vehicle is on schedule, which in fact it may not.

Out of all the AVL real-time tracking solutions signpost and odometer is the oldest and most widely implemented vehicle tracking technique, in the public transport arena. It is currently being tested in Dublin by Bus Átha Cliath, Dublin’s bus service provider. The next section examines this technique and highlights some of its major flaws and looks at alternative techniques. Although GPS is increasingly becoming more popular in AVL systems, it has already been covered adequately in previous sections, so there is no need to discuss it here again.

2.4.1 Signpost and Odometer

The Signpost and odometer system consists of a series of beacons placed along the route of a vehicle. Each of the beacons emits a low powered radio signal with a unique id. The passing vehicle has a special receiver build in, which can decode the id of the beacon. When the vehicle has to report its location (either by polling or exception reporting) the location of the last visited beacon is sent along with the odometer reading. The odometer reading is the distance travelled since the last beacon. Signpost and odometer systems have been in use in various cities around the world since the early eighties. They are now falling out of favour due to a number of reasons:

- There is a high maintenance cost in keeping the signpost (beacon) network operational.

- If the vehicle strays off the predefined route, it cannot be tracked.
- There are many points of failure: the odometer, the beacon and the radio modem.

2.4.2 Dead Reckoning

The dead reckoning positioning technique stems from the dead reckoning navigation technique developed in Europe at the end of the 15th century. Columbus and many of his sailing peers used it to navigate their sailing ships. Columbus managed to navigate to North America using the technique. The technique works by having a fixed known location, and extrapolates a future location at a given time, using the current speed and direction. Dead reckoning in AVL systems works by positioning a series of beacons around the city. The last known location is derived from the time difference between passing through consecutive beacons. The actual location is calculated by using the time passed since the last known location was found. This system is only ever implemented as a backup system. It is usually used in conjunction with GPS tracking and takes over only when the GPS signal is weak (e.g. inside tunnel or in an urban canyon).

2.4.3 Radio navigation/location

This system is very similar to the E-OTD mobile positioning technique. Strategically placed towers emit 900Mhz signals across the city. The vehicles can infer their location using the TDOA technique.

2.5 Manual Positioning Techniques

Manual positioning, while often overlooked, can yield high accuracy and requires minimal technical infrastructure. Even though we may not realise that we are doing it, this is the method we use to position someone on the other end of a mobile phone conversation. In this case, manual positioning is essentially a positioning technique, using verbal handshaking. Typically the caller or callee, will start the handshaking sequence by asking, “Where are you”. A sequence of questions will follow, until both parties have knowledge of the others location. Obviously both parties must have knowledge of the terrain for this handshaking to be successful. Voice positioning is also the dominant form of positioning used by taxis over a *push-to-talk* radio network.

Manual positioning can be assisted, through the use of a graphical or textual computer-generated interface. This to some extent compensates for a lack of terrain knowledge by the user. The fact that tourists can - without any previous terrain knowledge - successfully navigate around a new city using a street map, reinforces this. Using a graphical map, users can find their location by comparing place names and street names on the map with their surrounding environment. In an urban environment this technique is usually quite accurate. If the location pinpointed by the user can be inputted into a computer system – possibly using a WAP-enabled phone – accuracy’s can be achieved that are better than some the network based mobile positioning techniques.

The WML user interface implemented for this thesis uses manual positioning to pinpoint which bus stop the user is at. Given that a user knows which route they are travelling on, the system presents them with a list of possible bus stops they can be at. By examining their environment (street names and landmarks), the user can accurately choose the exact bus stop they are at. Errors are only introduced, if the user chooses an incorrect bus stop. Therefore assuming the user is correct, the accuracy of the location is only dependent on the method use to geocode the bus stop in the initial design of the system. Geocoding will be covered in more detail in chapter 5.

2.6 Summary of Positioning Techniques

This chapter discussed the different techniques that can be used in a mobile network, to pinpoint the location of a user terminal (phone). The techniques were classified as either network based positioning, each having their advantages and disadvantages. Network Assisted GPS provides the best accuracy of the all the mobile positioning techniques. CGI + TA requires no modifications to be made to the network or terminal, but provides poor accuracy. The best network based positioning techniques provide accuracies of about 125 metres while AGPS is accurate to about 10 metres. DGPS is better still, providing at least three-metre accuracy, but it requires strong signal strength and has a long time-to-first-fix.

Traditional AVL positioning techniques were also looked at. They all require heavy capital investment in communications infrastructure and on-vehicle computer systems. This thesis argues that with mobile positioning techniques being as good as - and in some cases better than – AVL positioning systems, there is no need for the transport company to invest in the propriety systems and communications infrastructure. Mobile positioning, provides a low cost means of both tracking real-time position and relaying data to a central source for no up-front capital investment. Leaving the infrastructure problem to a third party achieves economies of scale, and lets the transport company focus on maintaining the transport infrastructure. Since mobile positioning technology is only going to get better with the advent of GPRS, EDGS and WCDMA, it is a natural choice for a 21st century real-time passenger information system.

CHAPTER 3

Passenger Information Systems

Increasingly in urban centres, passengers are confronted with multiple modes of transport. Passengers can take trains, buses, taxis, trams etc. Recently - through the Telematics Application Programme (TAP) Transport Sector initiative - the EU has heavily funded a new area of research called intermodality. Intermodality involves the complementary use of multiple types of transport to get assets from one point to another efficiently. Intermodality was first made popular by the logistics industry, and is now being applied to public transport systems to make them more effective. The goal of intermodality in a public transport context is to get more passengers to their destinations faster without increasing the number of transport vehicles.

Traditionally Passenger Information Systems have been developed from the ground up as monolithic systems, the most capital-intensive part of the system being the communications and tracking infrastructure, i.e. the Automatic Vehicle Location (AVL) system. The choice of AVL system heavily influenced the design of the software systems built on top of it. Since there is no standard interfaces for integrating with AVL systems, passenger information systems developed upon one AVL system need extensive rewriting when applied to another.

Intermodal passenger information systems, by their nature, require an information feed from many different transport providers. These providers will range in size from small taxi companies to large public transport companies. Unless the AVL systems are homogenised and require relatively little capital investment, many transport companies will not implement them which results the goal of intermodality not being realised. This thesis proposes an AVL systems framework that is designed to maximise the transport network providers budget and to reduce vendor tie-in by using standardised communications and systems technologies. It allows the underlying AVL system to be de-coupled from the information systems upon it, by exploiting the standardisation work being done for 3G mobile networks. This AVL systems framework is described in section 3.4 and also in chapter 4.

While there is not much standardisation work being done in the area of AVL systems for public transport networks, passenger information systems is an area that is being heavily researched. The state of the art in the area of intermodal passenger information systems is covered by two TAP funded projects: Infopolis 2 and PROMISE. Infopolis 2 covers the ergonomics of passenger information systems and suggests guidelines for system implementers. The PROMISE (Personal Mobile Traveller and Traffic Information Service) project proposes a service-chain-based architecture for a real-time, position-dependent, and multimodal[5] traveller and traffic information service.

The rest of this chapter will summarise the most important finding of the Infopolis 2 and PROMISE projects. An AVL system framework is proposed and contrasted against three existing passenger information systems in use today around Europe.

[5] A multimodal passenger information system provides information about the different modes of transport but does not provide facilities for the complementary use of the different modes to get from an origin to a destination (i.e. an intermodal passenger information system).

3.1 Telematics Application Programme

Telematics is a branch of engineering research that involves the complementary use of both computer and communications technology to provide information. The Telematics Application Programme[6] (TAP) is an applied research programme funded under the EU forth framework programme. It is a user driven research programme, focusing on the societal applications of information and/or communication technologies. Its activities are closely related to the Information Technologies programme and the ACTS programme. Together, the three programmes represent 28% of the total EU research budget. The two research projects that this thesis focuses on (Infopolis 2 and PROMISE) are funded by sector two of the Telematics Application Programme – Telematics for Transport.

3.2 Infopolis 2

Infopolis 2 (1998-2000)[7] was a research project funded by the EU Telematics Application Programme (Transport Sector) which finished in July 2000. It built upon the work covered in the earlier – year long – Infopolis project. The original Infopolis project focused primarily on the effectiveness of Human Computer Interfaces (HCI) to passenger information systems. Infopolis 2 extended the scope of the original Infopolis project to include intermodality (the use of different types of transport for the same trip).

The primary goal of Infopolis 2 was to improve user access to electronic intermodal traveller information. In order to achieve this goal, three sub-goals were identified:

1 The compilation of a database of existing passenger information systems
2 The development of concrete guidelines for system designers
3 The analysis of passenger needs i.e. what information do passengers want, when do they want it and how do they want it presented.

Many of the guidelines proposed by Infopolis 2 were implemented by the Helsinki 423 system, which is described in section 3.5.

[6] For more details see *http://158.169.50.95:10080/telematics/transp/transport.html*
[7] For more details see http://www.ul.ie/~infopolis/index.html

3.2.1 Passenger Information System Families

The Infopolis 2 project examined over 600 telematic based passenger information systems in use in Europe. It categorises these passenger information systems into seven system families:

1. Public interactive terminals
2. Electronic bus stop display
3. On-board information
4. Enquiry office terminal
5. At home/office information
6. Hand-held terminal
7. Website

In general these systems build on information that is already available in existing media (timetables and network maps), to provide more reliable (near real-time) data. Each system family has its own particular presentation characteristics and input mechanisms (if any). It is important to note at this stage that neither Infopolis 2 nor PROMISE look at the underlying mechanism or infrastructure for the provision of the realtime data, they just assume that it can be provided.

<u>Public interactive terminals</u>
Public Interactive Terminals (PIT) are passenger information systems that mainly provide information to travellers before their trip to enable them to make informed decisions about modes (type of transport to choose), routes and departure times. Terminals also help travellers during their journey at connections. PITs are very often located near public transport network facilities, in stations or at stops.

<u>Electronic bus stop display</u>
Real time at-stop information is probably one of the best ways to meet user expectations. At-stop displays usually display waiting times. Also, the location of the arriving vehicle can be shown. The knowledge of waiting time greatly improves the conditions of a journey in two main ways:

- By removing uncertainty (When will the bus arrive? Has the bus already passed?)
- By minimising waiting time (e.g. passenger can do last minute shopping).

On-board information

On-board (in-vehicle) displays have two main roles:

- To enable the passenger to get information on his/her bearing when the vehicle is moving. Systems can give information, for example, on the destination of the vehicle, the next stop and connections.
- To entertain and to inform about the transport network and city activities.

Enquiry office terminal

Enquiry office terminals differ from other systems studied by the Infopolis 2 project because the users are not passengers but information personnel from transport companies. Their main purpose is to help personnel to answer user requests. The systems have to be flexible, dialogues have to be compact and answers have to be received quickly. The information usually includes trip planning, timetables, fares and tourist information.

At home/office information

This area covers systems such the French Minitel system. The information is presented similar to a website. However the quality of the graphics is much lower on the Minitel system.

Hand-held terminal

Hand-held systems studied included intelligent mobile phones (ie Nokia Communicator), SMS based information systems and pager based information. Most of the research data for this area was taken from the PROMISE project, which focuses on passenger information for the mobile user. The PROMISE research was completed before WAP was standardised, but the same principles apply to WAP based mobile phones and indeed any personal handheld mobile device that has some degree of interactivity and can send/receive data over a wireless network.

Website

This area is has the largest growth out of all the information system families. It is largely confined to route planning activities, advertising, and fare information.

Chapter four describes a device independent architecture that allows data be delivered to each of these system families.

3.3 The PROMISE Project

The PROMISE (Personal Mobile Traveller and Traffic Information Service) project is primarily concerned with the development of a personalised traveller and traffic information service that can be viewed on a personal mobile terminal. Its objective is to provide travellers with a range of easy-to-use multimodal traveller and traffic information services. The project is heavily user-focused, in the hope that it will produce commercially viable services that consumers will want and will pay for. While the PROMISE project focuses on both traffic information systems and passenger/public transport information systems only the research results relevant to passenger/public transport information systems presented here.

The project, which ran from January 1996 to February 1999, was partially funded by the EU TAP Programme. The PROMISE Consortium consists of a diverse range of telecommunications providers, electronic equipment manufacturers, car manufacturers and public authorities. The key project participants were Nokia (Project leader), Volvo Technological Development, BT Laboratories, Rijkswaterstaat, IBM Deutschland, Renault Research, Eutelis and BMW.

The project was conducted in a series of phases; each phase produced one or more project deliverable documents. The first phase involved extensive user-needs analysis. The user-needs analysis was conducted in 6 EU countries, its goal was to identify services that would:

1) Satisfy user needs
2) Be commercially attractive to service providers, information providers and infrastructure providers.

Phase 2 built upon the user needs analysis research results to develop what is called the *PROMISE service concept*. This consists of two core services aimed at the mobile traveller:

1) Trip planning service: This covers pre-trip and on-trip planning and the corresponding guidance support to make this possible.
2) Traveller and traffic related information services: This covers miscellaneous information such as yellow pages data, places of interest, weather data, parking data etc.

A formal service specification is described in detail in project deliverable D3.1 [1]. Using the service specification, the PROMISE consortium developed a generic system architecture [2] that supports the delivery of traveller and traffic information services to portable and in-vehicle terminals. The systems architecture described in chapter 4, incorporates many of the aspects of both the PROMISE service concept and the PROMISE generic system architecture.

3.3.1 User needs analysis

The main drive behind the definition of the PROMISE services was to take the user as the starting point. The idea was that if all the attention were focused on the users needs, the resultant services would match those needs and deliver usable products. Usability is the extent to which a product can be used to achieve specific goals with effectiveness, efficiency and satisfaction in a specified context of use[8]. The objective of the PROMISE user needs analysis was to identify highly user friendly, usable solutions and to define a clear strategy for the provision of multimodal traffic and travel information for the mobile user.

The first step in the analysis was to include *actual* and *representative* potential users of the system. This was done through focus groups in each of the six countries. Then, by using interviews, questionnaires and scenario analyses, the users' needs for information in real traffic and travel situations were investigated. Although many of the six EU countries involved in the project (Britain, Holland, France, Finland, Germany and Sweden) approached the user needs analyses in different ways, the results from each country are quite similar. The most important conclusions of the studies are summarised below:

- Travellers emphasised the need for a wide and diverse range of information/services. They want more than just pre-trip and on-trip planning and guidance information. They want weather data, parking availability and points of interest.

- There is a need for dynamic services, not only those that provide access to real-time information, but also those that warn of any changes and/or react to those changes.

- Travellers would like personalised information. Services should be customisable, based on a user profile.

- People wanted to access the mobile information services through a small personal device similar to a mobile phone or PDA. This research was conducted in 1996, since then, a whole infrastructure with corresponding protocols has been developed to enable the

[8] The formal ISO definition of usability

mobile user access web-based information services. A WAP enabled mobile phone operating over a GSM network matches very closely the requirements of the conceptual PROMISE terminal specified in the user analyses studies.

- People were interested in accessing the PROMISE information services at home, at work (on a PC) and through a fixed terminal on the public transport vehicle.

- People were prepared to buy a dedicated (propriety) PROMISE user terminal and pay for the service. This point highlights how desirable this sort of information service is if people are willing to pay for a dedicated terminal. With recent innovations in mobile technology i.e. WAP, users will not have to buy a dedicated terminal. Also with the advent of GPRS, EDGE and WCDMA, users will pay less for accessing the service because they will only pay for the data transmitted.

3.3.2 Service Definition

Using the results from the user needs analyses, the PROMISE consortium identified services that would satisfy user needs and be commercially attractive to service providers. The basic PROMISE service concept is to provide two core services:

1) Pre-trip and on-trip planning
2) Travel related information services

Other peripheral services that users wanted, but did not fall into the two core categories were grouped into

1) Additional services: emergency, payment, advertising, bookings and mobility agenda services.
2) Horizontal applications: fax services, email, web browsing, SMS, telnet, FTP. With the advent of WAP, these services are already available.

The PROMISE service concept is best viewed according to how the end user will use the services. This is shown in Figure 3.1

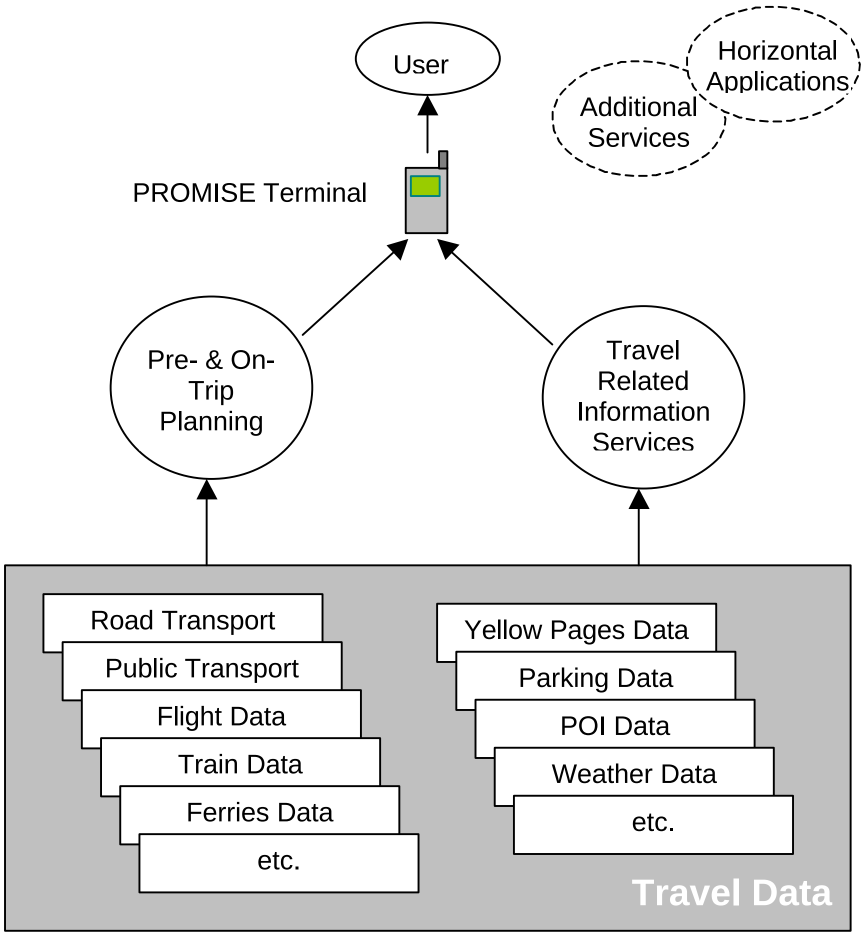


Figure 3.1: The PROMISE service concept

3.3.3 Core Service 1: Trip Planning

The Trip-Planning Service is closely modelled on the process the traveller undergoes when he/she actually makes a journey (Fig 3.2). When a user makes a trip, they do so for a *non-travel* reason such as a business meeting, shopping, cinema etc. Therefore the starting point for a trip is when the user selects a *Service Location* (SLOC) that will fulfil his/her non-travel criteria. The SLOC can be inputted in a number of ways

- The SLOC could be an address in a personal address book
- The SLOC could be an address that the user knows and manually enters.
- The user may not know the address and select the destination from a Points-of-Interest (POI) list. The POI list contains locations such as tourist attractions,

entertainment complexes and shopping centres. Similarly the user can find a business address from a yellow pages facility.

Once the user has selected an SLOC the pre-trip planning process begins. The user must enter an origin (this can be either the current position or another specified location). A pre-trip plan is created. Once the user starts the trip this plan becomes the on-trip plan. Throughout the trip, the on-trip plan gives feedback to the user. The feedback is dependent on the actual position of the user. The user's position can be captured in a number of different ways:

1. GPS positioning: The PROMISE terminal may have a GPS unit installed in it. Alternatively one of the mobile positioning techniques mentioned in chapter 2 could be used to track the location of the phone.

2. User positioning: The user can manually enter their position.

3. Timetable positioning: The system can guess where the user is based on time. This assumes that the vehicle is closely following the times in the timetable. This would be only suitable for trams, trains etc. (vehicles with high probability of conforming to the times in the timetable)

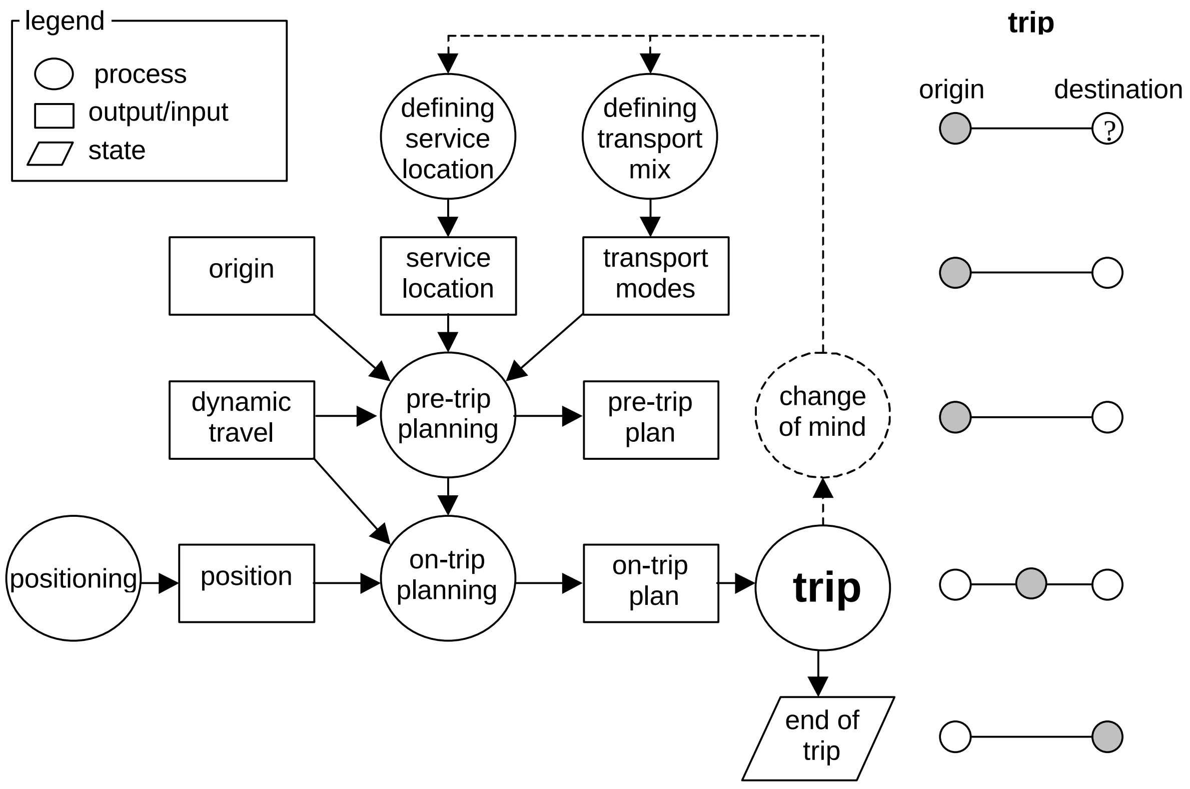


Figure 3.2: The PROMISE trip planning process

The Trip Planning service provides the user with dynamic multimodal travel planning. The system supports the user both before the trip is made and during the trip. The system facilitates the selection of origin and destination points using a number of different mechanisms. The user can select the mode of transport (bus, train, car, walk). The system provides guidance to the user through the use of real-time travel and traffic information

3.3.4 Core Service 2: Information Services

Information services cover both static and dynamic information for users of public transport and cars. Many of these services are used to support the trip-planning process. The most important information services relating to public transport are detailed below:

<u>Public Transport Information</u>

This information service provides the user with real-time and non-real-time public transport information that supports both pre-trip and on-trip planning. This information comes directly from individual transport operators. The service provides the following information:

- Taxi Services: this service gives information based on the location of the user. It provides contact info, fares, taxi data (no of seats, disabled facilities). The user can book the nearest taxi or search for a special taxi matching his/her needs. The user can also book a taxi in advance.

- Stations and Stops information: Gives locations of stations and stops both the street address and displayed on a map. Provides the distance and time between stations/stops. Provides guidance to the users travelling between stations/stops

- General Public Transport (PT) information: Provides timetable and route information. Provides a description of tariffs used on each route. Provides locations where tickets can be purchased, and locations of lost-property offices etc.

- Real-time event information: This is possibly the most useful PT information, it provides the user with information about delays, cancellations, extra departures, or any other sudden changes in the public transport system.

Yellow Pages Information

The PROMISE yellow pages information service provides enhanced access to the business listings in the standard yellow pages directory. The user can query the yellow pages database based on where the user currently is through the use of a mobile positioning technique. The system can show the position of the business on a graphical map. The route planning service can then be used to plan a trip to this location.

Typical user queries would be “find me an Italian restaurant with 5 minutes (walking) of where I am where the price of the meal will not be greater than 15 pounds” or “the nearest ATM”

Point-of-Interest Information

The Point of Interest (POI) information service provides the user with up-to-date information about his/her area of interest. It provides access to a wide range of entertainment/event related information such as current events, concerts, exhibitions, museums, films etc. It also can act in tandem with other PROMISE services such as the Trip Planning/Route Guidance service and the Reservation Service.

The POI service differs to the Yellow Pages service in terms of focus. Instead of searching for the nearest cinema, you would search for the nearest cinema showing the latest film. Instead of searching for the nearest bar, you would search for the nearest bar with a salsa band currently playing. In short the POI takes into account real-time event data, which the yellow page service does not. The POI service also limits the locations the user can search for by focusing on locations that have entertainment/interest value (i.e. taxis ranks, banks, petrol stations etc. would not be listed).

The user can enter what he/she is interested in, in three different ways:

1) Event Category: Allows the user to enter a search for a location by browsing event lists. The user can choose from a range of different event categories: film, concert, art exhibition, business show etc.
2) POI category: Allows the user to enter a search based on actual physical locations that have an entertainment focus, i.e. cinema, theatre, concert hall, nightclub, leisure park etc

3) Object name: Allows the user to obtain the latest programme listings from a previously known location, i.e. the films currently being shown in the local cinema.

Additionally the user can add constraints to focus the results of the search:

1) Attributes: This returns locations with specified attributes e.g. support for parents with young children (changing facilities, special children's activities etc.)
2) Nearest: This returns the nearest location that matches the search criteria. This is calculated relative to the users current location or another specified location.
3) Radius: Allows the user to specify a maximum distance from their current location or specified location.
4) Area: The user can specify a town, city or country. For example the user could search for any business exhibitions in the next two months happening in Dublin.

The results of the search are returned as a list of locations and can be sorted alphabetically or by distance. The user can get detailed information about each location such as admission prices, opening times, telephone number, address, location on a map etc. The user can also automatically phone the location (they do not have to manually type in the telephone number).

Using other PROMISE services the user can also:

- Plan a trip to the location and receive on-route guidance (using Multimodal Trip Planning service).
- Book and pay for admission using the Reservation Service

<u>Reservation Service</u>

This service allows the user reserve entities such as a public transport ticket, a taxi, a car parking space etc. The PROMISE service specification only allows for reservation (i.e. the reserved ticket must be purchased at the train station, car park etc.). Billing and electronic payment systems are beyond the scope of the PROMISE project.

The ticket reservation service can be used as either a stand-alone service or embedded in another service (e.g. the multimodal trip planning service). When used, as a standalone service the user must enter details such as the route, the origin, the destination, seat number etc. However when embedded in the trip planning service, the trip plan is inputted automatically.

When the details of the reservation have been entered, the user presses a *Make Reservation* button and will receive either a confirmation of success or failure. If the reservation is successful, the user will be quoted a unique reservation number. If the reservation is not successful the user will be given an error message stating why the reservation failed (e.g. not enough seats available, or could not contact reservation system etc.).

3.3.5 Generic PROMISE System Architecture

The mission statement used in the design of the PROMISE system architecture is:

To develop a Personal Travel and Traffic Information Service (PROMISE) which uses portable and in-vehicle terminals in order to provide users with quick and easy access to a number of useful services using mobile communication. These services are packaged and offered by a Value Added Service Provider who has reached agreements with a number of Content Provider Centres for accessing their value added content.

There are five actors in the PROMISE system:

- Travellers: There are different types of travellers (e.g. business travellers, commuters, tourists etc.)
- Value Added Service Providers (VASPs): Establish services in order to make profit
- Content Providers: Own value added content which is used in service provision
- Terminal Manufacturers: Sell user terminals to travellers
- Network Operators: Operate the communications network used to transport data to terminals

The PROMISE system architecture defines interfaces or high-level protocols for the communication between the different actors in the system. It also identifies the role of each actor in the realisation of the PROMISE service concept. The system is also described from a functional, informational and data communications viewpoint. Figure 3.3 gives an overall view of the PROMISE system architecture.

The PROMISE system architecture is based on an integrated value chain model, which is described in chapter 4.

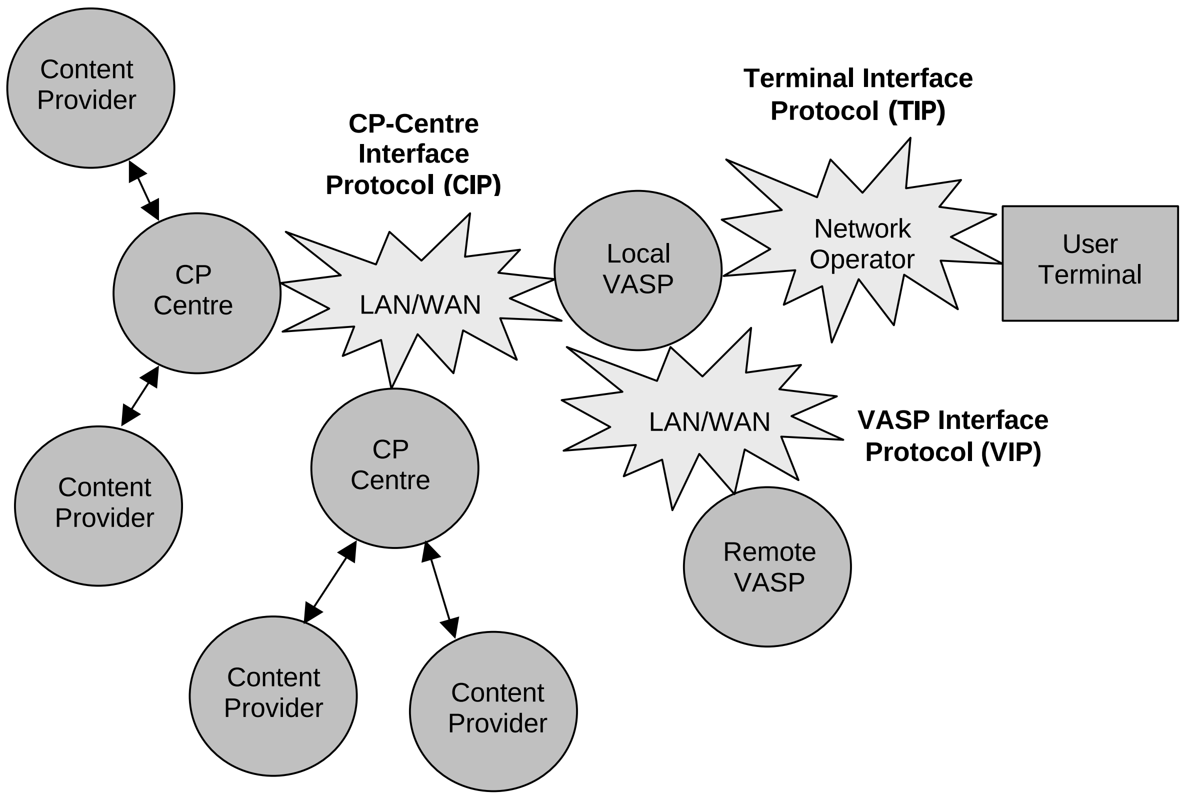


Figure 3.3: PROMISE system overview

Content

This is the raw data stored in the database.

Content Provider (CP)

The owner of one or more databases, which are used as a basis for a content provision service

Content Provider Centre (CPC)

Acts as a central point for aggregating the content from the content providers, in order to provide value added travel and/or traffic information service to the VASP.

Value Added Service Provider (VASP)

This entity packages the data from the different CPCs and delivers it to the end user. Performs user billing and authorisation.

Network Operator

Operates the communications network used for the delivery of information to the user terminals.

User Terminal
The piece of equipment used to access the PROMISE services. Can be portable (eg WAP phone, PDA) or in-vehicle terminal.

User
The traveller who uses the terminal to access the service.

Content Provider Interface Protocol (CIP)
The protocol used to interface between the Content Provider Centre and the Value Added Service Provider.

Terminal Interface Protocol (TIP)
The protocol used to interface between the Value Added Service Provider and the user terminal.

Value Added Service Provider Interface Protocol (VIP)
The protocol used to interface between two or more Value Added Service Providers in order to exchange information between service centres and geographical areas.

The PROMISE architecture defines a three level Reference Model: Level 3, Level 2 and Level 1.

- The Level 3 Reference Model is defined at the highest level of abstraction. It defines the authorities (institutional entities) involved in the system and what responsibilities they undertake in the provision of the service

- Each authority defined in the Level 3 reference model has an associated Level 2 reference model. The Level 2 reference model uses a layered model – similar to the OSI reference model – to describe the functionality to be provided by each of the entities identified in the Level 3 reference model.

- The Level 1 reference model defines the functional, information and data communications architecture of the PROMISE system. The model is described on a conceptual level to avoid any implementation dependent issues.

3.3.6 Level 3 Reference Model

The Level 3 Reference Model identifies 3 separate authorities:

<u>User Terminal Manufacturer</u>
The User Terminal Manufacturers function in the model is to provide an attractive user terminal and to sell as many as possible

<u>Value Added Service Provider (VASP)</u>
The function of the VASP is to optimise contracts with CPCs in order to provide attractive services to the users and to be compatible with every terminal on the market

<u>Content Provider Centre (CPC)</u>
The function of the CPC is to supply value-added content to as many VASPs as possible.

The aim of the Level 3 Reference Model is to describe an open architecture that will provide a healthy (competitive) market for the provision of mobile travel information services. Users can choose terminals from many manufacturers. Information can be requested from many different VASPs and VASPs can choose to get data from many different CPCs.

3.3.7 Level 2 Reference Model

The 3 authorities identified in the Level 3 reference model, have a layered (Level 2) reference model associated with them. The layered approach is a common systems design pattern, which is used heavily in the telecom/networking systems (eg TheOSI model, TCP/IP etc.). The layers in this reference model refer more to system concepts rather than to actual system function layers.

User Terminal Layered Reference Model

Layer	Name	Functions
5	User Interface	Menu handling, Graphics and text display
4	Information Decoding	Translation of sent/received data elements
3	Cache Storage of Data	UI performance optimisation, Session data
2	User/Terminal Identification	Storage of user profile and terminal characteristics
1	Protocol Handling Interface	Application Layer Protocol, Secure transmission of data
0	External Devices Interface	Interface to devices such as GPS, provide data transfer capabilities

Value Added Service Provider (VASP) Layered Reference Model

Layer	Name	Functions
8	User Administration	Registration, Account opening
7	Authorisation (System Access)	Password validation (or other authentication mechanism)
6	Mailbox Control	Determination of pending/off-line requests (Storing request, Showing result of polling/regular request)
5	User Profiling	Determination of user requests (Default profile or request determination, Answer request, Accounting associated with request)
4	Analysis of the request	Determine action to be taken on requests (Optimise request, Log request, Integrate information and identify CPC)
3	CPC Profile	Consolidate request and CPC profile (Access CPC profile, Identification of execution/tasks, Logging)
2	Router/Director	Execute individual request, reception/validation of information (Request issuance/optimisation, Determine physical location(s) of data, Integrate different locations)
1	Execution	Local reception of individual database access request (Optimisation, Local interpretation, Logging per access)

0	Database access	Collection of data to answer request

The VASP layered reference model is layered in the way a user request is logically processed and answered by the VASP.

Content Provider Centre (CPC) Layered Reference Model

Layer	Name	Functions
4	VASP Administration	Registration, Account opening, Billing
3	VASP Communication	Packaging of requested information, Handshaking with VASP
2	Conversion according to VASP profile	Access VASP profile, Convert information to the required format
1	Content Integration	Analyse request, Handle multiple requests to Content Provider, Accounting
0	Proprietary API for each Content Provider	Optimisation of dialogue, Logging

3.3.8 Level 1 Reference Model (Functional Architecture)

The functionality of the PROMISE system is delivered by the three main entities identified in the Level 3 Reference Model (CPC, VASP & Terminal). The applications used to provide the functionality run on both central servers and the user terminal. It is desirable to use standard terminal software to access the services so that the services can grow and change. The client – server model (Request/Response) is used to access most services, however client call-back is used to alert the user of special events. The functionality of each of the three entities is summarised below:

<u>Content Provider Center (CPC) functionality</u>

The role of the CPC is to provide value added content, which is produced by integrating content from different content provider databases. For example a CPC could integrate information from different databases containing bus, train and taxi data. The CPC would provide this multimodal transport information to a VASP through a single access point. The CPC communicates with the VASP using the CPC-VASP Interface Protocol (CIP). The basic functions of a CPC can be summarised as follows:

- Acquire information from different content provider systems
- Create value added content by integrating information from different content provider systems.
- Provide multilingual content
- Deliver value added content to the VASPs via the CIP interface. The content delivered to each VASP is dependent on the VASPs profile i.e The format of the data delivered will vary between VASPs
- Maintain databases containing VASP billing data and VASP profiles

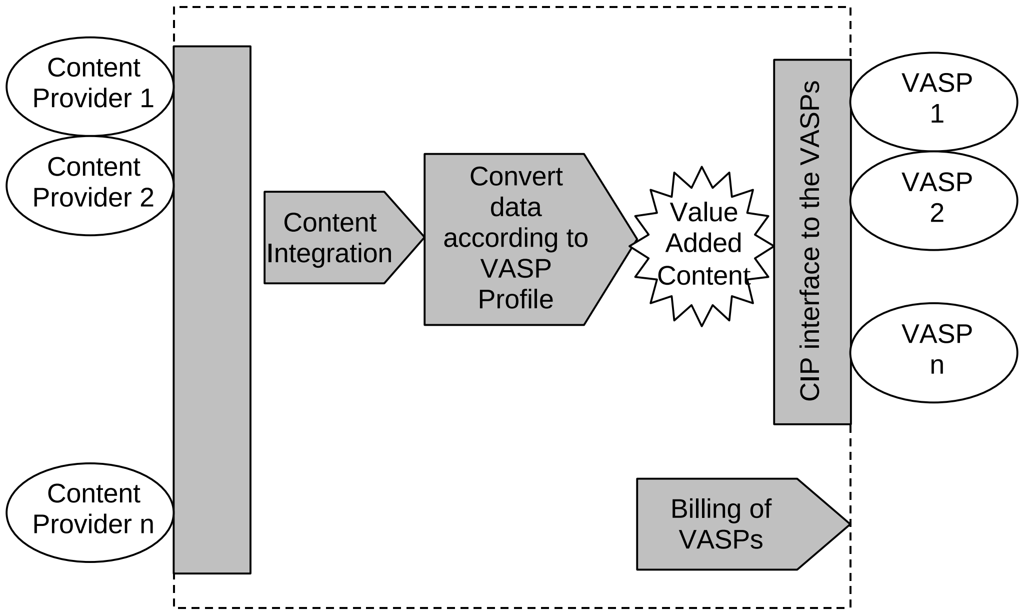


Figure 3.4: CPC functionality diagram

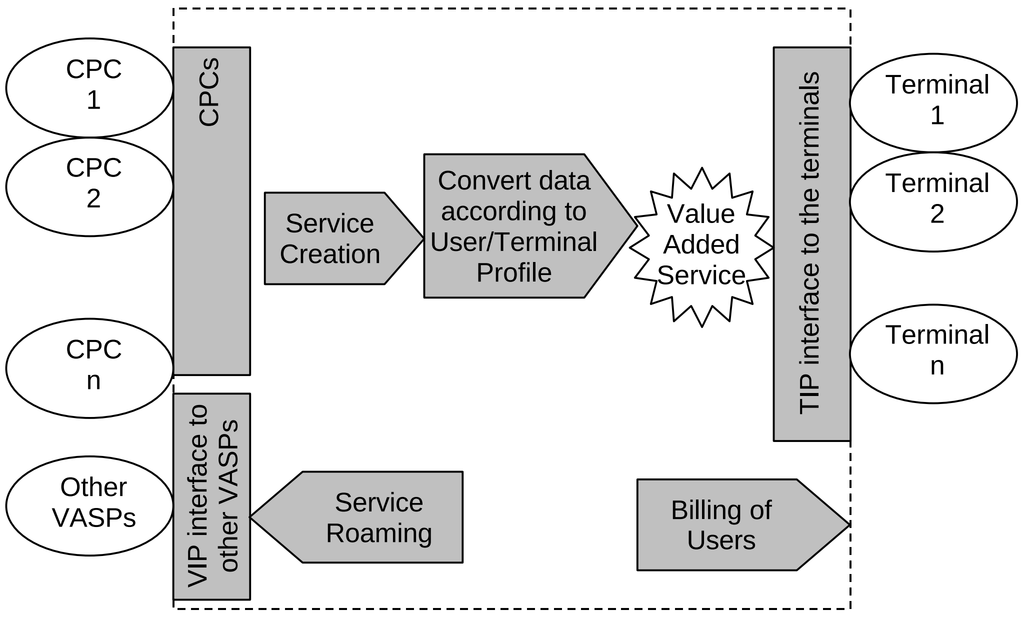
<u>Value added service provider (VASP) functionality</u>

The role of the VASP is to provide Value Added Services from the value added content provided by the CPC. VASPs communicate with CPCs via the CIP interface and with Terminals via the TIP interface. Normally the VASPs services terminal requests by acquiring the necessary information from the various CPCs and delivering it to the user in a format specified by the user/terminal profile. The VASPs may also have to push service announcements to the terminal without being requested (e.g. emergency announcements).

A major function of the VASP is to maintain customer databases for billing and user/terminal profiles. The billing systems should be able to adapt to alternative electronic payment methods (e.g. Digital Cash - eCash). User/Terminal profiling is required to provide personalised services and to determine the format in which to deliver the data to the terminal. Another requirement of the VASPs is to interface with other VASPs in order to provide service roaming (i.e when the user is not in the area of the local VASP. This is similar to the service roaming concept that mobile phone network operators use to allow customers make phone calls abroad).

The basic functions of a VASP can be summarised as follows:

- Acquire value added content from the CPC via the CIP interface
- Create value added services from the value added content
- Convert data to necessary format
- Deliver value added service to users through the TIP interface
- Maintain database of user/terminal profiles, for personalisation and billing purposes
- Enable multiple devices access services through the conversion of data based on terminal profiles
- Provide service roaming capabilities, communication will occur through VIP interface
- Establish agreements with Content Provider Centres for accessing their databases
- Establish agreements with Network Operators for using their networks
- Market the services
- Calculate the User Terminal position using the data received



User Terminal functionality

The role of the terminal is to provide the user interface to the promise services. This interface can be provided by the terminal, or by the VASP (e.g. a WML interface). The basic functions of the terminal can be summarised as follows:

- Communicate with the VASP via the TIP interface
- Render user interface
- Store secure user/terminal identification data
- Provide caching for reasonably fluid operation
- Interface with external devices e.g. GPS receiver

3.4 Existing Passenger Information Systems

Superoute 66 and Project 423 are two of the most advanced telematic-based passenger information systems in Europe today. Superoute 66 was a BT-led research project based in Ipswitch, Great Britain. While it is now discontinued, the system was installed on the 66 route, and was in use for a full year, ending November 1999. Project 423 is an ongoing project based in Helsinki, Finland. Project 423 is called such because the system is installed on the 4 bus route and the 23 tram line. The following sections examine these two systems and contrast them against the AVL framework proposed by this thesis.

3.4.1 Superoute 66

The Superoute 66 AVL system used GPS positioning and Band 3 radio for communication. LED or LCD displays at stops showed the estimated time of arrival of the next bus. Multiple delivery channels for passenger information were developed, using a Band 3 RF signal as the real-time location feed:

- **Website**: The HTML section presented static timetable information and real-time predicted arrival times at stops. A Java applet presented predicted arrival times and the real-time location of buses on a stylised map of the route.

- **SMS/Pager**: The SMS/Pager system allowed them – through the website – register to have SMS messages sent to them as certain events occur e.g An SMS could be sent to them, when the first bus after 5:30 is 10 minutes from their bus stop.

- **Smart SMS**: This is essentially a WAP interface, but implemented using the propriety smart messaging protocol of the Nokia 8110i phone. It allows the user get the estimated time of arrival of a bus at a particular bus stop.

- **Interactive Voice Response (IVR)**: Using a BT-developed voice recognition and text-to-speech platform, real-time bus location information was given over a standard telephone. Problems with large-scale deployment of IVR were identified. The BT report [12] stated that it would be difficult to design an IVR system capable of informing a traveller unfamiliar with an area, because there is no definitive list of stop names and locations. Unless users know what to say to the IVR system previously, it is difficult to navigate around, due to lack of presentation of available options.

All these delivery mechanisms communicate with the central server via TCP/IP interfaces. The architecture of the system is shown in figure 3.6

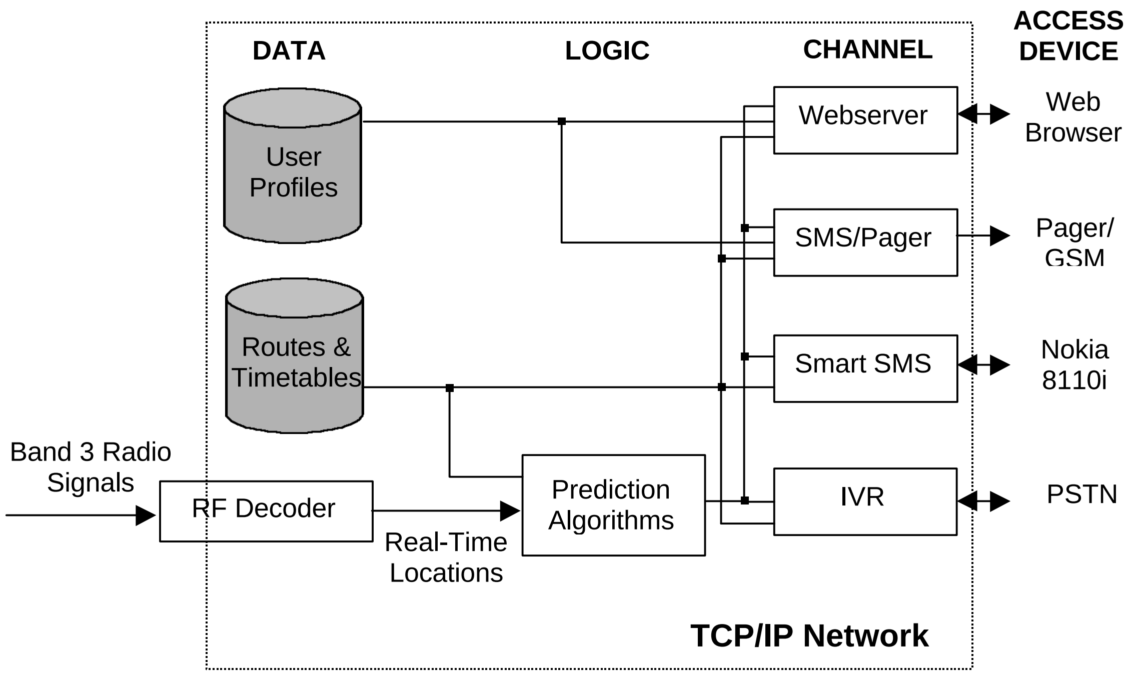


Figure 3.6: Superoute 66 Systems Architecture

3.4.2 Project 423

Project 423 is a public transport telematic system that was launched in Helsinki, Finland in 1999. The system is installed on the 4 tram line and the 23 bus route. The 4 tram line carries over 37,000 passengers daily, while the 23 bus route carries 5,000 passengers daily. Like the Superoute 66 project, the AVL system is GPS based. The positioning technique is slightly different to traditional GPS-based AVL systems:

- GPS is used to position a bus on a bus stop window – the immediate area surrounding a bus stop.
- When a bus is within a bus stop window and the doors are opened, then it is assumed to be at the bus stop.
- In between bus stops, the location is calculated from the odometer.

Another innovative feature of the 423 system is the use of traffic signal priority. A radio modem on the bus transmits a message to the signal controller approximately 150 metres before reaching a set of traffic lights. If the bus is behind schedule, the signal controller either extends or calls the green signal. When the bus passes the lights, a second message is sent to the signal controller to cancel the green signal.

The system communicates via a city-owned radio network on six different frequencies. Three frequencies are used for polling of the buses (which is done every 10th second). One frequency is allocated for signal priority, another is used to update the at-stop and on-board displays. The sixth frequency is used during the night for data maintenance at depots. The radio network consists on 3 base stations mounted on utility chimneys across the city. The real-time information, is presented using the following channels:

- **On-board driver display**: A small LCD display unit is placed next to the steering wheel. The unit displays, the next stop name, the distance to the next stop and a schedule monitoring value. The schedule monitoring value is a positive or negative time showing how much or how behind schedule the vehicle is. This allows the driver to either increase or decrease speed so that the vehicle arrives at the bus stop exactly according to schedule.

- **At-Stop display**: LED display units are placed at every stop in the route. Each unit displays the route number of the next arriving bus, the name of the bus stop and the estimated time of arrival at that stop. Arrival times are rounded to the nearest minute.

- **In-vehicle display**: LED display units are installed inside each vehicle. They display in rotation:
    - The line number, the terminus
    - The next stop name

  A speaker also announces the next stop when a vehicle leaves a stop.

- **Website**: A website is provided that shows estimated time of arrival at bus stops. The system is queried through a HTML form. It is only accessible to staff.

The next section looks at how the AVL system for Project 423 and Superoute 66 could be replaced with an inexpensive mobile positioning solution.

3.5 Proposed AVL Systems Framework

Public transport operators have a fixed budget for implementing passenger information systems. In order to maximise this budget and achieve economies of scale, the operators need to be able implement 3rd party off-the-shelf solutions. There is a need for a standard off-the-shelf communications and tracking infrastructure, i.e. AVL systems. Off-the-shelf software systems should be able to access the underlying AVL system in a standard fashion. These software systems should have precisely defined roles. The transport operators should be able to pick and choose which systems they want to use, based on their unique requirements. There should be systems to provide:

- **Internal Network Management:** These systems are developed for the network operator to be used internally. They should provide functionality such as:
    - The ability to visually track all vehicles in the network at a central control location.
    - Assist in future route planning by providing feedback about where the greatest demand for transport is.
    - On-board support systems to assist drivers.
    - Integrated ticketing and possibly support for electronic payment systems

- **System Interfaces:** These interfaces should provide standard APIs, which 3rd party applications can use to provide value added information services. The 3rd party application should be able to access information such as:
  - The real-time location of vehicles.
  - Real-time vehicle parameters such as speed, number of passengers on board, current route.
  - Static information such as Timetable/Scheduling and route information.
  - Geographical data such as the location of bus stops and stations.
  - Customer service data such as locations of where tickets can be purchased and locations of lost property offices.

The proposed AVL system framework provides all the above software systems on a common TCP/IP platform. Each vehicle in the network has a user terminal installed on it. The user terminal can simply be a GSM chip with a SIM card attached, i.e. there is no need for a keypad or user display on the user terminal. With mobile positioning, communications and positioning are integrated, which means there is no need for a separate radio network and hence no infrastructure to maintain. The position information of each of the buses is sent to the transport network central server via a TCP/IP link from the mobile phone network provider.

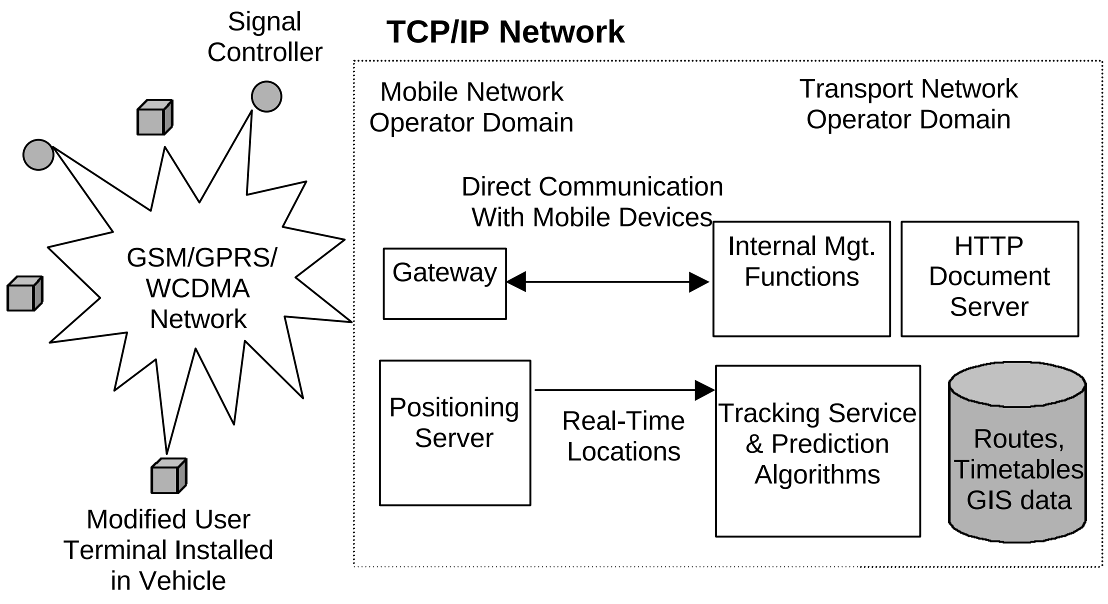

Signal Controller
TCP/IP Network
Mobile Network Operator Domain
Transport Network Operator Domain
Direct Communication With Mobile Devices
GSM/GPRS/ WCDMA Network
Gateway
Internal Mgt. Functions
HTTP Document Server
Positioning Server
Real-Time Locations
Tracking Service & Prediction Algorithms
Routes, Timetables GIS data
Modified User Terminal Installed in Vehicle

Figure 3.7: Proposed AVL Systems Framework

A server in the transport network operator domain takes the incoming real-time location feed and applies prediction algorithms to it. The internal management functions, control equipment such as LCD stop signs, on-board passenger/driver information systems, and signalling systems. Communications with the mobile equipment is done directly over TCP/IP via the gateway. In a GSM environment, the gateway must establish a virtual circuit with the mobile device, this will be billed per second. However newer mobile networks such as GPRS, are packet switched, which means the gateway can just route the packets to the correct mobile devices. Packet-switched networks can be billed per byte, whether this happens or not depends on the network operator.

An HTTP document server provides the systems interface. Using a device independent architecture described in chapter 4, the HTTP server detects what sort of content, the device expects. It then hands off the interaction to an interface tree for that device. In this way, all external access to the system can be controlled through simple HTTP access.

From now on the proposed AVL system framework shall be referred to as the Mobile-AVL or M-AVL system.

CHAPTER 4

Design

The design of the passenger information system implemented for this thesis, borrows ideas from a wide cross section of current technologies and reassembles them in a way that is quite unique. The architecture borrows many aspects of the its architecture from the PROMISE generic system architecture. Specifically it uses a service chain architecture to deliver information to the end user. Service chains or integrated value chains as they are also known allow multiple content and application service providers present a service to the user as if the service was provided by just one entity. The entity at the end of the chain provides a unifying interface to a multitude of service chains. Each service chain can consist of a number of links each link in the chain adding value to the content and/or functionality provided by the previous link. Service chains are essentially Porters value chain [13] applied to a loosely coupled distributed system.

Another unique aspect of the systems architecture takes a tried and tested design pattern – the Model View Controller pattern – and applies it to the problem of device independent document delivery. This allows the document server to dynamically deliver different interface trees to different devices e.g. web browsers are delivered a html interface tree and voice-only telephones can be delivered a VoiceXML interface tree.

The system is designed to accept a real-time location feed from an AVL system as described in section 3.5. However, given the uncertainty about which positioning technique will become the dominant one, the overriding systems design assumption is to build the system flexible enough to accept any position feed no matter how accurate or inaccurate it is.

The passenger information system implemented for this thesis – which from now on will be called the Dublin Bus Tracking Service - consists of four main components:

- **The Simulator**: This component simulates a network of buses, which have implemented the proposed AVL system framework discussed in chapter 3. The main purpose of this component is to simulate the output of a Positioning server. This simulated real-time location feed is continuously sent to the tracking service

- **The Tracking Service**: This component is the heart of the passenger information system. It decodes the incoming location feed, figures out which message belongs to which bus on which route, and then recalculates the buses location on a map. At this point it predicts a new estimated time of arrival at the next bus stop.

- **The GIS Service**: This geographical information system component provides road and route vector data to the tracking service. It also provides locations of bus stops. The GIS service provides algorithms to overlay the bus stops on routes to create what’s called a driving execution plan.

- **The HTTP Document Server**: This component serves the user interface to the user agent. A user interface is chosen to match the characteristics of the user agent on the first request of that user agent. After that, the chosen user interface tree handles all interactions.

The rest of the chapter describes these components in terms of a service chain architecture, a functional architecture and a communications architecture. Finally the Model View Controller pattern is introduced and the device independent architecture used by the Dublin Bus Tracking Service is explained.

4.1 Service Chain Architecture

Since no single organisation runs the entire transport infrastructure of a country, only an independent 3rd party can provide a truly intermodal passenger information system. The role of the 3rd party should be to aggregate the content feeds from the different transport network providers and to deliver a unified information system to the end user. This sort of system is best described using the service chain metaphor. The different entities in the service chain are shown in figure 4.1

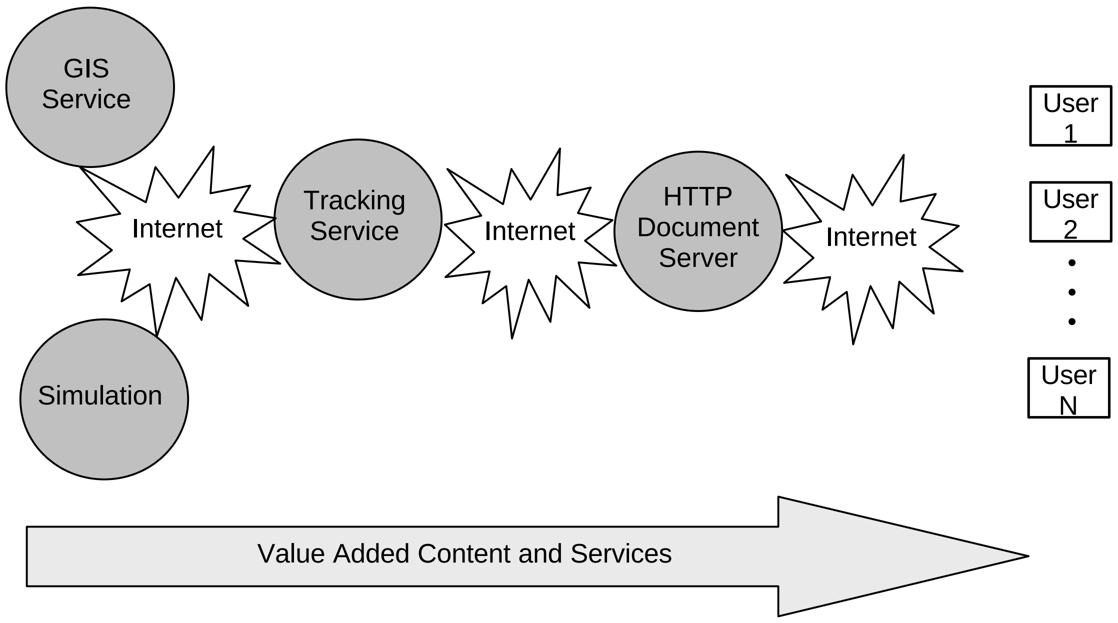


Figure 4.1: Service Chain Architecture

In terms of the PROMISE architecture the Content Providers in the system are the GIS service and the Simulator. The Content Provider Centre (CPC) is the Tracking Service. The Value Added Service Provider is the HTTP Document Server.

4.1.1 Value Added by Tracking Service

The Tracking Service adds value to content and service in the following ways:

- **Integrates content**: Content is taken from two sources, the Simulation and the GIS Service.

- **Internal tracking tables**: Tracking tables are used to keep track of the locations and attributes of all buses currently on route. The tracking tables are grouped in a number of different ways to enable fast lookup. Every time the location of a bus is updated the tracking table for the bus is recalculated. This involves, calculating the bus's position on the route vector, calculating the direction the bus is travelling in, calculating the next and last stops and calculating the estimated time of arrival at the next stop.

- **Prediction algorithms**: The tracking service provides prediction algorithms to predict when a bus will arrive at a particular stop.

- **Provides a simple VASP interface**: The HTTP Document Server (VASP) can access the tracking service through a simple interface.

4.1.2 Value Added by Document Server

The Document server adds value to content and service in the following ways:

- **Provision of User Interface**: A user interface tree is dynamically generated for a device depending on its device type. Once the initial detection of a device occurs, the document hands the request over to the relevant interface tree. The interface tree handles all generation of content and interaction logic for a user session.

- **Content Provider Centre**: The HTTP document server can act as a Content Provider Centre to other VASPs. It dynamically generates XML on request to describe the location of all the buses on a particular route. It provides a very simple way for another VASP to integrate into this VASP's service chain. In terms of the PROMISE architecture this is called the VIP interface.

4.2 Model View Controller Architecture

The HTTP Document Server uses a software design pattern called Model-View-Controller (MVC) pattern to provide the appropriate user interface to the end-user. The MVC pattern was first made popular by the programming language Smalltalk [14]. The MVC pattern is enforced when designing user interfaces in Smalltalk. The advantage of the MVC pattern is that it limits and defines the interaction between the interface components and the underlying problem-domain classes. In Smalltalk the Model is the Problem domain class. The view is the class that receives input from the user (e.g. mouse clicks, keyboard presses) and displays output to the user. Finally the Controller class intercepts all events and sends the relevant ones to the view when appropriate (e.g. a click within the bounds of a graphical object)

This pattern can quite easily be applied to device independent web architecture. In this model the Controller is a servlet which listens for all or a subset of incoming requests. The controller is configured to listen for subsets of incoming events, in this model the events that the controller handles are the URLs it listens for. For example a controller servlet may be configured to listen for the `http://localhost/foo/` event and to ignore the `http://localhost/bar/` event. In the traditional MVC pattern the Controller passes the event to the relevant view, in this model it is no different. The controller servlet queries the HTTP accept and user-agent headers to find out what type of device is making the request. It then passes the request to the appropriate view – or more accurately an interface tree.

The role of the View class in smalltalk is to present visually a representation of the underlying model. In this case the view is a document-based user-interface tree, and the underlying model is a list of bus tracker objects. The UI tree is actually a series of hyper-linked views. When the controller intercepts the event it passes the event onto the root view in the UI tree. After this all interactions now continue without the controller and are contained with in the UI tree.

The user interface trees are document-based because they are written in a declarative user interface language such as HTML, WML or VoiceXML and can be delivered to the end user via an HTTP document server. Device independence is therefore achieved by delivering different views to different devices while keeping the underlying problem domain class - the Model - common.

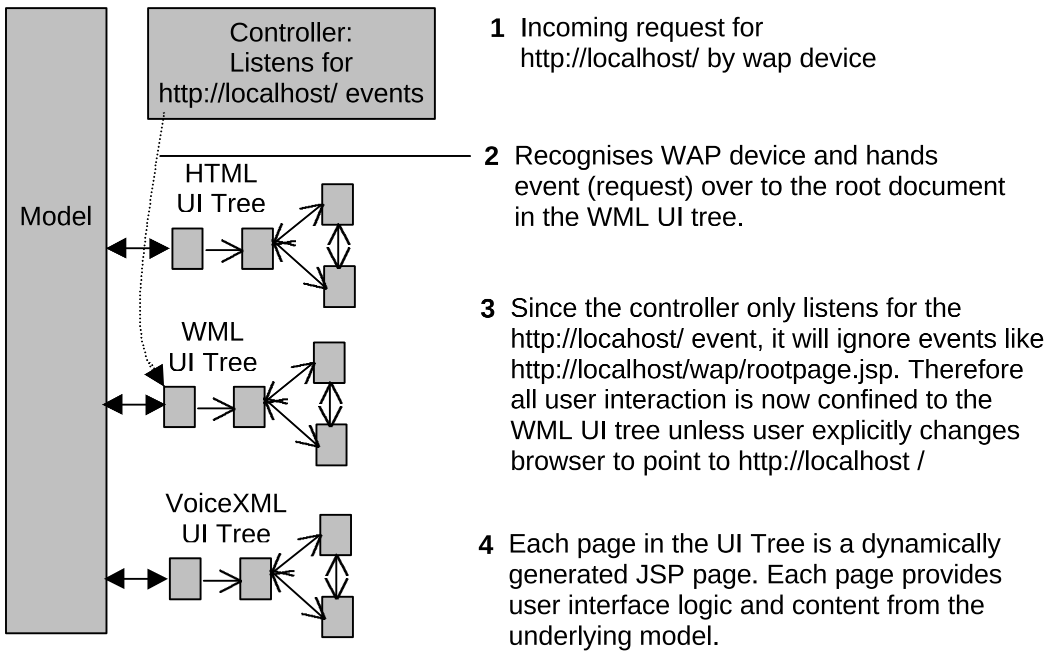

Model
Controller: Listens for http://localhost/ events
HTML UI Tree
WML UI Tree
VoiceXML UI Tree
1 Incoming request for http://localhost/ by wap device
2 Recognises WAP device and hands event (request) over to the root document in the WML UI tree.
3 Since the controller only listens for the http://locahost/ event, it will ignore events like http://localhost/wap/rootpage.jsp. Therefore all user interaction is now confined to the WML UI tree unless user explicitly changes browser to point to http://localhost /
4 Each page in the UI Tree is a dynamically generated JSP page. Each page provides user interface logic and content from the underlying model.

Figure 4.5: Adapted Model-View-Control Pattern

4.3 Functional Architecture

The Dublin Bus Tracking service consists of four functional entities which work together to provide a coherent service.

4.3.1 GIS Service

The GIS Service provides an interface to a database of geographical information consisting of

- **Centre-line road vector data**: This is a network of interconnected line segments. Each line segment corresponds to a theoretical line running down the centre of an actual physical road.

- **Bus Route vector data**: This is that subset of road vectors that make up one particular bus route. It is different to road vector data in that the set of road vectors that make up a bus route have a direction, whereas individual road vectors are bi-directional.

- **Bus stop data**: The GIS service provides access to the list of bus stops. Each bus stop is geo-coded, which means its x, y co-ordinate is given a human readable name. Each bus stop also can be queried for the street it is on.

- **Street names**: A list of street names is provided along with the set of road vectors that make up that street.

The GIS Service also provides functionality to overlay the bus stop data onto the bus route vector data. The overall functionality provided by the GIS service is summarised in figure 4.2

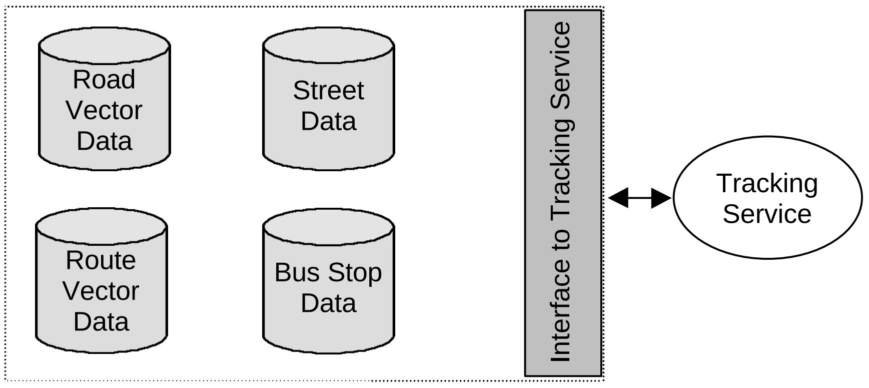


Figure 4.2: GIS Service Functional Architecture

4.3.2 Simulator

The role of the simulator is to simulate the output of the Positioning Server described in the M-AVL system. To do this requires simulating the Dublin bus transport network. Buses depart from virtual termini according to timetables stored in a database. They drive along virtual roads provided by the GIS service. They stop at virtual bus stops and take on (and let off) virtual passengers. At regular intervals each bus uses the simulator proxy object to send a message to the tracking service stating its location, speed and number of passengers on board.

Each bus in the network has a unique id. If the real-time feed came from the M-AVL system this id would be the SIM number of the mobile positioning device. When a bus starts a trip, it registers with the tracking service, passing its unique id and the route it is on. In the M-AVL system, the driver or someone who is monitoring the bus would have to enter this information manually.

The functional architecture is shown in figure 4.3

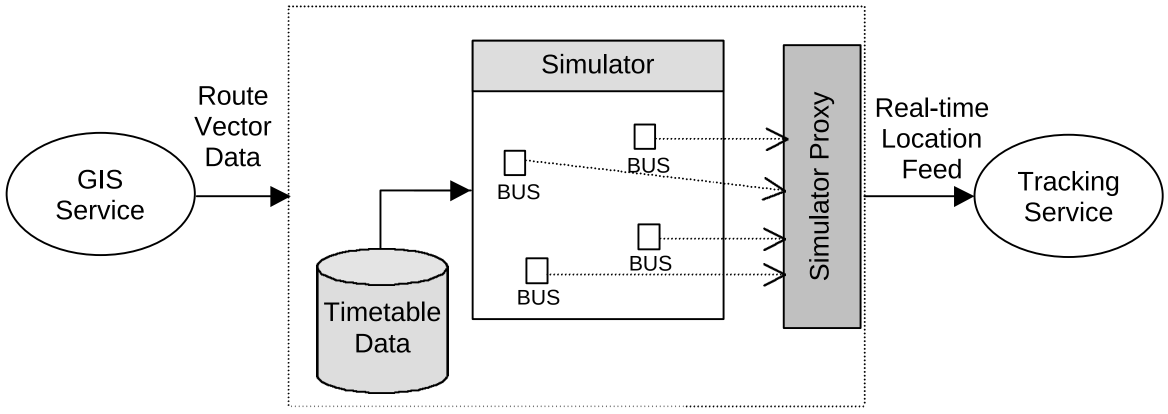


Figure 4.3: Simulator Functional Architecture

4.3.3 Tracking Service

The role of the tracking service is to maintain the location of all the buses currently driving on a route in internal tracking tables. It must keep also track of speed and the number of passengers on the bus. When the tracking service receives a registration message from the simulation/M-AVL Position Sever it records the unique id and corresponding route in the tracking tables. When subsequent location messages are received, only the unique id is sent with them. Since the tracking service knows what route that bus is on it can accurately update the location and attributes of the bus. The updated location message may state a position that is not exactly on the route, in this case the tracking service finds the nearest node or line and snaps the location to this point. Each bus currently on a route has an associated BusTracker object associated with it in the tracking service. BusTracker objects are created when the tracking service receives registration messages. These objects are then indexed in a number of ways to enable fast lookups. The functional architecture is shown below, where N in the diagram is the number of buses currently on the route.

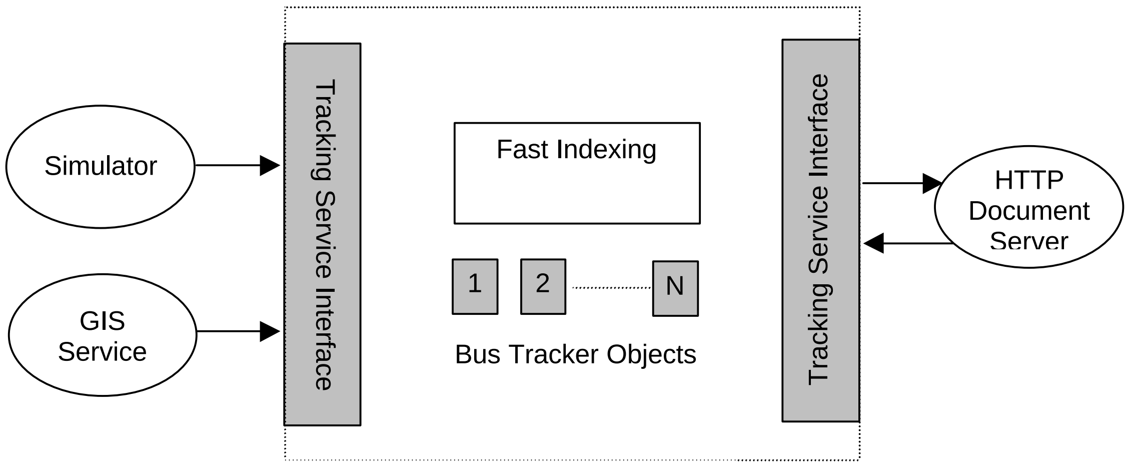


Figure 4.4: Tracking Service Architecture

4.3.4 HTTP Document Server

The HTTP Document Server uses the Model-View-Controller pattern to provide a device independent interface to the tracking service. This allows devices with different display characteristics access the same URL and receive an appropriate interface for that device. For example if a PC-based web browser and a WAP-enabled mobile phone point to the same URL, the web browser will get an HTML interface and the WAP-enabled mobile phone will get a WML interface.

The Web Server is called an HTTP Document Server to reinforce the fact that it serves document-based interfaces to multiple devices. A web server is usually associated with serving HTML and images to PC-based web browsers.

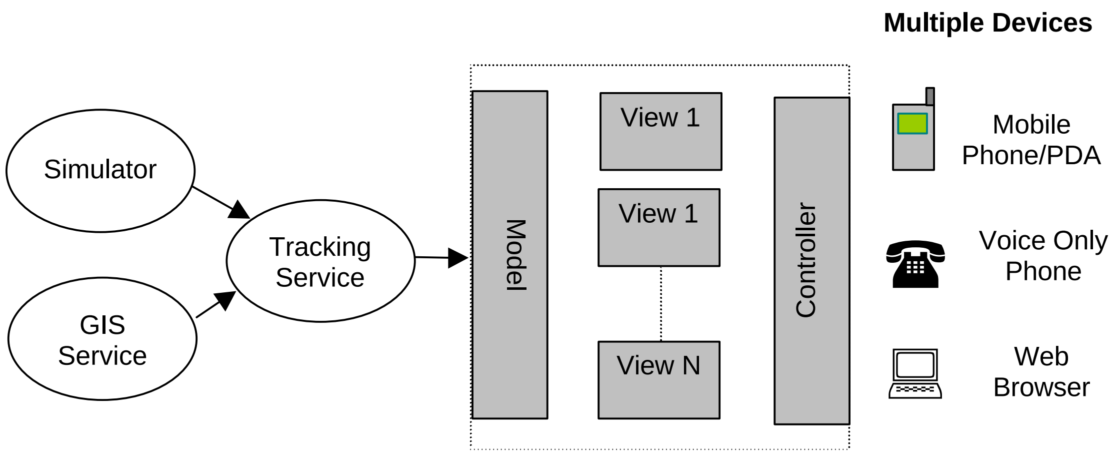


Figure 4.5: HTTP Document Server Architecture

4.4 VoiceXML

In order for a voice only phone to interpret the VoiceXML document served by the HTTP Documents server, an extra two layers need to be added to the architecture. These layers are placed between the HTTP document server and the phone. They consist of a VoiceXML Interpreter Context, a VoiceXML Interpreter and an Implementation Platform. This Architectural model is defined by the VoiceXML Forum [15].

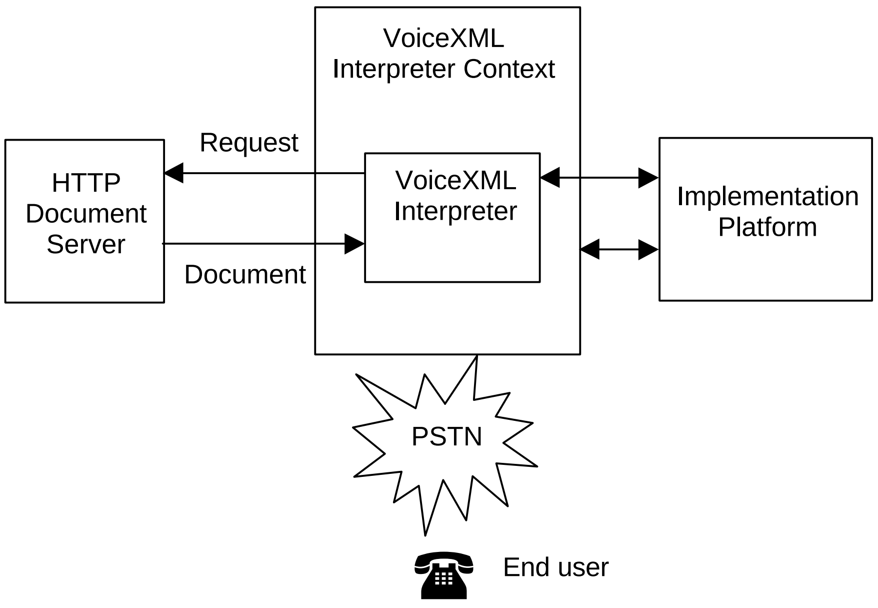


Figure 4.5: VoiceXML Architectural model

The implementation platform performs the following functions:

- Provides Voice recognition capabilities, and delivers the data to the Interpreter or Interpreter Context
- Recognises DTMF tones, and passes them to the Interpreter or Interpreter Context
- Provide text-to-speech (TTS) capabilities for audio presentation of content. It must also be able to play back pre-recorded audio files.

The Interpreter Context, responds to incoming call requests. When a call comes in, the Interpreter Context retrieves the root document of the UI tree and passes it to the Interpreter. The Interpreter Context must also listen for special escape codes independent of the interaction logic defined in the VoiceXML document. For example an application may define an escape code to be the word “operator” or DTMF tone 1. When the Interpreter Context hears this escape code, it hands the voice call over to a human operator.

The interpreter parses and validates all VoiceXML documents. It is responsible for conducting the interaction logic defined in the VoiceXML document. It receives voice/text input from the implementation platform and sends text to the implementation platform for audio presentation.

The Interpreter Context, the Interpreter and the Implementation platform would usually reside on a different node to the document server. In fact in a real-world scenario, they would probably reside on a load-balanced farm of nodes because performing speech recognition on simultaneous voice streams is computationally very expensive.

4.5 Communications Architecture

If implemented in a real-world scenario, each of the four components in the Dublin Bus Tracking Service would have different owners and would reside on geographically separated IP nodes. In fact this is the reason that the service chain architecture was used in the first place. The service chain architecture is a very good way of integrating geographically distributed applications in an Internet environment when there are multiple owners of the different components in the distributed system. This section describes the complete communications architecture of the Dublin Bus Tracking Service in a real world environment.

In a real-world system, there would be no simulation component. The M-AVL Positioning Server would replace the simulation component and would reside in the domain of the mobile network service provider. A third party that specialises in GIS systems would own the GIS service component. The GIS component would either reside in the transport operator’s domain or in the third parties domain. The choice of location would be dependent on performance criteria.

Both the tracking service and the HTTP document server would be maintained and operated by the transport network service provider. Both components would reside in the domain of the transport network service provider but would reside on different nodes for scalability reasons. The VoiceXML application software would reside on a third party server. This third party would specialise in delivering voice applications and would be better able to absorb the cost of deploying a concurrent real-time voice recognition engine. Finally the system may interface with the systems of the road service provider (this is the organisation that maintains the traffic light network). The road service provider would need to install a Proxy Server that could communicate with tracking service. Note that all inter-domain communication is via a TCP/IP network. The actual protocol used e.g. Sockets, IIOP, HTTP, RMI, DCOM or SOAP is left to the implementation of the system. The proposed communications architecture is shown in figure 4.6, a more detailed description of the M-AVL system is given. Section 6.3 shows how this architecture can be modified to make it more scalable.

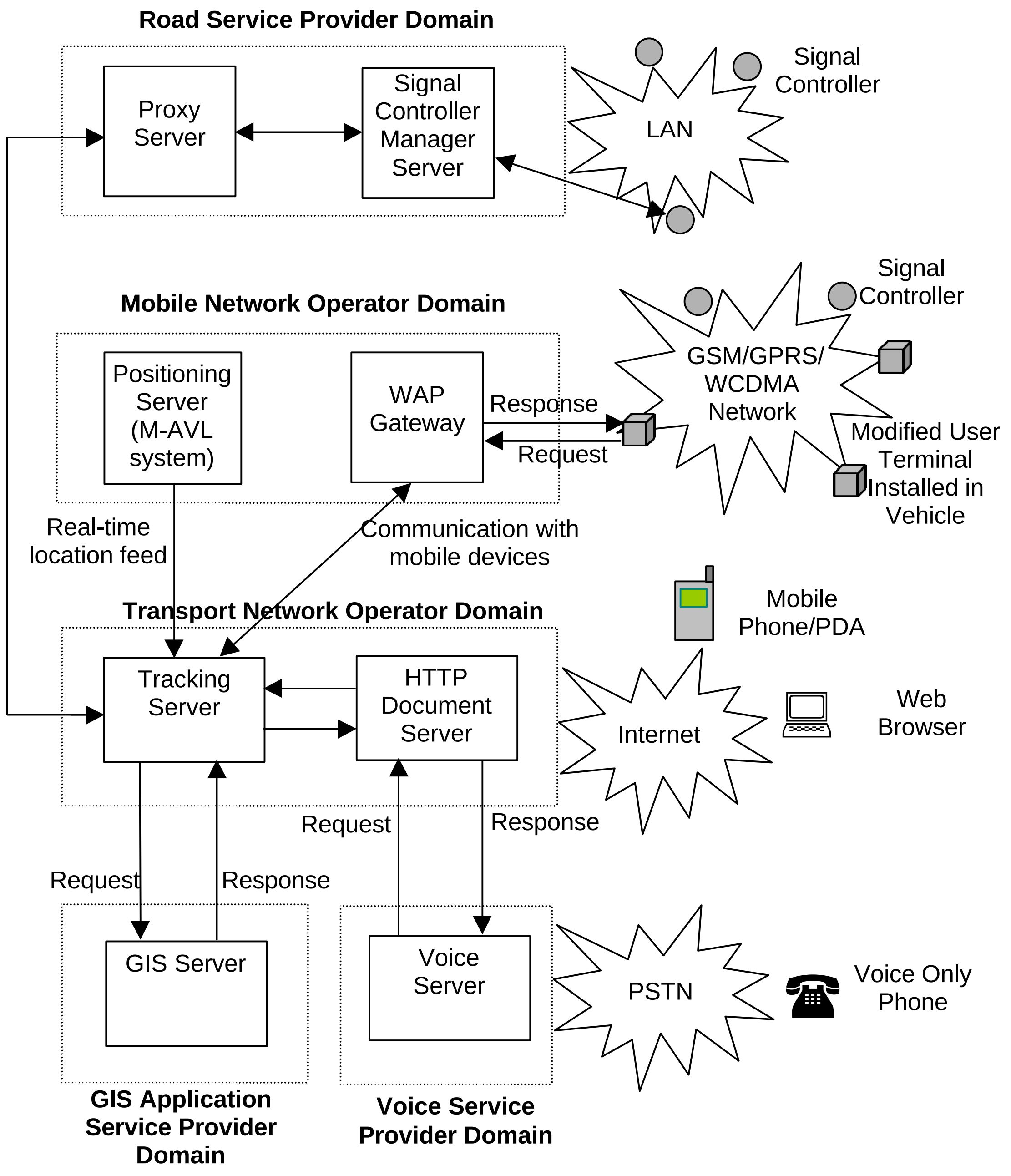

Road Service Provider Domain
Proxy Server
Signal Controller Manager Server
LAN
Signal Controller
Mobile Network Operator Domain
Positioning Server (M-AVL system)
WAP Gateway
Response
Request
GSM/GPRS/ WCDMA Network
Signal Controller
Modified User Terminal Installed in Vehicle
Real-time location feed
Communication with mobile devices
Transport Network Operator Domain
Mobile Phone/PDA
Tracking Server
HTTP Document Server
Internet
Web Browser
Request
Response
Request
Response
GIS Server
Voice Server
PSTN
Voice Only Phone
GIS Application Service Provider Domain
Voice Service Provider Domain

Figure 4.6: Proposed Communications Architecture

The M-AVL system consists of network of mobile devices connected to a mobile network. These devices can be either

- **Tracking devices**: Installed on buses and positioned using the Positioning Server. Each of the tracking devices can send messages to the tracking service via the WAP gateway. For example, when the number of passengers on the bus change, the device can inform the tracking service of this fact via the WAP gateway.

- **Signal Controllers:** These are micro-controllers with the ability to communicate over the mobile network. When the tracking server calculates that a bus is behind schedule and is near a set of traffic lights, it sends a message via the WAP gateway to the signal controller instructing it to extend its green signal. However it may not be necessary to modify individual signal controllers if they are already networked. In that case the tracking server would communicate with the server that controlled the signal controllers.

The tracking service either continuously polls the positioning server or the positioning server continuously sends updates to the tracking service. In either case the result is a real-time position feed delivered to the tracking service. This position feed consists of the x, y position of the bus, its speed and the direction it is travelling in. However the tracking service also needs to know how many passengers are on board. This value can be sent periodically by the tracking device on the bus via the WAP gateway. Counting passengers can be done by sensors on the doors or manually by the driver.

On starting up, the tracking service loads in the necessary geographical data from the GIS service. Finally the HTTP Document server provides interfaces to multiple devices that allow them to query the internal state of the tracking service in a user-friendly manner. Note that WAP-enabled phones and PDA’s communicate with the HTTP Document Server via a WAP Gateway. This WAP Gateway may be the same gateway used by the tracking service to communicate with the mobile devices on the buses. It will be the same if the user has the same mobile network operator as the tracking service.

The next chapter looks at a prototype implementation of this architecture.

CHAPTER 5

Implementation

A Passenger information system is a large system consisting of many components. This thesis does not attempt to implement a full real-world passenger information system. The goal of this thesis was to develop a prototype system that would validate the design in chapter 4. Some of the aspects of the real-world system were not included in this proof-of-concept prototype.

Firstly since the M-AVL system could not be implemented, there was a need to build a bus simulation to provide an artificial position feed. However in order to simulate buses driving on virtual roads and stopping at virtual stops, geographical data is needed. Therefore the first system to be implemented was the GIS service. After that was the simulator, then the tracking service and finally the HTTP Document Server and corresponding views. In fact the implementation sequence followed the same sequence as the service chain described in section 4.1

Three views were implemented an HTML view, a WML view and an applet view. The decision was made to not implement a VoiceXML view for two reasons:

- This is a proof-of-concept implementation, and VoiceXML has been incorporated into the design anyway

- The Voice Recognition Software is only commercially available

Other than this most of the other concepts from the design have been implemented in the Dublin Bus Tracking service prototype

5.1 GIS Service Implementation

Geographical data provides the foundation on which the simulator is built. The first stage in the implementation of the GIS service was the digitising of a raster map of south Dublin. The next stage was to associate street names with the corresponding vectors. This process is called geocoding. Bus routes where created from the road vector data. A survey of all the 15, 15a and 15B bus stops was conducted, resulting in x, y co-ordinates and names for all these bus stops. All this geographical data was imported into the MySQL database shown in Appendix A.

With the data gathering exercise complete, the GIS service was developed using Java 1.3. Objects were created to represent the different geographical objects in the database. Algorithms were developed to overlay groups of geographical objects to create a vector path joining all the objects. The following sections describe the implementation process in more detail.

5.1.1 Map Digitising & Geocoding

The Dublin Street Guide [16] was used as a basis to create the road vector data needed for the simulator. This provides detailed street data of Dublin city and district at a scale of 1:15,000. However this data resides on printed paper, separated into 58 A4 pages. Therefore all the pages corresponding to south Dublin – from Lucan out to Killiney – had to be scanned. The pages were then joined together in Photoshop to create one large raster map of south Dublin.

A software package called MapInfo Professional 6.0 (see figure 5.1) was used to digitise the raster map. Firstly the raster image was imported into the package. In order for MapInfo to calculate the co-ordinate of each pixel in the raster, fifteen known co-ordinates were entered. These co-ordinates where points on the map whose co-ordinates were previously known, e.g. the co-ordinate of the top left of the Lucan page is 303,000, 236,000. These co-ordinates use the Irish Transverse Mercator Grid co-ordinate system.

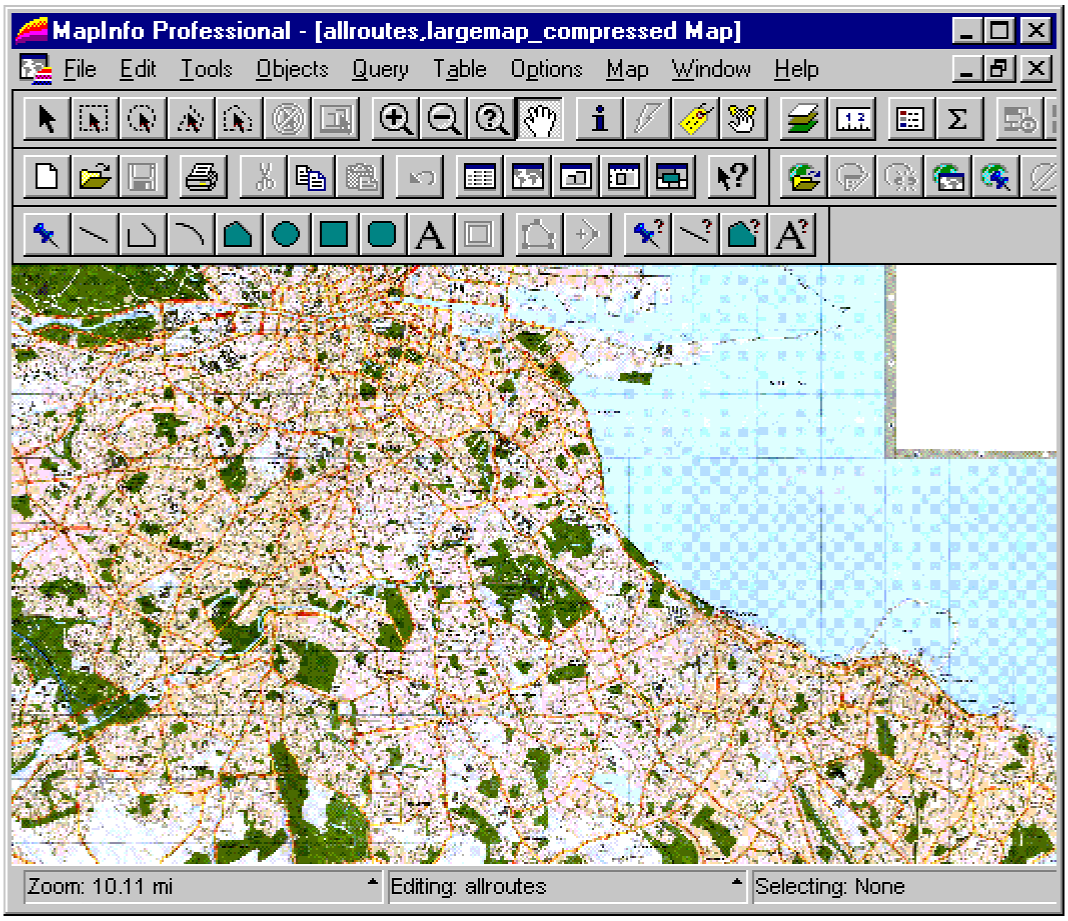


Figure 5.1: MapInfo Professional 6.0 showing the Dublin Raster Map

Digitising the road data consisted of overlaying polylines on top of the road network displayed in figure 5.1. After that street names were assigned to groups of polylines since each street consists of one or more polylines. Each poyline was assigned the id of the street it was on. Note that the polylines were plotted along the centreline of each road. This vector data was exported to a MIF (MapInfo Interchange File) file, which was parsed and the data was entered in the roadsegments and streets tables of the MySQL database (see Appendix A).

Bus routes were entered into MapInfo next. Each polyline on the route was assigned a route id and an index. Each polyline on the route was indexed sequentially from the start of the route to the end. This vector data was also exported to a MIF file and parsed. The parsed data was entered into the routes and busroutes tables. A separate file containing timetable information for that route was also parsed, with the data being entered into the timetable table.

Finally bus stop data was gathered for the 15, 15A and 15B bus routes. This involved doing a survey of the route. Each bus stop on each route was visited, its position on the map was recorded. Each stop was also given a name that was descriptive enough for users to recognise on a list. For example the 15 bus stop in Rathmines beside dominos pizza was called *beside dominos pizza.* After the survey was complete, the positions of all the bus stops were entered into MapInfo and assigned names. Each route was then assigned a list of bus stops. Again the data was exported to a MIF, parsed and then entered into the database.

5.1.2 Java Implementation

Each of the geographical entities in the database has a corresponding object wrapper. The objects are Node, LineSegment, Street, BusStop, RoadNetwork, RouteNetwork, BusRoute and BusRouteNetwork. The Node class extends java.awt.geom.Point2D.Double and BusStop extends Node. A LineSegment consists of two Nodes and extends java.awt.geom.Line2D.Double. All Nodes have a corresponding Street.

An interesting aspect of the GIS service implementation is the algorithm used by the BusRoute object to overlay the bus route with the corresponding bus stops to create a driving plan. A driving plan is a list of sequential nodes that a bus must visit on its journey. The bus starts at the first point in the list and ends at the last. The algorithm is as follows:

1) A list of sequential bus stops and sequential road vectors (the route) is given.
2) Pop a bus stop of the top of the list if it is not empty otherwise finish
3) Continue to pop road vectors of the top of the road vector list until a road vector is reached that is on the same street as the bus stop. As each road vector is popped off the list, its points are added to the driving plan.
4) Create a list of line segments representing that street.
5) Calculate the nearest perpendicular line segment and the nearest node to the bus stop.
6) If a line segment is nearer than a node then calculate the intersection point with the line segment. This is the insertion point.
7) Otherwise the node is the insertion point
8) Add all the road vectors on the current street that occur before the insertion point to the driving plan.
9) Add the insertion point to the driving plan
10) Add the bus stop to the driving plan
11) Add the insertion point to the driving plan again
12) Go to step 2

The GIS service is a singleton object, which means there can only be one instance of the service in a single virtual machine. It is loaded into the simulator by calling the static method `GisService.getInstance()`

5.2 Simulator Implementation

The simulator consists of four Java Classes: Bus, BusDispatcher, Scheduler and SimulatorProxy. The simulator is multithreaded, with each bus running in its own thread. The Scheduler thread looks up the timetable to see when the next buses are leaving. It then creates a bus dispatcher thread, passing it the time the next buses are due to depart. The scheduler thread then puts itself to sleep until this time.

The scheduler thread wakes up, and starts the dispatcher thread. The scheduler thread then looks up the timetable database again, and follows the same steps as before. The dispatcher thread meanwhile is running at the same time. The dispatcher thread looks up all the buses that are scheduled to leave at the given time. It creates a new bus thread for each of these buses and starts them. The bus thread is the workhorse of the simulation and is covered in section 5.2.1. Once all the bus threads are created and started, the work of the dispatcher thread is over and it is marked for garbage collection.

5.2.1 The Bus Thread

When a bus thread is first started it registers itself with the tracking service. It does this through the SimulatorProxy object, which uses RMI to communicate with the tracking service. The decision was taken to use RMI for inter-process communication for ease-of-implementation reasons. However in a real-world scenario RMI would probably not scale well enough. Registration involves passing the buses unique id, route id and its depart time to the tracking service.

After registration, the bus thread loads its driving plan from the GIS service. It also loads the bus.properties file (see Appendix B) which affects how the bus drives along the route. There is a property in the bus.properties file called *`update-thread-frequency.`* A bus drives at a speed of *`avg-speed`* when it starts a journey and when it leaves a stop. Every *`update-thread-frequency`* seconds a random deviation is applied to the speed. The deviation can be positive or negative but has a maximum absolute value of *max-speed-deviation*. A bus can travel no faster than *speed-limit* kilometres per hour.

Every *update-thread-frequency* seconds the bus thread wakes up and recalculates its position on the route. It calculates the distance travelled in metres since it last woke up. The bus knows the last visited node in the driving plan and the next node in the driving plan. The last known location is a point somewhere in between the last node and the next node. The bus thread calculates the distance from the last known location to the next node. If this distance is smaller than the distance travelled then the next node becomes the last node and the last known location is the last node. Otherwise the last known location becomes that point which is distance_travelled metres ahead of the previous last known location. This process continues until all the nodes in the driving plan have been visited.

When the bus reaches a node it checks to see if it is an instance of BusStop. If it is, then it calculates the number of passengers that are boarding and the number of passengers that are de-boarding. To explain how these values are calculated consider figure 5.2

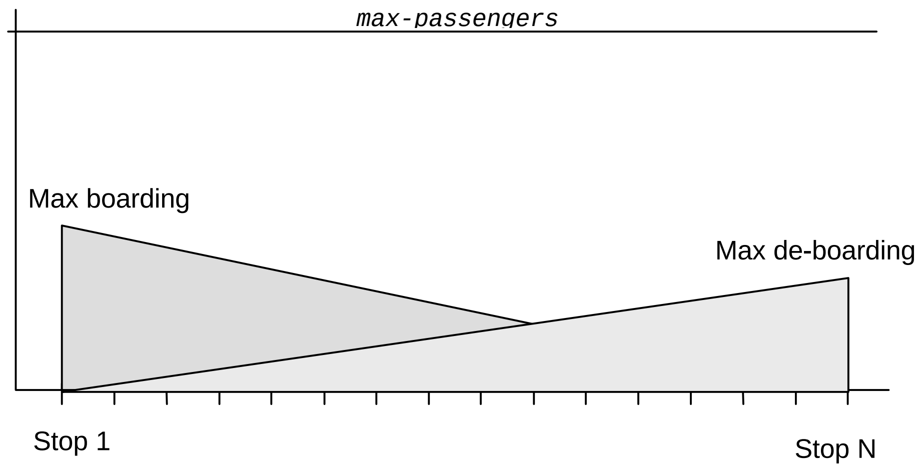


Figure 5.2: Passenger boarding and de-boarding envelopes

Figure 5.2 describes the boarding and de-boarding envelopes for a route with N stops. The maximum number of passengers that can fit on the bus is *max-passengers*. The number of passengers boarding the bus at each stop decreases linearly from a peak value of max boarding. The value of max boarding is *max-passengers * max-boarding-percentage / 100*. Similarly the number of passengers de-boarding the bus increases linearly from zero to a maximum value of max de-boarding. The value of max de-boarding is *max-passengers * max-deboarding-percentage / 100*. Therefore, the number of passengers boarding or de-boarding at a stop is the y value of the boarding or de-boarding envelope at that stop. In order to add some variation to the simulation, a random deviation is applied to the number of passengers boarding and de-boarding. The bus will stay at the stop either (*`passengers_boarding * seconds-to-board)`* seconds or (*`passengers_deboarding * seconds-to-deboard)`* seconds, depending on which value is greater.

Every *`update-tracking-service-frequency`* seconds the bus sends a message to the tracking service via the SimulatorProxy. The message consists of the current x, y co-ordinates (in Irish Transverse Mercator Grid format), the current speed and the number of passengers on board.

5.3 Tracking Service Implementation

When the tracking service receives a registration message, it creates a BusTracker object for that bus. This object will live as long as the bus is on a route. The tracking service also makes entries into the following three tables for fast indexing:

- Bus id to Route id mappings: This allows the corresponding route id to be found from a particular bus id.
- Tracking table: This table stores BusTracker objects, indexed by bus id.
- Route tracking table: This table stores groups of BusTracker objects, indexed by route id.

On startup, the tracking service registers with the RMI registry under the name *TrackingService*. It also loads in the necessary geographical data from the GIS service. When there is at least one BusTracker created, the tracking service can process the update messages (i.e. the real-time location feed). Each update message contains the bus id, x & y co-ordinates, speed and number of passengers on the bus. The relevant BusTracker is looked up in the tracking table, and its `updateLocation` method.is called.

5.3.1 The Bus Tracker

When the `updateLocation` method is called, the bus tracker sets the last known location to the x and y co-ordinate given. The current speed and number of passengers is also set. However because the x, y location may not always be accurate, an algorithm is run to calculate the buses actual location on the route. Every time the `updateLocation` method is called the buses position is calculated from scratch. The algorithm consists of the following procedures:

1) Each node on the driving plan is examined, and the distance between each node and the x, y co-ordinates given is calculated. The nearest node to the updated location (the x, y co-ordinates) is found.
2) A test is done to check if the updated location is before or after the nearest node.
3) The intersection point of the updated location with the nearest perpendicular line segment is calculated. This becomes the new last known location.
4) The previous bus stop and next bus stop is calculated.

Using this procedure minimises errors arising from inaccuracies in the real-time location feed.

The BusTracker class also provides a simple prediction algorithm for predicting when the bus will arrive at a particular stop. The bus tracker calculates how many metres it is to the given stop from the last known location. Then using the current speed it estimates the time it will take to get to the stop. If the bus is stationary, the average speed is used to calculate the estimated time of arrival.

5.3.2 Tracking Service as the Model in the MVC pattern

The tracking service returns the state of a particular bus as a table of name value pairs. These tables are called location tables. The tracking service provides methods for views, to allow them query the state of the tracking service. A view can query a particular route and will get a list of location tables in return. Each location table corresponds to a bus currently on that route. The tracking service also provides the capability of retrieving location tables for individual buses or all the buses currently on the route network. The names in the location table are as follows:

- BUS ROUTE
- ROUTE NUMBER
- TERMINUS DEPART TIME
- CURRENT LOCATION
- CURRENT X
- CURRENT Y
- LAST STOP
- NEXT STOP
- CURRENT SPEED

- PASSENGERS ON BOARD
- ETA AT NEXT STOP

5.4 HTTP Document Server Implementation

The web server - or HTTP Document server as it is called in this thesis – used to deliver documents to users is Tomcat 1.0. This is the reference implementation of the Java Servlet 2.2 and JavaServer Pages 1.1 specifications developed by the Apache software foundation. It is a fully compliant Servlet container and JSP engine. The interface trees described in chapter 4 are implemented as JSP pages. HTML and WML user interface trees were implemented. In the MVC pattern, the UI trees are the views and the Tracking Service is the model.

The JSP pages use the WebServerProxy Object to communicate with the Tracking Service. On startup the WebServerProxy looks up the TrackingService in the RMI registry. Like the GIS service the WebServerProxy is a singleton object. A handle to the object is obtained by calling the static method `WebServerProxy.getInstance()`.

5.4.1 Device Independence

The Controller from the MVC pattern described in chapter 4 is implemented as follows:

- Tomcat is set up to run on port 80

- The default file for the root directory is set to device.control instead of index.jsp or index.html. i.e. when the user types in http://localhost/ they are actually calling http://localhost/device.control

- A servlet called Controller is set up to listen for requests for http://localhost/device.control. In the MVC architecture requesting this url generates an event which is handled by the Controller servlet.

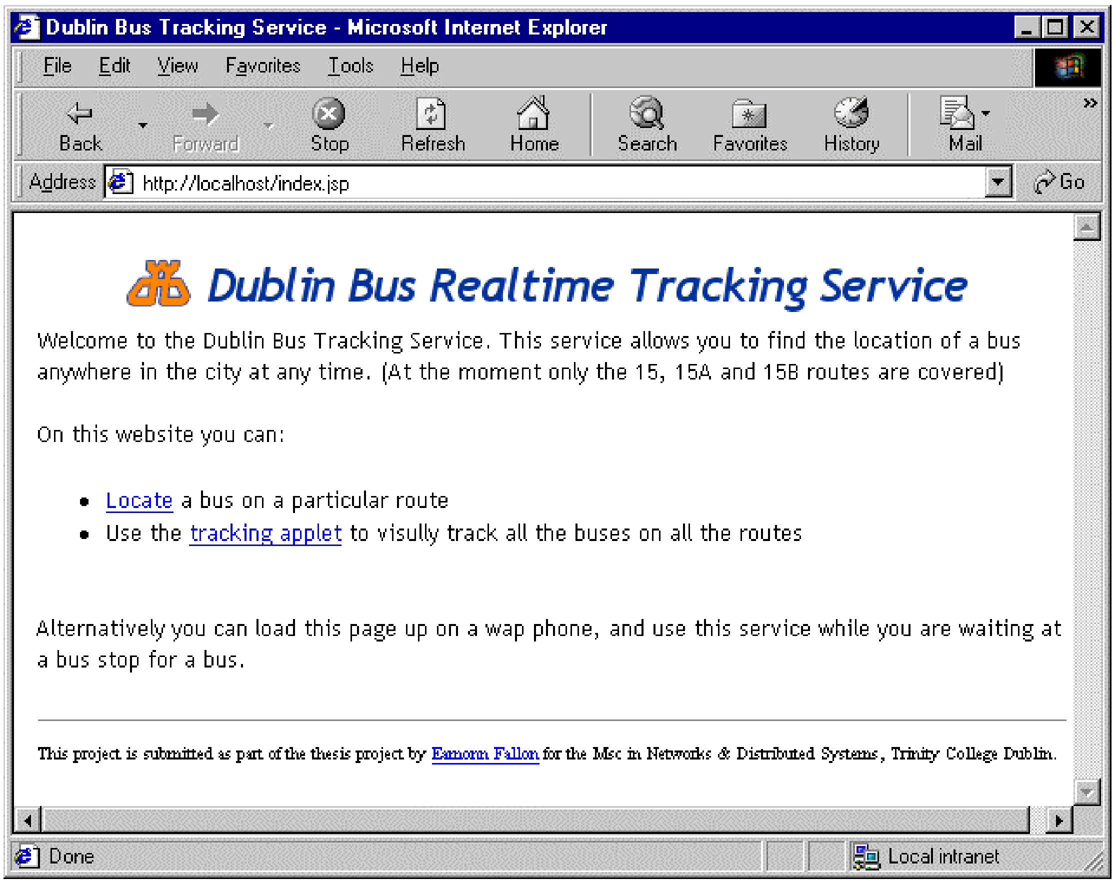
- The controller searches for particular patterns in the user-agent and accept HTTP headers. If any of these patterns are found, then the requesting device is a WAP device. This device is then given the root JSP page (http://localhost/wap.jsp) in the WML user interface tree. For a list of these patterns see appendix C.

- If none of these patterns are found then the device is a web browser and it is given the root document in the HTML user interface tree - http://localhost/index.jsp

See appendix D for listing of the Tomcat configuration files.

5.4.2 HTML View

The root document in the HTML user interface tree is http://localhost/index.jsp. This is shown in figure 5.3.


Dublin Bus Tracking Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://localhost/index.jsp
Go
Done
Local intranet

Dublin Bus Realtime Tracking Service

Welcome to the Dublin Bus Tracking Service. This service allows you to find the location of a bus anywhere in the city at any time. (At the moment only the 15, 15A and 15B routes are covered)

On this website you can:

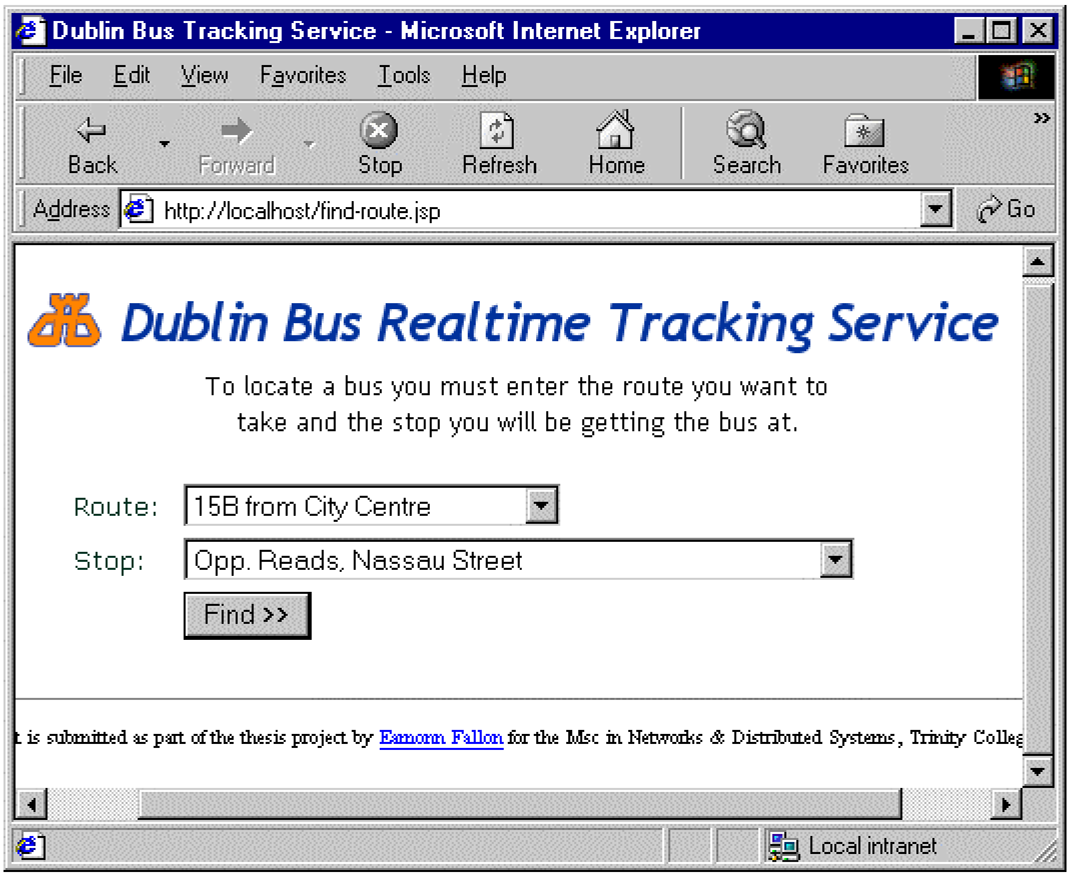
- Locate a bus on a particular route
- Use the tracking applet to visully track all the buses on all the routes

Alternatively you can load this page up on a wap phone, and use this service while you are waiting at a bus stop for a bus.

This project is submitted as part of the thesis project by Eamonn Fallon for the Msc in Networks & Distributed Systems, Trinity College Dublin.

Figure 5.3: Root document in the HTML user interface tree

From this page the user can click on *Locate a bus on a particular route* which provides a HTML form for querying the tracking service. They can also click on *tracking applet* which provides an interactive applet so the user can see visually the entire internal state of the tracking service. Clicking on *Locate a bus on a particular route* displays the find-route.jsp document. This is shown in figure 5.4.


Dublin Bus Tracking Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/find-route.jsp

Dublin Bus Realtime Tracking Service

To locate a bus you must enter the route you want to take and the stop you will be getting the bus at.

Route:	15B from City Centre
Stop:	Opp. Reads, Nassau Street

Find >>

t is submitted as part of the thesis project by Eamonn Fallon for the Msc in Networks & Distributed Systems, Trinity Colleg

Local intranet

Figure 5.4: HTML Form for querying Tracking Service

On this page the user selects the route they want to travel on and the bus stop they are waiting at. When the user clicks on *Find>>* the JSP page uses the WebServerProxy to contact the TrackingService. In the example shown in figure 5.4 the tracking service will return a list of location tables for the 15B from City Centre route. When the WebServerProxy receives the list of location tables, it formats the data for an HTML device. The following content is returned to the JSP page:

```
The next bus is scheduled to leave the terminus at 12:35<br>
<br>
<br>
There is a bus between the stops Opp. Reads, Nassau Street and Kildare
Street<br>
The bus has already passed your current location, it is 1 stop ahead
of your stop.<br>
There are 60 seats left.<br>
```

```
<br>
<br>
There is a bus between the stops Opp. Lakelands Park, Templeogue Road
and Opp. Terenue College Entrance, Templeogue Road<br>
The bus has already passed your current location, it is 22 stops ahead
of your stop.<br>
There are 17 seats left.<br>
<br>
<br>
```

5.4.3 WML View

The root document in the WML user interface tree is http://localhost/wap.jsp, this is shown in figure 5.5.

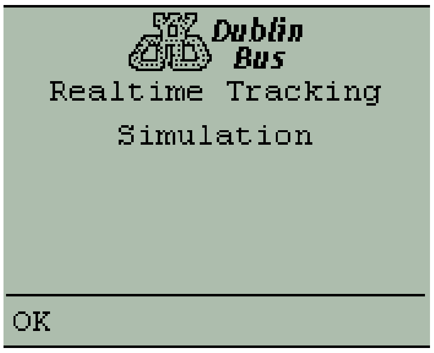


Figure 5.5: Root document in WML user interface tree

After 10 seconds this page automatically forwards to http://localhost/wap/index.jsp shown in figure 5.6. On this page the user selects the route they want to travel on.

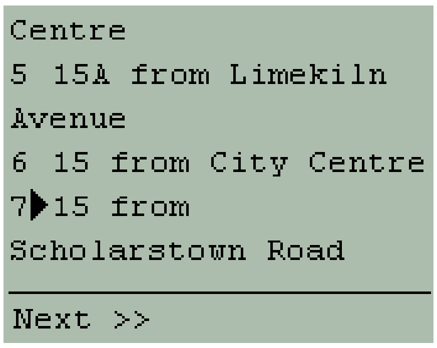

Centre
5 15A from Limekiln
Avenue
6 15 from City Centre
7▶15 from
Scholarstown Road

Figure 5.6: WML page for selecting bus route

In the page shown in figure 5.6 the *15 from Scholarstown Road* route is choosen. The next page (figure 5.7) allows the user choose one of the bus stops on this route. This is the bus stop they are waiting at. In the example shown the bus stop choosen is in templeogue village.

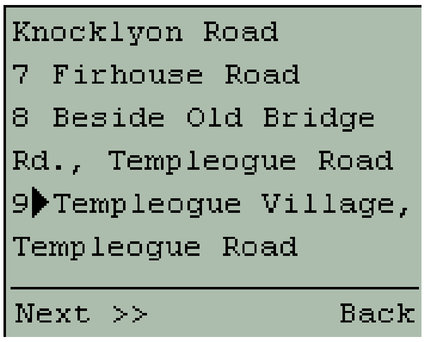

Knocklyon Road
7 Firhouse Road
8 Beside Old Bridge
Rd., Templeogue Road
9▶Templeogue Village,
Templeogue Road

Figure 5.7: WML page for selecting bus stop

When the user clicks on *Next>>* the JSP page uses the WebServerProxy to contact the TrackingService. For this example the tracking service will return a list of location tables for the 15 from Scholarstown Road route. When the WebServerProxy receives the list of location tables from the tracking service, it then formats the data for a WML device. The WebProxyServer knows that the calling page is a WML page because it was passed the `WebProxyServer.WML` constant by the JSP page. The following content is returned to the JSP page:

```
The next bus is scheduled to leave the terminus at 13:05
<br/>
<br/>
There is a bus between the stops Terminus, Scholarstown Road and
Beverly Drive, Scholarstown Road, 7 stops from your current
location.<br/>
The bus will arrive in 3 minutes, 45 seconds, There are 68 seats
left.<br/>
<br/>
<br/>
There is a bus between the stops Opp. Spar, Rathgar Road and Near
Esso, Rathgar Road. The bus has already passed your current location,
it is 12 stops ahead of your stop.<br/>
There are 3 seats left.<br/>
<br/>
<br/>
```

As can be seen here, a bus is due to arrive at the user's bus stop in 3 minutes and 45 seconds. There are 68 seats left. He is also told the another bus will leave the terminus 13:05. This location sensitive information is invaluable to the passenger. It allows them make informed travel decisions and takes the uncertainty out of using public transport.

5.4.4 Applet View

The applet view is actually part of the HTML user interface tree. It provides a visually rich and interactive environment for querying the underlying model i.e. the tracking service. The applet is shown in figure 5.8

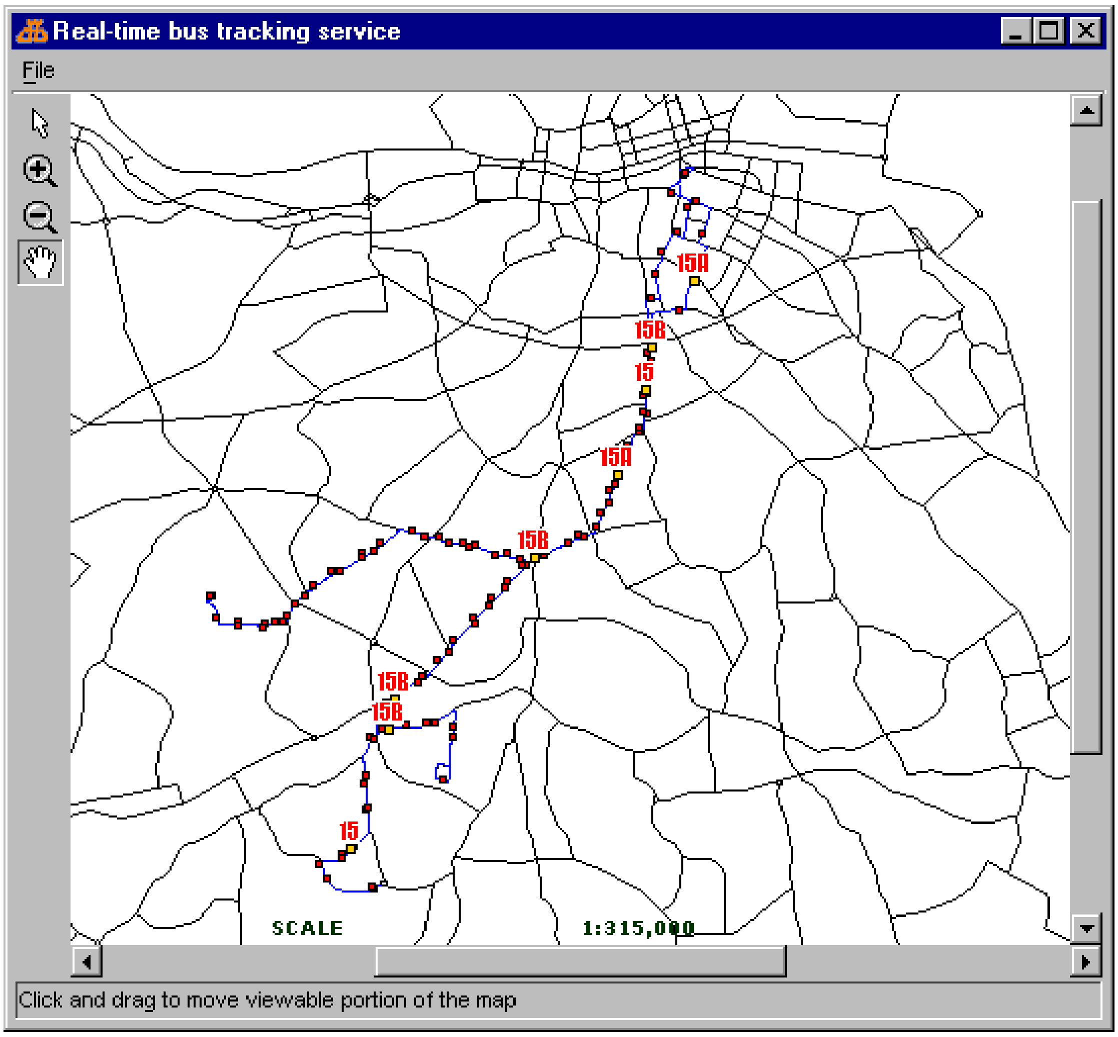

Real-time bus tracking service
File
15A
15B
15
15A
15B
15B
15B
15
SCALE
1:315,000
Click and drag to move viewable portion of the map

Figure 5.8: Applet showing real-time location of all buses

The applet uses a request/response architecture just like the other views. An updater thread runs in the background and periodically requests the location tables for all the buses in the network. By clicking on *File* and *Settings*, the user can change the frequency that the updater thread requests the location tables from the tracking service (see figure 5.9).

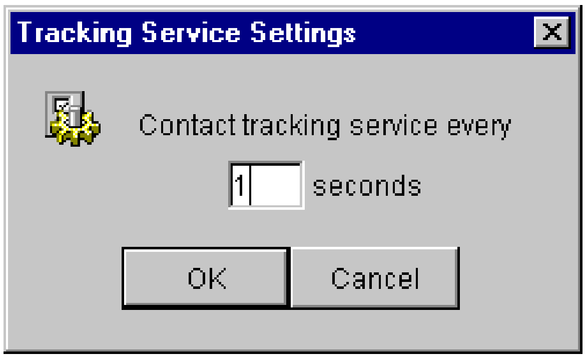
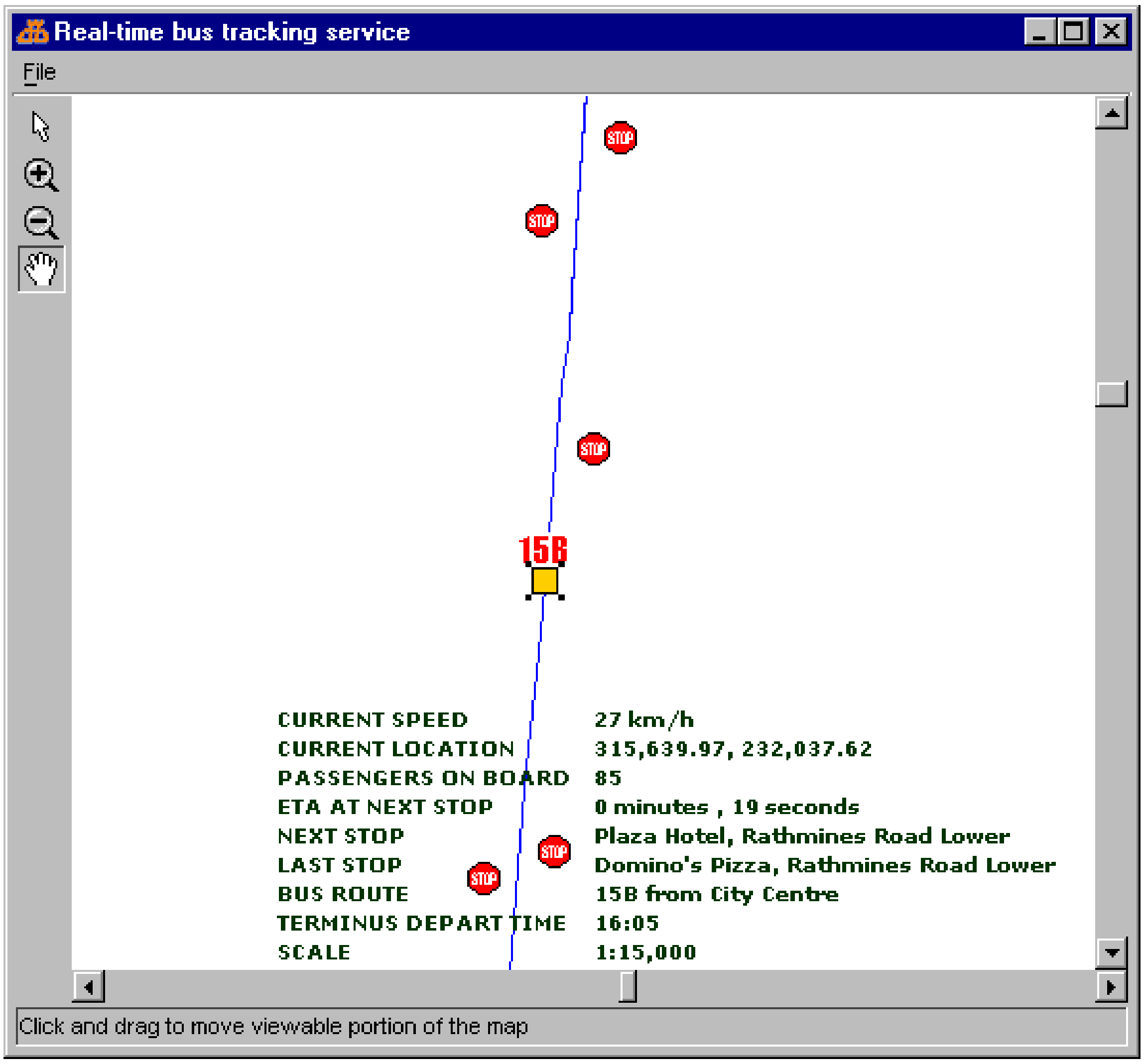


Figure 5.9: Applet Update Frequency Dialog

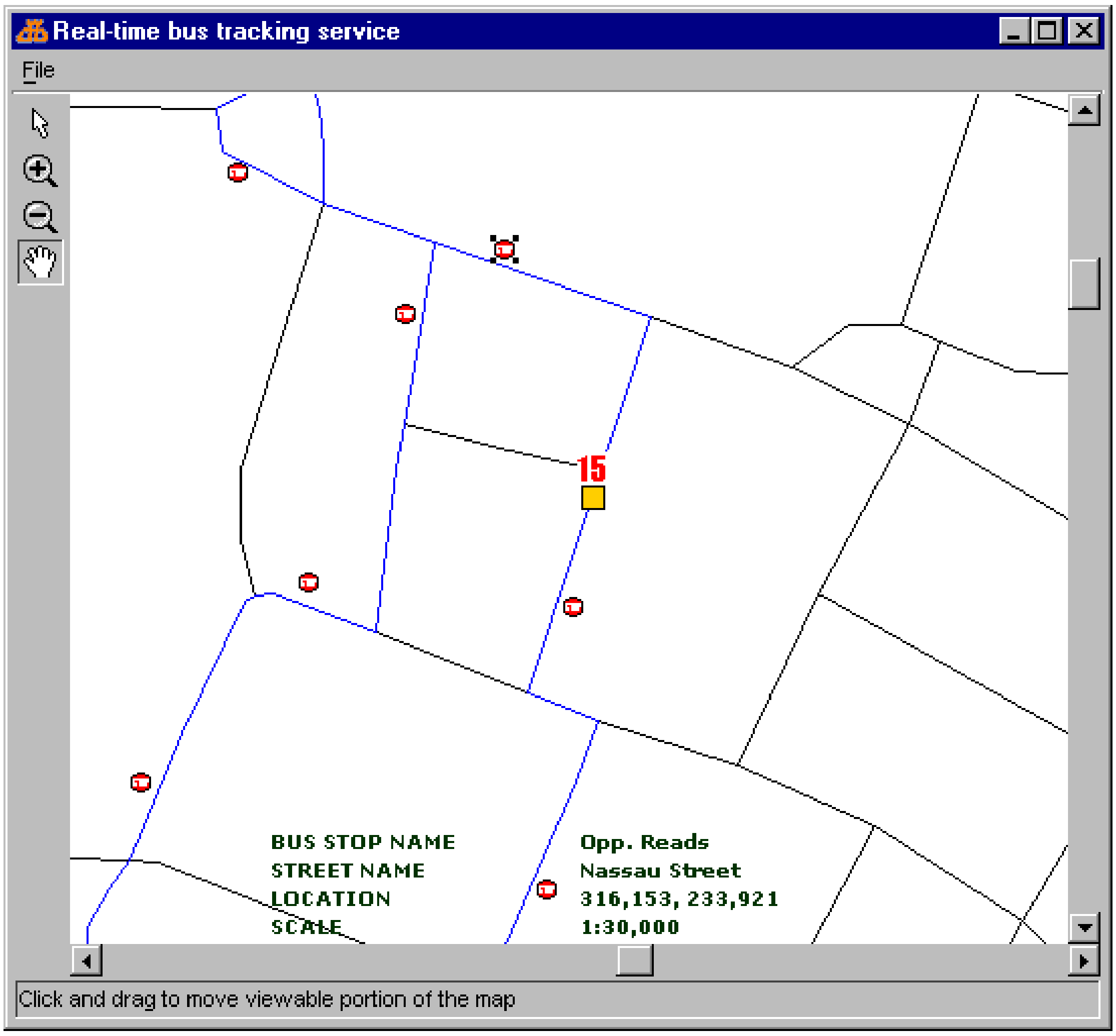
The applet provides a rich interface with intuitive querying of data. The applet displays the entire road and route network data. The applet then places bus icons on top of the route, according to the data stored in the location tables. The user can then dynamically query the attributes of a bus by just clicking on the icon (see figure 5.10). The information provided is location sensitive and gives the estimated time of arrival at the next stop.


Real-time bus tracking service
File
STOP
15B
Click and drag to move viewable portion of the map

CURRENT SPEED	27 km/h
CURRENT LOCATION	315,639.97, 232,037.62
PASSENGERS ON BOARD	85
ETA AT NEXT STOP	0 minutes , 19 seconds
NEXT STOP	Plaza Hotel, Rathmines Road Lower
LAST STOP	Domino's Pizza, Rathmines Road Lower
BUS ROUTE	15B from City Centre
TERMINUS DEPART TIME	16:05
SCALE	1:15,000

Figure 5.10: Querying the attributes of a single bus

The user can also query static map data object. By clicking on the bus stop icon, the user is presented with the name of the bus stop, its location and the street it is on


Real-time bus tracking service
File
15
Click and drag to move viewable portion of the map

BUS STOP NAME Opp. Reads
STREET NAME Nassau Street
LOCATION 316,153, 233,921
SCALE 1:30,000

Figure 5.11: Querying the attributes of a bus stop

As well as the intuitive data querying, the user is given zoom and drag capabilities for easy navigation around the map.

CHAPTER 6

Evaluation & Future Work

From the point of view of architecture and systems design, this thesis contributed to the state of the art and compares favourably with existing systems. From an implementation perspective it is difficult to compete with multimillion-pound passenger information systems. While issues such as scalability and fault tolerance have been dealt to a certain degree within the implementation, in a real-world environment, systems need to be tested rigorously under multiple load scenarios. These tests are needed to expose weaknesses in the system, which may not be otherwise obvious. Therefore in terms of scalability and stability, the Dublin bus tracking service can not be compared with existing passenger information systems such as project 423 and Superoute 66.

This chapter contrasts the Dublin bus tracking service with two other state-of-the-art passenger information systems, project 423 and Superoute 66. Finally a quick summary of some possible future work is given that would build upon the Dublin bus tracking service prototype and architecture.

6.1 Evaluation of Dublin Bus Tracking Service

This section contrasts the Dublin bus tracking service prototype with the two passenger information systems introduced in chapter 3. A look is also taken at the issues that were not fully addressed by the prototype.

6.1.1 DBTS versus Project 423 & Superoute 66

The table below contrasts the features of the Dublin bus tracking service architecture and the implementation with the Helsinki-based project 423 system and U.K.-based Superoute 66.

PIS Systems	DBTS architecture	DBTS implementation	Project 423	Superoute 66
Web Browser support	YES	YES	Limited to staff	NO
WAP-enabled Phone support	YES	YES	NO	YES
Voice-based Phone support	YES	NO	NO	YES
At-Stop Display	YES	N/A	YES	YES
On-board display	YES	N/A	YES	NO
Driver support system	YES	NO	YES	NO
Interactive Java Applet	YES	YES	NO	YES
Positioning infrastructure	M-AVL	Simulation	GPS & Odometer & Door Sensors	GPS & Odometer
Low infrastructure costs	YES	YES	NO	NO
Communications infrastructure	Public Cellular Radio network	N/A	Dedicated Radio Network	Dedicated Radio Network
Integrated with traffic light network	Limited	N/A	YES	NO

6.1.2 Issues not addressed

There are many features specified in the design that were not implemented as can be seen in the table above. These features could not be implemented because they rely on underlying infrastructure. However below are some software issues that were not properly addressed by the Dublin bus tracking service prototype.

Scalability

Scalability is not adequately designed for in the implementation. A possible remedy to this is described in section 6.2.1.

Voice Interface

Although specified in the architecture, a voice interface was not implemented in the prototype. This is an area for future work.

XML data exchange format

In the architecture described in chapter 4, the HTTP document server could also function as a Content Provider Centre. It could provide information to other VASPs to integrate into their systems. Other VASPs need to be able to access content provided by the Dublin Bus tracking service in a structured way. Since the VASP has no need for formatting information, the best way to provide the content to a VASP is as an XML data exchange format.

6.2 Future work

While many features and additions can be made to the added to the Dublin bus tracking service, the most interesting short-term additions and improvements are mentioned below. Obviously the long-term goal is to produce a national intermodal passenger information that spans all modes of public and private transport. This goal is more hindered by organisational issues than technological ones. However when the systems become cheap enough the organisational issues will become much less important.

6.2.1 Scalability

In a production (i.e. real-world) environment, the most critical points of failure are the HTTP document server and the tracking service. Figure 6.1 suggests modifications that could be made to the architecture described in figure 4.6 to enable it to

- Scale linearly to cope with increased load
- Recover from failure of the tracking service

Implementing this architecture requires no modifications to the software to enable load balancing. Level-4 switches or IP load balancers are hardware-based network layer devices.

Using these switches, incoming requests are sent to the servers in the farm on a round robin basis. If a server fails it will not be sent any requests. Figure 4.6 shows how incoming HTTP requests can be sent to different HTTP document servers on a round-robin basis.
Clients of the HTTP Document Server are unaware that there is more than one HTTP Document Server. Each HTTP document server accesses the tracking service through a level-4 switch. Like its own clients, the HTTP document server is not aware that there is more than one tracking service. The real-time location feed is sent to a multicasting network device, which sends the feed to all tracking servers simultaneously. Each tracking service operates independently of the others.

If an HTTP document server fails and recovers, it has no impact on the system. This is because HTTP servers are stateless, which is not the case with the tracking service. In order to maintain consistent behaviour in a fault-tolerant scenario, modifications will need to be made to the tracking service software. A possible solution would be that when a tracking service fails and recovers, it requests the entire state of a neighbouring tracking service.

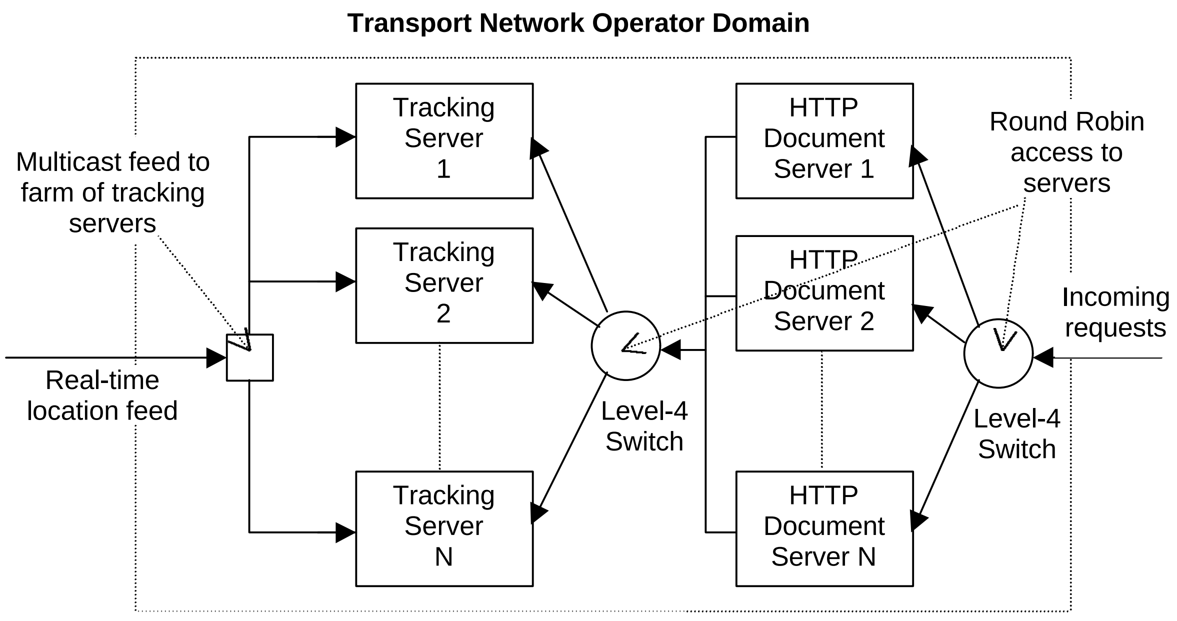


Figure 6.1: Enhanced Scalable Systems Architecture

6.2.2 Implementation of M-AVL

The M-AVL system could be integrated into the system without installing it on a network of buses. An interesting project would be to feed a real-time location feed from a group of individuals into the Dublin bus tracking service. In this case it would be a people tracking service. The concept is the same as tracking buses, except instead of positioning mobile devices installed on buses, individual’s personal mobile phones are being positioned.

Bibliography

[1] TS 22.071 V3.2.0: 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects; Location Services (LCS).

[2] FCC 0036: Federal Communications Commission: Revision of the Commission's Rules To Ensure Compatibility with Enhanced 911 Emergency Calling Systems 3rd Generation Partnership Agreement CC Docket No. 94-102, Forth Memorandum Opinion and Order
http://www.fcc.gov/Bureaus/Common_Carrier/Orders/2000/fcc00326.pdf

[3] 3rd Generation Partnership Agreement 18 January 2000
http://www.3gpp.org/About_3GPP/3gppagre.pdf

[4] U.S. Coastal Guard, Global Positioning System FAQ
http://www.navcen.uscg.mil/faq/gpsfaq.htm

[5] White House Press Release May 1 2000: Statement by the president regarding the United States decision to stop degrading global positioning system accuracy.
http://www.navcen.uscg.mil/news/archive/2000/May/SA.htm

[6] Andrew S. Tanenbaum; Prentice Hall, 1996 International Edition. *Computer Networks Ch2 p99*
ISBN 0-13-394248-1

[7] G. L. Turin, W. S. Jewell, T. L. Johnston, Simulation of Urban Vehicle-Monitoring Systems, IEEE Trans. on Vehicular Technology, Vol. VT-21, No. 1, pp 9-16, February 1972.

[8] F. Cristian, A probabilistic approach to distributed clock synchronization. *Proceedings 9th International Conference on Distributed Computing Systems*, June 1989

[9] Bibl, K and Reinisch B.W., The Universal Digital Ionosonde.
Radio Science.Vol. 13, No. 3, pp 519-530, 1978.

[10] PROMISE Project TR1043
Deliverable D3.1 “Specification of PROMISE Services”
Version 7, July 1996

[11] PROMISE Project TR1043
Deliverable D4.1 “Generic PROMISE System Architecture”
Version 2, August 1996

[12] BT Laboratories, Ipswitch: Using Telecommunications to Deliver Real-time Transport Information; “Superoute 66 Live” Trial Results
November 1999

[13] Michael E. Porter; Free Press 1998; *Competitive Strategy: Techniques for Analyzing Industries and Competitors*
ISBN 0-684-84148-7

[14] Goldberg, Adele & David Robson; Addison-Wesley, 1983 (reprinted 1985 with corrections). *Smalltalk-80: The Language and Its Implementation*
ISBN 0-201-11371-6.

[15] VoiceXML Forum, *Voice eXentensible Markup Language Specification version 1.00*
7 March 2000

[16] Ordanance Survay of Ireland, *Dublin City and District Street Guide 2nd Edition*
ISBN 0-904996-16-6

Appendix A: Database DDL

```
create table busroutes (
       busrouteID int NOT NULL,
       routeID int NOT NULL,
       busRouteName VARCHAR(80) NOT NULL,
       busRouteNumber VARCHAR(10) NOT NULL,
       stopsID int NOT NULL,
       UNIQUE INDEX pk_busroutes (busrouteID)
);

create table busroutestops (
       stopsID int NOT NULL,
       stopIndex int NOT NULL,
       stopID int NOT NULL,
       UNIQUE INDEX pk_busroutestops (stopsID,stopIndex)
);

create table busstops (
       stopID int NOT NULL,
       stopName VARCHAR(100) NOT NULL,
       streetID int NOT NULL,
       x int NOT NULL,
       y int NOT NULL,
       UNIQUE INDEX pk_busstops (stopID)
);

create table timetable (
       busrouteID int NOT NULL,
       departday VARCHAR(9) NOT NULL,
       departtime TIME NOT NULL,
       INDEX busrouteidindex (busrouteID),
       INDEX timeindex (departtime)
);

create table streets (
       streetID int NOT NULL,
       streetName VARCHAR(100),
       UNIQUE INDEX pk_streets (streetID)
);

create table roadsegments (
       segmentID int NOT NULL,
       nodeIndex int NOT NULL,
       streetID int NOT NULL,
       x REAL NOT NULL,
       y REAL NOT NULL,
       UNIQUE INDEX pk_roadsegments (segmentID,nodeIndex)
);

create table routes (
       routeID int NOT NULL,
       routeIndex int NOT NULL,
       segmentID int NOT NULL,
       reversed ENUM('false','true') NOT NULL,
       UNIQUE INDEX pk_routes (routeID,routeIndex)
);
```

Appendix B: Bus Configuration File

This is a sample bus configuration file (bus.properties). It defines attributes that the simulator applies to all buses driving on a route.

```
//Maximum random speed deviation of the bus (km/h).
//A random speed deviation is applied every update-thread-frequency seconds.
max-speed-deviation=2

//This is the fastest speed that a bus can travel at.
speed-limit=48.2

//This is the average speed of a bus in km/h.
//When a bus leaves a stop is will initially travel at this speed.
avg-speed = 30

//This is the time (in seconds) between sending update messages to the
//tracking service
update-tracking-service-frequency=1

//This is the time (in seconds) a bus thread sleeps before waking up and
//recalculating its location
update-thread-frequency=1

//The maximum of people that will fit on a bus
max-passengers = 85

//The max size (as percentage of max-passengers) of the boarding envelope
max-boarding-percentage = 16

//The random max percentage deviation from the linearly decreasing number of
//passengers boarding at each stop.
max-boarding-deviation = 4

//The max size (as percentage of max-passengers) of the deboarding envelope.
max-deboarding-percentage = 15

//The random max percentage deviation from the linearly increasing number of
//passengers deboarding at each stop.
max-deboarding-deviation = 4

//The average time it takes 1 person to get on a bus
seconds-to-board = 2
```

```
//The average time it takes 1 person to get off a bus
seconds-to-deboard = 1
```

Appendix C: User-Agent Patterns

This appendix specifies the algorithm used to identify WAP-enabled devices.

1) If the HTTP `accept` header (converted to uppercase) contains VND.WAP.WML then it is a WAP device.
2) Otherwise if the HTTP `user-agent` header converted to uppercase contains any of the following patterns it is a WAP device

```
NOKI
ERIC
WAPI
MC21
AUR
R380
UP.B
WINW
UPG1
UPSI
QWAP
JIGS
JAVA
ALCA
MITS
MOT-
MY S
WAPJ
FETC
ALAV
WAPA
```

3) Otherwise it is a HTML web browser

Appendix D: Tomcat Configuration Files

The following lines are added to the server.xml file.

```
<!—-This directive instructs tomcat to listen on TCP port 80 -->
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
    <Parameter name="handler"
      value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
    <Parameter name="port" value="80"/>
</Connector>


<!—-Setting path to "" makes this the default web application-->
<Context path="" docBase="webapps/sim" debug="0" reloadable="true" />
```

This is the entire web application descriptor file web.xml:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.2//EN" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

    <display-name>Dublin Bus Tracking Simulation</display-name>
    <description>
      Provides a web interface to the Dublin Bus Tracking Service
    </description>

    <!—-gives the alias device controller to the Controller servlet -->
    <servlet>
        <servlet-name>
            devicecontroller
        </servlet-name>
        <servlet-class>
            webserver.Controller
        </servlet-class>
    </servlet>
    <!—-Makes the Controller servlet handle all urls that have the *.control
        pattern in them -->
    <servlet-mapping>
        <servlet-name>
            devicecontroller
        </servlet-name>
        <url-pattern>
            *.control
        </url-pattern>
    </servlet-mapping>


    <!—- Makes the default file in a directory to be device.control -->
    <welcome-file-list>
             <welcome-file>
            device.control
        </welcome-file>
    </welcome-file-list>

</web-app>
```