

University of New Hampshire
University of New Hampshire Scholars' Repository

Master's Theses and Capstones

Student Scholarship

Fall 2009

APCO project 25 wireless data services over land mobile radio channel for smaller law enforcement agencies

Ivan Elhart

University of New Hampshire, Durham

Follow this and additional works at: <https://scholars.unh.edu/thesis>

Recommended Citation

Elhart, Ivan, "APCO project 25 wireless data services over land mobile radio channel for smaller law enforcement agencies" (2009).
Master's Theses and Capstones. 474.
<https://scholars.unh.edu/thesis/474>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.

**APCO PROJECT 25 WIRELESS DATA SERVICES OVER LAND
MOBILE RADIO CHANNEL FOR SMALLER LAW
ENFORCEMENT AGENCIES**

BY

IVAN ELHART

B.S., University of Novi Sad, 2006

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Electrical Engineering

September, 2009

UMI Number: 1472059

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

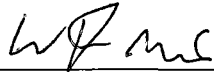
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.



UMI Microform 1472059
Copyright 2009 by ProQuest LLC
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

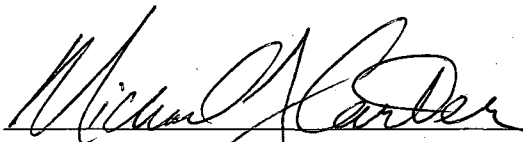
This thesis has been examined and approved.



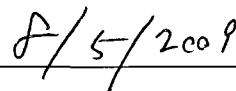
Thesis Director, Dr. W. Thomas Miller, III
Professor of Electrical Engineering



Dr. Andrew L. Kun
Associate Professor of Electrical Engineering



Dr. Michael J. Carter
Associate Professor of Electrical Engineering



Date

ACKNOWLEDGMENTS

First of all, I would like to thank my thesis adviser Dr. Thomas W. Miller, III for his constant guidance, patience, and support throughout the course of this work.

I would also like to thank my academic adviser Dr. Andrew L. Kun for giving me the opportunity to obtain my Master's degree at the University of New Hampshire and for his constant support throughout the course of my research.

I would like to thank Dr. Michael J. Carter for his help during my graduate studies and for serving on my thesis committee.

I would like to thank my loving wife, Isidora, and my parents for their love and support during many years of my education.

Last, but by no means least, I would like to thank all Project54 colleagues for helping me throughout the course of my research.

This work was supported by the U.S. Department of Justice under grants 2005CKWX0426 and 2006DDBXK099.

TABLE OF CONTENTS

| | |
|---|-------------|
| ACKNOWLEDGMENTS | iii |
| TABLE OF CONTENTS | iv |
| LIST OF TABLES | vii |
| LIST OF FIGURES | viii |
| LIST OF ACRONYMS | xi |
| ABSTRACT | xii |
| CHAPTER I INTRODUCTION | 1 |
| Problem Description..... | 2 |
| Goal of the Thesis | 5 |
| Approach | 6 |
| Thesis Overview..... | 6 |
| CHAPTER II BACKGROUND | 9 |
| Invisible Communication | 10 |
| Intra-Vehicular Systems | 11 |
| Vehicle to Infrastructure Networks | 13 |
| Land Mobile Radio Communication for Law Enforcement | 18 |
| <i>Project 25</i> | <i>18</i> |

| | |
|---|-----------|
| <i>Project 25 Data Communication</i> | 21 |
| CHAPTER III SOFTWARE DEFINED APCO PROJECT 25 DATA BASE | |
| STATION | 24 |
| System overview | 24 |
| Reception and Decoding of Digital Data Packets | 28 |
| <i>Status Symbols</i> | 30 |
| <i>Frame Synchronization Word</i> | 31 |
| <i>Network Identifier</i> | 32 |
| <i>Inverse Interleaving</i> | 35 |
| <i>Data Error Correction Decoding</i> | 36 |
| <i>Data Header Block Format</i> | 41 |
| <i>Unconfirmed Packet Data Block Format</i> | 49 |
| <i>Confirmed Packet Data Block Format</i> | 51 |
| <i>Response Block Format</i> | 54 |
| The Data Base Station Application | 55 |
| <i>The Base Station Main Thread</i> | 56 |
| <i>The Main CAI Packet Input Thread</i> | 59 |
| <i>The Main IP Packet Input Thread</i> | 63 |
| CHAPTER IV TESTING | 67 |

| | |
|--|------------|
| Tests with a Single Mobile Data Client | 69 |
| Tests with Multiple Mobile Data Clients | 79 |
| Voice Priority Assurance | 84 |
| <i>Project 25 Vocoder</i> | 84 |
| <i>Tests with Parallel Voice and Data Communication</i> | 90 |
| <i>Tests with Simultaneous Voice and Data Transmissions from a Single Client</i> | 94 |
| CHAPTER V CONCLUSION | 96 |
| REFERENCES | 100 |
| APPENDIX A CUSTOM KENWOOD RADIO INTERFACE CABLE | 104 |
| APPENDIX B KENWOOD TX 7180 CODE PLUG | 105 |

LIST OF TABLES

| | |
|---|----|
| Table 1 - C4FM Frequency Deviations | 28 |
| Table 2 - Status Symbol Codes..... | 31 |
| Table 3 - Frame Synchronization Word sequence..... | 32 |
| Table 4 - Network Identifier | 32 |
| Table 5 - Data Unit Identifier Values | 33 |
| Table 6 - BCH Code Generation Matrix..... | 34 |
| Table 7 - Interleave Table..... | 35 |
| Table 8 - Rate 1/2 Trellis State Transition Table..... | 38 |
| Table 9 - Rate 3/4 Trellis State Transition Table..... | 38 |
| Table 10 - Constellation to Dibit Pair Mapping..... | 39 |
| Table 11 - SAP Identifier Values..... | 43 |
| Table 12 - Response Packet Class, Type, and Status Specification..... | 47 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1 - Current state of communication in local NH departments..... | 4 |
| Figure 2 - A typical voice communication system | 25 |
| Figure 3 – Project 25 Data Base Station designed in this thesis..... | 26 |
| Figure 4 - The Data Base Station Architecture | 27 |
| Figure 5 - CAI to IP packet Conversion | 29 |
| Figure 6 - Trellis Encoder Overview | 37 |
| Figure 7 - Trellis Encoder Block Diagram | 38 |
| Figure 8 - Unconfirmed Data Packet Header Block | 41 |
| Figure 9 - Confirmed Data Packet Header Block | 42 |
| Figure 10 - Acknowledgment Data Packet Header Block | 42 |
| Figure 11 - Unconfirmed Last Data Block | 49 |
| Figure 12 - Confirmed Data Packet Data Block | 51 |
| Figure 13 - Confirmed Data Packet Last Data Block | 53 |
| Figure 14 - Response Packet Data Block..... | 55 |
| Figure 15 - Flow Diagram of the Base Station Main Thread | 56 |
| Figure 16 - The IP Reflector Architecture | 58 |
| Figure 17 - Flow Diagram of the Main CAI Packet Input Thread..... | 59 |
| Figure 18 - Client Table..... | 60 |
| Figure 19 - Flow Diagram of the Client Threads..... | 61 |
| Figure 20 - Flow Diagram of the Client CAI Packet Input Thread | 62 |
| Figure 21 - Flow Diagram of the Main IP Packet Input Thread..... | 64 |

| | |
|---|-----|
| Figure 22 - Flow Diagram of the Client IP Packet Input Thread..... | 65 |
| Figure 23 - End-to-end Initial Testing Setup | 69 |
| Figure 24 - A segment of a Client Log File | 72 |
| Figure 25 - A segment of a Server Log File | 73 |
| Figure 26 - A segment of a Data Radio Log File..... | 74 |
| Figure 27 - A segment of a Data Radio Receiver Log File | 75 |
| Figure 28 - A segment of a Data Radio Transmitter Log File | 77 |
| Figure 29 - An Example of a P25 CAI Waveform | 78 |
| Figure 30 - An example of Data Radio In/Out Log File..... | 82 |
| Figure 31 - Block Diagram of the Project 25 Vocoder..... | 85 |
| Figure 32 - Project 25 Voice Recording Setup | 86 |
| Figure 33 - Comparison of Played and Recorded Signals ("Troop B Boston") in the Time and Frequency Domains | 88 |
| Figure 34 - Comparison of Played and Recorded 300 Hz Sine Wave in the Time and Frequency Domains | 89 |
| Figure 35 - Parallel Voice and Data Communication over the same Radio Channel..... | 90 |
| Figure 36 - An Example of a Recorded Signal, an Amplitude Analysis, and a Cross- correlation | 92 |
| Figure 37 - The Final Testing Setup with a Single Mobile Client..... | 95 |
| Figure 38 - Custom Kenwood Radio Interface Cable..... | 104 |
| Figure 39 - Test Channel Frequency Settings..... | 105 |
| Figure 40 - Radio COM Port Settings..... | 106 |

| | |
|--|-----|
| Figure 41 - Radio Modulation Line Settings | 106 |
|--|-----|

LIST OF ACRONYMS

| | |
|--------------|--|
| APCO | Association of Public Safety Communication Officials |
| P25 | APCO Project 25 |
| CAI | Common Air Interface |
| IP | Internet Protocol |
| VHF | Very High Frequency (radio frequency range from 30 to 300 MHz) |
| FM | Frequency Modulation |
| WiFi | Wireless Local Area Networks based on IEEE 802.11 |
| LMR | Land Mobile Radio |
| NHDS | New Hampshire Department of Safety |
| VPN | Virtual Private Network |
| GPS | Global Positioning System |
| TIA | Telecommunications Industry Association |
| IMBE | Improved Multi-Band Excitation |
| FCC | Federal Communications Commission |
| FDMA | Frequency Division Multiple Access |
| TDMA | Time Division Multiple Access |
| C4FM | Constant Envelope 4-Level FM |
| CQPSK | Compatible Differential Offset Quadrature Phase Shift Keying |

ABSTRACT

APCO PROJECT 25 WIRELESS DATA SERVICES OVER LAND MOBILE RADIO CHANNEL FOR SMALLER LAW ENFORCEMENT AGENCIES

by

Ivan Elhart

University of New Hampshire, September, 2009

Digital data messages are very important in modern communication systems and advanced mobile data technologies have opened the door to a wide range of applications and services in the public safety environment. Still, the availability of mobile data services among public safety agencies is hampered by two issues of the implementation of data communication: the reliability of commercial data services and the high cost of the equipment needed to support mixed voice and data transmissions over private land mobile radio channels.

This thesis describes the design and development of an inexpensive Software Defined APCO Project 25 Data Base Station that allows smaller law enforcement agencies to enable data services in their cruisers in a cost effective way. The data base station is comprised of a standard PC interfaced to a commercial analog VHF FM transceiver via a commercial PC sound card. The base station is compliant with commercial P25 digital mobile radios and operates in parallel to commercial P25 digital voice communications equipment.

CHAPTER I

INTRODUCTION

The traditional way of communicating in a mobile environment is to use voice messages. For example, in applications such as fleet management, the driver uses a mobile radio or cellular phone to provide the vehicle's current position to a command center. Based on the current position of the vehicle the command center may modify the driver's assigned tasks. However, operating a manual radio interface and using voice messages for communication require the driver's attention and time which can degrade driving performance [1]. On the other side, some communication can be automated, so the driver only needs to pay attention to the result. For example, the vehicle can automatically update the current position every few seconds to the command center and an in-vehicle system can inform the driver only if there are updated or new tasks. The most important part in supporting automated communication is the utilization of digital data messages.

Today, many modern communications systems support the access, storage, and manipulation of desired information utilizing digital data messages. Those systems allow for a wide range of data services along with traditional voice communication. A well known example of such a system is a cellular phone network that supports both a Short Message Service and a Multimedia Messaging Service. In addition, advanced cellular

data technologies (e.g. General Packet Radio Service) that use Internet Protocol (IP) for the routing and exchanging of data packets have opened the door to broad IP based applications and services such as WWW, email, navigation, and even TV for a mobile environment [2].

There are several other wireless communication technologies which are often used for sending and receiving data messages in mobile settings. These technologies are: Wi-Fi and Land Mobile Radio (LMR). More often, the combination of commercial wireless and wired networks has been used for various public services ranging from fleet management and telemetry to providing additional information to users.

Problem Description

Commercial public data services can be of benefit to public safety agencies in supporting, organizing and tracking their personnel on patrol. However, these services are not widespread in the law enforcement community. The availability of mobile data services among public safety agencies is hampered by two issues of the implementation of data communications: the reliability of commercial data services and the high cost of the equipment necessary to support mixed voice and data transmissions over private LMR channels.

First, mission critical operations cannot rely on commercial data services unless they meet strict requirements of availability, survivability, security, and quality of service. Even with a vision that commercial systems can augment and may completely

replace public safety networks in the future, natural disasters (floods, hurricanes, and earthquakes) or terrorist attacks can compromise the commercial systems' ability to operate. This is not acceptable for mission critical operations [3].

Second, first responders primarily depend on secure, agency operated LMR communication and the spectrum reserved by the Federal Communication Commission for public safety use. LMR systems were initially designed to operate in analog mode and to provide only voice communication. Later, the analog systems were replaced with digital ones by implementing standards for digital communication. A set of common technical standards for digital communication, known as the Association of Public-Safety Communication Officials (APCO) Project 25 (P25), added data transmissions to public safety LMR channels [4]. This new data functionality introduced a new piece of equipment to the LMR system called a Data Base Station. The commercially available P25 data base stations are usually designed to support a large number of mobile radios and operate on multiple channels in either conventional or trunking configurations. This complexity raises the cost of such equipment and its installation. This complexity is not needed by smaller public safety agencies which require only a single conventional data radio channel.

For example, the New Hampshire Department of Safety (NHDS) was among the first agencies to implement a statewide P25 radio infrastructure that supports mixed voice and data communication. The NHDS directly benefits from the system by supporting multiple state agencies and processing approximately 20,000 data queries per month over

the same LMR channels used for voice communication. Systems like the one used by the NHDS may cost hundreds of thousands of dollars, which is not affordable for smaller local agencies due to their limited budgets.

Presently, nearly all local police departments in NH are equipped with P25-compliant mobile radios capable of data communication. However, the local departments do not utilize the data portion of the standard because they lack the expensive data capable base stations. Therefore, the departments use only voice communication to provide their cruisers with information on patrol (e.g. vehicle and driver record information). This current state of communication in local police departments is illustrated in Figure 1.

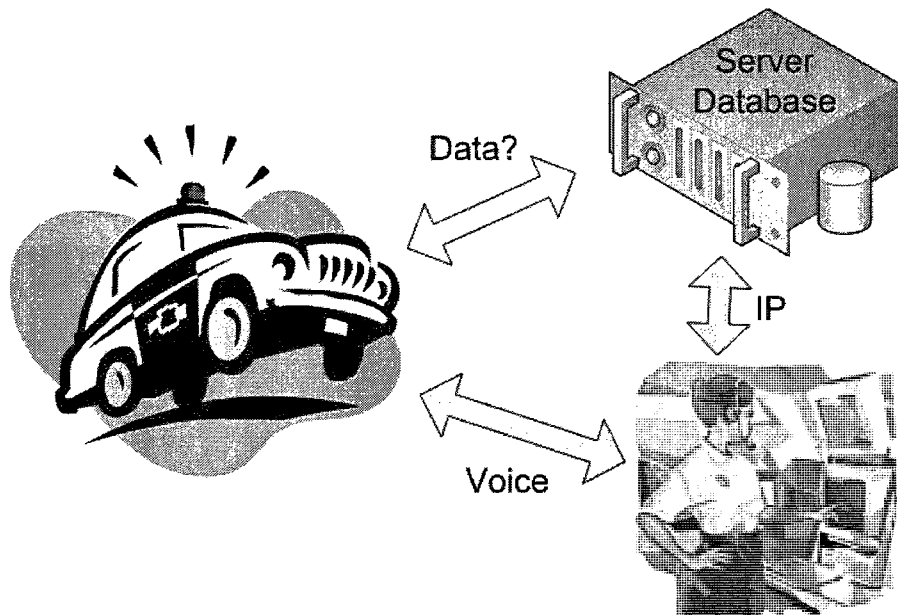


Figure 1 - Current state of communication in local NH departments

Goal of the Thesis

The goal of this work was to develop a system that would allow smaller law enforcement agencies to enable wireless data services in their cruisers in a cost effective way and with a high level of reliability. This goal was achieved by designing and implementing an inexpensive software defined APCO P25 data base station.

The software defined data base station is comprised of a standard PC, with a standard network interface, interfaced to a commercial analog VHF FM transceiver via a commercial PC sound card. The base station logic and the digital baseband modulation and demodulation are implemented in software on the PC. The analog radio performs the RF modulation and demodulation. The data base station is capable of operating in parallel to preexisting P25 digital voice communication lines involving no additional communication equipment at headquarters. The compliance with commercial P25 digital mobile radios, which can be found in almost all local departments in New Hampshire, further reduces the cost of providing data services to vehicles. The existing digital mobile radios can be used for both voice and data signaling without requiring additional in-vehicle wireless communication devices.

The base station developed in this thesis can be used in local police departments with several cruisers for data communication between headquarters and mobile units on patrol. The officers would use the data channel in combination with the Project54 system to query remote vehicle and driver record databases [5]. In combination with the Project54 speech user interface, the textual record queries should provide a safer way to

obtain information while driving [1] [6]. Furthermore, the system could be extended to support data services such as fleet management and vehicle telematics [7] [8].

Approach

In order to accomplish the goal of this thesis and provide vehicular wireless data services to smaller law enforcement agencies, a series of four steps were proposed. The first step was to extend the software defined APCO P25 Data Transmitter, developed by Eric Ramsey [9], to support the confirmed type of transmission. The second step was to implement a software defined APCO P25 Data Receiver capable of handling both confirmed and unconfirmed types of communication. The third step was to incorporate the data transmitter with the data receiver and implement APCO P25 data base station logic with collision avoidance and retransmission mechanisms. The fourth and final step was to test the APCO P25 Data Base Station according to various stages of P25 compliance, to test its interoperability with commercial P25 compliant digital radios, and to ensure voice priority over data messages.

Thesis Overview

This thesis is organized into five chapters and two appendices.

Chapter I, Introduction, gives an introduction to the topic of information sharing in a mobile environment utilizing digital data messages. First, the chapter introduces data communications and discusses its potential benefits. Second, it provides an overview of the current problems of the implementation of data communications for smaller public safety departments. Finally, it states the goal and presents the approach taken in this work.

Chapter II, Background, gives an overview of the commercial and research projects in the field of vehicle to infrastructure communications. The chapter starts with Weiser's vision of ubiquitous computing and invisible communication. Then, it discusses projects of intra-vehicular and vehicle to infrastructure networks. Lastly, it gives an overview of a set of digital standards for Land Mobile Radio communications, called APCO Project 25.

Chapter III, Software Defined APCO Project 25 Data Base Station, describes the design and implementation of a software defined P25 data base station. The first part gives an overview of a typical P25 system and presents the architecture of the base station. The second part talks about the reception and decoding of P25 Common Air Interface packets. The last part of the chapter presents the software application and describes the data base station logic.

Chapter IV, Testing, describes the precise and methodological testing procedure performed in the laboratory. The testing procedure has been divided into three parts: tests

with a single mobile client, tests with multiple mobile clients, and tests that assure the priority of voice communication.

Chapter V, Conclusion, gives the conclusion drawn after the testing procedure was completed and provides an insight to the future work and potential deployment of the base station in a real world scenario.

Appendix A, Custom Kenwood Radio Interface Cable, provides a schematic of a radio interface cable used to connect the radio transceiver to the PC's serial (RS 232) port and sound card.

Appendix B, Kenwood TX 7180 Code Plug, shows the steps needed to set the analog transceiver to operate on a desired frequency, to enable the radio's data communication port, and to enable data messages to control the radio's PTT signal.

CHAPTER II

BACKGROUND

The ability to access and manipulate information and services anytime and anywhere are the most important features of today's mobile communication technologies. Information can be accessed through numerous wireless networks using various communication devices such as personal digital assistants, cell phones, and digital radios. These communication devices, capable of transmitting and receiving messages, are used to bring information to a mobile environment. Hence, they are usually combined with intra-vehicular systems to support communication and share information with infrastructure systems. Furthermore, most of the communication between vehicles and infrastructure has been automated by utilizing digital data messages. This automation has resulted in the development of numerous vehicular data services. Today, commercial infrastructure systems offer a wide collection of data services ranging from entertainment to safety. However, use of commercial data services is not widespread among law enforcement agencies because of reliability, high safety requirements, and expensive equipment.

First, this chapter will introduce Weiser's vision of invisible communication and digital data messages that connect various elements of the mobile environment. Then, several intra-vehicular systems and their user interfaces will be mentioned. The intra-

vehicular systems are usually combined with infrastructure systems which offer different vehicular data services. Since the vehicular data services are the main interest of this thesis, various commercial wireless data technologies and data services in mobile environments will be reviewed. Finally, we will look at current communication standards and requirements used by law enforcement and mention several projects which deliver secure and reliable data services over LMR channels.

Invisible Communication

The transparent integration and disappearance of computers and communication technologies into people's everyday lives was first introduced by Weiser [10]. In his vision, different components of the ubiquitous environment are capable of sensing information and sharing the data. For example, a ubiquitous scanning pen can detect a quote from a newspaper and send it to a distant location, or a car can help in avoiding traffic congestion and finding a parking spot. The communication between elements of the mobile environment is completely in the background, providing an invisible connectivity. Furthermore, the invisible connectivity is increased by the constant reduction in size and weight of hardware, improvements in wireless networking, and development of user interfaces [11]. In Weiser's world, people are surrounded with numerous components of the ubiquitous environment, but they are not engaged in the interaction all the time. People perceive the events through different user interfaces which bring information from the periphery to the center of their attention [12].

Intra-Vehicular Systems

Today, cars are equipped with tens or even hundreds of embedded computers that control almost everything from satisfying emission-control standards to automatically adjusting the volume of a car audio system. This subsection will look at several projects that integrate different components into intra-vehicular systems.

Vehicular systems are designed to provide assistance to the driver by integrating vehicular devices, processing inputs from multiple sensors such as cameras and radars, and representing sensed data through user interfaces. For instance, a driver assistance system which uses vehicular radar can perceive the surroundings of the vehicle through near and far distance radar sensors and can be used in applications such as adaptive cruise control, pre-crash sensing, blind spot detection, parking assistance, lane change assistance, collision warning, and urban collision avoidance [13]. In addition to driver assistance systems that just warn the users of the safety issues, some systems can assist in driving by performing adaptive cruise control or by taking full control of driving [14].

The introduction of intra-vehicular systems into law enforcement vehicles has increased the efficiency of the officers on patrol and has given them the ability to perform usual tasks more easily. Although the installation of computers may allow officers to receive safety-critical information in a timely manner, it also may increase the driver's workload, especially in mission-critical operations. According to [6], a typical computer mouse-keyboard interface used in police vehicles to perform a usual task could increase the driver's workload by more than seven hundred percent compared with performing the

same task using a manual user interface. Also, by conducting studies in the field they showed that a speech user interface reduces the workload even for the task of performing a license plate query. This was measured by the number of glances necessary to complete the task. Another study also showed that performing a task of changing the radio channel using a speech user interface reduces the driver's distraction compared to a manual user interface [1]. One of the systems which provide a speech user interface and the system used in previously mentioned studies is Project54 [5].

Project54 is a research and development effort between the University of New Hampshire and the New Hampshire Department of Safety [15]. The system is a highly integrated in-vehicle hardware/software system whose main goal is to improve the safety and functionality of NH State Police and local police cruisers. It is a completely computer based system that simplifies the interaction with in-vehicle electronic devices and allows officers to control them using a speech user interface. Through a physical network and the use of open hardware and software standards, the system communicates with all common police devices such as the lights, siren, and radio. One of the features of the system is the ability to support data communications. This is possible by using digital radio equipment or connecting to a Wi-Fi network [16].

Project54's ability to support data communication and the data communication channel developed in this thesis can be combined to enable the task of performing license plate and driver queries over a mobile radio channel. Similarly as in [6], the combination

of a speech user interface and data communications should allow the officers to perform the queries without taking their eyes off of the road or hands off of the wheel.

Vehicle to Infrastructure Networks

This subsection will give an overview of commercial systems and technologies which are commonly used for communication between vehicles and infrastructure systems.

The automotive and public transportation industries have been the focus of many research projects. Advanced wireless networks have allowed for information sharing between vehicles and transmitting data to fixed infrastructure systems. The most used wireless communication systems in commercial automotive applications are cell phone and Wi-Fi networks.

In traffic information sharing systems, vehicular sensors and cell phone networks have been used to update information about the traveling speed of vehicles on every road segment to the server [17]. The server manages real-time measured velocities from the vehicles and sends back the traveling speed information of approaching road segments on the route. The vehicles receiving the speed information of the approaching segments from the server can avoid traffic congestion, calculate travel time, and find minimal-time route to the destination. The accuracy of the systems depends on continuous updating of the traffic information from mobile nodes to the server. The communication between the nodes and server can generate a lot of data traffic and create delays in the communication

channel. However, reducing the amount of communication lowers the accuracy of the system. To maintain a trade-off between the amount of wireless data and accuracy of the system a randomized update policy with a transmission probability strictly smaller than 1 has been used.

A project by Skordylis and Trigoni delivers sensed data from vehicles to fixed infrastructure nodes in urban settings using Wi-Fi ad-hoc networks [18]. The information propagates hop-by-hop, from vehicle to vehicle, to roadside access points connected to the fixed network. In this approach, the information about traffic congestion, road faults, and accidents is delivered to a traffic monitoring center that can call for assistance from public safety departments. They propose different algorithms that intend to deliver messages from vehicles to an access point with limited delay by minimizing the number of multi-hop transmissions between the vehicles. Also, message priorities are important in systems like this one, where information about serious accidents must be delivered to the monitoring center much faster than information about road faults or traffic congestion.

Numerous research projects from the public transportation field allow users to access mobile databases and get information about schedule, events, and available services. The users of a passenger support system can make their travel plans by accessing several databases using mobile terminals and gathering information about route, fare, area map, station map, and real-time operation schedule [19]. The mobile terminals, connected to the system using Wi-Fi connections, can serve as travel agents

which inform passengers of current railway conditions, make travel plans according to the users' requests, purchase tickets, and guide the passengers during their entire travel experience. The user interface of the terminals can be personalized to support many passengers with special needs such as visually disabled persons, aged persons, foreign tourists, and people who are not familiar with public transportation services. Visually disabled passengers are guided to the destination by a voice navigation user interface. Furthermore, mobility agents connected to the cell phone network can deliver highly personalized instructions to people with limited ability to perceive, recognize, understand, interpret, and respond to information [20]. Based on the data about the travelers' specific needs, the agents can track, guide, provide memory prompts and cues for what to do and where to go, call for help, and send emergency messages when it is necessary.

The users of transportation systems usually spend long periods of time on board while traveling. That provides a very good environment to provide entertainment and various mobile data services. An on-board data service platform, called BlueBus, provides localized Wi-Fi and Bluetooth data services to the passengers [21]. The local database on BlueBus is updated at the bus terminals along the route through WiFi access points. The bus leaves the bus terminal with the latest content and provides travelers with fresh news and information on the way. The travelers can obtain the information using a Bluetooth mobile device, access multi-user services such as chat and multi-player games, and share information through mobile blogs. In addition to information access and entertainment, wireless access to the internet in a mobile environment can be used for

educational purposes. An online knowledge testing tool allows students to take multiple-choice tests on mobile devices while on their way to school [22].

The development of wireless data applications and services has been driven by convenience and safety. Therefore, in some situations users have to access data services with the information of their physical location. However, revealing user position raises serious privacy and security concerns [23]. For example, unwanted persons can access the information to track the location of the service users. The simplest solution to these problems is to use a fake identity. However, the simple usage of fake identity is not very suitable for location-based services because the location can reveal the true identity of the user. The true identity of a home owner can be revealed by asking for the nearest restaurant to the house, even if the owner is using a fake identity. To address this problem, Mokbel *et al* developed a framework, called Casper, in which users can use location-based services without revealing private location information [24]. Casper blurs the exact location of the users into cloaked spatial areas based on specific privacy profiles. Besides the location, automotive telematics applications may include personal and sensitive information that can threaten the driver's privacy. A data protection framework, proposed by Duri *et al* enables data aggregation before data is released to a service provider [25]. The framework provides flexible privacy policies that minimize the disclosure of private information.

One of the systems that provides additional information to police officers on patrol and increases their awareness of incident location is a location-based notification system

called *Attentive Service* [26]. The system was designed to provide auditory signals and pop-up messages and proactively notify police officers with the location of incidents, other colleagues, and crime hotspots in their current vicinity. With the system, the officers are able to handle incidents faster, rely less on communication with the dispatcher, and increase their awareness of incidents because they are notified about relevant information on location. However, the system's complexity and its dependence on Bluetooth, Wi-Fi, VPN, and GPS connections result in a lack of robustness and occasional system malfunctions. Also, the field evaluation showed that potential distraction and interruption by irrelevant notifications have to be addressed in the design of innovative location-based notification services for police officers.

Most of the wireless data services currently available would be beneficial for law enforcement agencies in tracking and organizing their personnel. However, even with the numerous benefits that the data services can offer, they are not widely used among first responders because of concerns about the security, survivability and redundancy of commercially available networks. Public safety and mission-critical operations cannot count on public networks unless they satisfy high requirements of security, reliability, priority in traffic, traffic behavior, load conditions, and quality of service [2] [3]. For example, certain events may compromise a commercial system's ability to operate. Natural disasters can damage the public infrastructure, extreme crowds that might occur during riots and attacks can overload the communication bandwidth, or simple adding of a new data service can cause unpredictable traffic behavior. For these reasons, law

enforcement agencies often rely on agency operated LMR communication for mission critical applications.

Land Mobile Radio Communication for Law Enforcement

This subsection will focus on standards for digital radio communication as well as several commercial and research projects that allow for data communication over land mobile radio channels.

Law enforcement communication primarily depends on secure, agency operated Land Mobile Radio (LMR) channels and the spectrum reserved by the Federal Communication Commission (FCC) for public safety use. Standards for digital communication, called APCO Project 25, have introduced and added data signaling to conventional LMR systems. However, only a few currently deployed LMR systems have the very expensive but necessary equipment to support the data portion of the standards.

Project 25

The Association of Public-Safety Communication Officials (APCO) established Project 25 (P25) to address the radio interoperability problem and to make the usage of scarce radio frequencies more efficient. P25 represents a set of common technical standards, developed by the Mobile and Personal Private Radio Standards Committee

(TIA TR-8) of the Telecommunications Industry Association (TIA), that outline digital two-way land mobile radio communications [27] [28].

The P25 suite of digital radio standards is designed to meet radio interoperability requirements among local, state, and national public safety agencies. The suite specifies a definition and description of P25 system elements, interfaces, and system's architecture, allowing different manufacturers to develop interoperable equipment. A typical P25 radio system consists of radio units, base station(s), and other fixed radio equipment. Radio devices are often called subscriber units, which include mobile radios for use inside vehicles and portable radios for handheld operation. A typical base station consists of a central unit, a receiver module, a transmitter module, and supported interfaces. Other fixed radio equipment is used for console and wide-area operations, as well as for data communications with the fixed network (computer equipment) [4].

A set of open intra- and inter- system interfaces allows for interoperable digital communication between all system elements. The interfaces defined by the standard are: Common Air Interface (CAI), Subscriber Data Peripheral Interface, Fixed/Base Station Subsystem Interface (FSSI), Console Subsystem Interface (CSSI), Network Management Interface, Data Network Interface, Telephone Interconnect Interface, and Inter-RF Subsystem Interface (ISSI) [27]. The most important interface and a key for digital communications is CAI. It enables digital wireless communication, both data and voice, among multiple subscriber units, base stations, and other fixed equipment with a maximum bit rate of 9600 bps [4]. Data communication is further described in an

additional four P25 documents: Data Overview [29], Packet Data Specification [30], Circuit Data Specification [30], and Radio Control Protocol [31]. Analog voice signals, at the input of the system, are converted into digital signals using an analog-to-digital converter and a voice coder. The voice coder is often called the vocoder and it is based on the Improved Multi-Band Excitation (IMBE) voice coding algorithm. Once the digitally represented voice is transmitted over the channel using CAI, it is decoded back to analog by passing through the vocoder and a digital-to-analog converter [32].

The standard specifies two modes of operation: conventional [33] and trunked [34]. While conventional systems have no centralized management and the channel access is manually controlled by the users, trunked systems provide automatic control of all parts of radio system operation, including call routing and channel access. However, all P25 systems and equipment, designed according to the standard, should support both conventional and trunked operation. The only difference between these two types of operation is in the supported feature set and the channel access method [28].

P25 endeavors to achieve FCC spectrum efficiency requirements by moving to narrowband channel spacing through a two-phase plan. The goal of Phase 1 is to provide a channel spacing reduction from 25 kHz to 12.5 kHz using a Frequency Division Multiple Access (FDMA) scheme and employing a 4 level frequency modulation (C4FM). Phase 2 provides a further reduction in channel spacing to 6.25 kHz utilizing a Time Division Multiple Access (TDMA) technique with a differential offset quadrature

phase shift keying (CQPSK), a digital modulation which creates two slots of 6.25 kHz in a 12.5 kHz channel [27] [35].

Project 25 Data Communication

The implementation of data communication over an LMR channel requires the necessary equipment, mobile units and base station, which supports both voice and data signaling. Nearly all public safety agencies in New Hampshire already have digital mobile radios capable of data communication. In contrast, commercially available base stations which support the data part of P25 are still very expensive and they are not affordable by small local departments.

Many third party companies have worked on the implementation and development of P25 Base Stations. Those companies, such as *Tyco Electronics* [36] and *Etherstack* [37], provide a comprehensive range of software-defined P25 voice and data solutions. *Tyco Electronics* developed the VIDA communication network that supports a line of communication systems including P25^{IP}. This system enhances the P25 standards with the advantages of an IP-based infrastructure. *Etherstack* offers several variants of APCO P25 compliant base stations. These variants are written in highly portable ANSI C/C++ with a layered architecture abstracted from the underlying hardware platform and operating system. Because these solutions are software-based, they are highly flexible, so the reconfiguration or updating of the system can be easily achieved. Still, these systems keep a high level of complexity by integrating voice and data signaling, by supporting

transparent hand-off across multiple repeaters and trunked channels, and so forth. Such complexity is not necessary for small public agencies that already have fully operational voice networks and operate on a single conventional channel. They only need an inexpensive, stand alone data capable base station.

Work on a single, stand alone P25 data base station started at the University of New Hampshire with the overall design and development of the software based P25 data packet transmitter [9]. This work demonstrated how data packets can be broadcast over the air using a desktop computer and an analog radio. The desktop computer ran digital signal processing software that received IP packets through a standard Ethernet interface. When IP packets were received by the application, they were encoded into CAI data packets which were then passed to an analog radio as analog waveforms through the computer's sound card. The waveforms were broadcast using the analog FM transmitter on a predefined channel. A mobile digital radio tuned to the same frequency captured the waveforms and decoded the sent IP packets. Although this work was not directly useful to public safety agencies because it provided only one-way communication, it represents a successful step toward a software defined Project 25 data base station.

The complete software defined data base station described in this thesis, which consists of a transmitter, receiver, and base station logic, enables small departments to deliver additional information to the cruisers in a cost effective way. Also, the usage of a single digital radio for both voice and data traffic reduces the cost of providing data services to the vehicles without involving additional wireless devices. In this work, the

design and development of the software defined receiver and Project 25 data base station will be presented.

CHAPTER III

SOFTWARE DEFINED APCO PROJECT 25 DATA BASE STATION

APCO Project 25 represents a set of technical standards that defines digital two-way communication between mobile, portable, and base station radios over a LMR channel. The communication between the radios is described through an open interface called the Common Air Interface (CAI). The CAI allows for both voice and data transmission over a single LMR channel. This chapter focuses on the overview of the base station architecture, the reception and decoding of CAI data packets, and the design of the base station software application.

System overview

Nearly all local NH police departments have implemented phase 1 Project 25 digital LMR communication and have P25 compliant digital mobile radios. A typical voice system that can be found in local police departments consists of a dispatch console and a voice base station. The system provides only voice communication, between the dispatch and mobile units on patrol, over a single conventional radio channel. This is illustrated in Figure 2.

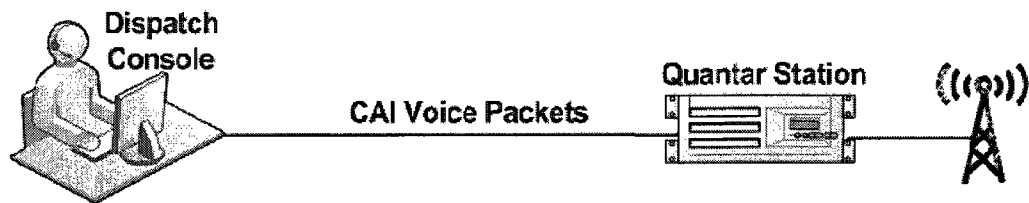


Figure 2 - A typical voice communication system

In this setup, a mobile unit obtains information by operating a microphone of a P25 compliant digital radio and listening to the radio speaker. The input to the radio is the officer's voice, which is digitized, encoded, frequency modulated, and transmitted over a LMR channel in the form of voice CAI packets. At headquarters, the voice base station receives the signals, performs demodulation, and decodes the voice CAI packets. Decoded voice CAI packets are then played over a speaker at the dispatch console as analog voice waveforms. As a response, the dispatcher voice is passed through the system in the reverse direction and it is played to the officers on patrol over the radio speaker.

In addition to the voice communication, the CAI also describes wireless data communication among subscriber units, data base stations, and other fixed end computer equipment [4]. Following the standard, both voice and data can be transmitted and received on the same radio channel using a Time Division Multiple Access (TDMA) technique [35]. Using this technique, voice messages are given priority and data messages can be transmitted only when there is no voice signaling on the channel.

In this thesis we implemented a software defined P25 data base station that implements the data portion of CAI. The designed base station is compliant with commercial P25 digital radios and operates on the same conventional radio channel and in parallel to the P25 voice system shown in Figure 2. The base station is comprised of a desktop computer with a standard network interface and an analog FM radio. This is illustrated in Figure 3.

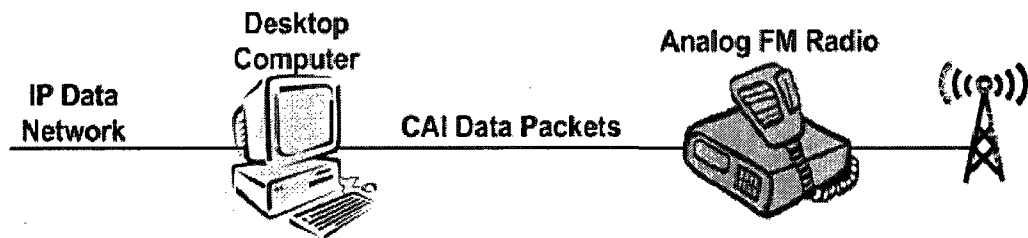


Figure 3 – Project 25 Data Base Station designed in this thesis

With the data base station, mobile units on patrol can access information stored on the remote servers without contacting dispatch personnel using voice communication. When a mobile unit needs information, it can generate and transmit a data query over a LMR channel in the form of a data CAI packet using a P25 digital radio. At headquarters, the analog radio receives the data CAI packets and performs frequency demodulation. The analog version of the digital data packet on the output of the analog radio is passed to the computer through a sound card input. At the computer, the analog signal is digitized and decoded into an IP query which is then placed on the IP data network through the computer network card. The response to the IP query, received through the IP data network, is passed through the base station in the reverse direction. The block diagram of the base station architecture is shown in Figure 4.

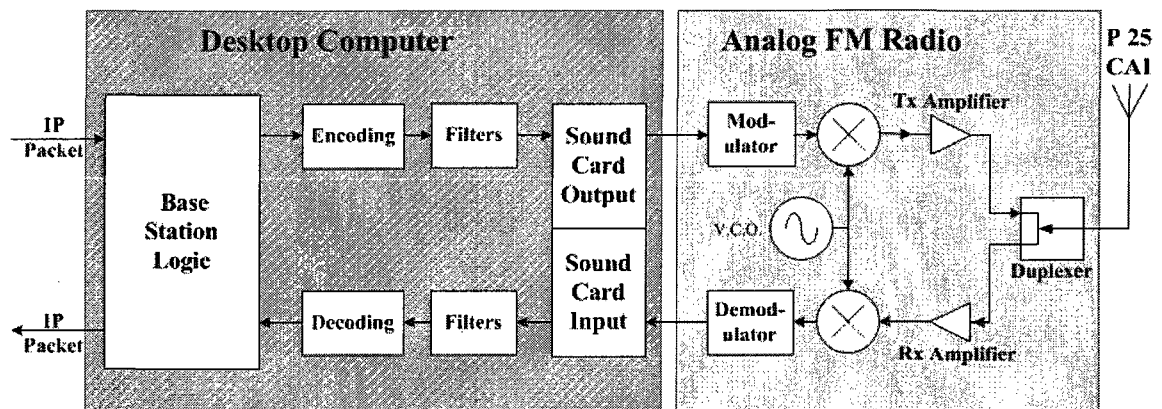


Figure 4 - The Data Base Station Architecture

The flow of data packets between the analog radio and network interface is managed by a software application which runs on the computer. The application is comprised of a transmitter, a receiver, and a base station logic component. The transmitter is used to transform IP packets into digital versions of data CAI packets and pass them to the analog radio through a sound card output. The transmission process involves encoding and baseband filtering, which were described and implemented by Ramsey [9].

The receiver takes the digitally sampled version of CAI packets from the input of the audio card. The receiver decodes IP packets which are then passed to the network interface. The synchronization between transmission and reception of both CAI and IP data packets is managed by the base station logic. Also, the logic component controls push to talk signaling (PTT), detects collisions between packets, and retransmits collided

packets. The next two subchapters focus on the reception and decoding of CAI data packets and the base station control logic.

Reception and Decoding of Digital Data Packets

The CAI uses a packet technique to transfer digital data messages over a radio channel. The data message is split into fragments which are formed into packets. Then, the packets are split into a sequence of information blocks. Each information block is protected by a trellis code, for error correction, and a check sum which is used to verify the reception and possible error correction. The information block structure differs based on the type of delivery. Data can be delivered with either confirmed or unconfirmed type. The type of delivery depends on whether the sender of the packet requires an acknowledgment of receipt or not.

The data packets are modulated using the 4-frequency modulation called C4FM. The C4FM specifies four different frequency deviations which correspond to a set of two bits, also called a dibit. Each dibit is usually represented by a symbol. The frequency deviations and their corresponding bits and symbols are listed in Table 1.

Table 1 - C4FM Frequency Deviations

| Information bits | Symbol | C4FM Deviations |
|------------------|--------|-----------------|
| 01 | +3 | 1.8 kHz |
| 00 | +1 | 0.6 kHz |
| 10 | -1 | -0.6 kHz |
| 11 | -3 | -1.8 kHz |

Next, we will look in detail at how the CAI data packets are decoded for both confirmed and unconfirmed types of transmissions. The block diagram of the decoding process is shown in Figure 5. Complete technical specification for the CAI can be found in [4].

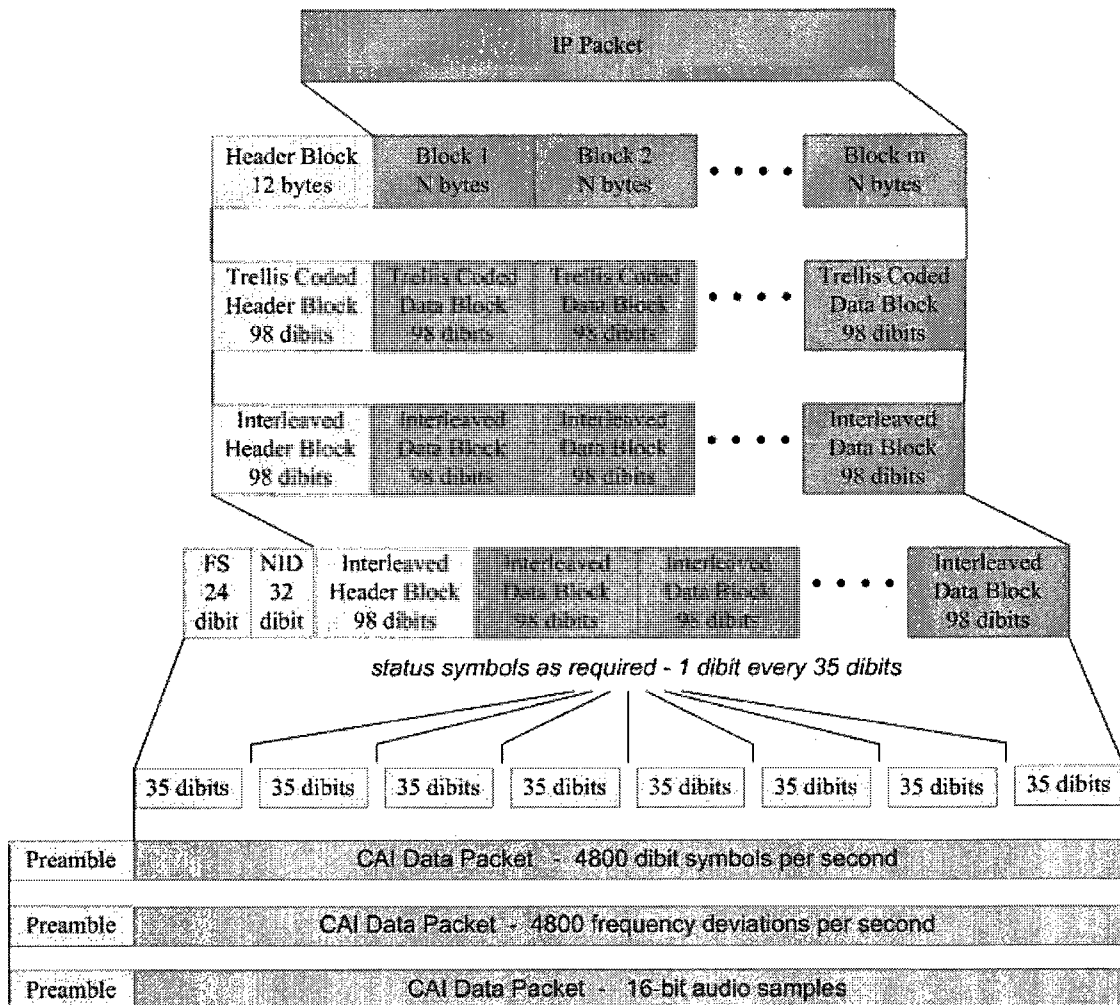


Figure 5 - CAI to IP packet Conversion

The input to the receiver is a clock synchronization preamble followed by a CAI data packet, in the form of a 16-bit digital signal sampled at the rate of 48000 samples per

second. This is illustrated at the bottom of Figure 5. The clock preamble is used to synchronize the receiver clock to the proper phase shift for each modulated symbol. The preamble signal is a repeated sequence of bits: 01 01 11 11 (frequency deviations of: 1800 1800 -1800 -1800 Hz). The resulting frequency deviation waveform during the preamble is a 1200 Hz sine wave with a peak amplitude (peak FM frequency deviation) of 2880 Hz. The sinusoidal preamble signal is used as a reference for implementing baseline offset adjustment and automatic gain control (AGC), and for synchronizing the symbol extraction clock. When the receiver clock is synchronized with the input signal, the 16-bit sample stream is downsampled to 4800 symbols per second. The symbol samples are converted into the four ideal frequency deviations and their corresponding bits (Table 1). Among the bits are status symbols which indicate the status of the radio channel.

Status Symbols

A typical operation on a radio channel uses a radio frequency pair. One frequency is used to transmit outbound messages while another frequency is used to receive inbound messages. With the radio frequency pair, it is possible to simultaneously transmit and receive messages. During the transmission of a message, the information about the status of the inbound channel (idle or busy) is sent to all the listening subscriber units. A subscriber unit can transmit data messages only when the inbound channel is idle.

Information about the status of the inbound channel is interleaved throughout the data information blocks such that there are two status bits (one status symbol) after every

70 bits (35 symbols) of data information. Status symbol codes are given in Table 2, and their position in the data stream is graphically illustrated in the middle of Figure 5.

In this work, a single frequency is used for the transmission of outbound and the reception of the inbound messages. Therefore, the status code 10 is used to denote that the inbound channel can be used for both transmission and reception of the messages.

Table 2 - Status Symbol Codes

| Status Symbol | Meaning |
|---------------|--------------------------------------|
| 01 | Inbound Channel is Busy |
| 00 | Unknown, use for talk-around |
| 10 | Unknown, use for inbound or outbound |
| 11 | Inbound Channel is Idle |

The very first bits that follow the clock synchronization preamble are used to mark the beginning of the data packet. These bits are called a frame synchronization word.

Frame Synchronization Word

The frame synchronization word is a special sequence of bits, known to both the transmitter and receiver, used to mark the location of the first bit of the message data. It is required at the beginning of every voice and data packet, immediately after the clock synchronization preamble and immediately before the first information bits. The synchronization word consists of 48 bits (24 symbols) which are specified by the standard. These bits are listed in Table 3.

Table 3 - Frame Synchronization Word sequence

| First bit | | | | | | | | | | Last bit | |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0101 | 0101 | 0111 | 0101 | 1111 | 0101 | 1111 | 1111 | 0111 | 0111 | 1111 | 1111 |
| 0x5 | 0x5 | 0x7 | 0x5 | 0xf | 0x5 | 0xf | 0xf | 0x7 | 0x7 | 0xf | 0xf |

The frame synchronization word is followed by a network identifier.

Network Identifier

The network identifier is used to address radio networks or specific repeaters depending on the radio system configuration and to identify the type of message. The identifier is placed at the beginning of each information block sequence. The subscriber units use the network identifier to receive the data packets addressed to their network and reject the traffic from other radio networks. Also, the identifier contains information about the type of message (e.g., voice or data) and allows the receiver to perform the correct decoding and error correction. Therefore the network identifier has two fields, network access code and data unit id, which are encoded in 16 bits of information. These are listed in Table 4. These 16 bits of information are protected with a BCH code (described below), which results in a total of 64 bits of the network identifier that are placed at the beginning of each data unit.

Table 4 - Network Identifier

| Network Access Code | | | | | | | | | | | | Data Unit ID | | | |
|---------------------|-----|----|----|----|----|----|----|----|----|----|----|--------------|----|----|----|
| N11 | N10 | N9 | N8 | N7 | N6 | N5 | N4 | N3 | N2 | N1 | N0 | D3 | D2 | D1 | D0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

The Network Access Code carries 12 bits of information. This information is not specified by the standard and can be any arbitrary code that uniquely identifies the radio network. For the testing procedures in this work we chose the network access code to be 0x54.

The Data Unit ID contains 4 bits of information. The unit id specifies the type of the message that follows the network identifier and can take any of the values listed in Table 5.

Table 5 - Data Unit Identifier Values

| Data Unit ID | Parity Bit | Data Unit Usage |
|---------------|------------|--|
| 0000 | 0 | Header Data Unit |
| 0001 and 0010 | 1 | Reserved |
| 0011 | 0 | Terminator without subsequent Link Control |
| 0100 | 0 | Reserved |
| 0101 | 1 | Logical Link Data Unit 1 |
| 0110 | 1 | Reserved |
| 0111 | 0 | Trunking Signaling Data Unit |
| 1001 and 1001 | 1 | Reserved |
| 1010 | 1 | Logical Link Data Unit 2 |
| 1011 | 0 | Reserved |
| 1100 | 0 | Packet Data Unit |
| 1101 and 1110 | 1 | Reserved |
| 1111 | 0 | Terminator with subsequent Link Control |

The value 1100 denotes the packet data unit and it is used by the data base station to prepare data messages for transmission. All incoming messages that are not marked with the packet data code are disregarded.

The information contained in the network identifier is protected with a (63, 16, 23) BCH code. The code is generated using the extension Galois Field which uses a

generator polynomial of the 47th degree with 27 non-zero terms. The polynomial written in octal notation is:

$$g(x) = 6331\ 1413\ 6723\ 5453.$$

This coding process produces a sequence of 63 bits which is appended with a single parity bit given in Table 5 for each data unit identifier. The final generator matrix for the full code of 64 bits is shown in Table 6. The matrix is given in octal notation, where the left 16 bits form an identity matrix and the right 48 bits form a parity check matrix.

Table 6 - BCH Code Generation Matrix

| | Identity 16 x 16 | | 16 x 48 Parity Check | | | |
|----|------------------|------|----------------------|------|------|------|
| 1 | 10 | 0000 | 6331 | 1413 | 6723 | 5452 |
| 2 | 04 | 0000 | 5265 | 5216 | 1472 | 3276 |
| 3 | 02 | 0000 | 4603 | 7714 | 6116 | 4164 |
| 4 | 01 | 0000 | 2301 | 7446 | 3047 | 2072 |
| 5 | 00 | 4000 | 7271 | 6230 | 7300 | 0466 |
| 6 | 00 | 2000 | 5605 | 6507 | 5263 | 5660 |
| 7 | 00 | 1000 | 2702 | 7243 | 6531 | 6730 |
| 8 | 00 | 0400 | 1341 | 3521 | 7254 | 7354 |
| 9 | 00 | 0200 | 0560 | 5650 | 7526 | 3566 |
| 10 | 00 | 0100 | 6141 | 3337 | 5170 | 4220 |
| 11 | 00 | 0040 | 3060 | 5557 | 6474 | 2110 |
| 12 | 00 | 0020 | 1430 | 2667 | 7236 | 1044 |
| 13 | 00 | 0010 | 0614 | 1333 | 7517 | 0422 |
| 14 | 00 | 0004 | 6037 | 1146 | 1164 | 1642 |
| 15 | 00 | 0002 | 5326 | 5070 | 6351 | 5373 |
| 16 | 00 | 0001 | 4662 | 3027 | 5647 | 3127 |

Upon the reception of 64 bits that encode the network identifier, the receiver first extracts the network access code. If the code matches the real network code, the data unit

id is extracted. Only the packet data unit value is accepted, while all other messages are disregarded. Once the proper NAC and DUID are extracted, the parity check is calculated using the BCH code generation matrix. The generated parity check is compared to the parity check received as the last 48 bits of the network identifier. If the parity checks match, the following data blocks are passed to the inverse interleaving block.

Inverse Interleaving

The data blocks, received after the network identifier, contain the data payload that has been interleaved. Interleaving is used to spread burst errors due to Rayleigh fading over each 98 dibit block. The error burst length is minimized by rearranging the dibit sequence to form another dibit sequence according to the interleave index pairs shown in Table 7.

Table 7 - Interleave Table

| Output Index | Input Index | Output Index | Input Index | Output Index | Input Index | Output Index | Input Index |
|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|
| 0 | 0 | 26 | 2 | 50 | 4 | 74 | 6 |
| 1 | 1 | 27 | 3 | 51 | 5 | 75 | 7 |
| 2 | 8 | 28 | 10 | 52 | 12 | 76 | 14 |
| 3 | 9 | 29 | 11 | 53 | 13 | 77 | 15 |
| 4 | 16 | 30 | 18 | 54 | 20 | 78 | 22 |
| 5 | 17 | 31 | 19 | 55 | 21 | 79 | 23 |
| 18 | 72 | 44 | 74 | 68 | 76 | 92 | 78 |
| 19 | 73 | 45 | 75 | 69 | 77 | 93 | 79 |
| 29 | 80 | 46 | 82 | 70 | 84 | 94 | 86 |
| 21 | 81 | 47 | 83 | 71 | 85 | 95 | 87 |
| 22 | 88 | 48 | 90 | 72 | 92 | 96 | 94 |
| 23 | 89 | 49 | 91 | 73 | 93 | 97 | 95 |
| 24 | 96 | 25 | 97 | | | | |

Since the data information blocks have been interleaved at the transmitter, the receiver has to perform inverse interleaving to reconstruct the original dibit sequences. The inverse operation is performed by switching input and output indexes given in Table 7. After the inverse interleave operation is completed, the data information blocks are passed through the error correction decoding block.

Data Error Correction Decoding

All data information blocks are encoded with a trellis code. The trellis code is used to protect the payload and make it more robust to the errors that may happen during the transmission. All header and unconfirmed data blocks are always encoded with a rate $\frac{1}{2}$ trellis code while all confirmed data blocks use a rate $\frac{3}{4}$ trellis code. Before explaining how to decode the data information blocks, we will look at how they are encoded.

The encoding process for both rate $\frac{1}{2}$ and $\frac{3}{4}$ trellis codes is similar. This is illustrated in Figure 6. The input to the rate $\frac{1}{2}$ encoder is 96 bits ($n = 12$ octets) and to the rate $\frac{3}{4}$ encoder is 144 bits ($n = 18$ octets). The data octets are serialized in the same way, but they are separated into 48 dibits ($k = 2$) for the rate $\frac{1}{2}$ encoder and 48 tribits ($k = 3$) for the rate $\frac{3}{4}$ encoder. The dibits or tribits are appended with a flushing dibit (00) or tribit (000), to round the number of dibits or tribits to the final $m = 49$, and then passed through a trellis finite state machine. The output of the trellis finite state machine is a sequence of 49 constellation points or 98 dibits (196 bits), where each constellation point represents a pair of dibits.

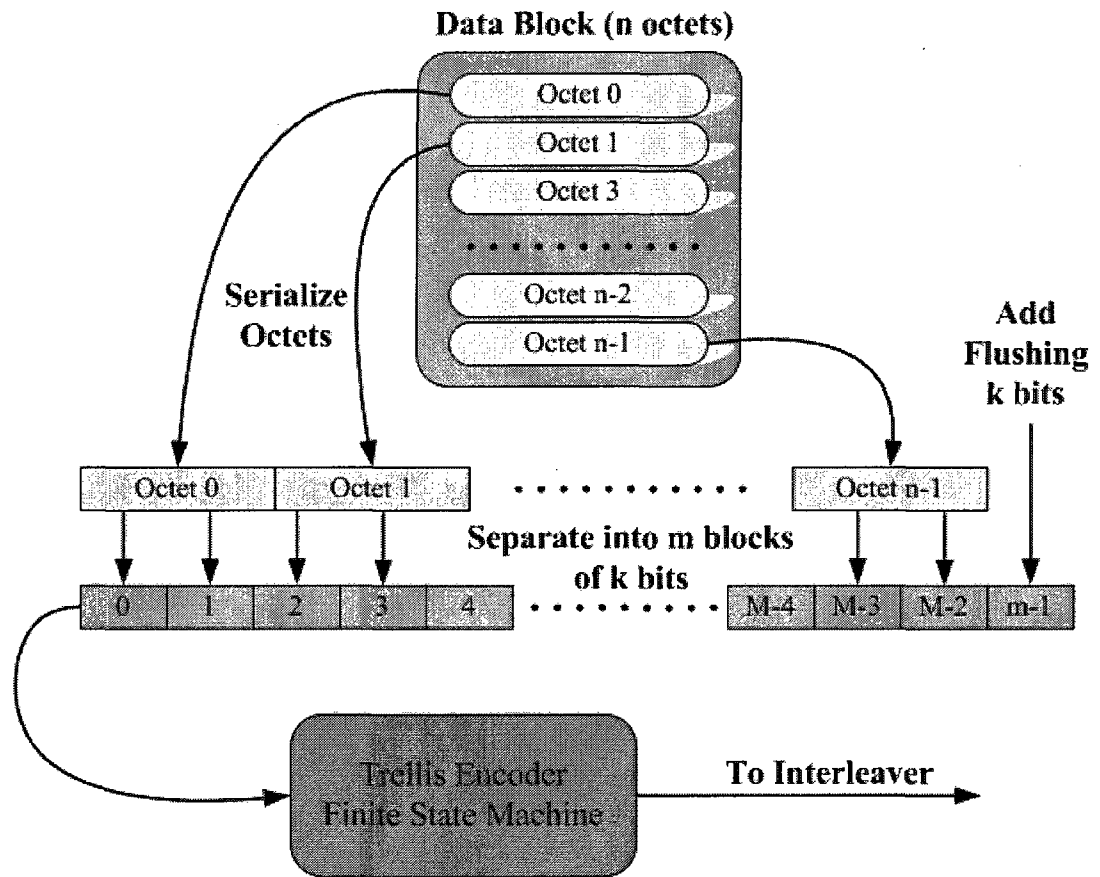


Figure 6 - Trellis Encoder Overview

The trellis encoder is implemented as a 4-state finite state machine for the rate $\frac{1}{2}$ code and an 8-state finite state machine for the rate $\frac{3}{4}$ code. Both machines have 00 (000) for the initial and final states and use the current input as the next state. This is diagrammed in Figure 7.

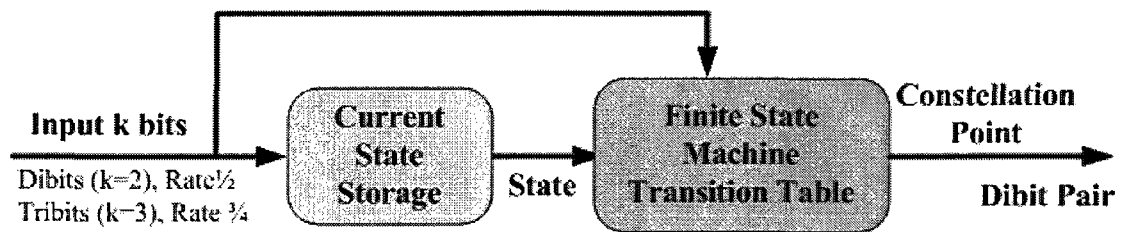


Figure 7 - Trellis Encoder Block Diagram

The four state machine uses the transition states listed in Table 8, and the eight state machine uses the transition states listed in Table 9. For each dibit or tribit input, the output of the state machine is one of the 16 constellation points. Each constellation point represents a dibit pair as listed in Table 10.

Table 8 - Rate 1/2 Trellis State Transition Table

| Rate 1/2 | | Input Dibit | | | |
|-----------|----|-------------|----|----|----|
| | | 00 | 01 | 10 | 11 |
| FSM State | 00 | 0 | 15 | 12 | 3 |
| | 01 | 4 | 11 | 8 | 7 |
| | 10 | 13 | 2 | 1 | 14 |
| | 11 | 9 | 6 | 5 | 10 |

Table 9 - Rate 3/4 Trellis State Transition Table

| Rate 3/4 | | Input Tribit | | | | | | | |
|-----------|-----|--------------|-----|-----|-----|-----|-----|-----|-----|
| | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| FSM State | 000 | 0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 |
| | 001 | 4 | 12 | 2 | 10 | 6 | 14 | 0 | 8 |
| | 010 | 1 | 9 | 5 | 13 | 3 | 11 | 7 | 15 |
| | 011 | 5 | 13 | 3 | 11 | 7 | 15 | 1 | 9 |
| | 100 | 3 | 11 | 7 | 15 | 1 | 9 | 5 | 13 |
| | 101 | 7 | 15 | 1 | 9 | 5 | 13 | 3 | 11 |
| | 110 | 2 | 10 | 6 | 14 | 0 | 8 | 4 | 12 |
| | 111 | 6 | 14 | 0 | 8 | 4 | 12 | 2 | 10 |

Table 10 - Constellation to Dibit Pair Mapping

| Constellation Point | Dibit 0 | Dibit 1 | Constellation Point | Dibit 0 | Dibit 1 |
|---------------------|---------|---------|---------------------|---------|---------|
| 0 | 00 | 10 | 8 | 11 | 01 |
| 1 | 10 | 10 | 9 | 01 | 01 |
| 2 | 01 | 11 | 10 | 10 | 00 |
| 3 | 11 | 11 | 11 | 00 | 00 |
| 4 | 11 | 10 | 12 | 00 | 01 |
| 5 | 01 | 10 | 13 | 10 | 01 |
| 6 | 10 | 11 | 14 | 01 | 00 |
| 7 | 00 | 11 | 15 | 11 | 00 |

The decoding is a reverse process of the encoding. The input to the decoder is a sequence of 98 dibits (196 bits) received from the inverse interleaving block. The dibits are grouped into 49 pairs and for each pair the constellation point is obtained using Table 10.

The constellation points are then fed to the input of the inverse trellis finite state machine. Each constellation point is matched with the output and the state of the inverse finite state machine using tables Table 8 and Table 9. Note that in the decoding process the input and the FSM state shown in Table 8 and Table 9 become the output and the FSM state of the inverse finite state machine, respectively. The inverse FSM performs error correction differently for the rate $\frac{1}{2}$ and the rate $\frac{3}{4}$ decoders. The initial state at which the decoding starts is 00 or 000 depending on the rate.

For the rate $\frac{1}{2}$ decoding, any inconsistency between the state of the machine and current constellation input point marks an error in transmission. The error correction

algorithm calculates the errors associated with the difference between all the points that match the current state of the machine and the received input point. The point that matches the machine state and has the smallest error associated with it is taken as a correct constellation point. If the following input point matches the next state of the machine, the corrupted bits are successfully corrected. If they do not match, the signal was distorted during transmission and cannot be corrected by using this rate decoder. A maximum of two consecutive bits can be corrected using the rate $\frac{1}{2}$ decoder.

For the rate $\frac{3}{4}$ decoding, the errors associated with the difference between the points matching the current state and received point are calculated for every state. The point that has the smallest error is taken as a correct one. The errors are summed across all states and the path which has the smallest error is used to generate the FSM output sequence.

The FSM output sequence is a series of $m=49$ blocks of k bits ($k=2$ for the rate $\frac{1}{2}$ and $k=3$ for the rate $\frac{3}{4}$). The 49th block is a flushing block which is disregarded. The first 48 blocks are rearranged into octets to form a data information block. The header and unconfirmed data blocks are of $n=12$ octets duration, while confirmed data blocks consists of $n=18$ octets. The arrangement and meaning of bits contained in the octets are given in the next subsections.

Data Header Block Format

The first information block of each data message is a header block. The header block contains 10 octets of address and control information and 2 octets of a header CRC error detection code. The CAI specifies three data header formats: unconfirmed, confirmed, and acknowledgment. These formats are shown in Figure 8, Figure 9, and Figure 10, respectively. The meaning of each field in the formats is discussed below.

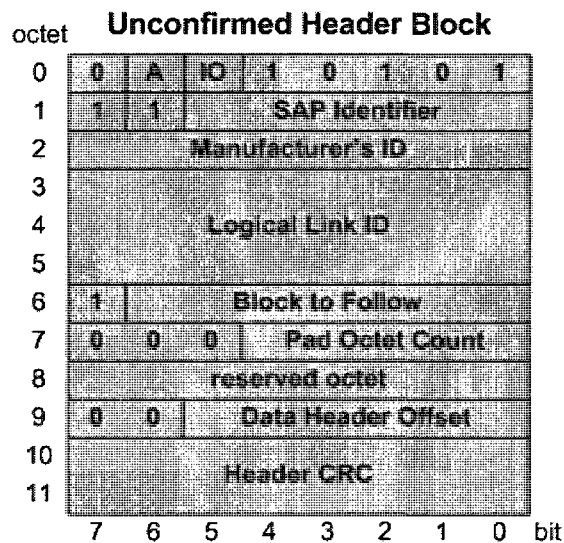


Figure 8 - Unconfirmed Data Packet Header Block

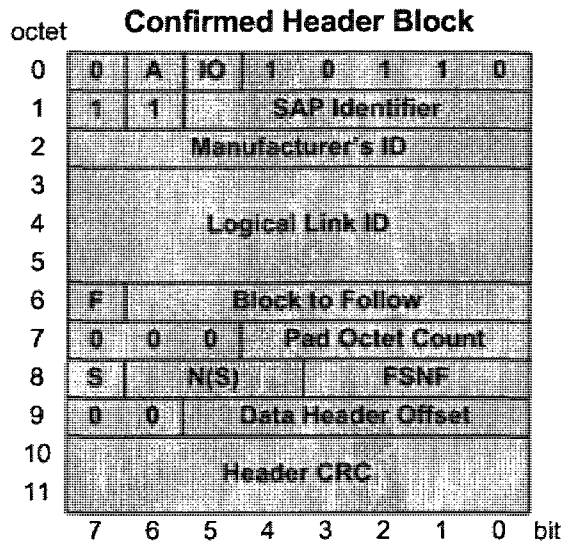


Figure 9 - Confirmed Data Packet Header Block

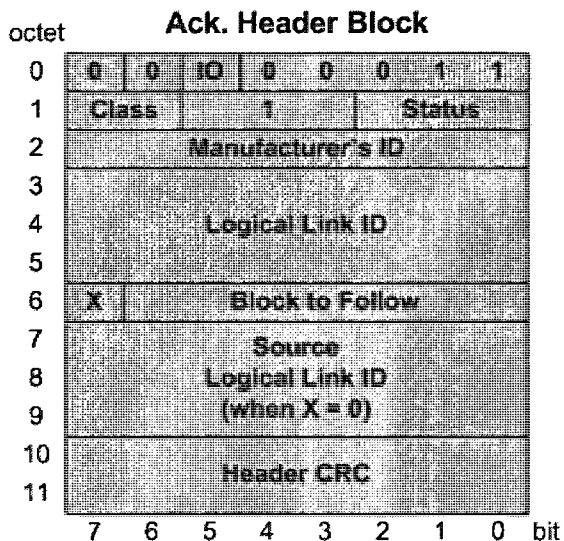


Figure 10 - Acknowledgment Data Packet Header Block

Field **A** (octet 0, bit 6) is used to indicate if confirmation for the packet is required. This field is set to 0 for unconfirmed and acknowledgment packets and to 1 for confirmed packets. Field **IO** (octet 0, bit 5) defines whether the packet is an inbound or outbound message. The value 0 is used to denote a message from a mobile client

addressed to the base station (inbound), while 1 is used at the transmitter to prepare an outbound message for transmission. The **Format** field (octet 0, bits 4 - 0) identifies the message as a data packet with confirmed delivery (10110), a data packet with unconfirmed delivery (10101), or an acknowledgement packet (00011). The **SAP Identifier** (octet 1, bits 5 - 0) describes the final network Service Access Point to which the message is directed. The identifier can take one of the twenty values listed in Table 11. Since the base station provides only data transmissions, the value 0x04 (Packed Data) is always used.

Table 11 - SAP Identifier Values

| SAP Value | Meaning |
|-----------|------------------------------------|
| 0x00 | Unencrypted User Data |
| 0x01 | Encrypted Data |
| 0x02 | Circuit Data |
| 0x03 | Circuit Data Control |
| 0x04 | Packet Data |
| 0x05 | Address Resolution Protocol (ARP) |
| 0x06 | SNDCP Packet Data Control |
| 0x1F | Extended Address |
| 0x20 | Registration and Authorization |
| 0x21 | Channel Re-assignment |
| 0x22 | System Configuration |
| 0x23 | MR Loopback |
| 0x24 | MR Statistics |
| 0x25 | MR Out-of-Service |
| 0x26 | MR Paging |
| 0x27 | MR Configuration |
| 0x28 | Unencrypted Key Management Message |
| 0x29 | Encrypted Key Management Message |
| 0x3D | Non-Protected Trunking Control |
| 0x3F | Protected Trunking Control |

The **Manufacturer's ID** (octet 2) contains information referring to the manufacturer of the radio that is used for the message transmission. This id has a standard value of 0x00. However, every manufacturer can use a different value. The value 0x00 is used in this work. The **Logical Link ID** (octets 3 - 5) identifies the subscriber unit which originates the packet for an inbound message or the subscriber unit to which the packet is directed for an outbound message. This is equivalent to the Ethernet Media Access Control address and the base station uses it to designate the specific radio client who requested the information. When a new radio client sends a data message for the first time, the radio logical link id is paired with the client's IP address at the base station. In this way the base station can respond to any incoming IP packet that is addressed to the specific client by addressing the outbound message with the corresponding radio logical link id. The logical link id is specified for each digital radio on the radio network and has a unique value. The **Block to Follow** field (octet 6, bits 6 - 0) specifies the number of data blocks that follow the header block. It varies depending on the size of a contained IP packet. It is used at the receiver for proper receiving of all data blocks in the packet. The **Pad Octet Count** (octet 7, bits 4 - 0) specifies the number of pad octets (0x00) appended to the data octets to form an integer number of blocks. This value is used at the receiver to discard the pad octets before assembling the IP packet. The actual number of data octets N can be calculated using the following formula:

$$N = M * \text{Block to Follow} - 4 - \text{Pad Octet Count} ,$$

where $M = 12$ for unconfirmed and $M = 16$ for confirmed delivery. The **Data Header Offset** (octet 9, bits 5 - 0) is a pointer that divides the data block into a data header and

data information. This field is used only in specific applications which need more address and control information than can be contained in a regular header block. The extra information is placed at the beginning of the following data block in a format specified by the application. As the data base station uses the regular header formats, the value 0x00 is always used for the data header offset.

The confirmed delivery format specifies several extra fields which are not used in unconfirmed delivery. The **F** field (octet 6, bit 7) represents the full message flag. It is used to distinguish the first message from its retries. If the complete message is transmitted for the first time this bit is set to 1 and for any subsequent retry it is set to 0. This bit is set to 1 for all unconfirmed messages since they do not require a response and cannot be automatically retransmitted. For the acknowledgment (response) format, this field is marked with **X**. It is set to 1 if a response packet is sent in response to the confirmed delivery and to 0 in response to enhanced confirmed delivery. The enhanced confirmed delivery is not supported by the base station designed in this thesis.

The **reserved octet** (octet 8) in the unconfirmed delivery is set to 0x00 and it is not used to carry any particular information. However, this octet is used in the confirmed format to specify a synchronization flag, a sequence number, and a fragment sequence number. The **S** field (octet 8, bit 7) is a synchronization flag used to re-synchronize the physical sub-layer sequence numbers. This flag disables the rejection of duplicate messages when it is set to 1. It is used in specially defined registration messages. In the normal operation of transmitting user data messages, it is set to 0. The **N(S)** field (octet 8, bits 6 - 4) is the sequence number of the packet. The sequence number identifies each

request packet and serves at the receiver for correct ordering of the received message fragments and eliminating any copies. The receiver keeps the sequence number of the last valid received packet $V(R)$. If the next packet has $N(S) = V(R)$, the packet is a copy, and if $N(S) = V(R) + 1$, the packet is the next one in the sequence. The **FSNF** field (octet 8, bits 3 - 0), the fragment sequence number, is used to number the consecutive message fragments that together make up a longer data message. The most significant bit is used to mark the last fragment in the message. The other three bits (FSN) correspond to the fragment number, initially set to 000 for the first fragment, and it is incremented for each subsequent fragment. When the increment reaches the maximum value, 111, the next increment is 001 instead of logical 000. For example, the FSNF for a single physical message is 1000.

The response header block has an additional four specific fields: class, type, status, and a source logical link identifier. The **Class** (octet 1, bits 7 and 6), **Type** (octet 1, bits 5 - 3), and **Status** (octet 1, bits 2 - 0) specify the meaning of the response message. The possible values and meaning of these fields are listed in Table 12. For the normal confirmed delivery, $N(R)$ is the sequence number of the packet $N(S)$. The **Source Logical Link ID** (octets 7 - 9) is set to zeros when the packet is a response for the normal confirmed delivery, $X = 1$. When the packet is a response to the enhanced address confirmed delivery, $X = 0$, this identifier is set with the address of the responding subscriber unit or base station.

Table 12 - Response Packet Class, Type, and Status Specification

| Class | Type | Status | Meaning |
|-------|------|--------|--|
| 00 | 001 | N(R) | ACK - All blocks successfully received |
| 01 | 000 | N(R) | NACK - Illegal Format |
| 01 | 001 | N(R) | NACK - Packet Error, Data CRC check failure |
| 01 | 010 | N(R) | NACK - Memory Full |
| 01 | 011 | FSN | NACK - Out of logical sequence FSN |
| 01 | 100 | N(R) | NACK - Undeliverable |
| 01 | 101 | V(R) | NACK - Out of sequence |
| 01 | 110 | N(R) | NACK - Invalid User disallowed by the system |
| 10 | 000 | N(R) | ACK - Selective Retry for some blocks |

The last two octets (octets 10 and 11) of all three header block formats contain cyclic redundancy check (CRC) bits. The CRC is used for detecting errors caused by noise in the transmission channel. It is calculated from the first 10 octets (80 bits) of the header block using the cyclical redundant coding procedure, called CRC-CCITT, which produces outputs of 2 octets (16 bits). This procedure is defined by four polynomials: $M(x)$, $G_H(x)$, $I_H(x)$, and $F_H(x)$. Polynomial $M(x)$ is of degree 79 and takes the first 80 bits of the header block as its coefficients:

$$M(x) = x^{79} + x^{78} + x^{77} + \dots + x^1 + x^0,$$

where the most significant bit of the zero-th header octet (octet 0, bit 7) matches x^{79} and the last significant bit of the ninth header octet (octet 9, bit 0) matches x^0 . The second polynomial $G_H(x)$ is the generator polynomial defined as:

$$G_H(x) = x^{16} + x^{12} + x^5 + 1.$$

The third polynomial $I_H(x)$ is the inversion polynomial defined as:

$$I_H(x) = x^{15} + x^{14} + x^{13} + \dots + x^2 + x + 1.$$

The fourth polynomial $F_H(x)$ is the header CRC polynomial computed using the following equation:

$$F_H(x) = [x^{16} M(x) \bmod G_H(x)] + I_H(x),$$

where mod stands for the modulus operator (the remainder following binary division).

After computation, the header CRC polynomial can be written in the following form:

$$F_H(x) = x^{15} + x^{14} + x^{13} + \dots + x^1 + x^0.$$

The 16 coefficients of the final $F_H(x)$ are the result of the CRC procedure. At the transmitter, these bits are placed at the last two octets (octet 10 and 11) of a header block. The most significant bit of the eleventh octet of the header block (octet 11, bit 7) matches x^{15} , and the last significant bit of the eleventh octet (octet 11, bit 0) matches x^0 .

At the receiver, when the header data block is received, the CRC code is calculated over the address and control bits, the first 10 octets. When the CRC is computed it is compared with the last two octets, the CRC calculated at the transmitter, of the received header block. If they match, the header block is received with no errors. The address and control bits are extracted as was described above. The information from the data header block is used for proper reception and assembling of the following user data blocks. The formats of the following data blocks are explained in the next subsections for each type of delivery. If the CRCs do not match, the header is received with an error and the following blocks are disregarded with the possible indication for the retransmission of the entire data packet.

Unconfirmed Packet Data Block Format

The information of the type of delivery is contained in the data header block. If the message is sent with unconfirmed delivery the blocks which follow the header block consists of 12 octets of user data. The user data in this work is the IP packet. With the unconfirmed delivery, the data blocks are not protected separately and they do not contain any serial number. The total number of data blocks and the number of pad octets in the last block are extracted from the header block. The last data block contains user data followed by pad octets and a 32 bit CRC, calculated over all data octets. The CRC is placed in the last four octets of the last data block (octets 8 - 11). The format of the last unconfirmed data block is shown in Figure 11.

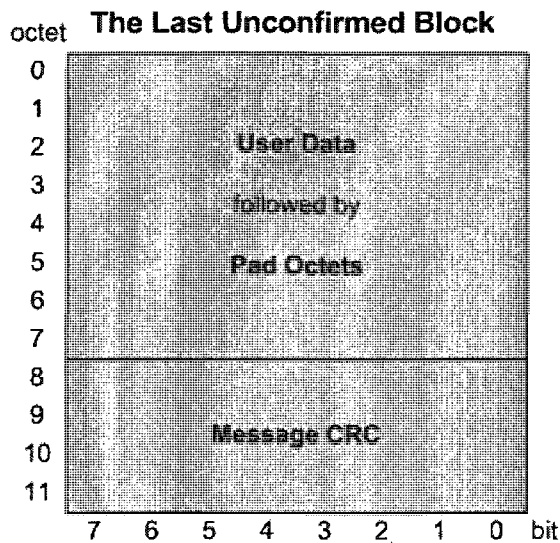


Figure 11 - Unconfirmed Last Data Block

The message CRC is a 4 octet (32 bits) cyclic redundancy check coded over the entire user data octets included in the intermediate blocks and the data octets and pad octets of the last data block. The CRC procedure is defined by four polynomials: $M(x)$,

$G_M(x)$, $I_M(x)$, and $F_M(x)$. Polynomial $M(x)$ is of degree $k - 1$, where k is the total number of user information and pad bits over which the message CRC is calculated. $M(x)$ takes the k bits of the data as polynomial coefficients:

$$M(x) = x^{k-1} + x^{k-2} + x^{k-3} + \dots + x^1 + x^0 ,$$

where the most significant bit of the zero-th message octet corresponds to x^{k-1} and the last significant bit of the last message octet corresponds to x^0 . The second polynomial $G_M(x)$ is the generator polynomial defined as:

$$G_M(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

The third polynomial $I_M(x)$ is the inversion polynomial defined as:

$$I_M(x) = x^{31} + x^{30} + x^{29} + \dots + x^2 + x + 1.$$

The fourth polynomial $F_M(x)$ is the message CRC polynomial computed using the following equation:

$$F_M(x) = [x^{32} M(x) \bmod G_M(x)] + I_M(x),$$

where mod stands for the modulus operator (the remainder following binary division).

Once computed, the message CRC polynomial can be written in the following form:

$$F_M(x) = x^{31} + x^{30} + x^{29} + \dots + x^1 + x^0.$$

The 32 coefficients of the final $F_M(x)$ are the result of the CRC procedure. At the transmitter, these bits are placed at the last four octets of the last unconfirmed message block. The most significant bit of the eighth octet of the last block (octet 8, bit 7) matches x^{31} , and the last significant bit of the eleventh octet (octet 11, bit 0) matches x^0 .

When all unconfirmed data blocks are received, the 32 bit CRC code is calculated and matched with the last four data octets of the last data block. If the CRC codes match, it is assumed that all data block have been received with no errors and they are assembled into the original IP packet. The IP packet is analyzed and placed on the IP network. If the codes do not match the message is disregarded.

Confirmed Packet Data Block Format

The data blocks that follow a confirmed header block contain 18 octets of information. The first 2 octets contain 7 bits of a serial number and 9 bits of a CRC code. Each data block is protected with a CRC-9 code and if an error occurs the data block can be retransmitted separately by specifying its serial number. The remaining 16 octets contain user data. The intermediate confirmed data block is diagrammed in Figure 12.

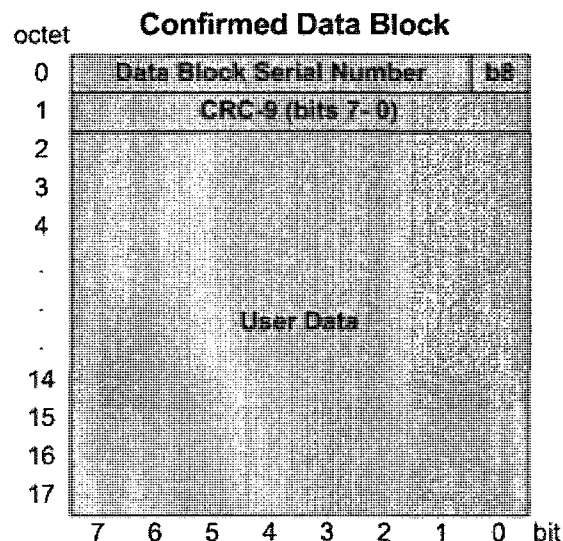


Figure 12 - Confirmed Data Packet Data Block

Serial Number (octet 0, bits 7 - 1) is the number of the block within the entire message. The numbers start at 0 and increment up to L-1, where L is the number of the blocks that follow the header block. The **CRC-9** field (octet 0, bit 0 and octet 1) contains a 9 bit cyclic redundancy check code of the data block. The CRC-9 is calculated by the following procedure. The seven bits of the serial number and the 16 octets of user data are arranged into a 135 bit sequence. The serial number bits are the first seven bits in the sequence. The sequence bits are then taken to be coefficients of a 134 degree message polynomial $M(x)$:

$$M(x) = x^{134} + x^{133} + \dots + x^{128} + x^{127} + \dots + x^1 + x^0 ,$$

where bit 6 of the serial number corresponds to a coefficient of the x^{134} term, bit 0 of the serial number corresponds to the x^{128} term, bit 7 of octet 2 (user data) corresponds to the x^{127} term, and bit 0 of octet 17 corresponds to the x^0 term. The generator polynomial $G_9(x)$, and the inversion polynomial $I_9(x)$ are defined as:

$$G_9(x) = x^9 + x^6 + x^4 + x^3 + 1$$

$$I_9(x) = x^8 + x^7 + x^6 + \dots + x + 1.$$

The CRC-9 polynomial $F_9(x)$ is computed using the following equation:

$$F_9(x) = [x^9 M(x) \text{ mod } G_9(x)] + I_9(x),$$

where mod stands for the modulus operator (the remainder following binary division).

The final CRC-9 polynomial can be written in the following form:

$$F_9(x) = x^8 + x^7 + x^6 + \dots + x^1 + x^0 .$$

The resulting 9 coefficients are CRC-9 bits that protect each confirmed data block. At the transmitter, these bits are placed in the CRC-9 field with the most significant bit corresponding to bit 0 of octet 0, the next most significant bit corresponding to bit 7 of octet 1, and the least significant bit corresponding bit 0 of octet 1.

The last confirmed data block contains up to 12 octets of user data and pad octets and 4 octets of 32 bit CRC code. If the number of pad octets is larger than 12, then the additional pad octets are included in the second to the last data block. The last four octets consist of the message CRC calculated over all used data octets. The 32 bit CRC is generated in the same way as for unconfirmed data packet (section *Unconfirmed Packet Data Block Format*). The last confirmed data block is shown in Figure 13.

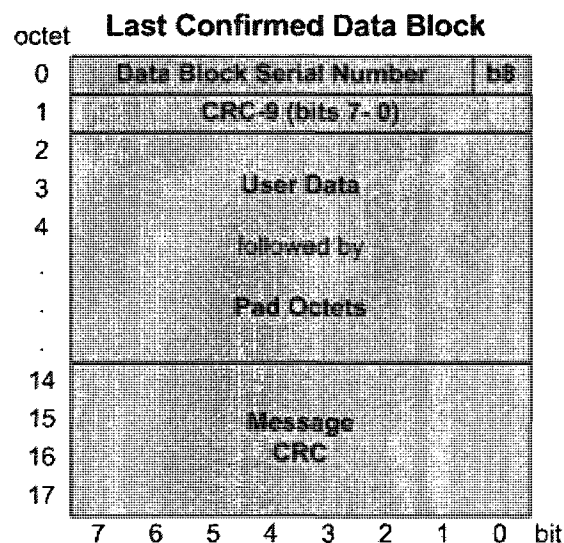


Figure 13 - Confirmed Data Packet Last Data Block

At the receiver, when the confirmed data block is received, the CRC-9 bits are calculated using the serial number and user data octets. If the CRC-9 codes do not match

the block is received with errors and the retransmission of the block is indicated. When the last data block is received, the 32 bit CRC code is calculated over all used data and pad octets. If the 32 bit codes do not match, the entire message is disregarded and the retransmission of all data blocks is indicated. If they match, the data octets are assembled into the original IP packet which is then delivered to the IP data network.

Response Block Format

A response packet is used to confirm delivery for a data packet and to request the retransmission of packets received with errors. In normal operation to confirm delivery, the response packet consists of only the header block. However, in particular applications and to request retransmission, the response packet may have data blocks following the acknowledgment header block. The response data block consists of flag bits and a CRC code. The response data block format is shown in Figure 14.

The flag bits are used to confirm the receipt of the corresponding blocks, when set to 1, and to indicate that the particular block should be retransmitted, when set to 0. The flags can indicate selective retries for up to 64 data blocks. If more flags are necessary, additional response blocks should be used. The CRC is the 32 bit cyclic redundancy check, the same as is used for unconfirmed and confirmed delivery.

The Base Station Main Thread

The main thread of the base station starts the Main CAI and IP input threads and initializes the IP network interface, PTT control, and computer audio card. The flow diagram of the main thread is shown in Figure 15.

The Main CAI packet input thread listens to the audio card input and detects any activity on the radio channel. If there is a signal at the audio card input, the CAI thread sets a busy flag and prepares the receiver for proper reception of the incoming CAI packet. The CAI thread will be explained in more detail in section *The Main CAI Packet Input Thread*.

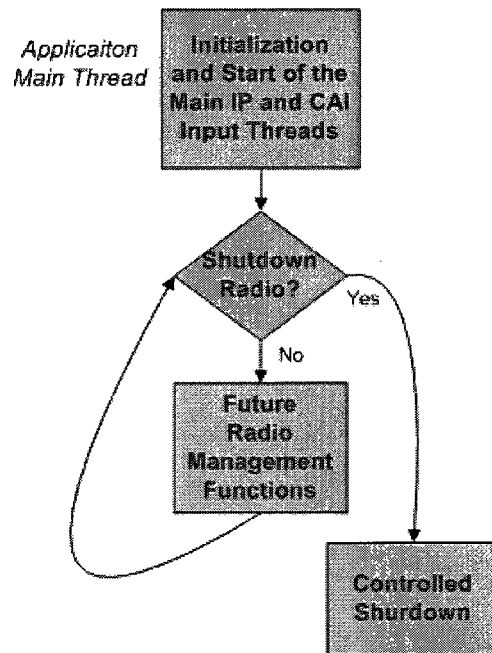


Figure 15 - Flow Diagram of the Base Station Main Thread

The Main IP packet input thread receives IP packets from the network interface and sets the transmitter for proper conversion of IP to CAI packets and transmission of the CAI packets over the radio channel. The IP thread will be further explained in section *The Main IP Packet Input Thread*.

The standard IP network interface is implemented as an IP Reflector. The IP Reflector creates a virtual subnet on the application computer and captures all IP traffic addressed to that subnet by the application running on the same computer. This provides a consistent method for implementing IP data communication over specialized devices (radio modems; in this work, radio modem is the P25 Data Base Station) for which standard Windows network adapter drivers are not available. The IP Reflector consists of two components: a Windows 2000/XP network miniport driver which runs at the kernel level, and a companion Windows service which runs at the OS application level. This is illustrated in Figure 16.

On one side, the IP Reflector communicates with a remote server application which provides the data information through the standard IP network. On the other side, the IP Reflector exchanges UDP packets with the Base Station application on a specific IP Address and Port. The Base Station main thread initializes the UDP/IP communication with the IP Reflector Service through the IP Address and Port.

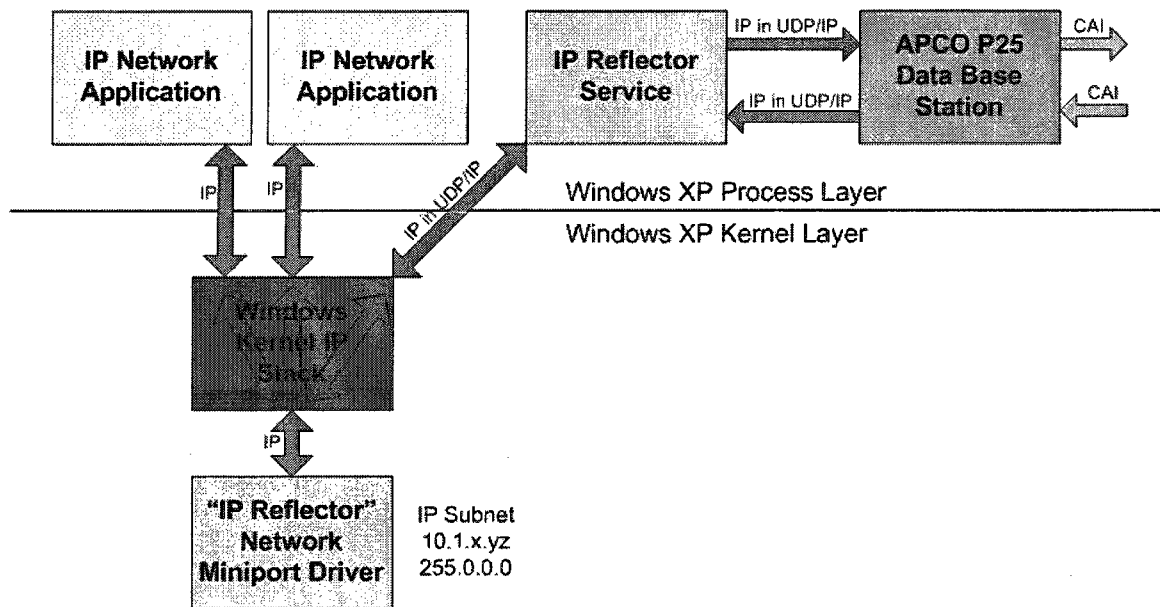


Figure 16 - The IP Reflector Architecture

The analog versions of CAI digital data packets can be transmitted over the radio channel only when the push-to-talk (PTT) signal is active on the analog radio. The application controls the PTT signal on the analog radio by sending digital messages (PTT ON, and PTT OFF) over a serial RS-232 connection. The main thread opens a COM port and initializes the RS-232 communication between the application and the analog radio over a custom radio interface cable (Appendix A).

The CAI waveforms are played and received over the computer sound card. The sound card is controlled using a standard Windows winmm.lib multimedia library. Using the API functions from the multimedia library, the main thread opens the sound card with a sampling rate of 48 kHz and initializes input and output buffers.

The Main CAI Packet Input Thread

The CAI packets at the input of the computer audio card are detected and captured by the main CAI input thread. The CAI packets are then passed to the receiver which performs decoding and assembles the received IP packets. The receiver extracts the address of the physical client radio from the Logical Link ID field of the header block. This address will be referred to as a CAI address. The CAI address is used to identify the client in a client table. If the client is found in the table, the incoming CAI packet is placed into a client specific CAI packet queue. If the CAI address is not found among the client table entries, a new client is added to the client table and a new client thread is created. The flow diagram of the main CAI input thread is shown in Figure 17.

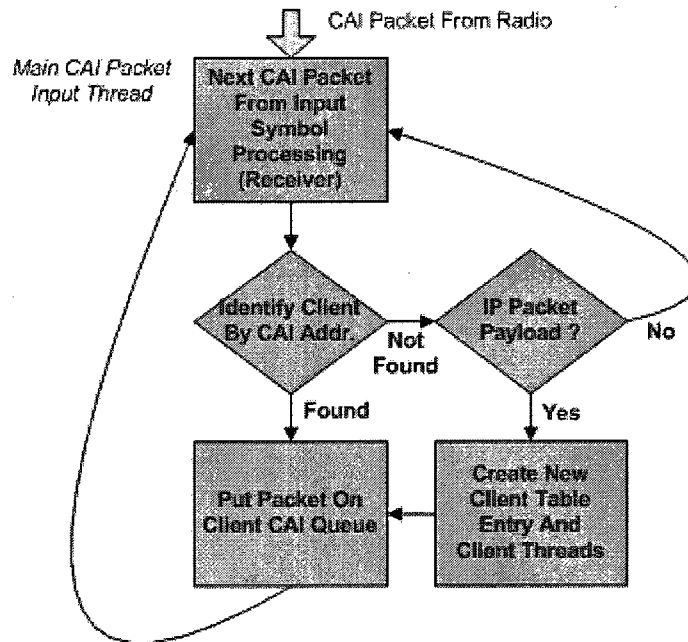
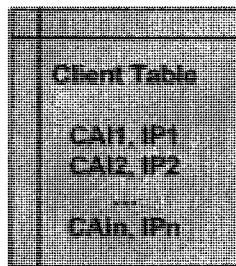


Figure 17 - Flow Diagram of the Main CAI Packet Input Thread

The client table has only two fields for each client. It is used to match the CAI address (Logical Link ID) and the client IP address extracted from the IP packet payload. The table is initially empty and a new entry is made when the first message from a new client is received. The client table is illustrated in Figure 18.



| Client Table | |
|--------------|--|
| CAI1, IP1 | |
| CAI2, IP2 | |
| ... | |
| CAIn, IPn | |

Figure 18 - Client Table

For each new client in the table, a new pair of client threads is created. These two additional threads handle client input CAI packets (client to IP network) and client input IP packets (IP network to client). The flow diagram of the client threads are shown in Figure 19.

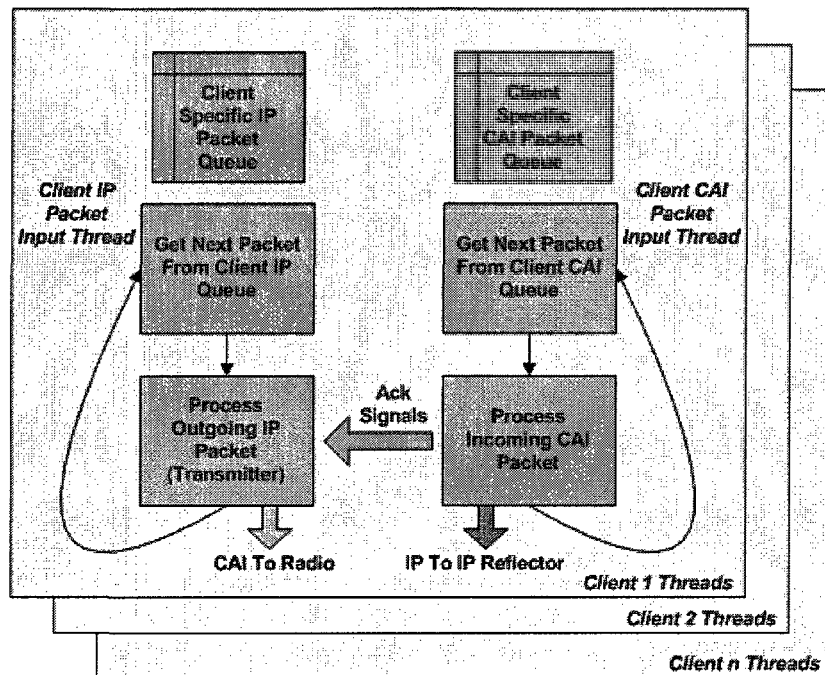


Figure 19 - Flow Diagram of the Client Threads

The client CAI packet input thread takes CAI packets from the client specific CAI packet queue and processes them. The processing involves the proper action based on the type of the packet and the type of delivery. The flow diagram of the processing of the incoming CAI packet is shown in Figure 20. The incoming CAI packet can be a response to a previous confirmed packet delivery or can carry a new incoming IP packet from a radio client. If the packet is the response, it can carry information of a successful or unsuccessful delivery. In both response cases, the packet sequence number is checked and a positive or a negative acknowledgment signal is sent to the transmitter. If the acknowledgement is negative, the transmitter resends the packet a maximum of four times. Otherwise, the transmitter sends the next IP packet from the client specific IP packet queue.

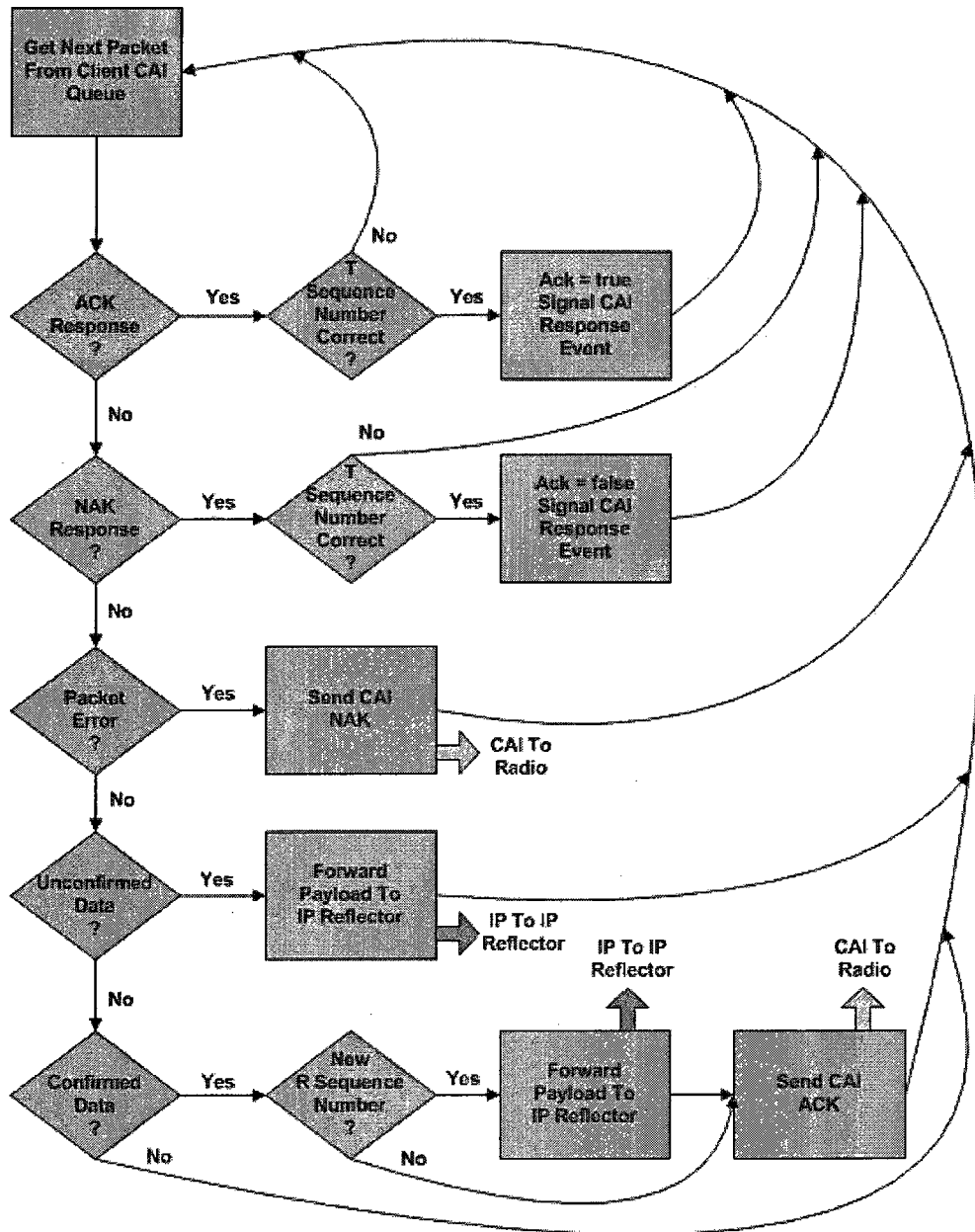


Figure 20 - Flow Diagram of the Client CAI Packet Input Thread

If a new incoming packet is received with an error, due to the noise in the transmission channel, a negative response CAI packet is transmitted to the source client. For the case of unconfirmed data reception, a new IP packet is forwarded to the IP

Reflector. Finally, if a new packet is received with the confirmed delivery and new sequence number, the packet is forwarded to the IP Reflector and the acknowledgement of successful reception is sent to the source client. If the packet is received with an old sequence number, it is a copy which is disregarded, but the response to its reception is sent to the client.

The client IP packet input thread will be explained in the next subsection.

The Main IP Packet Input Thread

The main IP Packet input thread receives IP packets from the IP Reflector service through the specific IP address and Port. For each received IP packet, the thread extracts the destination IP address and tries to identify the client in the client table, shown in Figure 18. If the IP destination address is a valid client table entry, the IP packet is placed on the client specific IP packet queue together with the client CAI address. However, if the IP destination address is not in the client table the IP packet is disregarded. The flow diagram of the main IP packet input thread is shown in Figure 21.

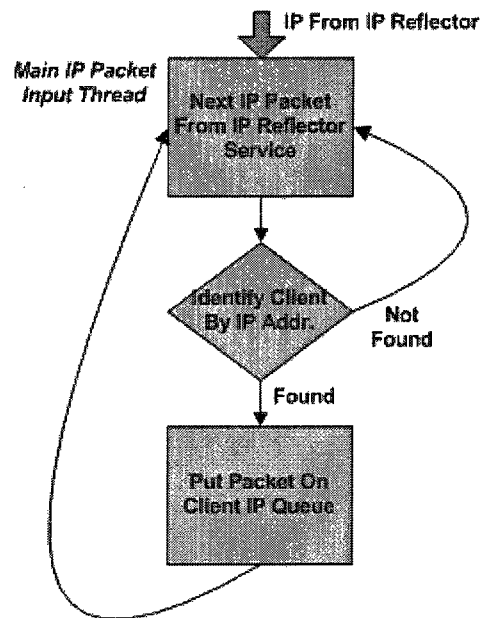


Figure 21 - Flow Diagram of the Main IP Packet Input Thread

The IP packets are taken from the client specific IP packet queue by the client IP packet input thread, shown on left hand side in Figure 19. The client thread processes and prepares IP packets for transmission over the air to the destination radio client. The IP packets are transformed into CAI packets which can be sent with either unconfirmed or confirmed delivery. The flow diagram of the client IP packet input thread is shown in Figure 22.

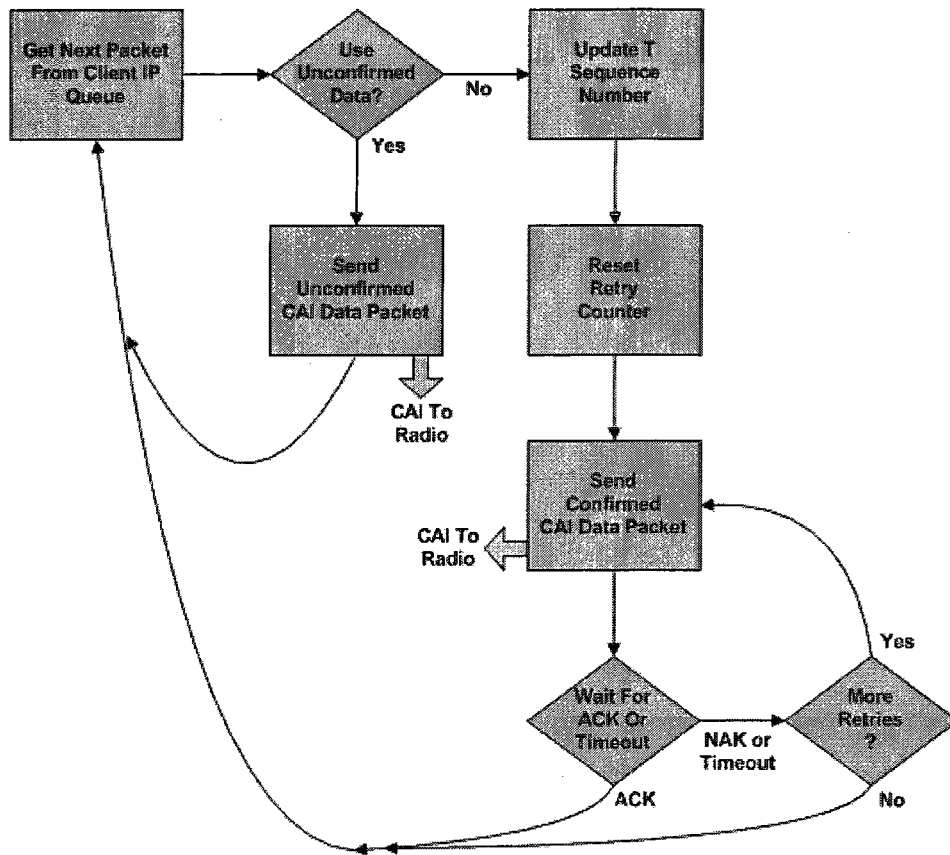


Figure 22 - Flow Diagram of the Client IP Packet Input Thread

For unconfirmed delivery, the IP packet is transformed into a CAI data packet and directly sent to the radio. While unconfirmed delivery does not provide any feedback information about successful transmission of the packets, confirmed packets require a response from the client about the status of the packet transmission. Thus, for a new confirmed packet the sequence number is updated and a retry counter is reset. Then, the CAI packet is sent and the thread waits for the acknowledgement of the reception which is received from a client radio through the client CAI packet input thread. If the negative

acknowledgement packet is received or the timeout event is set, the transmitter retries transmission of the packet. The maximum number of retries is usually set to four.

The transmitter can transmit packets only when the busy signal at the receiver is not set. The busy signal is generated by the receiver when there is any activity on the radio channel. The busy signal indicates when the channel is idle and when the transmitter can use the radio channel to broadcast data packets. Otherwise, the packets will collide with the signals already present on the channel.

The negative acknowledgement response is generated due to either noise in the transmission channel or the collision between multiple packets. Data packets can collide when a client and the base station attempt to transmit packets at approximately the same time. When the transmission of the data packet is started, the packet is transmitted entirely. This is the only time when the packets can collide since the transmitter can broadcast the packets only when the busy signal is not active. Collisions distort the contents of the packets, which is possible to detect, for data packets, at the receiver by performing the cyclic redundancy checks. If any of the checks fail, the negative response packet is broadcasted to indicate the retransmission of the packet.

Voice packets contain a digital representation of 20 ms of an actual voice signal. The distorted voice packets are ignored at the vocoder [32]. This could introduce a noise to the actual voice signals. The influence of packet collisions on both voice and data transmission will be discussed in the next chapter.

CHAPTER IV

TESTING

The most important part of the completion of this work was the testing process. Multiple tests were designed to show data base station compliance with commercially available P25 digital mobile radios, to assure proper functioning of the transmitter, receiver, and base station logic, and to demonstrate integration with a preexisting P25 voice system.

All tests were performed in laboratory conditions with equipment that mimics a setup that can be found in a local police department. The test messages were exchanged between the newly designed data base station and a laboratory setup that mimicked the radio system found in a local police cruiser, called the “mobile client”. Each mobile client consisted of a laptop computer and a commercial Project 25 compliant digital radio (Motorola Astro Spectra or Motorola Astro XTL-5000). The mobile client used in data only tests was called the “data mobile client”.

Each test consisted of 10000 queries which resulted in 20000 messages for unconfirmed and 40000 messages for confirmed delivery. While for unconfirmed delivery there were two physical messages for each request, a request and a requested data packet, for confirmed delivery there were two more response messages to confirm

the reception of the request and the requested data. All queries and data messages were sent through the system in the form of IP packets.

The size of the data packets varied between 12 bytes for the acknowledgment packet and 84 bytes for the requested data packet. The duration of voice test signals varied between 2 and 15 seconds. The change in the rates at which the packets were transmitted resulted in the change of the channel usage from 4% to 35% for data only tests and the channel usage increased from 24% to 50.1% for mixed voice and data signaling. As a reference, the maximum channel usage observed was 22% when monitoring the activity on three separate NH police radio channels for a time period longer than a year. The channel usage during testing was high enough to simulate the utilization of a radio channel used by local NH police departments.

During the testing process, black-box and white-box testing methods were used [38]. The black-box technique was used for initial end-to-end tests. Later, white-box tests were performed to track missing and collided messages. All phases and events on the base station were logged and later analyzed to reveal any unpredicted behavior.

The tests were divided into three stages: initial end-to-end tests with a single mobile client, tests with multiple mobile clients, and tests that show the priority of voice communication.

Tests with a Single Mobile Data Client

Tests with a single mobile data client were used for initial end-to-end communication between the base station and a mobile data client. These initial tests were performed as black box tests by passing IP request packets from a data client to the base station and transmitting requested data from the base station to the client. The initial testing setup is shown in Figure 23.

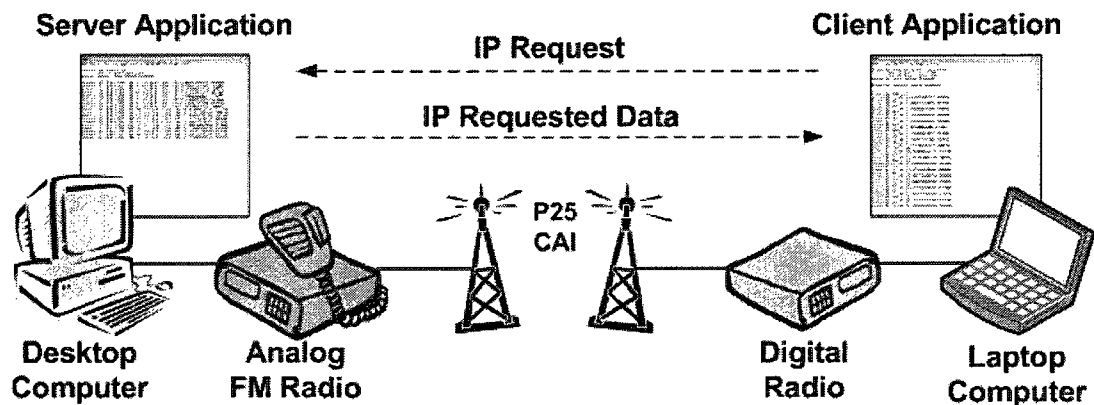


Figure 23 - End-to-end Initial Testing Setup

For testing purposes two test applications were developed: Server and Client. These applications mimicked real world applications that support IP data services, such as the Project54 records application. The client application was used to generate IP request packets and pass them to the digital radio. Each request packet was marked with a request serial number. The request serial numbers were consecutive numbers from 1 to 10000.

The server upon the reception of the request packet performed the search of a data test file based on the requested number and transmitted back the requested data. Each entry of the data test file consisted of a request number and a randomly generated sequence of numbers with a maximum length of 20 bytes, called the requested data.

At the client, the received requested data were saved into a client test file together with the matching request number. However, if the requested data packets were not received in a predefined time interval, the client went on to the next request packet. At the end of the test the server and client test files were compared to indicate any unsuccessful transmission.

For the tests in laboratory conditions, we expected to see no failed transmissions between the base station and the client. However, comparing the data test files, 0.37% of the messages were missing in the client file. These lost messages could be lost request packets, which the server never received, or lost requested data packets, which never reached the client. Therefore, we applied a white-box testing approach to track down the missing packets.

The white-box testing initially involved five log files. The first log file was used to record the events and actions performed by the client application. The other four were used to track packets at the server and record actions by receiver, transmitter, and the base station logic. In addition we recorded digital versions of the packets at the input and the output buffers of the sound card and looked at the analog PTT signal for each transmission on an oscilloscope.

A segment of a log file used to track actions performed by the client application is shown Figure 24. The log file started with six initial lines: a title line, starting date and time, the client (sender) IP address, the server (receiver) IP address, the maximal waiting time for data to be retrieved in milliseconds, and the total number of request packets in the test. The initial lines were followed by two lines for each transmitted request. The first line indicated the number of the request sent by the client and a number of seconds the client was waiting before the transmission. The waiting time was used to change the rate of transmissions and to simulate different channel usage conditions. The second line showed the requested data received from the server preceded by the request number and the client id. Later, the request number and the client id were used in tests with multiple clients to reveal any possible interference between the clients and their packets.

When analyzing the client log file it was easy to identify missing requested data packets, since the second line dedicated to the request would be empty. For each missing packet we looked into a server log file to see if the request packet was received by the server or not.

```

1 P25 Client01 Log File!
2
3 s_time 04/15/09 12:02:21.562
4 s_ip 10.1.0.1:33
5 r_ip 10.1.0.2:36
6 time_out = 15000
7 n_requests = 10000
8
9 1,3
10 c01,1,814723686393.364260,
11 2,5
12 c01,2,905791937075.713380,
13 3,10
14 c01,3,126986816294.379070,
15 4,7
16 c01,4,913375856139.105960,
17 5,12
18 c01,5,632359246225.777100,
19 6,5
20 c01,6,97540405000.311981,
21 7,11
22 c01,7,278498218867.769900,
23 8,6
24 c01,8,546881519205.436950,

```

Figure 24 - A segment of a Client Log File

A segment of a server log file is shown in Figure 25. After the initial title line which indicated the start date and time of the test, each following line corresponded to a request received from the client. The first part of a line indicates the time of reception, the client (sender) IP address, the client id, and the serial number of the request. The second part of the line represents the server response to the received request packet. It consists of the response time, the client IP address, the client id, the request number, and the requested data. If there was an error in transmission from the client, the request packet would not be received by the server and an entire line in the log file would be missing.

```

1 P25 Server Log File 04/15/09 12:23:39.203!
2
3 12:23:57.984,10.1.0.2:36, c01,1, 12:23:57.984,10.1.0.2:36, c01,1,814723686393.364260,
4 12:24:04.718,10.1.0.2:36, c01,2, 12:24:04.718,10.1.0.2:36, c01,2,905791937075.713380,
5 12:24:16.125,10.1.0.2:36, c01,3, 12:24:16.125,10.1.0.2:36, c01,3,126986816294.379070,
6 12:24:24.765,10.1.0.2:36, c01,4, 12:24:24.781,10.1.0.2:36, c01,4,913375856139.105960,
7 12:24:39.31,10.1.0.2:36, c01,5, 12:24:39.31,10.1.0.2:36, c01,5,632359246225.777100,
8 12:24:45.421,10.1.0.2:36, c01,6, 12:24:45.421,10.1.0.2:36, c01,6,97540405000.311981,
9 12:24:58.328,10.1.0.2:36, c01,7, 12:24:58.328,10.1.0.2:36, c01,7,278498218867.769900,
10 12:25:06.296,10.1.0.2:36, c01,8, 12:25:06.296,10.1.0.2:36, c01,8,546881519205.436950,
11 12:25:11.750,10.1.0.2:36, c01,9, 12:25:11.750,10.1.0.2:36, c01,9,957506835434.340090,
12 12:25:17.859,10.1.0.2:36, c01,10, 12:25:17.859,10.1.0.2:36, c01,10,964888535199.311650,
13 12:25:34.62,10.1.0.2:36, c01,11, 12:25:34.62,10.1.0.2:36, c01,11,157613081678.390660,
14 12:25:39.859,10.1.0.2:36, c01,12, 12:25:39.859,10.1.0.2:36, c01,12,970592781760.645140,
15 12:25:44.359,10.1.0.2:36, c01,13, 12:25:44.359,10.1.0.2:36, c01,13,957166948242.988400,
16 12:25:55.46,10.1.0.2:36, c01,14, 12:25:55.46,10.1.0.2:36, c01,14,485375648723.355830,
17 12:26:06.640,10.1.0.2:36, c01,15, 12:26:06.640,10.1.0.2:36, c01,15,800280468888.999880,
18 12:26:12.921,10.1.0.2:36, c01,16, 12:26:12.921,10.1.0.2:36, c01,16,141886338628.073460,
19 12:26:22.750,10.1.0.2:36, c01,17, 12:26:22.750,10.1.0.2:36, c01,17,421761282626.853210,
20 12:26:27.750,10.1.0.2:36, c01,18, 12:26:27.750,10.1.0.2:36, c01,18,915735525189.151370,
21 12:26:41.390,10.1.0.2:36, c01,19, 12:26:41.390,10.1.0.2:36, c01,19,792207329559.762210,
22 12:26:51.968,10.1.0.2:36, c01,20, 12:26:51.968,10.1.0.2:36, c01,20,959492426392.943480,
23 12:27:06.859,10.1.0.2:36, c01,21, 12:27:06.859,10.1.0.2:36, c01,21,655740699156.931150,
24 12:27:22.671,10.1.0.2:36, c01,22, 12:27:22.671,10.1.0.2:36, c01,22,35711678575.153839,

```

Figure 25 - A segment of a Server Log File

Analyzing the server log files, it was observed that for approximately 35% of the lost messages their request packets were not received by the server. For the rest of the lost messages, the request packets were successfully received by the server and the requested data were sent to the base station but never received by the client. In both cases we looked at a data radio log file used to record the actions performed by the base station.

For each lost packet in the system, it was possible to find the time, or at least an approximate time, of the transmission by looking at the previous and the next received packets in the server log file. When the time of transmission was found, we searched the data radio log file for more details about sent and received packets at the data base station. A segment of a data radio log file is shown in Figure 26. Each line in the data radio log file started with the time of the transmission followed by type (received or sent),

type of delivery (confirmed, unconfirmed, or response), and additional information about sequence number and retry counter.

```
1 12:23:31.234, P25 Data Radio Log File 04/15/09!
2
3 12:23:57.296, r, c0, confirmed, sequenceNumber -1 = 3,
4 12:23:57.984, s, c0, confirmed_ack,
5 12:23:58.750, s, c0, confirmed, retry = 0,
6 12:23:58.953, r, c0, response_ack,
7 12:24:04.468, r, c0, confirmed, sequenceNumber 3 = 4,
8 12:24:04.718, s, c0, confirmed_ack,
9 12:24:05.484, s, c0, confirmed, retry = 0,
10 12:24:05.656, r, c0, response_ack,
11 12:24:15.859, r, c0, confirmed, sequenceNumber 4 = 5,
12 12:24:16.125, s, c0, confirmed_ack,
13 12:24:16.890, s, c0, confirmed, retry = 0,
14 12:24:17.78 , r, c0, response_ack,
15 12:24:24.484, r, c0, confirmed, sequenceNumber 5 = 6,
16 12:24:24.765, s, c0, confirmed_ack,
17 12:24:25.546, s, c0, confirmed, retry = 0,
18 12:24:25.671, r, c0, response_ack,
19 12:24:38.765, r, c0, confirmed, sequenceNumber 6 = 7,
20 12:24:39.31 , s, c0, confirmed_ack,
21 12:24:39.859, s, c0, confirmed, retry = 0,
22 12:24:39.968, r, c0, response_ack,
23 12:24:45.171, r, c0, confirmed, sequenceNumber 7 = 0,
24 12:24:45.421, s, c0, confirmed_ack,
```

Figure 26 - A segment of a Data Radio Log File

For example, the first four data lines in the data radio log file shown above were four messages that corresponded to one request with confirmed delivery. When a request packet was received with a new sequence number (3), different than the previous one (initially -1), the base station generated the response packet (ack) to confirm the successful reception of the request packet. When the requested data were retrieved from the server, the base station transmitted the data and set the retry counter to 0, indicating the first transmission. The response packet (ack) from the client concluded the successful delivery of the requested data. If the response packet from the client was not received in a predefined time (usually 15 seconds), the base station would have transmitted the data again and have incremented the retry counter.

Analyzing the data radio log file, it was found that the missing request packets at the server were never received by the base station. Also, the requested data packets not received by the client were transmitted by the base station. An additional two log files were used to provide more information about actions performed by the receiver and transmitter.

In the cases when the request packets were not received by the base station, the actions performed by the receiver were logged into a data radio receiver log file. A segment of a data radio receiver log file is shown in Figure 27. The receiver log file contained information about the reception process explained in CHAPTER III.

```
1 12:23:31.234, P25 Data Radio Receiver Log File 04/15/09!
2
3 12:23:57.156, sync_detected,
4 12:23:57.171, fs_detected,
5 12:23:57.171, nid_detected, duid = 12,
6 12:23:57.265, header_confirmed,
7 12:23:57.296, confirmed_packet,
8 12:23:57.296, sync_lost,
9 12:23:58.875, sync_detected,
10 12:23:58.875, fs_detected,
11 12:23:58.875, nid_detected, duid = 12,
12 12:23:58.953, header_response,
13 12:23:58.953, response_packet,
14 12:23:58.953, sync_lost,
15 12:24:04.265, sync_detected,
16 12:24:04.343, fs_detected,
17 12:24:04.359, nid_detected, duid = 12,
18 12:24:04.359, header_confirmed,
19 12:24:04.468, confirmed_packet,
20 12:24:04.468, sync_lost,
21 12:24:05.578, sync_detected,
22 12:24:05.656, fs_detected,
23 12:24:05.656, nid_detected, duid = 12,
24 12:24:05.656, header_response,
```

Figure 27 - A segment of a Data Radio Receiver Log File

The first six lines, after the title, in the receiver log file shown above were the steps that the receiver performed to properly receive a request packet. Before the packet was received, the receiver had to get in synchronization with the incoming clock synchronization word. When the receiver was synchronized, the frame synchronization and network identifier were received. The network identifier was used to separate voice and data messages. Only data messages, messages with data unit id equal to 12, were received while other messages were discarded. When the entire packet was received, the receiver lost synchronization and went to the initial state to wait for the next incoming packet.

Tracing down the lost request packets, it was observed that the receiver never synchronized with the clock synchronization word. The reason for that was a short clock synch word, with initial length set to 40 milliseconds. The short synch word didn't provide enough samples for the receiver to reliably get in synchronization before the rest of the packet was received. Changing the duration to 60 milliseconds improved the reception to the maximum rate (100%).

In the case when the base station sent the data packets, but they were not received by the client, we looked at the data radio transmitter log file. A segment of a data radio transmitter log file is shown in Figure 28. It showed all actions performed by the transmitter.

```

1 12:23:31.234, P25 Data Radio Transmitter Log File 04/15/09!
2
3 12:23:57.296, start sending, llid = 54, sn = 3,
4 12:23:57.796,  s1, prepare ack packet,
5 12:23:57.812,  pl, start transmitting message,
6 12:23:57.812,  pl, activate output,
7 12:23:57.812,  pl, activate PTT,
8 12:23:57.812,  pl, play soud buffer,
9 12:23:57.812,  pl, wait for sound to be played,
10 12:23:57.953,  pl, deactivate output,
11 12:23:57.953,  pl, release PTT,
12 12:23:57.968,  pl, end transmitting message,
13 12:23:57.984, end sending, llid = 54, sn = 3
14 12:23:57.984, start sending, llid = 54, sn = 1,
15 12:23:58.484,  s1, prepare confirmed packet,
16 12:23:58.500,  pl, start transmitting message,
17 12:23:58.500,  pl, activate output,
18 12:23:58.500,  pl, activate PTT,
19 12:23:58.515,  pl, play soud buffer,
20 12:23:58.515,  pl, wait for sound to be played,
21 12:23:58.734,  pl, deactivate output,
22 12:23:58.734,  pl, release PTT,
23 12:23:58.734,  pl, end transmitting message,
24 12:23:58.750, end sending, llid = 54, sn = 1
25 12:24:04.468, start sending, llid = 54, sn = 4,
26 12:24:04.468,  os, wait while input or output is busy,
27 12:24:04.531,  s1, prepare ack packet,
28 12:24:04.546,  pl, start transmitting message,

```

Figure 28 - A segment of a Data Radio Transmitter Log File

Before the data packets were played through sound card, the data base station had to activate the PTT signal on the analog radio. The PTT signal was set through a serial computer port and it was kept active until the entire packet was played. The transmitter log file showed that the base station responded to all received requests and properly transmitted all data packets.

Analyzing the packets at the output of the sound card and analog PTT signal on an oscilloscope, it was observed that the client radio was not receiving some of the packets due to the time when the PTT signal was set and released. The PTT signal was not set in advance enough before the sound card buffer was played, which resulted in

cutting off the first part of the packet. Also, for some packets the PTT signal was not held long enough after the buffer was played and the client received uncompleted packets which were discarded. To resolve this, the PTT was set in advance, 20 milliseconds before the packet was passed through the audio card, and it was held for an additional 50 milliseconds after the signal was transmitted. An example of a data packet is shown in Figure 29.

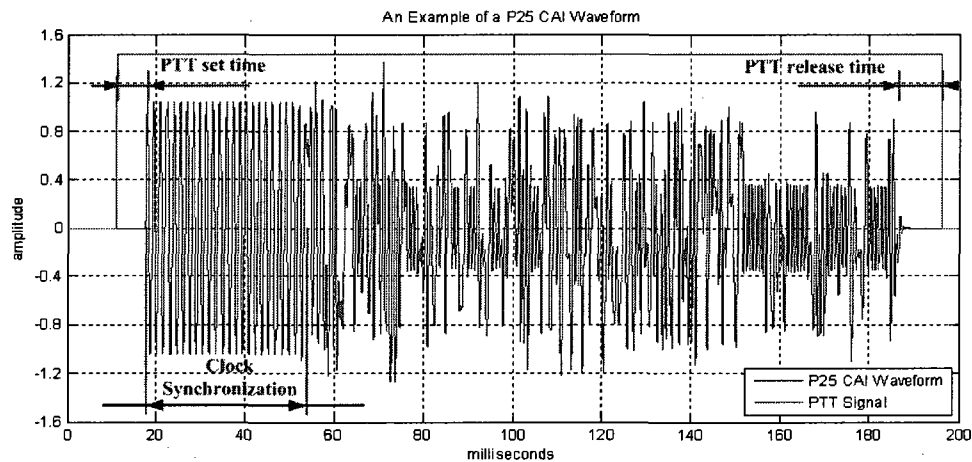


Figure 29 - An Example of a P25 CAI Waveform

In his thesis, Ramsey [9] reported that during the testing of the transmitter a small number of transmitted messages were not received by the client. He argued that those messages, 0.0099% of all test messages, were not received because the software occasionally did not get the CPU time needed to handshake with the client radio. After the testing procedure that we performed with a single radio client, it is more likely that those messages were not received by the client because the PTT signal was not held long enough.

When the PTT signal was properly adjusted to enable the reception of the entire packet at the receiver, a set of ten additional tests, each consisting of 10000 requests, showed successful reception and transmission of all data packets. The successful data signaling with one radio client led to the next testing stage which introduced an additional radio client to the system.

Tests with Multiple Mobile Data Clients

When the tests with a single data client were successfully completed, an additional data client was added to the testing system. The additional client consisted of the same hardware components as well as the same client application as used in previous tests with a single client. During this testing stage, 21 data tests were performed with both clients communicating with the server simultaneously. Each test consisted of 20000 request packets, 10000 requests per client, which resulted in 40000 messages for unconfirmed and 80000 messages for confirmed type of delivery.

The rate at which data messages were transmitted was varied by introducing a pause between requests at each client. The changing rates of data transmissions resulted in changes of channel utilization. Also, an addition of a new radio client into the system resulted in an increase of channel utilization. Channel utilization was varied between 4% and 35% during the data tests with two radio clients.

Since the clients can transmit messages simultaneously, we expected to see a small number of messages collided with each other. Collision happened when the clients or the base station attempted to transmit at approximately the same time. The signals interfered with each other and the content of the packets was distorted. The distortion of the packet resulted in unsuccessful reception of data packets. However, it was possible to detect collisions and use them to test the repetition mechanism at the data base station.

There were two distinguishable types of collisions. The first one occurred when messages were transmitted by mobile clients at the same time. The interfered message signals received by the base station were, most of the time, so badly distorted that the base station couldn't decode any information from the received signal. Since the base station had no information about the sender of the messages, it could not send negative acknowledgement response packets. The base station disregarded the received signal and waited for repeated or new data packets. At the client, if response to the request packet was not received in a predefined time, the request packet was retransmitted.

Analyzing the log files as it was explained in the section *Tests with a Single Mobile Data Client*, it was found that collided messages which were not received by the base station were transmitted from the clients in a 40 milliseconds interval. This interval represented the time necessary for a client to recognize activity on the channel. When activity was detected prior to starting transmission, the client did not attempt to transmit until the channel was available again. Therefore, the collisions happened only when both

clients started the transmission of messages within an approximately 40 millisecond interval.

The second type of collision happened when the base station and one of the clients were transmitting at the same time. In this case, the interval at which collisions happened was slightly longer, a maximum of 60 milliseconds. This indicated that the base station needed a longer time to recognize activity on the channel. The data radio was waiting to get in synchronization (to receive a clock synchronization word) with the received signal to detect activity on the channel. This took a slightly longer time, but it was more accurate than setting up a simple threshold and checking the amplitude of the received samples, which could be triggered by any noise signal larger than the threshold. When the base station started a transmission, it transmitted the entire packet even when a collision happened, due to the blocking state of the winmm.lib system library function used to play data packets.

In order to find the most likely time and place when the second type of collision was occurring, an additional collision log file was created. The collision log file was used to record input activities on the receiver and output activities on the transmitter. A segment of a collision log file is shown in Figure 30. The states of both the receiver and transmitter were logged in the same file to reveal any overlap between the receiving and transmitting signals. From the log file, it was observed that there was no output activity on the transmitter after the receiver detected the presence of the input signal. In other words, the base station transmitter did not interrupt any incoming signals after they were

detected by the receiver. The collisions happened when the base station and client attempted to transmit in the same approximately 60 millisecond interval. The interval represents the time necessary for receiver to get in synchronization with an incoming signal.

| | | |
|----|---------------|---|
| 1 | 11:40:07.921, | P25 Data Radio In/Out Log File, 03/27/09: |
| 2 | | |
| 3 | 11:40:22.953, | <u>in on</u> |
| 4 | 11:40:23.078, | <u>in off</u> Request 1 |
| 5 | 11:40:23.593, | <u>out on</u> |
| 6 | 11:40:23.750, | <u>out off</u> Ack |
| 7 | 11:40:24.296, | <u>out on</u> |
| 8 | 11:40:24.546, | <u>out off</u> Data |
| 9 | 11:40:24.656, | <u>in on</u> |
| 10 | 11:40:24.750, | <u>in off</u> Ack |
| 11 | 11:40:25.750, | <u>in on</u> |
| 12 | 11:40:25.953, | <u>in off</u> Request 2 |
| 13 | 11:40:26.015, | <u>out on</u> |
| 14 | 11:40:26.171, | <u>out off</u> Ack |
| 15 | 11:40:26.718, | <u>out on</u> |
| 16 | 11:40:26.953, | <u>out off</u> Data |
| 17 | 11:40:27.453, | <u>in on</u> |
| 18 | 11:40:27.562, | <u>in off</u> Ack |
| 19 | 11:40:28.156, | <u>in on</u> |
| 20 | 11:40:28.359, | <u>in off</u> Request 3 |
| 21 | 11:40:28.437, | <u>out on</u> |
| 22 | 11:40:28.593, | <u>out off</u> Ack |
| 23 | 11:40:29.140, | <u>out on</u> |
| 24 | 11:40:29.375, | <u>out off</u> Data |
| 25 | 11:40:29.468, | <u>in on</u> |
| 26 | 11:40:29.546, | <u>in off</u> Ack |

Figure 30 - An example of Data Radio In/Out Log File

The number of collisions depended on channel utilization and varied between 0.015% for the channel usage of 4% and 1.61% for the channel usage of 35%. To prevent the loss of data messages, the collided packets were retransmitted. With the confirmed type of delivery, collisions were detected by waiting for the response packet. If the response to the successful delivery was not received in a predefined time, the base station or mobile client assumed that the packet was lost and performed the retransmission. The packets were retransmitted a maximum of 4 times in a 15 second interval. During the

tests, it was observed that using a constant time of 15 seconds on all clients produced successive collisions. When a collision happened, both clients waited the same time and retransmitted the packet again at approximately the same time. To avoid consecutive collisions we introduced randomly selected waiting times at the clients and the base station. The waiting times at the clients were set through the radios' code plug settings.

The unconfirmed type of delivery does not provide any possible way to detect lost packets in the radio channel. However, it was possible to implement the similar repetition mechanism at the application level for unconfirmed messages. With a repetition mechanism all collided messages were repeated and delivered successfully. The last ten tests showed that all data messages were delivered for both confirmed and unconfirmed types of delivery.

Increasing the rate of transmissions or adding a new radio to the system increases the radio channel usage which results in a higher number of collisions. Collisions cannot be avoided in systems with multiple clients and collision handling involves waiting times and retransmitted messages. This decreases the speed and performance of a communication channel. A similar effect could be seen with mixed voice and data signaling. While collided data messages can be retransmitted, the voice packet cannot be repeated and information contained in voice packets would be lost. Therefore, collided voice packets can degrade the quality of voice communication. Since voice communication has priority on the radio channel, any interruption of voice packets by data signaling cannot be accepted. These effects will be examined in the next subsection.

Voice Priority Assurance

When the tests with multiple data clients were successfully completed, the next testing stage was designed to assure the voice priority. The voice priority was tested by mixing voice and data signaling in a radio channel. Initially, voice messages were exchanged between two voice clients and data messages were exchanged between the base station and a single data client. Later, a voice and a data client were merged together into a single voice-data client that supports both voice and data communication. The voice-data client was used to mimic an actual communication system found in a police cruiser.

Hence, voice assurance tests were divided into two groups: tests with parallel voice and data radio communication and tests with a voice-data client used for simultaneous voice and data transmissions.

Potential collisions between voice and data messages could degrade the quality of voice communication. However, before we examine the potential influence of data messages to the voice communication, we will examine how voice is handled and transmitted over a radio channel.

Project 25 Vocoder

The Project 25 standard uses a voice coder, so called vocoder, for voice encoding and decoding. The vocoder is based on the Improved Multi-Band Excitation (IMBE)

voice coding algorithm [32]. Through speech analysis, the IMBE encoder estimates three speech model parameters: the pitch (fundamental frequency), the voiced/unvoiced periods of speech, and the spectral amplitudes that characterize a spectral envelope. These parameters are then quantized, encrypted, encoded, and formed into a bit stream. At the decoder, the speech parameters are decoded, decrypted, reconstructed, and used for speech synthesis. Following the standard, if the decoder cannot decode the speech model parameters, the speech synthesis block will mute its output. The block diagram of the Project 25 vocoder is shown in Figure 31.

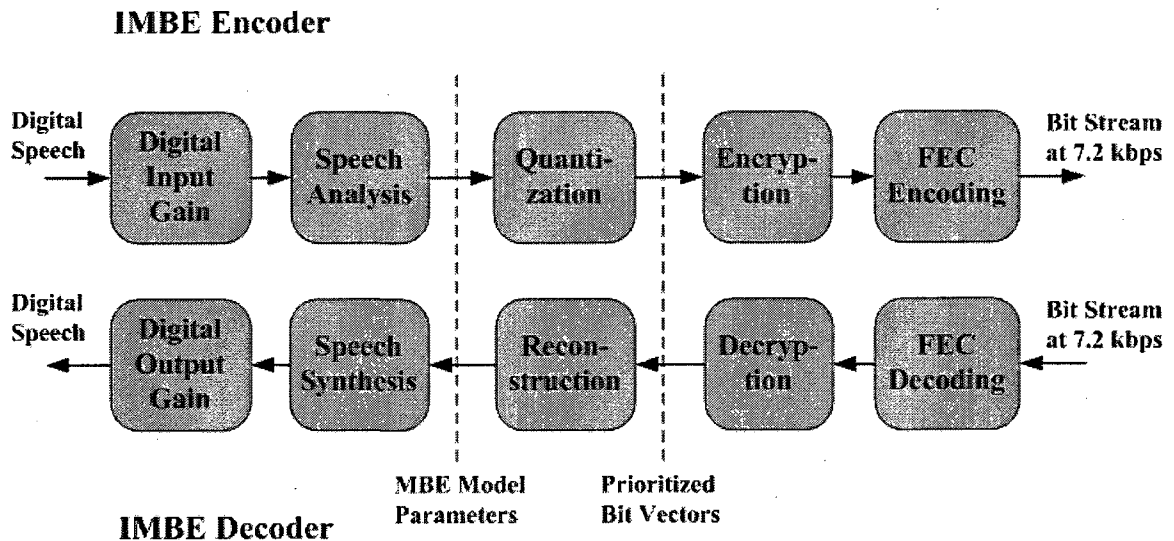


Figure 31 - Block Diagram of the Project 25 Vocoder

For testing purposes, we were interested in any distortion of audio signals introduced by the vocoder, which could be mistaken with a possible influence of data signaling. Therefore, five different test signals were passed through the Project 25

vocoder (both encoder and decoder) a thousand times. Signals at the output of the decoder were recorded and compared with an originally played test signal.

First, four different utterances of approximate length between 2 and 15 seconds were recorded and used as input audio signals. They were passed through a laboratory setup which consisted of two Project 25 mobile radios, each connected to a computer. The setup is illustrated in Figure 32. The utterances were played at the first computer and passed to the first digital radio through an audio card. The first radio encoded (performed speech analysis) the speech signals and transmitted them to the second radio over a radio channel. The second radio decoded (performed speech synthesis) the received signals and passed them to the second computer. Finally, received signals were recorded at the second computer.

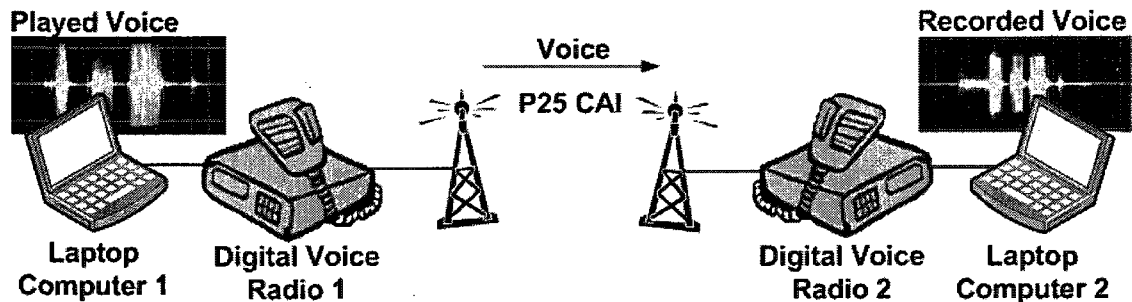


Figure 32 - Project 25 Voice Recording Setup

The original and recorded voice signals were compared in both the time and frequency domains. This is shown in Figure 33. By comparing signals in the time domain, it was obvious that vocoder distorted the amplitude of the signal. The difference between signals was much larger in the frequency domain. Those differences were

caused by the vocoder speech analysis and synthesis algorithms. The analysis algorithm was estimating speech model parameters for every 20 milliseconds of the signal. After estimation, the parameters were refined, enhanced, and smoothed. Such operations defined by the standard caused the speech model parameters used for speech synthesis to differ from the estimated ones. In addition, the estimation of the parameters was not consistent and each recorded signal was more or less different than any previous one. Those inconsistent differences could easily mask any influence of data messages.

Usually, the minimum algorithmic delay between input and output of the vocoder is 80 milliseconds. In practice this delay is longer due to the additional delays at each digital radio. However, in Figure 33 the signals are aligned for easier comparison. The 1024 point FFTs shown in the figure are averaged over the whole signal excluding the silence before and after the signal.

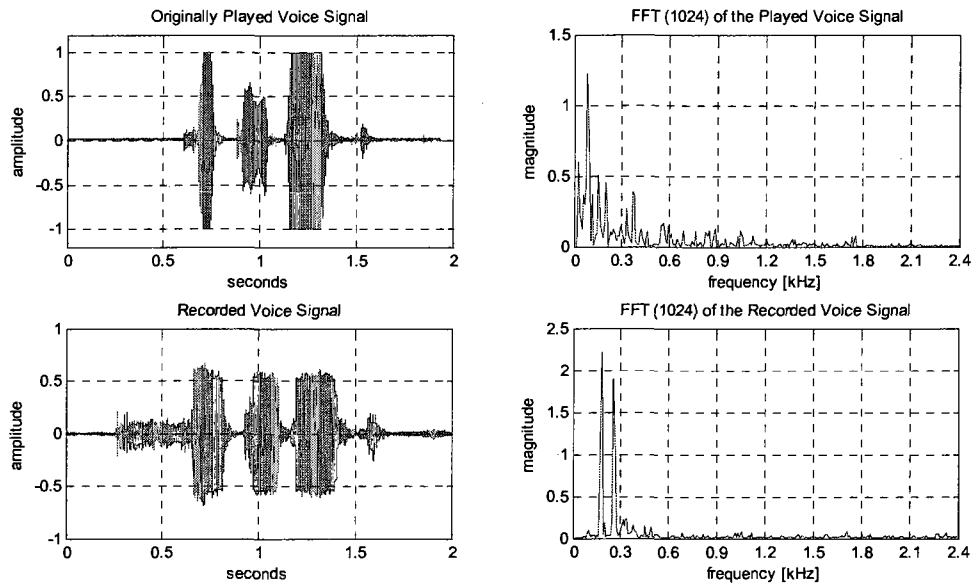


Figure 33 - Comparison of Played and Recorded Signals ("Troop B Boston") in the Time and Frequency Domains

To simplify comparison of the input and output signals, we decided to use sine wave signals of a single frequency. We experimented with frequencies in the range from 100 Hz to 3 kHz. For all sine wave test signals the differences between original and recorded signals were observed. The differences were greater for signals of a higher frequency. An example of a played and recorded sine wave of 0.3 kHz is shown in Figure 34. As it can be seen in the figure, the vocoder introduced side frequency components around the originally played signal.

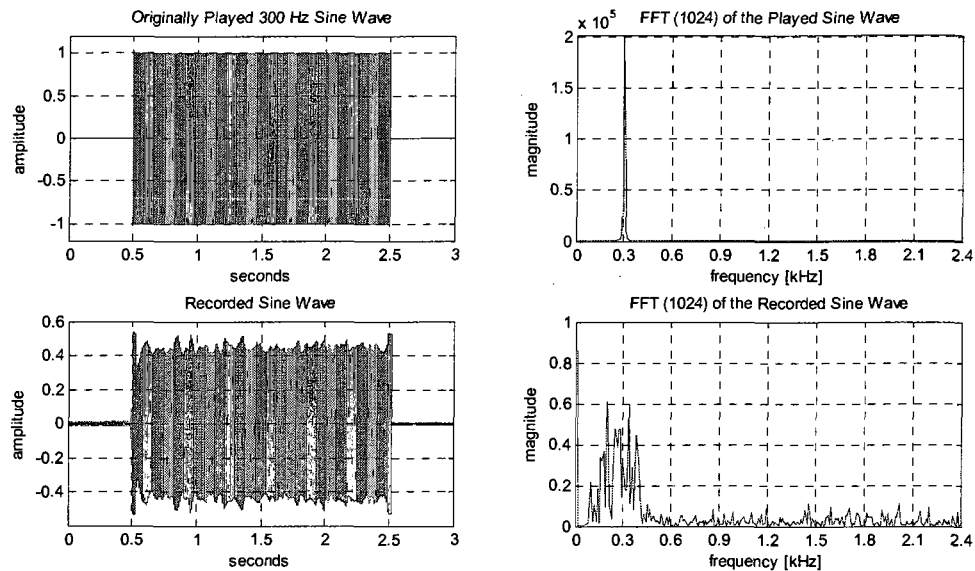


Figure 34 - Comparison of Played and Recorded 300 Hz Sine Wave in the Time and Frequency Domains

In tests with mixed voice and data signaling, five audio test signals were used. The first three test signals were utterances: “Kittery”, “Troop Up”, and “Troop B Boston” (Figure 33). The fourth test signal was a 15 second segment of a larger audio file. Lastly, the fifth test signal was a sine wave of 300 Hz as shown in Figure 34. The test files were passed through the laboratory setup, shown in Figure 32, a thousand times. The recorded signals were used as a reference and were compared with the audio test signals passed through the system in parallel with data packets. Since the frequency components of the audio signals were distorted by the vocoder, the test signals were analyzed in the time domain by comparing amplitudes and calculating cross-correlation with the referent test signals.

Tests with Parallel Voice and Data Communication

The laboratory setup used in the tests with mixed voice and data transmissions is illustrated in Figure 35. Voice signals were played at the first voice client, passed through a radio channel, and recorded at the second voice client. The voice part of the setup is the same as shown in Figure 32. A data client was used for initiating data requests which were transmitted to the base station and server application over the same radio channel. The data part of the setup is the same as the setup used for initial end-to-end tests (Figure 23). These tests were designed to reveal any potential influence of data signaling or collisions of the voice and data messages on the quality of voice communication. Each test consisted of a thousand data requests and a thousand retransmissions of one of the five voice test signals. For each voice test signal two tests were performed.

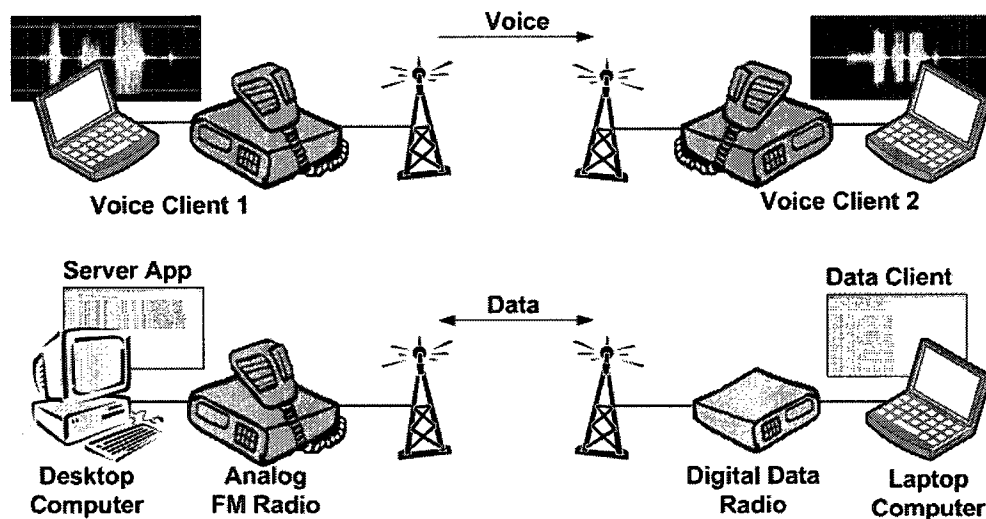


Figure 35 - Parallel Voice and Data Communication over the same Radio Channel

In the tests with mixed voice and data communication, the channel utilization was varied by introducing a pause between data requests at the data client and a pause between transmissions of the voice test signal at the first voice client. The channel utilization for data signaling was between 6.1% and 13.8%, and for voice signaling was between 20.2% and 35.9%. Therefore, the total channel utilization for parallel voice and data communication was varied between 26.3% and 49.7%.

During the tests all data messages were successfully transmitted and received. Analyzing the data log file, as it was explained in the section *Tests with a Single Mobile Data Client*, the number of collided data messages on average (over 10 tests) was around 1%. However, those collisions were properly handled by the base station and they were successfully retransmitted.

All transmitted voice signals were successfully recorded on the second voice client radio. An example of a recorded 300 Hz sine wave is shown in the top part of Figure 36. The recorded voice signals were analyzed after the test in the time domain. First, the amplitude of each recorded signal was compared to the amplitude of a reference signal (middle of Figure 36). The envelope of the reference signal was calculated and minimum and maximum values of the envelope were used as reference points (green lines). The envelope of the recorded signals should lie in-between the reference points (red signal). Any recorded signal for which the envelope crossed the reference points would be marked as a signal with unexpected distortion.

Second, the cross-correlation of each recorded signal was calculated. The cross-correlation was computed using the reference signals recorded without data signaling on the channel. An example of a cross-correlation is shown in the bottom part of Figure 36. For both amplitude and cross-correlation analysis a combination of three reference signals were used: originally played signal, any of the recorded signals described in the section *Project 25 Vocoder*, and an average of a thousand recorded signals as described in section *Project 25 Vocoder*.

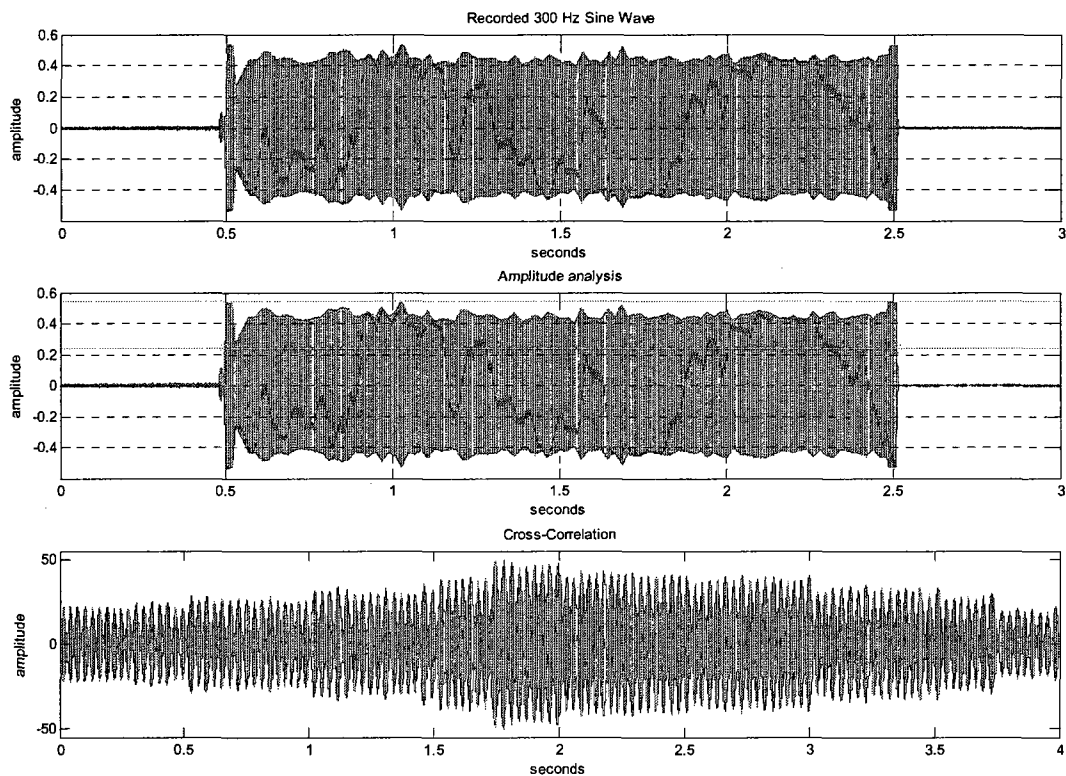


Figure 36 - An Example of a Recorded Signal, an Amplitude Analysis, and a Cross-correlation

Analysis of the amplitude and cross-correlation did not reveal any undesired and unexpected distortion of the audio test signals. Here we were looking for missing parts of the recorded signals caused by the collisions with data messages. The payload of collided voice packets would be lost and the vocoder should disregard those packets. Without speech model parameters, the vocoder would mute its output until the next voice packet with the parameters was received. The envelopes of the recorded signals were in the range of the reference points and the cross-correlation did not have any distortions.

Since we knew that approximately 1% of the data messages were collided, we extracted more information about collisions from the data log files. Tracking the collided data messages, it was possible to find the recorded voice signal with which the data messages were collided. Analyzing those recorded voice signals in particular did not reveal any overlooked distortions. Playing those signals on a computer speaker did not reveal any noticeable distortion of the audio signals as well.

The explanation of no influence of collisions on the test audio signals is twofold. First, by analyzing the data collision log file, section *Tests with Multiple Mobile Data Clients* - Figure 30, it was observed that there were no data transmissions when voice activity was detected on the channel. This means that collisions could happen only when voice and data message were transmitted at approximately the same time. Therefore, the collisions could influence only the first part of the voice signals. Second, the first part of the voice test signals, 500 milliseconds, was a silence and the maximum duration of the data packets used in the tests was 250 milliseconds. Thus, the collisions which only

occurred at the beginning of the voice transmission caused the vocoder to disregard voice packets and produced no signal at the output which matched the initial silence of the original signal. A half second of silence before and after the voice signal was used to simulate real voice communication on the radio channel, simulating pauses between operating a PTT button and actual speech.

Tests with Simultaneous Voice and Data Transmissions from a Single Client

When the tests with parallel voice and data signaling over the same radio channel were successfully completed, the final tests with a single voice-data client were performed. In these tests a single mobile radio client was used for both voice and data transmissions. A single mobile radio was used to mimic an actual radio setup found in local police cruisers. The laboratory setup used for these tests is illustrated in Figure 37.

These tests were performed two times with the same test signals described in the section *Tests with Parallel Voice and Data Communication*. Firstly, the voice signals were played at the first voice client and recorded at the mobile client. Secondly, the voice signals were transmitted in opposite direction from the mobile client to the first voice client. In both scenarios, data messages were transmitted back and forth between the base station and the mobile client.

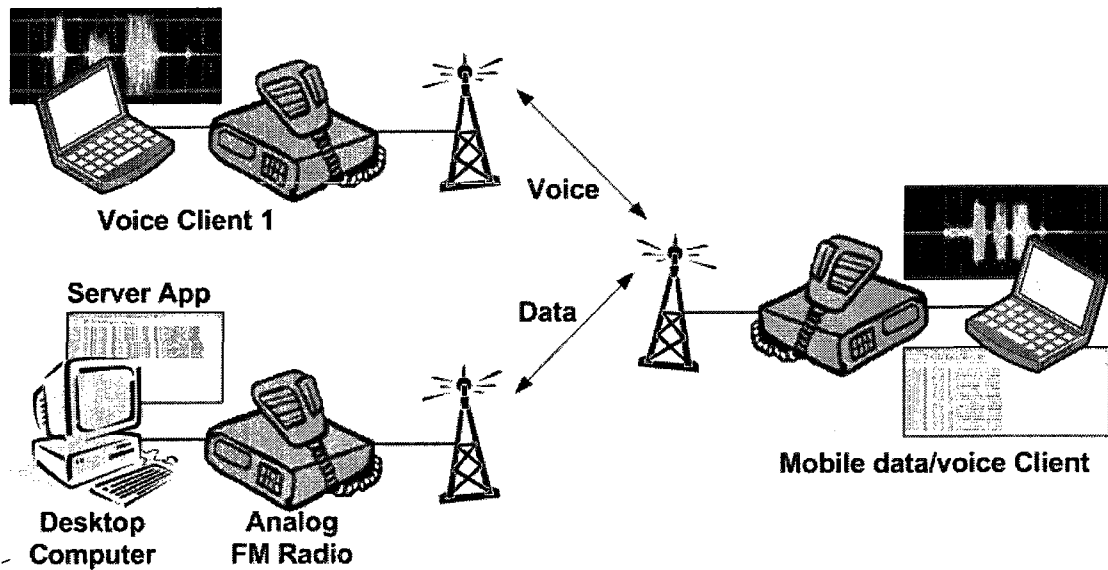


Figure 37 - The Final Testing Setup with a Single Mobile Client

After the tests, the similar results were observed as in the tests with parallel voice and data signaling (section *Tests with Parallel Voice and Data Communication*). Since the same testing application, the same test signals, and similar test parameters were used, the channel utilization was in the similar range between 24% and 50.1% (data channel utilization was between 3.8% and 14.2%, and voice channel utilization was between 20.2% and 35.9%). During the tests with a single mobile client, all data and voice messages were successfully transmitted and approximately the same number of collisions was found. The number of collisions was between 0.11% for the channel usage of 24% and 2.64% for the channel usage of 50.1%. Applying the same approach to analyze voice signals, no influence of collisions was detected.

CHAPTER V

CONCLUSION

Today, many modern public data services can be of benefit to public safety agencies in providing their personnel with additional information while on patrol. Digital data messages can enhance information flow and provide easier ways to access information in mobile settings. However, first responders and mission-critical operations still cannot completely rely on the survivability and redundancy of publicly available data services. The implementation of private APCO Project 25 data services requires an additional piece of equipment called a data base station. The high cost of the commercially-available Project 25 data base radios hinders the implementation of private data services for small departments due to their limited budgets.

The goal of this work, to design and develop a cost effective software defined APCO Project 25 data base station, has been achieved. The data base station is comprised of a standard desktop computer, a commercial analog radio, and a software application. Data packets are transmitted in parallel with voice signaling over a single conventional radio channel using a time division multiple access communication technique. Mixed voice and data signaling over a radio channel can be accessed using a single P25 compliant digital mobile radio. This lowers the cost of providing data services by involving no additional wireless devices in the police cruisers, and increases security and

reliability of data services since the radio infrastructure is owned and operated by the agency. Also, the system relies on the preexisting P25 digital radio network which is standardized in the state of New Hampshire, and which provides a secure and reliable communication infrastructure for first responders. This work represents a good basis for small police departments to enable data services in their cruisers on patrol.

Testing was a crucial part to the completion of the data base station. A precise and methodological laboratory testing procedure has been designed and presented in this work. The testing procedure was divided into three stages: tests with a single data client, tests with multiple data clients, and the assurance of voice priority. The stages were designed to test different aspects of data and mixed voice and data communication over a land mobile radio channel. All three stages were successfully completed and they demonstrated proper transmission and reception of all data messages. Also, the last testing stage showed the proper channel multiplexing between voice and data packets and showed no influence of data messages on the quality of voice communication. The results of the testing procedure are very promising prior to real world deployment.

Each data tests consisted of 10000 request packets while each test with mixed voice and data signaling consisted of 1000 data requests and 1000 retransmissions of one of the five voice tests signals. The size of the data packets was between 12 bytes for acknowledgment packet and 84 bytes for the requested data packet. The duration of voice tests signals was between 2 and 15 seconds. The tests were performed in both unconfirmed and confirmed types of operation and with different channel usage

conditions. The channel usage was varied between 4% and 35% for data only tests and between 24% and 50% for mixed voice and data communication tests. During the tests the channel utilization was kept high to produce a higher number of collisions which were used to reveal any potential influence of data signaling on the quality of voice communication.

The base station will be deployed in a local New Hampshire police department with several cruisers equipped with the Project54 system. Initially, the developed data channel will be used for textual records queries initiated by the Project54 records application. The combination of the data channel and the Project54 speech user interface should provide a safer way to obtain vehicle and driver information on patrol. Also, the data channel can be used to share information from cruisers to headquarters in applications such as remote fleet management, diagnostics, and telematics.

Since the data messages are of a shorter duration and can carry more information than voice messages, we believe that data packets will result in a reduction of the channel utilization. However, the channel utilization can be impacted by collisions and the retransmission of data messages. Also, the channel utilization can be affected by adding new clients to the system or by increasing the amount of radio traffic in an emergency situation. The increase of radio traffic can cause unwanted traffic behavior which can reduce the speed of the channel and cause delays in communication. The data that we gathered during the testing stages should give a good insight to develop a complete model

of a police radio channel that can be used to simulate and predict undesired effects in the radio channel, especially in emergency situations.

REFERENCES

- [1] Z. Medenica and A. L. Kun, "Comparing the influence of two user interfaces for mobile radios on driving performance," *Driving Assessment 2007*, Stevenson, WA, 2007.
- [2] D. Staehle, K. Leibnitz, and K. Tsipotis, "QoS of internet access with GPRS," *Wireless Networks*, vol. 9, no. 3, pp. 213-222, 2003.
- [3] N. Jesuale, "Spectrum policy issues for state and local government," *International Journal of Network Management*, vol. 16, no. 2, pp. 89-101, 2006.
- [4] *Project 25 FDMA Common Air Interface New Technology Standards Project Digital Radio Technical Standards*, Telecommunications Industry Association TIA/EIA-102.BAAA, 1998.
- [5] A. L. Kun, W. T. Miller, III, and W. H. Lenharth, "Computers in police cruisers," *Pervasive Computing, IEEE*, vol. 3, no. 4, pp. 34-41, 2004.
- [6] C. Maria and Z. Lorna, "Usability on patrol," in *CHI '07 extended abstracts on Human factors in computing systems* San Jose, CA, USA: ACM, 2007, pp. 1709-1714.
- [7] S. Y. Kim, K. Wilson-Remmer, A. L. Kun, and W. T. Miller, III, "Remote Fleet Management for Police Cruisers," *IEEE Intelligent Vehicular Symposium*, Las Vegas, NV, 2005.
- [8] I. Cassias, "Project54 Vehicle Telematics for Remote Diagnostics, Fleet Management and Traffic Monitoring." M.S. thesis, University of New Hampshire, 2008.
- [9] E. Ramsey, "A Software Based APCO Project 25 Data Transmission Base Station for Local Police Headquarters." M.S. thesis, University of New Hampshire, 2007.
- [10] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3-11, 1999.
- [11] R. Want, G. Borriello, T. Pering, and K. I. Farkas, "Disappearing Hardware," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 36-47, 2002.
- [12] M. Weiser and S. J. Brown, *The coming age of calm technology*, Copernicus, 1997, pp. 75-85.

- [13] R. Penazzi, P. Capozio, M. Duncan, A. Scuderi, M. Siti, and E. Merli, "Cooperative safety: a combination of multiple technologies," in *Proceedings of the conference on Design, automation and test in Europe* Munich, Germany: ACM, 2008, pp. 959-961.
- [14] J. Navarro, F. Mars, and J. Hoc, "Lateral control support for car drivers: a human-machine cooperation approach," in *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!* London, United Kingdom: ACM, 2007, pp. 249-252.
- [15] W. T. Miller, III, A. L. Kun, and W. H. Lenharth, "Consolidated Advanced Technologies for Law Enforcement Program," IEEE Intelligent Transportation Systems Conference, Washington, DC, 2004.
- [16] J. LeBlanc, T. Hurton, W. T. Miller, III, and A. L. Kun, "Design and evaluation of a vehicle data distribution and collection system," Fifth International Conference on Pervasive Computing (Adjunct Proceedings), Toronto, Canada, May 13-16, 2007.
- [17] M. Tanizaki and O. Wolfson, "Randomization in traffic information sharing systems," in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems* Seattle, Washington: ACM, 2007, pp. 1-8.
- [18] A. Skordylis and N. Trigoni, "Delay-bounded Routing in Vehicular Ad-hoc Networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing* Hong Kong, Hong Kong, China: ACM, 2008, pp. 341-350.
- [19] K. Goto and Y. Kambayashi, "A new passenger support system for public transport using mobile database access," in *Proceedings of the 28th international conference on Very Large Data Bases* Hong Kong, China: VLDB Endowment, 2002, pp. 908-919.
- [20] A. Repenning and A. Ioannidou, "Mobility agents: guiding and tracking public transportation users," in *Proceedings of the working conference on Advanced visual interfaces* Venezia, Italy: ACM, 2006, pp. 127-134.
- [21] Y. Kin Choong, C. Lin, and L. Xiaoyu, "BlueBus: a scalable solution for localized mobile service in a public bus," in *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology* Singapore: ACM, 2007, pp. 712-715.
- [22] C. Wuthrich, G. Kalbfleisch, T. Griffin, and N. Passos, "On-line instructional testing in a mobile environment," *J. Comput. Small Coll.*, vol. 18, no. 4, pp. 23-29, 2003.

- [23] M.Hazas, J.Scott, and J.Krumm, "Location-Aware Computing Comes of Age," *IEEE Computer*, vol. 37, no. 2, pp. 95-97, 2004.
- [24] M. F.Mokbel , C.-Y. Chow, and W. G.Aref , "The new Casper: query processing for location services without compromising privacy," in *Proceedings of the 32nd international conference on Very large data bases* Seoul, Korea: VLDB Endowment, 2006, pp. 763-774.
- [25] S. Duri, J. Elliott, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J. M. Tang, "Data protection and data sharing in telematics," *Mob. Netw. Appl.*, vol. 9, no. 6, pp. 693-701, 2004.
- [26] J. W.Streefkerk, M. P. van Esch-Bussemaekers, and M. A. Neerincx, "Field evaluation of a mobile location-based notification system for police officers," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services* Amsterdam, The Netherlands: ACM, 2008, pp. 101-108.
- [27] *Project 25 Document Suite Reference*, Institute for Telecommunication Sciences, 2008.
- [28] *APCO Project 25 System and Standards Definition*, Telecommunications Industry Association TSB102-A, 1995.
- [29] *Project 25 Data Overview - New Technology Standards Project - Digital Radio Technical Standards*, Telecommunications Industry Association TIA/EIA-102.BAEA, 2000.
- [30] *Project 25 Circuit Data Specification New Technology Standards Project Digital Radio Technical Standards*, Telecommunications Industry Association TIA/EIA-102.BAEC, 2000.
- [31] *Project 25 Radio Control Protocol (RCP) - New Technology Standards Project - Digital Radio Technical Standards*, Telecommunications Industry Association TIA/EIA-102.BAEE, 2000.
- [32] *Project 25 Vocoder Description*, Telecommunications Industry Association TIA/EIA-102.BABA, 1998.
- [33] *APCO Project 25 Common Air Interface Operational Description for Conventional Channels*, Telecommunications Industry Association TIA/EIA TSB102.BAAD, 1994.
- [34] *APCO Project 25 Trunking Overview*, Telecommunications Industry Association TIA/EIA TSB102.AABA, 1995.
- [35] W. Stallings, *Data and Computer Communication*. Upper Saddle River, NJ: Parson Prentice Hall, 2007.

- [36] Tyco Electronics. *Expanding Digital Communication, P25^{IP} VIDA Network System*. Massachusetts, USA : Tyco Electronics M/A-COM.
- [37] Etherstack, "APCO P25 Base Station," Amsterdam, Etherstack, Inc.
- [38] J. Z. Gao, H. S. J. Tsao, and Y. Wu, *Testing and Quality Assurance for Component-Based Software*. Norwood, MA: ARTECH HOUSE, INC, 2003.

APPENDIX A

CUSTOM KENWOOD RADIO INTERFACE CABLE

Since the data base station designed in this work was comprised of a PC and an analog radio, a custom radio interface cable was used to connect the Kenwood TK 7180 radio to a COM port of the PC. The cable was used to control the PTT signal (through a serial data connection) and to provide an audio interface between the radio and the PC audio card. The schematic of the radio cable is shown in Figure 38.

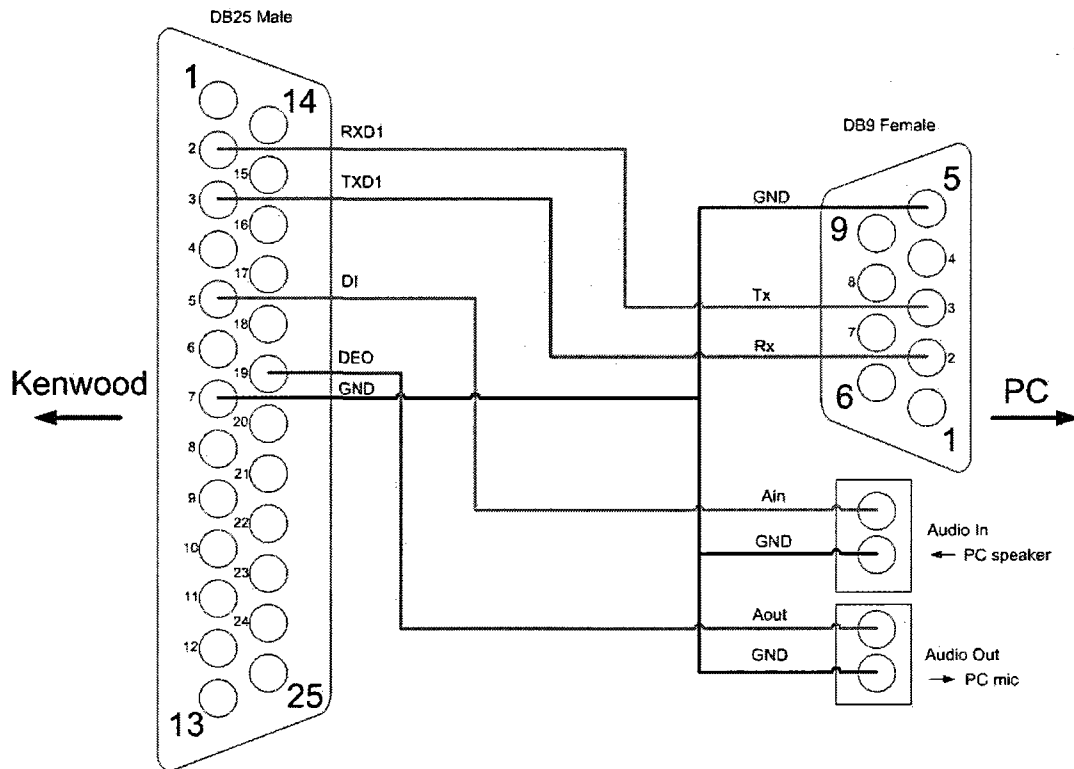


Figure 38 - Custom Kenwood Radio Interface Cable

APPENDIX B

KENWOOD TX 7180 CODE PLUG

The Kenwood radio code plug settings needed for proper functioning of the data base station were edited using the programming software provided by Kenwood. First, appropriate test radio channel frequencies were set. In this work, the same frequency 155.37 MHz was used for both transmission and reception of CAI data packets. The channel settings are shown in Figure 39.

The screenshot shows a software window titled 'Zone Information [Zone : 1 Channel : 1]'. It contains a table with 10 columns: No., RX Frequency, TX Frequency, QT/DQT Dec, QT/DQT Enc, Channel Name, Power, V/M, Scan Add, and Opt Signal. The first row (No. 1) is populated with the following values: RX Frequency: 155.37000, TX Frequency: 155.37000, QT/DQT Dec: None, QT/DQT Enc: None, Channel Name: P25Server, Power: Low, V/M: Wide, Scan Add: No, Opt Signal: None. The window also includes a 'Zone Type' dropdown set to 'Conventional Group', a 'Zone Name' field with '1', and a 'Free Area' indicator showing '28224 bytes'. At the bottom, there are buttons for 'Zone Up', 'Zone Down', 'Zone Edit', 'Channel Edit', 'Close', and 'Help'.

| No. | RX Frequency | TX Frequency | QT/DQT Dec | QT/DQT Enc | Channel Name | Power | V/M | Scan Add | Opt Signal |
|-----|--------------|--------------|------------|------------|--------------|-------|------|----------|------------|
| 1 | 155.37000 | 155.37000 | None | None | P25Server | Low | Wide | No | None |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | |
| 16 | | | | | | | | | |

Figure 39 - Test Channel Frequency Settings

Second, in order to control the PTT signal over a serial data line, a radio COM port had to be enabled and set to accept data messages. The settings for a radio COM port are shown in Figure 40. Finally, in the radio modulation settings, the external data PTT

signal had to be set to connect the data signal input (DI) line to the modulator avoiding the audio processor, as it is shown in Figure 41.

Optional Features

Common-Page 1 | Common-Page 2 | Common-Page 3 | **Conventional** | Trunking | VGS-1

Battery

Battery Saver:

Battery Warning:

☒ Battery Status

PTT ID

PTT ID Type:

Beginning of Transmit:

End of Transmit:

| COM port No. | Function | Polarity |
|--------------|----------|----------|
| COM port 0 | None | Reversed |
| COM port 1 | Data | Normal |
| COM port 2 | None | Reversed |

Figure 40 - Radio COM Port Settings

Extended Function

Optional Board | AUX | Remote Zone-CH/GD | **Modulation Line**

| PTT | Connect to Modulation Line | | | w/QT/DQT | w/STE |
|----------------------|----------------------------|------------|------------|----------|-------|
| | Mic Line | M12 Line | DI Line | | |
| MIC PTT | Connect | Disconnect | Disconnect | Yes | Yes |
| External PTT (Voice) | Disconnect | Connect | Disconnect | Yes | Yes |
| External PTT (Data) | Disconnect | Disconnect | Connect | Yes | Yes |
| Data PTT | Disconnect | Disconnect | Connect | Yes | Yes |

Modulation Line by External PTT (Data)

Diagram showing the connection of Mic, M12, and DI lines to the Audio Processor and Modulation Circuit. The DI line is connected to the Modulation Circuit, bypassing the Audio Processor.

```

graph LR
    Mic((Mic)) -- Disconnect --> AP[Audio Processor]
    M12((M12)) -- Disconnect --> AP
    DI((DI)) -- Connect --> MC[Modulation Circuit]
    AP --> MC
    MC --> ANT[ANT]
  
```

Figure 41 - Radio Modulation Line Settings