ESD-TR-83-054

(12

AD A136938

Technical Report
663

Protocol Software for a Packet Voice Terminal

C.K. McElwain

16 November 1983

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002.

DTIC
ELECTE
JAN 18 1984
A

DTIC FILE COPY

Approved for public release; distribution unlimited.

84 01 17 130

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3673).

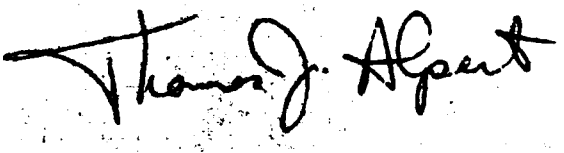
This report may be reproduced to satisfy needs of U.S. Government agencies.

The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United States Government.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document when it is no longer needed.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

PROTOCOL SOFTWARE FOR A PACKET VOICE TERMINAL

C.K. McELWAIN

Group 24

TECHNICAL REPORT 663

16 NOVEMBER 1983

Approved for public release; distribution unlimited.

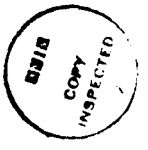
JAN 1 8 1984

A

LEXINGTON MASSACHUSETTS

ABSTRACT

A Packet Voice Terminal (PVT) has been developed at Lincoln Laboratory to provide voice access to an experimental wideband internetwork packet system. The PVT employs a modular, microprocessor-based structure to provide voice processing, packet voice protocol, and network interface functions. The packet voice protocols are implemented in software in the Protocol Processor (PP) module, which is the primary controlling module of the PVT and which handles interfaces to a voice processor, a network interface processor, and a user instrument. This report describes the software implemented in the Protocol Processor. The implementation of the Network Voice Protocol (NVP-II) and the Stream (ST) protocol are described. Call set-up functions for both point-to-point calls and conferencing, and the methods used for packetization and reconstitution of speech, are described. Problems encountered and solutions which have been implemented are discussed.



A1

CONTENTS

ABSTRACT iii

1. INTRODUCTION 1

2. PACKET VOICE TERMINAL ARCHITECTURE AND DESIGN 4

2.1 Overview 4

2.2 Protocol Processor Architecture and Interfaces 7

2.3 Protocol Processor Software Modules; Function and Size 16

3. VOICE PROTOCOLS 17

3.1 Brief History of Voice Protocols 17

3.2 Network Voice Protocol II 18

3.3 Stream Protocol 18

4. CALL SETUP 19

4.1 Overview 19

4.2 Setting Up a Point-to-Point Call 21

4.3 Conferencing 26

5. DIALING CONVENTIONS 35

5.1 Interaction with the Caller 35

5.2 Dialing a Point-to-Point Call 36

5.3 Conference Dialing 37

5.4 The Echo Extension 37

5.5 Using the Switched Telephone Network Interface (STNI) 39

5.6 Special Dialing Sequences 40

6. SPEECH DATA 41

6.1 Packetizing Speech Data 41

6.2 Silence Detection 47

6.3 Vocoder Dependent Modules 48

6.4 Efficient Handling of Speech Data 49

6.5 Reconstitution of Speech Data 52

7. RELIABILITY 55

7.1 Reliable Transmission of Control Messages 55

8. REAL-TIME STRUCTURE OF PVT SOFTWARE 57

8.1 Assembly Language Code 57

8.2 Polling Loop 58

8.3 Buffer Availability to BC 59

8.4 Output Message Formation 59

8.5 Timing 59

9. LANGUAGES AND SUPPORT FACILITIES 61

9.1 Choosing Languages 61

9.2 Support Facilities 62

9.3 Downloading Facilities 62

9.4 PROMS 63

10. MONITORING AND DEGUGGING AIDS 63

10.1 Diagnostic Record Keeping 63

10.2 "Talking To Yourself" 64

10.3 Echo Extension 65

10.4 Providing Information to the User 65

11. COMMENTS 65

11.1 Implementing Protocols in a PVT 65

11.2 Use of Checksums 66

11.3 RAM Memory 66

11.4 Implementation on Packet Radio Network 67

11.5 Implementing NVP-II and ST 67

11.6 Support Facilities 68

12. SUMMARY 68

ACKNOWLEDGEMENTS 69

REFERENCES 70

APPENDIX I - Acronyms and Abbreviations 72

APPENDIX II - Size and Function of PP Modules 73

1. INTRODUCTION

An experimental wideband packet internetwork system is being implemented under sponsorship of the Defense Advanced Research Projects Agency (DARPA) to develop and demonstrate techniques for achieving the advantages of integrating packet voice with data in a realistically large-scale system [1,2,3,4]. The experimental system consists of research facilities at multiple sites which are linked by a wideband packet satellite network including a satellite transponder channel, earth stations, high-performance burst modems, and demand assignment processors. The sites have local packetized speech access facilities including concentrators, local distribution networks, and packet voice terminals.

Lincoln Laboratory has developed the Lincoln Experimental Packet Voice Network (LEXNET) [5] to act as a local access area network for voice traffic traveling to and from the satellite. The LEXNET provides broadcast connectivity among a set of Packet Voice Terminals (PVTs) by means of a wideband coaxial cable. The PVT digitizes and packetizes speech for transmission over the local network. For local calls, these packets are sent directly to another PVT on the same LEXNET. To reach a PVT on another network, the packets are sent to a special interface unit, the LEXNET Concentrator Interface (LCI), which forwards the packets to the remote net via a speech concentrator. The speech concentrator acts as a GATEWAY between two LEXNETS and/or between a LEXNET and the satellite. The GATEWAY consists of a PDP-11/44[1] augmented with peripheral UMC-Z80[2] processors to aid in

---

[1]Digital Equipment Corporation, Maynard, MA

[2]Associated Computer Consultants, Santa Barbara, CA

I/O operations. A flexible packet-satellite demand assignment multiple access (DAMA) processor[3] (PSAT) provides scheduling for the satellite channel. A flexible burst modem serves as an Earth Station Interface[4] (ESI) with the earth station transmitter and receiver.

The experimental network currently includes four sites: Lincoln Laboratory; the Defense Communications Engineering Center, Reston Virginia; the Information Sciences Institute of the University of Southern California, Marina Del Rey, California; and SRI International, Palo Alto, California. The satellite channel is supported by Western Union's WESTAR III satellite.

The PVTs with their attached telephone instrument serve as the interface with the voice user. The PVTs prepare speech for transmission through a packet network by digitizing the speech, preparing speech data packets, and sending speech data messages. The PVT handles the speech coming in from the network. The protocols required to establish communication with other PVTs are also implemented.

In earlier experiments sending packetized speech over ARPANET and SATNET, the function of a PVT was performed on large host computers such as PDP-11s. The PVT has the advantages of being modular, compact (microprocessor based) and generalizable. It can be programmed to handle many different single card voice processors (vocoders) and can be adapted to interface to a variety of networks. A PVT is shown in Figure 1.

The purpose of this report is to describe the protocol software for the PVT. The report will focus on the present implementation but much is generalizable. In Section 2 the PVT architecture and design is shown.

---

[3]Bolt Beranek and Newman, Cambridge, MA

[4]LINKABIT Corporation, San Diego, CA

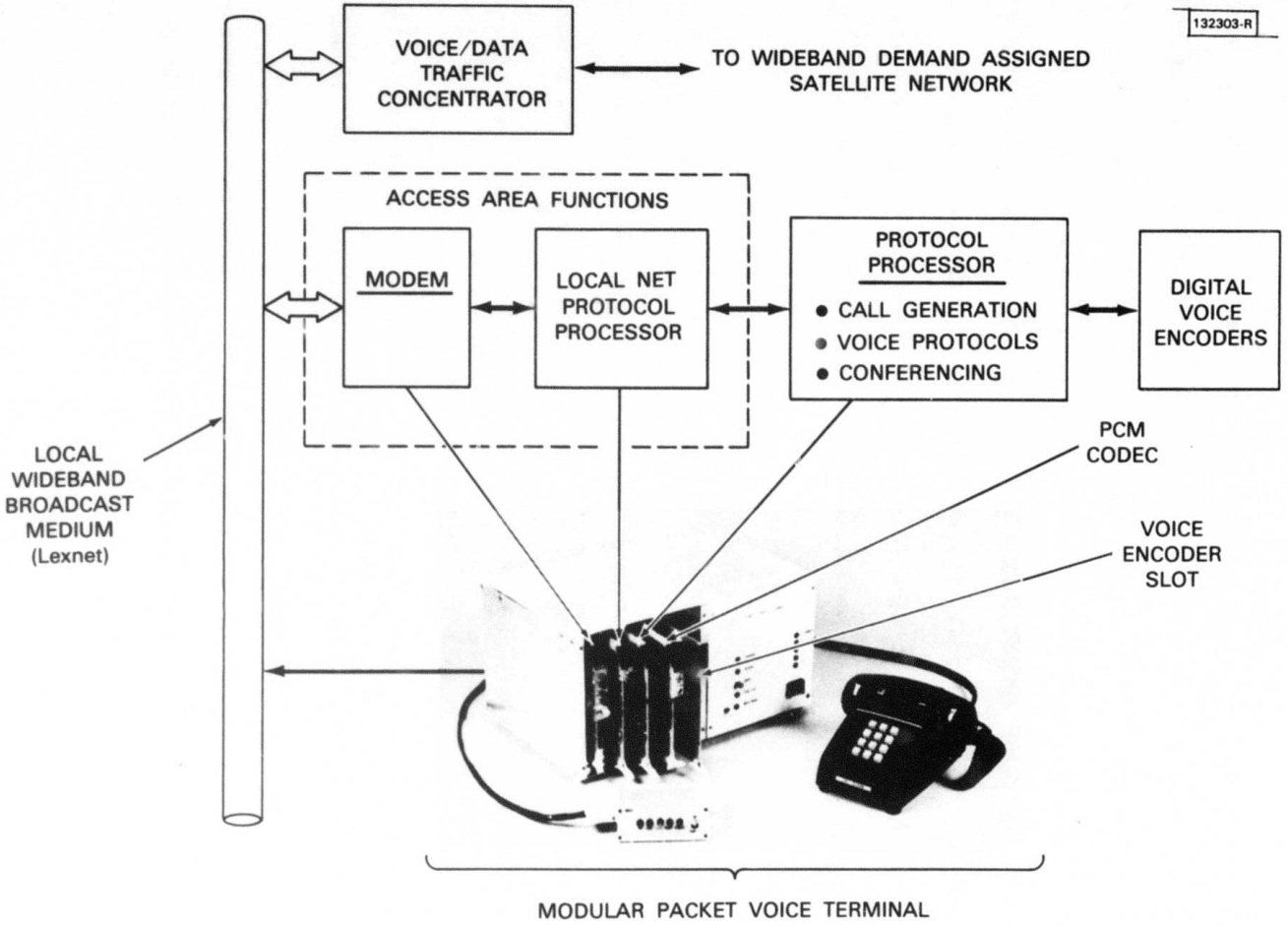


Fig. 1 Lincoln Experimental Packet Voice Network

Section 3 discusses the two protocols used and gives a brief history of the development of these protocols. Next, call setup functions are described, including the standard point-to-point call, conference calls and calls to and from phones in the commercial telephone system. Dialing conventions, efficient handling of speech data and the problem of reliable transmission are addressed. Then the real time structure of the PVT software, the languages used, the support facilities, and the debugging aids are covered. The report ends with some comments and conclusions that have emerged from this effort.

2. PACKET VOICE TERMINAL ARCHITECTURE AND DESIGN

2.1 Overview

This section describes briefly the design of the Packet Voice Terminal [6], the modules that comprise it, and the tasks that the protocol processor performs while servicing the interfaces between modules.

2.1.1 Design of a Packet Voice Terminal

The experimental Packet Voice Terminal (PVT) was designed to provide a 64 kbps PCM capability, and also to provide a flexible interface for any of a number of single-card narrowband speech processors. The terminal is partitioned into three functional units; the speech digitizer, the protocol processor, and the access area processor. Speech can be digitized by a PCM CODEC or by a speech processor internal or external to the voice terminal unit. Speech digitization is independent of the transmission process. The protocol processor (PP) forms the packets and provides the necessary protocol functions to insure that the packets can be delivered to a distant network and played out at the proper time. The protocol functions are designed to be

independent of the transmission medium. The access area controller or buffer controller (BC) accepts packets from the PP and transmits them across the LEXNET to a similar controller in another terminal. The BC is only concerned with packet transport on the LEXNET.

2.1.2 Modular View of Packet Voice Terminal Functions

The PVT can be thought of as being made up of the modules shown in Figure 2. The voice processor module represents the vocoder currently being used. The PVT is programmed to handle three different vocoders; 64 kbps Pulse Code Modulation (PCM), 2.4 kbps Linear Predictive Coding (LPC) [8] and a variable-rate 16-64 kbps Embedded Continuously Variable Slope Delta modulation (ECVSD) [7]. (PCM and ECVSD are actually speech waveform encoders but the term vocoder will be used when referring to them.) The box labeled user telephone represents the phone-like instrument that the user uses to communicate. The buffer control (BC) processor connects directly to the LEXNET and sends/receives messages from LEXNET. The protocol processor (PP) module handles the protocols involved in setting up Point to Point calls and Conference calls. Two protocols are implemented in the PP; the second generation Network Voice Protocol (NVP-II) [9] and the Stream protocol (ST) [10]. These two protocols are used to negotiate the connections and transmit the speech data. The PP module is also equipped with a vocoder preference selection switch (VPSS) by which the user can indicate which vocoder he desires to use.

The protocol processor services the interfaces to the other modules including: packetizing and depacketizing speech data, monitoring and processing signals from the phone's key pad, monitoring and adapting to the

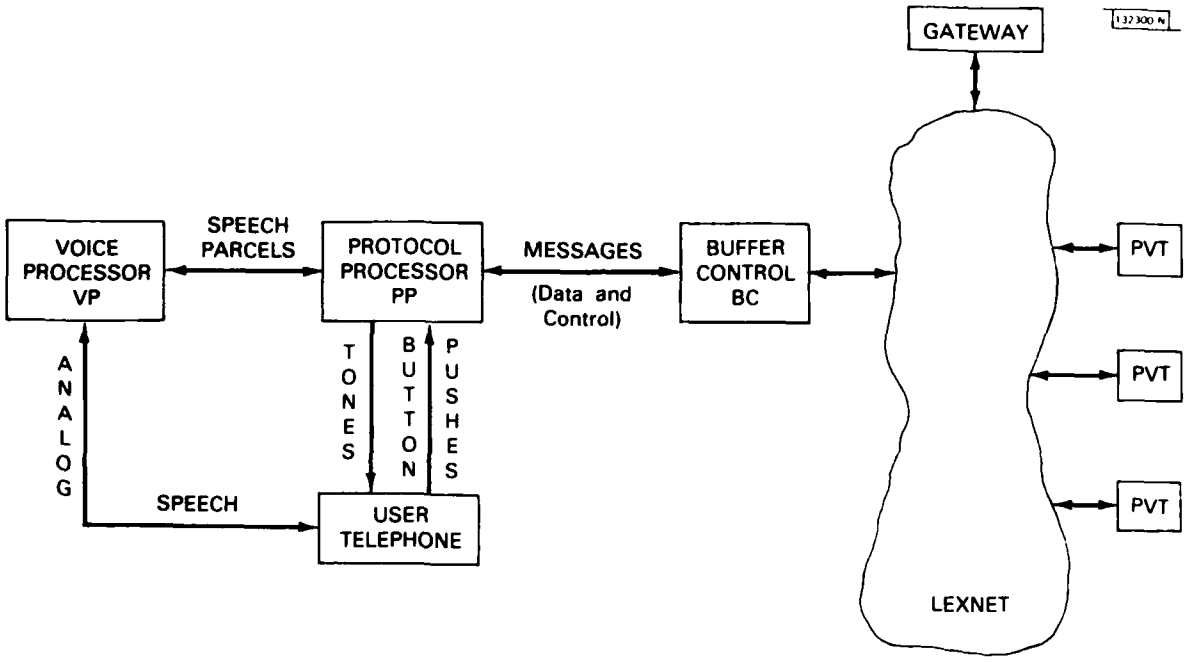


Fig. 2 Modular View of a PVT

desired vocoder, and processing messages going to and from the LEXNET. This report is concerned with the functions of the protocol processor (PP) module and their implementation in software.

2.2 Protocol Processor Architecture and Interfaces

2.2.1 Protocol Processor Architecture

The protocol processor resides on a card consisting of an INTEL 8085 microprocessor CPU, a data memory, a program memory, a DMA controller,and a USART port. A block diagram is shown in Fig. 3. A separate memory extension card is included allowing the total memory to be about 40K bytes. Two versions of this extension card are available. A RAM card can be used for network sites where a downloading or crossnet loading capability is available and for software development. A PROM version of the card provides a stand-alone capability. Only 4096 bytes of the RAM memory can be accessed by the DMAs. (This limitation has significantly complicated the software.) The term "DMA memory" will be used to refer to this memory. The current NVP/ST protocol program requires about 31K bytes.

A DMA controller is used to pass data in both directions to the access controller and to the speech processor. The USART is used as a serial channel to the telephone instrument for signaling and display. Communication between the protocol processor and the speech processors is via two pairs of byte-parallel channels. The first pair is used by the speech data and is controlled by the protocol processor's DMA chip. The second pair is a control channel which has several possible uses depending on the specific speech processor. At start-up the speech processor presents an identification code on the control input channel. The protocol processor

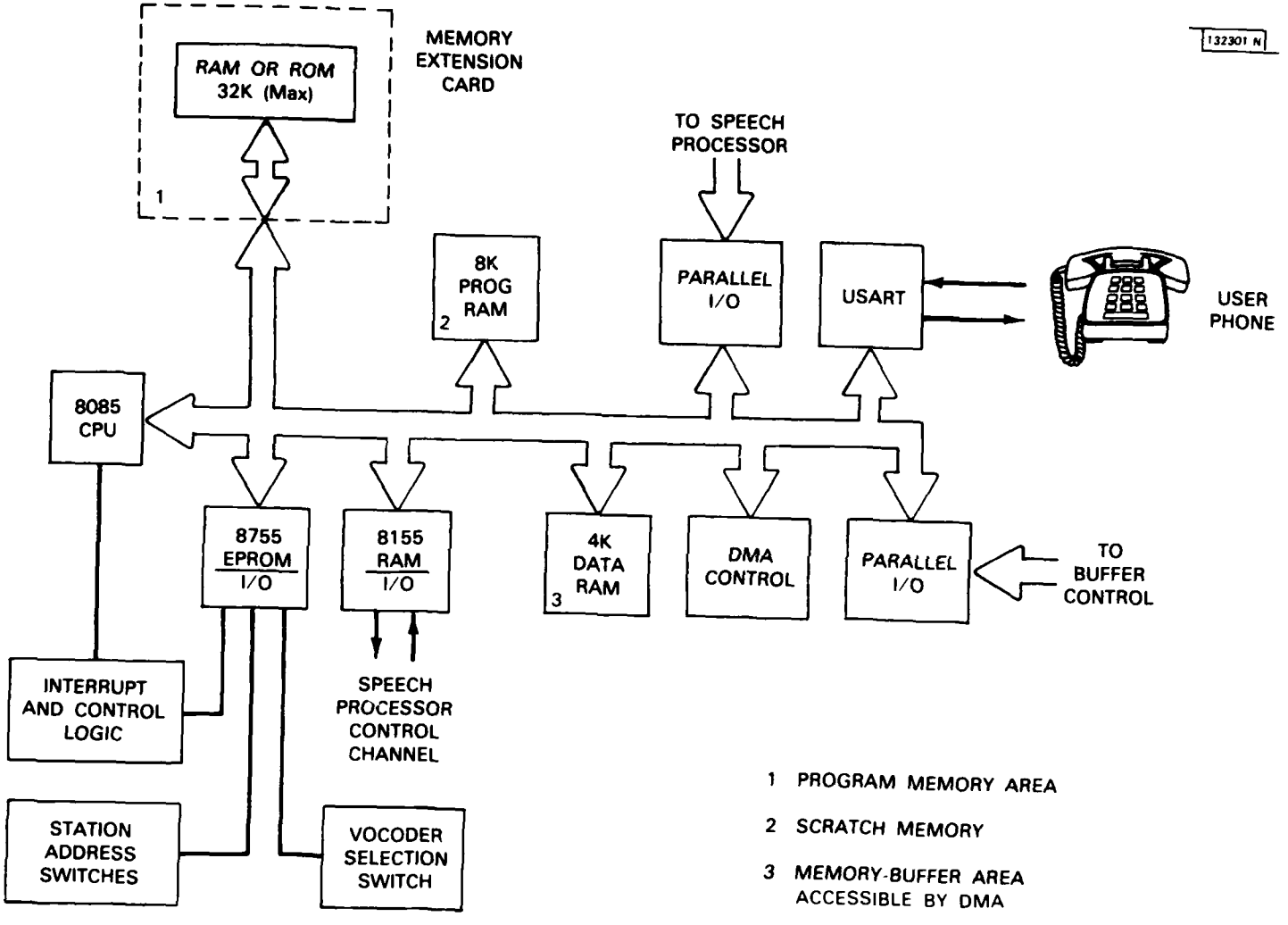


Fig. 3 Protocal Processor block diagram

uses this code to determine which speech algorithm is active and to select appropriate program parameters. The control channel can also be used for passing control parameters to variable-rate speech systems or for loading programs into programmable voice processors.

2.2.2 User Telephone

The telephones are standard key pad telephones which have been modified to match the needs of the PVTs. Internally, the phone contains a small microprocessor system consisting of an 8085 CPU, ROM memory, keypad and a USART. Communication with the PP is via an RS-232 serial interface. Drivers and receivers for the analog signals passing to and from the handset are contained in the telephone set. Commands transmitted over the serial line by the PP cause signaling tones to be added to the analog signal driving the earpiece.

The PP can send the following five commands to the phone: ring, dial tone, busy signal, ringing tone and silence. Once a command is sent to the phone it will continue in effect until another command is sent. The protocol processor uses these commands to create a calling environment similar to what the user expects from a regular phone.

The telephone is equipped with a standard key pad with the twelve characters 1 thru 9, plus *,0,and #. The PP receives a signal when the phone goes on or off hook as well as when a key is pushed. When a digit is pushed, a tone is sent to the phone for approximately 9 msec. This causes the phone to seem to be "alive" to the user and reassures him that his key push was noted. A LEXNET phone is shown in Fig. 4.

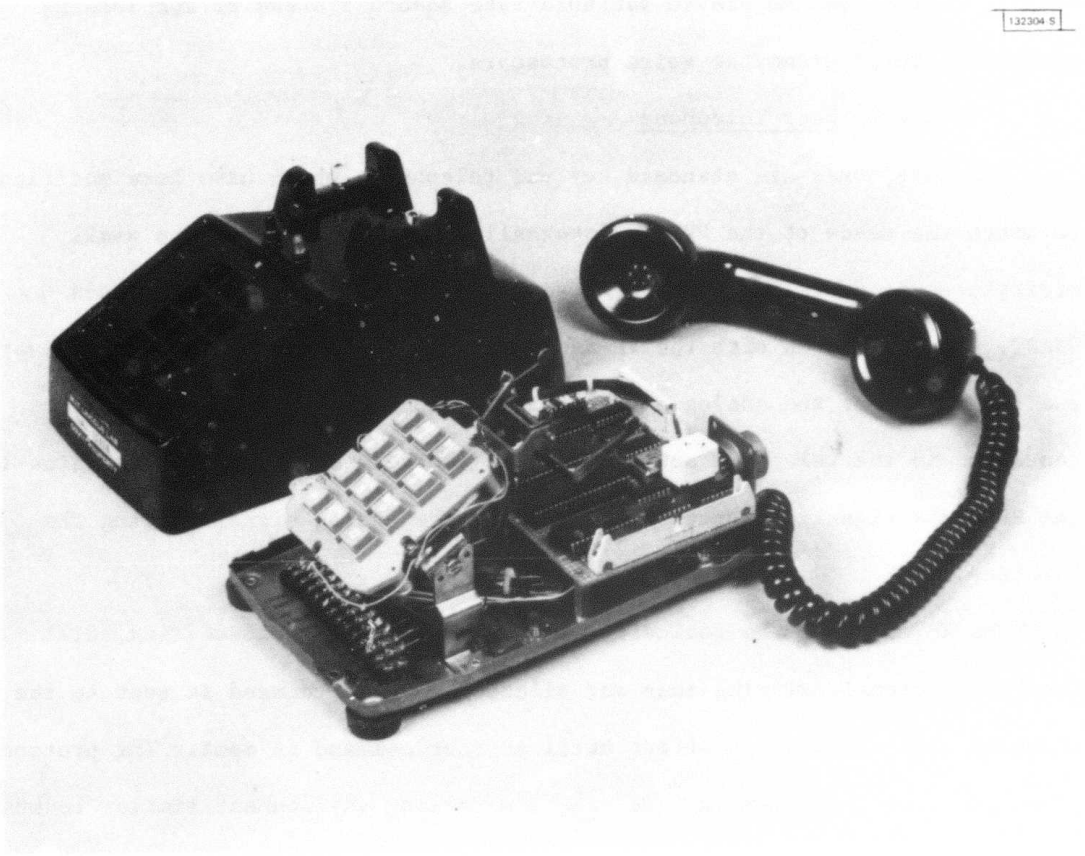


Fig. 4 LEXNET telephone

2.2.3 Voice Processor Selection

All PVTs are equipped with a PCM vocoder card. There is an extra card slot which is wired to accommodate an alternate vocoder. The PP software supports two alternate vocoders; a 2400 bps LPC Vocoder [8], and a variable-rate Embedded CVSD Vocoder [7] which operates at rates of 16, 32, 48, or 64 kbps. The caller is supplied with a switch so that he can express a preference between PCM and the vocoder plugged into the alternate vocoder slot. The preference switch is a manual switch that is monitored by the program. The actual selection of the vocoder is controlled by the program. The switch is a three position momentary switch. Its rest position indicates no preference. The PP constantly monitors the position of the vocoder preference switch. If a new preference is indicated, the PP will record the fact and take appropriate action.

2.2.4 Connection to the Voice Processor

Speech to and from the vocoder is sent in units called parcels. For a narrowband vocoder such as LPC, a parcel consists of the data generated by the vocoder in analyzing one unit or frame of speech. The basic frame time for the LPC vocoder used in the PVT is 20 msec. The PCM and ECVSD vocoders do not have a basic frame time. Data for these vocoders are passed in parcels which comprise 22.5 msec of speech.

Speech goes to and from the vocoder via two direct DMA channels working in auto-initialize mode. Separate buffer space is reserved in the DMA memory for each channel. In the auto-initialize mode the DMA reads/writes speech from/to the buffer until it reaches the end. It then automatically begins again at the beginning of the buffer. The buffer areas are sized so that they always end at a parcel boundary.

Transfers in each direction are independent. An interrupt is provided once per parcel for each direction. A one-byte header precedes each speech parcel in the buffer. The header byte tells whether the following parcel is speech or silence. On reading a parcel which is marked as silence, the vocoder plays out an internally stored parcel of silence. When the vocoder finishes reading a parcel, the end-of- parcel interrupt executes code which marks the parcel as silence by clearing the speech indication in the header byte. This ensures that a speech parcel will not be played out more than once even if no new speech is received before the DMA reaches this buffer again. When writing speech parameters into the speech buffer, the PP sets the header byte to indicate that the parcel contains speech.

2.2.5 Access Area Interface

All messages on a LEXNET begin with a two-word LEXNET header which consists of the destination address and the source address for the message. During a conference, messages containing speech data use the broadcast LEXNET address augmented with the ID number assigned to the conference as a destination address. The Buffer Control card forwards to the PP card all messages on the LEXNET which are specifically addressed to it plus all messages addressed to any conference ID which the PP has asked to receive. Once the buffer control card has notified the PP that it has a message for it, the PP must read in the entire message or it will get out of sync with the buffer control card.

The PP expects to receive three distinct types of messages (see Section 3 on Protocols). Control information is transmitted in two different types of messages corresponding to the two protocols IP (DoD standard Internet

Protocol [11]) and ST. (See Section 3). The third type of message contains speech parcels and may also have control information preceding the speech. The PP first reads in the number of bytes equal to the smallest correct header it can receive. If analysis of this data shows the message to be defective (e.g., bad checksum), the remaining bytes are read into "nonexistent memory" and thereby discarded. If the message contains only protocol information, all the remaining bytes are read into the PP read-in buffer. For a message containing speech parcels, all the bytes in the message which precede the first speech parcel are read into the PP read-in buffer. Then each parcel of speech is transferred via DMA directly into a selected buffer in the set of buffers which contain speech to be played out. Efficient handling of incoming speech parcels is discussed in Section 6.

The PP creates the same three types of messages to send out to the LEXNET via the Buffer Control card. Messages which contain only control information are formed in one of two output buffers in DMA memory. One buffer is used for IP messages, the other for ST messages. These messages are transferred directly from the output buffer to the Buffer Control card. The header and any accompanying control tokens for messages containing speech parcels are generated in a third output buffer. After this information is sent to the Buffer Control card, the accompanying speech parcel(s) are transferred via DMA one by one directly from the circular buffers which receive speech from the vocoder. For vocoders with low bit rates and correspondingly few data bits in a parcel, several parcels are usually sent in one message. For a high bit rate vocoder such as PCM, only one parcel is sent in a message.

2.2.6 Summary of the PP Functions

The primary function of the PP is to implement the two protocols NVP-II and ST, thereby giving a user access via packet speech to other PVTs and to regular phones. It does this through interaction with the four interfaces connected to it. The PP does not spontaneously generate any activity. A summary of the interactions with each interface is contained in Table I.

TABLE I

SERVICES PROVIDED BY THE PROTOCOL PROCESSOR

User Instrument Interface

- Receive signals from the instrument via the USART
- Interpret Signals Received
- Implement action caller desires
- Send commands for the appropriate tones to the instrument to reflect current status
- Send ring command to the instrument when call received from LEXNET

Voice Processor (Vocoder) Selection

- Monitor Vocoder Preference Selection Switch (VPSS)
  - When changed, reconfigure buffer assignments and storage space dynamically to use newly requested vocoder and select vocoder.
- Switch Vocoder Selection internally to satisfy "Request to Talk"

Voice Processor (Vocoder)

- Set up buffer space, pointer tables, etc., according to type of Vocoder in use
- Start speech transfer (both directions) when call established
- Package frames of speech into speech packets and send
- Stop speech transfer when connection ends or is broken

Access Area Processor Interface

- Accept, analyze, and act on all incoming messages
- Apply reconstitution algorithm to incoming speech messages
- Unpack speech parcels in message and DMA into correct Vocoder buffer
- Respond to incoming protocol messages
- Create and Send Outgoing messages
- Determine which speech parcels to send, pack and send them
- Control protocol dialogue which establishes requested connections
- Insure reliable transmission of Protocol Messages

2.3 Protocol Processor Software Modules; Function and Size

The software system for the PP is written partly in the high level language C and partly in an assembly language called A-Natural [12]. A-Natural is the assembly language generated by the C compiler used for this project. (See Section 9.1).

The A-Natural routines handle all the input/output interaction between the PP and 1) the USART attached to the user phone, 2) the DMAs to and from the buffer control, 3) the DMAs to and from the vocoders, and 4) the two frame sync interrupts for the vocoders. Routines in A-Natural also initialize the various I/O devices and set up and initialize the speech buffers.

The C routines handle the analyses of incoming messages, process all the protocol messages, and direct the actions of the lower level A-Natural routines.

There are 114 C routines of various sizes which are contained in six separate modules with a total length of 28,464 bytes.

There are 17 routines written in A-Natural which are contained in two modules and are 2855 bytes long. There are three routines to initialize/reinitialize various I/O interfaces, a routine to set up speech buffers, routines to start interrupts, and to start and stop the speech DMAs. Two routines run at interrupt level and handle the frame sync interrupts from the vocoder, three routines handle the output to the phone, two routines handle error conditions, two routines compute checksums (word checksum and byte checksum). The final routine is the general control loop which polls the status of all I/O interfaces and causes appropriate action to

be taken. The Whitesmith system [12], which compiles and links these modules, adds sixteen library routines which are a total of 566 bytes long.

More detail on the size and function of each module is contained in Appendix 2.

3. VOICE PROTOCOLS

3.1 Brief History of Voice Protocols

The initial efforts beginning around 1973 to conduct packet voice experiments on the ARPANET focussed on the development of appropriate network protocols for voice. The Network Control Protocol (NCP) then in use was not satisfactory for the throughput and delay requirements of real-time speech. A protocol was needed that: handled real time data, was vocoder independent, was network independent, and separated data and control. Network Voice Protocol (NVP-I) was developed by the Information Sciences Institute (ISI) [13] and was used in initial speech experiments in 1974. NVP-I was extended to handle conferencing and became the Network Voice Conferencing Protocol (NVCP). Then after IP became a widely-used DoD-standard protocol, NVP-I was improved and modified so that its control tokens can be carried as the data portion of IP datagrams. The resulting protocol is NVP-II [9]. (Like NCP, the DoD Standard Transmission Control Protocol (TCP) [11] is not suitable for real-time voice transmission.) Stream Protocol (ST) [10] was developed as an extension to IP to efficiently transmit the speech data after a protocol negotiation had been conducted using NVP-II and to handle the multi-address delivery requirements of conferencing.

Voice Protocols define a sequence of messages that are sent between terminals in order to set up and maintain a packet voice connection between

two voice terminals or a conference among many voice terminals. Two kinds of messages are involved--Control messages and Speech Data messages. Control messages contain information needed to set up or maintain the call. This includes information about the vocoder to be used, the addresses of the terminals involved, dynamic conference control information etc. The speech messages normally contain only speech data although there is a provision for including control information. The PVT software normally sends NVP-II control information as the data portion of IP datagrams. NVP-II speech parcels are sent as the data portion of ST messages.

The implementation of the two voice protocols NVP-II and ST is the primary subject of this report.

3.2 Network Voice Protocol II (NVP-II)

NVP-II handles the vocoder-type negotiations, ringing, timestamping, etc., as well as dynamic conference control functions. NVP-II defines a large number of protocol "tokens" which pass pertinent information. These tokens are sent either as the data portion of IP datagrams, as is done in call set-up, or along with data in an ST message. NVP-II also defines the form of the speech data parcels which are sent in ST messages.

3.3 Stream Protocol (ST)

ST provides an internet transport mechanism for the delivery of speech parcels for both point-to-point conversations and conferences. ST differs from IP because it creates a virtual circuit as opposed to a datagram protocol. In order for a host or GATEWAY to be able to interpret an ST message header, it must be told about the virtual circuit (called a "connection" in ST) by a setup process that precedes the transmission of any

speech messages. The setup process involves finding an internet route for the connection and informing all GATEWAYs that might have occasion to deal with packets for the connection. During the process, GATEWAYs build tables containing information about the connection. It is these tables that permit ST to offer capabilities that cannot be provided by a datagram protocol such as IP. After a connection is established, speech messages are sent with a minimal header which contains the name of the connection to be used. The tables in the GATEWAYs contain the information needed to route these messages correctly.

To set up a point-to-point (PTP) call between two PVTs, the parameters of the call are first negotiated using IP datagrams containing NVP-II tokens. Then a connection or route is established using ST. When this is complete and the remote site has answered the phone, speech can flow. Point-to-point call setups are discussed in Section 4.2. Setting up a conference requires establishing an ST conference connection between each pair of voice terminals. Speech packets sent on a conference connection are distributed automatically to all participants. Conference setups are discussed in Section 4.3.

4. CALL SETUP

4.1 Overview

The PVT software supports calls between any two PVTs as long as there is a route available between them. Figure 5 shows the possible paths between PVTs in a typical wideband system configuration. The special lines show the route which would be taken by a call between PVT 231 and each of the other PVTs. The "number" by which a PVT is known is the number of its local LEXNET

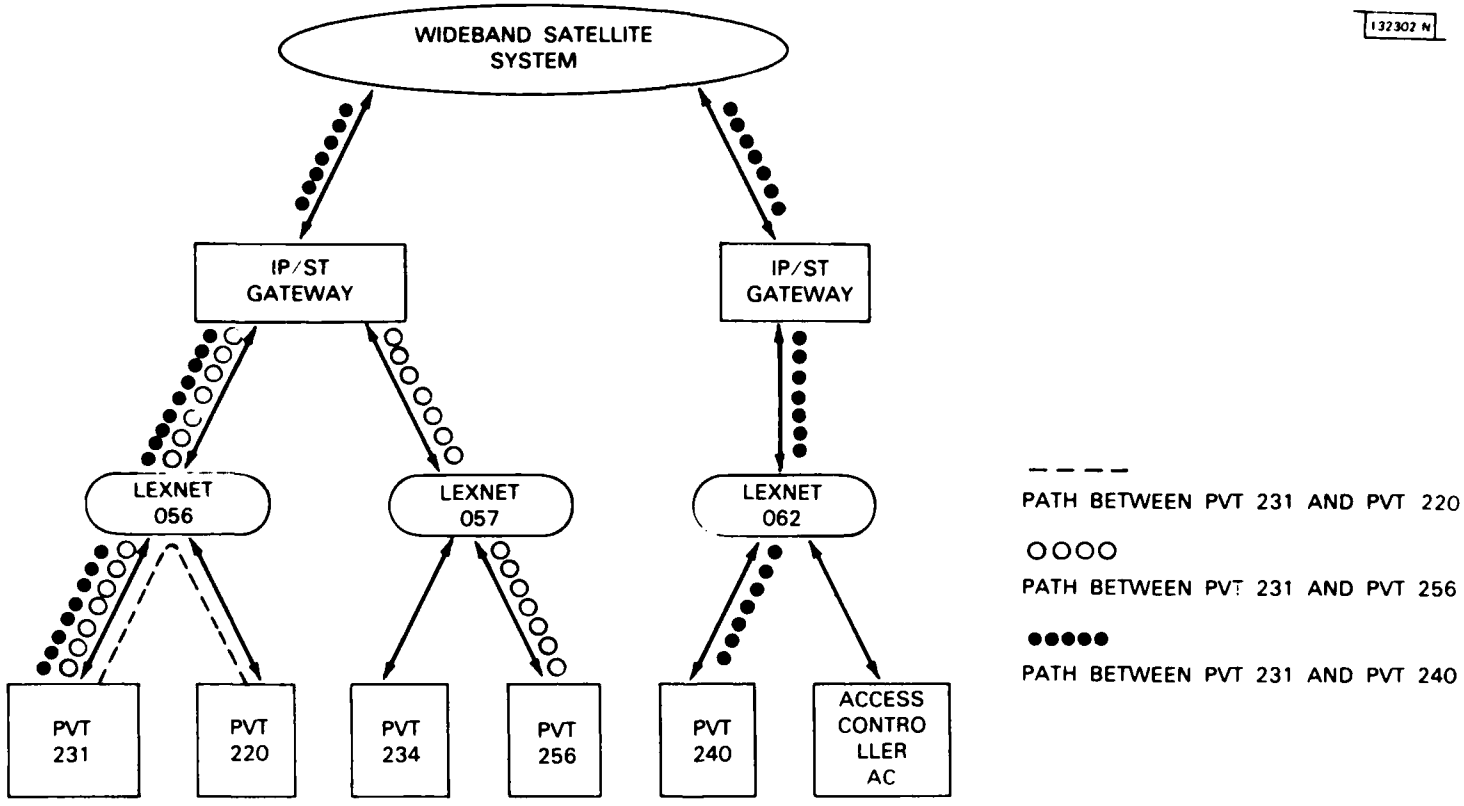


Fig. 5 Typical wideband configuration

followed by its local number. To make a call to another PVT the caller merely dials the number of the PVT he wishes to reach. If a call is being made to another PVT on the same local LEXNET, the LEXNET number may be omitted when dialing. See Section 5 for a discussion of the dialing conventions. If the caller dials a PVT on another LEXNET, the call must go through a GATEWAY on the local LEXNET. If no GATEWAY is attached to the local net, such a long distance call is impossible. The PP gives the caller back a Dial Tone to signal that the call could not be established. Otherwise the PP attempts to establish the call using NVP-II and ST protocols. Insuring reliable call set-ups is discussed in Section 7. The messages exchanged in setting up a successful PTP call are shown below.

4.2 Setting Up a Point-to-Point Call

The NVP-II protocol is very general and allows any combination of tokens in a message. To make implementation easier and to avoid ambiguity, the implementation in the PVTs defines which tokens must be present in various call setup messages. Three tokens must be included in the initial message called a Want-To-Talk message (WTT). The Please Echo token needed by the reliability code (see Section 7) is included in every NVP-II protocol message. Other tokens may be added but the WTT will be rejected if the basic three tokens are not included. Since the Connection-Name Token identifies messages relating to this call, it must be the first token in the messages. Figure 6 diagrams a "Want-To-Talk" message.

The WTT always includes:

Token	CONNECTION-NAME	includes caller's address
Token	VOCODER-TYPE	specifies vocoder
Token	I-AM-READY	specifies caller is ready
Token	PLEASE-ECHO	needed for reliability

The answering Accept message is required to contain:

Token	CONNECTION-NAME	
Token	PLEASE-ECHO	
plus either		
Token	I-AM-RINGING	notes phone is being rung
or		
Token	I-AM-READY	

The receipt of an NVP-II Accept message is considered to signal acceptance of the proposed vocoder. If the Accept message contained the Token I-AM-RINGING,then when the phone is answered, a message containing the Token I-AM-READY would be sent by the called PVT. No connection is complete until both signal that they are ready. After receiving the NVP-II Accept message, the calling PVT begins the ST protocol exchange which is described by example below. Normally the NVP-II and ST protocols are completed long before the receiving phone is answered. The ringing signal is sent to the phone before the ST negotiation is begun. If the ST negotiation fails - possibly because some network cannot allocate resources for another call - a message closing down the call is sent back to the PVT. The local PVT will then silence the phone and be ready for another call. This could be annoying if it happened often. It can be avoided by completing the ST connection before ringing the phone. The ST connection negotiations have never failed on our net, after a successful NVP-II negotiation. Ringing the phone as soon as possible shortens the total set-up time.

The following shows the sequence of protocol exchanges for PVT 220 on LEXNET 056 establishing a PTP call with PVT 240 on LEXNET 062. Since these

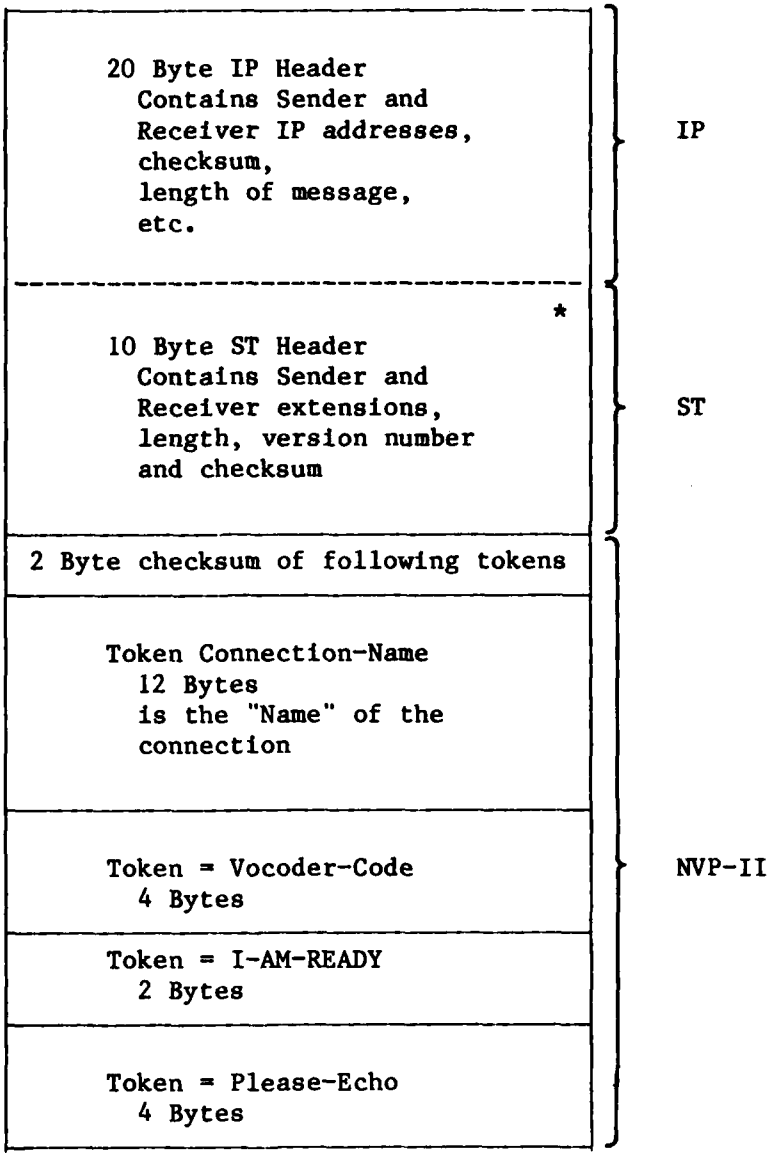


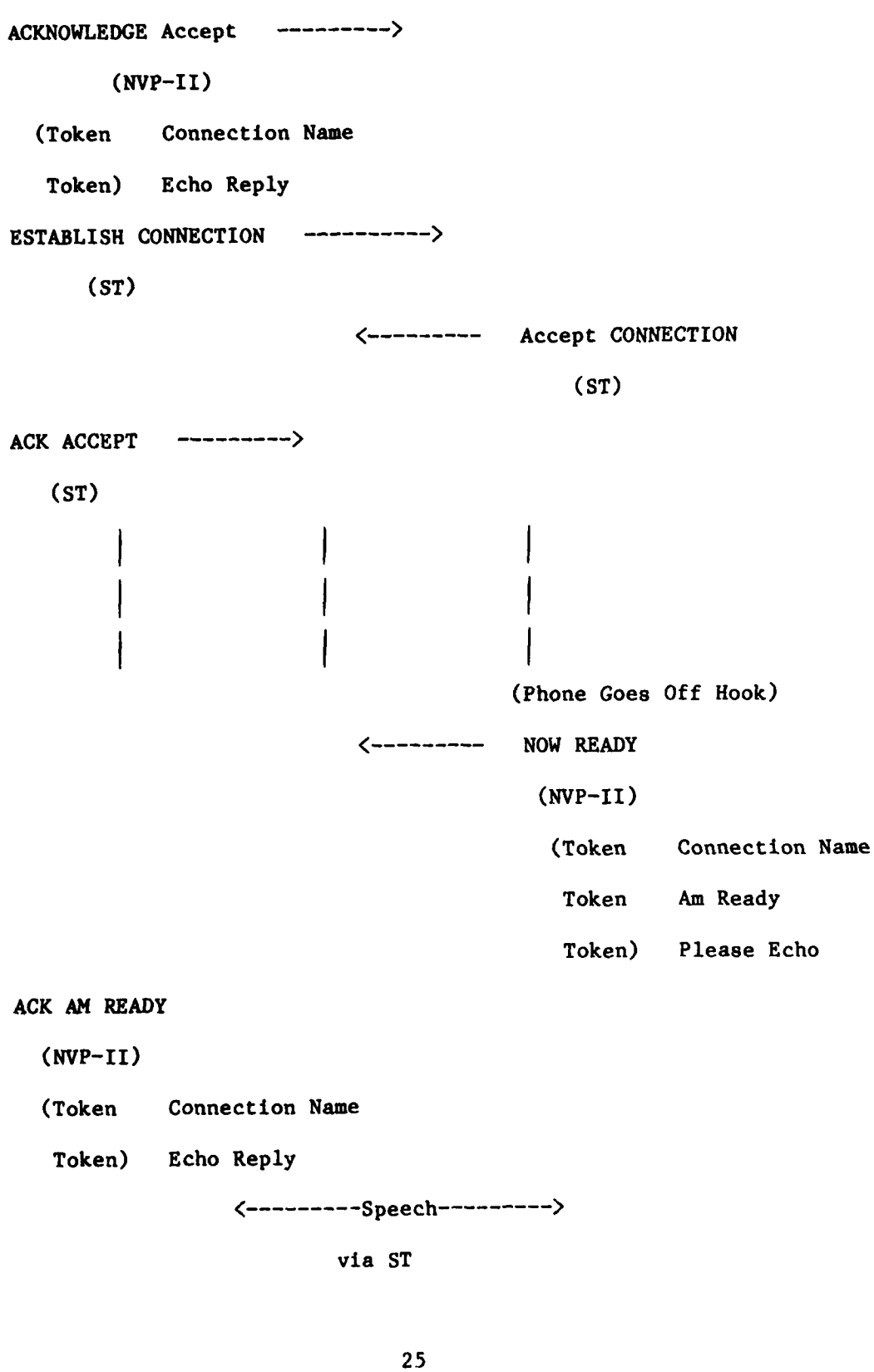
Fig. 6 INITIAL MESSAGE TO ESTABLISH A POINT-TO-POINT CALL

*This ST header is inserted between the IP header and the NVP-II data to provide an additional address field. This is needed to distinguish between callers if there is more than one at a single site.

PVTs are on different LEXNETs, the protocol messages pass through one or more network GATEWAYs. The GATEWAYs simply pass the IP messages through. The ST messages CONNECT and ACCEPT inform the GATEWAYs about the call being set up. The GATEWAYs determine the "best" route for the call, allocate the necessary resources and create tables containing this information. These tables determine how the ST speech messages are forwarded. This is transparent to the PVTs.

MESSAGES SENT TO ESTABLISH A TYPICAL PTP CALL

PVT (056-220)		PVT (062-240)
---		---
INITIATE CALL	--------->	
(NVP-II)		
(Token Connection Name		
Token Vocoder Type		
Token Am Ready		
Token) Please Echo		
		(Ring Phone)
	<---------	Accepting
		(NVP-II)
		(Token Connection Name
		Token Am Ringing
		Token Please Echo
		Token) Echo Reply


ACKNOWLEDGE Accept ---------->
(NVP-II)
(Token Connection Name
Token) Echo Reply
ESTABLISH CONNECTION ---------->
(ST)
<---------- Accept CONNECTION
(ST)
ACK ACCEPT ---------->
(ST)
(Phone Goes Off Hook)
<---------- NOW READY
(NVP-II)
(Token Connection Name
Token Am Ready
Token) Please Echo
ACK AM READY
(NVP-II)
(Token Connection Name
Token) Echo Reply
<----------Speech---------->
via ST

4.3 Conferencing

4.3.1 Access Controller

In addition to PTP calls the PP software can set up a conference call [14] among any number of callers who have access to PVTs. The conference capability utilizes an ACCESS CONTROLLER (AC). The AC is a system-wide resource whose address must be known to all potential conference particiants. Such an address is referred to as a "WELL KNOWN ADDRESS". The AC is implemented in a PVT which currently resides on a LEXNET at Lincoln Lab. Its address is Net 62 Host 156. This is shown in Figure 5. The AC can contain information about many conferences. Currently the AC contains information about three conferences - one for each implemented vocoder. This information has been prestored and is always available for use. Ideally a user could use his telephone key pad to cause the information defining a new conference to be sent to the AC. However, this has not been implemented. The information defining a conference includes: conference name, password, vocoder to be used and optionally a list of the IP addresses of acceptable participants. If the particpant list is omitted anyone is allowed to join.

While a conference is in progress the AC also maintains a list of the IP addresses of the conferees. This information is supplied to each conferee as he successfully joins a conference. Information about conference participants may always be requested and received by conferees and interested GATEWAYs.

4.3.2 Setting Up a Conference

A conference may be set up or joined in several ways. Our primary implementation uses a "MEET ME" style. A group of prospective conferees

agree in advance on the time of the conference and the vocoder they wish to use.

On or after the appointed time each conferee attempts to join the conference. To join such a conference a conferee dials the name of the conference which uses the agreed upon vocoder. His PP sends a WANT TO JOIN message to the AC asking to join the conference.

The NVP-II protocol for a conference specifies that a conference should have a password known to the AC. The AC will only accept a PVT that supplies the correct password as part of its WANT TO JOIN message. The current implementation considers the conference password to be identical to the conference name. At a later date a more sophisticated password system can easily be installed.

If the PVT is accepted into the conference, the AC will supply information about the conference such as the vocoder to be used. As soon as the PVT is accepted by the AC, the PP sends a RINGING signal to its phone. If others are already in the conference it is the PPs responsibility to establish a connection with each one using ST. As soon as the first connection is established, the RINGING tone is turned off and the vocoder is started. During a conference each site keeps a list of the sites to which it has a connection. When a participant hangs up, disconnect messages must be sent to the AC and to all the other participants. A participant who hangs up or whose site crashes can rejoin the conference at will.

A person who wishes to spontaneously set up a conference may use the "INVITE IN" feature. First he dials the conference name of the type of conference he wishes to establish. After being accepted by the AC he then

can dial the phone number of others he would like to have join his conference. His PP recognizes this dialing as a desire to have a remote PVT join the current conference. The PP sends the remote PVT an IP datagram which contains the NVP-II token "PLEASE JOIN MY CONFERENCE". The token contains the necessary information to allow the remote PVT to join. The PP at the remote PVT immediately rings its phone. When the phone is answered the PP sends a "WANT TO JOIN" message to the AC and proceeds as described above. These two ways of entering a conference may both be used to set up a single conference. If a "MEET ME" conference is in progress and someone else is needed, any participant can "INVITE IN" the needed PVT. One participant who has been "INVITED IN" can then "INVITE IN" another.

A conference can also be set up using voice commands. VCOP (Voice Conference Operator) [15],[16] is now operational. VCOP includes: 1) a word recognition system to identify spoken words; 2) a speech synthesizer to produce speech feedback, prompts and acknowledgements; 3) interfaces from the recognizer and the synthesizer to a PVT, and software to control the recognition system and the synthesizer. The Threshold Technology T580 speaker-dependent word recognition system and a Lincoln LPC Speech Processing Periperal [8],[16] have been used. Communication between VCOP and the PP occurs through the USART. To the PP the VCOP takes the place of the user telephone. One PVT with a WELL KNOWN ADDRESS is connected to VCOP. To establish a conference using VCOP, the conference initiator dials a PTP call to the PVT connected to VCOP. When the connection is made, the caller is greeted by VCOP. VCOP asks the caller questions about the conference he wishes set up. After VCOP has information on the vocoder to be used and the

names of the people to be asked into the conference it requests the caller to hang up and wait for the conference to be set up. When the connection between the VCOP PVT and the caller is closed, VCOP translates the names of the desired conferees into PVT addresses and selects the correct conference name. VCOP then passes this information to its PP. The PP sends each conferee a "PLEASE JOIN MY CONFERENCE" message. To ensure reliability all conference protocol messages use the retransmission mechanism described below. When this is completed VCOP is available to set up another conference.

The Access Controller acts as a central storehouse of information about conferences. The protocols could allow as many as 128 participants in a conference, but the current AC implementation is limited to 32 participants. As each PVT is accepted into a conference it is assigned an identification number. These identification numbers are assigned in numerical order from 1 to 32. The identification number corresponds to a bit in a 32-bit conference bit map. A conference bit map is maintained by the AC, the PVTs and the GATEWAYs involved in the conference. The bit map provides an efficient means of referring to the conference participants. When a PVT notifies the AC that it is leaving the conference, its bit is cleared. A bit in the bit map is never reassigned to another PVT during the life of a conference. If a conferee hangs up and then later rejoins a conference, his PVT will be assigned a new identification number. The conference ACCEPT message from the AC to a PVT tells the PVT its identification number (bit number in the bit map) and also contains a copy of the current conference bit map. The PP should set up a connection to every PVT with a lower bit number than it's own and be ready to accept a connect from PVTs with higher bit numbers.

This system has made provisions for a crashed site to rejoin a conference. When a site crashes, neither the other conferees nor the AC know that it is out of the conference. If the site comes up and redials the conference, the AC will reassign it its former bit number. To the AC, the "WANT TO JOIN" message is either a duplicate sent by the site's reliability/ retransmission mechanism or the site crashed and is reentering the conference. The AC sends out a duplicate ACCEPT message reassigning the site its original bit in the bit map. The PP of the reentering PVT issues an ST CONNECT to all conferees with lower bit numbers and issues an ST ACCEPT to all conferees with higher bit numbers. The PP has been programmed to accept these ST CONNECTs and ST ACCEPTs from conferees who it thought were already in the conference or to whom it had initially been unable to connect. The ACCEPT message from the AC to a PVT contains a copy of the current conference bit map. When the PP receives the ACCEPT message from the AC, it can ascertain how many other conferees are in the conference by counting the number of bits set in the conference bit map. The PP does not know who the participants are. The PP sends messages to the AC requesting the address of each participant.

Two commands are needed for the PVT to request and receive the addresses of the other participants from the AC. These were not defined in NVP-II, but have been provided in ST. The commands are TELL-ME and INFO. Since there is no ST connection between a PVT and the AC these commands are sent as the data portion of an IP message. The voice protocols allow NVP-II tokens and ST commands to be transmitted via ST or IP. Commands and tokens are treated identically regardless of the transport protocol. The AC forms a bridge

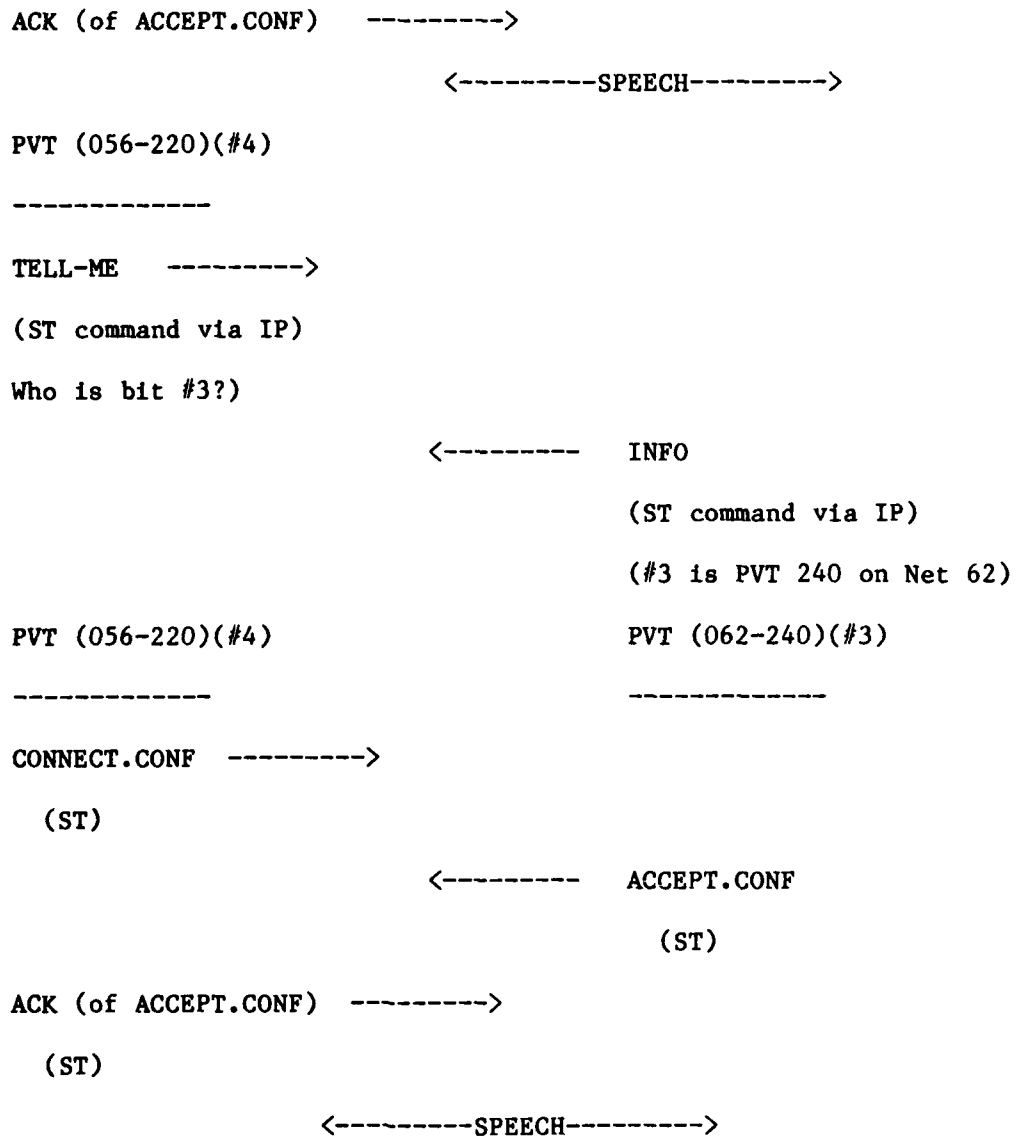
between the IP and the ST protocol levels, but since the AC receives only IP messages, it must be told whether the data portion of a message contains ST commands or NVP-II tokens. Therefore, IP messages containing NVP-II tokens are sent to extension 122 at the AC, while those containing ST commands are sent to extension 123.

TELL-ME is issued by a PVT to the AC and requests information about one or more conference participants. The participants are identified by their bit number in the conference bit map. The AC responds with an INFO message which contains the requested addresses. On receiving this information, the PP issues the appropriate conference connect (CONNECT.CONF) or conference accept (ACCEPT.CONF) message.

The following scenario illustrates the protocol messages sent when PVT(056-220) joins a conference. PVT(062-240) and PVT(062-156) are already in the conference.

PVT (056-220)		AC (062-156)
WANT TO JOIN	--------->	
(NVP-II)		
(Token	Want To Join	
Token	Conference ID	
Token	Conference Password	
Token	User Address	
Token)	Please Echo	

```
                                <---------   WELCOME
                                              (NVP-II)
                                              (Token      Conference ID
                                               Token      Connection ID
                                               Token      Conference Style
                                               Token      Vocoder Type
                                               Token      Conference Bit
                                                            Map
                                                  (Your bit number is 4.
                                                     Bit numbers 1 and 3
                                                     are active.)
                                               Token)     Echo Reply
PVT (056-220)(#4)
-------------
TELL-ME    --------->
(ST command via IP)
(Who is bit #1?)
                                <---------   INFO
                                              (ST command via IP)
                                              (#1 is PVT 235 on Net 56)
PVT (056-220)(#4)                             PVT (056-235)(#1)
-------------                                 -------------
CONNECT.CONF   --------->
  (ST)
                                <---------   ACCEPT.CONF
                                               (ST)
```



4.3.3 Conference Floor Controller

Since a vocoder can only play out one received speech signal at a time, some selection method is required. A distributed Floor Controller has been implemented. The same small control module operates in each PVT in a conference. This module controls the sending and playing out of speech data at its local site. It is a floor controller for its own PVT. The floor

controller is voice activated. It only attempts to send speech when the silence detection mechanism indicates that the user is talking. The Floor Controller will not transmit speech messages unless it has not received speech for a sufficient period to merit deciding that the previous speaker has finished. When a PP begins sending out speech, it also sends an IP control message containing the NVP-II token "I WANT TO TALK". This token is also included in the speech message. A PVT receiving this token considers the sender to "have the floor". A conflict can arise if two conferees start talking at roughly the same time. This is resolved by assigning each conferee a priority. If a PVT is listening to one site and receives an "I WANT TO TALK" token from a higher priority site it begins listening to the new site. The first site should also receive this token and yield the floor. This preemption occurs only in the short time window when there is no speaker established and more than one site attempts to transmit. This could be extended into an interrupt capability if desired. Currently priority is determined by a PVT's bit number in the conference bit map. Bit 1 has highest priority. This tends to give high priority to the person who initiated the conference. Other schemes for deciding priority could easily be implemented.

4.3.4 Routing Conference Speech Messages

Each PVT in a conference must maintain a current copy of the conference bit map. A PVT which is currently the conference talker sends out only one copy of his speech messages. The LEXNET which is the local network for PVTs is a broadcast network. Each PVT notifies its BC that it wishes to receive messages relating to this conference. The BC checks each message that it

finds on its LEXNET and passes to its PP all messages addressed to it and all messages with a conference address for which the PP has requested to receive messages.

The GATEWAYs handle the routing of speech messages to PVTs on other nets. The ST CONNECT.CONF and ACCEPT.CONF between the local PVT and the remote PVT go to the GATEWAY when the connection is set up. These messages contain the identification numbers (bit numbers in the conference bit map) of the sender and receiver of the message. The GATEWAY issues TELL-ME messages to the AC to obtain the IP addresses which correspond to these identification numbers. The GATEWAY then selects a route and propagates the messages to another GATEWAY or PVT. The GATEWAY records this information and builds tables to guide it in handling the conference speedmessages.

Each speech message carries in its header the conference bit map. The PP sets the identification bit in the conference bit map of all the PVTs to which it is connected. It clears its own identification bit. A copy of the message is to be sent to each participant whose bit is set. The header of a speech message contains the conference ID and the bit map. The GATEWAY forwards these messages to each PVT whose bit is set. This mechanism could be used to send messages to a subset of the current conferees if desired. Currently all speech messages go to all conferees.

5. DIALING CONVENTIONS

5.1 Interaction with the Caller

PVT dialing conventions differ from those of the regular phone. No timer is used to determine the completion of a dialing sequence. All dialing sequences can be parsed to completion. The characters * and # are used to

delimit special dialing sequences. Such sequences are dialed before the destination number is dialed. The caller by dialing can make a Point To Point (PTP) call to another phone, Join a Conference, Invite others into a conference and give some limited instructions to his PVT.

When an "Off Hook" is received, a dial tone is sent to the phone. When an "On Hook" is received, silence is sent to the phone. When a dialing sequence finishes, the PP will attempt to honor the caller's request.

To establish a connection, the PP enters into a protocol dialog. If the remote phone is busy, a busy signal is sent to the local phone. If the remote phone can not be reached, a dial tone is sent to the local phone to indicate that the call has been aborted and that another call may be tried. If the protocol dialog is successful, the remote PP will ring its phone and the local PP will send a ringing signal to its phone. When the remote phone is answered, a silence command is sent to both phones and speech transmission begins. If the remote phone fails to answer, the connection is terminated when the local phone hangs up.

5.2 Dialing a Point-To-Point Call

A simplified addressing convention has been implemented in the PVTs. It is expected that at a later time the address of a PVT will be redefined in keeping with a more general wideband network addressing scheme. Currently the address of a PVT consists of two three-digit numbers--its LEXNET number and its own "host number" on the LEXNET. A LEXNET normally supports several PVTs and is also connected to a GATEWAY. PVTs on different LEXNETs can communicate via these GATEWAYs. A typical configuration is shown in Figure 5. To call a PVT on the local net a caller merely dials the three digit host

number of the other PVT. To call a PVT on a distant LEXNET the caller first dials '9' to indicate that the call must be routed via the GATEWAY and then dials the three digit LEXNET number followed by the three digit host number. A caller can route a call to another PVT on his local net through the GATEWAY by dialing '9' followed by the local LEXNET number and the other PVTs host number. For testing purposes the capability, not found in normal phones, to call oneself was added to the system. A Dialing Matrix showing the correct dialing between sites is shown in Figure 7.

5.3 Conference Dialing

Several special dialing conventions have been implemented. These conventions use the special characters * and # to indicate the boundaries of the string being dialed. To join a conference ("Meet Me" style) the name of the conference is dialed preceded by a # and followed by a *. #234* means that the caller wishes to join a conference called 234. A string of digits preceded by a # and followed by a * is also used to pass to the PVT the address of a PVT which should be "Invited Into" a conference. To "Invite In" PVT 234 on LEXNET 062, one keys in #062 234*. If both PVTs are on the same LEXNET, only #234* must be keyed in. The PVT distinguishes these cases by noting whether or not it is in a conference when it receives a string of digits preceded by # and followed by *.

5.4 The Echo Extension

A dialed number may optionally be preceded by an extension number (the extension number must itself be preceded and followed by *). For a PVT only extension 1 has meaning. Extension 1 is the ECHO extension. When a call comes in for extension 1 the phone is not rung and all the received speech

FROM	TO LL (062)	LL (063)	ISI (061)	SRI (053)
LL (062)	NNN	9-063-NNN	9-061-NNN	9-053-NNN
LL (063)	9-062-NNN	NNN	9-061-NNN	9-053-NNN
ISI (061)	9-062-NNN	9-063-NNN	NNN	9-053-NNN
SRI (053)	9-062-NNN	9-063-NNN	9-061-NNN	NNN

Fig. 7. Dialing matrix for point-to-point calls. The three-digit number assigned to the LEXNETs at each site is fixed. The three-digit host number assigned to a PVT is determined by the setting of a thumb wheel switch on the back panel. Since the host number of a PVT may change, it is represented by NNN in the matrix. LEXNETs 062 and 063 are at Lincoln Laboratory. LEXNET 061 is at ISI and 053 is at SRI.

packets are merely echoed back to the sender. This has proved to be useful for testing and for demonstrating the system when no one is available at a remote site. To call the ECHO extension for PVT 234 on LEXNET 062 from another LEXNET, the dialer keys in *1* 9 062 234. This causes a 1 to be put in the extension field of the called terminal's address in the IP header.

5.5 Using the Switched Telephone Network Interface (STNI)

Information Sciences Institute has developed the Switched Telephone Interface card (STNI) [17] to allow interconnection between the packet network and the commercial telephone system. The STNI is contained on a single card and plugs into the PCM card slot in any PVT. The STNI communicates with the PP module over the USART. To the PP the STNI card acts like the user's telephone which is usually connected to the USART. An STNI equipped PVT runs the same PP program as other PVTs except that the dialing module written to handle the user phone is replaced with a dialing module written by ISI specifically for the STNI. An STNI equipped PVT becomes a "GATEWAY" between a packet net and the commercial phone system. Since such a PVT is not equipped with a phone, it may not be used to place a call.

When a call is to be made from one PVT through another PVT equipped with an STNI card out into the regular phone system, the regular phone number to be called by the STNI must be dialed in. This is done by dialing #, the regular phone number, then *, followed by the number of the STNI-equipped PVT. The digits between the # and the * are then passed to the STNI card in the Want-To-Talk message as the data of a "PLEASE DIAL" token. This outside phone number preceded by a # and followed by a * must be dialed before the address of the PVT containing the STNI card. To call the weather information

number in Los Angeles through an STNI-equipped PVT whose host number is 244 on LEXNET 062 from a PVT on another LEXNET, one would dial #9 554 1212* 9 062 244. Requiring a caller to dial the regular phone number in this manner was a convenient way to implement this feature in the PVT code. Other conventions are feasible such as implementing a timing mechanism to determine when the user has finished dialing a number. Then a second dial tone could be used to signal the user that the second phone number could be keyed in. This would avoid the use of # and *.

All STNI-equipped PVTs at Lincoln have been assigned a regular extension number from the Laboratory switchboard. To call a PVT from a regular phone, the user first dials the Lincoln extension assigned to an STNI-equipped PVT. After an initial ringing tone, a second dial tone is heard. Now the number of the destination PVT must be dialed. (Any commercial phone system number may also be dialed instead of the PVT number.)

5.6 Special Dialing Sequences

When a special dialing sequence is to be used in conjunction with a normal dialing sequence, the special sequence must be keyed in first. This avoids a waiting loop in the dialing code. If an optional special sequence could follow the dialing of a remote site, the protocol would either have to wait some indeterminate amount of time to ascertain whether the dialer was going to key in such a sequence or would have to send the special sequence in a later control message.

Dialing a *# at anytime is equivalent to hanging up the phone and then picking it up again. The call in progress is cancelled and the program is reinitialized. While shutting down is in progress, the PVT can not handle

another call. The caller will hear a fast busy signal and his dialing will be ignored. As soon as the shut down is complete, the caller will hear a dial tone and is free to dial again.

The *# feature is needed by the STNI card. A call may be set up between a regular phone and a PVT by routing the call through an STNI-equipped PVT. It is important that if either talker terminates the call this knowledge reaches the STNI-equipped PVT and the other caller. Then the connection can be taken down and the equipment made available for another call. If the PVT phone hangs up, the ST protocol message DISCONNECT (or REFUSE) is sent to the STNI and the call is correctly taken down. If the regular phone hangs up, some local switches (such as the one at Lincoln) do not send out a disconnect signal. Using a regular telephone, once a connection is established the tones generated by key pushes are passed through on the line. The STNI is able to receive and interpret these tones. Therefore, if such a regular phone user keys in *# before hanging up his phone, the STNI card will interpret the tones correctly and take down the connection. The general facility for any caller to be able to use *# provides one convenience not available on most regular phones.

A string of digits preceded and followed by a # is ignored. This provides a handy way of aborting dialing if an error is made in the conference name or in the address of a PVT to be "INVITED IN".

6. SPEECH DATA

6.1 Packetizing Speech Data

In addition to PCM, the PP can handle a 2400 bps Linear Predictive Coder (LPC) vocoder and an Embedded CVSD (ECVSD) vocoder. The ECVSD vocoder can operate at one of four rates: 16000, 32000, 48000, and 64000 bps. Speech

data are digitized by the vocoders on a parcel basis. Buffers are established in the 4K bytes of DMA memory in the PP. Each buffer is the size of a parcel of speech plus an initial one-byte header. Speech goes to and from the vocoder via two direct DMA channels each working in auto-initialize mode. In this mode each DMA reads/write speech from/to its buffers until it reaches the end of the last buffer assigned to it. It then automatically begins again at the beginning of it's first buffer.

A speech message consists of an ST header and and NVP-II header optionally followed by NVP-II protocol tokens and then one or more consecutive parcels of speech. The one byte header used in the vocoder buffers is not included in the message. The number of parcels of speech included in a speech message varies depending on the vocoder in use. The timestamp of the message is the timestamp of the first parcel of speech in the message. Figure 8 shows a speech data message.

For PCM and ECVSD, a parcel has been defined to represent 22.5 msec of speech and consists of 180 bytes of data. Space is available for a total of 20 buffers (3620 bytes). For PCM, 4 of the 20 buffers are used to hold speech coming from the vocoder. This is enough to insure that the speech has been packetized and sent out before it begins to be overwritten four frames (90 msec) later. Sixteen buffers are used to collect and reconstitute the incoming speech data. Packetizing ECVSD speech is discussed below.

A parcel of LPC data represents 20 msec of speech and is 6 bytes long. Because of the comparably small parcel size there is ample buffer space in DMA memory. For LPC, 128 buffers are provided in each direction. LPC could have been implemented with fewer buffers but as long as space is not a

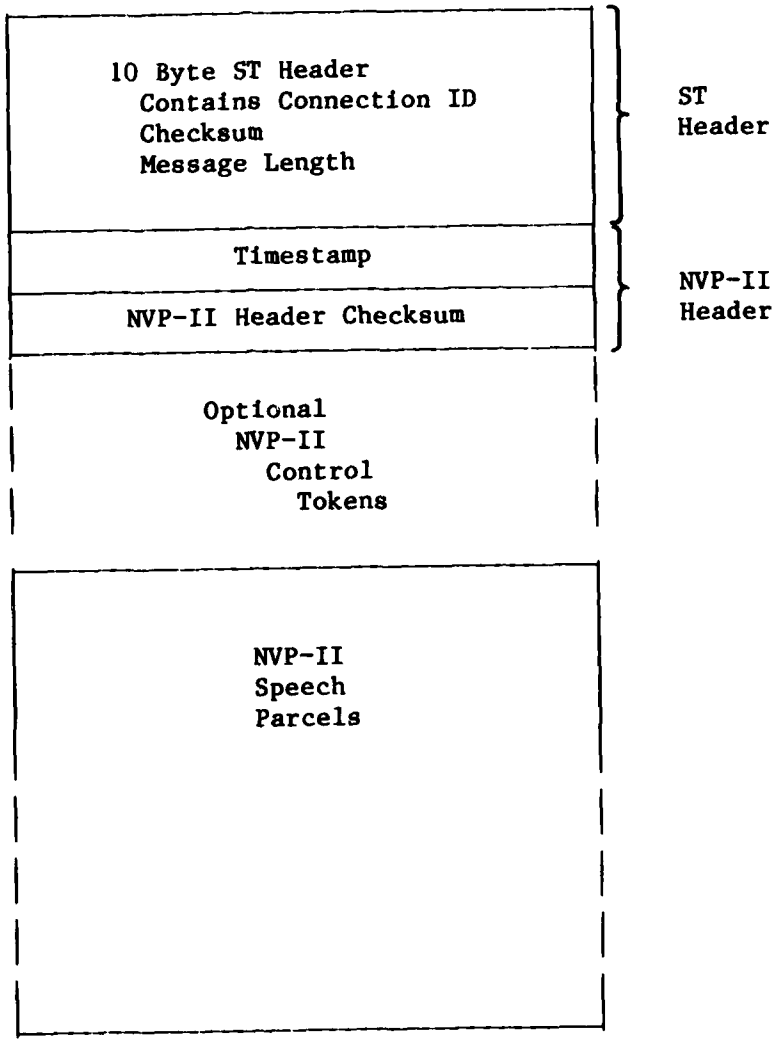


Fig. 8. NVP-II speech parcels transmitted via ST.

problem any convenient number of buffers can be used. Using a relatively large number of buffers lessens the likelihood that a very late message will be mishandled by the reconstitution algorithm. This is discussed further in Section 6.4.

For PCM with a parcel size of 180 bytes, one parcel of speech data is put in each message. For LPC with a parcel size of 6 bytes, the number of parcels in a message is a variable which can be set to match the demands of the transport medium.

Packetizing ECVSD data is more complicated. The ECVSD algorithm was designed to allow the vocoder to transmit and receive at four bit rates: 16, 32, 48, and 64 kbps. To do this, the vocoder arranges the data in four 45-byte slices in the parcel. Slice 1 - the first 45 bytes - contains the data required for 16 kbps CVSD. Slice 1 plus slice 2 is equivalent to 32 kbps CVSD, etc. Four bits are used in the header byte to indicate the presence of each slice. As the slices arrive and are transferred via DMA into their proper position in a speech buffer the bit which indicates that this slice has arrived is set. To reconstitute the speech parcel, at least slice 1 must be present. The vocoder will use slice 1 and all the other consecutive slices which have arrived to reconstitute the parcel. Speech quality depends on the number of slices available.

At times various nodes in a network become overloaded and are forced to discard messages. If messages were prioritized the nodes could discard the least important messages in their queues [18]. The ECVSD algorithm makes it possible for the PP to send speech messages with four priority levels. Only slices of the same priority from adjacent parcels are combined in one speech

message. If a speech message contains slice 1 data from one or more parcels of speech, it is a priority 1 message. If it contains slice 2 data, its priority is 2, etc. The priority of each message is specified in its header.

This allows a heavily loaded GATEWAY to check the priority of speech messages and discard low priority messages when necessary. Since there can be up to four slices in a data parcel, the PP must be able to combine slices across four consecutive parcels. Therefore, it is not adequate to provide only four buffers for speech coming from the vocoders. For ECVSD eight buffers are used for speech coming from the vocoder and twelve buffers are available for speech coming in from the local net. Sending a large number of messages puts an added processing burden on the GATEWAYs. Therefore the PP never sends more than one data message per frame time. When sending ECVSD at a rate of 64 kbps, the slice 1 data from parcels 1 thru 4 will be sent in one priority 1 speech message during frame 5. The slice 2 data will be sent during frame 6, etc. If sending at a 16 kbps rate, a one slice message can be sent every frame, or a two slice message can be sent every other frame, or a three slice message can be sent every third frame or a four slice message can be sent every fourth frame. To allow experience to be gained on the results of different packetizing methods, command sequences on the key pad may be used to change the bit rate and packetizing method at any time. The program is initialized to send at 64 kbps. The command sequences are discussed below.

The receiving PVT can unpack the data rather easily. The priority of the message tells what level slices it contains. The timestamp is the timestamp of the first of consecutive data parcels. The size of the data

contained in the message determines the number of slices it contains. Therefore the two PVTs can transmit at different rates and do not need to notify the other when they change rates.

To specify a bit rate and packetizing method for ECVSD two digits are keyed in preceded and followed by a star. This may be done at any time after a connection is established. The second digit is the number of slices of data to send from each frame. A 1 is one slice or 16 kbps up to a 4 for four slices or 64 kbps. The first digit is the number of adjacent frames across which slices should be combined into one message. *41* would send 1 slice per frame (16 kbps) but would combine the first slice from four adjacent frames into one message. Therefore a speech message containing four priority 1 slices would be sent every four frames. *11* would also send at 16 kbps but a speech message containing one priority 1 slice would be sent every frame. In order to send only slices of the same priority in each speech message the number of frames combined must be equal to or greater than the number of slices sent per frame. Possible combinations are shown in the Table II below.

TABLE II

NUMBER OF FRAMES TO BE COMBINED

		(first digit)			
		1	2	3	4
(second digit)					
Number of	1	x	x	x	x
Slices	2	-	x	x	x
Per	3	-	-	x	x
Frame	4	-	-	-	x

Note: Both extension numbers and ECVSD packetizing instructions are given by preceding and following the digits by *. Confusion is avoided by the following rules: Extensions are one digit long; ECVSD instructions are two digits long; all other lengths are ignored.

Several factors influence how many speech parcels are packed into one message. The overhead is reduced if many parcels are included since the size of the header does not change. Some speech packets may be lost in transmission. If a single message contains too much speech its loss will seriously effect the quality of the received speech. Also if many parcels are contained in a single message, an added initial delay is imposed on the system since a message can not be sent until sufficient parcels have been buffered up.

A conference speech message containing no data tokens has a header of nine 16 bit words (144 bits). Sending two LPC parcels per message results in a message containing 9 words of header and 6 words of data. It represents 45 msec of speech and imposes an initial 45 msec delay before the first message is sent. Sending 18 parcels results in a message containing 9 words of header and 54 words of data. However, this message would represent slightly over 400 msec of speech. Its loss would cause a troublesome glitch in the received speech.

6.2 Silence Detection

During a conversation, each participant can be expected to be silent slightly more than half the time. It is wasteful to send speech messages when the data represents silence. Each of the vocoders used by the PP provides Silence Detection as part of its algorithm. The vocoders store the

current parcel of "speech" parameters into the PP buffers at all times. Based on its Silence Detection algorithm (which is vocoder dependent), the vocoder program sets the SPEECH bit in the one byte header following the speech data only when "speech" is present. The vocoders generally smooth the speech by allowing several low energy parcels to occur before declaring "silence". The PP checks this flag when preparing speech packets. No speech message is sent unless it contains at least one parcel of speech.

For use in debugging and to provide a constant stream of speech messages for experiments, a three position switch is provided on each PVT. In the center or normal position, the Silence Detection algorithm decides when speech messages should be sent. In the up position "speech" messages are sent continuously regardless of the results of silence detection. In the down position no speech messages are sent.

6.3 Vocoder Dependent Modules

Each vocoder type (i.e., PCM, LPC and ECVSD) requires different handling in the PP. The PP has special routines tailored to handle each type of vocoder data. The routines which send and receive data check which vocoder is currently in use and call the appropriate subroutine. The initialization routines also contain special code to correctly set up the buffers and to initialize variables for each vocoder. The PP at initialization reads the Vocoder Selection Switch and prepares to handle that vocoder.

If the setting of the selection switch is changed the PP executes the routines done at initialization time and "flips" vocoders. This can be done at any time although if it is done in the middle of a talkspurt, a slight

transient may be noticed. However, since changing vocoders may change the data rate and since the GATEWAYs are not notified, it should be done only when "Talking-To-Yourself" or when talking to another PVT on the same net. The implementation in the PP allows switching at any time, but the protocol does not. The protocol could be extended to include this. A new protocol message announcing a change of vocoders would have to be defined and implemented if this feature is ever supported on an internetwork basis.

The same vocoder must be used by both talkers. When a Want-To-Talk message is received from a remote phone it specifies what vocoder it wishes to use. If the vocoder currently selected is not the same as the requested vocoder, the PP will flip the selection switch internally and check the other available vocoder. If a match is achieved, the incoming call will be accepted. Otherwise the selection switch is returned to its former position and the incoming call is rejected.

6.4 Efficient Handling of Speech Data

It is essential to use DMAs to move the data across the interface between the PP and the BC and across the interface between the PP and the vocoder. Without DMAs, speech parcels would have to be copied across each of these interfaces using the CPU on the 8085 chip. Copying over data via programmed I/O is very time consuming and a PVT without DMAs could only support low bit rate vocoders. It could not keep up with a 64 kbps PCM vocoder.

Using a DMA to read speech data on a parcel basis requires resetting the DMA for each parcel. The DMA must be provided with the new address and byte count. When a 64 kbps vocoder is in use, the two DMAs servicing the vocoder

read and write a byte every 125 microseconds. There is not time enough between operations to reliably reset the DMAs without disrupting the speech.

DMAs can be operated in a so-called auto-increment mode. A separate data area is reserved in the DMA memory for each channel. When each DMA channel is started up, it is given the length and address of its data area. The DMA begins reading or writing speech from or to its data area and when it reaches the end it automatically starts over again at the beginning. By using the auto-increment mode, we avoided the timing problems involved in resetting the DMA. However, speech parcels coming from the vocoder must be moved out of the data area before they are written over by the DMA on its next pass. Conversely, the incoming speech parcels must be moved into place in the data area before the DMA reads that part of the data area again.

The PP has access to only 4096 bytes of DMA memory. This must be divided into data areas for the two DMA channels servicing the vocoder, space for messages coming from the BC, and space to form messages to be sent out to the BC. The restrictions imposed by the need for speed and by the lack of space strongly influenced the final design.

Protocol messages from the BC and the headers of speech data messages from the BC are read into an input buffer for which 96 bytes of DMA memory have been set aside. 380 bytes are used as buffers to form messages that must be sent to the BC. A maximum of 3620 bytes of DMA memory are used for the data areas of the DMA channels servicing the vocoder. The amount of memory needed depends on the vocoder and its parcel size. 3620 bytes provides for twenty 181-byte buffers. PCM and ECVSD have 180-byte parcels and each parcel needs a one-byte header. For PCM a data area the size of four of

these buffers is used for speech coming from the vocoder and a data area the size of sixteen buffers is used for the incoming speech. ECVSD uses eight for speech from the vocoder and twelve for incoming speech. LPC uses a 7-byte buffer (6 bytes of data plus one byte of header). For LPC a data area of 896 bytes (128 buffers) is used in each direction. The size of the data area for each channel is always a multiple of the buffer size. This insures that the location of the buffers in a data area does not change. During vocoder initialization a table is set up which contains the pointers to these buffers.

Software in the BC puts messages on the LEXNET and reads messages from the LEXNET. To send a speech message the header and any accompanying control tokens are assembled in an output buffer. After this is transferred via DMA to the Buffer Control, the speech data are transferred via DMA directly from the vocoder buffer to the Buffer Control. When a message comes in from the LEXNET the Buffer Control software alerts the PP by sending the byte count of the message via the DMA. The PP first uses the DMA to read in the two bytes of data which contain the length of the message. If the length is reasonable, the PP next reads in the number of bytes which correspond to a speech message header. If it is a speech message and it does not contain any control tokens, the next byte to be read is the first byte of speech data. If the message contains control tokens preceding the speech, their length can be determined from a value in the header and they are also transferred in using the DMA. The speech is then transferred via DMA directly into buffer(s) in the vocoders' data area one parcel at a time. Then the speech activity bit is set in the header byte of the buffer(s).

This method avoids having to move the speech data from the PP read-in buffer to the vocoder buffer and means that the read-in buffer can be much smaller than a PCM parcel in length. For a high data rate vocoder such as PCM, time is critical. This method saves the time and CPU cycles that would be used in moving the data using a load store loop. For a vocoder with small parcels such as LPC, it would actually be faster to use a programmed "load-store" loop to move a parcel. However, since time is not critical here, the DMA routines are also used to read in LPC data.

6.5 Reconstitution of Speech Data

Speech messages received at a voice terminal may experience a wide dispersion in their arrival times. If more than one route is possible between source and destination, messages may arrive out of order. Each message carries in its header the timestamp associated with the first speech parcel in the message. The timestamp is a ten bit number which is incremented once per vocoder frame. This timestamp can be used to determine where in the speech stream each incoming speech parcel belongs. Sixteen buffers are available for incoming PCM speech. Each incoming timestamp is adjusted as described in the following paragraph. Originally this adjusted timestamp mod 16 became the number N of the buffer where the speech should be put. To find the Nth buffer in the data area, the program used the buffer pointer table which was created when the DMA was initialized. The speech parcel was then transferred via DMA directly from the BC into buffer N. If a data message contained more than one parcel of speech, the parcels were transferred via DMA into consecutive buffers.

When the vocoder is initialized, the timestamp is set to zero. The timestamp is incremented once per parcel by the code which services the vocoder interrupt. The low bits in the timestamp always correspond to the buffer being used by the vocoder. Therefore, using the local timestamp (mod 16) to select a buffer in the data area locates the buffer from which speech is currently being transmitted to the vocoder. The difference between the local timestamp and the timestamp of the first speech message is calculated and called the OFFSET. Adding this OFFSET to the timestamp of an incoming message and using the result, should select the current buffer. Therefore, a reconstitution delay is added to the adjusted timestamp before it is used to select a buffer. The reconstitution delay should be large enough to ensure that speech parcels have arrived before they are needed for play out. The best value depends on the expected arrival dispersion and the total trip time. Since a large value increases the overall delay in the system, the value should be kept as small as possible.

To correct possible drifts in the clocks at different sites and to minimize the effect of calculating the OFFSET value from an atypical timestamp, the OFFSET is recalculated whenever speech has not been received for a quarter of a second. This tends to cause the OFFSET to be recalculated at the beginning of every talkspurt.

In PCM tests over the wideband satellite channel, cases occurred where the dispersion exceeded eight messages and this method of employing the 16 available buffers did not handle this dispersion well. Sixteen buffers cannot handle effectively a message stream with a dispersion greater than eight. Therefore, to create the smoothing effect available with 32 buffers,

a pointer table with 32 entries is now used. Sixteen of these entries point to the 16 actual buffers. The remaining 16 entries point to an imaginary buffer in nonexistent memory. The timestamp of an incoming speech message is adjusted as described above. Then the adjusted timestamp (mod 32) is used to select a pointer from the pointer table. The speech is stored in the buffer pointed to by this pointer. If the buffer pointed to is in nonexistent memory, the parcel of speech will be discarded. Using a pointer table with 32 entries allows maximum use of the 16 available buffers.

When the DMA is initialized, the timestamp is set to zero and the pointer table is created. The first 16 values in the pointer table are the locations of the 16 real buffers in order. The last 16 entries in the pointer table point to the imaginary buffer. The pointer table is constantly updated. After each parcel of speech is read out of a buffer to the vocoder an interrupt occurs and a special routine is executed. This routine copies the pointer to the just used buffer into the slot 16 entries later in the pointer table (mod 32). The original entry is then changed to point to nonexistent memory. After buffer zero is transferred to the vocoder, its pointer is copied into the 17th slot in the buffer pointer table and the zeroth entry in the table is set to point to nonexistent memory. We do not expect to receive a message whose adjusted timestamp is zero for roughly 25 frame intervals. A pointer to a real buffer will be written into slot zero of the pointer table in 16 frame intervals. If such a message is received before a pointer to a real buffer is put into slot zero, it is probably correct to discard the message. This method, suggested by James Forgie,

causes us to have 16 buffers available where we expect to need them and causes late arriving speech to be automatically discarded. For ECVSD there are only 12 buffers available. Therefore, the buffer pointer table contains only 12 pointers to real buffers and 20 pointers to the buffer in nonexistent space. For LPC the pointer table has 128 entries. Each entry is a pointer to a real buffer. The pointer table does not require updating.

Count is kept of the number of parcels discarded and a dispersion distribution is maintained. If two consecutive data messages are discarded, it could indicate that the value of the OFFSET is wrong. Therefore, the OFFSET is recalculated.

Since only the low bits of the timestamp are used, this algorithm could be fooled by a speech message that was very late in arriving. This is not a problem on our net. The high bits of the timestamp could be used to detect this case on any net where it was necessary. For networks with high dispersion, more buffers and therefore more low bits should be used. In this way, the algorithm self-adjusts and lessens the need to use the high bits in the timestamp.

7. RELIABILITY

7.1 Reliable Transmission of Control Messages

The system must insure that a PTP or Conference call is completed if possible. Control messages occasionally fail to reach their destination for many reasons. Therefore a retransmission mechanism is needed. The NVP protocol does not specifically provide for automatic retransmissions. NVP-II does define a token called "PLEASE ECHO". This token carries one word of data which the receiver returns to the sender as the data of an "ECHO REPLY"

token. Using these two tokens, an acknowledgement/retransmission routine was written. The acknowledge/retransmission routine maintains a retransmission table of control messages sent but not yet known to have arrived. Every NVP control message contains an "ECHO REQUEST" token whose data is a pointer to its entry in this table. Since most messages reach their destination safely, a copy of outstanding messages is not kept. The acknowledgement/ retransmission table contains sufficient information so that the message can be recreated and resent if necessary.

When an NVP control message arrives containing an "ECHO REPLY" token, the data are used to remove the corresponding message from the retransmission table. The table is constantly monitored. If a message has not been removed within two seconds, it is a candidate for retransmission. First the program checks to ascertain that it is reasonable to repeat the message. (There is no reason to resend a "WANT TO TALK" message if the caller has hung up.) A message will be retransmitted a maximun of ten times. If no acknowledgement is received, the call is terminated and the program is re-initialized.

The ST protocol provides for reliability checks. In ST every control message must be answered with either an ACK or a responding control message. ST control messages are also put on the retransmission table and removed when the corresponding ACK or answering message is received. Speech messages are never retransmitted. By the time the retransmission was accomplished, the time to play out the retransmitted speech would have passed.

In closing down a conference, a large number of disconnects may need to be sent. If one or more sites do not respond immediately or have crashed and cannot be reached it will be 20 seconds or so before all the retransmissions

are done. The PP cannot handle a new call until it has completed the shut down process. Therefore, if the user picks up the phone while shut down is in progress he will hear a fast busy signal and his dialing will be ignored. When the shutting down is complete, a dial tone will be sent to the phone and the user may place another call.

8. REAL-TIME STRUCTURE OF PVT SOFTWARE

8.1 Assembly Language Code

At first it was hoped that virtually all of the PVT software would be written in a higher level language so as to gain some of the advantages available with such a language. Code written in a higher level language is usually easier for someone else to read and understand. Because of the constraints imposed by the compilers of higher level languages the code they produce is often easier to debug than is assembly language. However, because of real-time constraints in handling some of the I/O and because of limitations in the compiler which was used for the project, a significant amount of code was written in the assembly language A-Natural [12].

The code which handles the I/O over the interfaces and the code which is time critical is written in assembly language. Every time the vocoder finishes reading or writing a parcel of data, an interrupt is generated. This parcel time for PCM and CVSD is 22.5 msec. For LPC, it is 20 msec. Both interrupt routines maintain a parcel count. The transmit parcel count is used as the timestamp for data messages. The number of buffers of speech coming from the vocoder is always a power of 2 (4 for PCM, 8 for ECVSD and 128 for LPC). The low order bits of the parcel count tell which buffer was just filled. The interrupt routine, which is called when a new parcel

arrives from the vocoder, increments the transmit timestamp signaling to the higher level routines that a parcel is available to be sent out. The interrupt routine which is called when a parcel is played out to the vocoder must locate the parcel and clear the header byte. This signals that there is no data in the parcel and if no new data have been put into this buffer when it is next read, the vocoder will play out a prestored parcel of silence. For each vocoder there is a buffer pointer table which contains pointers to the speech buffers. For LPC every register in the buffer pointer table points to a real buffer. However for ECVSD and PCM some of the registers in this table point to nonexistent memory. The reconstitution algorithm for PCM and ECVSD that was previously discussed requires that this interrupt routine update the buffer pointer table.

Two other types of routines are coded in assembly language. All the control messages and the headers of the speech messages are sumchecked. Since sumcheck routines worry about the carry bit they are easier and faster written in assembly language. The main routine which services the I/O interfaces is also written in assembly language.

8.2 Polling Loop

Except for the two interrupt routines, the remainder of the program operates on a polling system. The main "C" program is controlled by a polling loop that never ends. The I/O service routine sets flags for the "C" program. The "C" program checks for these flags and executes the appropriate routine when the flags are set. The "C" loop checks the transmit parcel count and if it has advanced, calls the routine which sends a speech data message (if appropriate). Each time around the polling loop the "C" program calls the I/O service routine so it may check the state of the I/O interfaces.

8.3 Buffer Availability to BC

Simplicity is enhanced by refusing to accept a second message from the BC until the current one is completely processed. Since processing an input message is likely to cause an answering message to be created, the PP does not accept a new message from the BC until its output buffer to the BC is free. The BC has five buffers used to handle messages from the local net. If the PP is slow taking messages from the BC, the BC routines buffer up the four most recent messages. The fifth buffer is kept free to handle priority messages. By taking advantage of this, the PP can process each message to completion. There is only room in the PPs DMA memory for one input buffer for messages from the BC. Reading a second message before completing action on the first could involve copying the first message into scratch memory.

8.4 Output Message Formation

In a PTP conversation the PP is normally sending messages to only one remote site. Space is reserved in DMA memory for the two normally used protocol headers (IP header and ST control header) and the ST data message header. These are set up when the connection is established and since only a small percentage of the bytes in a header change between messages, a new message can be sent very quickly. A fourth buffer area is reserved for the formation of answers to calls from other sites. In a conference, speech messages are sent to a fixed broadcast address and contain a bit map to denote which conferees are to receive copies. The headers change very little and the same strategy is employed.

8.5 Timing

Because the Protocol Processor software includes interrupt level routines and handles asynchronous DMAs, and because the main software

routines are interlocking polling loops, it has not been possible to ascertain precisely how much idle CPU time a PVT has. However, a PVT has been timed to see how much time is used sending and receiving speech data messages. For the PCM vocoder, time between parcel interrupts is 22.5 msec. Two DMA channels are used, one for sending messages to the Buffer Control and one to receive messages from the BC. The two DMAs are independent of each other.

To time PCM, a logic analyzer was used to measure durations. A point-to-point call was set up between two PVTs on the same LEXNET. One PVT was set to send continuously. The other PVT was prevented from sending but was receiving continuously. The sending PVT got a vocoder interrupt every 22.5 msec signaling that a parcel of data was available for sending. The time between the occurrence of this interrupt and the completion of sending the resulting speech message was measured. The data are shown in Table II. The time varied from a minimum of 3.4 to a maximum of 5.2 msec.

The PVT which was only receiving was measured to see how much time elapsed between getting the signal from the BC saying that a message had arrived and finishing reading in the message. The time fluctuated slightly around the value of 7.95 msec. Then both terminals were set to send continuously and therefore were also receiving continuously. The measurements were repeated. The time required to send a message increased by less than a msec while the time required to read in a message increased by 1.5 msec.

When there is a message to read and a message to send, the message to send is given priority. No message is read in until the output buffer is

free (see Section 8.3). This probably accounts for the greater increase in the maximum time required to read in a message when the terminal was also sending.

These times recorded in Table III show that a PVT can easily handle a PCM conversation even when both parties are talking at the same time. Not reading in a message until the output message queue is empty undoubtedly causes some extra delay. However, since the PVT is a dedicated machine and was able to meet the real-time requirements for the highest rate vocoder (i.e., PCM), there was no reason to complicate the program to reduce this delay.

TABLE III

MEASURED SEND AND RECEIVE TIMES

	Time to Send		Time to Receive	
	Min.	Max.	Min.	Max.
Sending Only	3.4 ms	5.2 ms	———	———
Receiving Only	———	———	7.95 ms	7.96 ms
Sending and Receiving	3.4	6.08	8.5	9.5

9. LANGUAGES AND SUPPORT FACILITIES

9.1 Choosing Languages

When this effort began, an investigation was made of the various means available to produce INTEL 8085 code. Writing most of the program in a

higher level language was highly desirable. However, the routines operating at interrupt level had to be written in assembly language because of time constraints and because this was the only way to run the DMA channels and handle I/O. The development machine available was a DEC PDP 11/70. Whitesmith, Ltd., markets a system which runs on a PDP 11/70, compiles code written in the higher level language "C", and produces INTEL 8085 code. The Whitesmith system also includes an assembly language called A-Natural. The Whitesmith compiler first compiles "C" code into A-Natural code. Then a Whitesmith assembler produces a relocatable binary file from the A-Natural file. Code written in "C" and in A-Natural can be linked together. The PP program has been written in "C" and A-Natural using the Whitesmith package.

9.2 Support Facilities

Unfortunately the Whitesmith package provides very little information to aid in debugging the code it generates. A *map* of core is produced which gives the core location of the beginning of each "C" routine, each label in the A-Natural code and each named variable. (Each label and variable contained in the A-Natural module must be specifically declared "PUBLIC" at the beginning of the module if its location is to be listed in the core map.) The ability to label a line in a "C" routine and have its location appear on the map would have been invaluable but the Whitesmith compiler deletes such a label unless it is used in a GOTO statement. A cross reference would also have been very helpful.

9.3 Downloading Facilities

During the development stage it is necessary to have some means of downloading the INTEL 8085 code produced on the PDP 11/70 into the RAM memory

of a PVT. Several so-called "debugger boxes" were built. A debugger box contains a small program which can receive characters over a dedicated asynchronous I/O line from the PDP and store them in the RAM memory of a PVT. The debugger box also provides limited debugging facilities. When the program is halted variables can be examined by keying in their locations and reading a display register. The debugger box also allows for trapping on reference to a given location. However, trapping when a location is changed is not available. Debugging a large program under these circumstances was painful.

9.4 PROMS

The program is limited to 32768 bytes. This limit is observed in order to keep the program contiguous and on one card. By doing this we can put the program on EPROMs and send a set to a site which does not have a downloading capability. Currently the program is 31885 bytes long. There are 4096 bytes of data memory which the DMA channels can read and write. This is why there is limited space available for buffering to and from the vocoder. There is ample space available for scratch memory.

10. MONITORING AND DEBUGGING AIDS

10.1 Diagnostic Record Keeping

Records are kept by the PP to aid in spotting errors in the system and to monitor the performance of various parts of the system. Counts are kept of all messages sent and received. The PP notes how many speech messages were discarded because their timestamp, when adjusted and used as an index, selected a pointer to nonexistent memory. The PP also keeps a distribution histogram of the variation of the incoming timestamps from the value

expected. The number of times the offset used to adjust the incoming timestamp needed to be recalculated is recorded. There are many reasons why an incoming protocol message might be discarded: bad checksum, illegal message, message too big or too small, etc. A speech message will be discarded if it is not from the conferee who the PP thinks "has the floor". There are thirty-four different reasons why a message might be rejected. Whenever a message is discarded, that fact is recorded and a count is maintained of the number of messages discarded for each reason. Whenever the system is not functioning properly, these records often give an indication of why. For instance, the delay distribution plus records kept by the GATEWAY can indicate trouble on the satellite channel. Conference speech messages thrown away because the sender does not "have the floor" may mean trouble with the Floor Controller.

10.2 "Talking to Yourself"

Possibly the most useful diagnostic tool we have is the ability to "Talk To Yourself". A user can pick up a PVT phone and dial his own number. As soon as a connection is set up, he can "Talk To Himself". Setting up this type of connection requires a reasonable amount of special code because one site is both caller and callee. The PP checks the address dialed to see if it is its own address. If so, it notes that fact and handles the protocol messages differently. If the call is dialed as a "long distance" call, the messages will be sent to the GATEWAY which will then send them back to the calling PVT. The GATEWAY can be instructed to send the messages over the satellite channel and back to the caller. This allows the system to be probed and evaluated from a single PVT. A user can check to see if his PVT

is operating correctly, if both vocoders are functioning correctly, if there is an operational GATEWAY on his net, etc. If the GATEWAY is instructed to route messages over the satellite, a single PVT can be used to check out the channel.

10.3 Echo Extension

When a user dials extension ONE (the ECHO extension) at a remote site, his speech will be echoed back to him without disturbing anyone at the remote site. One reason for adding this feature to the PVT code was to provide another means of bouncing speech off a remote site, either as a demonstration or to check the channel.

10.4 Providing Information to the User

The telephone instrument has only four tones it can play out to its user. This allows a rather limited amount of information to be passed back to the user. Often the PP has exact information on why a connection fails, why a remote phone does not join a conference, etc., but it cannot pass the information back to its user. The system would be much more "user friendly" if it had more ability to inform its user.

11. COMMENTS

11.1 Implementing Protocols in a PVT

The PVT has many advantages when implementing a system of packet voice speech. The biggest advantage is that it is a dedicated machine; no other user requirements had to be accommodated, and it was not necessary to interface the protocols to an existing operating system. General issues associated with interfacing protocols with the underlying operating system are discussed in [19]. This interface problem can be a great source of

performance degradation and delay. One delight of programming these protocols in the PP of a PVT was that being a dedicated machine, no other requirements had to be accommodated.

The PP was provided with DMA channels to facilitate the moving of data from the access area to the vocoder buffers and then directly to the vocoder. This allowed the message handling routines to move messages and data without using the CPU.

11.2 Use of Checksums

To ensure reliability, all protocol messages carry a checksum. The PP checksum routines have been carefully coded in assembly language in an attempt to gain efficiency. The speech data are not checksummed. Vocoder data with a small percentage of bits in error will normally produce better speech output than would be produced if the data were not used and a silence frame was played. Speech data cannot be retransmitted because it will not arrive in time to be played out in its proper place in the speech stream. Therefore, only the headers of speech data messages are checksummed. If a control message does not checksum correctly, it is simply discarded. This is appropriate for two reasons. First, the address of the sender could be where the error is and therefore is not a reliable address to which a complaint can be sent. Second, the sending PVT on not receiving an acknowledgement will resend the message.

11.3 RAM Memory

When the PVTs were built, the best available chips which would support a DMA channel were considerably larger in size than the standard INTEL memory chips. Because of space considerations on the boards, only 4K bytes of

DMA memory were provided in the PVTs. This limit on DMA memory had a considerable impact on the design of the PP software. The resulting program has worked well although a considerable amount of extra time and effort had to be expended to fit the I/O into the given space. There are now available chips which contain DMA memory and are the same size as the standard RAM memory chips. In new voice terminals now being designed and built, all on-board RAM memory will be DMA memory.

11.4 Implementation on Packet Radio Network

NVP-II and ST have also been implemented at SRI International in their Speech Interface Unit (SIU) on the Packet Radio Network (PRNET) in the San Francisco Bay Area. Their implementation duplicated the restrictions on NVP-II messages that are in use at Lincoln. Point-to-point calls have been placed between an SIU at SRI and a PVT at Lincoln. "MEET ME" style conferences using the SIU at SRI and PVTs at Lincoln and ISI have been demonstrated. These conferences used the Access Controller at Lincoln Laboratory. A description of the initial SRI work in radio nets is covered in [20] and [21].

11.5 Implementing NVP-II and ST

Both NVP-II and ST are implemented in the PP. NVP-II is a very free-form protocol. The user can combine tokens as he wishes. There is no definition of the tokens to be contained in any message. ST is a very structured protocol. Specific messages are totally specified for each protocol function.

Implementing the two protocols was instructive. At first NVP-II was implemented allowing the full freedom the protocol gives. There was trouble

immediately. If an initial message arrived and did not contain enough information for the PP to judge whether or not to accept the call - what then? Should the receiving terminal hold, waiting for another message with the remaining information? That message may never come. Therefore, some structure was imposed on NVP-II. The PVT implementation requires that various messages contain at least a certain set of tokens. Even then NVP-II was tricky to implement because there was so much freedom in the protocol.

ST was very easy and straightforward to implement. To handle the internet case, it developed that two words containing the address of the current sender had to be added to each command. Otherwise ST was implemented as originally specified.

11.6 Support Facilities

The support facilities used on this project were extremely primitive. A very large amount of debugging time was required. Timing dependent errors were particularly difficult to find. A HP64000 Logic Development System [22] has recently been purchased. Utilizing a system such as this to prepare the program, to debug it using the emulation capabilities, and finally to burn PROMs for the PVTs would have saved a great deal of time and frustration.

12. SUMMARY

The software implemented in the Protocol Processor of Packet Voice Terminal has been described. Point-to-point speech and conferencing capabilities in both local nets and over the wideband satellite have been discussed as well as the ability to place a call between a PVT and a telephone in the commercial telephone system. The PVT provides a compact and versatile speech interface to a packet network.

ACKNOWLEDGEMENTS

Many people have contributed to the work reported in this paper. Dan Cohen of ISI designed the NVP-II protocol. James Forgie designed the ST protocol. The vocoders used with the PVTs are the work of Joel Feldman, Marilyn Malpass, and Joseph Tierney.

Gerald O'Leary designed the PVT and supervised its development. He also wrote the software for the Buffer Control module and provided invaluable assistance when problems arose in debugging the Protocol Processor software.

William Kantrowitz developed the GATEWAY software and we worked together in checking out the system. James Forgie provided the algorithm which is used to correctly buffer incoming messages for playout. He was also very helpful in many other software design decisions. Clifford Weinstein has provided many ideas for this paper and has spent much time reviewing it.

REFERENCES

[1] C. J. Weinstein and J. W. Forgie, "Experience with Speech Communication in Packet Networks," to be published in IEEE Journal on Selected Areas in Communications, December 1983.

[2] J. W. Forgie, "Network Speech Implications of Packetized Speech," M.I.T. Lincoln Laboratory Annual Report to the Defense Communications Agency, 30 September 1976. DDC-AD-A045455/3

[3] C. J. Weinstein and H. M. Heggestad, "Multiplexing of Packet Speech on an Experimental Wideband Satellite Network," Proc. AIAA 9th Comm. Sat. Systems Conf., San Diego, CA, March 1982.

[4] H. M. Heggestad and C. J. Weinstein, "Experiments in Voice and Data Communications on a Wideband Satellite/Terrestrial Internetwork System," ICC'83 Conf. Rec., Boston, MA, June 1983.

[5] D. H. Johnson and G. C. O'Leary, "A Local Access Network for Packetized Digital Voice Communication," IEEE Trans. Comm., Vol. COM-29, pp. 679-688, May 1981.

[6] G. C. O'Leary, P. E. Blankenship, J. Tierney and J. A. Feldman, "A Modular Approach to Packet Voice Terminal Hardware Design," AFIPS Conference Proceedings (NCC'81), Vol. 50, May 1981.

[7] J. Tierney and M. L. Malpass, "Enhanced CVSD - An Embedded Speech Coder for 64-16 kbps," Proc. 1981 IEEE Int'l. Conf. on Acoust., Speech and Signal Processing, Atlanta, GA, 30 March-1 April 1981.

[8] J. A. Feldman and E. M. Hofstetter, "A Compact, Flexible LPC Vocoder Based on a Commercial Signal Processing Microcomputer," Electro'82, Session 22/5, Boston, MA, 25-27 May 1982.

[9] D. Cohen, "A Network Voice Protocol NVP-II," University of Southern California Information Sciences Institute (unpublished memorandum), April 1981.

[10] J. W. Forgie, "ST - A Proposed Internet Stream Protocol," (unpublished memorandum).

[11] J. B. Postel, "Internetwork Protocol Approaches," IEEE Trans. Comm., vol. COM-28, pp. 604-611, April 1980.

[12] C Computer Systems Interface Manual for 8080 Users, Copyright © 1979 by Whitesmith, LTD.

[13] D. Cohen, "Specifications for the Network Voice Protocol," University of Southern California Information Sciences Institute, Rpt. ISI/RR-75-39, March 1976.

[14] J. W. Forgie, "Voice Conferencing in Packet Networks," ICC'80 Conf. Rec., pp. 21.3.1-21.3.4, June 1980.

[15] M.I.T. Lincoln Laboratory, Semiannual Technical Summary to the Defense Advanced Research Projects Agency on Packet Speech Systems Technology, 30 September 1982. DTIC-AD-A126880

[16] M.I.T. Lincoln Laboratory, Semiannual Technical Summary to the Defense Advanced Research Project Agency on Packet Speech Systems Technology, 31 March 1982. DTIC-AD-A120433

[17] I. H. Merritt, "Providing Telephone Live Access to a Packet Voice Network," University of Southern California Information Sciences Institute, Rpt. ISI/RR-83-107, February 1983.

[18] T. Bially, B. Gold and S. Seneff, "A Technique for Adaptive Voice Flow Control in Integrated Packet Networks," IEEE Trans. Comm., Vol. COM-28, pp. 325-333, March 1980.

[19] D. Clark, "Modularity and Efficiency in Protocol Implementation," Massachusetts Institute for Computer Systems and Communication Group, RFC: 817 (unpublished memorandum), July 1982.

[20] P. Spilling and E. Craighill, "Digital Voice Communication in the Packet Radio Network," ICC'80 Conf. Rec., pp. 21.4.1-21.4.7, June 1980.

[21] N. Schacham, E. J. Craighill and A. A. Poggio, "Speech Transport in Packet Radio Networks," submitted to IEEE Journal on Selected Areas in Communications.

[22] M. Davis, J. A. Schmarrer and R. G. Weekliff, Jr., "Extensive Logic Development and Support Capability in One Convenient System," Hewlett-Packard Journal, March 1983.

APPENDIX I

Acronyms and Abbreviations

AC	-	the Conference Access Controller
ARPA	-	Advanced Research Projects Agency (also referred to as DARPA)
ARPANET	-	the ARPA network
BC	-	the Buffer Control Module
CPU	-	Central Processor Unit
CVSD	-	Continuously-Variable Slope Delta Modulation
DARPA	-	Defense Advanced Research Projects Agency (also referred to as ARPA)
DMA	-	Direct Memory Access
DoD	-	Department of Defense
ECVSD	-	Embedded CVSD
EPROM	-	Eraseable Programmable Read Only Memory
GATEWAY	-	the module which connects two or more networks in an internet
INTEL 8085	-	microprocessor developed by INTEL Corporation
IP	-	Internet datagram Protocol; a DoD standard datagram protocol
LCI	-	Lexnet Concentrator Interface
LEXNET	-	Lincoln Experimental packet voice network developed by Lincoln Laboratory
LPC	-	Linear Predictive Coding
NCP	-	Network Control Protocol; the basic host-host data protocol used in the ARPANET
NVCP	-	Network Voice Conferencing Protocol
NVP-II	-	Network Voice Protocol II
PCM	-	Pulse Code Modulation
PP	-	Protocol Processor
PROM	-	Programmable Read Only Memory
PTP	-	Point-to-Point
PVT	-	Packet Voice Terminal
RAM	-	Random Access Memory
ROM	-	Read Only Memory
SATNET	-	the Atlantic Packet Satellite Network
ST	-	Stream protocol; an internet transport protocol for speech and other real-time traffic
STNI	-	Switched Telephone Network Interface developed by ISI
TCP	-	Transmission Control Protocol; DoD standard reliable transmission protocol
USART	-	Universal Synchronous/Asynchronous Receiver Transmitter
VCOP	-	Voice-controlled Operator
VP	-	Voice Protocol
VPSS	-	Vocoder Preference Selection Switch
WB SATNET	-	the Wideband Packet Satellite Network
WTT	-	Want-to-talk; Button on conference control box used in SATNET

APPENDIX II

Sizes and Functions of Protocol Processor Software Modules

A. C-Language Modules

There are 114 C routines of various sizes which are contained in 6 separate modules with a total length of 28,464 bytes.

C-Module-1: CONTROL. This is the general control module for the system. It contains 23 routines and is 5822 bytes long. It contains the main control loop. Once initialization is completed this loop runs forever and monitors the condition of the phone and the activity on the various I/O interfaces. Each pass through the main loop polls a series of status registers about jobs that are pending. Control is passed to the various modules as appropriate. At the end of the main C loop the main A-Natural routine is called so that it can poll and service the hardware I/O.

CONTROL contains, besides the main loop, four initialization routines, a routine to handle output to buffer control and one to handle input from buffer control, five routines involved in shutting down a connection or a conference, a retransmission control routine, two routines to handle turning vocoders on and off, three routines to handle errors in incoming messages, a routine to echo back speech input, and four service routines.

C-Module-2: IPIN. Handles incoming IP messages. It contains twenty routines and is 5203 bytes long. It consists of two control routines, seventeen token processing routines and a final analysis routine. The control routine checks and processes each incoming IP message, invokes the appropriate token processing routine to handle each token and finally invokes an analysis routine which determines the correct response to this message.

C-Module-3: IPOUT. Handles creating and sending IP messages. It contains twenty-seven routines and is 4496 bytes long. There are seven control routines which create a particular type of message by invoking one or more of the sixteen routines which add particular tokens to the message being formed. These seven routines create messages such as: initial IP connect message, accept of another IP connect, request to the Access Controller (AC) to join a conference, request for information from AC about participants in a conference, notification to the AC on departure from a conference, invitation to another PVT to join a conference, and refusal of a request to join. There is a service routine which determines the correct destination for each outgoing message (local terminal or GATEWAY) and two routines which control the actual sending of IP messages.

C-Module-4: STIN. Handles incoming ST messages. It contains nineteen routines and is 5655 bytes long. An ST message either contains one token of ST control information or it contains speech data. (ST messages which contain speech data may also contain NVP-II protocol tokens). The module

contains two general control routines, two routines which control readin of speech data, an ST control token-processing routine, thirteen token-handling routines, and a general checking routine. Since an ST control message contains only a single control token, each control processing routine determines what answering message (if any) should be sent.

C-Module-5: STOUT. Handles creating and sending ST messages and sending speech data. It contains eighteen routines and is 4058 bytes long. There is a general control routine for sending speech data which uses one of three vocoder-dependent subroutines to correctly send each data message. There are two control routines which create various ST protocol messages, a record keeping routine, and two service routines.

C-Module-6: PHONE. Handles the analysis of input characters coming from the PVT phone via the USART. It contains seven routines and is 3230 bytes long. It consists of a general control routine, a switching routine, two routines which handle the special characters * and #, a routine to handle an incoming digit, and a routine to handle VCOP. When a PVT is acting as the front end of VCOP, it receives a special character (an ASCII "v") over the USART from the attached PDP-11 when it tries to "ring the phone". This notifies the PVT that the VCOP routine should be used.

C-Module-6': STNI. When a PVT is used as an STNI, the Lincoln phone-handling module is replaced by a module written by Ian Merritt of ISI [20]. This module handles the features unique to an STNI. The module is approximately 3030 bytes long (about 200 bytes shorter than the Lincoln PHONE module).

B. A-Natural Modules

There are seventeen routines written in A-Natural which are contained in two modules and are 2855 bytes long. There are three routines to initialize/reinitialize various I/O interfaces, a routine to set up speech buffers, routines to start interrupts, and to start and stop the speech DMAs. Two routines run at interrupt level and handle the frame sync interrupts from the vocoder, three routines handle output to the phone (two for VCOP and one for general use), two routines handle error conditions, two routines compute checksums (word checksum and byte checksum). The final routine is the general control loop which polls the status of all the I/O interfaces and causes appropriate action to be taken.

C. Library Routines

The Whitesmith system, which compiles and links these modules, adds sixteen library routines which are a total of 566 bytes long. These library routines are used by the compiler to implement functions such as multiplication and division which are allowed in the "C" language but for which there are no corresponding machine instructions for the INTEL 8085.

D. Summary

C-Routines

IPIN	20	Routines	5203 bytes
IPOUT	27		4496
STIN	19		5655
STOUT	18		4058
PHONE	7		3230
CONTROL	23		5822
	114	Routines	28464 bytes

A-Natural Routines		
I/O Routines	15	2774
Checksum Routines	2	81
	17	2855
Library Routines	16	566
Total	147 Routines	31885 bytes

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-83-054	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Protocol Software for a Packet Voice Terminal		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER Technical Report 663
7. AUTHOR*(s)* Constance K. McElwain		8. CONTRACT OR GRANT NUMBER*(s)* F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element Nos. 61101E, 62708E Project Nos. 3D10 and 3T10 DARPA Order 3673
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 16 November 1983 13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified 15*a*. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ARPANET	network voice protocol	protocol
internetwork	packet	stream protocol
LEXNET	packet speech	vocoder
network speech	packet voice terminal	voice conferencing

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A Packet Voice Terminal (PVT) has been developed at Lincoln Laboratory to provide voice access to an experimental wideband internetwork packet system. The PVT employs a modular, microprocessor-based structure to provide voice processing, packet voice protocol, and network interface functions. The packet voice protocols are implemented in software in the Protocol Processor (PP) module, which is the primary controlling module of the PVT and which handles interfaces to a voice processor, a network interface processor, and a user instrument. This report describes the software implemented in the Protocol Processor. The implementation of the Network Voice Protocol (NVP-II) and the Stream (ST) protocol are described. Call set-up functions for both point-to-point calls and conferencing, and the methods used for packetization and reconstitution of speech, are described. Problems encountered and solutions which have been implemented are discussed.