CULLER/HARRISON, INC.
150-A Aero Camino
Goleta, California 93017
(805) 968-1064

December 1975
Revised:
April 1978

SPEECH SIGNAL PROCESSING RESEARCH

Final Technical Report

March 1973 - December 1976

PRINCIPAL INVESTIGATOR: Dr. Glen J. Culler

PROJECT SCIENTISTS: Dr. Michael McCammon
Dr. James F. McGill
Mr. Ray E. Bjorkman
Mr. Benjamin K. Lum
Mr. Dale E. Taylor
Mr. Jan M. Vanderford

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2359 Contract No. DAHC15 73 C 0252

CONTENTS

		Page
SUMMARY		iii
1.	SIGNAL SYSTEM HARDWARE	1
	1.1 MACROPROCESSOR	1
	1.2 ARRAY PROCESSOR	2
	1.3 INPUT/OUTPUT PROCESSOR	3
	1.4 MULTICHANNEL ANALOG SUBSYSTEM	3
2.	SIGNAL SYSTEM SOFTWARE	4
	2.1 SYSTEM OPERATION	4
	2.2 MACROPROCESSOR	4
	2.3 ARRAY PROCESSOR	5
	2.4 INPUT/OUTPUT PROCESSOR	5
3.	SIGNAL INTERACTIVE MATH SYSTEM	9
	3.1 KEYBOARD INPUT AND USER PROGRAM EXECUTION	9
	3.2 OPERATOR EXECUTION	10
	3.3 MATHEMATIC VARIABLES	10
4.	ARPA NETWORK	13
	4.1 RELIABLE TRANSMISSION PROGRAM (RTP)	13
	4.2 NETWORK CONTROL PROGRAM (NCP)	16
	4.3 NETWORK VOICE PROTOCOL	17
5.	REAL-TIME SPEECH COMPRESSION ON THE ARPA NETWORK	18
	5.1 THE LINEAR PREDICTIVE CODING SPEECH COMPRESSION	18
	5.2 NETWORK VOICE COMMUNICATION	21
6.	LABORATORY EXPERIMENTS IN PHYSICAL ACOUSTICS	23
	6.1 USE OF THE SIGNAL SYSTEM IN AN ACOUSTIC LABORATORY	23
	6.2 THE OPTIMAL FILTER AS AN AREA FUNCTION PREDICTOR	24
7.	NETWORK VOICE CONFERENCE RESEARCH	26
	7.1 MODIFICATIONS TO THE SIGNAL SYSTEM FOR NETWORK VOICE CONFERENCING WORK	26
	7.2 VARIABLE FRAME RATE CODING OF LPC PARAMETERS	28
APPENDICIES: (Not included in this version)		

SUMMARY

This report covers the work performed by Culler/Harrison, Inc. (CHI) under contract No. DAHC15 73 C 0252 for the period March 1, 1973 to December 1, 1976.

During this period, Culler/Harrison, Inc. has developed a powerful, versatile signal processing capability for ARPA and utilized this capability to perform significant and diverse signal processing experiments. The contract called for construction, operation and maintenance of the CHI SIGNAL SYSTEM, a combination of equipment and software providing a unique marriage of computing power and on-line interactive control of that power. The system has proved its flexibility and reliability on a broad range of applications encompassing speech compression, on-line control of laboratory experiments, text editing and controlled thermonuclear reactor simulation. Chapters 1 and 2 of this report cover the hardware and software of the SIGNAL SYSTEM.

An on-line Interactive Math System was developed and implemented on the CHI SIGNAL SYSTEM under this contract. As a research tool, the SIGNAL Interactive Math System offers researchers the ability to develop sophisticated signal processing processes, to try out new approaches and receive numerical and graphical feedback on-line interactively, and all with button push turnaround time. The Interactive Math System has been retrofitted to an identical CHI SIGNAL SYSTEM which is playing an integral part in the ARPA Research Center, or ARC, located at Moffett Field, California. The Interactive Math System, described in Chapter 3 of this report, offers improvements over similar on-line interactive mathematical systems implemented by Dr. Culler and other CHI personnel on other equipment: the RW-400 at TRW, the IBM 360/75 at UCSB and at UCLA, and an SEL 810A at CHI.

As part of this contract, the CHI SIGNAL SYSTEM was connected to the ARPANET as a Very Distant Host. Involved in this task was the design and development of a modem interface, as well as a Network Control Program and a Reliable Transmission Program. The network connection was used for real time speech compression experiments, out of which came the Network Voice Protocol. These programs are described in Chapter 4. An obvious next step in the development of the connection to the ARPANET is for CHI to support the TELNET Protocol with graphics capabilities and the File Transfer Protocol. With this capability remote researchers could apply the power and convenience of the Interactive Math System to their problems.

Also covered by this contract was the use of the CHI SIGNAL SYSTEM as a research tool in experiments in physical acoustics and real time speech compression over the ARPA network. Investigations into the use of the linear predictive coding optimal filter as an area function predictor when applied to cavities with known area functions were performed and are reported in Chapter 6. The speech compression system implemented at CHI is based on the Markel and Gray algorithm, which uses the autocorrelation method of linear prediction to develop a set of parameters which describe the vocal track of the speaker and measure of pitch, amplitude and voicing of the speech. In this effort the CHI system achieved analysis and synthesis in 60% of real time, (almost twice real-time). A pilot test of the intelligibility of several speech compression systems conducted for ARPA by Stanford Research Institute gave the CHI system the highest rating of those studied. This system has been used since November 1974 for experiments in low bandwidth digital voice communication over the ARPANET. We have used these experiments to develop a protocol for controlling the transmission of speech parameters on the network including procedures for variable rate transmission. These speech compression experiments are covered in Chapter 5. Extensions to these experiments to incorporate improved transmission algorithms and network voice conferencing are discussed in Chapter 7.

In summary, we believe that the work performed by CHI as described in this report has made a significant contribution to ARPA's general capability in signal processing and specific contributions in the areas of speech compression and network voice communication.

CHAPTER 1. SIGNAL SYSTEM HARDWARE

The CHI SIGNAL SYSTEM is a high-performance signal processing system comprised of two processors and associated peripheral devices. The main processor is the Macroprocessor, the MP-32A, which has 32K (K=1024) words of 36-bit MOS memory. The peripheral devices, three 2314-type disk drives, four consoles, each with a keyboard and a storage CRT display, a 16-bit parallel HOST interface, and an analog subsystem, are connected to the MP* The second processor is the Array Processor, the AP-90, which has 4K of 36-bit MOS memory. The AP can perform floating point, fixed point, and complex fixed point calculations on internal data or data supplied by the MP. The system architecture allows for independent concurrent processing by the two processors as well as tightly coupled processing in which the AP operates as an extended arithmetic unit with respect to the MP. Typically, the MP handles input/output, task management, and bookkeeping while the AP performs the arithmetic operations.

1.1 MACROPROCESSOR

Several manuals exist which fully document the hardware of the MP-32A. The main document is the MP-32A Macroprocessor Reference Manual (TMB2), included as Appendix 1. This manual describes the physical and functional characteristics of the macroprocessor, first as a single component, then as a part of the SIGNAL SYSTEM. Installation, operation, theory of operation and maintenance are also covered in the reference manual. The MP-32A Macroprocessor Maintenance Manual (TMB4), Reference 1, contains the complete set of assembly drawings and logic diagrams. The MP-32A Macroprocessor Wire List -- SN/2, Reference 2, contains the wire list for every wire wrap connection in the MP. The wire list is comprised of both a logic sort and a connector sort. The logic sort is ordered by logical signal name and lists the location, load or drive, and logic drawing number for each wire wrap connection in the MP. The connector sort has the same information but is ordered by location instead of logical signal name. The disk drives are documented by the technical manual, Reference 3, provided by the manufacturer, Century Data Systems (now Calcomp), and the storage display unit is documented by the instruction manual, Reference 4, provided by Tektronix. The maintenance manual, wire list, disk drive technical manual, and display unit manual, because of their bulk, are not included

*As shown in Figure 1, one of the three 2314-type disk drives and three of the four consoles are CHI-owned equipment and are not part of the ARPA purchase.

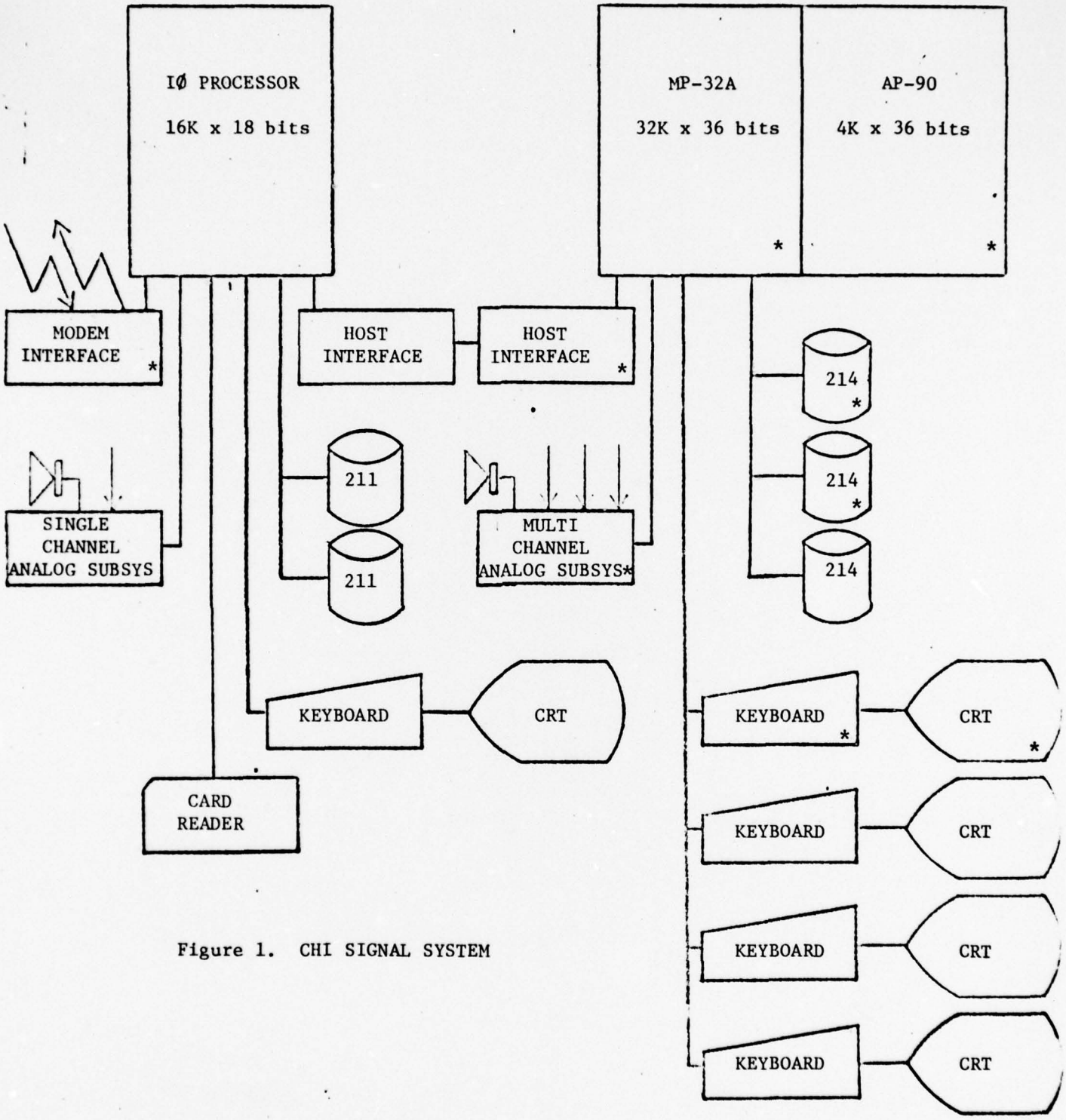

Figure 1. CHI SIGNAL SYSTEM

*starred components were purchased by ARPA under this contract

in this report. They are available for MP maintenance at the installation site. Programming manuals for the MP are described in Chapter 2 of this report.

1.2 ARRAY PROCESSOR

A similar set of manuals exist which fully document the hardware of the AP-90. The main document is the AP-90 Array Processor Reference Manual (TMB8), included as Appendix 2. This manual provides a general description of the unit, as well as physical and functional characteristics, operation, input/output ports, programming and maintenance. The AP-90 Array Processor Maintenance Manual (TMB5), Reference 5, contains the complete set of logic drawings for the AP. The AP-90 Array Processor Wire List -- SN/2, Reference 6, contains the wire list for every wire wrap connection in the AP. Again, the wire list is comprised of both logical signal name sort and connector, or location, sort. The wire list and the maintenance manual are not included in this report. They are available at the installation site for AP maintenance. The programming manual for the AP is described in Chapter 2 of this report.

1.3 IØ PROCESSOR AND MODEM

Connected to the MP-32A via the 16-bit parallel Host interface is the IØ Processor. While not part of the SIGNAL SYSTEM itself, the machine serves as an IØ processor to the system. The IØ Processor is the production prototype for the MP-32A. As such, it is often referred to as SN/0. While the instruction set for SN/0 is essentially identical to that of the MP-32A, there are differences between the two machines. Most notable is the difference in main memory. In the MP-32A, memory is 32K words of 36-bit dynamic MOS memory. Access and cycle time is 667 nsec, but when used sequentially, the four-way interleaving results in a 167 nsec access time for a 36-bit word. In SN/0, memory is only 16K words of 18-bit core memory. Access and cycle times are 1 μsec, with no interleaving. To read 36 bits in SN/0 requires 2 μsec, while in the MP-32A a read of 36 bits requires as little as 167 nsec.

The IØ Processor has a set of peripherals attached to it: two 2311-type disk drives, a console with keyboard and storage CRT display, a 16-bit parallel Host interface, a card reader, a modem, and a two-channel (one input/one output) analog subsystem. Only the modem and the analog subsystem have relevance to the SIGNAL SYSTEM. The modem is used to connect the system to the ARPANET as

a Very Distant Host through a 303 Data Set. The analog subsystem will be used for analog input and output until the multichannel analog subsystem connected to the MP-32A is operational. The modem is described in detail in the reference manual, the Very Distant Host (VDH) Interface to the MP-32 (TMB7), included as Appendix 3. The analog subsystem was described in Chapter 3 of the May 1975 Quarterly Technical Report, Reference 7. The analog subsystem has not changed since that documentation, however, one error is worth correcting. The D/A converter is an Analog Devices MDAC-L, and not an Analogic MP1412 as was stated in Reference 7.

1.4 MULTICHANNEL ANALOG SUBSYSTEM

A multichannel analog subsystem has been designed, assembled and installed on the MP-32A macroprocessor. This subsystem, as described in the May 1975 Quarterly Technical Report, and further defined in the CHI technical note CHI-TN-74-38, entitled Specification of Analog Input/Output Capability for Serial 2 MP, included as Appendix 4, is in the last stages of checkout. Since the April Quarterly Report the following changes have been made in the analog subsystem:

1. Addition of a sample/hold deglitching amplifier to the D/A module.
2. Construction of a filter module containing two 4.9KHz low pass filters and two 3.2KHz low pass filters.
3. Replacement of the modular 5 volt/1 amp per card analog power supply with a single 5 volt/12 amp supply.
4. Modification to the A/D module input preamplifier which is still in progress.

CHAPTER 2. SIGNAL SYSTEM SOFTWARE

2.1 SYSTEM OPERATION

The SIGNAL SYSTEM is a combination of software and hardware specifically designed to provide an interactive signal processing facility. Key software features include independent controllers for each I/Ø unit; keyboard, display, analog, timer, disk, and host interface. System facilities are organized as modular coordinated subsystems; operating, library, editing, mathematical, utility and diagnostic. Multi-level programming languages permit the user to focus his efforts to optimize critical sections of his task. The User Manual (TMA2), included as Appendix 5, provides the user with a description of the tools made available to him through the software of the SIGNAL SYSTEM. Included in this manual are descriptions of the library subsystem and document operations, text construction and editing, and the Signal Interactive Math System. Researchers using the SIGNAL SYSTEM would have to refer only to this manual. Programmers would have to refer to programming manuals described in the next sections of this chapter. System utilities and operators are described in the System Operation Manual, included as Appendix 6. Also covered in this manual are the system diagnostics. The SIGNAL SYSTEM acceptance tests, described in the Operation Manual, also serve as diagnostics.

2.2 MACROPROCESSOR

Programming of the MP-32A occurs in either micro-language or macro-language. Both languages are supported by the symbolic system assembler, which will build executable object code from text documents. The system assembler and the macro-language are described in the MP-32A Macro-Programming Manual (TMA3), included as Appendix 7. The micro-language is covered in the MP-32A Micro-Programming Manual (TMA4), included as Appendix 8.

2.3 ARRAY PROCESSOR

The AP-90 is programmed in the micro-language described in detail in the AP Programming Manual (TMA8), included as Appendix 9. This manual presents the AP instruction set, as well as the AP assembler. This assembler is different from the system assembler in that symbolic relocateable code is generated. The AP linker then builds the final AP executable code. The relocateable assembler and linker are described in Sections 4 and 5 of the AP Programming Manual.

2.4 INTPUT/OUTPUT PROCESSOR

Programming the IØ Processor (SN/0) is basically the same as the MP-32A (SN/2). Macro-language instructions are functionally identical but execution times are longer, due to the slower memory. Micro-instructions are identical except where hardware changes are reflected in the instruction set. Memory timing and unique input/output devices are the primary differences. The system assembler of the MP-32A builds the executeable code for the I/Ø Processor.

The Operating System in SN/0 has three I/Ø devices to coordinate: the Host interface with SN/2, the modem interface to the network and the analog interface which includes both D/A and A/D converters. Both the analog and modem devices are closely interrelated to the Host interface because they receive their output data from the Host and they send their input data to the Host. The modem and analog devices are quite similar in that there are rather stringent timing requirements imposed by both devices, both for input and output. The Host interface does not impose such strict timing requirements, and it can move data much more rapidly than either of the other two.

A data flow scenario would look like this: analog input data comes in thru the A/D converter on SN/0, over through the Host interface to SN/2 where it is processed and a network output message is generated which comes back to SN/0 through the Host interface and out to the network via the modem interface; conversely, the input message comes in through the modem interface from the network over to SN/2 through the Host interface where it is analyzed and the analog output data generated is sent through the Host interface back to SN/0 where it is output through the D/A converter.

For Host-to-Host transfers (SN/0 to SN/2 and vice versa), the sender always initiates the transfer by sending to the receiver a command code which is unique

for the type of transfer requested. When the receiver is ready to accept data a response is returned and the transfer begins.

SN/0 is restricted in these Host transfers by the required interrupt response times of the modem and analog devices. Thus the Host transfers are designed to run "in parallel" with the other transfers. By "in parallel" here is meant that a block transfer from host to host may be interrupted at any point by an analog or modem I/O request.

The analog hardware which includes both the D/A and the A/D converters is capable of running at sample rates up to 50 KHz (one sample every 20 microseconds); however, for the network speech experiments, sample speeds of 150 microseconds are used. The implications of this sampling rate to the SN/0 operating system are that analog interrupts must be serviced within 150 microseconds including both an analog output request and an input request; in addition, modem requests must be handled and occasional host-to-host transfers must be performed.

With all these operations occuring simultaneously, it is necessary to have the capability to delay some transfers until more time is available for them. The analog I/Ø is non-stop, whereas the modem I/Ø comes in spurts, and the host transfers can occur at the time deemed appropriate by the operating system. With this in mind, the analog controller has been allocated five buffers of 128 words each, which rotate in use to allow sufficient backup of buffers to ensure no breaks in the input or output of data. At sampling speeds of 150 microseconds per point, each buffer contains 19.2 milliseconds worth of analog data.

Each analog buffer originally contains data for the D/A converter. The analog output interrupt occurs at the sample time and the input interrupt occurs a few microseconds later, so upon the output interrupt a word is offered to the D/A, and later the word from the A/D is read into the place where the output word came from. So the buffer which began as a full output buffer ends up as a full input buffer, ready to send to SN/2.

Every time a buffer is filled it is sent to SN/2 as soon as possible. When SN/2 receives a full input buffer it immediately knows to send the next output buffer to SN/0. When this new output data is sent, it is placed in the next available output buffer of the five. Thus there is an output pointer (to point at the buffer currently being output thru the D/A), a "fill" pointer (to indicate the next buffer to fill up with data from SN/2), and a "send"

pointer (to indicate the next buffer to send to SN/2). The SN/0 cycler uses these pointers to tell when to send data to SN/2: when the "send" pointer is different from the output pointer there is a buffer waiting to send.

The modem interface controller is discussed in greater detail elsewhere in this report, so here only a brief summary appears. During a packet transfer, the interface will interrupt once every 160 microseconds, either requesting a byte for output or indicating that a new input byte has entered or both. This interrupt must be honored within 160 microseconds or the transfer will be incorrect and will have to be resent. Typically, the interrupts will occur in bursts of up to 128 since that is the packet size and typically only one packet is in transmission at a time.

Roughly speaking, we send seven messages per second plus the IMP/Host protocol, which makes about 16 packets of approximately 20 16-bit words each per second transmitted and a like number received. At 160 microseconds per byte, this takes approximately 6.4 milliseconds to send or receive an average packet, or roughly 100 milliseconds out of each second are spent sending network data and another 100 milliseconds receiving data from the net. Since the interface is full duplex, the transmission may be overlapped with the reception. Resends are sometimes necessary so the figures here are slightly high, and the indication is that an upper limit of roughly 20% of the time is spent sending and/or receiving network data.

Although most of the I/Ø on SN/0 generates interrupts to ensure that attention is given them, there remain some tasks which are not entirely interrupt driven. For the modem I/Ø, once a packet has begun being received or sent, the interface generates the interrupts which cause the system to take proper action to complete the transfer. The question remains, however, of how the transfer is initially begun in the case of output and, once the bytes of an input packet have been received, how processing is continued. The details of the processing are discussed in the sections on the RTP and NCP, and here the scheduling of that processing is discussed. Similarly, for the analog I/Ø, the analog interface generates the interrupts which notify the system that another single word A/D or D/A transfer is required, but the scheduling of the buffer processing has not been discussed. And finally, there is the scheduling of the host transfers; both analog and network types are scheduled by the system process scheduler.

Since the modem input packets arrive sporadically and occasionally in bursts of several in a row, the scheduler gives highest priority to the processing of incoming packets from the modem. Whenever an incoming packet is processed, any output packets ready to send are sent, so the processing of modem input also causes modem output scheduling. There is also a system timer running which the scheduler uses to schedule the sending of the necessary periodical "HELLØ" packets; the sending of "I HEARD YOU" packets is caused by the processing of incoming "HELLØ's."

Only if there are no outstanding input packets from the modem to process does the scheduler query as to whether an analog input buffer requires processing. If there is a full analog input buffer, the scheduler immediately sends it to SN/2 for processing. SN/2 responds to the receipt of an analog input buffer by immediately sending a full analog output buffer back to SN/0. The output from the output buffers is self-scheduling since at interrupt time the next output word is used, hence occasionally a new output buffer is fetched and used by the analog interrupt handler.

The scheduler has two more tasks which must be accomplished periodically. The first of these is to schedule resends of output packets to the modem which have not been acknowledged within a certain allowed time frame. This time frame is large compared to the time to fill an analog buffer, for instance (it is on the order of 120 milliseconds) so the priority of resending is low. The other task which the scheduler schedules is the processing of completed input messages from the IMP. These messages arrive, on the average, every 140 milliseconds so again the priority of processing them can be quite low compared to the analog buffers or modem input packets.

CHAPTER 3. SIGNAL INTERACTIVE MATH SYSTEM

The CHI Interactive Math System runs as a background job on the CHI MP-32A and AP-90 computer system. The reader may refer to the enclosed section of the System Operations Manual (TMA9), entitled "Operating System Philosophy" for a description of a background job. The background job which comprises the math system is submitted by the math interactive track, which is one of the 13 tracks which are assigned to the user upon his signing on to the SIGNAL system. The only condition which can prevent the user's being allowed to enter the math system is if he has not declared any data library as his current data library. Once the job has been submitted for the user, it is swapped in and out as deemed necessary and appropriate by the operating system to allow time sharing with other users running simultaneously. This swapping is invisable to the user for the most part; only occasionally and with several other users running at the same time can it be noticed.

3.1 KEYBOARD INPUT AND USER PROGRAM EXECUTION

As mentioned in the enclosed note, "Operating System Philosophy," keyboard interaction is possible from a background job. There are several system-resident programs available for fetching keys from the keyboard along with higher level routines in the math system which use these resident system routines to fetch keys and, using the keys fetched, build special keyboard operands such as decimal integers, floating-point values, and variable names. The math job enters wait state when there is no keyboard input available to it, and is automatically resumed by the job scheduler when a keyboard input statement has been completed.

It is possible for the user to build user programs which will execute within the math system. These programs are simply sequences of keys which are put together,using the SIGNAL editing facility, and stored in a library as a named text document. The entire sequence of keys may then be invoked from the math system be merely pushing the PGM key followed by the name of that text document. What happens internally within the math system under this circumstance is that the named text document is fetched from the library and converted from its text format into a keyboard execution list format on the user's math interactive track. The station keyboard input pointer is then set

to this new keyboard execution list and normal control is continued. All the keyboard input programs use this keyboard input pointer for the source address of the keys so the same keyboard input routines are used whether the station is in math program mode or normal interactive mode.

3.2 OPERATOR EXECUTION

The basic SIGNAL operating system is statement oriented. This merely means that rather than processing each key as it is entered from the keyboard, the keys are saved until the statement terminator (carriage return) is entered. Once the completed statement is entered and placed on the appropriate interactive track, the keys are fetched one at a time until all the keys in the statement have been processed. In the case of the math system, there is one basic control program which processes the statements as they enter, either from the keyboard or from a user program. This control program really only processes the operator keys from a statement; the individual math system operator programs process whatever argument keys are necessary as parameters for the program to execute properly.

The math system control program, once it has a statement to process, fetches the next input key and tests it to be an operator key. If it is not an operator key it is ignored unless it is a carriage return, in which case the control program fetches the next input statement. If the key is an operator key, the address of the correct operator program to use is fetched from the math system operator table (indexed by the key-code) and executed by the control program. All operator programs terminate by passing control to a specific entry point in the control program.

3.3 MATHEMATICAL VARIABLES

Variables within the math system are named and may be integer or floating point, real or complex, scalars or arrays, in memory or on disk. For each defined variable the system keeps a multiple-word descriptor whose length varies with the type of variable it refers to. The system saves these descriptors along with the data to which they refer, whenever the user leaves the math system, and restores them whenever he returns. This is possible because the first time the user goes to the math system after signing on fresh, nine tracks

of disk space are allocated to him for saving his math system variables. The track number of this space is kept on his math interactive track so that when he returns to the math system after leaving it his memory variables may be restored to their previous values.

Most math system operators perform a transformation on one or more math variables. In order that all the operator programs not have to do their own argument decoding, there is a system subroutine which all the operator programs use to do this decoding for them. The result of this decoding is a code word describing the type of the variable and three words which describe the variable starting address, the increment which is added to the address of one element of the variable to get to the next, and the limit address of the variable. Typically, a math system program will call this argument decoding subroutine which will gather all pertinent keyboard input describing the argument variable and pass back the type code of the variable and the three-word internal descriptor. The calling program will then check that the variable type is one of the allowed ones and proceed to do its defined transformation on each element of the variable, sequencing through the vector until it has reached the last element and then terminate.

This internal descriptor scheme allows for performing transformations on subsets of larger arrays by entering keys which describe the desired subset. The argument decoding subroutine gathers the keys and derives the appropriate internal descriptor from this keyboard specification. The limitation is that the described subset be a one-dimensional list of one or more elements, so that one increment can be used and only one pass through the data need be made. The internal descriptors apply for output as well as input.

There are two other higher level argument fetching programs used by the math system mathematical transformation operators; one for the unary operators and one for the binary operators. The function of these routines is to resolve the type of call to the operator into a standard form which the operator can use without resorting to special logic for the different cases. The need for such programs arises due to the amount of flexibility allowed in using these mathematical operators: there are four different styles of calling for a unary operation for instance (see the enclosed User Manual, TMA2, Section 4.5). So these intermediate argument fetching programs set up one or two source argument descriptors and a destination argument descriptor and pass these to the operator program which always transforms the source argument(s), placing the result in the destination argument.

Most operations are designed to work only on data which is in memory. For disk variables, typically the data will be moved into memory from disk using the "MOVE" operator, transformed in some way, and moved back onto disk again using the "MOVE" operator. Due to memory size limitations, typically one track of disk data is done at a time in a loop which sequences through and repeats the operation until each track of the disk data set has been transformed. There are, of course, exceptions to this rule; that is, there are some operations which will work directly on a disk data set, such as D/A output of a disk data set and A/D input.

There are other types of operations which can run within the framework of the math system. Two types of disk-based programs currently exist which can be accessed and executed from the math system. One of these types of programs is referred to as math system OPS, or system operators. They all exist on a specific track on the system resident disk pack, are read into memory by the math system OP processor and called as subroutines. They then have access to all the keyboard argument fetching and other special math system facilities. These are generally the rarely used and long programs where the user will not notice the extra disk action required to access them.

The other type of disk-based programs are the user defined OPS. These are different from the user programs in that they are written in macro-language and are assembled using the system assembler. They exist as program library items and are located, read into memory, and executed as subroutines by the math system when requested by the user (using the SYST key, followed by the program name). Again, this type of disk-based program has access to all the math system facilities such as keyboard argument fetching and curvilinear display output.

In summary, then, the SIGNAL Math Interactive System is a statement oriented, interpretative system which allows construction and execution of user programs, bypassing any sort of assembly step. Great flexibility is allowed in describing subsets of data sets by allowing very specific variable qualifying statements while not requiring it, and easy transformation of data set types by simply moving from one type of variable to another type (fixed point to floating point for instance). Saving of results is done automatically by the system until specifically requested by the user to purge the context, so he can easily run for days using the same exact data sets if he so desires. This amount of flexibility so simply attainable is one of the finest aspects of the SIGNAL interactive math system.

CHAPTER 4. ARPA NETWORK

As part of the work on this contract, the SIGNAL SYSTEM was interfaced to the ARPANET. This involved the hardware connection described in Chapter 1, Section 3 and software described here. After the connection was completed, the system was used to implement a real time LPC algorithm for speech compaction, transmission and reconstruction over the ARPANET. This experiment, covered more completely in the next chapter illustrated the need for uncontrolled network messages that minimize network overhead. The resulting protocol, called Network Voice Protocol (NVP), is covered in Section 3 of this chapter.

4.1 RELIABLE TRANSMISSION PROGRAM (RTP)

The Reliable Transmission Program (RTP) is designed to run in conjunction with the Modem Interface to send and receive meaningful data between the IMP and the Network Control Program (NCP). The fact that the connection to the IMP is a "Very Distant Host" (VDH) connection dictates the protocol to be followed by the RTP. This protocol is specified in detail in BBN report 1822, "Specifications for the Interconnection of a Host and an IMP," Appendix F.

The RTP and NCP both execute in SN/0, a computer similar in architecture and nearly identical in instruction set to the MP-32A where the processing and generation of incoming and outgoing data, respectively, occurs. SN/2 sends messages to the NCP in SN/0 which passes them on to the RTP. The RTP then breaks them up into packets, adds on the necessary code values at the beginning and end (and sometimes in the middle of packets under certain special circumstances), computes and adds on three 8-bit bytes of cyclic redundancy check code, and sends the data one byte at a time to the modem interface which sends it on to the IMP. For incoming messages, the modem interface signals the RTP when a byte has entered its input buffer from the IMP, the RTP accepts each byte and assembles them into packets, checks the cyclic redundancy code for errors, removes all special transmission codes, assembles the packets into messages and sends the messages to the NCP which, in turn, sends them on to SN/2 for processing.

There are several requirements which the RTP must meet in order to successfully exchange data with the IMP. One of these requirements is that the RTP must constantly let the IMP know that it is up and able to properly

exchange data. The method used for this involves special short packets called "HELLØ" and "I HEARD YOU" packets. Both the IMP and the HOST (the RTP) must independently send a "HELLØ" packet once every 1.25 seconds. Upon receipt of a "HELLØ" packet, each side must immediately respond with an "I HEARD YOU" packet. If either side fails to receive four "I HEARD YOU" packets in a row as responses to its "HELLØ" packets, it must declare the line down and not attempt to send any packets of any sort or accept any incoming packets for a period of ten seconds. After this mandatory ten-second period of no transmissions, both sides must receive four "I HEARD YOU" packets as responses to four consecutive "HELLØ" packets before normal transmission may resume.

Once the transmission and reception of normal packets has begun, there are some requirements to ensure that packets are not received out of order. There are two "channels" for transmission and two for reception. These "channels" are strictly logical devices used to keep packets in order. For each channel, there is an odd/even counter. Each packet being sent contains both its channel number and its odd/even parity on that channel. In addition to these channel numbers and parity bits, each packet carries acknowledge bits to acknowledge receipt of previous packets. So for each channel, it is necessary to receive an acknowledgement for the previous packet sent on that channel before attempting to send another packet on that channel. This, combined with the requirement that the channels be given packets to send in strict order, ensures that packets are received in order without having to number them.

In order to enable the modem interface on either end of the line to distinguish a packet from quiet line, there are specific codes which are used to begin and end each packet transmission. There is a two-byte sequence at the start and a different two-byte sequence at the end of each packet, in addition to a three-byte cyclic redundancy check (CRC) code at the end which provides some insurance that the correct bits are received in each packet.

The Modem Interface in SN/0 is very simple. On input, it monitors the line, checking for the quiet line byte (SYN code) and as long as it is finding the SYN code it does nothing. As soon as any non-SYN code appears, the interface interrupts the mainframe and the mainframe has 160 μsec to respond to the interrupt, before another input byte comes in. The interface will continue interrupting on every byte until the mainframe issues the command which tells the interface to go back watching SYN codes without interrupting. It is the job of the software to detect the end of a packet by watching for the code

bytes and CRC bytes. Similarly, on output, the interface will send SYN codes every byte time without interrupting the mainframe until the mainframe commands it to send other data. The interface then interrupts when a byte time expires, and the mainframe has to provide the interface a new output byte within 160 μsec. Again, it is the program's responsibility to issue the command which tells the interface to resume sending SYN codes until further notice. The software computes and checks the CRC bytes and adds on (or removes) the extra code bytes necessary to indicate packet start-up and termination.

The interface between the NCP and the RTP for sending to the IMP is the "Host-Imp Queue" (HIQ). This queue is a first-in first-out queue upon which the NCP places messages which it wants sent and from which the RTP removes messages when it is prepared to send them. When the RTP finds a message on the HIQ, it removes the message from the queue and begins breaking it up into packets for transmission. It builds the first packet from the message by placing the packet control word at the front of the packet and placing the actual words of the packet in one of the permanently allocated output buffers. There are two such output buffers, one for each output channel, and the RTP uses a program variable, ØCNLNØ, which keeps track of the current output channel number, to determine which of the output channels to use. There is, in addition to the channel number, a status word which indicates whether that buffer is available for use. This status word is updated by the input side of the RTP when an acknowledgement is received for that output channel. The output channel number is then updated and the other output buffer is filled with the next packet of the message if the buffer is available for use.

The RTP normally sends a packet as soon as it is built. Sometimes a packet must be resent because no acknowledgement was received for it. When it is time to send or resend a packet, the output channel is flagged busy and the CRC is computed at the same time as the code bytes are added on and the entire packet as it is to appear on the line is placed in the modem output buffer, one byte per 16-bit word. As soon as the CRC bytes are calculated and placed in the output buffer, the modem output is begun. The modem controller then sequences down the modem output buffer, providing the interface with data whenever an interrupt occurs requesting output until the end of the output list is reached. Since the "HELLØ", "I HEARD YOU," and NULL packets (packets sent to acknowledge input packets received in the absence of output packets to send) do not have a channel number, they bypass the channel output buffers and are merely placed right into the modem output buffer with their proper code and CRC bytes.

The modem input controller has a pool of four buffers, which rotate so that one is always available to use for an input packet, should one arrive. The controller writes every byte of the input data from the modem interface into its current buffer, checking for the packet termination byte sequence. When a packet has completely entered from the interface, the controller sets a new input buffer address for itself and flags the just-filled buffer as full, for the CRC program to check. Once the CRC has been checked, the code bytes are removed as the packet type is examined. If it is a special packet, i.e. "HELLØ," "I HEARD YOU", or NULL, the appropriate information is extracted and used, and the packet is immediately discarded; however, if it is a regular packet, it is placed in the correct channel input buffer according to the channel number and parity bits contained in the packet itself and the channel status words which indicate whether the channel input buffers are in use or not. The output parity bits are set at this time to reflect the fact that the packet has been properly received, and the acknowledge bits are extracted and used to determine which, if any, output packets have been received by the IMP.

The RTP keeps an input channel number, and when the channel input buffer corresponding to the input channel number is correctly filled, the RTP uses the packet therein to construct a message. The packets contain information to reflect whether they are the last packet of a message, so when the RTP has completed construction of an input message, the message is enqueued on the IMP-Host Queue (IHQ), which serves as the interface between the RTP and NCP for input in the same way as the HIQ serves for output.

4.2 NETWORK CONTROL PROGRAM (NCP)

For the special purpose of Network Voice Connections only a minimal NCP is necessary. The initial connection and all the message leader building is done by the application program, which only leaves a minimal set of functions for the NCP to perform.

One of the functions performed is to pass messages between the process running on SN/2 and the RTP which does the actual transmitting and receiving. This involves a transfer from computer to computer via our "Host interface" and dealing withthe two queues which are used to pass information to and from the RTP.

The other function of the NCP is to discard inappropriate messages which arrive from the network periodically.

The application program running in SN/2 will send a message to the NCP in SN/0 when it has one ready to send. The NCP will first locate an unused memory buffer from its queue of memory buffers, then accept the message and place it in this buffer. The next step is to build a "Message Queue Element" (MQE) which describes the buffer address and length, and place this MQE on the IMP-HØST queue, which the RTP queries periodically to check for new output messages. At this point, the NCP's job is done as far as that message goes. The RTP will release the buffer used for that message when the message is successfully sent to the IMP.

The RTP, upon receiving a completed message from the IMP, places the message on the Host-IMP Queue. The NCP removes messages from this queue and determines what type of message it is. If the message is inappropriate for the speech connection, it is ignored and the buffer it is in is freed; if the message is pertinent to the speech connection, it is passed on to SN/2 for processing, and then the buffer is freed. RFNM's, if they arrive, are passed on to the application program as a means for the program to be aware of how far ahead or behind the network is relative to the program's output message list.

4.3 NETWORK VOICE PROTOCOL (NVP)

Chapter 5 of this report covers an experiment using the SIGNAL SYSTEM to implement a real time LPC algorithm for speech compression, transmission, and reconstruction across the ARPANET. In this experiment, delays imposed by network handling of the packetized messages were of the order of three seconds. While these delays did not destroy the communication, they were too inconvenient for users to be satisfied with. Consequently, the Network Voice Protocol, or NVP, was defined. This new network message type eliminates the normal message ordering and error control mechanisms in the IMP. As a result, messages may arrive out of order or not arrive at all.

The removal of IMP message control does speed up transmission of messages, but places a burden on the HOST-HOST protocol to provide enough information in the messages themselves to allow whatever reordering is needed, as well as adjusting for lost messages. The Network Voice Protocol, as defined in NSC Note 43 by Danny Cohen entitled "Specifications for the Network Voice Protocol (NVP)," provides for a four-bit sequence number and a 15-bit time stamp in each data message. Detailed description of the use of the NVP on the SIGNAL SYSTEM can be found in Chapter 1.3 of the May 1975 Quarterly Report, Reference 7.

CHAPTER 5. REAL-TIME SPEECH COMPRESSION ON THE ARPA NETWORK

The Culler/Harrison signal processing computer system used for our research in speech compression was manufactured and installed in 1973 as part of this contract. The system includes two separate processing units, capable of either independent or closely coupled operation. The first of these, the MP-32A, is a general purpose computer with 16-bit addition, subtraction and logical operations and a 64K word main memory. The second processing unit is the AP-90 array processor, an arithmetic processor with its own data and program memories. Its general use is as a very high-speed arithmetic processor, optimized for computation of FFT, convolution and correlation operations on arrays of data. It performs 16, 24 or 32-bit addition, subtraction and multiplication in from 167 nanoseconds to 1 microsecond per operation.

5.1 LINEAR PREDICTVE CODING SPEECH COMPRESSION

The speech compression system used for low rate network voice communication uses the autocorrelation method of linear prediction to develop a set of parameters which describe the vocal tract of the speaker plus measures of pitch, amplitude and voicing of the speech. The speech data is analyzed at fixed time intervals to calculate these parameters which are then coded for transmission. The receiver decodes the parameters and computes a synthetic speech signal using the pitch excited filter defined by the parameters.

In network speech applications, the MP has responsibility for scheduling both processors, for buffering data, and for all aspects of network and user control. Secondary storage (disk) units and user consoles are interfaced directly to the MP. The AP is used to perform all arithmetically significant tasks for the system. These include most parts of both the analysis and synthesis processes for linear predictive coding of speech. Only the final pitch and voicing decisions, encoding and decoding, and network message preparation are performed in the MP. One additional processor, a prototype version of the MP, is used for analog input and output and the reliable transmission of data to and from the ARPANET IMP. Figure 2 illustrates the role of each processor in the network speech system.

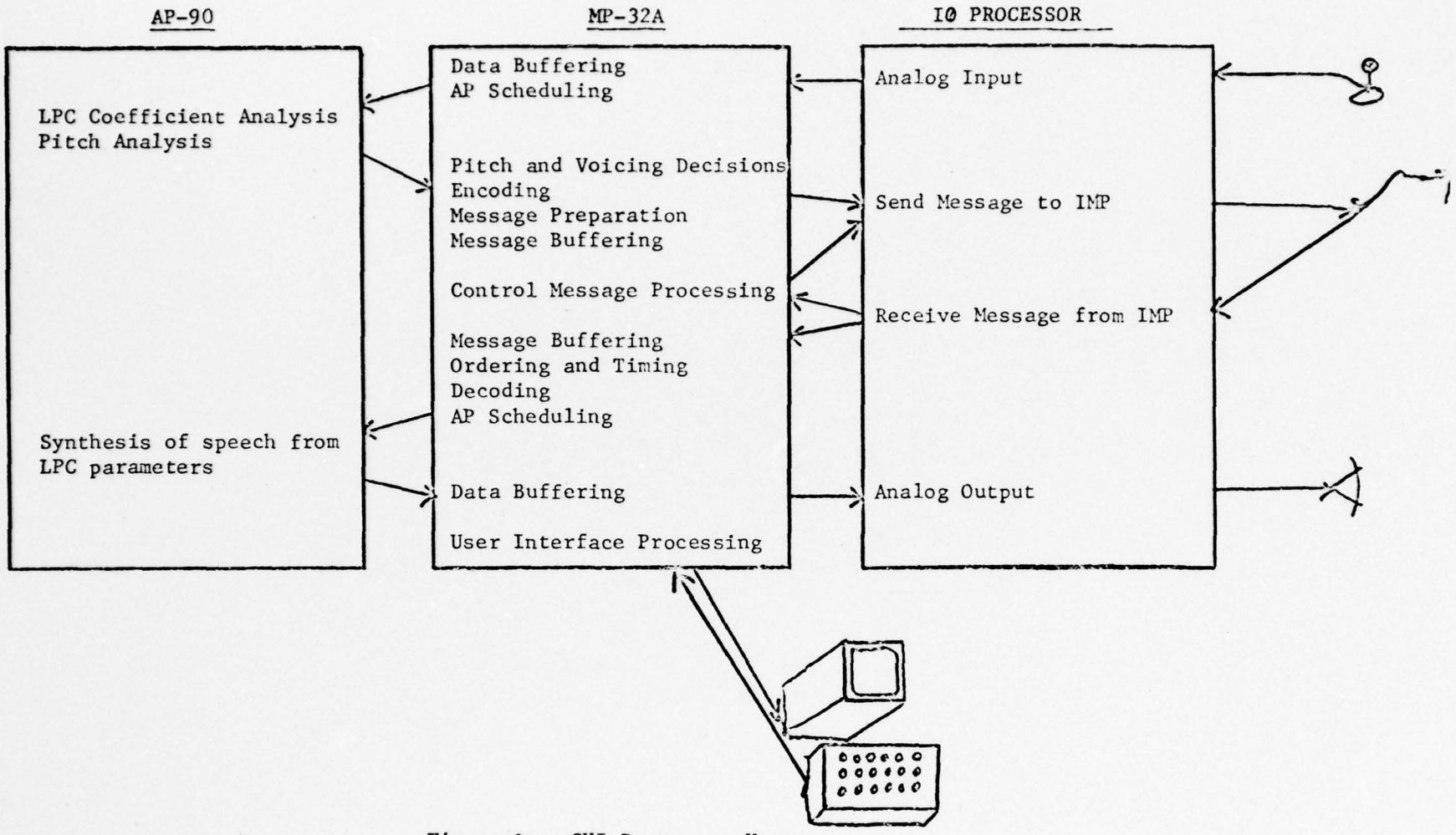

Figure 2. CHI Processor Usage

The linear predictive analysis of speech data requires by far the greatest amount of processing within the system. Each 9.6 milliseconds new reflection coefficients, pitch and gain parameters are calculated. The computations require over 7000 multiplies and 7000 additions for data sampled at 6.67KHz. The AP completes this computation in under 5 milliseconds. The calculation of a synthetic speech signal from the LPC parameters requires over 1300 multiplies and 1300 additions, and uses less than 1 millisecond of AP processing for the same interval. The AP is thus busy with analysis and synthesis about 60% of the time during real time full duplex speech processing. The remaining time is available for use in synthesizing additional data streams or computing acoustic encoding.

This real time implementation of the LPC speech compression algorithms has made it possible for us to judge their performance for a wide variety of speech signals. Our experience with this algorithm, as summarized in our quarterly technical report of May 1975 is:

a. The reconstructed signal generally does an excellent job of sounding like the speaker and conveys the intended message.
b. Whenever the quality of the input signal is lowered by spurious noises the output does not degrade as gracefully as we would like. There is a tendency to pop and crackle.
c. The pitch extraction part of the algorithm provides good results for the most part, but degrades too easily for low amplitude inputs.
d. For high pitched female voices, there is an artificial increase in gain for some sounds.

An additional point of some importance is that the perceived degradation in quality of the synthetic speech is largely due to the effects of the sharp frequency cutoff at 3.2KHz used with the 6.67KHz sampling frequency. In other words, the differences between the filtered original and the synthetic output are generally much less apparent than between the unfiltered and the filtered original signal.

In addition to our judgments on the quality of the LPC algorithm, there has been an independent measure of our implementation. A pilot test of the intelligibility of several speech compression systems conducted for ARPA by Stanford Research Institute gave our LPC system the highest rating of those studied. The LPC implementation on the CHI SIGNAL SYSTEM has been essentially unchanged since November 1974 and has been a stable, reliable system. Our

total network voice communication system has been very reliable with numerous experiments and demonstrations lasting from a half-hour to well over an hour with no failure. The hardware has demonstrated none of the unreliability sometimes associated with high speed signal processors while performing complex computations.

5.2 NETWORK VOICE COMMUNICATION

This system has been in use since November, 1974, for experiments in low bandwidth digital voice communication over the ARPANET. We have used these experiments to develop a protocol for controlling the transmission of speech parameters on the network, including procedures for variable rate transmission. These permit suppression of transmission when either speaker is quiet for more than a second. This feature alone permits reduction in the amount of information transferred by a factor of two to three. In early experiments, clear and understandable speech was received over the network, but delays in excess of three seconds from the time one speaker stopped to the time the beginning of the other speaker's reply was received were common. To decrease this network transit time as much as possible, further experiments were carried out using uncontrolled network messages to eliminate much of the normal network overhead. Procedures were developed through experimentation for properly dealing with the uncontrolled network messages, which sometimes are lost or received out of order. This experimentation continued through May 1975.

All the early experiments were carried out by CHI and MIT Lincoln Laboratories. The Lincoln Labs' system, a TX-2 computer with a special purpose processor, the FDP, was deactivated in June, 1975, ending that set of experiments. Subsequently, experiments in network voice communication have involved a new Lincoln Labs' system, using a PDP-11 with an LDVT, a high performance mini-computer specially designed for real-time speech processing, and a system at Information Science Institute using a PDP-11 with an SPS-41 signal processor as well as ourselves. All sites involved in these experiments are ARPA contractors working as part of the network speech compression group. These three sites will also be the initial participants in the planned network speech conferencing experiments.

As a result of our experiments in speech compression on the network we have developed experience in the problems which arise in such communication

and which must be faced in the extension to voice conferencing. For example, when several sites must implement protocols to control their communication, errors or differences in interpretation will inevitably occur. The solution to these differences may depend on the ability of participants to report back to each other exactly how their messages differed from what the receiver expected. In early experiments we were able to identify such problems as differing definitions of the K parameters (reflection coefficients) being transmitted, and incorrect calculation of the message time stamp and length by examining a trace record of the messages received. Another problem which occurs is how to determine when one person has finished speaking so that another may start. Even in two-person communication it has been difficult to avoid interrupting. This problem is due primarily to the relatively long transit time of speech over the ARPANET, compared to normal telephone or face-to-face conversation. This problem, if not controlled, can only be worse in conferences where many participants may wish to speak.

The May 1975 quarterly report, Reference 7, covers both the algorithm analysis and the network experiment in detail. It also includes a description of the modifications and refinements made to the algorithms and programs described by Markel and Gray.

CHAPTER 6. LABORATORY EXPERIMENTS IN PHYSICAL ACOUSTICS

The SIGNAL SYSTEM is the basis of an experimental acoustics facility at CHI. The analog subsystem (described in Reference 7) and a computer terminal are located in the laboratory and provide interfacing and programmatic control. The SIGNAL SYSTEM proved to be a flexible and very powerful research tool. The initial experimental work performed with this system was a test of the optimal filter as an area function predictor; results are detailed in the August quarterly report (Reference 8).

6.1 USE OF THE SIGNAL SYSTEM IN AN ACOUSTICS LABORATORY

The structure of the SIGNAL SYSTEM and the analog subsystem is described elsewhere in this report and in other documents. In this section the actual use of the system in the laboratory environment is briefly discussed.

Following is a list of the functions performed by the SIGNAL SYSTEM in the laboratory:

1. simultaneous output and input of audio data,
2. programmatic control of laboratory modules,
3. threshold detection and automatic signal averaging,
4. real time data analysis,
5. real time graphics (CRT) display of data and results of analysis,
6. synthesis of steady state and pulse signals for use in stimulating acoustical cavities and transducers.

The power of the SIGNAL SYSTEM as an experimental tool is further illustrated by considering a typical experimental situation. The impulse response of an acoustic cavity is measured in the following manner. An output waveform with sufficient frequency content to cover the range of interest is synthesized in the computer and loaded into an output buffer. This output is used to drive a DAC and a speaker thus supplying a driving signal to the cavity. The cavity response is monitored by a microphone which drives an ADC. DAC output and ADC input occur simultaneously so the response to the computer generated signal is fed back to the computer. By repeating the above procedure on a duty cycle the cavity response is signal averaged by adding the results of a sequence of cycles. Statistical calculations are done in real time to determine the amount of averaging necessary and the validity of a particular sample.

The power spectrum of the averaged data is computed and divided by the power spectrum of the system response in the absence of the cavity under test. This spectrum and the averaged impulse response are displayed on the CRT terminal.

It should be noted that the entire sequence described above may be initiated with as few as two button pushes at the computer terminal.

6.2 THE OPTIMAL FILTER AS AN AREA FUNCTION PREDICTOR

Under this contract we began investigating acoustical cavities as tools to verify the validity of discrete wave equations. Since we were also contracted to implement a real time LPC speech compression system, an obvious preliminary experiment was to use the LPC optimal filter to predict the cavity shape from the measured cavity response. This technique had been reported as a means to compute vocal tract area functions from speech data so we felt that simple cylindrical cavities would be no problem for the optimal filter technique. The results of these tests were inscrutable as were the results of similar attempts made on speech data. We could not see the shape of simple cavities in the reconstructed area function derived from the optimal filter.

In order to resolve the above mentioned anomalies, we were led to experimentally investigate the use of the optimal filter as an area predictor. The optimal filter is, of course, at the heart of the LPC speech compression algorithm. It has been shown by other workers that if the vocal tract is modelled as a series of segments, each with constant cross section, then the optimal filter coefficients derived from speech data yield the cross sectional areas of the segments. This has been tested in the past on theoretical cavity data and on speech data. In this report we present the first application of the technique to actual cavity data derived from a cavity with a directly measureable area function.

There are several reasons for performing such a series of experiments. First, they provide the most direct test of the theoretical link between optimal filter coefficients and cavity segment area functions. It is necessary to establish confidence in this very physical interpretation of the optimal filter if one is to accept the vocal tract area functions derived thereby as having validity.

Another motivation for the work was the feeling that if the optimal filter does indeed represent an extraction of the vocal tract area function from

the speech data, then a modification of the filter which makes it better represent the area function may have consequence in the speech compression applications of the filter. The work has not yet arrived at a point where this contention can be verified, but the experiments completed have indicated a length correction modification to the filter is necessary in order that it extract the actual area function from cavity data.

A third motivation for the experiments is that the specifications of the vocal tract area function during articulation is a problem which has generated a great deal of work quite independent of the question of speech compression. The optimal filter then represents the intersection of these two independent and quite active speech research trajectories, and as such is obviously fertile ground for investigation.

The study although preliminary in respect to the above mentioned questions has nevertheless yielded some quite interesting conclusions. The major accomplishments may be summarized briefly in the following list:

1. The impulse response of a collection of acoustical cavities has been measured. All of the cavities are constructed of uniform circular segments.

2. The cavity spectra as derived from the measured impulse response, has been fit fairly accurately by a model based on the lossless transmission line equations but with an extra inductive length correction added at each segment junction in the cavity.

3. The optimal filter technique has been used to derive the cavity area functions from the experimentally observed cavity spectra, as well as from the theoretically predicted spectra. These derived area functions are then compared to the actual cavity.

From this work the following conclusions may be drawn:

1. The derived area functions are always longer than the original cavity.

2. The area functions derived from actual data always seem to be more difficult to interpret than those derived from theoretical spectra, even though the theoretical spectra fit the data quite well.

3. Step changes in the area function of the original cavity were always larger than the corresponding changes in the derived area function.

This work is described in detail in the August Quarterly Report, Reference 9.

CHAPTER 7. NETWORK VOICE CONFERENCE RESEARCH

In December 1975 a one-year program of research in the use of Linear Predictive Coding for low bandwidth digital voice conferencing on the ARPA network was begun. This work was an extension of the network voice communication effort and built on it, adding additional protocols and controls necessary for a conference environment. As part of the network voice conference work, a major revision to the acoustic coding and transmission sections of the system was completed. This revision permits variable rate transmission of the speech parameters based on their acoustic information content, significantly lowering the average bandwidth required for digital voice communication. This work has been fully covered in three quarterly technical reports issued during the contract [9,10,11].

7.1 MODIFICATIONS TO THE SIGNAL SYSTEM FOR NETWORK VOICE CONFERENCE WORK

Conferencing, as used in this context, refers to voice communication among a group of participants located at diverse sites on the ARPANET. Each conference has a chairman, who controls who is to speak at any time. Normally, all participants listen to a single speaker. The speaker's voice is digitized locally and analyzed to produce the LPC transmission parameters which are then sent to each other participating site over the ARPANET. The receiving sites use these parameters to drive their LPC synthesizers to produce the speech signal which is played out to their participants.

This conference environment requires several extensions to the basic network voice protocols used for point-to-point conversations. The control protocol is extended to provide for multiple participants without requiring separate pairwise negotiations between all combinations of sites by having each potential participant negotiate only with the conference chairman. It then becomes the chairman's responsibility to provide the speaker's host with necessary information to transmit the speech data to each other site, and to inform the other hosts from whom they should expect data. Additional control messages are required throughout the conference to allow participants to request the floor and to give it up when they are through speaking.

Since there is only one speaker in the conference at a time, a single speech processor can service many local participants in the conference. To implement multiple local participants, we have used an analog switch controlled by the computer system to enable loudspeakers at those terminals participating in the conference and to select one terminal's microphone for input to the A/D system and analysis. This switch has been incorporated in the multi-channel analog subsystem described in chapter 1. The conference protocols provide for an extension field which is used to identify specific participants in both control and data messages.

The conference control functions of the system have been implemented in two logically independent sections, the local conference controller or LCC and the conference chairman or CHAIR. The LCC has the responsibility for digital interaction with the local participants at their terminals and control over the switching of analog signals to and from these terminals. It interfaces with the conference chairman through control messages communicated over the ARPANET, forwarding requests from participants to the CHAIR and enabling or disabling speakers and microphones in response to messages from the CHAIR. It also sets up control information for the speech data transmitter and receiver, providing the list of destinations for data messages when a local participant has the floor and selecting what speaker's data messages are to be accepted for synthesis.

The CHAIR program interacts almost exclusively with LCC's at the various participating sites over the ARPANET. It defines the parameters to be used for speech coding and transmission and negotiates with each LCC in turn as it first joins the conference. It maintains a queue of requests to speak received from participants through their LCC's and grants these requests in turn by sending control messages to all sites each time the speaker changes. The CHAIR may also be controlled by a local chairman who can communicate directly with it from his terminal. Each conference requires only one CHAIR, and this chairman need not be associated with a speech processor at all since its functions require only digital control information.

7.2 VARIABLE FRAME RATE CODING OF LPC PARAMETERS.

The phase I transmission algorithms for LPC speech communication over the ARPANET send a complete set of parameters every 19.2 milliseconds. This method is called a fixed frame rate system. By examining the parameters being produced by the analyzer, and only transmitting parameters when a significant change has taken place from the last set transmitted, a much lower number of frames will have to be transmitted. This approach, called variable rate transmission, was developed by Makhoul and Viswanathan at Bolt, Beranek and Newman, Inc. [12] and adopted as the phase II algorithm for network LPC experiments.

We implemented the first real time versions of the variable frame rate algorithms at CHI during the second quarter of 1976 [10]. The variable rate analysis algorithm requires the computation of a distance measure, the likelihood ratio,between each new set of prediction coefficients and those last selected for transmission. If the ratio exceeds a preset threshold, the new parameters are transmitted. Separate threshold tests are performed on the pitch and the gain parameters to determine if they should be transmitted. Thus, for each analysis frame, all parameters, no parameters, or any combination of the three (pitch, gain, reflection coefficients) may be transmitted. To identify which parameters are included, a three-bit header has to be transmitted for each analysis frame. Even with this extra overhead, it was possible to shorten the analysis frame interval to 9.6 milliseconds and still reduce the average bit rate to around 2000 bits/second for continuous speech.

The implementation of the variable frame rate system, called LPC-II, required major changes to the encoding and message preparation sections of the transmitter and to the corresponding decoding sections of the receiver. The distance measure computation for the prediction coefficients produced by the analysis and the interpolation of the transmitted parameter to fill in missing values required additional computations in the array processor. The effect on total utilization of the processors was very small, however, and the system has no difficulty in completing the computations in real time for a full duplex network voice conversation.

By the completion time for this contract, initial experiments had taken place over the ARPANET using the variable frame rate LPC-II system. These experiments have been conducted between CHI and USC Information Sciences Institute. They have demonstrated that transmission rates below 2000 bits/second are sufficient for reasonable quality digital voice transmission. The LPC-II algorithms are now being used as the standard for network voice conferences on the ARPANET, although experiments in this area are limited by the sites which at present support this protocol.

References

1. MP-32A Macroprocessor Maintenance Manual (TMB2).

2. MP-32A Macroprocessor Wire List - SN/2.

3. Model 114 Disk Drive Technical Manual, Century Data Systems (now Calcomp).

4. Instruction Manual for Type 611 Storage Display Unit, Tektronix, Inc. Beaverton, Oregon.

5. AP-90 Array Processor Maintenance Manual (TMB8).

6. AP-90 Array Processor Wire List - SN/2.

7. Real Time Implementation of an LPC Algorithm, ARPA Quarterly Technical Report, CHI, May 1975.

8. Experimental Investigation of the Optional Filter as an Area Function Predictor, ARPA Quarterly Technical Report, CHI, Aug. 1975.

9. Network Voice Conferencing, ARPA Quarterly Technical Report, CHI, March 1976.

10. Network Voice Transmission, ARPA Quarterly Technical Report, CHI, July 1976.

11. A Variable Frame Rate Network Voice System, ARPA Quarterly Technical Report, CHI, September 1976.

12. Specifications for ARPA-LPC System II, NSC Note 82, Bolt Beranek and Newman, Inc., February 1976.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
CHI-FIN-101R		

4. TITLE (and Subtitle)
SPEECH SIGNAL PROCESSING RESEARCH
~~Final Technical Report~~
~~Speech Signal Processing Research at CHI~~

5. TYPE OF REPORT & PERIOD COVERED
Final Technical Report
Mar 73 - Dec 76

7. AUTHOR(s)
~~Glen J. Culler,~~
Michael McCammon, James F. McGill, Dale E. Taylor, Jan M. Vanderford

8. CONTRACT OR GRANT NUMBER(s)
DAHC15 73 C 0252; ARPA Order 2359

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Culler/Harrison, Inc.
150-A Aero Camino
Goleta, California

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Program Code: P5P10

11. CONTROLLING OFFICE NAME AND ADDRESS
Defense Supply Service -- Washington
Room 1D 245, The Pentagon
Washington, D.C. 20310

12. REPORT DATE
December 1975 - Revised Apr 78

13. NUMBER OF PAGES
37 P.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Defense Contract Administration Services District, Van Nuys
18321 Ventura Blvd.
Tarzana, California 91356

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

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

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2359.

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

SIGNAL SYSTEM	Speech Compression
Macroprocessor	Speech Analysis-Synthesis
Array Processor	Linear Predictive Coding
SIGNAL Interactive Math System	Network Voice Protocol
ARPANET	Acoustic Data Collection

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

This report covers the work performed by Culler/Harrison, Inc. (CHI) under Contract No. DAHC15 73 C 0252 for the period March 1, 1973 to December 1, 1976. During this period, Culler/Harrison, Inc. has developed a powerful, versatile signal processing capability for ARPA and utilized this capability to perform significant and diverse signal processing experiments. The contract called for construction, operation and maintenance of the CHI SIGNAL SYSTEM, a combination of equipment and software providing a unique marriage of computing power and on-line interactive control of that power. The system has proved its flexibility

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 276

and reliability on a broad range of applications encompassing speech compression, on-line control of laboratory experiments, text editing and controlled thermonuclear reactor simulation.