ESD-TR-79-281

LEVEL III

(12)

Semiannual Technical Summary

Information Processing Techniques Program

Volume I: Packet Speech Systems Technology

30 September 1979

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-78-C-0002 by

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

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 F19628-78-C-0002 (ARPA Order 2006).

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

FOR THE COMMANDER

Raymond L. Loiselle

Raymond L. Loiselle, Lt. Col., USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

INFORMATION PROCESSING TECHNIQUES PROGRAM
VOLUME I: PACKET SPEECH SYSTEMS TECHNOLOGY

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

1 APRIL – 30 SEPTEMBER 1979

ISSUED 6 MARCH 1980

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes work performed on the Packet Speech Systems Technology Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the period 1 April through 30 September 1979.

CONTENTS

INTRODUCTION AND SUMMARY

The long-range objectives of the Packet Speech Systems Technology Program are to develop and demonstrate techniques for efficient digital speech communication on networks suitable for both voice and data, and to investigate and develop techniques for integrated voice and data communication in packetized networks, including wideband common-user satellite links. Specific areas of concern are the concentration of statistically fluctuating volumes of voice traffic; the adaptation of communication strategies to conditions of jamming, fading, and traffic volume; and the eventual interconnecting of wideband satellite networks to terrestrial systems.

Previous efforts in this area have led to new vocoder structures for improved narrowband voice performance and multiple-rate transmission, and to demonstrations of conversational speech and conferencing on the ARPANET and the Atlantic Packet Satellite Network.

The current program has two major thrusts; i.e., the development and refinement of practical low-cost, robust, narrowband and variable-rate speech algorithms and voice terminal structures, and the establishment of an experimental wideband satellite network to serve as a unique facility for the realistic investigation of voice/data networking strategies. Efforts prior to FY 79 in these two areas were reported separately in the Packet Speech and Wideband Integrated Voice/Data Technology Semiannual Technical Summaries. The current Packet Speech Systems Technology Program includes efforts in both areas.

This report covers work in the following areas: development of a custom-LSI-based narrowband channel vocoder; studies and real-time implementation of improved vocoder structures and algorithms; evaluation of speech processing capability of CCD-based spectrum analysis hardware; preparation of hardware and firmware for an upcoming speech algorithm test and evaluation program; design and implementation of packet voice terminals and local area access facilities; progress in satellite network and internetted packet voice conferencing and in the development of new packet voice network protocols; and design of a traffic emulation module for the wideband experimental network. Progress during FY 79 on experiment definition and planning for the wideband network will be reported in a separate Project Report.

Two prototype units of the LSI channel vocoder have been completed, subjected to duplex transmission tests, and readied for performance evaluation with the custom analyzer and synthesizer integrated circuits being completed at Texas Instruments. A hybrid vocoder using a Spectral Envelope Estimation (SEE) analyzer and a Linear Prediction Coding (LPC) synthesizer has been implemented, and listening tests have indicated a slight preference for the SEE-LPC hybrid over a complete LPC system. A real-time dual-rate version of the embedded coding subband-coder/channel-vocoder has been developed and prepared for formal testing. Experiments have been conducted in which the digital spectrum analysis in a

SEE vocoder is replaced by analog spectrum analysis carried out in a commercial CCD device; results indicate need for improved dynamic range, reduced DC drift, and reduced phase sensitivity in the CCD device. An EPROM-based Lincoln Digital Signal Processor (LDSP) adjunct to allow the LDSP to operate in a stand-alone mode for vocoder algorithm test and evaluation has been implemented. An initial prototype of the packet voice terminal and access network system has been designed in detail; key subsystems including the terminal processor, the access network buffer and control processor, and the cable modem have been fabricated, and preliminary testing has begun. An internetted voice capability has been developed and utilized to demonstrate internetted packet voice conferencing involving the ARPANET and the Atlantic Packet Satellite Network. Progress has been made in development of new protocols to support speech communications in packet networks. A traffic emulation module to support testing and performance measurements of the speech-traffic-handling capability of the wideband experimental network has been designed. A summary of the preliminary experiment plan for the wideband integrated network was presented in the 30 September 1978 Semiannual Technical Summary on Wideband Integrated Voice/Data Technology. Progress which has geen made in overall definition and planning of experiments for the wideband network, including updated planning in testbed development, system validation, and advanced system experiments, will be described in a separate Project Report.

PACKET SPEECH SYSTEMS TECHNOLOGY

I. CCD/BELGARD VOCODER

Two prototype units have been completed and subjected to duplex serial transmission tests at four switch-selectable data rates (Fig. 1). All control functions have been successfully demonstrated and verified including bit-stream synchronization, real-time in-out involving the analysis and synthesis devices, and coding/packing operations. Performance evaluations will begin immediately upon arrival of finished analyzer and synthesizer chips from Texas Instruments.

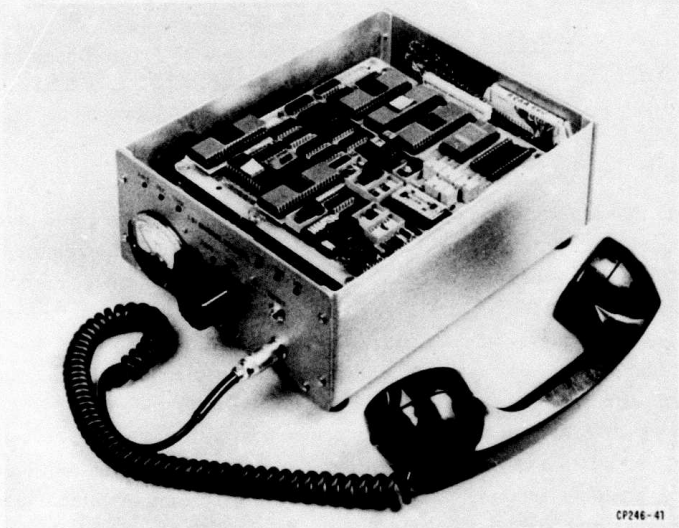


Fig. 1. Channel vocoder engineering prototype.

A new transmission format and concomitant synchronization protocol have been developed and tested which exhibit a fast acquisition capability and a low probability of spoofing. They are also compatible with U.K. 2400-bps standards thereby supporting interoperability between U.S. and British units at that rate.

Detailed investigations have been completed exploring the possibility of implementing a version of the McLarnon frame fill-in technique to yield a higher quality 1200-bps system. The studies indicated that by carefully designing the data buffering strategy there was sufficient remaining data memory capacity in the present control microprocessor configuration to support this algorithm. Present plans call for upgrading the current 1200-bps capability via the McLarnon technique subsequent to overall system performance evaluations with finished LSI chips.

The problem of retrofitting the current design for local access area compatibility was also explored in substantial detail. It was concluded that no hardware modifications of any consequence need be anticipated and only minor, straightforward revisions of the control microcode will be required.

II. VOCODER STUDIES

A. SEE Vocoder

A hybrid vocoder using a Spectral Envelope Estimation analyzer and a Linear Predictive Coding synthesizer has been implemented in both clear speech (Gold-Rabiner pitch) and noisy speech (noise suppression and maximum-liklihood pitch) versions. Both were left uncoded to provide a clearer indication of the fundamental system performance.

Preference tests have been performed on three vocoders: SEE, the hybrid (SEELPC), and LPC. Subjects were given an A-B presentation and asked to choose the better vocoder. Since three vocoders were compared, three tests were performed, each comparing two of the vocoders. Each test was composed of six talkers uttering two sentences. Ten to 13 subjects were used ranging from trained listeners to people who had probably never heard vocoded speech.

The untrained listeners' preferences were as follows:

Vocoders		Percent of Trials Choosing Preferred Vocoder
Preferred	Not Preferred	
SEE	LPC	59
SEE	SEELPC	63
SEELPC	LPC	59

The tests indicate that a simple rank ordering of the vocoder systems would place SEE at the top, LPC last, and SEELPC in between. Trained-listener opinion also resulted in this rank ordering. The difference between the LPC and SEELPC systems was judged not to be graphic enough to warrant the use of the SEELPC analyzer in an LPC-based communications network given acoustically benign background conditions.

An informal comparison was performed between LPC equipped with maximum-likelihood pitch and SEELPC with maximum-likelihood pitch plus noise suppression. E3A Airborne Command Post noise was used as the background acoustic corruption. As the noise suppression capability could only be applied to the hybrid system, systematic comparison would have been invalid. Therefore, trained-listener opinion was used as the means of comparison. Though the intelligibility of the systems was not considered markedly different, a slight preference for SEELPC emerged. The hybrid analyzer would probably be preferred in high background noise situations as it offers robust maximum-likelihood pitch estimation in addition to noise suppression. (While not tested here, a robust pitch estimate is known to significantly enhance the intelligibility of a vocoder with noisy input speech.)

A version of the SEE vocoder with maximum-likelihood pitch and background noise suppression has been developed for Narrowband Speech Consortium† testing. It is essentially identical to the earlier version except that the analyzer and synthesizer have been implemented in two separate LDSPs with a 2.4-kbps serial bit stream connection.

†The Narrowband Speech Consortium is a group of DoD organizations concerned with coordination of activities in the area of narrowband digital speech communication. An extensive series of intelligibility and quality tests of digital speech algorithms was conducted under consortium auspices in 1975 at Defense Communications Engineering Center in Reston, Virginia. The tests referred to here are a new series of consortium speech algorithms evaluations being initiated in November 1979, and conducted at the Rome Air Development Center COMSEC Engineering Office on Hanscom Field, Massachusetts.

B. Dual-Rate Vocoder

The multiple-rate channel vocoder discussed in a previous Semiannual Technical Summary† was computationally demanding and did not operate in real time. In order to meet the requirements necessary for participation in the Narrowband Speech Consortium tests, we have evolved a dual-rate (2400 and 9600 bps) embedded-coding vocoder from this earlier design which runs in a half-duplex, real-time mode. Because of time constraints, and because of the similarity of the 2400-bps system to Belgard, formal testing of the dual-rate vocoder will be carried out only at the 9600-bps rate.

1. Description of Algorithm Implementation

The dual-rate channel vocoder analyzer utilizes a bank of bandpass filters as in a 19-channel Belgard vocoder. The usual quantized log channel weights are transmitted along with pitch period and voicing information derived by a version of the Gold-Rabiner pitch detector incorporating median smoothing. The coding of this information results in a transmission rate just under 2400 bps.

Since the vocoder channels are more or less independent, each processing a different portion of the speech spectrum, they can be used as the basis for creating a set of "subband" signals which can be superimposed on the baseline channel vocoder to improve quality and robustness.

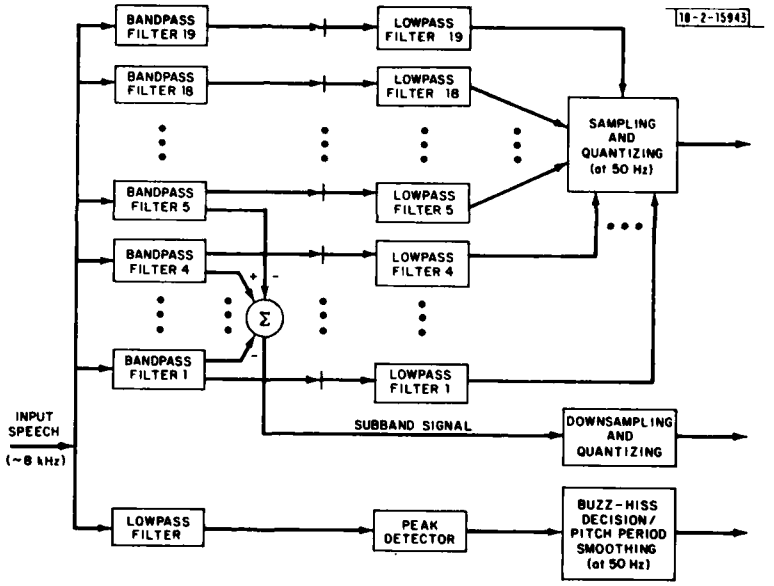


Fig. 2. Dual-rate channel vocoder analyzer.

These are downsampled, quantized versions of the summed outputs of a subset of contiguous bandpass filters. By using the first five channels to create a single subband signal which is downsampled 5 to 1, the analyzer can transmit the augmented vocoder information at an overall rate of under 9600 bps. Figure 2 depicts this configuration.

†Information Processing Techniques Program Semiannual Technical Summary, Vol. I: Packet Speech Systems Technology, Lincoln Laboratory, M.I.T. (30 September 1978), pp. 9-10 DDC AD-A066249.

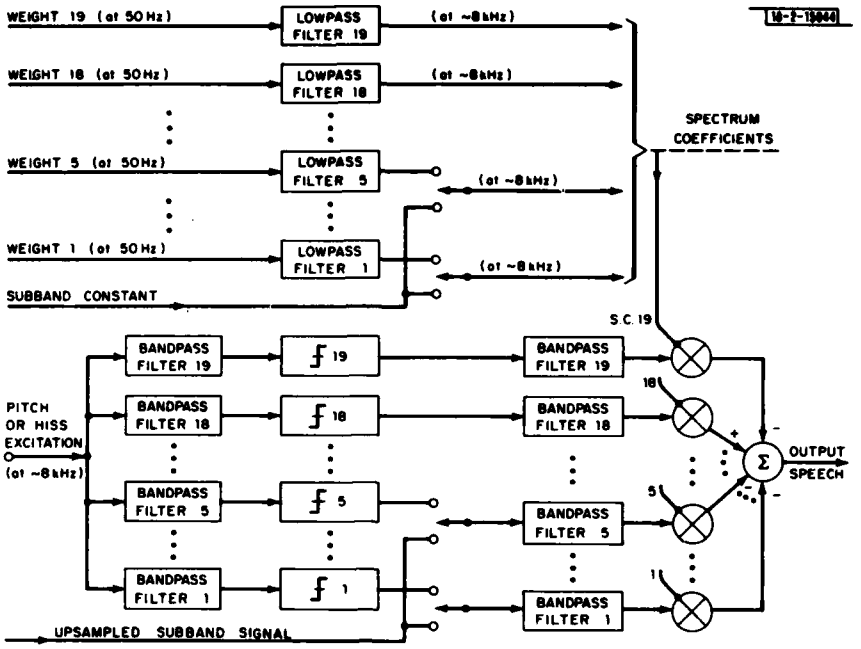


Fig. 3. Dual-rate channel vocoder synthesizer.

18-2-15945

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
PITCH		SUBBAND ENERGY PARAMETER						SYNCH PATTERN							
WT. 5 Δ MOD.		WT. 4 Δ MOD.		WT. 3 Δ MOD.		WT. 2 Δ MOD.		WT. 1 PARAMETER				PITCH ...			
WT. 13 Δ MOD.		WT. 12 Δ MOD.		WT. 11 Δ MOD.		WT. 10 Δ MOD.		WT. 9 Δ MOD.		WT. 8 Δ MOD.		WT. 7 Δ MOD.		WT. 6 Δ MOD.	
SUBBAND SIGNAL 1				WT. 19 Δ MOD.		WT. 18 Δ MOD.		WT. 17 Δ MOD.		WT. 16 Δ MOD.		WT. 15 Δ MOD.		WT. 14 Δ MOD.	
SUBBAND SIGNAL 5				SUBBAND SIGNAL 4				SUBBAND SIGNAL 3				SUBBAND SIGNAL 2			
⋮				⋮				⋮				⋮			
SUBBAND SIGNAL 33				SUBBAND SIGNAL 32				SUBBAND SIGNAL 31				SUBBAND SIGNAL 30			

Fig. 4. Format of frame of parameters of dual-rate vocoder.

The underlying structure of the dual-rate channel vocoder synthesizer is a spectrally flattened version of the 19-channel Belgard vocoder receiver. The received channel weights are decoded, anti-logged, and applied to a bank of 19 lowpass filters whose outputs are the 19 spectrum coefficients. When subband information is present, an empirically chosen constant becomes the spectrum coefficients for the affected set of channels. Also, when the subband signal is available, it is applied directly to the appropriate subset of final bandpass filters, thus bypassing the spectral flattening. This is shown in Fig. 3.

To achieve a rate of 9600 bps, a sampling interval of 121.25 μsec and a frame length of 165 samples were chosen. Every 20 msec the analyzer fixes the parameters to be transmitted. An 8-bit synchronization pattern always begins the frame of speech parameters. The subband "energy" (the maximum magnitude of the summed outputs of the subband during the last 20 msec) is quantized to one of 64 values ranging from 0 to 1 and a 6-bit table look-up parameter is sent to the synthesizer. For each input sample, the summation of the first five bandpass filter outputs is stored. The center frequencies of the subband filters are 240, 360, 480, 600, and 720 Hz. At the end of each frame, a table of eight fixed values is multiplied by the quantized value of the subband energy to generate a new table of values. (The synthesizer will generate an identical table.) The analyzer quantizes every fifth subband sample to one of the 8 values in the table and transmits a 3-bit value plus sign, resulting in 33 subband signals of 4 bits each. The synthesizer will apply these signals directly to the second set of bandpass filters in the subband region, padding out missing subband samples with zeros. The format of the coded parameters is shown in Fig. 4.

The underlying vocoder spectral information is transmitted using the standard Belgard coding scheme. After logarithmic compression, the lowest frequency spectral weight is quantized to 3 bits at 6 dB per step thereby spanning 48 dB of dynamic range. The remaining 18 parameters are encoded relative to the 3-bit reference using a 2-bit deltamodulation technique with steps of ±3 or ±9 dB. Therefore, 39 bits are required for the spectral characterization.

Pitch period information is linearly encoded to 7 bits and includes an implicit voicing indicator (pitch code = 0).

The mechanism which compensates for discrepancies between the modem and the vocoder transmission rates is as follows: if the vocoder analyzer is running on the average slower than the modem, causing the output buffer to empty, the analyzer repeats a frame of parameters as soon as a frame of silence is encountered. If the analyzer is running faster than the modem, causing the output buffer to overfill, the next frame of silence parameters is omitted. Similarly, if the vocoder synthesizer is running slower than the modem, thereby causing the input buffer to overflow, a frame of silence parameters is discarded. If the synthesizer is running faster than the modem, and the input buffer tends toward underflow, the next frame of silence parameters will be used to synthesize two frames of speech, thus allowing the input buffer to replenish itself. If either buffer reaches a critical state, the appropriate action is taken immediately.

The dual-rate vocoder has been implemented to run on two Lincoln Digital Signal Processors (LDSPs), the analyzer in one and the synthesizer in the other, with a unidirectional channel between the two transmitting a serial bit stream at the selected rate. The parallel-to-serial (P/S) and serial-to-parallel (S/P) conversions are functions of the analyzer and synthesizer, respectively. Analog input to the analyzer is converted to a digital signal by a signal conditioner unit whose sampling rate, filter, gain, and emphasis options are initially set by the analyzer in the LDSP. Similarly, the digital output of the synthesizer is converted to an analog signal via a

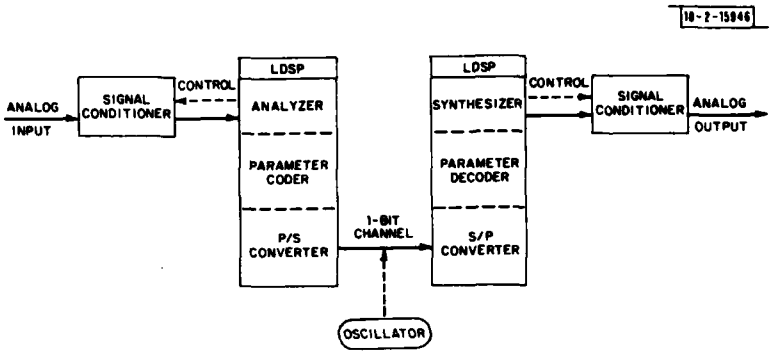


Fig. 5. Configuration of dual-rate vocoder in real environment.

signal conditioner whose mode is controlled by the receiving LDSP (see Fig. 5). Although the information received by the synthesizer includes the subband signal, the listener has the option of hearing the output of the 2400-bps portion of the vocoder alone.

2. Synchronization Issues

For the Speech Consortium tests, initial synchronization will be acquired in the absence of channel errors and then the synchronization mechanism can be disabled. This is because modem bit slot integrity is guaranteed. However, since a frame of parameters in the dual-rate vocoder includes room for an ample synchronization pattern, the particular synchronization algorithm used in this vocoder seemed an excellent candidate for a study of the issues that must be addressed if a synch algorithm is to be designed which is robust to the presence of channel errors and loss of modem bit integrity. These issues are discussed below in the context of the 9600-bps algorithm.

Synch acquisition in the absence of modem channel errors has been achieved in the past by using a fairly straightforward, computationally inexpensive algorithm which detects loss of synch instantaneously. When the synch pattern is incorrect, the next 23 8-bit patterns are checked. If a pattern matches, synchronization is assumed; and the remaining parameters are used by the synthesizer. If no pattern matches, parameters producing silence are injected; and 1 bit of the next frame is discarded such that a precessed set of 8-bit patterns will be checked in the next 20-msec frame slot. Synch is regained in a maximum of 8 frames (160 msec), provided that a random 8-bit pattern which matches the synch pattern is not encountered before the true synch pattern. The probability of acquiring synch falsely is not insignificant if an average of 96 searches is assumed before reaching the true synch pattern. The probability of a randomly chosen 8-bit pattern failing to match the synch pattern is 0.996+; and, therefore, the probability of acquiring synch correctly before encountering a false alarm is $(0.996+)^{96} \approx 0.687$. In testing this particular algorithm by artificially forcing loss of synch, the synch acquisition time has been found to be reasonably short in practice.

If the synchronization algorithm is to be modified to operate in a 5% channel error environment, the following problems must be faced:

(a) False indications of synch loss (spoofing) due to channel errors.

(b) Failure to detect legitimate loss of synch due to less stringent checking criteria directed at reducing spoofing.

(c) Maintaining a reasonable synch acquisition time without drastically increasing the chances of a false lock.

The first two problems are solved by checking for a near match of the synch pattern over a number of frames. The only negative feature of this solution is an increase in the expected time it takes to detect loss of synch. The third problem is more difficult, the solution for which involves considerable computation and increases the chance of acquiring synch falsely. A detailed discussion of the modified synch algorithm follows.

The probability of n bits in error in an 8-bit sequence, assuming a bit error probability of 0.05, is given by

$$P_r[n,8] = \frac{8!}{n!\,(8-n)!}\,(0.05)^n(0.95)^{8-n} \quad .$$

If the criterion for accepting the received pattern as the true synch pattern is that at least 7 bits are correct, then the probability of correctly identifying the noisy synch pattern is ≈0.943. In particular, the probability of falsely declaring the pattern incorrect is ≈0.057. To prevent inadvertent re-initiation of acquisition mode, the criterion for detecting loss of synch is that six consecutive synch pattern fields have failed to match. This results in a probability of 3.5×10^{-2} that a false synch loss indication will arise induced by channel errors. The expected interval between these occurrences is 159 hr, clearly a tolerable situation.

Given the less stringent synch maintenance criteria, a question arises as to the effect randomly occuring patterns will have on the detection of legitimate synch loss. Out of the 256 possible 8-bit patterns, 9 would pass the 7-bit match test, resulting in a probability of ~0.035 of accepting a given pattern or a probability of ~0.965 of declaring a mismatch. The probability of finding six consecutive mismatches is ~0.809.

To acquire synch in the presence of noise, the algorithm used for the noiseless environment must be supplemented with an additional test. If a pattern is encountered that matches synch perfectly, resynchronization takes place immediately. If the pattern has only a 7-bit match, the corresponding pattern in the following frame (which is available on the spot) is checked for a match of at least 7 bits. If the match is successful, synch is assumed. If the pattern is the true synch pattern, then the probability of correct recognition is:

$$P_r(\text{success}) = P_r(8 \text{ bits correct}) + P_r(7 \text{ bits correct}) * P_r(\text{at least 7 bits correct}) \approx 0.927 \quad .$$

If the pattern is not the synch pattern, then the following probabilities for random matches hold:

P_r(8-bit match)	$\approx 3.9 \times 10^{-3}$
P_r(7-bit match)	$\approx 3.1 \times 10^{-3}$
P_r(at least 7-bit match)	$\approx 3.5 \times 10^{-3}$

The probability of incorrectly accepting the random pattern is

$$P_r = p(\text{8-bit match}) + P_r(\text{7-bit match}) * P_r(\text{at least 7-bit match}) \approx 0.5 \times 10^{-3} \quad .$$

Thus, the probability of not accepting the random pattern is ~0.995. If an average of 96 searches is required to reach the true synch pattern, then the probability of not resynching until the true synch pattern is reached is $(0.995-)^{96} \approx 0.618$, as compared with 0.688 in the error-free algorithm.

By making yet a third frame of parameters available in the input buffer, this probability is raised to ~0.684.

This more robust synchronization algorithm has been incorporated in the dual-rate vocoder synthesizer and tested in the presence of channel errors at a rate of 5 percent. Reacquisition of synch, after a forced loss, usually requires a fraction of a second and occasionally on the order of a full second.

III. CCD CHIRP-Z TRANSFORM HARDWARE

Tests of the Reticon CCD Chirp-Z transform (CZT) hardware have been performed using the SEE vocoder as a vehicle. The CZT was used to implement the first transform in the SEE analyzer which is a 256-point Discrete Fourier Transform (DFT) of the windowed input signal. Since the CZT processes 512-point blocks, its input has to be upsampled by two to match the DFT in the vocoder.

The CZT was operated in a block-to-block mode to implement a true (not sliding) DFT. Since no synchronization input to the CZT is provided, the synchronization pulse must be detected in the (LDSP) handler routine to remove the resulting rotation of the CZT output buffer. A choice of synchronized or unsynchronized input to the CZT was provided by either waiting for or ignoring synchronization pulse arrival before entering the data buffer. In theory, input synchronization affects only the phase of the output of a DFT. Synchronization should not be an issue since only the magnitude of the DFT is returned by the CZT hardware to the LDSP vocoder algorithm. Also, provision was made for removal of the DC bias on the output of the CZT.

The CZT initially showed an abnormally low signal-to-noise ratio which required some special interface tailoring to meet its requirements. First, the adaptive time window of the vocoder was fixed at full size to maximize the input energy into the CZT. Also, the input was upsampled by two to fill the input buffer rather than downsampled at the output buffer. Finally, the input buffer was converted to block floating point to always provide maximum input level within overload limits.

The output of the CZT consists of a DC offset, a level indicating the spectral magnitude, and noise. The DC offset tended to drift and required careful cancellation (in the LDSP handler) each time the unit was operated. Offset removal from the magnitude signal proved to be very crucial to preserving effective system dynamic range. Maximum dynamic range was noted to be about 48 dB using a sine wave input

The real-time SEE vocoder software was provided with a switch input to select either the CZT-implemented DFT or an internal software FFT to allow A-B comparison. In spite of the carefully tailored preconditioning of the CZT input data and hand adjustment of the DC offset removal, the CZT was not found to perform as well as the software FFT. The CZT-processed speech exhibited a spectral flutter which was both audible as a pronounced roughness and visible on the real-time spectral displays of the vocoder. The DC bias removal adjustment was seen to imply a trade-off between either a muffled (offset removal >0) or a hollow (offset removal <0) type of sound.

Synchronized input to the CZT yielded less spectral flutter than unsynchronized input. Subsequent static tests inspired by this observation have shown the CZT to exhibit an anomalous phase sensitivity which is not currently understood. The problem appears to lie in the transform portion of the hardware rather than the spectral magnitude circuitry. However, tests designed to bypass the analog magnitude hardware in favor of an internal digital computation are currently

inconclusive due to noise problems in acquiring the necessary signals. It is possible that correction of the phase problem will significantly improve CZT performance.

Overall, our impression is that the tested CZT unit does not yield adequate performance to permit field use of the CZT in a SEE vocoder implementation. The dynamic range should be improved, the DC drift must be stabilized, and input phase sensitivity must be reduced. Certain improvements in hardware interface, such as a synchronization command, would enhance the utility of the device.

IV. CONSORTIUM TEST HARDWARE

Preparations for the Speech Consortium tests were completed during this period. Hardware in support of these tests has been fabricated and successfully tested. This consists of a pair of LDSP peripherals (shown in Fig. 6) specifically designed to bootstrap of a full 6K LDSP core image from six banks of dual 1K × 8 EPROMS resident in the peripheral boxes. Paper tapes containing programs of interest to the Consortium tests were prepared using software resident in the PDP-11/70, and the tapes were used to program the EPROMS.

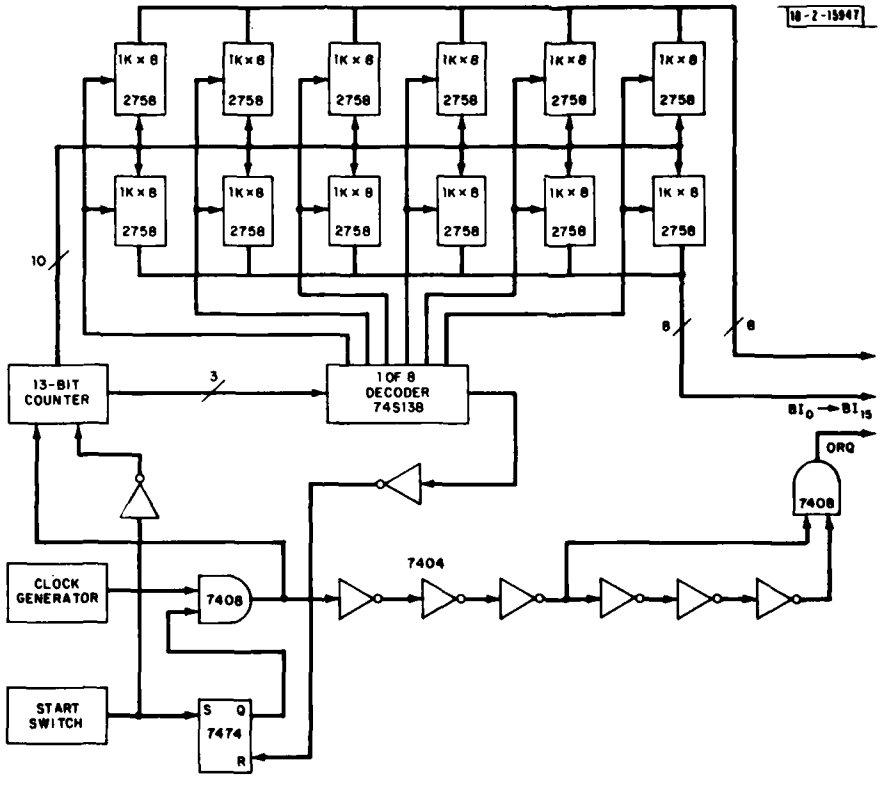


Fig. 6. LDSP Standalone Adjunct.

Loading is accomplished by virtue of a single push button which issues an external interrupt to LDSP input channel 7 while resetting the address counter in preparation for the first word transfer. The LDSP bootstrap asks for words in reverse order, and therefore, the mapping of prom addresses is organized as shown in Fig. 7. Words are transferred from the peripheral on a forced-handshake basis by providing an input acknowledge to the LDSP at a 1-MHz rate as determined by the timing in the peripheral box. A 13-bit counter/decoder addresses the six banks of EPROM in the order corresponding to the mapping required for the LDSP.

18-2-15948

		3777
BANK 0	M_P	2000
		1777
BANK 1	M_P	0
		7777
BANK 2	M_D	6000
		5777
BANK 3	M_D	4000
		3777
BANK 4	M_D	2000
		1777
BANK 5	M_D	0

Fig. 7. EPROM Map for LDSP Standalone Adjunct.

The two Lincoln Laboratory DARPA-sponsored algorithms which will utilize this hardware for Consortium tests are the dual-rate vocoder (to be formally evaluated only at 9600 bps) and the SEE vocoder (at 2400 bps). The firmware modules comprising the dual-rate system (an analyzer and synthesizer) have been loaded into a pair of LDSPs and tested successfully using a serial interconnection. When Consortium testing of this system is completed, the EPROMs will be erased and reprogrammed with the next algorithm to be tested. Control tapes have already been prepared for programming of both DARPA algorithms presently destined for testing. Level converters and associated clock shaping logic have been provided in each peripheral box to allow simple connection to the simulated modem serial link as provided by the Consortium test facility.

V. PACKET VOICE TERMINAL AND ACCESS-AREA DESIGN

The previous Semiannual Technical Summary compared a number of designs for packet voice access areas and described the architecture of a flexible packet voice terminal which could be utilized in conjunction with a variety of access networks and vocoder algorithms. A single-cable ETHERNET-like structure was selected for initial implementation of an access area, and progress in development of the microprocessor-based access area interface and the cable modem are described here. The design and development of a programmable microprocessor-based packet voice terminal processor, which handles voice protocols and communicates between the vocoder and the access network buffer and control processor, is also described. The overall system is being prepared for use in packet speech multiplexing experiments in the wideband network test bed.

A. System Architecture

The initial prototype of the voice terminal and access network system has been designed in detail, fabrication is complete, and testing has begun. During the final design review, some changes were made in the overall architecture which should simplify the hardware and the software. The principal change is elimination of the second processor for access control functions. Figure 8 shows the current functional split for the voice terminal. Both the terminal processor and the buffer and control processor are built around INTEL 8085 microprocessor integrated circuits.

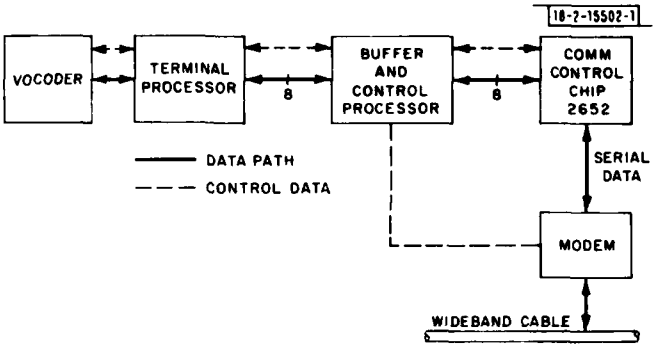


Fig. 8. Voice Terminal block diagram.

Most of the functions of the access control algorithm are implemented in the buffer-control processor. Some special-purpose hardware was added to handle time-critical functions.

A special card has been designed to aid in the debugging and testing of the 8085 microprocessor systems. The card works in conjunction with an operator's mini-console which allows control and monitoring of the processor-under-test. The card has demonstrated its utility in debugging the buffer-control system and should prove useful in testing of the terminal processor and overall testing of the integrated voice terminal. The board uses the CPU chip of the processor-under-test, but contains its own data and program memories. When testing is completed, the card may be removed without affecting the operation of the system.

B. Buffer Control Processor Architecture

The buffer-control processor along with the communication protocol control chip is constructed on a single 7-in. × 7-in. wirewrap board. The general architecture is shown in Fig. 9. The basic CPU is a 3-MHz INTEL 8085. It is used in conjunction with two special memory and I/O chips. The first is an 8755 EPROM with 2K bytes of program storage and two general-purpose I/O ports which are used for software control/monitoring of internal and external interfaces. The second special memory is the 8155 RAM. This device contains 256 bytes of read-write storage which is used as working space by the control program. The chip includes three I/O ports and an internal event timer. These parts are designed specifically for an 8085 minimum configuration and interface directly to the 8085 bus which time shares its lines between data and low-order address bits.

In order to use nonspecialized chips in this configuration, it is necessary to modify the bus to a more flexible configuration by demultiplexing the address and data information. The standard bus allows the option of connecting a number of other useful interface chips such as the 8257-5 DMA controller chip which provides four separate channels in and out of buffer memory. Each channel is dedicated to one of the four data transfers required to move data in and out of the

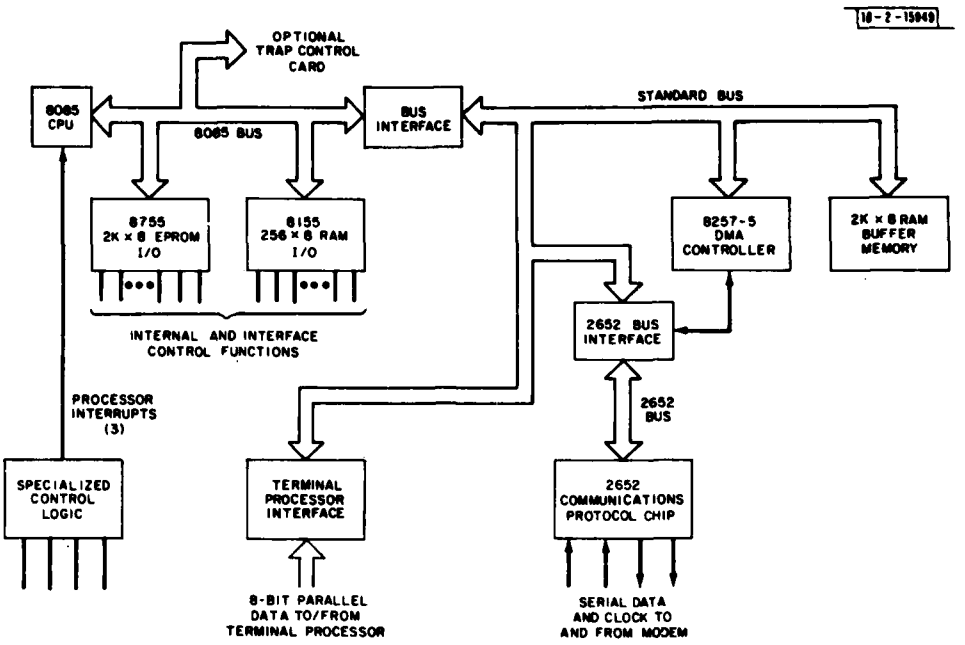


Fig. 9. Buffer Control Card block diagram.

buffer memory in each direction. The maximum transfer rate of the DMA chip is 750K bytes/sec (which leaves no time for the CPU). A 2K-byte RAM buffer is provided for storage of packets in transit. The standard bus is connected to the terminal processor through a bi-directional bus coupler (INTEL 8286). Transfers over this path are controlled by the DMA port which provides a standard set of "handshake" signals.

The Signetics 2652 communications controller chip (CCC) has two types of I/O connections. Certain control and status signals are brought out as separate lines. They are wired either to special hardware functions or some of the general I/O registers. The remainder, including the data input and output registers and some internal status/control registers, are connected on an internal bus. The standard bus is adapted to match the requirements of the CCC bus so that the processor can access the internal registers. In addition, this interface contains some special logic to allow the DMA to make transfers between the data registers of the CCC and the buffer memory.

Although most of the control functions with the buffer-control processor are handled in the CPU software, certain functions are implemented in hardware because of their time-critical nature. These functions either provide interrupts to the CPU or handle the necessary operations directly.

For example, the receive channel of the CCC is enabled automatically as soon as a signal is sensed by the modem. This allows the chip to be ready to recognize the leading synchronization frame which occurs about 10 μsec after the start of the signal. On the transmit side, special hardware is used to detect the end of the DMA transfer to the CCC. An interrupt is set which causes the program to start the end-of-message sequence before the chip runs out of data. The end-of-message signal must reach the CCC within about 15 bit times of the last transfer.

C. Buffer Control Software Structure

The buffer-control processor has four principal data transfer tasks. Each uses one channel of the DMA controller to minimize the processor load. The four tasks involve the transfer of data blocks (packets) in both directions between the buffer memory and the terminal processor and between the buffer memory and the CCC. One buffer is provided for transmit data and a minimum of three for receive data. The main features of the software tasks are described below.

1. Transmit: Terminal to Buffer

Whenever the transmit buffer is empty a "transmit buffer available" signal is sent to the terminal processor and a DMA channel is enabled. The terminal processor can fill the buffer whenever it has a packet to send. Almost no CPU action is required until the buffer is full, at which time a flag is set for the second transmit program segment.

2. Transmit: Buffer to Communication Controller Chip

When the transmit buffer is full, the program sets up a DMA transfer from the buffer to the CCC and then decides, based on the contention algorithm, whether to transmit immediately or wait. If the decision is to wait, the program monitors the cable for some interval and then makes a new decision. When a decision to transmit is made and the cable is not busy, a transmit signal is sent to the modem thereby turning on the transmitter. A start-of-message signal is sent to the CCC to start transmission of the synch frame, and the DMA transfer is enabled allowing the packet to be sent without further CPU intervention. While the packet is being sent, the CPU monitors the process looking for a collision signal from the modem. If a collision occurs, transmission stops and another attempt is made. If there is no collision, the CPU is interrupted at the end of the DMA transfer and end-of-message signal is sent to the CCC. This causes the CCC to transmit the checksum and end-of-message flag. The CPU can then disable the transmitter and mark the transmit buffer as empty.

3. Receive: Communications Controller Chip to Buffer

In the normal situation, a receive buffer is always available and DMA transfer to this buffer is set and enabled. As soon as the modem detects signal on the cable, the CCC receive channel is enabled. It detects the initial synchronization frame, and subsequent data bytes are transferred to buffer by the DMA. No CPU action is required until the end of the packet at which time the CPU is interrupted. The CPU immediately sets up a new receive buffer and enables the next DMA transfer.

4. Receive Buffer to Terminal

A background program monitors the full receive buffers. If a packet is received normally (no collision), then the destination address is checked. If the destination address matches the address (or one of the addresses) for which the terminal is looking, then the packet is sent to the terminal processor via another DMA transfer. At the completion of this transfer, the buffer becomes available again for receiving.

D. Trap Control Board Architecture

A second board, the trap control board (TCB), was designed and built as an aid in debugging the hardware and software. One of the problems in debugging a microprocessor-based system

is that the state of the system is stored in memory or in the processor and is not visible from the outside. The TCB allows the processor-under-test to be examined and its internal state and memory contents to be displayed. It works in conjunction with a control panel containing keys for entering command/data information and equipped with status indicators and an alphanumeric display. A block diagram of the TCB is shown in Fig. 10. The program and data memories are the same as used on the buffer-control subsystem. The 8279 keyboard/display interface chip handles all functions associated with the keyboard and digit displays. The chip provides signals for scanning the keyboard, detecting key closures, performs debouncing, and sets a flag when a valid keystroke is detected. In addition, it contains a buffer memory from which the display digits are refreshed.

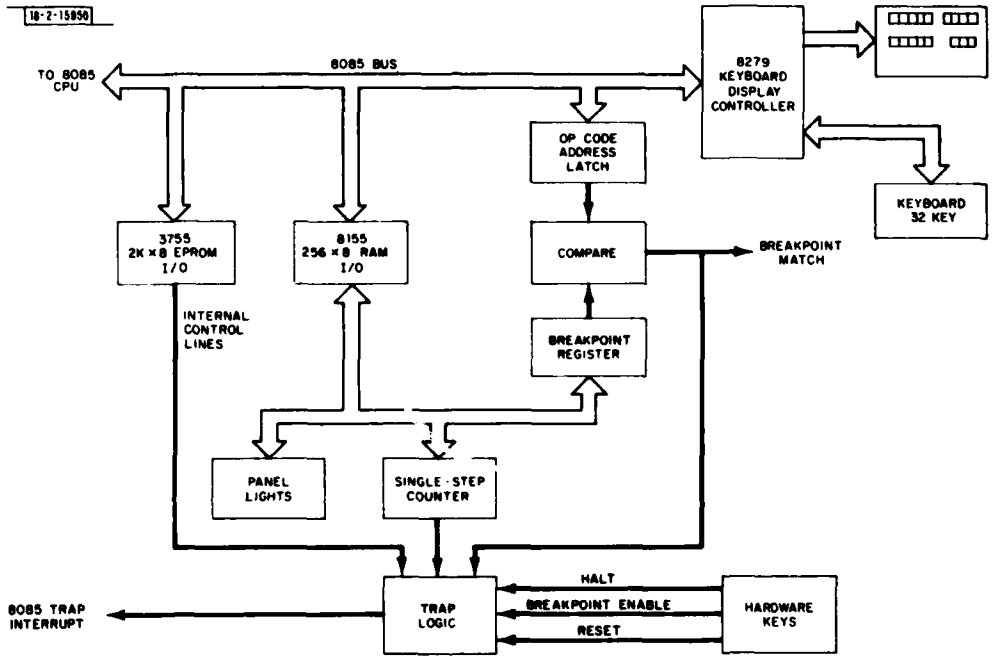


Fig. 10. TRAP Control Card block diagram.

The TCB does not contain its own CPU chip, but instead connects to the bus of the 8085 under test. The card, however, does contain program memory and data memory and controls the TRAP interrupt line of the CPU chip. The TRAP interrupt has higher priority than all other interrupts, is non-maskable, and also disables the interrupt system so that other interrupts are ignored until the TRAP program is completed. The TRAP interrupt forces the working program to a fixed location in program memory which contains a branch instruction to the program memory on the TCB. The program on the TCB saves the entire state of the CPU, displays the contents of the program counter and accumulator, and then enters a routine waiting for input from the control keyboard.

Two types of keys are provided on the keyboard: digit keys and command keys. Digit keys are buffered into numbers which will comprise the data or address fields of the commands.

A command key causes the program to branch to a routine which will implement the specified operation. There are two classes of commands. The first set displays or modifies the state of the system as stored when the TRAP interrupt occurred. Commands are available to read or set any memory location or I/O port, to enable or disable the interrupt system when control is returned to the main program, to load the breakpoint register, and to load a new program counter value to be used when the working program resumes. Since an image of the CPU active registers is formed in the memory of the TRAP card, any of these may be altered also. After command execution, control is returned to the routine which scans the keyboard.

Two commands (RUN and SINGLE-STEP) are available which return control to the working program. For both of these commands, the image of the CPU state (as modified) is returned to the CPU and control is passed to the location specified in the current image of the program counter. For the SINGLE-STEP command, a counter is set which causes another TRAP to occur after one instruction of the main program has been executed. The counter is loaded and enabled by the CPU several instructions before control is returned to the main program. The counter decrements on each CPU instruction fetch cycle, and a TRAP interrupt occurs when the counter reaches zero.

A breakpoint may be set at any program location. On each instruction cycle, the address of the first byte of the instruction is latched in a register. This address is compared with the contents of the breakpoint register which is loaded under program control. If a match occurs, and if the breakpoint system is enabled, a TRAP occurs and control is passed to the trap control program. The MATCH signal also provides a useful external trigger. In this mode, it provides an indication whenever a particular point in the program is reached. This has proved especially useful for capturing real-time events when used in conjunction with a logic analyzer. BREAKPOINT ENABLE enables the breakpoint match signal to cause a TRAP. This can be done while the program is running or after it has passed the breakpoint many times without stopping.

Several additional features of the TCB are enumerated below:

(1) LED panel lights are provided which indicate the state of the digit displays, the interrupt system, and the trap logic.

(2) The panel lights, the single-step counter, and the breakpoint register are all connected to a bus controlled by the RAM I/O ports.

(3) In addition to the keys which operate through the trap control board software, several keys operate directly on the trap logic. This means they can be used while the main program is running.

(4) HALT causes an immediate TRAP. Note that it does not stop the running of the CPU but only halts the execution of the main program. The state of the CPU is saved, and control is passed to the keyboard monitoring routine.

(5) RESET is the master system reset. It reinitializes the CPU and many of the peripheral circuits.

(6) The trap control card may be removed from the system without affecting the operation of the main program as long as the TRAP interrupt line is grounded.

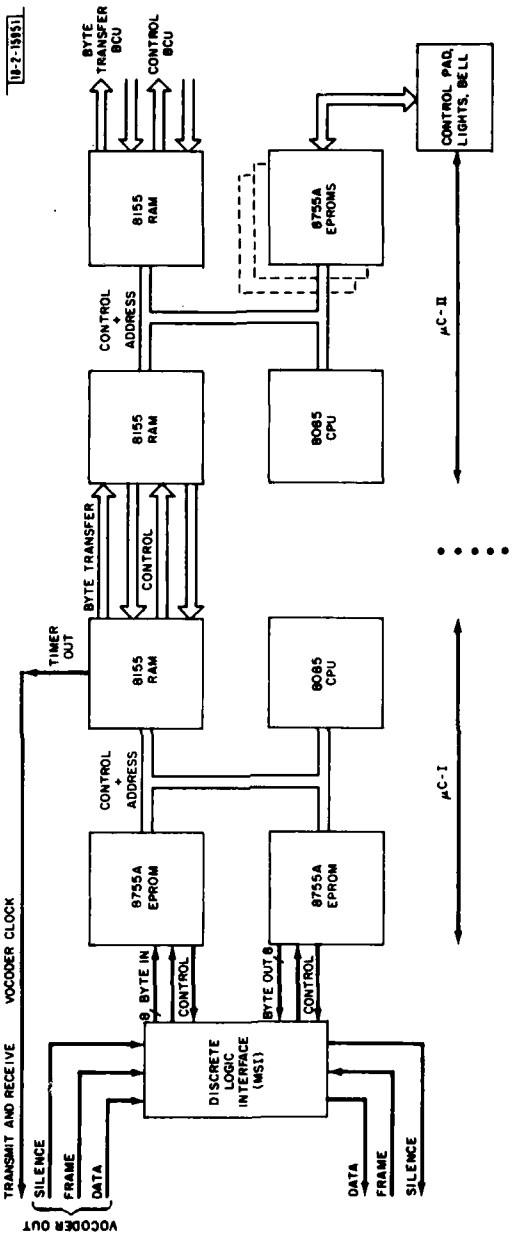


Fig.11. Terminal Processor.

E. Terminal Processor

In the previous Semiannual Technical Summary, we discussed the need for implementing a flexible access area control protocol as well as the requirement for operation with a variety of voice coding devices (from 2.4 to 64 kbps). Additional flexibility for driving either a hardwired access area modem or a packet radio modem was also described.

In order to satisfy these requirements, we have designed a processor geometry which uses two minimum-configuration INTEL 8085 8-bit microprocessor chip sets. As shown in Fig. 11, one chip set (μC-I) is programmed to interface with the voice coding device and communicates with the second chip set (μC-II) which runs the control protocol and communicates with the buffer-control processor. Assigning the functions in this way allows μC-I to deal with the real-time flow of serial data to and from the voice device, while μC-II can be structured to run a network voice protocol.

In more detail, μC-I consists of a basic 8085 CPU chip, two 8755 chips featuring 2K bytes of EPROM and two I/O ports each, and an 8155 256-byte RAM having three I/O ports and an on-chip counter-divider. This μC must accept single-byte data from a serial/parallel converter as well as sense beginning-of-frame and silence markers transmitted to the processor from the speech coder. As part of the initialization sequence, this section will automatically determine vocoder rate by examination of the frame structure and adjust its operating parameters to run the steady-state program. This subsystem also drives the voice coder receiver through a byte parallel/serial converter, senses frame markers provided by the vocoder receiver, and transmits silence markers to the receiver when data supplied by the buffer control processor has been exhausted. The voice coder routine operates at a high interrupt priority, while the interaction with μC-II takes place at a lower priority level.

μC-II consists of a basic 8085 CPU chip, two 8155 chips, and an expandable set of 8755 chips (1 to 3) sufficient to store NVP-like firmware. This section must communicate in packet format (byte-at-a-time transfers) with a buffer control or packet radio device and interface with μC-I. In a background mode, control pad inputs are monitored and indicator lights and bells are driven.

At present, initial operating code for μC-II has been written so that this half of the terminal processor can be operated with the buffer-control unit and modem to provide a functional connection to the access area. This basic software involves a control input handler, vocoder data handler, and a packet handler. It is planned to drive μC-II from an LDSP machine programmed to provide the voice digitization function for initial testing and early narrowband voice experiments. Compact low-cost hardware will replace the LDSP when it becomes available. An adjunct TCB will be used for startup testing. When this setup is operational, μC-I will be debugged and connected so that a complete terminal processor capability will be available for further experimentation.

F. Cable Modem

A single coaxial cable modem using baseband signalling was designed and tested using both pulse generators and the LDSP as a signal source. The cable transceiver, Fig. 12, was fashioned after the one used for CHAOSNET† by the M.I.T. Artificial Intelligence Laboratory. The transceiver consists of circuits which enable coupling of baseband signals between the buffer control processor and the cable while providing electrical isolation between them. The isolation

† T. Knight, J. Holloway, "Design of a CHAOSNET Transceiver," private communication, 6 April 1978.

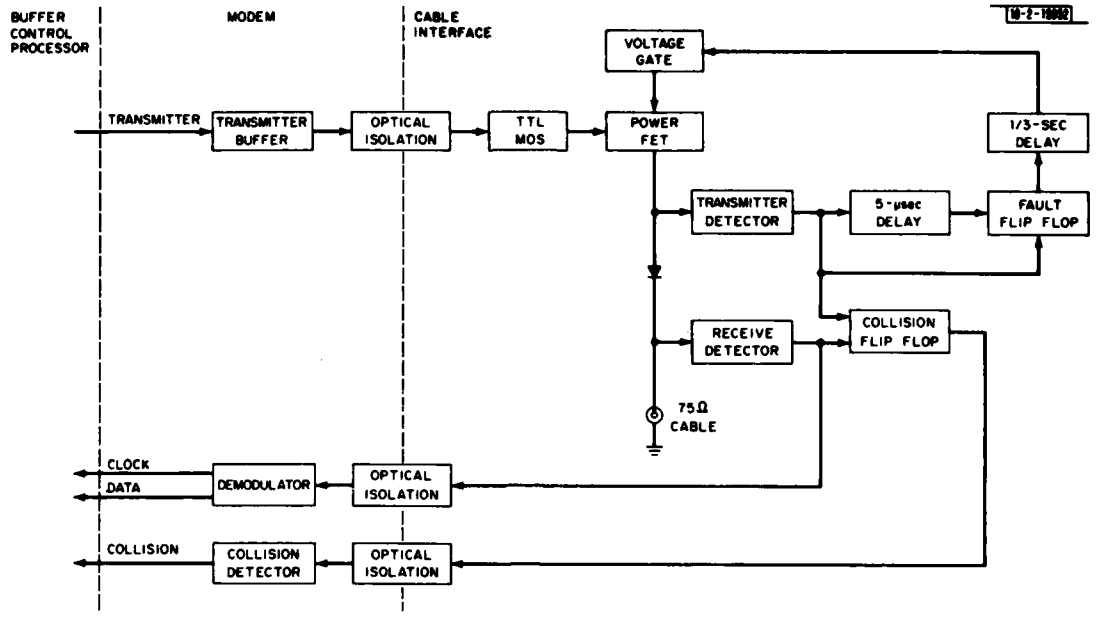


Fig. 12. Cable Transceiver.

is effected by virtue of optical couplers across the transceiver/buffer control processor interface boundary and the use of separate power supplies. Isolation between the cable ground and the local-processor ground is necessary to prevent potentially large currents from flowing in the cable due to the use of different power distribution systems employed in other processors attached to the cable.

On the transmitter side of the transceiver, the modulated cable signal is converted from a TTL to a MOS level which drives the VMOS power FET which in turn drives the cable through a series diode. The diode provides isolation of the cable from the driver circuit capacitance when the cable is not being driven. The receiver circuitry consists of a differential receiver which compares the cable voltage with a settable threshold. At the rising edge of any transmitted data, the state of the received data on the cable is examined in a collision flip flop. If the cable already has a signal present, a collision condition is detected and this is reported back to the buffer-control processor for appropriate action. A further malfunction is detected by fault circuitry which senses a condition where the buffer-control processor attempts to send more than 5 μsec of signal without transitions. In the event that such a situation is detected, a timing circuit disables the transmitter for 1/3 of a second. This fact is not reported back to the buffer-control processor and would be considered lost data by the intended receiver by virtue of erroneous parity or other critical packet information.

Tests conducted on the transceiver consisted of a pulse generator or an LDSP-based data source driving an RG59B/U coaxial cable of varying length with transceivers situated at various points and connected by coax "T" junctions along a 1050-ft cable span. The cable has a characteristic impedance (Z_0) of 75 Ω and is terminated at either end. Pulse tests of square waves up to 5 MHz indicated successful detection from end to end through the transceiver, but did not test recovery-associated problems with random test patterns. For this purpose, an experimental

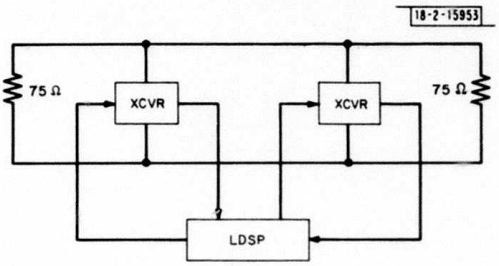


Fig. 13. Experimental Test Bed.

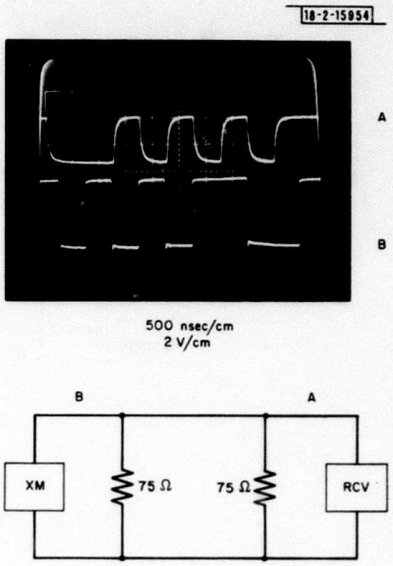


Fig. 14. Waveforms – Experimental Test Bed.

test bed, Fig. 13, was established using an LDSP to generate and receive random 16-bit data words. Programs have been written to enable the LDSP to insert Manchester encoded data at rates as high as 2 Mbps on the cable. Figure 14 depicts a typical set of waveforms using the experimental test bed.

The data signalling scheme used for the coaxial cable is the Manchester encoding/decoding scheme whereby each signalling interval consists of transmitting the complement of the data bit during the first half and the true sense of the data bit during the second half as shown in Fig. 15. This insures that there is at least one bit transition per data bit interval at the midpoint of the interval. Additional transitions between bit intervals may or may not occur depending upon the data bit pattern. This data modulation scheme simplifies data transient recovery problems associated with baseband cable signalling schemes although it requires a maximum transmission

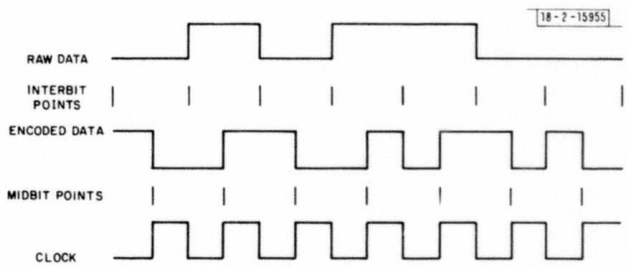


Fig. 15. Manchester encoding.

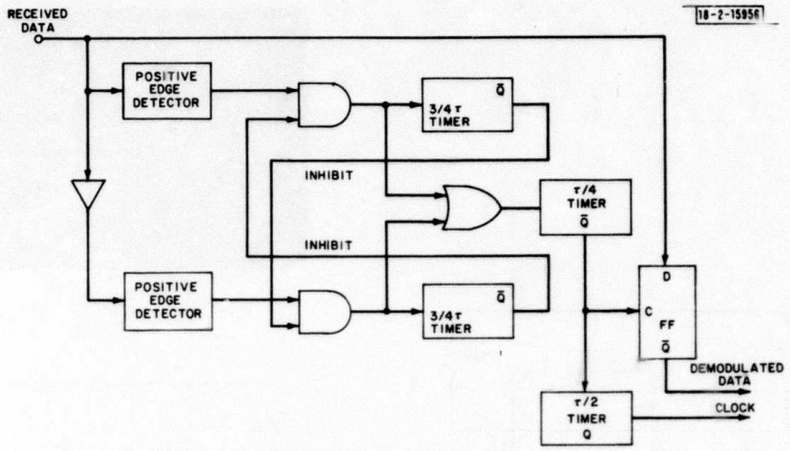


Fig. 16. Demodulator block diagram.

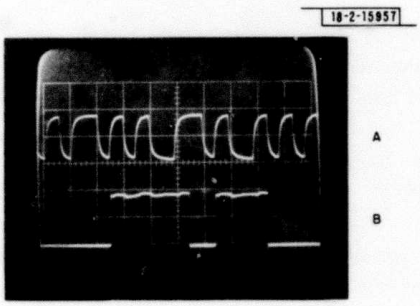
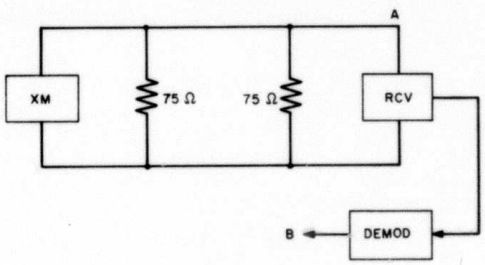


Fig. 17. Demodulator waveforms.



rate of twice the data bit rate. A demodulator has been designed to recover asynchronously the data pattern from the modulated bit stream as detected by the cable transceiver. A block diagram of the demodulator is shown in Fig. 16. The logic employs positive and negative edge detectors which sense either 1-0 or 0-1 data transitions. Since data transitions can occur at either mid-bit or inter-bit data points, some method of bit synchronization must be employed to correctly identify the mid-bit transitions as a reference point from which to accurately strobe the ensuing true data sense half-interval. The mechanism employed in the demodulator is to note the a priori fact that once having correctly determined the first mid-bit point the mid-bit point of the next data bit cannot occur in a time shorter than the reciprocal of the data rate. Thus, a data inhibit initiated upon correct detection of a presumed mid-bit transition and lasting for 3/4 of this time will prevent the edge detector from interpreting the next inter-bit transition (should the data dictate that it occur) as a mid-bit transition. In order to assure proper synchronization using this scheme, at least one 1-0 or 0-1 data transition is necessary. This fact is assured by virtue of the mode of data transmission to be employed by the buffer-control processor. Its protocol generates a string of "ones" followed by a "0" and six "ones" as a synchronization word. This assures at least two transitions (0-1, 1-0) at the start of every packet to correctly synchronize the demodulator. Figure 17 shows a received bit pattern and its demodulated data stream. The demodulator has been tested successfully up to 2 Mbps over 1050 ft of cable.

VI. SATELLITE AND INTERNETTED CONFERENCING

Lincoln Laboratory has provided hardware and software to support voice conferencing experiments as parts of the ARPA Atlantic Packet Satellite Experiment and the ARPA Internet Program. Four sets of speech hardware made up of a linear predictive vocoder and interface equipment are in place and operational. Three of the sets are connected to PDP-11 computers at Bolt Beranek and Newman, Inc. (BBN) in Cambridge, Massachusetts; University College (UCL) in London, England; and the Norwegian Defense Research Establishment (NDRE) in Kjeller, Norway. These computers also serve as gateways between the Atlantic Packet Satellite Network (SATNET) and ARPANET. The fourth set is connected to an LSI-11 computer at COMSAT Corporation in Washington, D.C., that can be connected to SATNET through a small earth station at COMSAT Laboratories in Clarksburg, Maryland.

Voice conferencing among BBN, UCL, and NDRE was demonstrated in May 1978 using an early version of SATNET that used a fixed time-division multiple-access (FTDMA) technique to share the satellite channel. Since that time, the conferencing software has been modified to use the broadcast stream capability offered in the current SATNET which uses a priority-oriented demand assignment (PODA) technique to share the channel. The stream represents a reservation of satellite channel capacity that can be shared among the sites participating in a conference. Streams are intended to provide a packet communication service well suited to supporting speech by offering less overall delay than datagram service as well as a guaranteed average data rate. Modification of the software to use streams involved major changes to accommodate a whole new protocol for communication between host computers and SATNET as well as changes directly related to the use of streams. Conferencing using streams has been operational since June 1979, but overall delays are greater than had been expected due to a greater than expected delay dispersion, and further work is indicated to determine the cause of the large delay dispersion.

Current work has been directed toward the realization of an internetted conferencing capability between SATNET and ARPANET. The approach taken was to interconnect two existing conferencing systems with as little change as possible to each. The result required a specialized program to run in the gateway between the networks. The specialized gateway software is now operational, and internetted conferencing was demonstrated at the ARPA Internet Meeting in London in September 1979. The present capability is cumbersome to set up, is not robust, suffers from excessive loss of speech due to delay dispersion, and has some problems with loss of vocoder synchronization. Work continues to correct these difficulties.

The following section describes the internetted conferencing capability in more detail. Conferencing in both SATNET and ARPANET makes use of a simple broadcast protocol with speaker selection determined by control signals generated by pushing buttons on control boxes at each speaker's site. The simple broadcast protocol specifies that one participant at a time may speak to the conference. A would-be speaker indicates a desire to talk by pushing a WANT-TO-TALK button. When the control algorithm decides to allow him to talk, he is signalled that he now has the "floor" by means of an indicator light on his control box. When he finishes talking, he pushes a DONE-TALKING button to tell the controller that it may switch to another speaker. Other lights provide information about the state of the conference and do not have common interpretations between SATNET and ARPANET.

In order for an internetted conference to take place, it is necessary either that the same speech encoding technique be used in both nets or that some translation between techniques be carried out. For the present experiments, it was decided to use the same technique since no translation between the techniques available in ARPANET and the hardware vocoders in SATNET would give acceptable speech quality. Since vocoders at the ARPANET sites are realized by software in signal-processing computers, it was relatively simple to implement the same vocoding algorithm used in the SATNET hardware. Programs to follow that algorithm were written at Lincoln Laboratory (LL) and the Information Sciences Institute (ISI) of the University of Southern California at Marina del Rey, California. The SATNET vocoder became the third type of LPC speech encoding available in the ARPANET. The representation of the encoded speech parcels was chosen to be the same in ARPANET and SATNET, but overall packet formats are different.

A. SATNET Conferencing

SATNET conferencing makes use of the broadcast (multi-address) capability of SATNET to send both speech and control packets. The conference control programs (CCPs) running at each site use a predefined group address to send packets to all CCPs on the net (including the sender). Control packets are sent as datagrams. Speech packets are sent using a SATNET stream that is requested by one of the CCPs during conference initialization. The stream parameters allow a packet to be sent every 200 msec. Each packet contains ten frames of vocoded speech plus a time stamp as well as other header information.

Control is distributed in a SATNET conference with any pair of CCPs being capable of starting or continuing a conference. When a CCP enters a conference, it begins by telling the SATNET interface software that it wants to receive any packets addressed to the predefined conference group address. (In a SATNET environment capable of supporting more than one conference, it would be necessary to have some prior protocol exchanges that decided on a group address for the particular conference to be entered.) The second step is to send a HELLO message notifying any other CCPs that a new CCP is ready to join the conference. The CCP will continue to

send HELLO messages at regular intervals, until it either hears a HELLO from another CCP or receives a conference stream message. As soon as two or more CCPs have exchanged HELLOs, a conference stream is set up by one of the CCPs (the one with the lowest host number). The stream is set up by sending a CREATE command to SATNET with parameters specifying the stream interval (average time between packets), packet length, and priority. After about 9 sec, SATNET responds indicating that the stream is ready, and the CCP that requested it begins sending IDLING messages on the stream and assumes MASTER status for the conference. MASTER status moves from CCP to CCP as the location of the speaker changes during the conference. Receipt of any stream message is interpreted by all CCPs as evidence of the existence of a conference. They stop sending HELLO messages, turn on an indicator light to show that the conference is up and ready, begin playing out received speech (if any), and wait for button pushes from the human participant to indicate his desire to talk, etc.

When a participant pushes his WANT-TO-TALK (WTT) button a WTT control message is made up and broadcast to all CCPs. On receiving a WTT message, each CCP makes an appropriate entry in the WTT list which is a queue of all CCPs that have indicated a desire to talk. Each CCP can appear only once in the WTT list, so a second button push has no effect. However, pushing the DONE-TALKING (DONE) button while one is on the WTT list will take that participant off the list. When the current speaker pushes his DONE button, his CCP examines its WTT list and, if it finds it non-empty, it sends a HAND-OFF message containing its WTT list in the stream. Actually, it sends a pair of HAND-OFF messages to increase the probability of at least one being received.

The HAND-OFF message is received and examined by all CCPs. The WTT list in the HAND-OFF message becomes the official conference queue. The CCP finding itself at the head of the queue becomes the new MASTER, turns on the YOU-HAVE-THE-FLOOR light for its participant, and begins sending speech messages on the stream. The other CCPs examine the agreement between the new WTT list and the desired status of their participants and send appropriate control messages to bring the list into the correct state. This procedure is required because any control messages that were in flight at the same time as the HAND-OFF message will not be taken into account in the HAND-OFF.

If the CCPs detect a string of missing stream messages, they assume that the MASTER CCP or its SATNET connection has failed and will enter a recovery mode in which HELLOs will be exchanged and a new MASTER established. Failure of the stream is detected by the receipt of a message with an error code indicating "no such stream" when attempting to send a stream message. In that case, a STREAM-LOST message is sent to all CCPs and they enter a different recovery mode in which a new stream is requested.

B. ARPANET Conferencing

Conferencing in ARPANET differs from that in SATNET in two major ways. Since there is no multi-address delivery capability in ARPANET, separate copies of messages must be sent to each site, and control is centralized in a program called the CHAIRMAN that runs at some site. Each participant has a local conference controller (LCC) that handles conference protocol for him. An LCC enters a conference by exchanging protocol messages with the CHAIRMAN. WANT-TO-TALK and DONE-TALKING messages go only to the CHAIRMAN. It maintains the WTT list and decides when to switch speakers. When a new speaker is to talk, the CHAIRMAN sends a LISTEN-TO message to all listeners and a SPEAK-TO-THE-FOLLOWING-LIST message

to the new speaker. The list associated with the latter message tells the speaker's LCC to which sites copies of the speech packets are to be sent.

The CHAIRMAN has the ability to arbitrarily stop a speaker and switch to another on the basis of time spent talking or special status of the new speaker (e.g., an interruption by some participant filling the role of a human chairman). This capability is not present in the SATNET, but it could have been provided there if desired. It has been omitted from the internetted conferencing capability to avoid augmenting SATNET CCPs to handle it.

C. Internetted Conferencing

Figure 18 shows the configuration used in the internet conferencing experiments on SATNET CCPs run at NDRE in Norway and at UCL in England, and on ARPANET LCCs run at ISI in California and LL in Massachusetts. The LCC at LL can support up to four talkers, the other sites support one each. The gateway at BBN performs the functions of a CCP on SATNET and both an LCC and a CHAIRMAN on the ARPANET as indicated schematically in Fig. 19. The conferencing gateway programs run under the ELF operating system in the PDP-11/40 at BBN and coexist with the Internet Datagram gateway that handles nonspeech internet traffic between ARPANET and SATNET. However, the amount of such traffic that can be handled while a conference is in progress is severely limited due to the heavy load posed by the conference.

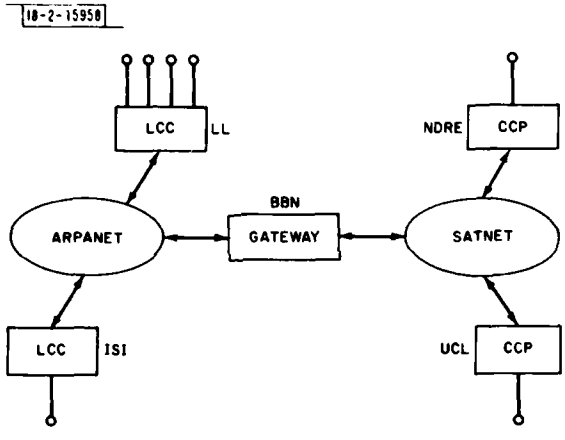


Fig. 18. Internet conferencing configuration.

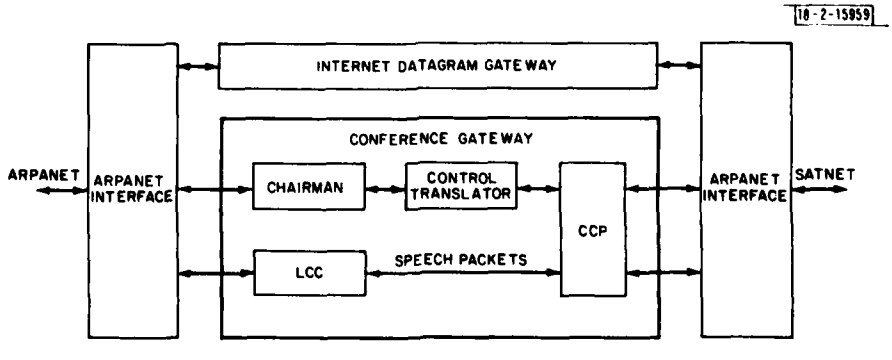


Fig. 19. Internet conferencing gateway.

When the conference speaker is at NDRE or UCL, speech packets received by the CCP in the gateway are passed through to the LCC. The LCC has been told by the CHAIRMAN to speak to ISI and LL. It therefore sends out on the ARPANET two copies of each SATNET packet received from the CCP. One goes to ISI, the other to LL. If the conference speaker was at ISI, the CHAIRMAN would have instructed ISI to send speech to LL and the LCC in the gateway at BBN. The LCC would then pass the packets through to the CCP which would send them on the SATNET stream. They would be received simultaneously by UCL, NDRE, and the BBN CCP that sent them. In this direction, a smoothing buffer in the gateway is used to match the random delay characteristics of the ARPANET to the regularly scheduled departure times for stream packets. The present buffer strategy is very simple. When a change to a new ARPANET speaker occurs, the gateway waits until two speech packets have arrived from ARPANET before sending any stream packets on SATNET. Thereafter, stream packets are sent in each stream interval. If no ARPANET packet is available, a special SILENCE packet is sent on the stream to keep the recovery mechanism in the CCPs from being triggered by an absence of stream messages. The LCCs on ARPANET do not transmit speech packets during silent periods, but SATNET assumes continuous transmission of speech packets. If packets are abnormally delayed in ARPANET such that stream slots go by that should have carried speech packets, the buffer in the gateway will tend to overflow when the late packets finally arrive. The present algorithm will discard packets if the number in the buffer exceeds three.

When the gateway is forwarding packets from SATNET to ARPANET, stream packets marked as carrying SILENCE are not forwarded to avoid unnecessary load on ARPANET. The SILENCE packets on SATNET do not represent extra load because SATNET schedules the time slots for transmission of stream packets whether or not actual packets are available. If no packets are transmitted for a relatively long time (2 min.), SATNET will close out the stream slot and recover the capacity for other use.

When an ARPANET conference participant pushes his WANT-TO-TALK button, a WTT control message is sent to the CHAIRMAN program in the gateway. This ARPANET WTT message is translated into a SATNET WTT message and broadcast to the CCPs. Only when the message has been successfully received by the CCP in the gateway is the request considered to be in effect for the internetted conference. Since the WANT-TO-TALK list maintained by the CCPs has only one position for each site, and all ARPANET talkers appear in SATNET to be on different extensions at the gateway site, it is necessary for the control translater to hold ARPANET WTT requests if the gateway site is already on the WTT list in SATNET and transmit them later when the gateway becomes the speaker site. This procedure gives some bias in favor of SATNET talkers in getting the floor in an internetted conference, but the bias is not important with only two other SATNET sites.

The technique used to effect ARPANET/SATNET conferencing by interconnecting two rather different conferencing systems is not attractive as a general approach to internetted conferencing since it requires specialized actions on the part of the gateway. A more satisfactory general approach requires the implementation of new network voice and stream handling protocols. A brief discussion of these protocols is given in the next section. Such protocols are currently being developed for application in the wideband satellite experiments.

VII. PACKET VOICE NETWORK PROTOCOL DEVELOPMENT

Two new protocols are being developed to support speech communications in packet networks. One is a new Network Voice Protocol called NVP-II. The other is an Internet Stream Protocol

called ST intended to support the efficient delivery of streams of packets to either single or multiple destinations. NVP-II handles the actual packetization of speech including time-stamping as well as the negotiations of vocoder types, etc., associated with setting up a point-to-point call or a voice conference. NVP-II also deals with the dynamic control of voice conferences. ST provides the transport mechanism that delivers NVP-II packets to their destinations in an internet environment. The focus of ST is on providing the guaranteed data rate required for successful speech communication as well as the multi-address delivery required for conferencing. Both protocols are aimed at application in the internet environment made up of the local access net and the wideband satellite network. They can have application in other networks such as ARPANET, SATNET, and the Packet Radio Net, but their advantages over earlier protocols will not be marked in those environments where the capacity is not sufficient to support significant speech communication experiments.

The development of these new protocols has involved interaction among a number of people in the ARPA Packet Speech and Internet Programs. The primary responsibility for specifying NVP-II has been carried by ISI and SRI. Lincoln Laboratory has assumed the responsibility for defining the ST protocol. That effort has resulted in a presentation that was made at the Internet Meeting in London in September 1979 as well as an Internet Experiment Note (IEN No. 119) that describes the protocol in some detail. Although further work on the protocol document remains to be done, the protocol was tentatively accepted at the meeting. A detailed discussion of the protocol will be presented in the next report in this series.

VIII. ACCESS AREA TRAFFIC EMULATION MODULE DESIGN

Traffic emulation will be essential for experiments on the wideband network, both to exercise and check out various parts of the network as they become operational and to support advanced systems experiments in areas such as multi-user packet speech and demand assignment multiple access. The design of a Traffic Emulation Module (TEM) to be connected to the local access net (Sec. V) is described here. This module will allow performance measurements on the local net, and will exercise mini-concentrator throughput functions, as well as providing traffic on the satellite channel. Traffic emulation will also be provided at other levels in the wideband net; for example, PSAT traffic emulation will allow measurements on the satellite network independent of concentrators and terminals.

The initial design of the local-net TEM attempts to be completely flexible in order to accommodate future requirements. The TEM will be interfaced directly to the local net and will be able to operate in many modes. In one mode, it will emulate the output from N independent voice terminals. Each of these emulated terminals may be assigned any combination of vocoder type, bit rate, and destination address. The traffic on the local-access net created by the TEM in this mode will be as close as possible to the real traffic which would be generated by the same number of voice terminals. In another mode, the TEM will multiplex the packets from N simulated terminals into a large packet to send to the mini-concentrator and then over the wideband satellite network (PSATNET). In a third mode, it will carry out the protocol required to set up connections at the various levels. The potential need for other modes of operation is recognized, and it is expected that new requirements will arise as the system develops.

The TEM will also provide a checkout measurement facility for the wideband net. Measurements will be performed at several levels. When the TEM is initialized, a run-time parameter will specify what level or section of the system should be evaluated. As the system is checked

out section by section, the corresponding measurement levels will be used less frequently but will always be available for use as network diagnostics.

It is unlikely that one main program can contain all the modes of operation and levels of measurements required. The TEM will be a family of programs, differing from each other in their mode of operation and/or their level of measurements.

The Traffic Emulator has been designed to meet the following goals:

Provide traffic to:

validate the local-access net

create background traffic for multi-user voice experiments using limited numbers of actual voice terminals

exercise mini-concentrator functions

test the PSATNET with the equivalent of multi-terminal traffic

Provide a means to validate all levels of protocol

Provide maximum flexibility for tests and experiments

Provide run-time options to configure these experiments

Provide statistics to evaluate performance at all levels

A. Traffic Model

The traffic model used will follow standard models for speech statistics. The voice traffic is modeled by a Poisson call arrival process of rate λ and exponentially distributed call holding times with mean μ^{-1}. The TEM will have the capacity to handle S calls at a time, with blocking. Off-hook callers will be assumed to alternate between talkspurt and silence with exponentially distributed talkspurt and silence duration with means α and β, respectively. Packets will be transmitted only during talkspurts. The caller population assumed to be large enough so that the call arrival rate is independent of the current number of calls. The maximum number of calls S will be a run-time parameter subject to some constraints. The run-time parameter S can never exceed the number of phones TEM can emulate in real time. The total bit rate produced by all the off-hook phones in talkspurt cannot exceed the available capacity on the local-access net. The available capacity, which is the total capacity of the access net minus the amount used by real traffic, will vary. If a call is received which exceeds the S, the call is blocked. If a call is received and the bit rate assigned it would cause TEM to exceed available network capacity, the call can either be blocked or the bit rates assigned it or other emulated phones can be lowered so that the available capacity is not exceeded. If during a run, real network activity increases and TEM is exceeding network capacity, new calls can be blocked and/or the bit rate of current calls reduced.

After the system has been properly initialized, the time T until the next call is randomly determined from the call arrival table. Then the arrival of the new call is simulated. A call duration time is randomly assigned using the call holding time table. Using the talkspurt/silence statistics, it is determined what state the call begins in – speech or silence – and the duration of this initial state. A bit rate and destination are also assigned to the call. Counters are set to time the state duration and the total call duration. When the state-duration clock runs out, the phone's state will be switched from silence to speech or vice versa. A new state duration is

randomly picked, and the state-duration clock reset. When the call-duration clock times out, the phone is considered to have hung up. During the time that an off-hook phone is in talkspurt, messages of the correct length and frequency to simulate the phone's vocoder type and assigned bit rate are sent to the Local-Access Net. During the run, performance statistics are gathered and stored off line for later evaluation.

B. *Equipment Configuration*

The equipment used to implement TEM will include a Lincoln Digital Signal Processor and two special interface boxes connecting the LDSP to the Local-Access Net. Experiment control and user interface will be via a general-purpose (GP) PDP-11 computer interfaced to the LDSP. Figure 20 shows the basic equipment configuration.

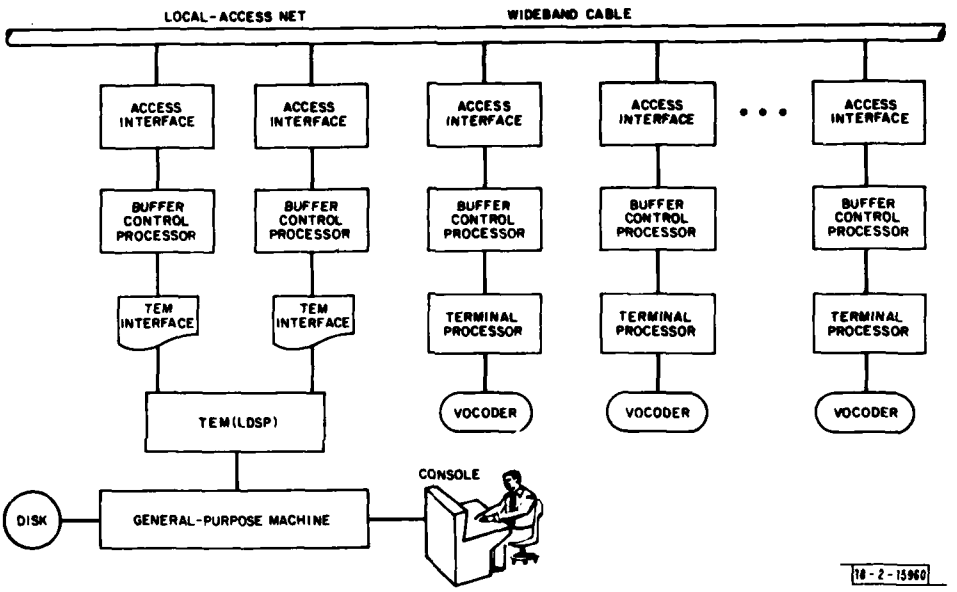


Fig. 20. Traffic Emulation Module Configuration.

The LDSP is a programmable signal processor with a 50-nsec cycle time. It will handle the fast interactive real-time processing and the exchange of packets with the Access Net. The LDSP does not have a convenient user interface for an experimenter, and it also lacks the storage capacity to hold a significant amount of measurement data. The program implemented on the GP machine will interact with the experimenter, take the measurement data from the LDSP and store it off line, and provide other low-level support for the LDSP. A program currently exists on the Group's PDP-11/45 which performs a support function similar to that envisioned for the GP program section of the TEM. This program will be adapted to provide the basic GP program and will be run on either the PDP-11/45 or the PDP-11/70.

The computational and I/O load on the GP machine will vary with the mode of operation. In situations where the number of emulated terminals must be maximized, the work done in the LDSP for each phone must be kept to a minimum. Therefore, the GP machine will perform the calculations on events which happen infrequently, mainly the call arrival processing, and pass

this information to the LDSP. This will conserve valuable program and data memory in the LDSP. However, in several anticipated modes of testing where the number of off-hook phones is small, or is held constant and minimal measurement facilities are needed, there will be little or no load on the GP machine once the experiment has started.

1. GP Machine Tasks

The GP program will perform all initialization tasks. It will query the experimenter for the desired values of all the run-time parameters, make up the initial state of the switchboard, load the LDSP, initialize it, and start it. This is all done under keyboard control.

To initialize, the mean call arrival rate λ and the mean call holding time are used to establish the average number of off-hook phones. The initial switchboard table has this number of phones off hook. Using the appropriate tables, these phones are assigned the necessary values: bit rate, call duration, speech/silence, etc., and the settings for their state and duration clocks are determined. This information is then sent to the LDSP to initialize it. The GP program maintains a copy of the switchboard table in its memory. The simulation of a new call and the timing of total call durations will be done in the GP program since these events occur relatively infrequently. The LDSP program maintains the state clock and handles the changes between silence and speech.

Information passes between the two machines throughout the running of the test. The GP program will tell the LDSP about new calls and the termination of old calls. The LDSP passes statistics information back to the GP program, which then writes the statistics out to the disk for later processing.

To lessen the amount of interaction between the GP machine program and the LDSP, some initial processing of the statistics may be done in the LDSP. The resulting data will be passed to the GP machine for storage on the disk. At a later time, another program running on a GP machine will retrieve the data and produce statistics, graphs, and histograms as required.

If desired, the GP program could inspect the data coming from the LDSP and alert the experimenter to unusual occurrences. It will also be possible to vary parameters while a test is running.

2. LDSP Tasks

The LDSP program has several basic tasks:

(a) Maintain the status clocks for the off-hook phones. Reverse the talkspurt/silence state for a phone when its clock times out. Pick a new duration and reset the clock.

(b) Determine in each time frame how many speech packets of each size should be sent. Construct a list of packets to send and times when they should be sent.

(c) Format necessary protocol packets and put on the list to send.

(d) Send packets to the access net.

An interrupt time of 20 msec is proposed to process tasks (a), (b), and (c). The actual sending of messages, Task (d), can be handled by a separate interrupt which is set according to the desired time interval between messages.

(e) Collect statistics.

(f) Exchange data with GP machine.

(g) Act on new data from GP machine.

Statistics are gathered as the program runs. Information on messages received/sent, delays noted, etc., is stored in the LDSP memory. The LDSP and the GP program will exchange information at an as yet unspecified rate. The LDSP will then clear its statistics buffers and act on any information received.

C. Packet Transmission on the Local-Access Net

When the TEM is operating in a mode where it is simulating N independent terminals, the traffic it produces on the access net must look like the traffic from N independent terminals. Terminals on the network with a packet to send first listen and if the network is judged free, they use a probabilistic algorithm to decide whether or not to send their packet in the current time slot. A collision occurs when two or more terminals send at the same time. A terminal detects that its packet has collided by listening to the network and noting that what it hears is not what it sent. When a collision occurs, the terminals involved resend their packets at some time in the future.

When emulating N terminals, the TEM cannot send all the traffic in a given time period as one giant packet. The appearance of a giant packet will artificially hold off the real terminals. The only chance of a collision with this giant packet is from a voice terminal which started transmitting at the same time the TEM did. (In a multiplexing mode of operation, a giant packet will be the desired output.)

If TEM feeds its output to the Local-Access Net one packet at a time, some may collide with real voice traffic, but this is still not realistic. N emulated voice terminals should cause collisions not only with real voice terminals but also with each other. When a phone comes off hook, it can randomly be assigned a time at which it can be expected to send its first packet. This is equivalent to the time when its imaginary vocoder would have had its first packet ready. Another packet can be expected from this phone every frame time while the phone is in talkspurt. When this is done for all phones and then combined with information on the bit rate and packet size required, it will be possible to determine when collisions should occur.

1. Collision Emulation

Now the TEM must emulate a collision on the local-access net. One possibility which was considered was for TEM to make up a special short "collision" packet and send it to the local-access net. TEM could then decide randomly when the colliding simulated terminals should retry. These retries may collide again either with each other or with some new site, etc. This quickly becomes a very complex tree and probably could not be implemented in real time. Also, when a packet (real or "collision") actually does collide with a packet from a voice terminal, the local-access net will go through its retry mechanism, hold off the TEM, and destroy the TEM's timing and simulation. Having a special interface to the local-access net which never retries itself but simply notifies the TEM of all collisions would drastically change the timing. The conclusion is that a collision between two terminals cannot be adequately emulated in a machine with only one network connection.

Therefore, the TEM design includes two interfaces between the LDSP and the local-access net. Now, when the TEM notes that two of its emulated phones have packets ready to go "at about the same time," it sends one packet on each connection "at about the same time." There is no way to guarantee creating a collision. The two connections are not synchronized. If one starts slightly ahead of the other, the second one may realize that the channel is busy and hold off. Also, each emulated off-hook phone will have an adaptive traffic estimation number associated with it and the local-access net will use this number to determine when to send. Since this is essentially what would happen at two independent terminals, the results should be quite realistic. When a collision does occur, the randomized retry mechanism for the two network interfaces will take over.

The local-access net connections to the LDSP interface run a program which differs from the program run in the connection to a voice terminal in two ways. A connection supporting a voice terminal maintains an adaptive traffic estimation number for the phone it is supporting. The TEM is "supporting" many phones and each needs its individual adaptive traffic estimation number. Since there is no room in the local access net connection to store all these numbers, the LDSP will provide the storage. Each packet sent to the access net by the TEM will have the phone's adaptive traffic estimation number in a predefined byte in the header. The access net will use this number to schedule the packet and place the updated value back in the header before sending the packet.

In addition, the local-access net will pass back to the TEM all successfully received traffic rather than just traffic addressed to the TEM. In this way, the TEM can extract each updated adaptive traffic estimation value and save it for the phone's next transmission. This special mode also facilitates the gathering of statistics on network traffic.

2. Interface to the Access Net

The local access net connection designed for the real speech terminals is not compatible with the LDSP. A new interface box must be provided. This box will take 16-bit words from the LDSP, split them into 8-bit bytes to hand to the access net connection and vice versa. The interface will manage the handshake protocol with the access net connection. It will control the speed of the data flow so that the LDSP does not send packets to the access net connection faster than they can be handled. The interface will permit the LDSP to deal in 16-bit words so that the number of CPU cycles required to send a packet to the access net connection is greatly reduced.

D. Measurements

The number of events on which an experimenter might wish to collect statistics far exceeds the capacity of the equipment. However, it is unlikely that any particular test needs to have records kept on every aspect of the system. Most of the measurements needed during the initial testing of any section of the system will rarely be used later except as a diagnostic tool. Therefore, it is proposed to provide measurements on several levels.

Level 1 would be the initial check out of the access net. Here traffic from one emulated terminal is desired with the ability to choose the bit rate and length of packet produced by that terminal. The packets will contain a standard header, a timestamp, a sequence number, an address (source and destination), and a known data pattern. TEM would record such statistics as number of packets sent and received, delays encountered, missing or out of sequence packets, and incorrect data patterns. Level 2 would be the next step in checking out the access net. Multiple phones would be emulated, but the experimenter would have the option of holding the

number of phones constant. Here, statistics would include delays, collisions, number of retries, idle time on the net, etc. Another level of TEM will check out the protocols involved in the access net and later in the mini-concentrator and the PSATNET.

It is envisioned that TEM will evolve to meet evolving experimental requirements. For example, when the mini-concentrator becomes available TEM can address many voice streams to it and test the "fairness" of what is discarded. When PSATNET is available, the TEM can help the mini-concentrator push the capacity of the PSATNET by providing giant packets into which it has multiplexed the packets from many terminals. For complex network experiments involving multiple sites, the local net TEM facility described here will be used in conjunction with other traffic emulation sources in the system. For example, network performance with a particular level of emulated access net traffic might be measured for a varying level of traffic generated from emulators in PSATs at other sites.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-79-281	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Information Processing Techniques Program • Volume I • Packet Speech Systems Technology • AD A066 249	5. TYPE OF REPORT & PERIOD COVERED: Semiannual Technical Summary 1 Apr – 30 Sep 1979
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Theodore Bially	8. CONTRACT OR GRANT NUMBER(s): F19628-78-C-0002; ARPA Order-2006
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order 2006 Program Element No. 62706E Project No. 9P10
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	12. REPORT DATE: 30 Sep 1979
	13. NUMBER OF PAGES: 40
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Electronic Systems Division Hanscom AFB Bedford, MA 01731	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

16 9P10

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

18. SUPPLEMENTARY NOTES

Supplement to ESD-TR-79-280 (Vol. II)

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

packet speech
network speech
voice conferencing
ARPANET
SATNET
homomorphic vocoding

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

This report describes work performed on the Packet Speech Systems Technology Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the period 1 April through 30 September 1979.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

207650 DM