**Inter-Office Memorandum**

| | | | |
|---|---|---|---|
| To | VoiceProject^ | Date | September 14, 1981 |
| From | W. Nowicki, L. Stewart | Location | Palo Alto |
| Subject | Voice Transmission Protocol Issues | Organization | PARC/CSL |

XEROX

Filed on: [Ivy]<Audio>Doc>EpProtocol.Press


## Abstract

A frequent criticism of the Ethernet local area network is that it is not suitable for real-time applications. Transmission of the human voice in the form of telephony is one application with severe real-time constraints. This memo describes some characteristics of voice transmission and the Ethernet. We have designed and implemented a real-time voice transmission protocol based on Pups. We also describe prototype implementations of the Etherphone and a Voice File Server.


## Facts about Voice and Telephony

*Medium-bandwidth*

Telephone quality voice can be achieved with transmission rates down to 8000 bits per second, but the required compression techniques are, at present, very computationally expensive. Intermediate bit rates are a possibility, but 64,000 bits per second is the present telephone industry standard. For this reason, we restrict our attention to 64 Kbps telephone industry compatible speech. Such digital voice signals consist of sampling the analog waveform 8000 times per second and representing each sample as an 8 bit digital encoding of the amplitude of the signal. The standard encoding is called m-255, a form of segmented logarithmic companding [AT&T 80].

*Real-time*

Voice communication from human to human (telephony) is a real time communications problem. The perceived delay must be fairly small and constant. Tolerable delays are generally below 100 milliseconds. [AT&T 80].

Voice filing, transmission of voice between a human and a storage device, is a half-duplex function. As such, it can tolerate higher delays provided that the *initial* delay, when a connection is set up, is not too long (on the order of a second).

*Echo*

Echoes are responsible for much of the perceived annoyance caused by delay in current long-distance telephone calls. There is a tradeoff betwen allowable delay and the loudness of echo.

Generally speaking, with more *return-loss* (less echo), longer delays can be tolerated. There are many sources of echo. Two important classes of echos are *acoustic echo*, which occurs when acoustic energy from the receiver (speaker) enters the transmitter (microphone), and hybrid echo, which is an electrical effect caused by reflections from hybrid circuits or impedance discontinuities in 2-wire voice paths [AT&T 80, Section 7.2]. The major concern is for "Talker Echo," which is generated on the receiver side (the person listening), but perceived on the transmitter side (the person talking).

*Conversation statistics*

Although a conversation is potentially full duplex (both people can talk at once), usually only one participant at a time is speaking. In addition, when a person is speaking, there are often gaps between words and sentences. On the other hand, both participants do occasionally speak at once. Over most conversations, about 47% of the full-duplex channel capacity is used [Bullington & Fraser 59].

The laws of large numbers apply to these statistics. Useful data points come from the telephone industry use of *Time Assigned Speech Interpolation* (TASI), in which a certain number of trunk circuits (such as transoceanic cables) are overcommited. If 24 full duplex trunks are available, usually 36 conversations can be supported, for a ratio of 1.5. If 150 circuits are available, 300 conversations can usually be supported, for a ratio of 2.0. [AT&T 80] These statistical effects are usually referred to as the *TASI advantage*.

It is also fairly well known that only a small fraction of phones will be in use at the busiest hour. Most of the time, almost all of the phones will be unused, but of course the system must be designed for worst case behavior. However, normal bussines phone usage statistics will probably not be valid in an integrated voice and data network.

*Error-tolerance*

To a certain extent, the human ear is tolerant of brief distortions in speech. For digital speech this means that small, transient errors in the digital representation of speech can be ignored. Dropouts of up to several milliseconds will be perceived as "pops" and "clicks," and will be tolerated as long as they are kept sufficiently infrequent.

*Virtual Circuit Service*

Once set up, a voice connection should maintain an adequate quality. In the presence of network overloading it would probably be better to reject connection attempts altogether than to offer poor quality. Thus, it is often better to block new calls than to degrade old ones.

## Consequences for Ethernet Transmission

The above characteristics of interactive voice have several consequences for the design of a datagram based voice transmission protocol. The two most critical requirements are that the voice transmission protocol and the end devices have sufficient throughput to support the voice data rate in a steady state and that the end-to-end delay be sufficiently small.

It is not sufficient for the system just to support the *average* data rate. The system must support the average data rate with sufficiently low variance to maintain a constant low delay. Some variation in the data rate can be compensated by increased initial delay. Voice data is buffered at the *receiving*

end so that the buffer runs dry with very low probability.

*Bandwidth Considerations*

The voice protocol must transmit enough packets per second to achieve a small delay. However, the number of packets per second must be low enough to be handled by software in the producer and consumer. Experiments have shown this limit to be about 100 packets per second. Inexpensive microprocessors will probably not do as well. For this reason speech compression techniques offer little help. They reduce the bandwidth, but the per-packet overhead is usually the limiting factor. The packet rate is related directly to the delay as:

$$D = 1/P + T_{Total}$$

Where $D$ is the delay, $P$ is the number of packets per second, and $T_{Total}$ is the total transmission delay, described later in more detail.

A simple sequence of arithmetic can tell us that the three Megabit per second Experimental Ethernet currently available within PARC can support about 40 simultaneous transmissions at 64 Kbps plus overhead. This means about 150-200 telephones could be supported on a single network, assuming 20% of the phones are busy during peak periods. On a 10 Mbps Ethernet, about 400 telephones could be supported, and about 100 phones on a 1.5 Mbps network. The number of telephones does not scale linearly with bandwidth because of the greater overheads necessary at higher speeds. The Ethernet slot time consumes more bits at higher data rates.

*Delay*

Most current digital voice transmission systems use some form of time-division multiplexing (TDMA). The advantage of TDMA is fixed delay, while the delay on CSMA channels is potentially unbounded. The total end to end delay is:

$$D = T_I + T_P + T_{S1} + T_D + T_X + T_{S2}$$

Where $T_I$ is the initial delay, $T_P$ is the packetization delay ($1/P$ above), $T_{S1}$ is the delay of software and process scheduling at the transmitter, $T_D$ is the defer time (waiting until the channel is not busy, or backing off after a collision), $T_X$ is the transmission time (time for the signal to propagate down the cable), and $T_{S2}$ is the delay of software and process scheduling at the receiver. These times are all probabilistic, so an accurate model must consider both their means and their deviations.

Although longer packets would improve efficiency, the total round-trip delay ($2D$) must be kept well below 100 milliseconds. Most of this delay is absorbed by the very process of packetization. The first sample of a packet cannot be sent to the receiver until the last sample of the packet has been digitized. The remaining delay has a certain minimum value corresponding to the minimum transmission delay between the two stations, but is actually made longer (by $T_I$) in order to smooth over *jitter* or variations in transmission delay. Note that the component of jitter associated with access to the Ethernet is typically much smaller than the jitter associated with the scheduling of processes in the sending and receiving stations. Several experimental studies at Lincoln Laboratories [Johnson & O'Leary 81], and PARC [Gonsalves 81] have shown that the defering delay is essentially zero up to 80% load. Even at loads of close to 100%, packet loss rates are only a few percent. Experiments with the prototype voice transmission system under heavy loads indicate higher loses, a topic which requires further study.

*Error Detection and Retransmission*

To the extent that the types and number of errors in the system are tolerable to the end users, a voice protocol does not need acknowlegements or retransmissions. Experiments have determined packet loss rates to be about 0.1% through all layers of software with no retransmission. This rate will be worse for heavier network loads, but could be better with improved Ethernet interface hardware. The Alto Ethernet interface has a brief window after receipt of a packet during which it cannot receive another packet. The result of this is loss of up to 1% of the packets when both sides of an interactive conversation are speaking at once. To a lesser extent, broadcast packets cause similar problems. These problems should be solved by an Ethernet interface in the Etherphone II which is capable of receiving back-to-back packets.

*Silence Detection and Echo Supression*

In order to benefit from the TASI advantage, a voice protocol must detect periods of silence and utilize reduced bandwidth while silence is present. This factor of two reduction in *average* bandwidth guarantees efficient utilization of the channel capacity [Shoch and Hupp], even when there are many simultaneous conversations. In fact, some studies have concluded [Weinstein 80] that the CSMA strategy of Ethernet is *more* efficient than traditional TDMA schemes.

Since digital voice transmission provides independent transmission and receive paths, equivalent to a "4-wire" telephone, it need not be directly concerned with echo. However, acoustic echo may exist at one or both ends, and a voice protocol connection might be connected in tandem with a 2-wire transmisssion path, leading to hybrid echo. The silence threshold must be high enough to avoid detecting this echo as speech.

## Etherphone 0 Transmission Protocol

The Etherphone 0 is an Alto I using an Auburn audio board. The VFS 0 is a Dorado running Pilot. The Etherphone 0 protocol permits real-time voice conversations between two Etherphone 0s, or an Etherphone and the Voice File Server.

Voice is digitized at 8000 samples per second. The samples are encoded in 8 bits using industry standard m-255 companding. When speech is detected, the Alto transmits 50 packets per second, each with 160 voice samples plus a sequence number indicating the sample number of the first sample of the packet. Since the samples are generated at a fixed 8000 Hz rate, this sequence number is equivalent to a timestamp.

*Instrumentation*

Since the voice transmission protocol does not retransmit lost packets, we are trying to engineer the system in such a way that it does not lose too many. Each packet has not only a sample number, but also a second sequence number indicating the *number* of packets which have been transmitted -- allowing the receiving station to accurately count lost packets.

*Silence Detection*

The Etherphone 0 audio microcode computes the sum of the upper 8 bits of the absolute value of the 12 bit linear encoded samples produced by the Auburn A/D converter before it converts them to the m-255 code. If this value, summed over a given 160 sample block, falls below a certain

threshold, the input is deemed to be *silent*. After a certain number of consecutive silent blocks, the originating machine stops transmitting packets. By this means, typically half the required communications bandwidth is saved. Care must be taken, however, to set the number of consecutive packets before shutdown high enough to avoid the anoying effect of shutting down between very short pauses, such as those that occurr between words.

During a silence interval, the receiving station plays silence to the listener. After a silence interval, packets again begin to arrive at the receiving station. In order to account for jitter in the arrival of future packets, the very first packet is delayed by 10 milliseconds before it is played.

*Delay*

We have chosen to allow about 30 milliseconds between the time a particular sample is digitized at the originating station and the time at which it appears at the D/A converter of the receiving station. Two thirds of this delay, 20 milliseconds (or 160 sample times) is due to the packetization process. The component of jitter associated with access to the Ethernet is essentially zero, while the jitter associated with the scheduling of processes in the Alto I can be a few milliseconds.

The implementation of the jitter reduction delay is as follows. An assumption is made that the *first* packet to arrive does so with a *typical* transmission delay. A 10 millisecond silence is placed on the D/A queue in front of the first packet. Assuming that the clocks of the sending and receiving station are running at the same frequency, the delay between A/D at the originating station and D/A at the receiving station becomes fixed at 20 msec for packetization, plus the transmission delay of the first packet, plus 10 msec smoothing delay inserted by the receiving station. This process is repeated at the end of each silence interval.

In fact, the resynchronization is done whenever the sequence number of an arriving packet does not match the expected sequence number, so a lost packet is treated exactly like a silence interval.

## Protocol enhancements: Etherphone I protocol

The Etherphone I system also uses Alto I/Auburn as the *Etherphone*. The system includes an *Etherphone server* and uses an enhanced transmission protocol. The Etherphone Server keeps track of the state of the entire Etherphone system and is responsible for setting up calls. The protocols for communication with the Etherphone server are described elsewhere [Swinehart 81].

*For Internetworking and System Control*

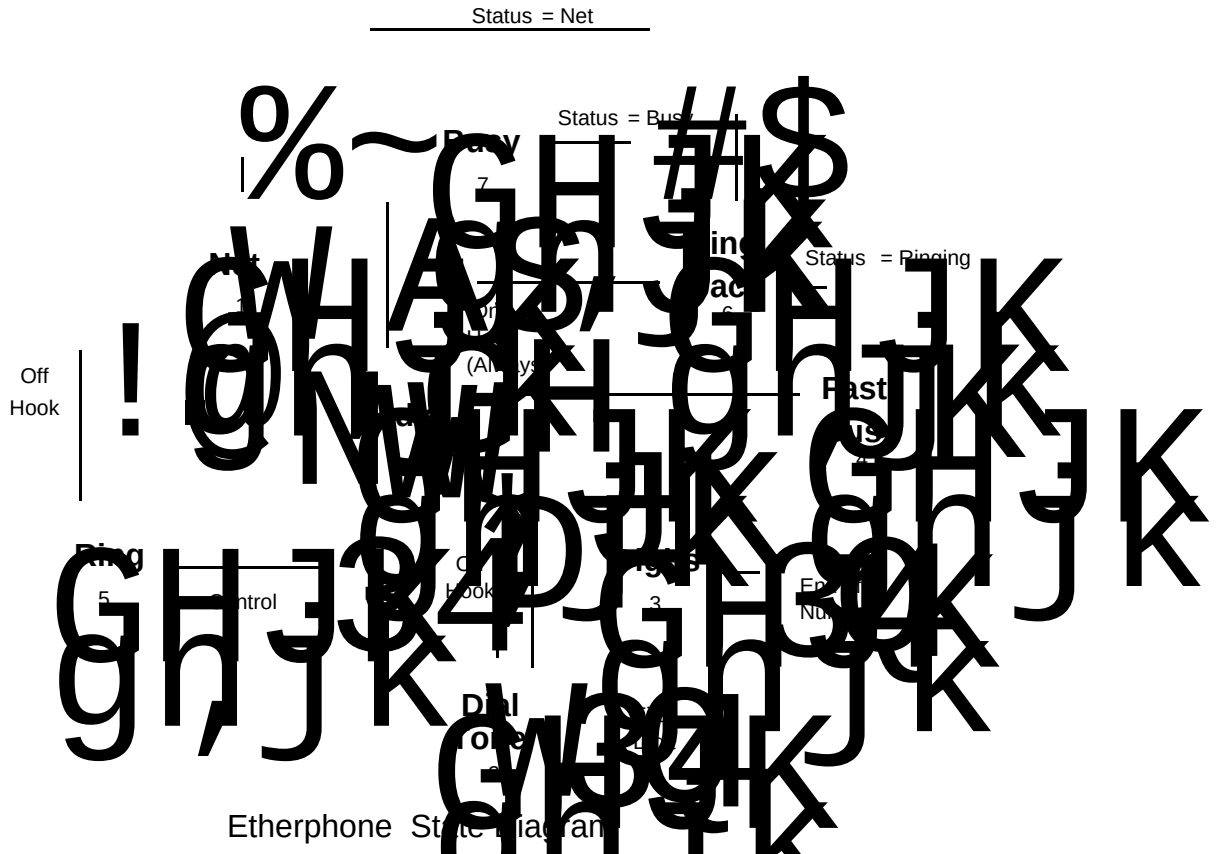
As a general principle, we are trying to avoid maintaining distributed state, and the associated synchronization problems. For example, two Etherphones must be able to tell whether a connection between them should be closed. In addition an existing voice connection should not be disturbed by other traffic. It is better to prevent a call from starting than to allow it to be disturbed once set up. For these reasons, Etherphones transmit packets even during silence intervals, which are shorter than packets carrying voice data and transmitted at a lower rate (two per second).

## Appendix: Implementation Details

This section briefly describes the implementation of "Etherphone 0," a test program used to simulate the operation of a telphone. The program is written in BCPL, and uses the Auburn Audio interface for an Alto I or an Alto II. The source to the program is in [IVY]<Audio>AubI>Etherphone.dm. The program itself is Etherphone.Run in this dump file. There is a boot file version available on Ivy (as Ep.Boot), so a local disk is not needed. The display is not needed, except for diagnostics information.

*States*

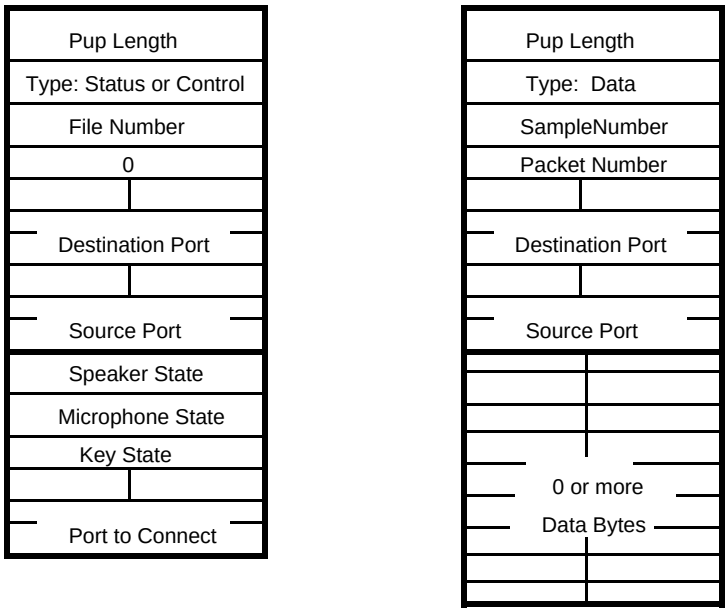
The following diagram illustrates the general operation of the EtherPhone program. There are also two other major binary state variables, which determine if the microphone and switches are active.




*Protocol*

The Etherphone protocol is currently very simple. It is a socket-level protocol with packets directed to Pup socket 200001 (octal). There are three kinds of packets: data (Pup type 372 octal), control (Pup type 373 octal), and status (Pup type 374 octal). The data packets consist of a sample number and packet number as the Pup ID, plus the 160 voice samples packed two to a word. The status and control Pups contain the state of the speaker (as given in the above diagram) in the first word, the state of the microphone in the second word (1 means on, 0 means off), and the state of the keys in the third word. Starting in the fourth word is the remote port to which the Etherphone is connected.

Samples are continuously digitized from the microphone at 8000 per second, using standard eight bit m-law encoding. The packets are sent on the Ethernet if the microphone state is on and the sum of the absolute values is greater than a given threshold (currently 40). This results in 50 packets per second when silence is not detected (one packet per 20 milliseconds). In the appropriate state the packets are played back with m-law decoding, with a 10 millisecond silence interval added if the sample numbers are out of sequence.




Etherphone Packet Formats

*Structure*

The program uses as many pre-existing packages as possible, including the standard Pup package (with checksums turned off!), the context package, and the queue package. AubIControl.Bcpl (written by Dan Swinehart and Larry Stewart) is the interface to the special audio microcode. Tones.Bcpl implements the generation of tones (touch tones, ringing, busy, etc.) including counting the correct number of 10 millisecond control blocks required to acheive AT&T standard timings.

There are four major processes (besides the Pup processes). KeyControl, in the file EpKeys.Bcpl, reads the keys and changes the state appropriately. It also prints out the status and the delay histogram. Each number in the histogram corresponds to a one millisecond interval. The first number is the number of packets played when the speaker queue was dry, the next number is for those with between one and two milliseconds of samples left, and so on.

The main code is in EtherPhone.Bcpl. PhoneToNet monitors the microphone input queue, recycling audio control blocks and sending packets when it is turned on. NetToPhone does most of the work, detecting off-on hook transitions, and reading packets from the network. The Speaker state is changed according to the state diagram by this process. Finally, ACBscavanger recycles audio control blocks that have been played, continuing the tones when they should be continued, and handles timeouts.

*Use*

When the Etherphone program is invoked, the display is turned off except for a telephone shaped cursor. The Shift-Lock key can be depressed to simulate taking the handset off hook, or you can use a modified telephone set which brings out the switchhook signal. A dial tone should be heard at this time. A number can be dialed as a 3 followed by he octal network address of the station to be called, or by typing a letter H and the hostname. If the called station is also running EtherPhone.Run and is on-hook, a ring-back should be heard and the remote station will ring. If the called station is off-hook, a busy signal will be played. If no response is received, a "fast busy" sound is heard. The digit 1 followed by three digits will store into the given file on a voice file server. A 2 followed by three digits plays back the given file.

Other commands which can be typed on the keyboard are:

| | |
|---|---|
| D | Gets a "drop" factor. This factor is taken to be the number of packets out of 32768 to drop on the average, to simulate missing packets. |
| H | Gets a host name to connect to, as if you had dialed a 3 followed by the host number in octal. |
| L | Gets block length. This is the number of samples to include in each packet sent. The default value is 160, or 20 milliseconds. |
| Q | Quits the program, after restoring the display and displaying the statistics. |
| S | Displays the statistics, including the delay distribution. Type any other key to turn the display back off. |
| T | Gets a new silence threshold value. Default is 40, -1 means continuously send packets. |
| V | Asks for the name of a voice file server host. |

When the remote station comes off-hook, the conversation can begin. The cursor moves horizontally when data packets are received, and vertically when packets are sent. To exit, terminate the call, and then use the Q command. The program prints the number of packets digitized, sent, and received in the last completed call before exiting.

**VFS0 - Voice File server**

The prototype voice file server program is written using Pilot Mesa. The source code is in [Ivy]<Nowicki>VFSImpl.Mesa, with a .df file under [Ivy]<Nowicki>VFS.df which refers to the .Config file and the many packages it needs. Just compile VFSImpl.Mesa, bind VFS.Config, and run the resulting BCD. Due to a bug in the Mesa Pup package, you must reboot Pilot after running the program. Control delete interrupts at any time. Currently only one connection is handled at any time.

## References

[AT&T 80]
*Notes on The Network*, American Telephone and Telegraph Company, 1980.

[Bullington & Fraser 59]
K. Bullington and J. M. Fraser, "Engineering Aspects of TASI," *Bell System Technical Journal*, Volume 38, 1959 pp. 353-364. *

[Cohen 76]
D. Cohen, "On Network Protocols for Speech Communication Communication," Proceedings of the Ninth Hawaii International Conference on Systems Sciences, Honolulu, January 1976, pp. 83-86.

[Cohen 77]
D. Cohen, "Issues in Transnet Packetized Voice Communication," Proceedings of the Fifth Data Communications Symposium, Snowbird, Utah, September 1977, pp 6-10 - 6-13.

[Cohen 78]
D. Cohen, "A Protocol for Packet-Switching Voice Communication," *Computer Networks*, 2:4/5 September/October 1978, pp. 320-331. *

[Cohen 80]
D. Cohen, "Flow Control for Real-Time Applications," *Computer Communication Review*, 10:1-2, January/April 1980, pp. 41-47. *

[Cohen 81]
D. Cohen, "Packet Communication of Online Speech," Proceedings of the International Conference on Computer Message Systems, Ottawa, April 1981. *

[Cohen 81a]
D. Cohen, "Packet Communication of Online Speech," AFIPS Conference Proceedings, National Computer Conference, Chicago Illinois, May 1981, pp. 169-181. *

[Forgie 75]
J. W. Forgie, "Speech Transmission in Packet-Switched Store and Forward Networks," AFIPS Conference Proceedings 44, National Computer Conference, May 1975, pp. 137-142 *

[Forgie & Nemeth 77]
J. W. Forgie and A. Nemeth, "An effecient Packetized Voice/Data Network Using Statistical Flow Control," International Conference on Communications, Chicago Illinois, June 1977. *

[Forgie 80]
J. W. Forgie, "Voice Conferencing in Packet Networks," International Conference on Communications, Seattle, June 1980. *

[Gold 80]
B. Gold, "Digital Speech Networks," *Proceedings of the IEEE*, December 1977, pp. 1636-1658. *

[Gonsalves 81]
T. A. Gonsalves, Personal Communication, September, 1981.

[Hayes & Sherman 72]
J. F. Hayes, and D. N. Sherman, "A Study of Data Multiplexing Techniques and Delay Performance," *Bell System Technical Journal*, Volume 51, November 1972 pp. 1983-2011. *

[Hatch 76]
Hatch, "Models for the Subjective Effects of Loss, Noise, and Talker Echo on Telephone Connections," *Bell System Technical Journal*, Volume 55, November 1976. *

[Johnson & O'Leary 81]
D. H. Johnson and G. C. O'Leary, "A Local Access Network for Packetized Digital Voice Communications," *IEEE Transactions on Communications*, 29:5, May 1981, presented at National Telecommunications Conference, December 1979, paper 13.4.

[McAuliffe 78]
D. McAuliffe, "An Integrated Approach to Communication Switching," International Conference on Communications, Toronto, June 1978. *

[O'Leary 80]
G. C. O'Leary, "Local Access Facilities for Packet Voice," Proceedings of the Fifth International Conference on Computer Communications, Atlanta, August, June 1980 pp. 281-286. *

[Ross 77]
H. Rudin, "Design Approaches and Performance Criteria for Voice/Data Switching," *Proceedings of the IEEE*, September 1977. *

[Rudin 78]
H. Rudin, "Studies on the Integration of Circuit and Packet Switching," International Conference on Communications, Toronto, June 1978. *

[Shoch 80]
J. F. Shoch, and J. A. Hupp, "Measured Performance of an Ethernet Local Network," Xerox PARC Report CSL-80-2, also presented at the Local Area Communication Network Symposium, Boston, May, 1980.

[Shoch 80a]
Shoch, J. F. , "Carrying Voice Traffic Through an Ethernet Local Network," IFIP Working Group 6.4 International Workshop on Local-Area Computer Networks, Zurich, August, 1980.

[Spector 81]
A. Z. Spector, "Performance Remote Operations Effeciently in Local Networks," Ph.D Thesis, Stanford University, June 1981.

[Swinehart 81]
Swinehart, D. C., "Etherphone Server Protocol Specification," Xerox PARC Internal Memo, September, 1981.

[Tobagi 79]
Tobagi, F. A. and V. B. Hunt, "Performace Analysis of Carrier Sense Multiple Access with Collision Detection Local Area Communication Network Systems," Stanford University CSL TR-173, June, 1979. *

[Weinstein 80]
C. J. Weinstein, A. J. McLaughlin, and T. Bially, "Effecient Multiplexing of Voice and Data in Integrated Digital Networks," International Conference on Communications, Seattle, June 1980. *

References marked with * have not been located by the author.