A Protocol for Packet-Switching Voice Communication

Dan Cohen
USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, California 90291, USA

This paper discusses the issues associated with real-time voice communication over packet switched networks. It suggests an approach for the design of a protocol to support this application. The ARPA Network Voice Protocol is presented as an example of such a protocol. In addition, two extensions to it are discussed. Most of the issues which are discussed in the context of the NVP, like separation of control from data, device independence, and performance monitoring are not unique only to voice application, but apply equally to any other real-time protocol. The major virtue of the real-time voice communication as far as networking is concerned is the real-time, not the voice, aspect of this communication.

Keywords: Conference, CVSD, high level protocol, LPC, negotiation, network voice protocol, NVP, packet switched network, PCM, protocol, real-time, speech compression, vocoding, voice communication

Dan Cohen recieved a B.Sc. degree in mathematics from Technion, Israel Institute of Technology, in 1963, and a Ph.D. degree in computer science from Harvard University, Cambridge, Massachusetts, in 1969. Since then he was on the faculty of Harvard University, Technion and the California Institute of Technology. In 1973 he joined the Information Sciences Institute (ISI) of the University of Southern California, where he leads several computer network related projects. His research interests include interactive real-time systems, graphics and voice communication, networking and computer architecture.

[1] This research is supported by the Advanced Research Projects Agency under Contract No. DAHC15 72 C 0308, ARPA Order No. 2223. Views and conclusions contained in this study are the author's and should not be interpreted as representing the official opinion or policy of ARPA, the U.S. Government or any other person or agency connected with them.

This paper has been presented at the Computer Network Protocols Symposium, held in Liege (Belgium) in February 1978 and organized by the University of Liege. The permission to reprint this paper is gratefully acknowledged.

1. Introduction

This paper discusses a high level protocol for real-time voice communication over a packet switched computer communication network. First, the problems associated with real-time digital voice communication and its transmission through a packet switched network are discussed. These problems are addressed by the design objectives of the protocol introduced in the next section, then the most popular techniques for digital voice encoding (vocoding) are summarized.

With this background, a specific NVP (Network Voice Protocol) which has been operational on the ARPANET is introduced. This protocol is then used as the basis for two higher level protocols. One of these protocols deals with multi-participant conferences, and the other with offline voice files.

2. Some problems in real-time voice communication

In order to communicate voice over digital communication media, a digital encoding of the voice, called **vocoding**, must be used. Many vocoding techniques have been developed by the signal processing community. In general, the higher the desired voice reproduction quality, the higher the data rate, measured in bits per second over the communication channel, has to be. However, for a given fixed quality the data rate may be reduced at the cost of more processing.

Real-time voice communication requires real-time vocoding; therefore, there is a tradeoff between processing and communication. The more online real-time processing is available, the less real-time communication bandwidth is required, and the more capable the communication, the less processing required.

In addition, for real-time communication the absolute delay between the time an utterance is spoken and the time it is heard should not exceed some maximum allowable time period, T, which may be as long as a few hundred milliseconds. Longer delays can cause degradation of the interactive nature of the human voice communication process. Packet switched communication is best suited to signals with a high peak-to-average bit rate ratio. Fortunately, human voice is such a signal. There are many discontinuities in the voice, due to breathing, intersentence pauses, listening to the other party, etc. Obviously, these discontinuities can be coded using very few bits. Advanced vocoding schemes, which are very compatible with packet switched communication, have a variable output data rate, and are capable of taking advantage of certain redundancies in the human voice (long vowels, for example) in order to decrease momentarily the data rate needed to maintain fixed quality.

Unfortunately, packet switched communication can have variable delays between packet arrivals. This variability can cause discontinuities in the voice delivery unless some corrective action is taken.

This corrective action introduces a delay in the voice delivery, in addition to the vocoding, packetization, and communication delays. Since the total delay should not exceed the threshold T, some compromises may have to be made. A detailed discussion of strategies for coping with the variable network delays can be found in [4].

The term "quality" is used here without a precise definition. This is not an oversight, but an indication of an unfortunate lack of objective measurement of speech reproduction quality. Even though there is no single metric for quality evaluation, there are several characteristics which can be measured subjectively, such as speaker recognition, word or phrase intelligibility, continuity, clarity, etc.

3. General high level protocol issues

In general, communications protocols have CONTROL and DATA portions. In this case the CONTROL portion deals with establishing a connection, alerting, vocoder compatibility, communication formats, validation, and the like. The **data** portion deals with the real-time communication of the voice parameters. There are several general issues which are common to many high level protocols. Some of these issues which also apply to real-time voice communication are

– Separation of Control and DATA,
– Device independence,
– Real-time binding of data formats,
– Robustness,
– Support of higher level protocols,
– Flow control, and
– Ability to adapt to variable network performance.

A voice protocol, like other high level protocols, should be designed to address only the content and the format of the messages exchanged between the participating systems, not the actual algorithms used to create them. This approach supports device independence, since neither the algorithms used for a given type of vocoding, nor the hardware used for their implementation, has any consequence outside the system where it is used as long as the defined protocol is complied with.

The CONTROL portion is independent of the vocoding technique in use. In fact, the same CONTROL can be used with a multitude of vocoding techniques, and therefore should provide for the choice and validation of the vocoding. The CONTROL portion establishes connections between the parties and verifies the status of the connection, alerting the user of the system to any communication difficulties, disconnection, etc.

In addition, the control portion is responsible for ensuring that both participants use the same vocoding technique, with parameters which maintain compatibility between both ends. Obviously, this compatibility is as important as establishing the communication. If the vocoding techniques preferred by the parties differ, some negotiation may take place in order to reach a mutual agreement. This negotiation usually can bridge minor differences in parameter setting for the same vocoding algorithm (e.g., the number of poles in LPC) but probably cannot handle major differences. This negotiation also has the advantage that it can provide for downward compatibility between systems which undergo gradual improvements.

The negotiation phase terminates either successfully, when a mutually agreed data format is reached, or unsuccessfully, if no agreement is reached. In this case, the connection is automatically aborted. After a successful negotiation, the ANSWERER notifies the CALLER (by using CONTROL messages) that it is ready for the beginning of the real-time voice communication.

Note that the CONTROL functions are VOCODER-INDEPENDENT, except the details and the parameters of the vocoding scheme negotiated. Therefore, in order to maintain the desired separation between CONTROL and DATA as well as device independence, there is a need to define a vocoder-independent table of the parameters needed for each vocoding scheme.

More detail on the structure of the negotiation will be presented in Section 5. The negotiation phase is very important because it allows RUN-TIME BINDING. For example, suppose that two similar systems use a vocoding scheme which requires communication of parameters which are smaller than the "word" size of these systems. If the communication is expensive but the processing is cheap, one should "pack" the parameters to be communicated within a "word". On the other hand, when the communication gets cheaper (as a result of a reduced load on the channel, for example) it might be advantageous not to pack the parameters in order to save processing. Obviously, this minor issue of packing should not be decided upon when the protocol is defined, but should be left open, to be bound (agreed upon) at run-time according to the prevailing conditions then.

Several other issues belong to the same category of "format" rather than "content". "Maximum message size" might be one of them. In general, the issues associated with the actual transmission parameters may be encapsulated in a lower transmission protocol, which (although necessary for supporting the voice communication) could be treated separately.

This separation is even more important for the support of real-time voice communication between systems which may be on different, but interconnected, communication networks. The performance of the communication channel has a major effect on the quality of the interactive communication. This effect is not just quantitative but also highly qualitative. Network performance is expressed in terms of "delay", "reliability" (errors) and the like. For many familiar applications (file transfer, for example), reliability is most important, and retransmission must occur if transmission errors are detected.

This is not necessarily the case for interactive voice communication. In many cases, such retransmission delay (the delay between the time that the retransmission is requested by the receiving process because of a detected transmission error and the time the retransmitted message arrives there) causes a discontinuity in the delivery of the reproduced speech. This retransmission discontinuity can be more disturbing to the communication than the original transmission error.

Thus, a voice protocol should have provisions to affect the tradeoff between delay and reliability in such a way that when interactive voice communication takes place (as opposed to offline communication between files, or person-to-file) the delays never exceed a given threshold T, even if this implies some loss of data.

Since the network performance has such an important role in the quality of the voice communication, it is also desirable to be able to monitor the performance and to adapt to it, in order to optimize the overall communication quality. For example, if a high data rate implies longer network delays, the vocoding scheme may be restricted dynamically to a lower data rate which decreases the network delay at the expense of the speech reproduction fidelity, such that the overall communication quality (delays and fidelity) is improved.

Vocoders which can adapt to a variable data rate are capable of "graceful degradation," which is an important feature for many systems. In order to achieve this, performance monitoring must be supported by the protocol.

A protocol should also provide robustness – the ability to recover from communication problems, "bad" messages, etc. No participating system should be able to tie up the resources of another system or to put them into any other undesirable state.

Loss or duplication of messages, out-of-order arrival, and any other possible error should never have significant effect, whether the problem occurs with CONTROL or DATA messages. Obviously, problems with CONTROL messages have a greater potential for ill effects than problems with DATA messages.

In order to achieve robustness it is important to avoid (or at least to minimize) the notion of STATE and the need to assume the state of the other system. This can be done by including the expected "state" in messages. For example, if a certain message means "CAN-YOU-DO-X?", the answer should not be a "YES" or "NO". These terse responses might cause problems if they arrive in the wrong order, or if they are duplicated. Instead, longer messages like "I-CAN-DO-X" and "I-CANNOT-DO-X" have a much smaller potential for disaster.

It is important for a system to be able to break any connection when it gets into trouble (like a buffer overflow). It is also important that systems can recognize broken connections and re-establish them easily and automatically. These abilities contribute significantly to the overall system's robustness, since they can clear many undesired side effects caused by unexpected behavior by one of the communicating systems or by the transmission media.

The DATA messages have to be marked not only with a sequence number (for loss and duplication detection) like most other protocols, but also with some indication of the time when they were created. This **"time stamp"** typifies not only a voice protocol, but all real-time protocols, where the timing of events is important.

Unlike several other protocols (ELECTRONIC MAIL, for example) the online voice protocol has the built-in notion of DATA-RATE. Therefore, the flow control problem is significantly simpler than that of systems without the intrinsic notion of **data rate**. Comparing the flow of real-time speech with the flow of a file being transferred is similar to comparing a steady water flow with pulses of various intensity and duration. In general, it is not necessarily the case that the same flow control mechanism is optimal for applications like file transfer and real-time speech. In order to maintain a small total voice communication delay, a spoken sentence may start to be reproduced at the destination before it is completed at the source. Due to increased network traffic, there might be a temporary delay increase in the delivery of the data messages. In this case the receiving system has to perform some compromise between "dumping" good voice data (as the data rate increases after the delay), which degrades the fidelity, and increasing the total communication delay, which also degrades the quality of the interactive voice communication.

Several strategies for coping with this problem can be found in [4]. It is most convenient if there is no need for the source system to be aware which strategy is used by the other system, since this issue is self-contained in each destination system.

A very important consideration in a design of protocols is the ability of a high level protocol to be used as a lower level protocol by an even higher level protocol. Section 6 will demonstrate how the control portion of a voice protocol can support higher level protocols.

4. Vocoding techniques

Digital representation (coding) of sound waveforms is necessary for voice communication over digital channels. Obviously, a limited number of data bits (or a limited data rate) can represent only a limited number of waveforms. Therefore, in general the vocoding distorts the waveforms in order to achieve an acceptable data rate, while minimizing the resulting perceptual distortion.

In this section several vocoding methods are described and compared. Data rate figures will purposely not be quoted, since any mention of bit rate figures has to be coupled to an exact definition of the quality (fidelity) obtained, which is too complicated for the scope of this paper. The reader who is interested in the general protocol issues, but not in the specifics of vocoding techniques may skip this entire section.

The interested reader is refferred to Flanagan's book [6] for an excellent detailed discussion of most vocoding techniques.

The voice communication is composed of:

(a) Digital representation (vocoding) of the voice signal,

(b) Transmission of this digital representation, and

(c) Reproduction of the voice from the received data.

Let the source signal be $X(t)$, and the reproduced signal $Y(t)$. In the ideal situation, with perfect reproduction and no processing and communication delays, $Y(t) = X(t)$. Unfortunately, this is not the case in real life systems, and an error $E(t) = Y(t) - X(t)$ does occur. To be more precise, the error should be defined as $E(t) = Y(t) - X(t-d(t))$, where $d(t)$ is the system delay, due to the processing and communication. For the sake of simplicity, $d(t)$ will be eliminated from this discussion, as if both the processing and the communication were instantaneous. Since it is impossible to guarantee that $E(t)$ vanishes, it is desirable to minimize it. There are many ways to define an "error-minimization," but obviously a metric which minimizes the perceptual effects is desired. A mean-square minimization seems to work reasonably well and is most convenient.

The most commonly used vocoding schemes sample the voice signal, $X(t)$, at some sampling frequency, F, and digitize its level (say its amplitude) at these discrete points in time, using B bits per sample. Hence this process has a data rate of BF. For example, if a sampling frequency of $F = 10$ kHz is chosen, and 8 bits are used per sample, then a data rate of 80 Kbps results.

Note that by using the above technique one gets a discrete representation of the source signal, which may be significantly different perceptually from it, if the parameters are not chosen properly. This discrepancy between the continuous source and discrete representation is due to their spectral differences. Sampling waveforms which contain frequencies higher than $F/2$ results in an undesired aliasing effect. For example, sampling a sinusoid with frequency F will result in an apparent "DC level", i.e., an apparent frequency of zero. Therefore, the sampling has a built-in ambiguity, which cannot be resolved after the sampling. In order to remove this ambiguity, all frequencies above $F/2$ are removed by analog filters from the signal before the digitization by the sampling A/D converter. Dually, at the destination system a discrete signal is reproduced, and some smoothing process is used to avoid another undesirable aliasing effect. This is usually performed by analog filters.

In order to reduce the communication requirements, the values of B and F are chosen to be as small as possible for a particular application. Obviously, high fidelity music requires higher values for both B and F than "telephone-quality" speech.

A method which sends all the BF bits is the ordinary PCM (Pulse Code Modulation) method. This technique requires less processing than any other technique at the expense of a high data rate to be communicated. If F is high enough, then the Differential PCM (DPCM) method can be used. In this technique the differences between successive samples are sent instead of the sampled values. If the sampling frequency is high enough in relation to the frequencies in the source signal, then the successive differences are smaller than the values themselves, and fewer bits are needed to describe them, which results in a lower data rate. By observing some of the properties of human voice, further improvements can be applied to these methods, such as ADPCM (Adaptive DPCM) and CVSD (Continuously Variable Slope Delta Modulation). The basic idea of adaptive techniques is that the codes which are transmitted represent values which depend not only on their codes but also on the previous codes. For example, a 1 after a 0 might mean an addition of a certain increment to the current value, and each successive 1 may represent an increment which is larger by a factor of K than the preceding one, and similarly for 0's, in the opposite direction. After the proper parameters are adjusted such adaptive techniques may achieve a great reduction of the data rate needed for a certain class of sounds at the expense of possible degradation of the quality of the reproduction of other classes of sound.

It is worth noting that all the techniques described so far are robust in respect to transmission errors which modify the values of the arriving bits (but not their number). Such errors usually cause a temporary audible "glitch," from which these systems recover momentarily. Of the above, CVSD is also robust in respect to lost bits (i.e., reception of a different number of bits than transmitted). This is because CVSD uses only one bit per sample, and is therefore immune to loss of synchronization.

These methods, and similar ones, are a clever application of information-theory techniques to reproduce waveforms which look similar to the source waveforms. In fact, by using more advanced information-theory techniques (e.g., Huffman [2] or Ziv-Lempel [3] coding) further data rate reductions may be achieved for some classes of sound.

A common property of all the above methods is that they transmit a description of the source waveform itself, and can be used on signals other than the human voice. The reproduced waveform looks as close as possible to the original source one.

Acoustic vocoding techniques are aimed at reproducing waveforms which **sound** like the voice source. Theoretically, these techniques encode only the acoustic features of the speech, without the acoustic redundancies which do not contribute anything to the perception process. Since less information is carried, fewer coded bits should suffice to transmit the speech. For example, musical notes for describing music played by a flute require a few orders of magnitude lower data rate than what is needed for a similar reproduction quality by ADPCM vocoding.

This tremendous data reduction is due not only to the use of the frequency domain rather than the time domain to describe the sound, but also to the availability of a very accurate model of the sound production system. The notes by themselves are not enough for an accurate reproduction, without adding the additional information that the sound was originally produced by a flute, rather than by a piano.

Obviously, when the wrong model is used, there are severe limitations on how close to the original sound the reproduction can be. The better the model the less data required for describing the sound. Once the model is well defined, the data has merely to be a set of parameters built into the model. On the other hand, when the model is well known and high data rate reduction is achieved, the system may have to be very sensitive to deviations from the assumed model, and may lose its robustness in this manner. In addition, when the data is compressed, the effect of data errors in the transmission is very likely to be more severe than in the case of less compressed data.

Linear Predictive Coding (LPC) is one of the most popular acoustic vocoding schemes. It is based on modeling the human sound production system as a tube with several sections, each with a different diameter used to transform (filter) an excitation signal. In the case of voiced sounds, the excitation is a series of impulses spaced at the pitch period, and in the case of unvoiced sounds the excitation is noise-like. In this model, the vocoding task is reduced to solving for the parameters of the vocal tract tube, and the pitch and amplitude of the excitation signal.

Even though LPC is a time-domain vocoding technique, it allows lower data rates than any of the above techniques, with the same quality. It reproduces waveforms which are acoustically similar to the voice source, but are not necessarily the same shape. The significant data rate reduction via LPC is paid for by additional computation. This price was prohibitive several years ago, but with today's advances in microelectronics it is not out of reach. Several vendors are in the process of implementing an entire LPC processor using only a few integrated circuits.

The most sensitive (and also the most time-consuming) part of the LPC computation is the pitch determination (tracking). Due to the variability of the human sound production system, it is not always possible to solve correctly for the pitch value, especially in the presence of background noise and other sounds which are present outside sound-proof chambers.

In order to overcome this difficulty, LPC may be used for all the parameters except the pitch and the gain, which are carried by the prediction error signal sent along with the LPC coefficients. In the case of a source signal which agrees perfectly with the assumed LPC model, the error signal is a series of excitation pulses, spaced a pitch period apart. In other cases, the predicted error signal is of a more complicated nature, but still can be coded using a relative low data rate. This combination can be used to overcome noisy situations, and other situations which deviate markedly from the LPC model. Obviously, this compromise is paid for by an increased data rate. The APC (Adaptive Predictive Coding) is such a technique.

Channel vocoders and formant vocoders are frequency-domain techniques. Both are based on communication of spectral information, i.e., information which is in the frequency-domain, rather than in the time-domain as in the vocoding methods described so far. Channel vocoders operate by extracting the energy levels in a set of frequency bands, and transmitting these energies. One can view the channel vocoding as a stepwise approximation to the spectrum. Since an analog implementation of this method has been known for many years, channel vocoders have been around longer than any other type of low-rate vocoder.

Formant vocoders use another approximation to the spectrum. They identify the first few formant frequencies (resonant frequencies of the vocal tract) and transmit their position and relative magnitude. This is an effective method, since the human perception process seems to be particularly sensitive to the formant frequencies.

Another good vocoder is the cepstral, or homomorphic vocoder, which operates in neither the time- nor the frequency-domain, but in the cepstral-domain. This domain is achieved by a series of mathematical transformations (Fourier transform, logarithm and Inverse Fourier transform).

The particular vocoder primarily used for experiments in digital voice transmission on the ARPANET was an LPC vocoder based on the ideas of J.D. Markel, J.I. Makhoul, and others. In this vocoder, the analysis system divides the sampled signal into "vocoding periods", or frames, of about 20 milliseconds each. Each frame is then analyzed, and the LPC coefficients, the pitch and the amplitude are computed. A larger (40 millisecond) frame is used for pitch determination. Reference [1] contains a detailed description of algorithms which may be used for these tasks. These algorithms are not unique, and different algorithms can be found in the literature.

After computing these parameters, they are encoded using a table lookup, i.e., intervals of each parameter are coded into an integer value which represents the interval to which they belong. The receiver has similar tables, which convert each code into a value which represents the entire interval. For each parameter a different table may be used. The tables are constructed such that the perceptual error is minimized under certain assumptions.

Analysis of each vocoding period also shows whether or not the speaker was momentarily silent. If the speaker is silent a special code is sent to indicate this silence. After a certain number of successive silent periods, no transmission occurs until actual voice is detected again.

If the frame is found not to be silent, it is compared with the previous frame. If the difference is found to be acoustically insignificant, then a special code is sent. If the difference is found to be significant, then either the description of the new period is sent, or a description of only the acoustic difference, whichever is found to require fewer bits. This is obviously a variable rate vocoder, which reduces the required data rate by almost one-third, compared with a similar fixed rate vocoder.

Variable rate vocoders therefore use more data when needed, and less during the "slower" periods, hence taking advantage of the fact that not all cases are "worst case", ideally suited to packet switching environment.

5. The ARPA network-protocol (NVP)

In late 1973 ARPA initiated a multi-site research project to investigate the feasibility and to demonstrate low data rate real-time high quality voice conferencing, over a packet switching communication network. Since such a task was never undertaken before the supporting research had to address a wide variety of issues ranging from real-time communication protocols to acoustic vocoding.

The first experiments used LPC among ISI (USC/Information Sciences Institute, in Los Angeles), LL (MIT Lincoln Laboratory in the Boston area), CHI (Culler-Harrison, Inc., in Santa Barbara), and SRI (Stanford Research Institute, in the San Francisco area). Different equipment was used at each site. At ISI a PDP11/45 with an SPS-41 signal processor was used. At Lincoln Laboratory their unique TX-2 augmented by their unique FDP signal processor was used. At CHI their own designed and built MP32A with their own AP-90 was used. At SRI a PDP11/40 with an SPS-41 was used.

It is interesting to note that in addition to the above diverse equipment, Lincoln Laboratory later replaced its processors with a PDP11/45 and the Lincoln LDVT, and ISI replaced its signal processor with an FPS AP-120B. Lincoln and ISI also used CVSD vocoding, implemented either with their general purpose signal processors or with special purpose hardware. The ARPA community has implemented a Network Voice Protocol (NVP) for real-time voice communication between several remote sites.

The first design objective of the protocol was to overcome the differences between these systems to allow real-time voice communication between them. In addition to the different hardware systems, different pitch analysis techniques were used. These differences meant that the protocol had to be device-independent. The need to support both LPC and CVSD meant that the vocoding-dependent and the vocoding-independent portions had to be separated. At the time the performance of the network under a number of steady high rate data streams was not well understood, so it was necessary to rely as little as possible on the lower level network transmission protocols, some of which were modified as a result of these experiments.

The basic NVP has 9 different CONTROL messages, which are used for establishing connections, monitoring them, conducting negotiation, and several other control functions.

In addition to the control portion there are several data protocols, each representing a specific vocoding scheme. Each of these data protocols defines the format of the data portion of the data messages, whose headers are defined by the vocoder-independent portion of the LPC.

In addition to these messages there are rules of "good citizenship", comprising a compulsory portion and a recommendation portion; these might differ among the different implementations. This point will be expanded later in this section.

The basic control messages of the NVP are:
⟨1⟩ CALLING (a request for a connection).
⟨2⟩ GOOD-BYE (connection refusal/termination).
⟨3⟩ NEGOTIATION INQUIRY.
⟨4⟩ POSITIVE NEGOTIATION RESPONSE.
⟨5⟩ NEGATIVE NEGOTIATION RESPONSE.
⟨6⟩ I-AM-READY.
⟨7⟩ I-AM-NOT-READY (or HOLD-YOUR-DATA!).
⟨8⟩ INQUIRY (how-are-you?).
⟨9⟩ RINGING.

The normal connection sequence starts when one party calls another by sending a ⟨CALLING⟩ message, requesting the connection. This message carries several parameters, identifying both the exact CALLER and the ANSWERER (like an extension specification). The ANSWERER may choose to accept the call or to reject it. Rejection may occur by ignoring the call or, following the recommendations for good citizenship, by issuing a ⟨GOODBYE⟩ message which contains the rejection code (like BUSY) or not.

In case of acceptance, the ANSWERER starts the negotiation phase, to verify the compatibility of the systems.

The negotiation is the process of achieving a mutual agreement on the setting of various parameters. It is conducted between two parties, one of whom is the NEGOTIATION-MASTER and the other the NEGOTIATION-SLAVE. The master suggests parameter values, and the slave has the option to accept or reject them. This process continues until a set of consistent parameter values is agreed upon. The asymmetric roles of the negotiating parties guarantees the loop-free nature of the process. At any point either system may decide that the negotiation has failed and break the connection, but only the master can declare it successful.

Such a negotiation can be implemented in several ways. It can be implemented by the master sending one parameter value vector, with a complete proposal for all the parameters. It can also be implemented by negotiating the parameters one by one, in any reasonable order. In case of immediate agreement the first way requires the exchange of fewer messages. However, if disagreements are expected, the second way may result in fewer exchanges.

Obviously, different vocoding techniques require that different parameters be negotiated. For example, LPC requires negotiation of 8 parameters, CVSD only 4 parameters. Three of them are common to both techniques (the sampling frequency, the maximum message size, and the "parcel" time duration). The rest are specific to each technique. These common parameters are, in a sense, vocoder-independent. They are included in the negotiation, since the binding of their values may be deferred until run-time.

At the time the protocol was defined, the details of the vocoding techniques had not been finalized, and it was not even clear which techniques would be supported by the protocol. Therefore it was decided to use the second method of negotiation, e.g., item by item.

Since the vocoding details do not change very often, a "shorthand" notation was introduced into the negotiation, the VERSION. Several versions were agreed upon. Each version is a complete description of a vocoding technique and all its associated parameter values. Hence a single agreement on the version can be reached much faster without losing the ability to accommodate different experimental parameter settings.

During the negotiation the master sends a negotiation inquiry message, ⟨3⟩, with several parameters, meaning "CAN-YOU-DO-⟨WHAT⟩-⟨HOW⟩?". For example: "CAN-YOU-DO VOCODING by LPC or CVSD?" Here ⟨WHAT⟩ = VOCODING, and ⟨HOW⟩ = {LPC,CVSD}. The slave then sends either a positive or a negative negotiation response message.

The positive response message, ⟨4⟩, carries with it parameters identifying both the ⟨WHAT⟩ and the ⟨HOW⟩. It is important for the robustness of the system to carry these parameters in order to avoid confusion due to a duplication of such a response, which might cause the master to believe that the slave agreed to some value which it did not actually receive. The negative response message, ⟨5⟩, must carry only the ⟨WHAT⟩. However, as part of the recommended good citizenship, systems are encouraged to send, along with a negative response, their preferred value for this particular ⟨WHAT⟩. It was recommended that masters take advantage of this preferred value, but this was not compulsory. Hence systems which do implement these recommendations can achieve agreements faster than systems which do not. However, each is still compatible with the other.

Being the negotiation master is the privilege of the ANSWERER. It is felt that for the sake of robustness, if any conflicts occur, then the communication should be conducted according to the ANSWERER's preference.

The successful termination of the negotiation is indicated by the master's sending a I-AM-READY message, ⟨6⟩. If a human has to enter the loop, another message is sent to tell the CALLER to wait. This is the RINGING message, ⟨9⟩. If the call is to be answered by an automatic system such as a disk, there is no need for this message.

While the answering system is in the ringing state, it is BUSY, and its resources are committed to the call. Since this is an undesirable state, the system automatically breaks the connection) after a certain time period, unless a HOW-ARE-YOU? message arrives in the meantime. This is repeated until some person answers the phone or until the calling system gives up.

Note that meanwhile the answering system knows who is calling and potentially can inform the human user. This issue, like several other similar ones, is part of the USER-INTERFACE, which is not a part of the intersystem communication protocol.

At any time during the connection each system may inquire about the "health" of the other by sending the HOW-ARE-YOU? inquiry message. To this message the response is either I-AM-READY, or I-AM-NOT-READY. At the beginning only, a RINGING is also an acceptable response to the HOW-ARE-YOU? inquiry. If no answer is received to an inquiry, another inquiry may be sent. After sending several inquiries without getting a READY or NOT-READY response, a system may break the connection, and notify the user about it. Note that there is no real need for the NOT-READY message for this purpose, since a lack of response is equivalent. However, by implementing it, the other system would not flood the network with further inquiries.

When the NOT-READY message is sent as a response to a HOW-ARE-YOU? inquiry, it means HOLD-YOUR-DATA-TEMPORARILY. This is useful when a system gets into a temporary problem (like buffer overflow) and would like to pause momentarily in order to perform some house cleaning operations, before resuming its normal operations.

The period of waiting time between sending an inquiry and assuming that either the message was lost or that the other system coult not respond positively, and the number of re-trials before giving up are important parameters, but their values are not a part of the protocol. Each system may choose different parameters without affecting the compatibility. These values affect only the robustness and the speed with which the system recognizes communication problems and responds to them. The above discussion has covered the 9 basic and compulsory types of CONTROL messages. In addition, there are 4 more CONTROL messages whose implementation is not mandatory, but rather a part of good citizenship.

These nonmandatory CONTROL messages are:
⟨10⟩ ECHO REQUEST.
⟨11⟩ ECHO RESPONSE.
⟨12⟩ RENEGOTIATION REQUEST.
⟨13⟩ RENEGOTIATION APPROVAL.

The ECHO messages, ⟨10⟩ and ⟨11⟩, are used to obtain system-to-system timing information by using a marked echo. This is important for monitoring the network performance in order to adapt to it. Since systems may not necessarily have this dynamic adaptation facility, not all of them issue the ECHO-REQUEST message. It is good citizenship to provide echoes to systems who ask for it, although lack of echoes from a system should never be a reason for breaking a connection.

It was mentioned before that the answering system has the privilege of being the negotiation master. It is sometimes desired that the caller be the negotiation master. In this case the caller may send a renegotiation request message, ⟨12⟩, requesting charge of the negotiations. The answerer uses the renegotiation approval to indicate his acceptance or rejection of this request. No change of "mastership" can never take place without the consent of both parties.

This feature is also used to change parameters during a connection, either for coping with variable network performance (like reducing the number of LPC poles, in order either to restrict the data rate, or to expand it in order to take advantage of the availability of higher communication bandwidth) or for experimental purposes.

In addition, there is obviously also a data portion. Each data message is composed of HEADER and DATA. The header format is part of the NVP, but the format of the data obviously depends on the vocoding technique used.

The header contains addressing information to identify the terminal user (the "extension") and the other parameters as required for the lower level communication protocol. In addition, as a part of the NVP, it also contains a TIME-STAMP, a COUNT and a SILENCE flag. These fields have the CONTROL flavor, but are part of DATA messages since they describe each DATA message. The **time-stamp** corresponds to the time at which the first data contained in the message was recorded. These stamps have no absolute meaning, but the difference between them contains all the needed information about the relative timing of the different portions of the speech. The **count** tells how many successive parcels (vocoding periods) are actually in this message.

The combination of both the **time-stamp** field and the **count** field indicated the relative timing between messages, the order of the messages, the length of each message, and whether some messages are missing. In the case of missing messages, it is also possible to determine the size of the lost portion, which is important for synchronized playback.

There are two reasons for the apparent loss of messages: one is the actual loss of messages due to transmission problems, and the other is silence. During silence periods no data messages are sent. If the lack of communication causes suspicion, control messages are used to verify the state of the connection.

When data continues to arrive after such a break, the receiving system uses different strategies in the case of silence and in the case of lost (or overdue) messages. The single bit SILENCE flag is used to indicate which case exists. The SILENCE flag could be added either at the beginning of the silence period or at its end. For a better error recovery it was chosen to include it at the end of the silence period, together with the first nonsilent data.

The exact format of the ARPA-NVP can be found in reference [5].

Due to the robustness which was achieved by this protocol, no need was found for ERROR messages. However, in certain cases it might be most beneficial to provide the mechanism for error reporting messages and their handling.

6. Higher level voice related protocols

It is difficult to define when a protocol is a HIGH level protocol, or how high is HIGH. However, it is easier to define what HIGHER means. If protocol X uses protocol Y, and Y does not use X, than X is a higher level protocol than Y. In that sense two higher level voice-related protocols are discussed in this section. The first is a real-time CONFERENCE protocol, and the other is a VOICE-FILE protocol,

Both of these protocols require different control structure, in addition to that provided by the NVP, but use the same data protocols as the NVP. Both of these protocols support capabilities which are similar to some found in the telephone system, where any phone which is capable of participating in normal point-to-point voice communications may also participate in voice conferences and can talk to dictating equipment (like the domestic phone answering devices) and can be used to retrieve these recorded messages. Conferencing will be discussed first. In general the idea behind conferencing is to create an environment in which every participant can hear every other participant, just as if they were all in the same room. This kind of voice conferencing is feasible because well-mannered people talk only one at a time. Even if they don't, the human perceptual process can discriminate between two voices, heard at the same time. When even this discrimination fails, a way is usually found to get only one of the participants to talk at a time. Unfortunately, the case of network voice conferencing is not so simple. First, vocoding is not necessarily additive. If only one vocodor is available, but several data streams arrive at the same time, it is not always possible to combine (for example, by addition) the coded speech, then synthesize it, such that a multiple speaker effect is obtained. This is possible only with PCM and DPCM but not with any of the adaptive or acoustic methods. In general, since highly compressed speech depends heavily on a model of a single human voice, and since the sum of several human voices does not follow this model, one should not expect linearity from low data rate systems.

Second, "floor switching" among the speakers is slightly more complex, due to variable network delays. In addition, the inability to interrupt or otherwise indicate a wish to talk is another obstacle.

Therefore the conferencing model requires that all the participants listen to only one source at a time. The conference is controlled by a CHAIRMAN, which is either a person or an automatic program. This chairman must give the floor to the right participant at the right time. There are several ways to distribute the voice data. One way is by concentrating it at some point and redistributing it from there to all the participants. Another is by taking advantage of the distributing capability of the network and sending it directly from each speaker to all the participants.

The first method is much simpler to control than the second. Each participant can communicate with one and only one point, the distribution center, which is controlled by the chairman, even though not necessarily colocated with it. The disadvantage of this method is that it might require more data communication than necessary. Since voice traffic might have a high data rate and require small delays, this can be a major disadvantage. In addition, since the distribution center both receives and transmits voice data at the same time, there is greater danger of congestion there, compared with direct distribution.

Direct distribution requires less data communication, but requires a more complicated control scheme to notify each participant whom to listen to, whom to talk to, and when. Since the control requires a very small bandwidth, a complicated control scheme is a small price to pay for simplified data communication. Therefore, in the ARPA Network Voice Conference Protocol (NVCP) it was decided to use the second method, which leaves the distribution task to the network itself.

In addition to the data flow, there is also control flow. Unlike the data, all the control flows between the participants and the chairman. Since the data rate of the control messages is orders of magnitude lower than that of the data, no problems result.

The control messages are used to support the user interface, e.g., for notifying the participants about a speaker change, notifying participants when they get the floor, when they are about to lose it, and when they do lose it. The participants use the control messages to indicate their desire to speak, and when they are done (which is needed when the chairman is a program, to distinguish between a temporary break and the end of a talk). The participants also use the control path to notify the chairman when they want to interrupt the speaker. This ability is most important in an environment where the feedback to the speaker is blocked. In addition, it is possible to put a question to the conference participants, like "any comments?", or to even take a vote, without polling each participant individually.

In order to improve the system performance, a Local Conference Controller (LCC) was introduced to each site. This LCC serves as a multiplexer/demultiplexer, allowing for a single data stream to represent all the participants from the same site in a given conference. In addition, the LCC handles all the control portion of the NVP, such a negotiation, connection validation, responses to inquiries, etc.

The NVCP is an extension only of the control portion of the NVP, and requires no modification to the vocoding techniques in use. In principle, any vocoding scheme which is implemented for a point-to-point NVP is trivially transferable into the conferencing environment. Obviously, in order to participate in a conference, an LCC has to be implemented, and the user interface has to be supported, to allow the participants to take full advantage of the conference capabilities.

All the NVCP control messages are exchanged between the chairman and the LCCs, which maintain the list of the local participants in the conference. Even though these lists are maintained locally, only the chairman can authorize additions to them.

Here is a summary of the NVCP extensions to the NVP:

[14]	LCC ⇒ CHR	A participant wants to join your conference
[15]	CHR ⇒ LCC	Add a participant
[16]	CHR ⇒ LCC	Remove a participant
[17]	CHR ⇒ LCC	Listen to ⟨new-speaker⟩
[18]	CHR ⇒ LCC	Speak to the following list
[19]	CHR ⇒ LCC	Cease talking now
[20]	CHR ⇒ LCC	Control information for users
[21]	LCC ⇒ CHR	Control information from users
[22]	CHR ⇒ LCC	Invitation to join conference

Another interesting application of the NVP is for storage and retrieval of voice files. In order to support this activity, a special process was coded which uses a storage device (e.g., a disk) instead of a vocoder. In order to make this application more useful, several fields were defined in a voice file, in addition to the voice itself. For sake of simplicity one can treat all of them as TEXT.

When this process is called, it immediately gives up its right to be the negotiation master, and instead proposes that the caller be the master. The caller then starts the negotiation with this recording process, which always accepts any option which is suggested during the negotiation without any understanding of its meaning, as long as it is part of the data specification. However, if an item is a part of the transmission control, such as the maximum message size, it does participate in the negotiation of such an item. Since the storage process is so agreeable, the negotiation is always successfully terminated. At that point the storage process skips the RINGING state, and immediately announces itself READY. All the arriving data messages are stored with the original time stamps, and if they are found to arrive out of order they are sorted. Even messages which are long overdue are not discarded, as in the online case, but are also stored in their appropriate sequence.

Control messages, like inquiries, termination, etc., are not recorded.

When the voice file is to be retrieved, the recording process initiates a negotiation session, according to the parameters which were stored during the original recording session. If the retrieval system cannot handle any of the parameters as stored, the negotiation is terminated (since it is too late to change the recording parameters) and the connection is broken.

In order to handle the recording and retrieval of the voice files, the control portion of the NVP was augmented to include the required changes, like defining the file name and its password, augmenting it with text fields, and retrieving them. Obviously, no change had to be incorporated into the data portion of the NVP, and any vocoding algorithm which is supported by the point-to-point NVP can be used for the voice files operation.

7. Summary

This paper discusses the issues associated with real-time voice communication over packet switched networks. It suggests an approach for the design of a protocol to support this application. The ARPA Network Voice Protocol is presented as an example of such a protocol. In addition, two extensions to it are discussed. Most of the issues which are discussed in the context of the NVP, like separation of control from data, device independence, and performance monitoring are not unique only to voice application but apply equally to any other real-time protocol. The major virtue of the real-time voice communication as far as networking is concerned is the real-time, not the voice, aspect of this communication.

Acknowledgements

I would like to acknowledge the help in preparing this paper which was extended to me by my colleagues, Drs. E. Randolph Cole, Robert M. Balzer, Stephen D. Crocker and Jonathan B. Postel.

References

[1] J.D. Markel, and A.H. Gray, Linear Prediction of Speech, Springer-Verlag, 1976.

[2] D.A. Huffman, (1962) A Method for the Construction of Minimum Redundancy Codes, Proc. IRE, 40, pp. 1098-1101.

[3] J. Ziv, and A. Lempel, A Universal Algorithm for Sequential Data Compression, IEFE Transactions on Information Theory, Vol. IT-22, May 1977, pp. 337-343.

[4] D. Cohen, Issues in Transnet Packetized Voice Communication, Proceedings of the Fifth Data Communications Symposium, Snowbird, Utah, September 1977. (IEFE Catalog Number 77Ch1260-9C) pp. 6-10/13.

[5] D. Cohen, Specification for the Network Voice Protocol, ISI/RR-75-39, USC/Information Sciences Institute, Marina del Rey, CA., March 1976, DDC AD AO23506.

[6] J.L. Flanagan, Speech Analysis, Synthesis and Perception, Springer-Verlag, 1972.