REPORT:

SPEECH PROCESSING APPLICATION STANDARDS

Authors: ***James G. Josenhans, John F. Lynch, Jr., Marian R. Rogers, Richard R. Rosinski,*** *and* ***Wendy P. VanDame*** *are all working at AT&T in areas related to speech processing applications. Mr. Josenhans and Mr. Lynch are members of the technical staff in the Speech Processing Department of AT&T Bell Laboratories in Murray Hill, New Jersey. Mr. Josenhans joined AT&T in 1963. He received a B.S. in physics from the University of Toledo and a Ph.D. in electrical engineering from the Ohio State University. Mr. Lynch joined AT&T in 1979. He received a B.S. in biomedical engineering and an M.S. in electrical engineering from Rutgers—The State University. Marian R. Rogers and Richard R. Rosinski are supervisors in the Applications Architecture and Industrial*
(continued on page 33)

Introduction

The usefulness of voice store and forward products and services depends on the compatibility and interoperability of different systems. This document describes a voice store and forwarding standard that guarantees the compatibility of AT&T products. Current AT&T voice store and forward (VSF) products share common algorithms, and procedures for speech coding, silence detection, and channel compression, based on 16 kilobits per second (kb/s) sub-band speech coding applications. The AT&T VSF standard includes the following components:

1. Voice storage
   - Speech coding algorithm
   - Recording and playback gain
   - Silence detection, coding and noise insertion
   - Channel encoding
   - Speech file format
2. Voice forwarding
   - Message transfer architecture
   - Content description
3. User interfaces

Voice Storage

Voice storage refers to all the processes required to prepare speech input for storage and to permit its interpretation at its destination. This includes speech coding, silence detection, channel encoding, and the later decoding of the speech. Figure 1 shows an overview of voice coding.

The AT&T standard is based on the sub-band coding (SBC) algorithm developed by R. E. Crochiere, S. A. Webber, and J. L. Flanagan.[1] (See also R. E. Crochiere, R. V. Cox, and J. D. Johnston.[2]) The AT&T standard is implemented on the digital signal processing integrated circuit (DSP-20).

Basic Design of the SBC Coder. Sub-band coding (SBC) is a speech compression technique that achieves up to a 4:1 bit-rate compression of stored speech over that of μ-law Pulse Code Modulation (PCM) generally used in the telecommunications network. In this approach, the input speech signal, $s(n)$, is divided into a set of sub-band signals by a filter bank analyzer. These sub-band signals are, in effect, signals that are bandpass-filtered, modulated to dc, and reduced in sampling rate. Each of the banks is encoded with independent APCM (adaptive PCM) encoders and the bits are multiplexed into a single data stream for transmission.

Figures 2a and 2b are block diagrams of the SBC. By carefully selecting the allocation of bits per sample for each of the sub-bands, the sub-band encoder can take advantage of properties of production and perception of speech. In the lower frequency bands, where pitch and formant structure must be accurately preserved, a large number of bits per sample is used. In the upper frequency bands, where fricative and noise-like sounds occur in speech, fewer bits per sample are used. As a result, there is less subjective noise.

The Filter Bank. The filter bank in the encoder processes the input signal before it is passed to the APCM encoders. The filter bank in the decoder is used to combine the decoded

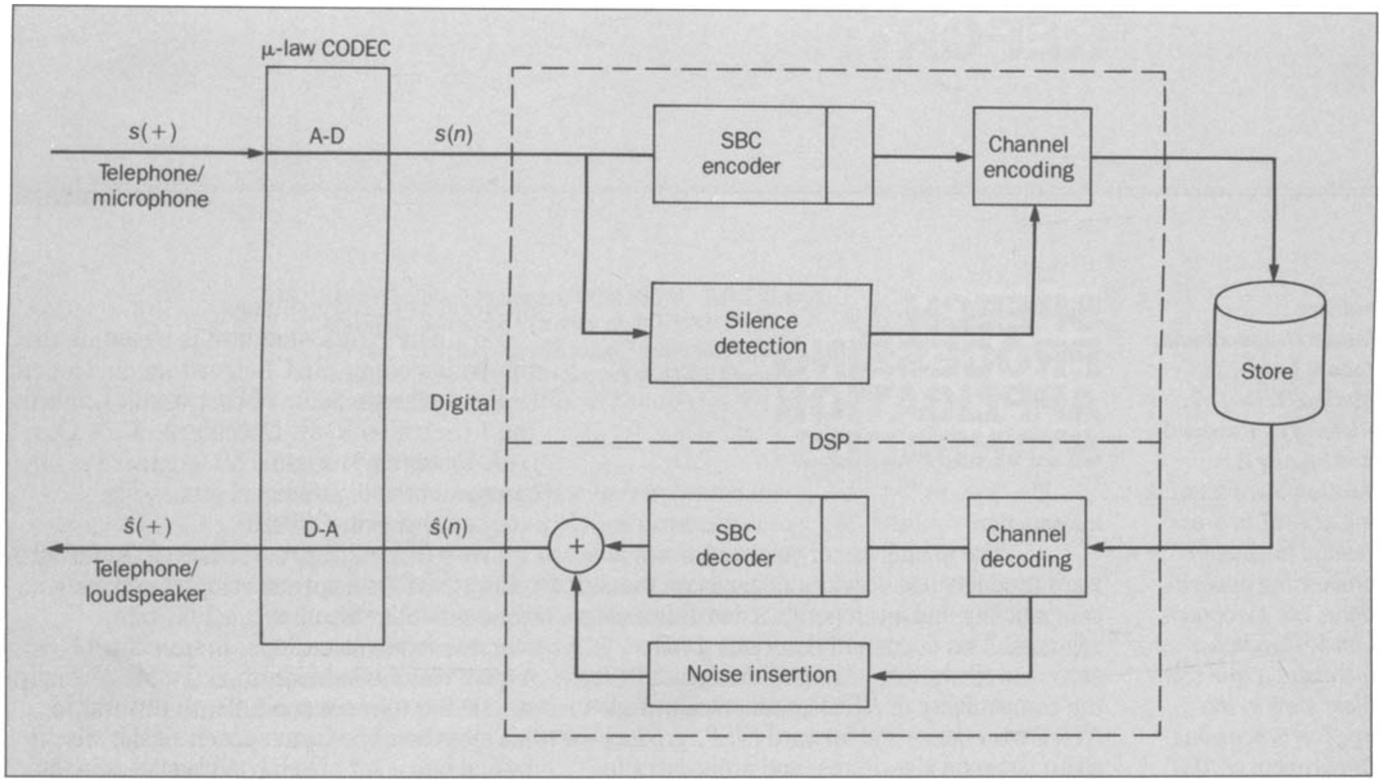


Figure 1. Overview of voice coding.

Table I. Filter Bank Design

Band	Frequency Range kHz	Sampling Rate kHz	Bit Assignment 16 kb/s	Bit Assignment 24 kb/s
1	0.0-0.5	1	4	5
2	.5-1.0	1	4	5
3	1.0-2.0	2	2	4
4	2.0-3.0	2	2	3
5	3.0-4.0	2	0	0

sub-band signals from the APCM decoders to form the output, $\hat{s}(n)$.

The basic computational structure of the sub-band encoder is designed around a five-band, nonuniform quadrature mirror filter bank (QMF). (See Figures 2a and 2b.) This filter bank is based on a tree structure of two-band QMF splits resulting in the choice of bands shown in Table I. Each two-band QMF allows its input signal band to be divided into two equally spaced high-pass and low-pass filtered sub-bands, which are reduced in sampling rate by a factor of two.

Each two-band QMF split is done with a pair of symmetric, finite impulse response (FIR) high-pass and low-pass filters. Because of the symmetry and the quadrature mirror relationship of these two filters, their coefficients are identical except for the signs of the odd-numbered coefficients. Thus, the computation can be shared between the two filters by separately computing the even taps, $h_e(n)$, and the odd taps $h_o(n)$, of the low-pass filter $h(n)$. The sum of these two partial computations gives the output for the lower band. The difference gives the output for the upper band.

Because outputs are required at half of the input sampling rate, the computation of this structure can be distributed over two time slots of the DSP. In cycle 0, the odd taps of the filter, $h_o(n)$, are computed. In cycle 1, the even taps of the filter $h_e(n)$, are computed. The structure leads naturally to implementation of the algorithm as an eight-cycle loop.

In the design of the filter bank, the first QMF split is performed with a 32-tap filter $h(n)$. The second stage uses a 16-tap filter $g(n)$ to divide these two bands into four. Finally the bottom band is divided once more with a similar 16-tap filter $g(n)$. In the receiver, the reverse reconstruction process takes place. A total reconstruction error of less than 0.25 dB is observed in the overall filter bank. The overall group delay of this system when implemented in the DSP is less than 18 milliseconds (ms).

APCM Quantization. Sub-band signals processed by the QMF filter bank are coded by APCM encoders, which then pass the encoded bits to a multiplexer for bit packing. From here, the data is sent into storage. Table I shows the bit allocation (bits/sample) used to encode each of the sub-band signals to obtain an overall bit rate of 16 or 24 kb/s.

Figure 3a shows the main elements of the encoder:

- A b-bit PCM quantizer, where the value of b can be varied from two to five
- Tables to store the step-size and inverse step-size
- A step-size adaptation circuit to control the choice of step-size (i.e., the addresses to the tables)
- A first-order fixed predictor.

As shown in the figures, the input sub-band signal $e(n)$ is adaptively quantized by first scaling it down by the inverse step-size $\nabla(n)$ and then limiting and rounding (PCM encoding) it to the b-bit integer $I(n)$.

By adding the value 2^{b-1} to $I(n)$, a positive b-bit codeword, $J(n)$, is obtained. This can be conveniently multiplexed with codewords from other sub-bands.

The b-bit integer, $I(n)$, is also used for obtaining a decoded output and for step-size control.

The step-size, $\Delta(n)$, and its inverse, $\nabla(n)$, are determined by the table address, which is adaptively incremented and decremented by the step-size control. The tables contain 64 words and span a 60-dB step-size range. Their address is obtained from the integer part of the variable, $d(n)$, which is limited to the range -32 to $+31$.

The value of $d(n)$ is stored in a "leaky" integrator with a leak factor of 0.98. The adaptation leak, γ, is set to 0.98. In this way, $d(n)$ drifts towards zero (the center of the tables) and mitigates the effects of channel error by reducing all parameters to a known state.

If an outermost (positive or negative) quantizer level is used, a positive value, $m(I(n)) = m_1$ is added to $d(n)$. If an innermost (positive or negative) quantizer level is used, a negative value, $m(I(n)) = m_2$ is added to $d(n)$.

In this way, the step-size is dynamically varied to match the range of $e_s(n)$ to the range of the PCM quantizer. Values of $m(I(n))$ are selected for best subjective performance of the sub-band coder and typically result in an adaptation response with a fast attack and slow decay rate of the step-wise amplitude.

Bit Packing and Multiplexing. Computation of the SBC is determined by the three-stage framework of the filter bank. Because the sampling rate is reduced by a factor of two at each stage, computation at different stages must be performed at different rates. This is done by distributing the total computation of the encoder (and decoder) over an eight-cycle process.

In each cycle, one input sample, $s(n)$, is received in the encoder, one path of the filter

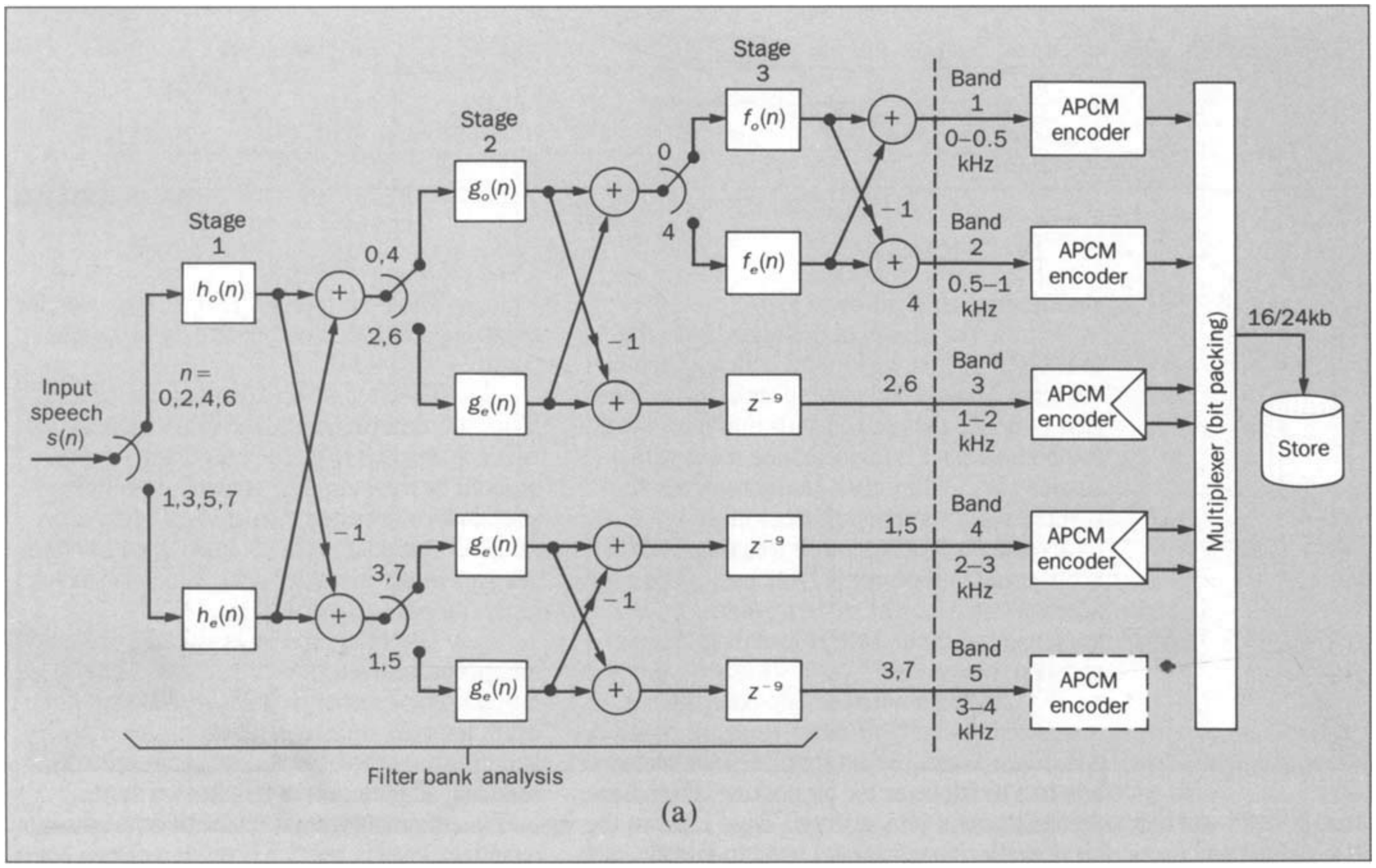


Figure 2a. SBC coder.

bank tree is computed and one APCM encoder is implemented. The cycle numbers $n = 0, 1, 2, \ldots 7$ in Figures 2a and 2b show the paths of computation taken in each cycle.

Stages and encoders in the tree that have higher sampling rates are computed more often than those with lower sampling rates. The tree structure also determines how the output bits from the APCM encoders are multiplexed and framed for transmission.

The total number of bits multiplexed in an eight-cycle process determines the minimum frame size of the multiplexing scheme. Based on the bit assignments shown in Table I, the frame sizes for 16- and 24-kb/s designs are 16 and 24 bits, respectively. The coder is further designed so that the 16-kb/s design is embedded in the 24-kb/s design with the appropriate least significant bits of the 24 kb/s design packed into one byte. By dropping this byte, the 24 kb/s regresses to 16 kb/s compatible with the 16-kb/s design.

For applications involving voice storage in small (16-bit) computers, the 16-kb/s SBC design is particularly convenient because the frame size matches the word size of the computer. It is also convenient in terms of the DSP, which can output words of 16 bits. Thus, the encoder structure in Figures 2a and 2b can be realized by taking in eight samples of *s(n)* and sending out one 16-bit word in each eight-cycle loop. Each of the five bands is attached to the multiplexer via APCMs. Bands 1 and 2 are attached by one APCM each, producing one sample each, while bands 3 and 4 are attached by two APCMs each, producing two samples each, one after a delay.

Adaptive level silence compression. The silence compression algorithm consists of silence detection and encoding in the transmit-

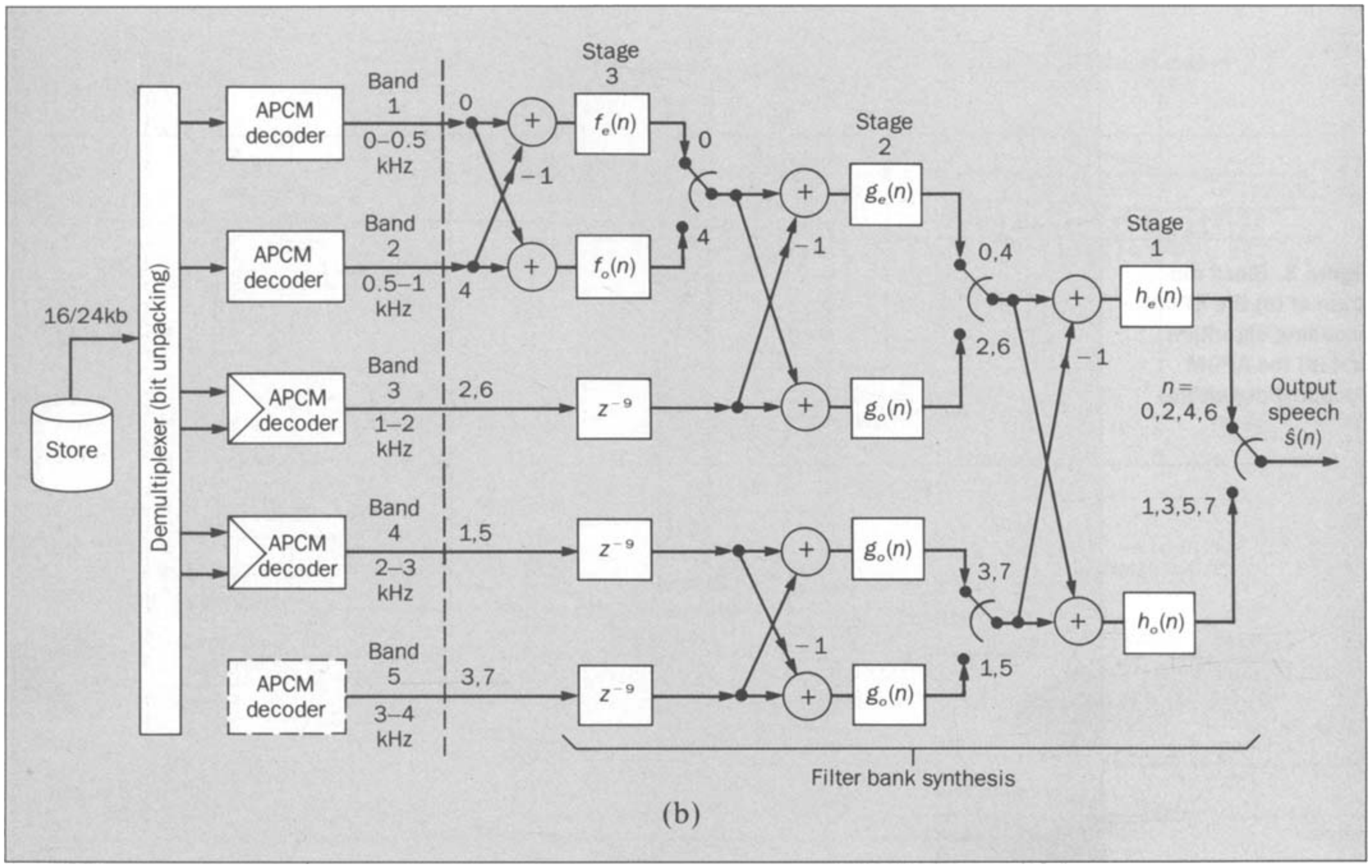


Figure 2b. SBC decoder.

ter, and silence decoding and reconstruction in the receiver. Silence compression occurs in real time and causes no delay in the transmitter or receiver.

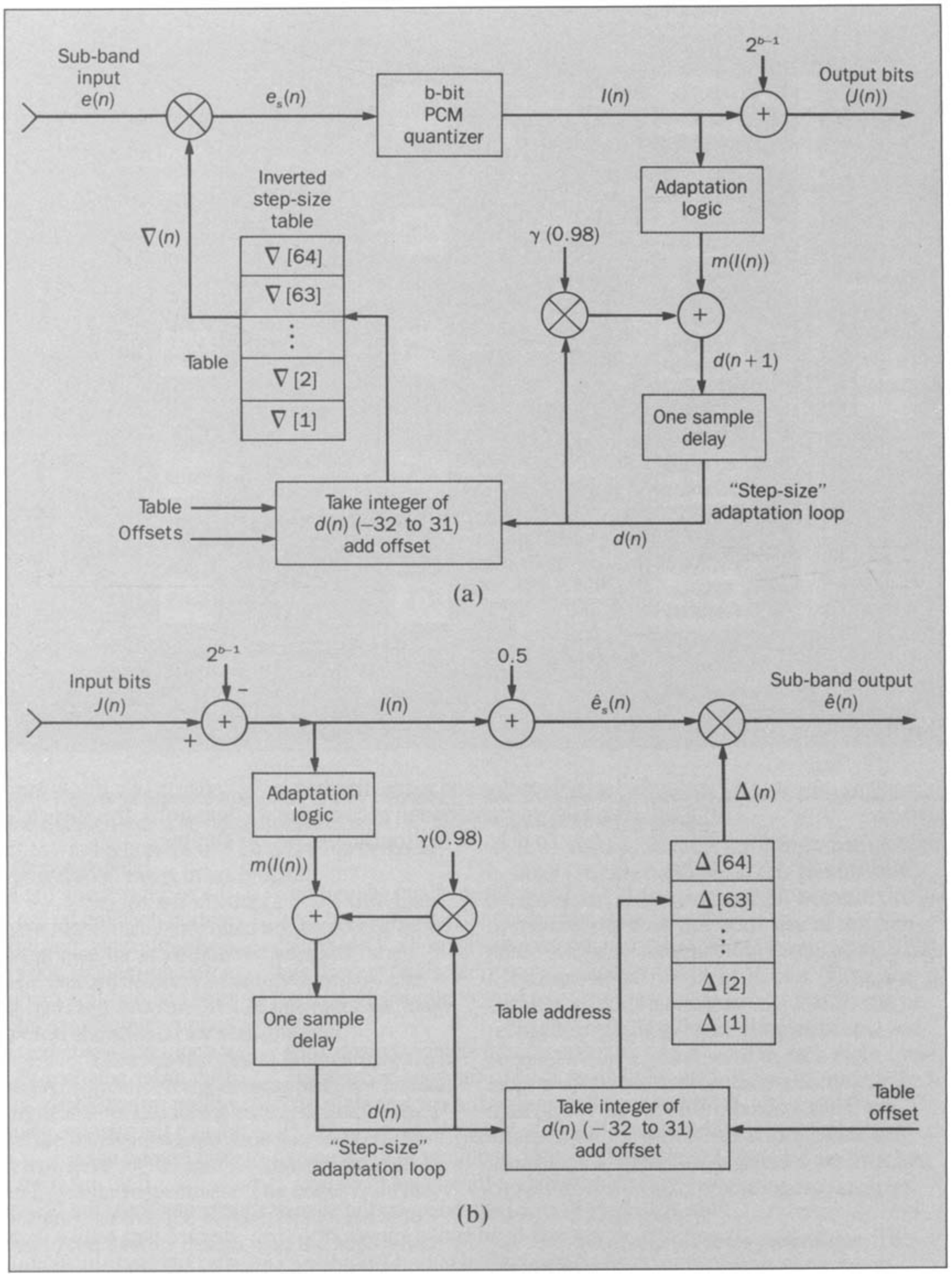
In the transmitter, silence detection uses speech and noise metrics generated by a set of production rules (Figure 4). These rules represent basic assumptions about speech and noise waveforms and are continuously updated by the input waveform.

The speech and silence metrics thus generated are passed to the silence decision logic, which determines the current state of the transmitter. The silence decision logic compares the speech and noise metrics using hysteresis and adaptive thresholds to control the switching speed. Depending on the silence decision, either sub-band coded speech or compressed silence is then output from the transmitter. Compressed silence information consists of a three-word packet containing a key-word, the silence level, and the silence duration.

Silence reconstruction is performed at the receiver using the level and duration information encoded by the transmitter. During silent intervals, noise fill is accomplished using a pseudo-random noise generator that is amplitude and duration modulated as defined by the compressed silence information. The pseudo-random noise is generated in the receiver by an 18th order polynomial[3] giving a 32-second repetition rate at 8 kHz. It is frequency shaped to match the average coloration of acoustical background noise.[4]

If an unknown packet header is encountered, the DSP will output silence and discard data until a meaningful packet header is encountered. In addition, the DSP will output silence for the duration of the time needed to

Figure 3. Block diagram of (a) the APCM encoding algorithm and (b) the APCM decoding algorithm.



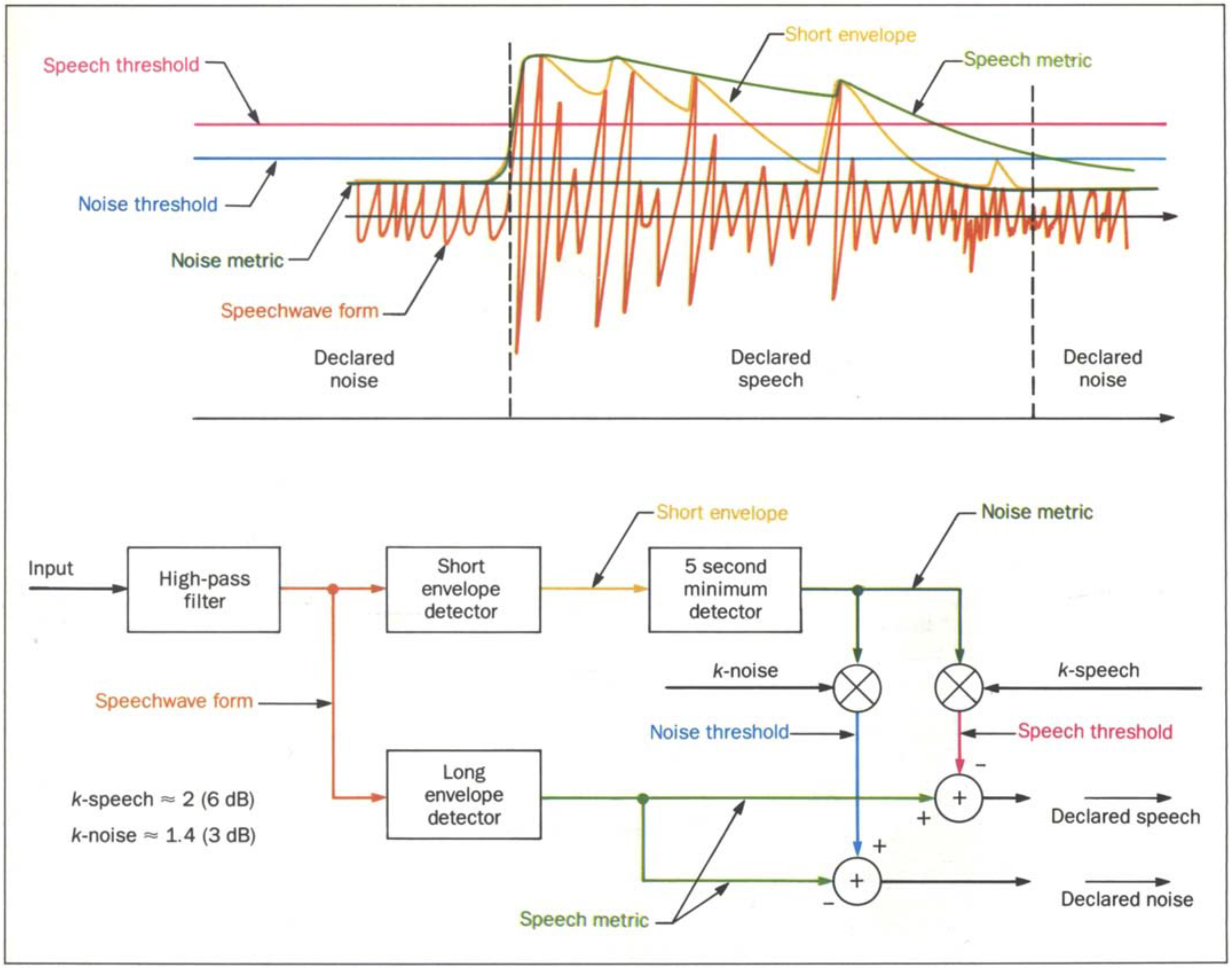


Figure 4. Silence detection.

process the data block.

Channel encoding. All subsequent encoding uses a common control header for silence packets, synchronization, and application-specific information. The same format is used for each of these purposes.

The distinguishing feature of each is an identification (ID) number in the third byte of the header that describes the data to follow. These headers should occur only on word boundaries, but are parsed on byte boundaries for purposes of safety. Subsequent processing of the sound input depends on whether or not silence is present in a given sample.

If it is determined that speech is present, a speech header is used to indicate the onset of speech and the sample is channel coded.

A data stream is produced as output for as long as the *noise measure-noise decision* process indicates that speech is being decoded. At the 16 kb/s rate, 2 bytes/ms are written. At the 24 kb/s rate, 3 bytes are written. A speech header is inserted at regular intervals to allow resynchronization if speech data are lost or corrupted.

The AT&T standard requires a speech header to occur at least once every 256 ms of speech. In voice store and forward, the common header format used for speech and resynchronization is distinguished from other uses of the header by its ID number.

If the noise decision process has determined that silence is present in a given sample, silence is encoded.

VSF products may need to include application-specific data within the speech data stream. The ability to include such information is generally useful, and will provide needed flexibility for future applications of this standard.

File Format. Although byte ordering within a file differs among products, the AT&T standard defines a common byte order to assure that voice mail can be sent between systems without knowledge of various native formats. All products consider voice files as a stream of bytes, using *least significant-most significant* byte ordering. All transmissions between systems use this byte ordering. Systems locally storing files in the reverse order are required to swap bytes before sending speech files to another system and after receiving speech data from another system.

Voice Forwarding

The preceding sections define the subband encoding portion of the AT&T VSF standard. Another major requirement for services compatibility is a provision for common interconnection between different services and equipment. Such capability at the application layer is provided by the *message transfer architecture* (MTA).

AT&T Messaging Architecture. AT&T is committed to supporting the CCITT (International Telegraph and Telephone Consultative Committee) message handling systems (MHS), the international standard for exchanging store-and-forward messages. MHS integrates heterogeneous mail services by defining standard protocols and service definitions for exchanging messages.

The AT&T messaging protocol, MTA, is the initial phase of AT&T's support for MHS. The MTA protocol supports both the architectural model and the message service elements of MHS.

MTA allows any type of object, including voice mail, electronic mail, voice attachments, and voice-annotated text to be sent with ease among mail systems. Most significantly, it provides a way for systems that may not have speech capabilities to receive a voice object and either translate it into an intelligible format or forward it to a voice server or some other device. With MTA, each receiving service determines the appropriate action (e.g., edit, translate, read only, forward) for a particular content type.

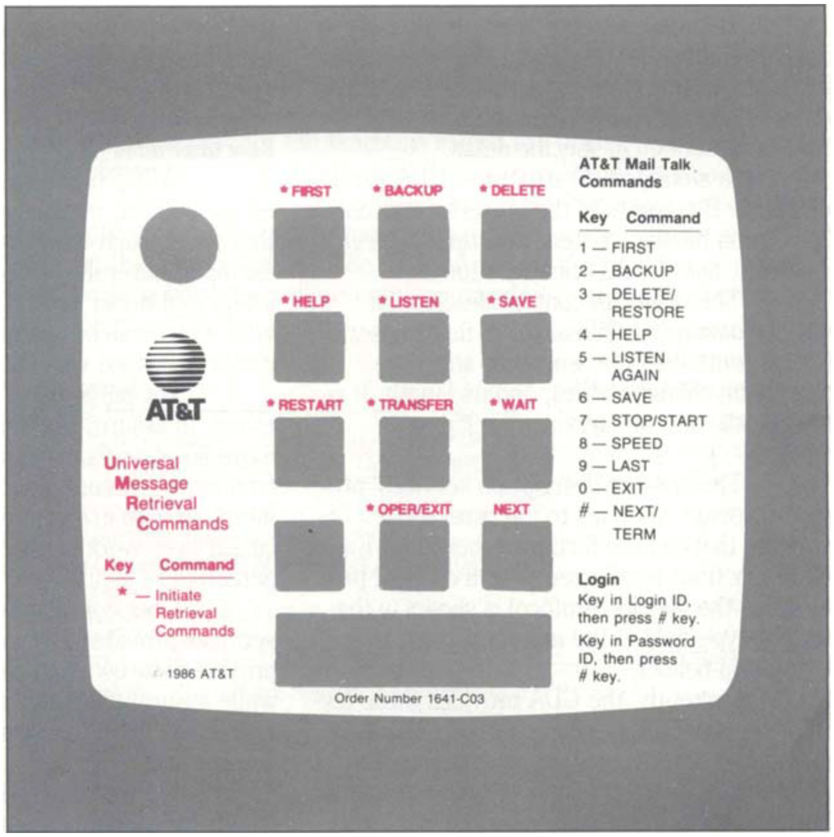
MTA contains two subarchitectures: the message service architecture (MSA) and the content description architecture (CDA).

Message Service Architecture. The message service architecture (MSA) provides the structure and functions required for exchanging information among messaging applications. It contains two hierarchically structured components: the universal message transport header and its associated universal message report header, and subsequent message service protocols.

The transport header serves as an envelope, specifying the information relevant to the transmission of a message.

The universal message report header

Figure 5. AT&T universal message retrieval template.



is at the same architectural level as the universal message transport header. It returns transmission-related status information about messages sent via the universal message transport header.

Each application that uses MSA has one or more message service protocols that perform the functions associated with that service. For example, the mail service uses a message protocol for interpersonal messaging and another for status information. The notification service requires one protocol for message waiting indicators and another for forwarding from an agent-based service to a subscriber's mailbox.

Content Description Architecture. The content description architecture (CDA) is designed to allow the intelligent handling of exchanged information. In this context, "intelligent" means that a receiving user (either a human or an application process) has enough information about an object to adequately determine what can be done to that object (e.g., open, edit, forward, store).

To handle objects intelligently, two conditions must be met. First, objects must be described and identified in a standardized form when being transmitted. Secondly, objects must be described in specific detail.

In architectural structure, CDA parallels MSA. It consists of the universal content description header, content description service protocols, and the described contents.

The universal content description header contains a minimal set of fields describing the content's type, encoding and encryption characteristics, and its length. It is used by all content-carrying messaging services.

The content description services' protocols provide functions to interpret those contents that require further elaboration. Each set of functions is referred to as a content protocol and the specific protocol is shown in the content-type field of the universal content description header.

Currently, the CDA provides three levels of support for describing objects. The first level allows objects that are accepted as standards across AT&T products to be described in the universal content description header by a registered name. For example, a voice file that conforms to the AT&T voice encoding standard may be classified as *Content-Type: AT&T Voice 16.* An object that achieves the status of an AT&T standard is assumed to conform to a structure and format that is well-known and recognizable across AT&T products.

Secondly, objects that are not standard can be transmitted with enough descriptive information about their attributes for receiving systems to determine the appropriate handling mechanisms.

Finally, application-specific objects can be described with registered protocols that are tailored to the requirements of particular services.

User Interfaces

AT&T has several services currently on the market, in development, or in planning that allow touch-tone control by the user. For example, several mail/messaging services are integrated under unified messaging. Each service allows touch-tone user control, and each can be accessed via other services' mailboxes.

For purposes of compatibility, it is necessary to assure: (a) that customers can easily move among products and services, (b) that familiarity with one product or service does not interfere with use of another, and (c) that similar services work in ways the customer perceives as similar.

Core commands for all products and services provide such compatibility. Services provide their own optimal user interfaces, while assuring the user that a core set of commands will work the same way in all products and services.

For example, the AT&T standard commands for touch-tone information retrieval include:

- "*1," which locates the initial item in the current working list and
- "*D (*3)," which DELETES the current item from the list.

Figure 5 is the AT&T universal message retrieval template based on these commands. All single-stroke commands (0-9) are intended as product-specific dial codes.

Command Syntax. Similar combining rules apply to all AT&T products and services that use touch-tone signaling for service control or information retrieval. For example,

" * " functions as a command introducer in all places. All services do not support all of the core commands. However, the standard commands are reserved for the functions shown in Figure 5.

AT&T products and services currently use these core commands or have agreed to support them in the future. The AT&T Architecture Area has issued these core commands as a common AT&T standard.

References

1. R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital Coding of Speech in Sub-bands," *Bell System Technical Journal,* Vol. 55, October 1976, pp. 1069-1085.
2. R. E. Crochiere, R. V. Cox, and J. D. Johnston, "Real-Time Speech Coding," *IEEE Transactions on Communication,* Vol. COM-30, April 1982, pp. 621-634.
3. W. Wesley Peterson and E. J. Weldon, Jr., *Error-Correcting Codes,* Second Edition, The MIT Press, 1972, p. 476.
4. D. F. Hoth, "Room Noise Spectra at Subscribers' Telephone Locations," *Journal of the Acoustical Society of America,* Vol. 12, April 1941, p. 499.

Biographies (continued)

Design Department of Bell Laboratories in Holmdel, New Jersey. Ms. Rogers joined AT&T in 1982. She received a B.A. and a Ph.D. in psychology from the University of Southern California. Mr. Rosinski joined AT&T in 1980. He received a B.A. in experimental psychology from The State University of New York and a Ph.D. in experimental psychology from Cornell University. Wendy VanDame is currently a consultant in the Standards Planning Department of Bell Laboratories in Holmdel. Ms.VanDame received a B.Sc. in economics from the University of Bristol and an M.Sc. in economics from the London School of Economics.

(Manuscript received July 18, 1986)