CCITT Study Group XV
Working Party XV/1
Specialists Group on Coding for Visual Telephony

Doc. # 171
Nov. 1986

Source: PicTel Corp.

Title: Transmission Format for m x 64 kb/s Video Telephony

Abstract

A transmission format suitable for videophone applications at m x 56/64 kb/s is defined. It is based on a packet format, that provides easy synchronization when more than one 56/64 kb/s connection is used. The packet format allows the definition of control packets to handle call setup, call clearing, and transmission of other supervisory information. It also provides a simple means of bit stuffing, when different bit rates need to be interfaced, e.g., when connecting a 56 and 64 kb/s network. The overhead, including forward error correction, is 2.8 percent.

1 INTRODUCTION

The following requirements should be met by the transmission format:

- Reliable operation in presence of channel errors
- Low delay
- Small overhead
- Allow transmission of control information for call setup, call clearing, and other supervisory information.
- Allow the use of n unsynchronized 64 kb/s channels.

The last two requirements are most easily accommodated by a packet format. By giving each packet a sequence number, the bit-stream can be re-assembled in the receiver even if the channels have different delays, slip, or even different bit rates.

Existing link layer protocols (e.g., LAPD for the ISDN D-channel) rely on retransmission of erroneous packets. Due to the delay requirements in video telephony, this is not a viable technique. Therefore, a packet format that incorporates error correction is defined in this document. Encryption is also considered, since many encryption methods give error propagation in case of an uncorrected channel error.

2 PACKET FORMAT

The packet format is illustrated in Figure 1. It consists of 255 octets, each octet being transmitted with the least significant bit first. The sync word is a unique pattern which is used to mark the beginning of the packet. The end of packet is always known because all packets are of fixed length. Bit times between packets if present are filled with ones.

The sync word has the hexadecimal representation 7AC8, where the least significant bit is transmitted first. It was chosen as the 16-bit word that has the maximum autocorrelation peak, i.e., it has properties similar to a Barker sequence.

There are two types of packets: control packets and data packets. The high order bit, i.e., the last transmitted bit, in the control field is set to one to indicate a control packet. In this case, the low order seven bits are interpreted as a control packet type. In the case of a data packet, the seven bits form a sequence number. Sequence numbers of successive data packets are assigned by incrementing modulo 128.

```
+---------------+
|   sync word   |    2 octets
+---------------+
| control field |    1 octet
+---------------+
|   data field  |    248 octets
|       .       |
|       .       |
|       .       |
+---------------+
|     parity    |    4 octets
+---------------+
```

Figure 1. Packet Format.

The forward error correction scheme uses a double error correcting Reed-Solomon code over GF(256). It generates 4 octets of parity, and it allows a maximum blocklength of 255 octets. The data field is chosen to be 248 octets; a multiple of 64 bits gives compatibility with DES encryption.

The parity is calculated based on the contents of the control and data fields. The receiving terminal calculates syndromes based on the control, data and parity fields which will tell whether zero, one, two or more integral octets in the packet were subjected to transmission errors. If only one or two octets are in error, they may be located and corrected. A larger error is uncorrectable.

In the case of the data packet (DT), the data field contains transparent data supplied by the higher layer. The data field is not interpreted by the link layer.

For control packets, the data field contains transparent or protocol related supervisory data. Examples of control packets are packets exchanged during call setup and clearing.

There is also a control packet type to carry transparent supervisory information between the terminals. This packet type is called unnumbered information packets (UI). The data field of the UI packet contains transparent data. In the case of all other control packets, the data field will consist of packet type specific parameter fields.

3 APPLICATION OF THE PACKET FORMAT

Before a video session starts, a connection at the physical layer is established by in-band or out-of-band signaling.

For a point to point connection, there are three phases to the video session: call setup, data transfer, and call clearing. Control packets are exchanged in the call setup and clearing phases. The detailed procedures for call setup and clearing are not treated in this document.

Sequenced data packets are used in the data transfer phase to carry transparent video, digital audio and computer data supplied by the higher layer between the terminals. If packet dropout or uncorrectable errors are encountered, the higher layer is alerted which can then undertake error recovery transparently to the link layer.

For a multipoint video session there is no call setup or call clearing phase. Each terminal sends sequenced data packets and intermittent control packets which identify the terminal. Channel control arbitration is implemented at a higher layer.

3.1 Procedure For Sequenced Data Transfer

When the data transfer phase is entered, the sending terminal may commence transmission of sequenced data packets. Transparent video session data is buffered into the format free data field in the packet. The data field must be completely filled. If encryption was selected for the video session, the data field will be encrypted. The packet is tagged with a sequence number and sent out on the channel with the least number of buffered packets. The first packet to be sent has sequence number zero. Successively sent packets are marked with sequence numbers which are incremented by one modulo 128.

The receiving terminal will perform error correction and, for the multi-channel case, reorder incoming packets via the send sequence number. As there may be a delay between channels, packets from one channel may have to be buffered temporarily before processing. Packets are eventually processed by conditionally decrypting and then depacketizing the contents of the packet data field. The transparent video session data is sent to the higher layer.

If uncorrectable receive errors are encountered or packet dropout is detected via a missing send sequence number, any buffered receive packets are discarded and the higher layer is signalled. The receiving terminal will then just start looking for the next packet in the receive stream without regard for the sequence number. Care must be taken in the multi-channel case, as the new next packet to receive can be delayed in transmission. Note that first sequence number expected upon entering the data phase is zero.

3.2 Procedure For Unsequenced Data Transfer.

After the data transfer phase is entered, the sending terminal may at any time inject a UI packet into the transmit packet stream. This packet will contain transparent supervisory information supplied by the higher layer within the packet data field. If encryption was selected for the video session, the data field will be encrypted. The UI packet may be sent out on either channel.

The receiving terminal will discard the received UI packet if any errors are detected. Otherwise, the contents of the data field are conditionally decrypted, depacketized and sent to the higher layer.

4 ENCRYPTION

If encryption is in effect for the video session, it is applied to the data field of DT and UI packets. Other control packets than UI are not encrypted at all.

A possible encryption method is the DES CBC (cipher block chain) algorithm. This is a variant of ECB (electronic code book). In ECB, each successive 64 bit sequence is encrypted via a one-to-one transformation into another 64 bit sequence. In CBC however, each 64 bit sequence except the first in each packet is first OR'ed with the result of the previous operation. Thus in general, two identical 64 bit input values will be coded differently, making the task of codebreaking more difficult. Since the CBC is restarted in each packet, error propagation between packets do not occur due to the encryption.

5 IMPLEMENTATION NOTES ON REED-SOLOMON CODES

A double error correcting Reed-Solomon code combines good performance with a reasonably simple implementation.

With a code defined over GF(256), each symbol consists of 8 bits. This means that two erroneous octets can be corrected. Hence, a 9-bit burst error can always be corrected.

5.1 Encoder

In the encoder, the parity symbols are computed from the information symbols as outlined in Figure 2. Essentially, a polynomial division is performed by the circuit. Each lookup table performs a multiply by a constant. The registers are initialized to zero at the beginning of a block. At the end of the block, the parity symbols are clocked out from the registers.

Four table look-ups and four XOR operations are performed per symbol. At 64 kb/s, this corresponds to 32000 table look-ups and 32000 XORs per second. The functions can be implemented in hardware or software.

5.2 Decoder

The decoder starts by calculating syndromes according to Figure 3. At the beginning of the block, the registers are initialized to zero. At the end of the block, the four syndromes are available in the registers. The number of operations is the same as for the encoding, and the implementation can be in hardware or software.

If the syndromes are not zero, one or more symbols were erroneous. The errors are found by the following steps:

1. Generate an error locator polynomial.

2. Find the zeroes of the error locator polynomial to obtain the error locations.

3. Calculate the error values.

For double-error correction, these steps correspond to less than 150 operations per block (< 5000 op/s at 64 kb/s). They are most easily implemented in software.

Reference: R E Blahut, Theory and Practice of Error Control Codes, Addison-Wesley 1983.

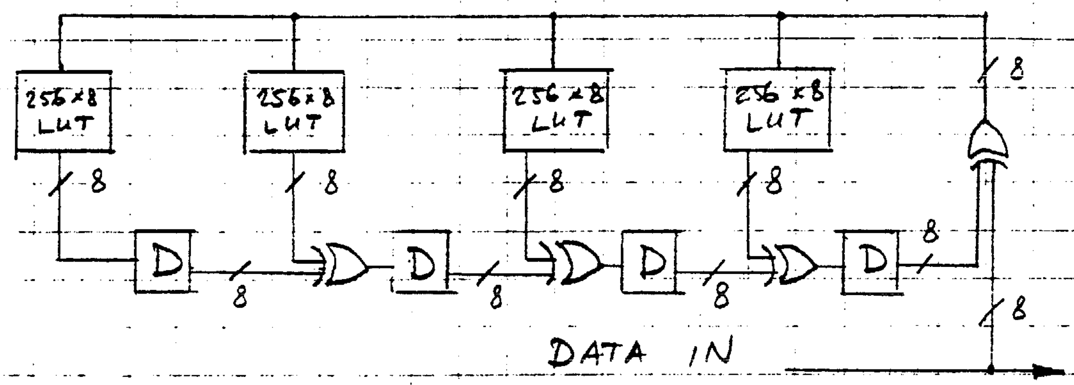

Figure 2. Parity symbol generation

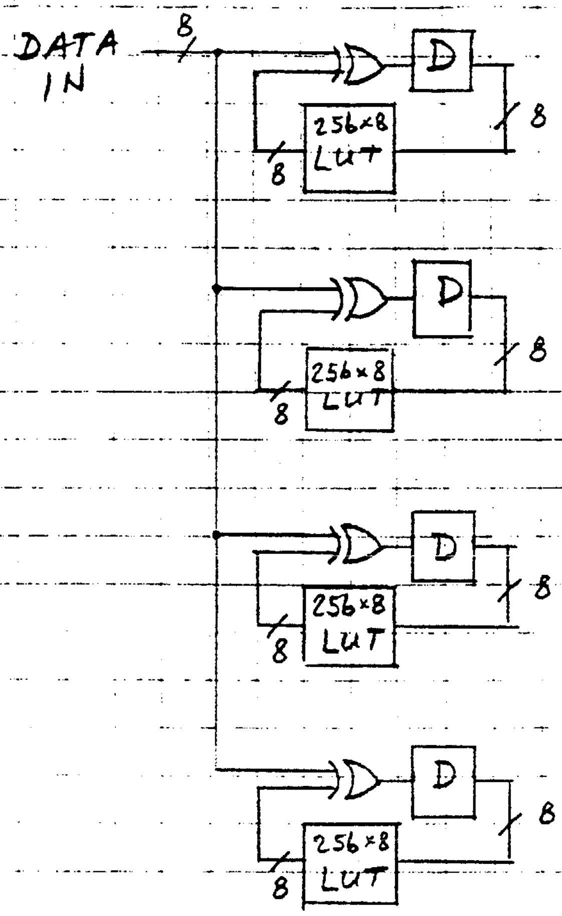

Figure 3. Syndrome generation