Telecommunication Standardization Sector
Study Group 15
Experts Group for ATM Video Coding
(Rapporteur's Group on Part of Q.2/15)

Document AVC-501
May 1993

INTERNATIONAL TELECOMMUNICATION UNION
TELECOMMUNICATION STANDARDIZATION SECTOR
STUDY PERIOD 1993-1996

COM 15- 27
May 1993
Original: English

Question: 3/15

STUDY GROUP 15 - CONTRIBUTION

SOURCE: BELLCORE

TITLE SIMULATION STUDIES OF 4 QCIF TO CIF VIDEO MIXING

Abstract

Attached is a Bellcore paper on H.261 video bridging that is being submitted to the IEEE Transaction on Circuits and Systems for Video Technology and the SPIE Visual Communications and Image Processing Symposium. This paper contains simulation studies of the 4 QCIF to CIF video mixing scheme proposed in an accompanying contribution.

Video Bridging Based on H.261 Standard

Ting-Chung Chen, Shaw-Min Lei, and Ming-Ting Sun

April 14, 1993

Abstract

Multi-point ISDN videoconferencing with video bridging in network-based servers represents a viable network service and is a new revenue source for network carriers. This paper presents a technical analysis of a continuous presence video bridge using the H.261 video coding standard. We first compare the pros and cons of coded domain versus pel domain video bridges. The architecture and the required operations of a coded domain bridge using H.261 are then investigated. We derive the bounds of the bridge delay and the required buffer size for the implementation of the bridge. The delay and the buffer occupancy of the video bridge depend on the order, complexity, and the bit-distribution of the input video sources. To investigate a typical case, we simulate the delay and the buffer occupancy of a video bridge. We also provide a heuristic method to estimate the delay in a typical case. Several techniques are discussed to minimize the bridge delay and the buffer size. Finally, we simulate an intra slice coding scheme and show that the delay and the buffer size can be reduced significantly using this technique.

1 Introduction

Multi-point videoconferencing is a natural evolution of two-point videoconferencing. To establish multi-point connections, coded video streams from the participants are sent to a Multi-point Control Unit (MCU). In a "switched presence" MCU, either a signal selected by the conference chairman or a signal selected based on audio channel activity is broadcast to all participants[1]. In a "continuous presence" MCU, multiple video streams from the participants are combined so that each participant can see multiple participants all the time. The "switched presence" MCU has been standardized recently[2, 3] while the "continuous presence" MCU is still under active research. In this paper, we focus on video combining in the "continuous presence" MCU to support network-based multi-point multimedia videoconferencing.

In the MCU, multiple coded video sources can be decoded, combined in the pixel domain, and then encoded for distribution. In some special cases, however, the video sources can be combined directly in the coded domain without transcoding. Video combining in the coded domain offers shorter end-to-end delay, better picture quality, additional security capability, and lower MCU cost. Bellcore has proposed to operate the user terminal in an asymmetric mode so that the MCU can combine four QCIF (Quarter Common Intermediate Format) H.261[4] coded videos into one CIF video in the coded domain for continuous presence multi-point videoconferencing. A QCIF video combiner will provide users a continuous presence view of up to four conferees at one time. Although the concept of the QCIF combiner is simple, many implementation issues need to be studied in more detail. The issues that need to be answered include the combiner architecture, required size of buffers, end-to-end delay, picture quality, and system complexity.

Two examples of continuous presence applications are shown in Fig. 1. In Fig. 1(a), conferees are involved in a multi-way videoconference. QCIF videos from the conferees are transmitted to

	Pel Domain	DCT Domain	VLC Domain
Delay	long	long	short
Buffer Size	large	large	small
Complexity	high	medium	low
Flexibility	high	medium	low
Degradation	more	a little	none

Table 1: Qualitative Comparisons between Three Approaches

the MCU with a transmission rate R. For each conferee, the MCU selects and combines four QCIF videos into a CIF video and transmits it back to the conferee with a transmission rate 4R. Fig. 1(b) is a remote classroom application where the MCU combines 4 QCIF remote-site videos into a CIF video so that the teacher can interact with 4 remote sites simultaneously. The video from the teacher is broadcast to all the remote students.

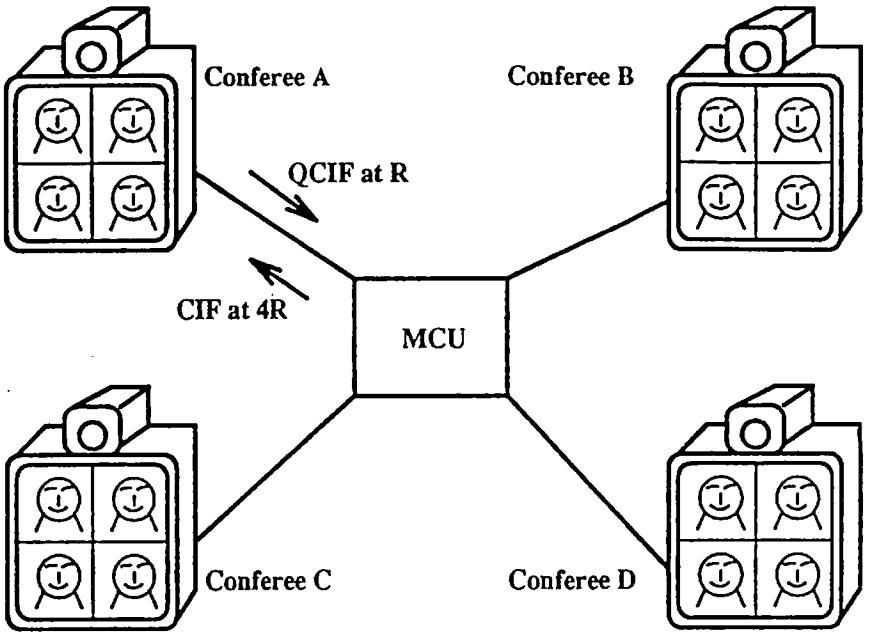
The organization of this paper is as follows. Section 2 describes and compares pixel-domain and coded-domain combining. The rest sections are more focused on the coded-domain QCIF combiner. Section 3 describes the architectures of the QCIF combiner. Section 4 presents theoretical analysis for the delay and buffer size. Section 5 shows some simulation results. Section 6 discusses techniques to improve the end-to-end delay. Finally, conclusions are provided in Section 7.

2 Coded-Domain vs. Pel-Domain Video Combining

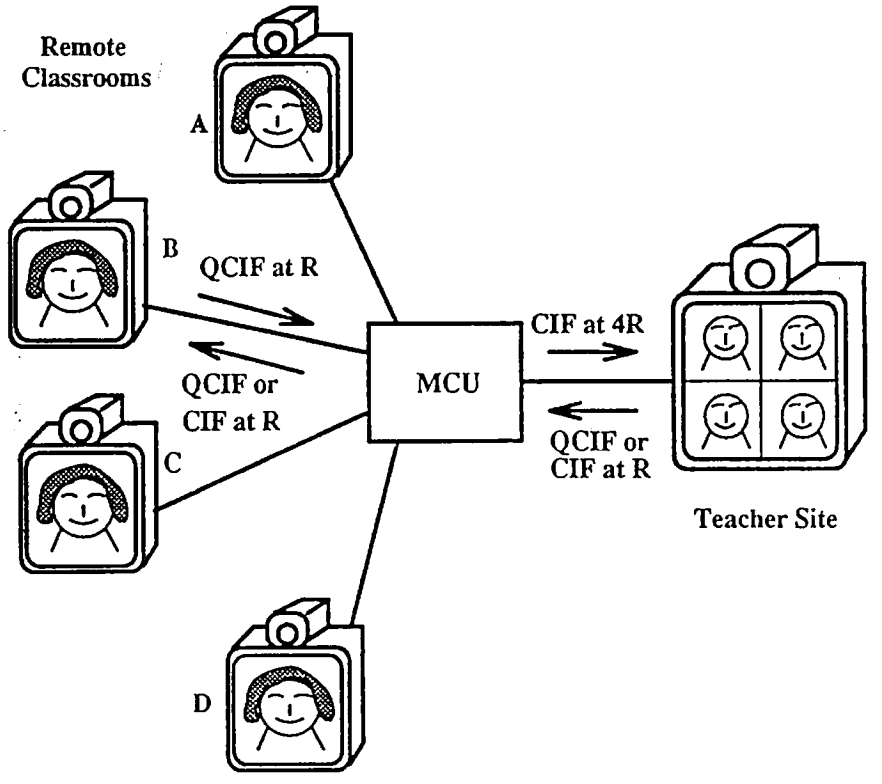
Since the raw video data rate is large and the network bandwidth is limited, video data usually have to be compressed for network transmission. Therefore, video bridging in the network normally requires video decoding, combining in the pel domain, and encoding for retransmission. For some coded bit streams, however, video combining can be done in the coded domain. We define coded-domain video combining as a process that does not decode the compressed data down to the pel domain. The compressed data are either not decoded at all or only partially decoded for combining. In the not decoded case, the video bridge only has to process data header and concatenate the remaining data stream without modification, e.g., the proposed QCIF combiner. In this case, we are mainly dealing with data coded by variable-length codes (VLCs). We will refer to this case by *VLC-domain approach*. In the partially decoded case, video bridging is usually done after decoding variable-length codes, e.g., DCT-domain video compositing[5]. For pel domain video combining, the compressed data have to be fully decoded and then, after combining in the pel domain, encoded again. Table 1 gives a qualitative summary of the pros and cons of these three approaches and we will discuss further in the following subsections.

2.1 Delay

It can be observed (see Sec. 4) that the buffers used to smooth out the variable-rate bit stream contribute a major delay, compared to other delays such as processing delay and transmission delay. In the pel-domain approach and the DCT-domain approach, this rate-smoothing delay has to be incurred twice due to repetitive VLC decoding and encoding. The VLC-domain approach (or QCIF combiner) only incurs such delay once so its total delay can be shorter. The delay issue will be discussed in detail in Sec. 4 and a quantitative comparison of delays in the VLC-domain and pel-domain combining will be provided in Sec. 5.



(a)



(b)

Figure 1: Two Examples of the QCIF-Combining MCU Applications

2.2 Buffer Size

In the pel-domain approach and the DCT-domain approach, the video bridge needs one buffer for each input signal and another buffer for the output variable-rate coded data (assuming only one output). However, the QCIF combiner (the VLC-domain approach) needs only buffers for inputs. It can be seen in Sec. 5 that the input buffers for different approaches all have a comparable size. If the output buffer has the size of all input buffers summed together, the QCIF combiner can save half of the memory by saving the output buffer, assuming only one output. If multiple different outputs are needed the saving is even more significant.

2.3 Complexity and Flexibility

The video bridge using the pel-domain approach has the highest complexity among the three approaches since a full decoder is needed for each input and a full encoder is needed for the output. The complexity of a QCIF combiner is the lowest since only data headers have to be modified. The detailed architecture of a QCIF combiner will be discussed in Sec. 3. The complexity of a video bridge using DCT-domain approach is in the middle. It requires VLC decoders and a VLC encoder, but not full video decoders and encoder.

Although the pel-domain approach has the highest complexity, it certainly has the highest flexibility. In the pel domain, video can be manipulated pel by pel. Different coding standards or algorithms can be accommodated and transcoding is straight forward. In constrast, video combining in the VLC domain is very restricted. Only those segments which can be identified by "clear" codewords[1] can be accessed as a whole. For example, in H.261 standard, the smallest unit that can be accessed without VLC decoding is GOB (group of blocks). We can only put together GOBs to form a CIF or QCIF picture. The flexibility of the DCT-domain approach is once again in the middle.

2.4 Picture Quality

One concern for the pel-domain approach is that the repetitive encoding and decoding processes may degrade the final picture quality due to the numerical inaccuracy in motion compensation, DCT, quantization, and their inverse operations. In contrast to this, the VLC-domain approach can fully preserve the transmitted picture quality. For the DCT-domain approach, the picture quality can also be preserved if no requantization is needed, otherwise the picture quality would degrade a little, but should be better than that of the pel-domain approach.

To investigate how much degradation a cascade of regular pel-domain combining bridge may introduce, we did two simulations using the H.261 video standard[4]. In the first simulation, we assume the macroblock boundary of the pictures does not change in combining. The sequence "Miss America" with a CIF-format was used as the input to the first H.261 codec, and the codec output was used as the input of the next codec. A total of four stages were cascaded, where all four codecs were set to have the identical coding rate, buffer size, and quantization step-size adjustment intervals. Fig. 2 shows the peak-signal-to-noise-ratios (PSNR's) for the four successive decoded outputs. In can be seen that the first frame has the same PSNR for all four stages because it is intra-frame coded at identical block boundaries and no motion-compensation reference is used. Starting from the second frame, the four decoded outputs have a spreaded PSNR's, with a difference of approximately 1 dB for four stages. Since the accuracy of motion compensation depends on the accuracy of the reference and since we use a high rate coding (approximately 512 kbps in this case),

[1] A "clear" codeword is a codeword with a special pattern which cannot be formed by any concatenations of other VLC codewords. Thus, it can be recognized without VLC decoding.

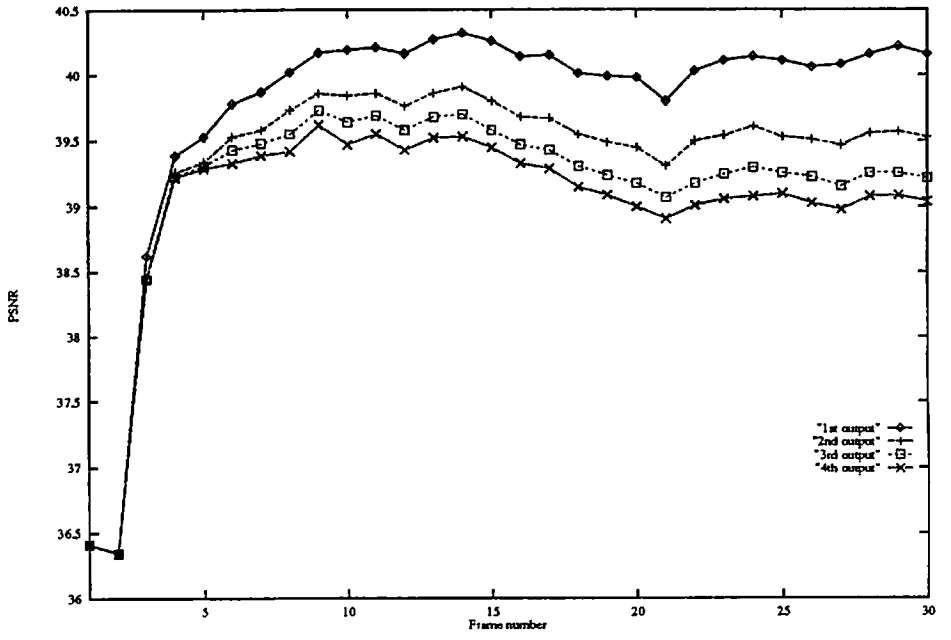


Figure 2: PSNR's for 4 successively coded 'Miss America' at 512 kbps

the visual picture quality does not degrade too much in this case. Indeed, under careful scrutiny we could not see a visual difference between the first and the fourth output. Similarly, when we decrease the coding rate down to about 128 kbps, the visual quality and the PSNR spread is also quite insignificant as shown in Fig. 3.

In the second simulation, we allowed the picture's macroblock boundary to change in combining. This considers the fact that in the pel-domain combining the boundary may not be identical. A two-pel and two-line increment was introduced for each of the first three stage outputs, and we compare only the overlapped portion of the four outputs. Fig. 4 shows the PSNR spread, which is about 1 dB larger than Fig. 2. The first frames also show different PSNR's because the block boundary was changed from stage to stage here.

From the above results it can be seen that although the pel-domain combining results in slightly degraded picture quality, the degradation is insignificant and almost undetectable subjectively.

3 Architecture and Operation of the QCIF Combiner

The rest of this paper will focus more on the QCIF combiner. This section will discuss the architecture and operation of the QCIF combiner. Hereafter, the coded-domain approach refers to the VLC-domain approach only, not the DCT-domain approach.

3.1 Architecture of the QCIF Combiner

The primary task of the QCIF combiner is to multiplex variable-length coded GOBs into a single bit stream. It is easy to understand that a buffer is needed for each input to accommodate variable-length GOBs and to hold them until they are transmitted. The basic architecture of the QCIF combiner is shown in the central portion of Fig. 5, which consists of four buffers and an intelligent multiplexer. Note that an output buffer is not required since the data can be held in the input

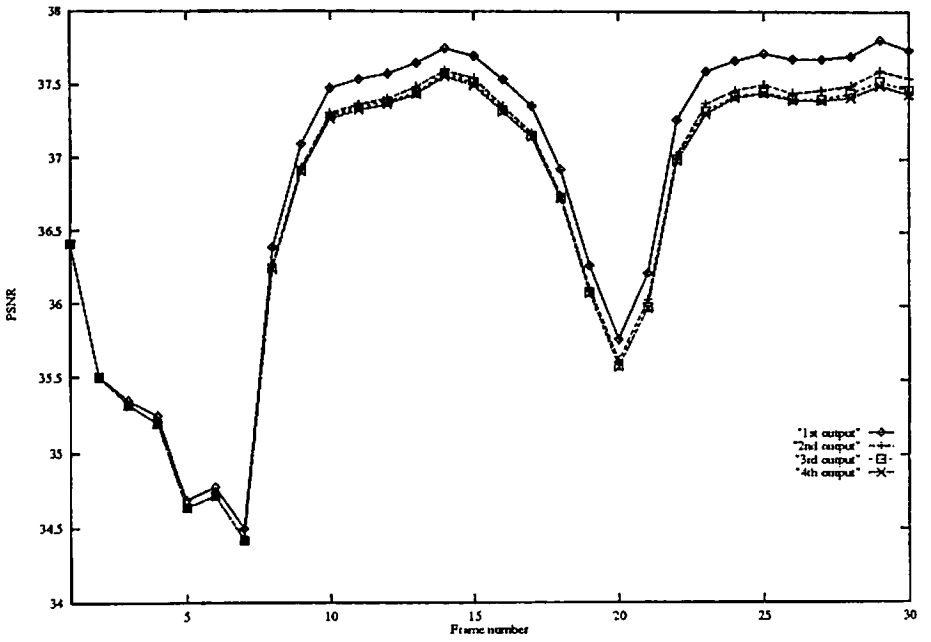


Figure 3: PSNR's for 4 successively coded 'Miss America' at 128 kbps

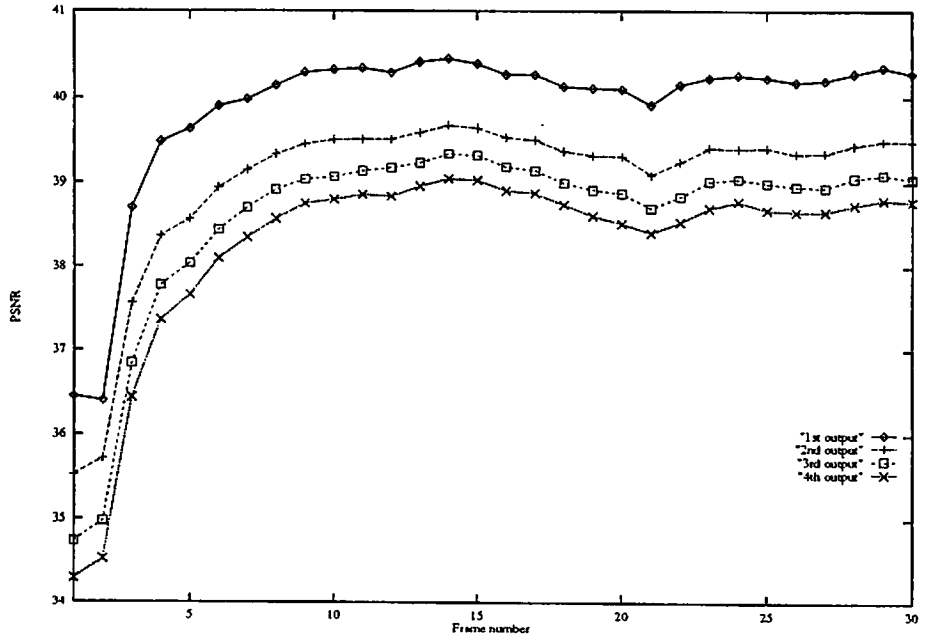


Figure 4: PSNR's for offset input sources, codecs set as in Fig. 2

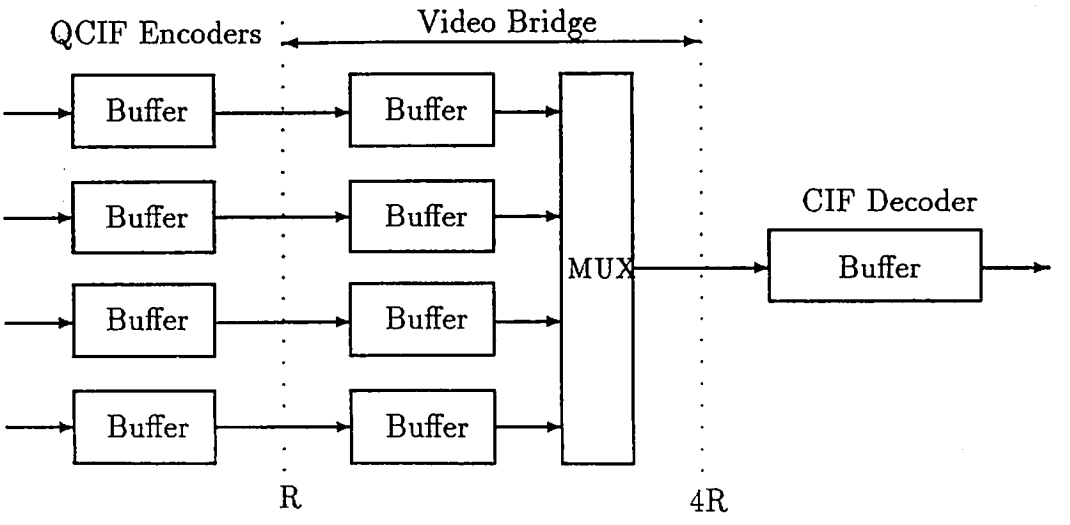


Figure 5: QCIF Combining System

buffers. Also note that each input rate is at R but the output rate is at $4R$. For teleconferencing applications as shown in Fig. 1(a), this would imply that an asymmetric connection between each conferee and MCU is required. Using the currently available ISDN standards, however, symmetrical links must be used. Each user has to reserve a $4R$ duplex channel and restrain their actually transmitted rate at R by bit stuffing or some other means. There are two possible ways to do bit stuffing within H.261: one is the *fill frame* specified under error correction frame, and the other is MBA (MicroBlock Address) stuffing. However, the MBA stuffing is variable-length coded. Only fill error correction frame can be easily detected and discarded by MCU without variable-length decoding.

In a more generic sense, a QCIF combining MCU can also offer a service which provides each user a customized continuous presence view of up to four conferees. A general architecture for such a service is shown in Fig. 6, where we assume that there are M different inputs and N different customized outputs. Each input data stream first undergoes preprocessing which includes end-to-end signaling, H.221 demultiplexing/control, forward error correction and the removal of redundant bits. The data after preprocessing are then broadcast to N memories, each corresponding to an output and used as a buffer. The multiplexers will combine the outputs according to users' selections or some control algorithm. The postprocessor does the inverse of the preprocessor, which includes bit stuffing, H.221 multiplexing/control, and forward error correction. Note that even though $M \times N$ memory units are shown here, only $4N$ memory units are actually required for the QCIF combiner. Fig. 7 shows a cross-bar switch which reduces the memory requirement to $4N$ units.

3.2 Operation of the QCIF Combiner

A convenient multiplexing unit for the QCIF combiner is the group-of-block (GOB). A GOB has a clear code delimiter that can be detected by the preprocessor without doing variable length decoding. In the H.261 syntax, only the picture layer and the GOB layer have clear codes. GOB layer multiplexing can render shorter delay than picture layer multiplexing and is thus the preferred approach.

To combine four QCIF coded data streams, the processor has to detect picture and GOB headers

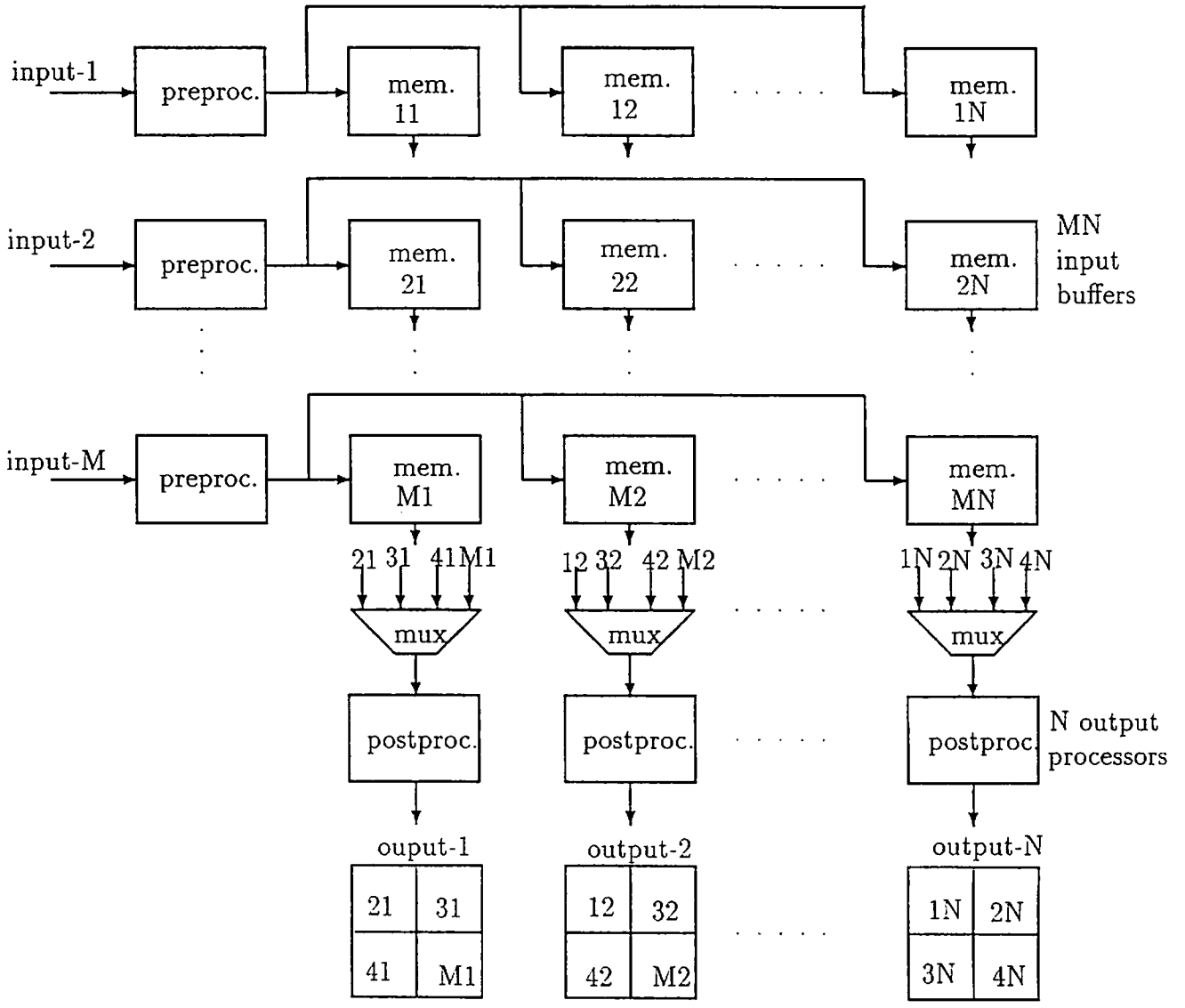


Figure 6: A General Architecture for a QCIF Combiner

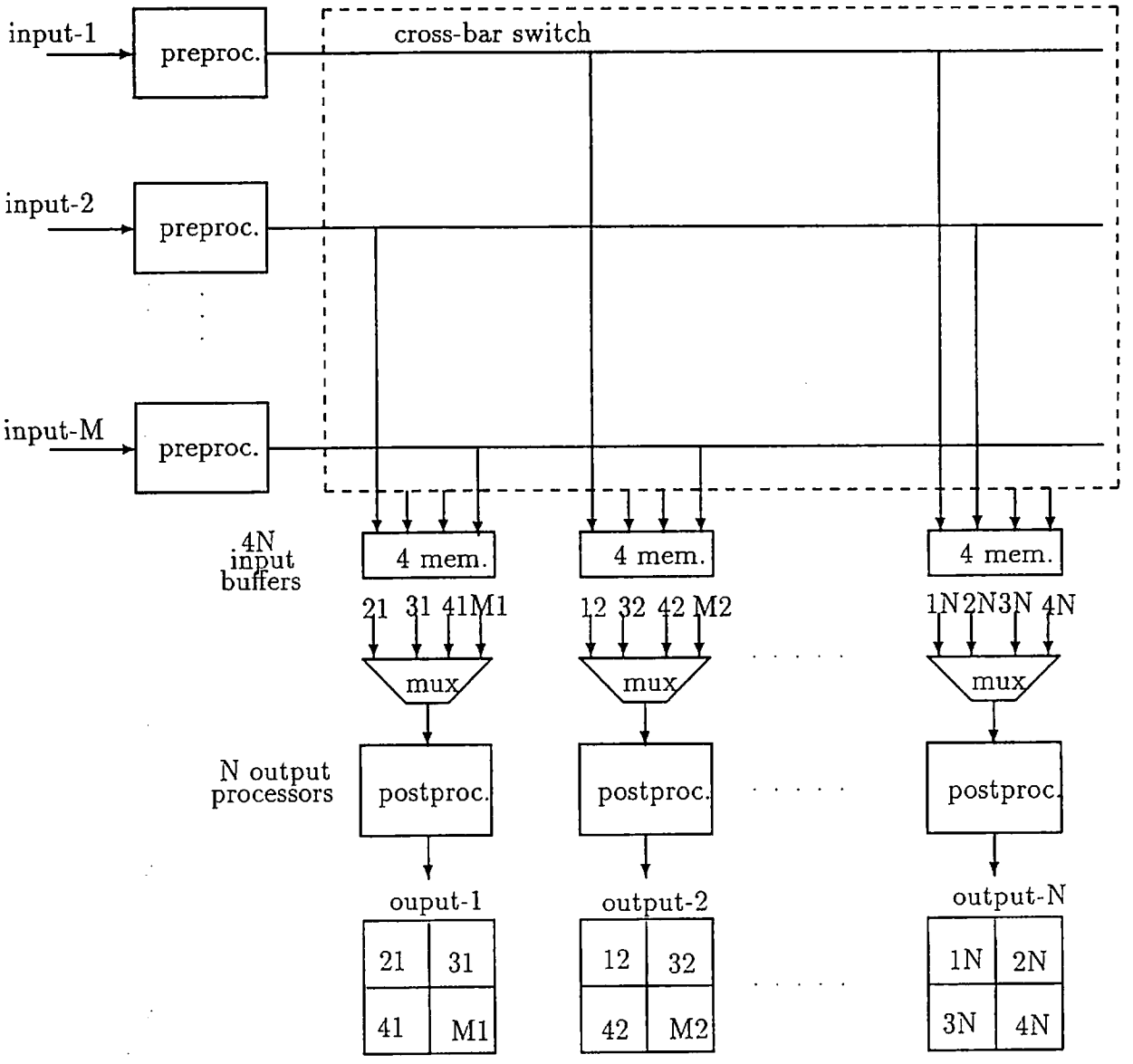


Figure 7: A Cross-bar Switch Reduces Input Memory to 4N Units

QCIF I		QCIF II		QCIF III		QCIF IV	
GN_I	GN_O	GN_I	GN_O	GN_I	GN_O	GN_I	GN_O
1	1	1	2	1	7	1	8
3	3	3	4	3	9	3	10
5	5	5	6	5	11	5	12

GN_I: The Group Number of the input QCIF.
GN_O: The Group Number of the output CIF.

Table 2: The Mapping of GNs

and then relabel these headers. The following lists some changes that have to be made by the MCU multiplexer.

1. **Picture Layer:** Only the picture headers of the first QCIF are retained. The other three QCIF picture headers are discarded.

   TR: Temporal reference for the CIF can be kept to be the same as the first QCIF's TR.

   PTYPE: Bit-4 of the PTYPE (totally six bits) is changed to "1" to indicate "CIF"

2. **GOB Layer:**

   GN: Group number has to be converted from (1,3,5) to (1 to 12) according to Table 2.

   GQUANT: The quantizer step-size of the group is kept unchanged.

3. **Macroblock layers and block layers** are unchanged and directly multiplexed after their GOB headers.

Bit stuffing may be needed to fill up the output stream and to maintain a constant-rate output channel. Such bit stuffing can be easily done by the *fill frame* specified under the error correction framing structure of H.261[4, Sec. 5.4.3]. Each error correction frame consists of 1 bit for framing and 511 bits of BCH(511,493) code. If the first bit of the 493 data bits is 0, the following 492 bits are filled with 1, otherwise they are real data. Please refer to [4, Sec. 5.4.3] for detailed description of the framing pattern. In H.261, MBA (MacroBlock Address) stuffing can also be used for bit stuffing. However, using MBA stuffing requires to detect macroblock boundary and is thus less desirable.

4 Theoretical Analysis

Currently, most video codecs use some kind of variable length codes to further compress video signals losslessly after some lossy coding schemes. In order to match a conventional constant bit-rate transmission channel, a buffer is needed at the encoders to smooth the variable rate output into a fixed rate output, and a similar buffer is needed at the decoders to reverse the process. It is observed that this bit-rate matching process in the buffers introduces a majority of delay, compared to processing (encoding/decoding) delay and transmission delay. If video bridging is done in the pel domain, the video bridge has to fully decode multiple images, merge them together, and then encode the combined image. Thus, overall delay for such video bridging is essentially twice the encoder-decoder-pair delay. On the other hand, if video bridging is done in the variable-length

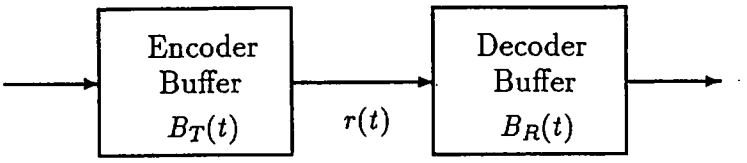


Figure 8: A Generic Buffer-Pair System

coded domain, although buffers are still needed in the video bridge for multiplexing, the overall delay can be only slightly larger than the codec-pair delay. In this section, we will give a theoretical analysis for the codec-pair case (without video bridge involved) which serves as a foundation for further understanding the case with video bridge. Unfortunately, the problem for the video bridge case is too complicated for a similar analysis. However, we still can derive a lower bound and an upper bound for the end-to-end delay as well as the required buffer size from some extreme cases. A heuristic method is also given to roughly estimate the end-to-end delay for typical cases. This estimate has been verified by the simulation results in Sec. 5 and Sec. 6.3.

4.1 The Codec-Pair Case

We will derive here the relationship between buffer size and the delay it introduces in the codec-pair case. The derivation and its results mostly follow [9], which should be referenced for details. Fig. 8 shows a generic encoder/decoder buffer pair system. The encoder and decoder buffer occupancies at time t are denoted by $B_T(t)$ and $B_R(t)$ respectively. Between these two buffers, there is a transmission channel. The transmission rate is represented by $r(t)$. The transmission delay is assumed to be zero since it is relatively small and roughly constant. We will focus here on the behavior of the two buffers. The buffers are subject to overflow or underflow if they are not controlled. The control of the encoder buffer usually involves some tradeoff between bit rate and picture quality. Namely, the encoder reduces its generated bit rate by using coarser quantizers, which implies lower quality, when the encoder buffer approaches overflow, and vice versa when approaching underflow. However, underflow of the encoder buffer generally is not considered to be a problem since it can be easily solved by sending stuffing bits. The details of encoder buffer control have been discussed by many papers[7, 8] and will not be covered here. Throughout this paper, we will only assume that the encoder buffer is under effective control and never overflows or underflows. We will derive the sufficient conditions for a decoder buffer without overflow or underflow.

For a real-time video codec, *the delay from the camera to a display is constant* if a smooth video display is produced without repeating or dropping frames. Since the processing delay in the rest of the system is usually constant, the delay from the time a particular data enters the encoder buffer to the time that data leaves the decoder buffer is also constant. We denote this delay by T. It follows from the constant delay property that the occupancy of the decoder buffer is strongly correlated to that of the encoder buffer. If we trace a particular data word d that enters the encoder buffer at $t = t'$, it will leave the decoder buffer at $t = t' + T$ according to the constant delay property. The data flowing through the transmission channel from t' to $t' + T$ will be those data in the encoder buffer when $t = t'$ and the data in the decoder buffer when $t = t' + T$. Thus, the relationship between the encoder buffer occupancy and the decoder buffer occupancy is as follows.

$$B_R(t' + T) = \int_{t'}^{t'+T} r(t)dt - B_T(t'), \qquad \forall\ t'. \tag{1}$$

In order to prevent decoder buffer overflow, its size, denoted by S_R, has to be sufficiently large so that it is always greater than or equal to the right hand side of (1). The most pessimistic case occurs when the encoder buffer is empty at $t = t'$ and r(t) is at its maximum rate, K, all the time from t' to $t' + T$. Thus, the sufficient condition for no decoder buffer overflow is

$$S_R \geq KT. \tag{2}$$

On the other hand, the decoder buffer will never underflow as long as all data reaches the decoder buffer within the time T after they entered the encoder buffer. The most pessimistic case occurs when the encoder buffer is nearly full and the data just entering the encoder buffer has to wait until all the data in front of it have been transmitted. The maximum waiting time is S_T/K, where S_T denotes the encoder buffer size. Consequently, in order to guarantee no underflow, the constant delay, T, has to be greater than or equal to this maximum waiting time.

$$T \geq S_T/K. \tag{3}$$

In summary, from (2) and (3), the sufficient conditions for a decoder buffer without overflow or under flow are as follows.

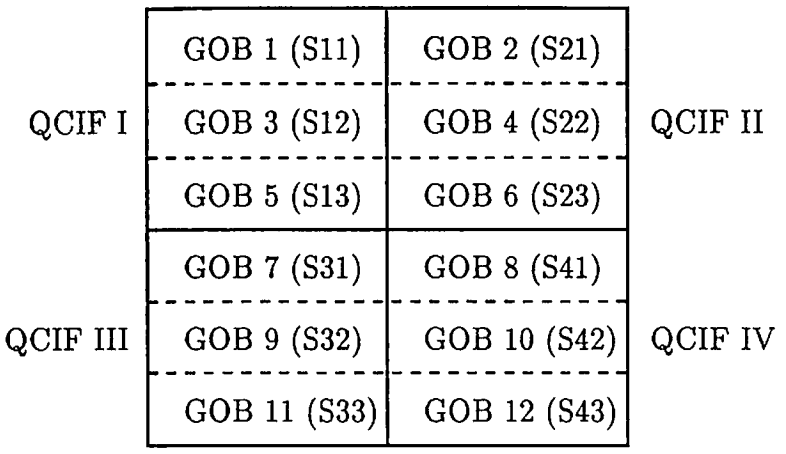
$$S_T/K \leq T \leq S_R/K. \tag{4}$$

Note that the delay is proportional to the buffer size (or more accurately the maximum buffer occupancy), and is inversely proportional to the transmission rate.

For an H.261 codec, the maximum buffer occupancy is usually incurred by an intra coded frame (I-frame) and is roughly about the size of the largest intra coded frame. H.261 limits the maximum number of bits that can be generated by a frame to 64 kbits for QCIF and 256 kbits for CIF. Such limits not only define the required buffer size but also bound the end-to-end delay. Assuming the transmission rate R is $p \times 64$ kbps, the maximum delay will be $1/p$ second for QCIF and $4/p$ seconds for CIF. In practice, the delay may vary over time because of repeat frame or skip frame operations. Usually, the delay is initially set by the first intra coded frame. This delay will not change until a repeat frame operation or a skip frame operation is performed. The repeat frame operation is needed to increase the delay when a new frame is larger than the frame which set the current delay and the current delay is not long enough to deliver the new frame (i.e., the receiver buffer underflow). Repeat frame can be easily done by the decoder without any action by the encoder. On the other hand, the skip frame operation, which reduces the delay, has to be done by the encoder to provide correct motion vectors. Otherwise the decoder is out of trace of the encoder, and can only recover until the next intra coded frame. If the encoder does not skip frames, the delay usually will be a non-decreasing function over time and is flat after displaying the largest frame. The final delay may not be equal to the bounds we just mentioned. The constant delay property holds between these step-wise changes or after a steady state is reached.

4.2 The Video Bridge Case

We define the total end-to-end delay as the time difference between the time that the first bit of the first QCIF frame enters the transmitter buffer and the time that the last bit of the combined CIF frame comes out of the receiver buffer. The videoconferencing terminal clocks can be locked to the network clock so that the data transmission is synchronous between the terminals and the network. These synchronous terminal clocks also ensure the frame rates of different input QCIFs are exactly the same but their frame phases may be different. However, we will assume they are all the same for the reason of simplicity. If the frame phases are not synchronous, the required buffer size for the bridge buffers may have to increase slightly to accommodate for the difference.



Sij: Number of coded bits for the corresponding GOB.

Figure 9: The Structure of Combining Four QCIFs into a CIF, where Sij

The exact amount of buffer increase is dependent on how the phase off-sync is handled but it is believed that the amount should be in the order of R/F. Throughout this paper, we also assume the I-frames of different input QCIFs are synchronous, unless mentioned otherwise.

4.2.1 Minimum Cases

The minimum delay in a codec-pair case occurs when the bit rate in each frame is constant over the frame time, such as in the fixed-length coder case. In this case, no buffer is required for transmission and receiving. Suppose F is the frame rate. Then, the transmission rate R of a QCIF encoder will be 3F GOB-units per second and the delay of such QCIF codec-pair is just 1/F (=3/3F) second, which is the frame period time and is 133.33, 66.67, and 33.33 ms for a frame rate of 7.5, 15, and 30 frames/sec respectively.

When a bridge is inserted in this minimum-delay case, the additional bridge delay can be derived using Fig. 9. Suppose pels are combined at GOB level in the bridge. At the end of 1/F second, all GOBs, from S11 to S43, are already loaded into the bridge input buffers, but only S11, S21, S12, S22, and S13 have been transmitted. The other 7 GOBs will be transmitted at rate 4R, which is equal to 12F GOB-units/sec. Thus, the total end-to-end delay is 1/F + 7/12F = 19/12F, which is equal to 52.78 ms for a frame rate of 30. The required buffer size for QCIF IV will be the largest among four since it is last transmitted. While it is waiting for its turn to transmit, the data in its buffer are accumulated. The maximum buffer occupancy is equal to the waiting time multiplied by the input rate (R). The waiting time is equal to the time required to transmit a frame of QCIF I, which is 1/F, added by the time required to transmit S23 and S31, which is 2/4R. Thus, the required buffer size for QCIF IV is 3.5 GOB-units. Similarly, the required buffer size for QCIF III is 3.25 GOB-units (= $[3/R + 1/4R]R$). The required buffer sizes for QCIF I and II are the same, 1.75 GOB-units (= $[7/4R]R$). All of the required buffer sizes are quite small. For example, if R is 64 kbps and frame rate is 30 frames/sec, the required buffer size for QCIF IV is about 2.5 kbits.

4.2.2 Maximum Cases

To find the maximum delay, we need to use a worst-case-dominant property of bridge buffers. Define the picture complexity of a QCIF as the number of bits required to code it using intra-frame mode. Assuming the four input QCIFs generate their intra-frame modes simultaneously, the worst-case-dominant property states that the end-to-end delay is determined by the most complex I-frame, regardless of when it happens in the bridging process. Using this property, we can thus assume that the maximum delay happens on the first frame, which simplifies the analysis.

The maximum delay happens when the 4 input QCIFs all have the highest possible picture complexity – in H.261, this is defined as 64 kbits for a QCIF picture. If the transmission rate is px64 kb/s, the maximum delay happens when QCIF II is distributed unevenly, i.e., S21 = S22 =0 and S23 = 64 kbits. In this case, at the end of 64k/(px64k) = 1/p second (the time finishing transmitting QCIF I), there are 3x64 kbits (= 192 kbits) left, which have to be transmitted at 4px64 kb/s. Thus, the end-to-end delay is 1/p + 3/4p = 7/4p seconds. The longest delay happens when p = 1, which is 1.75 sec.

The maximum occupancy of the QCIF IV buffer occurs when QCIF I has 64 kbits and S23 and S31 both also have 64 kbits. The 64 kbits of QCIF I have to be transmitted at only R (= $p \times 64K$) due to the bottle neck of input rate, but S23 and S31 can be transmitted at 4R. Thus, the required size of the QCIF IV buffer is

$$\left(\frac{64}{p \times 64} + \frac{2 \times 64}{4 \times p \times 64}\right) \times p \times 64 = 1.5 \times 64 = 96 \text{ kbits}.$$

Note that it is independent of p. The required size for all other QCIF buffers should not be larger than 96 kbits.

4.2.3 Typical Cases

It can be seen that the upper bound and lower bound on the end-to-end delay given above may be too loose and cannot provide a clear picture of the typical delay in practical cases. Here we will provide a simple heuristic method to estimate the typical delay in practical situations. Because of the worst-case-dominant property, we will use the maximum sizes of GOBs to estimate the delay. We denote $S1 = S11 + S12 + S13$ and similarly for S2, S3, and S4. The subscript max will be used to denote their maximum values in the actual video sequences. The end-to-end delay D can be estimated as follows.

$$D = \max(d_1, d_2, d_3, d_4), \tag{5}$$

$$\text{where } d_1 = \frac{S1_{max}}{R} + \frac{S23_{max} + S3_{max} + S4_{max}}{4R}, \tag{6}$$

$$d_2 = \frac{S2_{max}}{R} + \frac{S3_{max} + S4_{max}}{4R}, \tag{7}$$

$$d_3 = \frac{S3_{max}}{R} + \frac{S43_{max}}{4R}, \tag{8}$$

$$d_4 = \frac{S4_{max}}{R}. \tag{9}$$

As before, R is the transmission rate from each QCIF coder to the MCU and is also the target coded rate of each QCIF. Usually, d_i would be the maximum if Si_{max} is much larger than the other three such that the transmission of Si_{max} at rate R from a QCIF coder to the MCU is the bottle neck. If there is no dominant QCIF, d_1 is usually the largest one since it consists of more terms than

others. According to the simulation results presented in Sec. 5 and Sec. 6.3, this delay estimation is correct within 22 ms for most cases. However, it is not valid for cases where intra-mode coded frames of four input QCIFs are not in the same CIF frame.

5 Simulation Results

A simulation program was written to simulate the system shown in Fig. 5, which starts with four encoder buffers, goes through the QCIF combiner, and ends at the decoder buffer. This program reads four data sets, each for a QCIF video source. Each data set consists of a sequence of numbers and each number represents the number of coded bits for a GOB in a QCIF picture. The simulation uses a GOB (in a QCIF) period as a clock tick, which is 11 ms for a frame rate of 30 frames/sec. At every clock tick, the program reads four numbers (each of which represents the number of GOB coded bits for one of the four QCIF source pictures) and outputs the buffer occupancies of all nine buffers. Initially, the decoder buffer outputs nothing until all the coded bits in the first frame are in the decoder buffer. Once all the bits in the first frame have arrived at the decoder, these bits are removed from the decoder buffer in a clock tick and the first frame of the combined CIF picture is assumed to be displayed. The end-to-end delay is the time elapsed beginning from the time when the first bit of a frame entering the encoder buffer to the time when this frame is displayed. After the first frame is displayed, the decoder is assumed to display a frame of picture every frame period (which consists three clock ticks in our simulation). If the coded bits in the next frame have not all arrived at the decoder yet (which is a decoder buffer underflow situation), the decoder repeats the previous frame and the uncomplete frame is kept in the decoder buffer. Thus, each frame repeat will increase the end-to-end delay by a frame period. Eventually, the end-to-end delay will be long enough such that frame repeat will never happen again. The size of each buffer in the system is assumed to be large enough and never overflow. The phases and frequencies of four QCIF frames are assumed to be synchronous in our simulation.

In our simulation, we also assumed that the GOBs in a combined CIF frame are sent in the order of GOB number as shown in Fig. 9. Since the GOB number in H.261 standard is explicitly represented by 4 bits, it may be allowed to send GOBs in a frame in any arbitrary order. A QCIF combiner may be able to take advantage of this flexibility to reduce the delay by sending available GOBs in a frame first. However, some H.261 codec manufacturers may not interpret the standard to the extent of allowing random order of GOBs. Without changing H.261, we decided to follow the GOB number in sending them.

The source pictures used in our simulation consisted of four sequences: "Claire," "Miss America," "Salesman," and "Swing." Each of them was decimated to the size of a QCIF picture and then coded by an H.261 encoder very similar to the RM8[10], which has a simple buffer rate control algorithm. The frame rate is 30 frame/sec.[2] Two sets of data were generated for two different target rates, 64 kbps and 128 kbps respectively. An intra-mode coded frame (I-frame) is used in every 33 frames. Some interesting statistical data about the number of coded bits in a frame are shown in Table 3. The maximum number of coded bits in a frame occurred in one of the I-frames. These maximum numbers are all several times larger than the mean numbers. It is easy to understand these maximum number of coded bits in an I-frame is closely related to the maximum buffer occupancy and thus, as explained in the last section, is a major factor on determining the end-to-end delay. The maximum number of coded bits in a frame can be used as an index of picture

[2]The frame rate was chosen arbitrarily. The simulation results are equally valid for other frame rates by using equations (5) to (9) and the fact that the intraframe coded rate or complexity is independent of the frame rate using the RM8 rate control algorithm.

ID	Picture	64 kbps			128 kbps		
		max.	min.	mean	max.	min.	mean
M	Miss America	10043	795	2110	10043	1368	4129
C	Claire	13096	681	2106	13096	1344	4197
S	Salesman	14569	614	2071	14569	1523	4089
W	Swing	26074	334	2127	26074	1526	4225

Table 3: Statistics of the Number of Coded Bits in a QCIF Frame

ID	Picture	64 kbps			128 kbps		
		Delay	B^t_{max}	B^r_{max}	Delay	B^t_{max}	B^r_{max}
M	Miss America	16	8038	11248	9	6476	12729
C	Claire	18	11213	12798	9	9265	12798
S	Salesman	20	12436	14220	10	10356	14220
W	Swing	36	23941	25596	18	21808	25596

Delay: in clock ticks.
B^t_{max}: Maximum transmitter buffer occupancy (bits).
B^r_{max}: Maximum receiver buffer occupancy (bits).

Table 4: Simulation Results of a QCIF Codec

complexity. According to this index, the "Swing" has the highest complexity and "Miss America" has the lowest. Hereafter, we will refer to the pictures using the picture ID shown in Table 3. Note that the maximum numbers for the 64 kbps target rate and for the 128 kbps target rate are the same due to the simple rate control algorithm of RM8[10].

We also simulated the QCIF codec case (without QCIF combiner) for each of the source pictures. The simulation results on end-to-end delay and maximum buffer occupancies are shown in Table 4. These data will be used as reference points later. Note that, for each picture, the maximum receiver buffer occupancies for 64 kbps and 128 kbps target rate are almost all the same due to their identically coded I-frames. Also note that, for each picture except "Miss America," the delay for 128-kbps case is half of the delay for 64-kbps case due to twice of the transmission rate. The only exception is due to the quantum increase (3 clock ticks) of a repeat frame that occurred to "Miss America" and the discrepancy is within 3 clock ticks.

5.1 The End-to-end Delay

In this subsection, we focus on the end-to-end delay of the video combining system shown in Fig. 5. It is obvious that this delay depends on the bit-rate traffic generated by the four combined QCIF pictures. In order to obtain a maximum number of simulation cases with limited test source pictures, we tried four different source pictures in each of four different positions in CIF. Thus, 256 (4^4) cases were simulated. We will use a vector formed by the picture IDs to refer to each case. For example, the case (M,C,S,W) is the case with picture M, C, S, and W at the position I, II, III, and IV shown in Fig. 9 respectively. First, we will summarize the simulation results of four source pictures with target rate of 64 kbps. The delay distribution of 256 cases is shown in Fig. 10. The unit of delay time used in the figure is the clock tick of the simulation program, i.e., 1/90

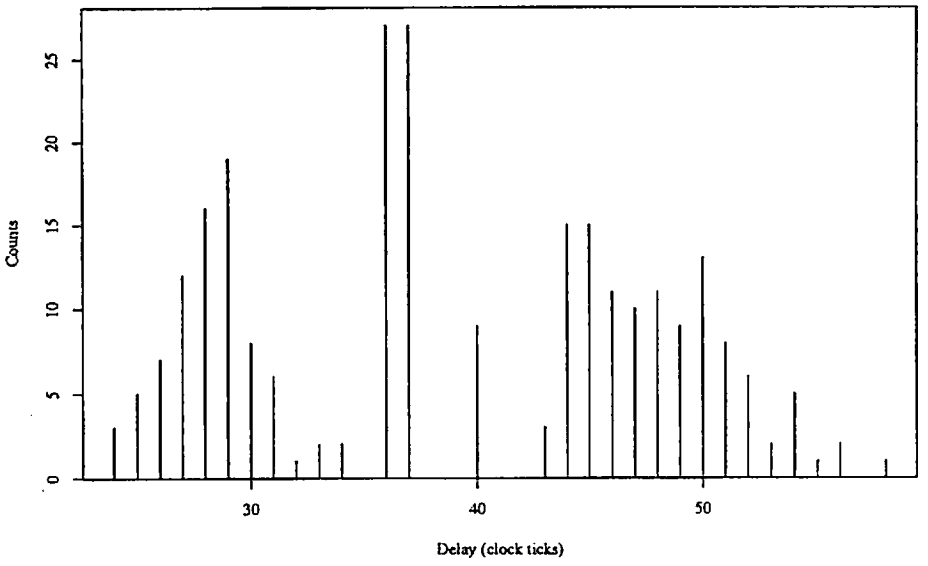


Figure 10: The Delay Distribution (QCIF Target Rate: 64 kbps)

sec. The shortest delay is 266.67 ms (24 clock ticks) and the longest delay is 644.44 ms (58 clock ticks). There are three cases with the shortest delay, (M,M,C,S), (M,M,M,M), and (M,M,S,C). Case (W,W,W,W) has the longest delay. Roughly, this is consistent with intuition that shorter delay results from less complex pictures. There are two peaks (at delay = 36 and 37) in Fig. 10. They are due to the dominant complexity of the "Swing" (W). The cases with W at IV and other pictures at other positions all have the delay of 36 clock ticks. The cases with W at III and other pictures at other positions all have the delay of 37 clock ticks. Thus, each has 27 (3^3) cases with the same delay. The cases with the same four pictures permuted at different positions may result in quite different length of delay. For example, with M, C, S, and W permuted, the delay can vary from 36 to 47 clock ticks. After reviewing all cases carefully, we concluded that the shortest delay among different position permutations usually results from placing QCIF pictures in the order from the least complex to the most complex. However, the longest delay may not result from a reverse order of the above. It usually results from placing the most complex picture at position I and the least complex one at position II. Note that these are only two rough rules of thumb. The actual cases may differ from these by 3 clock ticks or less caused by quantum effect of repeats. The most important observation is that the overall delays from all our experiments are less than two times the maximal codec delay (without the combiner) among four QCIF pictures, which is better than the transcoding approach.

We repeated the same simulations for the source pictures coded at the target bit rate of 128 kbps. The delay distribution of 256 cases with different pictures at different positions is shown in Fig. 11. Compared with the cases with the target rate of 64 kbps, the end-to-end delays of 128 kbps cases are roughly all reduced by half. This is consistent with the equations (5)–(9): the delay is inversely proportional to the transmission rate. Note that the number of coded bits in an I-frame with target rate of 128 kbps is the same as that of the same I-frame with target rate of 64 kbps due to the simple rate control algorithm of RM8. Thus, when transmission rate increases two times, we can expect that the delay will be reduced by half.

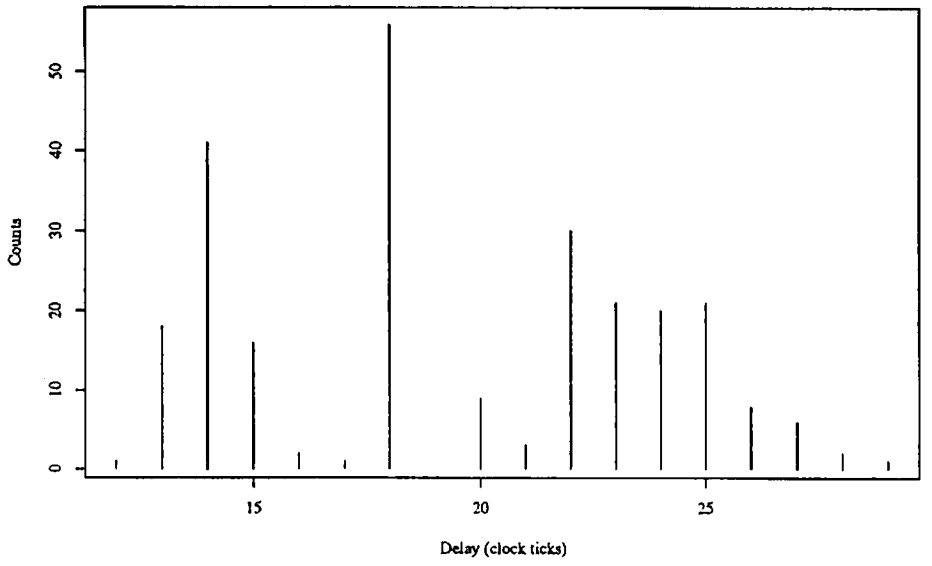


Figure 11: The Delay Distribution (QCIF Target Rate: 128 kbps)

5.2 Buffer Sizes

We need to know two things about the buffer sizes: 1) the required buffer size in the QCIF combiner, and 2) whether the required buffer size for the CIF receiver is affected. The QCIF combiner under consideration is more or less passive to the QCIF transmitter. Thus, the required buffer size for the QCIF transmitter is not affected. The required transmitter buffer size to accommodate all four test pictures of 64-kbps target rate is 23941 bits and that of 128-kbps is 21808 bits according to our simulations (both are set by "Swing").

The maximum buffer occupancies of the QCIF combiner appeared in our simulations for different target rates and different QCIF positions are shown in Table 5. The simulations consisted of 256 cases for each target rate with different pictures at different positions. Comparing these maximum buffer occupancies for different QCIF positions, we can observe that QCIF IV's is the largest, followed by QCIF II's, QCIF III's, and QCIF I's is the smallest. However, they are not substantially different. The maximum buffer occupancies for the target rate of 128 kbps are slightly smaller than those for 64 kbps. It is also interesting to note that most of the maxima for a bridge buffer occurred with the "simplest" picture at the position corresponding to this buffer and with the most "complex" picture at the other three positions. A last but important observation is that the required size of each bridge buffer is slightly larger than a QCIF receiver buffer shown in Table 4. The largest shown in Table 5, 29927, is about 17% larger than the largest shown in Table 4, 25596.

The maximum occupancy of the receiver buffer appearing in our simulations is 105892 bits for the target rate of 64 kbps, and 108051 bits for that of 128 kbps. Both of them are set by the case with "Swing," the most complex picture, at all four QCIF positions. They are only slightly greater than four times (actually, 4.14 and 4.22 times respectively) the maximum buffer occupancy of a QCIF receiver (25596 bits). Thus, the required buffer size of the CIF receiver is roughly unchanged, compared to the case with the same CIF being transmitted without the QCIF combiner.

Target Rate	QCIF I	QCIF II	QCIF III	QCIF IV
64 kbps	28536 (MWWW)	29607 (WMWW)	28440 (WWXX)	29927 (WWWM)
128 kbps	27496 (MWWW)	28646 (WMWW)	28440 (WWYX)	29862 (WWWY)

What shown in the parentheses is the combination of the pictures which set the maximum.
The picture IDs are defined as before.
X represents M, C, S, or W.
Y represents C, S, or W.

Table 5: The Maximum Buffer Occupancies (in bits) of the Four Bridge Buffers in Simulations

6 Possible Improvements

It is very important to keep the end-to-end delay low for interactive two-way (and multi-way) communication applications. The typical end-to-end delay for an H.261 codec (without video bridges) is already a quarter second long. The cases with video bridges would have even longer delays. It is thus desired to reduce the codec delay such that the cases with video bridges will also be improved. In this section, we will first discuss two possible simple methods to reduce the video bridge delay. However, these two methods are not applicable to reducing the codec-pair delay. Three more advanced schemes will then be discussed, which reduce delay in more fundamental ways – they reduce the codec-pair delay as well as the video bridge delay. Note that these three schemes have been considered in the low delay coding of MPEG2[11, Appendix H]. We then focus on the *intra slice* technique. Simulation results are presented to verify the delay reduction.

6.1 Two Simple Methods

In the above simulations, the I-frames of four QCIFs are assumed to be synchronous with one another. Thus, a CIF I-frame is formed by the QCIF combiner every 33 frames in our experiments. However, H.261 allows the coded mode to be specified on the macro-block level. Therefore, it is possible to let the I-frames of four QCIFs to occur in different CIF frames so that only one QCIF in a combined CIF frame is intra-mode coded. Such off-sync I-frames can be arranged by the MCU through the fast update signal in H.221[6]. In order to investigate the effect of this off-sync I-frame arrangement, we used the same source picture for each of the four combining QCIFs. For an off-sync distance of n frames, n, $2n$, and $3n$ "blank" frames were added in front of the original QCIF II, III, and IV sequence, respectively. A "blank" frame can be viewed as a frame of picture with all of its coefficients equal to zero, and its coded bit stream will consist of only the necessary headers without real bodies. The simulation results for four source pictures with target coded rate of 64 kbps are shown in Fig. 12. Roughly speaking, the delay decreases as the distance between I-frames increases. Note that, as the off-sync distance increases to more than 8 frames, the distance between the I-frames of QCIF IV and those of QCIF I is shorter than others and is decreasing. Thus, simulations for an off-sync of more than 10 frames are not necessary. The delays for pictures M, C, and S all reached their individual minimum (16, 21, and 23 respectively) when the off-sync distance (n) was increased to 6. The delay for "Swing" reached its own minimum (36) later when $n = 8$. Compared to the single QCIF codec delays shown in Table 4, these minima are the same as or within 3 clock ticks of those. We can conclude that such simple scheme can effectively reduce

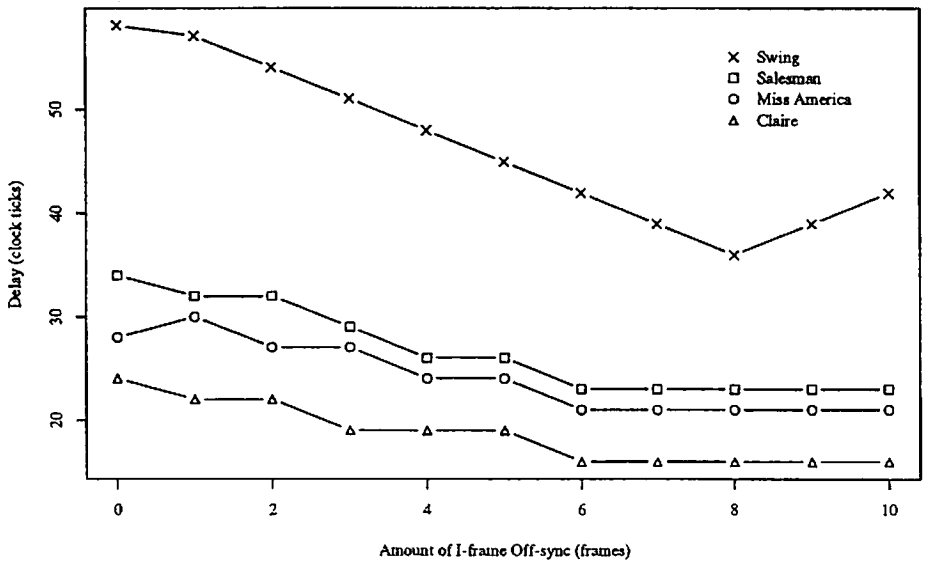


Figure 12: Delay versus Off-Sync Distance

the end-to-end delay. The only penalty is that we may have to wait longer to see the first good frame of QCIF IV in the combined CIF picture.

Another possible scheme to reduce delay is to use a symmetric duplex channel between users and MCU. Thus, the transmission rate from the transmitter to the MCU is the same as the user receiving rate, $4R$, where R is the target coded rate for a QCIF. If we utilize this full capacity to transmit data to the MCU, the end-to-end delay can be reduced. Certainly, bit stuffing will be needed when there is no data in the buffers of QCIF encoders. The delay distribution of 256 cases is shown in Fig. 13. Compared to the case with transmission rate at R (Fig. 10), these delays are reduced by 5 to 24 clock ticks, or by 20% to 50%.

6.2 Three Advanced Schemes

From the above analysis and simulations we see that variation in the coded data stream is the major source of delay. The MCU is basically a passive multiplexer which has very little control of the overall delay. If all coders generate bursty data that have very high rate peaks, the buffers have to accommodate the worst case and a long delay is inevitable. Such high peaks usually results from complete intra-mode frame or field coding. Low delay schemes have to either deal with these intra-mode coded frames or avoid using them.

The first scheme is called the *skip frame* method, which reduces the delay by skipping a few frames after a "large" frame, usually an intra-mode coded frame. The frame following the skipped frames uses the "large" frame as the reference for its motion compensation. To fill the display pipe, the frame prior to the "large" frame is repeated before the "large" frame is fully decoded and available. This method may result in quality degradation in the few frames immediately after the skipped frames since the reference frame for motion compensation is a few frames away. It also may result in jerky motion artifacts due to the skipped frames.

The other two techniques both try to distribute intra coded data across every frame and thus smooth out the bit-rate variation. In the *leaky prediction* scheme, each frame (except perhaps the

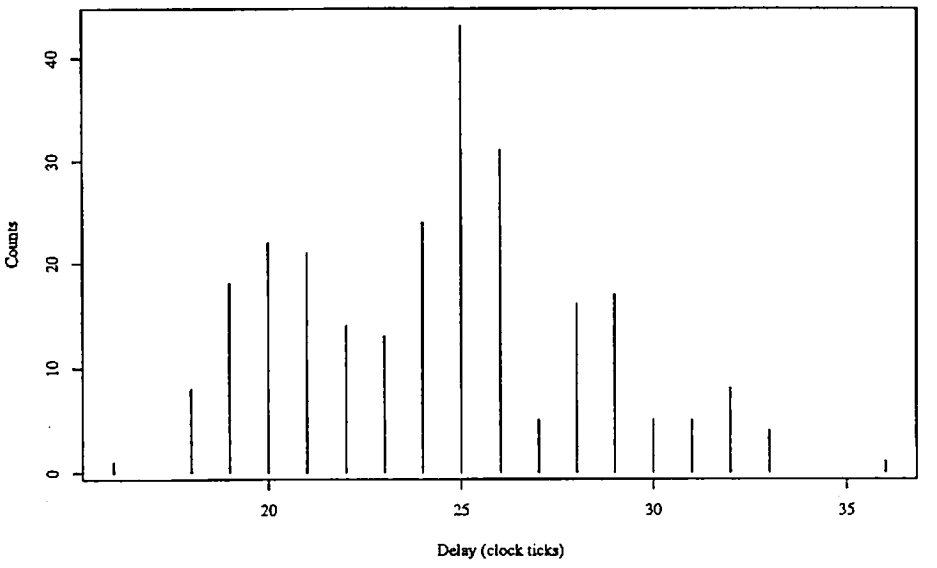


Figure 13: The Delay Distribution with $4R$ Transmission Rate from Transmitter to MCU ($R = 64$ kbps)

first frame) is leaky predictive coded using motion compensation. The leaky prediction distributes a tiny portion of the intra-frame information to each predictive frame so no complete intra-mode coded frame is generated. This results in smooth bit-rate variation and thus a lower end-to-end delay. The first frame can be intra coded but then skip frame has to be used to cut back the delay before leaky prediction can be applied. Alternatively, the first frame can be coded directly using leaky prediction with reference to a blank picture. In this way, the reconstructed images will be very rough for the first few frames and gradually build up to the normal images. The same thing happens when there is a scene cut.

The third technique is the *intra slice* scheme. The idea is to update (i.e., intra encode) a small segment in each frame. Thus, no "large" frame will be generated. Each frame consists of a segment that is intra coded and the rest of the frame is predictive coded by motion compensation. In this way, the number of bits generated from each frame varies smoothly frame by frame. Thus, it results in a lower delay. Similar to the leaky prediction scheme, the picture can initially be intra coded and then skip a few frames, or can be built up gradually. In the latter approach, segments of the picture remain blank initially until they are intra coded. As those segments are intra coded and appear one by one, a sweep sense of updating is created.

Note that the skip frame scheme and the intra slice scheme can be used without changing the H.261 syntax, but the leaky prediction scheme cannot. We will focus on the *intra slice* scheme for the rest of the section because it is simple, effective, and can be accommodated by the current H.261 syntax. In the H.261 syntax, the coding mode can be specified macroblock by macroblock. Each macroblock can be intra coded individually without changing the current H.261 syntax. Each updating segment can certainly contain more than one macroblock. Usually, it is convenient to have the number of macroblocks in the segment be a divisor of the total number of macroblocks in a frame. In our experiment, we updated 3 macroblocks every QCIF frame or 12 macroblocks every CIF frame. There are 99 macroblocks in a QCIF frame or 396 macroblocks in a CIF frame. It takes

ID	Picture	64 kbps			128 kbps		
		max.	min.	mean	max.	min.	mean
M	Miss America	2810	784	1900	7205	784	4175
C	Claire	2960	1335	2098	5485	1846	4162
S	Salesman	2665	1471	2102	6649	2715	4188
W	Swing	4541	1227	2100	8731	2646	4179

Table 6: Statistics of the Number of Coded Bits in a QCIF Frame Using Intra Slice Scheme

33 frames to build up or refresh the whole picture. If the frame rate is 30 frames/sec, the refresh cycle will be 1.1 seconds. It can be shown by the simulation results in the following subsection that the *intra slice* scheme can drastically reduce the end-to-end delay for both the codec-pair case and the video bridge case. The required buffer sizes are also drastically reduced. Using a QCIF transmission rate of 64 kb/s and 30 frames per second, the overall delays are reduced by 67% to 86% for different combinations of QCIF pictures. Before this modification, the overall delays ranged from 266.7 ms to 644.4 ms. After the modification, the overall delays became 66.7 ms to 111.1 ms. The required sizes of the transmitter buffers, the combiner input buffers, and the receiver buffer are reduced by 90%, 84%, and 78% respectively.

6.3 Simulation Results of the Intra Slice Scheme

The source pictures used in the *intra slice* simulations are the same four pictures as before. Some important statistical data of these pictures coded by the intra slice scheme are shown in Table 6. Compared to Table 3, their dynamic ranges have been reduced drastically, 4.56 times to 11.69 times for the target rate of 64 kbps and 1.35 times to 4.03 times for the target rate of 128 kbps. This dynamic range reduction effectively reduces the delay of the video codec. Table 7 shows the simulation results of the QCIF codec case (without QCIF combiner). All the delays can be reduced to only 6 clock ticks (= 2 frames) except for the 128-kbps "Miss America", whose delay is even better, only 3 clock ticks (1 frame). Note that the maximum buffer occupancies are also reduced drastically. For the most complex picture, "Swing," the maximum transmitter buffer occupancy is reduced by 9.94 times for 64-kbps target rate and by 4.88 times for 128-kbps. For "Swing," the maximum receiver buffer occupancy is reduced by 6 times for 64-kbps target rate and by 3 times for 128-kbps.

Similar to the simulations described in Sec. 5, the QCIF combiner simulations were done for 256 cases with four different pictures at each of the four QCIF positions. The simulation were done separately for the target rate of 64 kbps and 128 kbps. The end-to-end delay distribution of these 256 cases is shown in Fig. 14. The delays for 64-kbps target rate are shortened to only 6 to 10 clock ticks, and those for 128-kbps target rate are similar, from 6 to 9 clock ticks. These are very close to the delay (6 clock ticks) of the QCIF codec case (without QCIF combiner), which is the best we can get. Unlike the results shown in Sec. 5, the delay for a picture combination with 64-kbps target rate is similar to the delay for the same picture combination with 128-kbps target rate, not twice as much. However, more than half of the combinations with 128-kbps target rate have delay of 6 clock ticks while only more than one third of the combinations with 64-kbps target rate have the minimum delay. From these results, one can conclude that the intra slice scheme not only reduces the delay effectively, but also makes the delay more consistent, i.e. reduces its variance. Using intra slice scheme, the extra delay added by the QCIF combiner is very minor for all the cases.

ID	Picture	64 kbps			128 kbps		
		Delay	B^t_{max}	B^r_{max}	Delay	B^t_{max}	B^r_{max}
M	Miss America	6	965	4171	3	3315	4266
C	Claire	6	1933	4266	6	2716	8532
S	Salesman	6	1939	4266	6	3323	8532
W	Swing	6	2408	4266	6	4465	8532

Delay: in clock ticks.
B^t_{max}: Maximum transmitter buffer occupancy (bits).
B^r_{max}: Maximum receiver buffer occupancy (bits).

Table 7: Simulation Results of a QCIF Codec Using Intra Slice Scheme

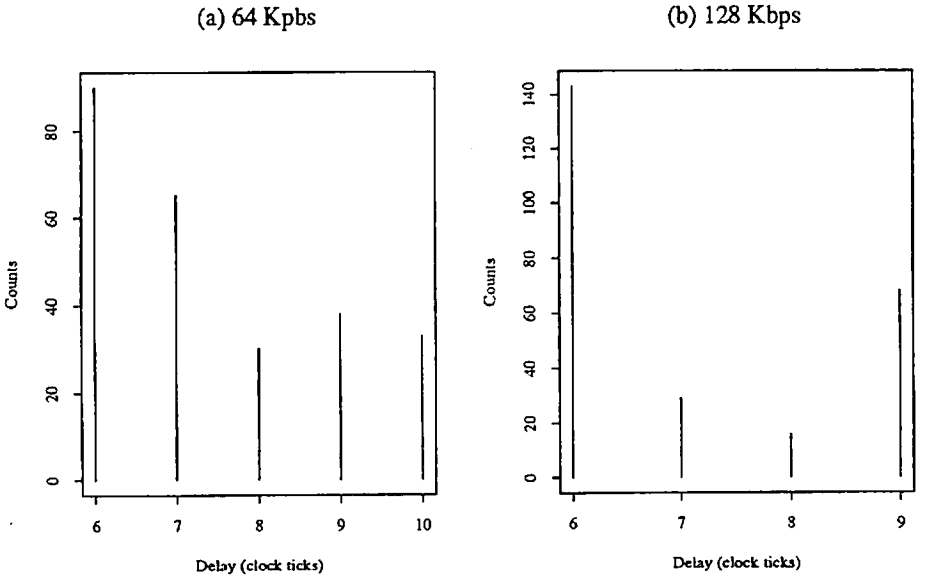


Figure 14: The Delay Distribution for the Intra Slice Scheme

Target Rate	QCIF I	QCIF II	QCIF III	QCIF IV
64 kbps	4266 (ZWWW)	4266 (WZWW)	4691 (WCSW)	4977 –
128 kbps	8532 (ZWWW)	8532 (WZWW)	8532 –	9954 (WWWW)

What shown in the parentheses is the combination of the pictures which set the maximum. The picture IDs are defined as before. Z represents M, C, or S.

Table 8: The Maximum Buffer Occupancies (in bits) of the Four Bridge Buffers in Intra Slice Simulations

The maximum occupancies of the four QCIF combiner buffers for different target rate are shown in Table 8. Compared to Table 5, the required buffer sizes for 64-kbps target rate are at least 6 times smaller, and those for 128-kbps target rate are at least 3 times smaller. Compared to Table 7, many of them are the same as the maximum receiver buffer occupancies in Table 7 (4266 bits for the 64-kbps case and 8532 bits for the 128-kbps case). Note that the largest in Table 8, 9954, is also 17% larger than the largest shown in Table 5, 8532.

The maximum occupancy of the receiver buffer after the QCIF combiner is 23728 bits for 64-kbps target rate and is 42146 bits for 128-kbps target rate. Both are much smaller than those (105892 and 108051 bits) without using intra slice. However, 23728 is 5.56 times of 4266 and 42146 is 4.94 times of 8532. Both are significantly greater than 4 times of a QCIF receiver buffer, which is expected for a CIF receiver buffer.

One major concern about the intra slice scheme is its error propagation property. A single error in transmission may propagate and grow from frame to frame due to motion compensation. In the worst case, such temporal and spatial error propagation can theoretically never be fully cleaned up by the intra slices and will propagate forever. However, according to our preliminary experiments, the possibility for such pessimistic scenarios to occur is very small in practice. One solution is to limit the motion vectors such that the reference blocks are all within the same updating segment. In this way, the errors are confined in the same segment and will be cleaned up by the forthcoming intra slice. However, this approach can either degrade the picture quality or reduce the coding efficiency due to the confinement of the motion vectors.

7 Conclusions

Video bridging is needed in providing multi-point video conference services through public switched networks. End-to-end delay, picture quality, and system complexity are the major issues in the design of the video bridge. In this paper, we discuss the design of a coded domain QCIF combiner and show that it can provide a better performance than a pel-domain combiner in terms of delay, picture quality, and complexity. We analyze the delay introduced by the combiner and decide the required size of the internal buffer for the QCIF combiner. We perform simulations to show the typical performance of the QCIF combiner. We also investigate several techniques to reduce the end-to-end delay. From these results, we show that coded domain QCIF combining can be reasonably achieved for multi-point networked videoconferencing applications.

References

[1] S. Oka and Y. Misawa, "Multipoint Teleconference Architecture for CCITT Standard Videoconference Terminals," *Visual Communications and Image Processing '92,* P. Maragos, Ed., Proc. SPIE 1818, pp 1502-1511, Nov. 1992.

[2] CCITT Study Group XV - Report R 93, Draft Recommendation H.231, *Multipoint control units for audiovisual systems using digital channels up to 2 Mbit/s,* May 1992.

[3] CCITT Study Group XV - Report R 94, Draft Recommendation H.243, *Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 2 Mbit/s,* May 1992.

[4] CCITT Study Group XV - Report R 95, Recommendation H.261, *Video Codec for Audiovisual Services at p x 64 kbit/s,* May 1992.

[5] S. F. Chang, W. L. Chen, and D. G. Messerschmitt, "Video Compositing in the DCT Domain," *IEEE Workshop on Visual Signal Processing and Communications,* Raleigh, NC, pp. 138-143, Sept. 1992.

[6] CCITT Study Group XV - Report R 95, Recommendation H.221, *Frame Structure for a 64 to 1920 kbit/s Channel in Audiovisual Teleservices,* May 1992.

[7] K.-H. Tzou, T.-C. Chen, and F. E. Fleischer, "An Intelligent Rate Control for an Intrafield Subband HDTV Coder," *Proc. Int'l Symp. of Circ. and Sys.,* Singapore, June 1991.

[8] C.-T. Chen and Andria Wong, "A Pseudo Fixed-Bitrate Video Coding Algorithm Using Self-Governing Rate Buffer Control Strategy," *IEEE Trans. Image Processing,* Jan. 1993.

[9] R. C. Lau, P. E. Fleischer, and S.-M. Lei, "Receiver Buffer Control for Variable Bit-rate Real-time Video," *Proc. of Int'l Conf. on Communications'92,* Vol. 1, pp 544-550, Chicago, Illinois, June 1992.

[10] CCITT Study Group XV - Document 525, *Description of Reference Model 8 (RM8),* June 9, 1989.

[11] ISO-IEC/JTC1/SC29/WG11, *MPEG II, Test Model 4,* Feb. 1993.