Exploring Adaptability for Video Conferencing*

Bharat Bhargava, Shunge Li
Purdue University
Department of Computer Sciences
West Lafayette, IN 47907
{bb,lis}@cs.purdue.edu

Abstract

Video conferencing application demands a lot of system and network resources. However, the amount of resources that can be used by the application always changes dynamically. Moreover, various anamolies further prevent it from running in a satisfactory state. This paper shows that adaptability is an approach to dealing with run-time resource constraints and various anamolies by showing how a video conferencing system maintains its quality services in a user-satisfactory manner when the network bandwidth decreases or when other anamolies occur. Several techniques for achieving adaptability of video conferencing tools are explored. The performance of an integrated adaptable video-conferencing system is studied in detail. This paper reveals that adaptability is a characteristics of video conferencing systems and can be exploited in environments where resources are limited and shared among multiple applications, and where the anamolies occur frequently and unpredictably.

1 Introduction

Adaptability is one of the features that a system can employ [4]. In the recent past, with support for multimedia becoming a reality, video conferencing systems have received a lot of attentions in commercial and research institutions [13]. Video conferencing involves multiple parties, possibly geographically interspersed, and exchanging real-time video data. The need for adaptability in video conferencing applications on the wide area network can be attributed to the static as well as dynamic nature of the communication environments. The static aspects of the environments include the link capacities of underlying networks, distance between participating sites, etc. The dynamic aspects of the environments include wide variance in packet loss rates and communication delays, available network bandwidth, message traffic, machine

*This research is supported by NSF under grant number NCR-9405931

loads, etc. Anamolies such as site/link failures and network partitioning also affect the effectiveness and utilization of the communication environments. Incorporating adaptability into a video conferencing system can minimize the effects of the variations in system environments on the quality of video conference sessions. Support for session recording feature provides the capability of saving video sessions when the quality of service becomes unacceptable or when some of the recipients become unreachable due to network partitioning or site/link failures. Furthermore, recording a session is also useful when the recorded video session is expected to be replayed and retrieved by future users interested in it.

For a dial-up user or an internet user to participate in a video conference, the quality of video is limited by the reliability and the bandwidths of the underlying network and the serial lines (phone lines). Current video conferencing systems [13] lack the ability of dynamically adapting themselves to the variations in the network bandwidths and anamolies. In these tools, changes in frame sizes, codec schemes, color depths, and frame resolutions can be made by users *interactively*. However, they cannot be made *automatically* based on the system measurements of current network conditions. This increases the users' burden in keeping the system most suitable to current environment, which is cumbersome and ineffective.

Current video conferencing systems lack support for recording in real-time conference sessions into databases for efficient retrievals. Some tools, such as NV [6], allow capturing of a single frame by users; others, such as the media-on-demand server [9], allow recording of whole conference sessions as archives. Obviously the usage of such archives is limited because from the user point of view, the recordings are collections of files, not tuples in databases. They can only be retrieved or replayed sequentially. Users cannot randomly select any part of a session via database queries, not to mention content-based retrievals by which users may retrieve any interesting portion of a session. It is also not easy to edit (add, delete, update) the video recordings without affecting other parts of the recordings. Even though some of the tools provide session recording feature, the enabling of this feature is not triggered automatically by the environmental changes, but handled manually by the users.

This paper is organized as follows. Section 2 addresses the relationship between adaptability and QOS control and presents the criteria for measuring the capacity of an adaptability scheme. In section 3, we briefly review techniques that make the video-conferencing system adaptable to available network bandwidth at run-time. The integrated adaptable video conferencing system that incorporates the adaptability features, session recording features, and a video-on-demand system is also described in this section. To study the performance of the adaptable video conferencing system, we conducted a lot of experiments. Section 4 gives detailed experimental data analyses, observations, and discussions. Section 5 presents related work in this area. Section 6 concludes the paper with the directions of future work.

2 Adaptability and QOS Control

Quality-of-Service (QOS) measurement and control have been a very interesting research area in real-time distributed multimedia system [14]. QOS control consists of two major services: negotiation and translation of QOS parameters. [15] classified QOS parameters into three layers: user QOS parameters, application QOS parameters, and system QOS parameters. Our research in adaptability is closely related to this area. For example, certain application layer QOS parameters, such as frame color or resolution, are traded dynamically for others, such as frame rate, based on the constraint of some system layer QOS parameters, such as network bandwidth. Therefore, the performance study of an adaptable video-conferencing system will provide guidelines for QOS negotiation and translation at both system layer as well as application layer.

In some sense, adaptability features are application layer QOS parameters. Adaptation is a process of dynamic QOS negotiation, based on the system and network conditions, such as resource availability. Adaptability features allow a system to provide the highest possible quality of service (QOS) in terms of user needs when the system faces anamolies, resource constraints, heterogeneity in network, and varying number of users. In order to perform QOS control, we have to quantify the QOS parameters at all three layers for a specific application. The benefit/loss function can thus be established for each QOS control action.

The criteria we use to measure the capability of an adaptation scheme are called Timeliness-Accuracy-Precision (TAP) property. Timeliness is defined as "when an event is to occur". Maintaining it means that the system meets a deadline. Accuracy is defined as "the degree to which the output conforms to the semantics and contexts of the applications". Maintaining it means guaranteeing the correctness of the data. For example, lossy compression algorithms cause the loss of accuracy. Precision is defined as "the quantity of information provided or processed". Maintaining it implies adjusting the amount of data being processed or transmitted over the network. For example, the number of frames per session, the number of pixels per frame, and the number of bits per pixel are parameters used to describe the precision of a video conferencing session.

TAP cannot be maintained at the highest level simultaneously during anomalies. We must trade among these attribute values. For example, the policy to maintain timeliness as the bandwidth decreases requires that the frame size or frame resolution be reduced, or the color depth be compressed. Since each choice can lead to different quality of services, the decision can be made by the user, or by the system automatically based on additional requirements provided by the users. Therefore, if the accuracy is to be maintained, the decision will be to reduce frame size, not to reduce frame resolution because accuracy is maintained when frame size is reduced unless the frame size goes below a certain value. On the other hand, both accuracy and precision will be sacrificed when the frame resolution is reduced.

3 Adaptability for Video Conferencing

3.1 Techniques for Achieving VC Adaptability

The basic idea for achieving adaptability for a video conferencing system is to trade some aspects of video quality for others. For example, the frame rate decreases as the available network bandwidth drops. Since the smoothness of a video session is sometimes more important than any other aspects of video quality, we may have to maintain a reasonable frame rate during a video conference session even though the network performance becomes degraded. To achieve this, we have to sacrifice some aspects of video quality, such as color or resolution of video frames, to reduce the amount of data sent over the network. However, any trading of this kind should be done in a user-transparent way, based on some user-specified criteria that must be satisfied.

We have developed several techniques, namely, color depth compression, resolution reduction, frame resizing, and codec scheme adaptation, that make the video conferencing tools adaptable to network conditions, namely, the available network bandwidths at run-time. We have proposed a new mechanism called VC Collaborator (VCC) to incorporate the adaptability features in a video conferencing application [11]. VCC dynamically monitors the system characteristics such as available network bandwidths, uses the information collected to determine the deliverable quality of video to the users, and reconfigures the system with the adaptability features. In the subsequent subsections, we will present these techniques in more details.

3.1.1 Color Depth Compression

More bits are needed to encode a color pixel than to encode a gray-scale pixel. In case of lack of the bandwidth, the system converts the original color video to gray-scale video, thereby reducing the network bandwidth requirement for transmitting a frame. A network probing module can monitor the network traffic and compute the currently available bandwidth. If currently available bandwidth is less than some threshold value of available bandwidth, a color frame is converted to a gray-scale frame (Figure 1 (a)). While this scheme is simple, it is hard to determine the threshold d. Even if the threshold d can be determined by experimentations, the color of the video tends to fluctuate when the bandwidth changes often across d. To overcome this problem, we have implemented a *dual-threshold* scheme (Figure 1 (b)). The color video is not converted to gray-scale unless the bandwidth is less than d_1. Similarly, the gray-scale video is not resumed to color video unless the bandwidth exceeds d_2. As long as $d_2 - d_1$ is large enough, color fluctuation is avoided. The choice of values for the two thresholds is determined empirically, so as to make the video insensitive to small bandwidth changes.

In scheme 1(a), two color levels, full color and gray-scale, are used to indicate the two extreme cases of large and small bandwidths. This classification has been refined to introduce multiple levels of colors. The intermediate levels of colors correspond to moderate network bandwidths (Figure 1 (c)). When combining this idea with the dual-threshold scheme, we get

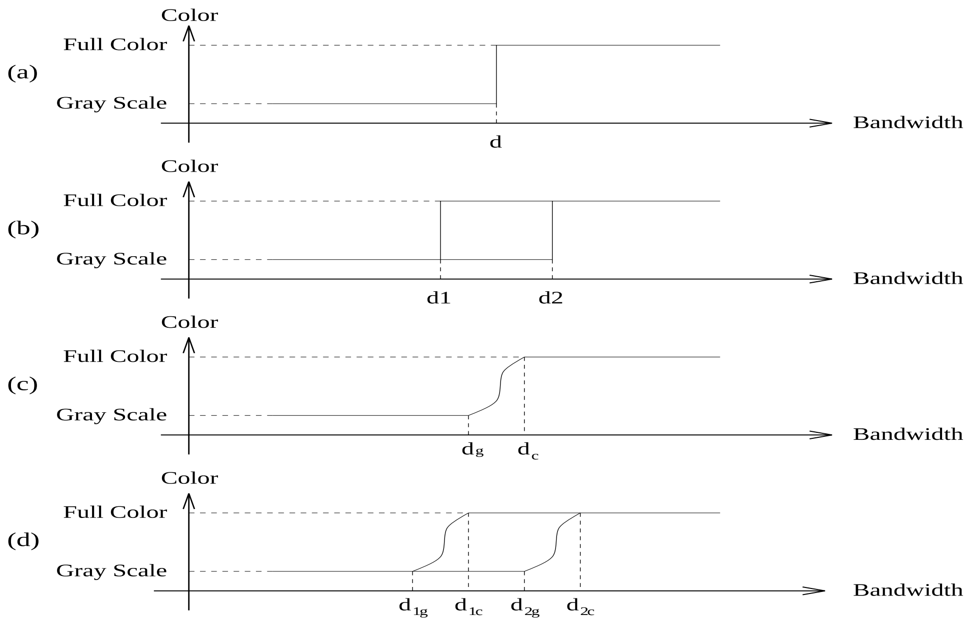

Figure 1: Thresholding for Color/Gray-Scale Conversion

scheme (d) (Figure 1 (d)).

Two color models are often used in practice: RGB and YUV [7]. Y stands for the luminance component of a pixel; U and V represents the chrominance components of a pixel. Many systems use YUV model because of its ease of adjustment for colors. For example, in YUV model, UV components can be changed easily and separately without disturbing Y components, and vice versa. In a typical video conferencing system, frames represented in RGB space are first captured and then converted into those in YUV space. Since human eyes are less sensitive to chrominance than to luminance, UV is further *subsampled* with an adjacent pixel pair sharing a common UV value while Y component is retained for each pixel. After a 4:2:2 subsampling, each pixel has 8 bits Y component and each adjacent pixel pair has 16 bits UV component. To further take advantage of human perception system, we can reduce the number of bits in encoding U and V components while retaining the number of bits in encoding Y components, and still guarantee relatively good quality in decoded color frames. Nevertheless, subsampling is lossy.

After subsampling, Y and UV components of a pixel are each 8 bits long. Color depth compression technique takes advantage of the redundancy of Y and UV components of pixels and uses fewer bits to represent Y and/or UV components, respectively. Apparently, to maintain the frame quality as good as possible, only the less significant bits of Y and/or UV components are discarded; the more significant bits of Y and UV components are retained. The experimental results will be shown in section 4.

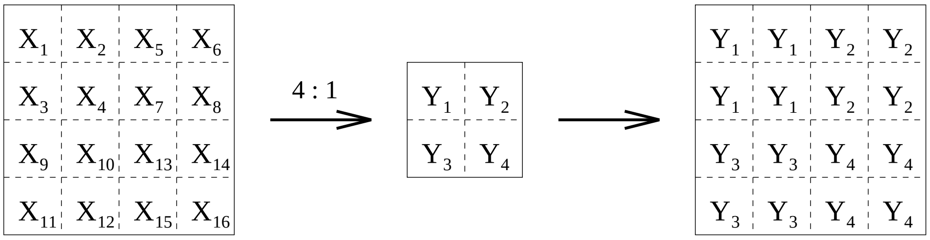

Figure 2: Resolution Reduction for Y Components

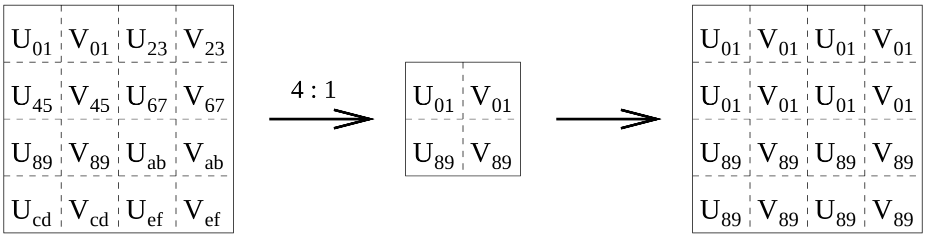

Figure 3: Resolution Reduction for UV Components

3.1.2 Frame Resolution Reduction

When the network bandwidth decreases, in order to maintain a reasonable frame rate, the amount of data in a frame must be reduced to have a frame fit into the smaller bandwidth. Resolution reduction is another way to achieve this. Figure 2 shows a mechanism to reduce the resolution of a frame by a factor of 2, reducing the size of a frame by a factor of 4. Unlike color depth compression, which is performed after a frame is encoded at the sender side and before it is decoded at the receiver side, frame resolution reduction is performed before a frame is encoded at the sender side and after it is decoded at the receiver side.

A frame can be divided into a matrix of blocks of pixels. Each block of pixels is then compressed into one pixel, based on some predefined function. Called *compression factor*, a global variable is defined to indicate the size of the block being compressed into one pixel. For example, a compression factor of 2 means a 2 by 2 block in an original frame is replaced with one pixel, resulting in a compression ratio of 4. The value of the new pixel is a function of the values of all the pixels in the corresponding 2 by 2 matrix. The function can be a projection function or a (weighted) averaging function. But the time taken in computing the new pixel should not highly depend on the function selected. Figure 2 uses the following function to calculate values of new pixels:

$$Y_i = (X_{4i-3} + X_{4i-2} + X_{4i-1} + X_{4i})/4 \qquad i = 1, 2, 3, 4$$

When the receiver receives the resolution-reduced frame, to keep the physical frame size unchanged, the system reconstructs the frame of original size by replacing every pixel with a 2 by 2 matrix. However, only an approximation of the original frame can be recovered. The

(320 x 240, CF = 1) **(320 x 240, CF = 2)**

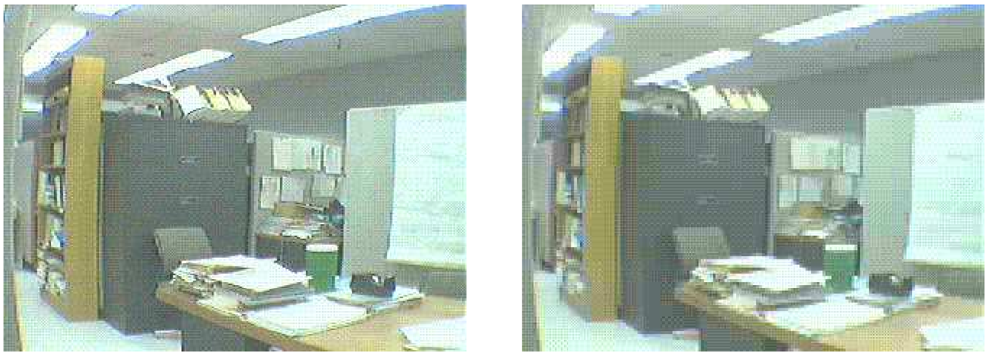

Figure 4: Effect of Frame Resolution Reduction

easiest way of reconstructing a frame is to construct a 2 by 2 matrix by replicating a single pixel, as shown in figure 2. Since this process is only an approximation and hence lossy, the effect is that the receiver gets a blurred image (see figure 4).

Since averaging does not make sense for UV components, we use a different formula to compute the UV components of new pixels, as shown in figure 3.

3.1.3 Frame Resizing

When the available network bandwidth decreases, an alternative approach to reducing the amount of video data sent over the network is to manipulate the frame size. Thus, for example, the system can make a transition from a frame of larger size (640 x 480) to a frame of smaller size (320 x 240), thereby reducing the bandwidth requirements to one-fourth. This is very similar to frame resolution reduction except that the logical frame size is also affected along with the physical frame size. That is, unlike frame resolution reduction where a frame is reconstructed to the original size before it is decoded at the receiver side, no frame reconstruction is necessary for the frame resizing before a frame is decoded at the receiver side. Therefore, the frame resizing will takes less CPU time at the receiver side than frame resolution reduction, as evidenced by the experiments to be shown in the next section.

3.1.4 Codec Scheme Adaptation

Different codec schemes can achieve different compression ratios. Though highly compressed frames take less time to transmit, they take more time to compress. So there is a tradeoff between the computation time and communication time [2]. If the available network bandwidth goes down, we can choose a codec scheme with high compression ratio to reduce the amount of data sent over the network. Of course, this is done in the cost of increasing computation time. Transcoding is a technique in which data in one codec scheme can be transformed to data in another scheme, without going back to the uncompressed version of data. This can not only save the computation time, but also suit to transmissions where the communication path consists of links with different bandwidths and capacities. This is certainly where adaptability can be applied to. However, data security is one of the major problems the transcoding scheme has. More work needs to be done before it becomes practically useful.

In order to inform the remote party of a video conferencing session of the change of the codec scheme used to encode current frames, each frame sent is associated with a *codec* type field. According to the value of this field, the remote party can choose appropriate decoders to decode the frames. However, things get complicated a little bit if the transition is from a codec scheme with interframe encoding to one without interframe encoding. The interframe dependency must be removed before the transition can take place. A natural way is to continue sending interframe-encoded frames using the old codec scheme until an intraframe-encoded frame is reached. Then the switching takes place. Thereafter, each frame is only intraframe-encoded using the new codec scheme.

3.2 Video Conference Session Recording

The needs for recording video conference sessions for future reuse are multifolds. First, the reliability and the limitation of the bandwidths of underlying networks may cause temporary interruption of conference sessions or cause the degradation of video stream quality. Secondly, users may not be able to attend a conference but may want to review it later. Thirdly, it is not easy or even possible to reach a common time for people living in different time zones to attend a conference at the same time. Fourthly, users may want to record the video sessions for editing purpose. All these require a facility to record video conference sessions into database for future reuse.

We have implemented a recording mechanism in a video conferencing software to record video sessions for future retrieval or replay. Recording a video session can be initiated either by the system to adapt to poor network bandwidth or remote site failure, or by a user to review or edit the video session in the future. Since VCR-types of operations as well as random access operation on the recorded video clips should be efficiently supported, indexing to the recorded video clips becomes crucial.

3.2.1 Indexing

The types of indexes to be built depend upon the types of operations users want to perform in order to retrieve the target frames quickly. Timing information is very important in accessing recorded video clips of video conference sessions. For example, the first five minutes of a recording usually contain the outline of a talk whereas the last five minutes usually contain the concluding remarks. A user may want to retrieve these parts of a recording first before jumping to appropriate parts for more details, or may want to stop playing back more detailed parts after browsing the outline and/or the concluding remarks. Indexing based on timing information will help users quickly find recordings of interest. Annotation is another way of building indexes. Notes made by conferees can be extrememly useful for them or others to retrieve related frames later. Annotation can be made by each conferees, or by a designated conference *secretary*, allowing other conferees to concentrate on the conference itself. The mechanism required for supporting annotation includes supporting variable length fields in database.

Carema breaks or scene changes are excellent points for an index to be inserted at because they are usually associated with changes of topics or events. While the video parsing technique [16] can be used to precisely detect camera breaks and scene changes, it can be computationally expensive. In case when real-time indexing is needed, for example, indexing while recording, some *light weight* version of video parsing technique can be used in building such indexes. For example, we can use the *distance* between two consecutive frames as a measure to indicate the degree of scene changes. If the distance is big enough, it is likely that a scene change occurs and an index should be established for the new scene. There are several ways to define the distance between two frames. The simplest way is to define the distance of two frames as their difference. The way of computing the difference between two frames varies. It can be pixel based or block based.

In some cases computing the difference between two consecutive frames is not an extra overhead because it is a built-in task. For example, some codec schemes with interframe compressoion actually compute and encode the difference between two consecutive frames. In order to save network bandwidth, some protocols do not transmit a frame if it is almost no difference from the previous one. Therefore, it can be very fast to determine if building an index for a frame is necessary.

The timestamp associated with each frame stored in the database is recorded to serve as an index for the video frame. In current implementation, we record a timestamp for each frame stored. To save storage space, we may store one frame for a series of consecutive frames with (almost) the same content. In this case, the recorded frame is actually a representative of these frames. We use a time interval (Time_Interval in Table 2) to indicate the length of that series. As we mentioned earlier, detecting and computing changes of two consecutive frames are not a problem since many softwares use them to do temporal compression, although the algorithms may be computation intensive.

Let f_1, f_2, ..., f_n be consecutive frames with few content changes. Their corresponding timestamps are T_1, T_2, ..., T_n. Initially, we record f_1 and the time interval $[T_1, T_1]$. When the successive frame comes in, the interval in turn becomes $[T_1, T_2]$, $[T_1, T_3]$, ..., up to $[T_1, T_n]$ during this process. We merely update a value of the time interval; we do not save those frames at all. Later on, when a frame in this series is retrieved by timestamp, we compare the timestamp with the interval and return f_1 if the timestamp is within the interval.

Table 1: Schema for Video Sessions

SID	Title	Speaker	Duration	Filename	Filesize	#_frames	...

Table 2: Schema for Video Frames

FID	SID	Time_Interval	File_Offset	Annotation	...

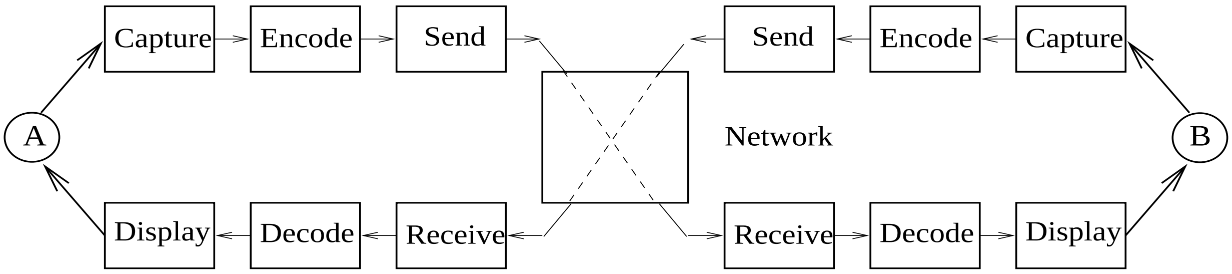

Figure 5: Modules in A Point-to-Point Video Conferencing System

3.2.2 Retrieval of Video Recordings

Currently, we construct the video database based on the relational data model. Physically, each video session is still stored as a file. But logically, it is a database tuple; the file name is an attribute of such a tuple. The related schemas for the video database can be represented with two relational tables (Table 1 and Table 2). SID and FID stand for session identifier and frame identifier, respectively. To support direct access of the recordings, the offset of the frame in the file is recorded also.

The field *Annotation* is actually a text field. Users can type anything they want into this field. Any queries on this field will be keyword-based, substring matching. If some keyword is contained in this field, the corresponding tuple will be selected.

We certainly can record as tuples video clips (or segments) rather than whole video sessions. In this case, more tables are needed to represent the hierarchical structure of video streams. This type of table structures will help higher-level retrieval of video information.

3.3 An Integrated Adaptable Video Conferencing System

We have implemented an integrated adaptable video-conferencing system [11]. Figure 5 shows the modules in a typical point-to-point video conferencing system. Figure 6 shows the

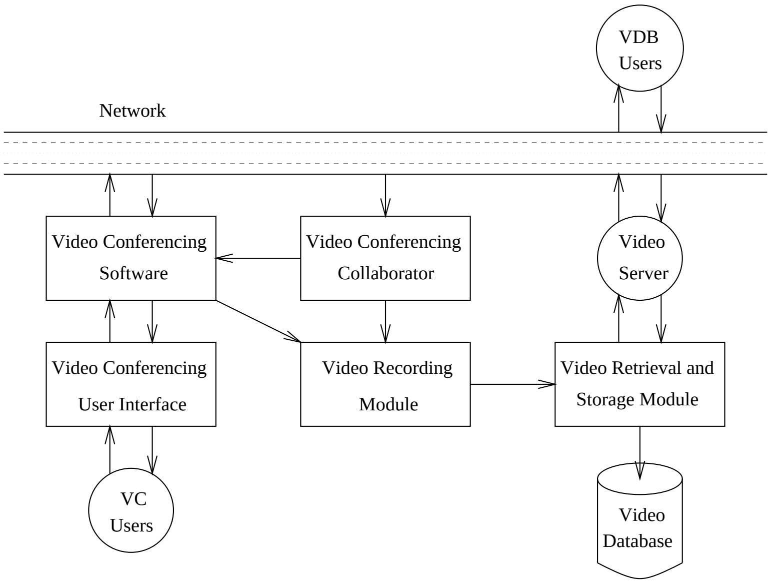

Figure 6: Architecture of the Integrated Video Conferencing System

architecture of this system that includes the video conferencing collaborator (VCC), a video conferencing software, a session recorder, a video-on-demand (VOD) system, and a video database system.

VCC dynamically monitors the system resources such as available network bandwidth, and uses the information to determine the *deliverable* quality of video to the users. This deliverable quality of video is a function of adaptable video conferencing features. VCC reconfigures these features by using such techniques as color depth compression, frame resizing, and frame resolution reduction. VCC also serves as an interface between the video conferencing software and our prototype video database. It is responsible for storing the recorded video sessions into the video database and for building corresponding indexes for them. The retrieval of the video recordings is achieved through a video-on-demand system.

3.3.1 A Distributed Video-on-Demand System for Video Conferencing

To support replay and selective retrieval of recorded video sessions by remote users, a prototype distributed video-on-demand (VOD) system that supports continuous video delivery over the Internet has been designed and implemented [3]. The prototype employs the client-server paradigm, dynamic buffering, and request-on-demand techniques to achieve high performance in video delivery.

In the VOD systems, the video servers are responsible for managing and delivering video files stored in the video database. Whenever a video server receives a database query sent by a user over the network for some video sessions, it will pass the database query to the video retrieval module for processing. The video retrieval module is actually a database engine which can process the database query and return to the video server the storage information of the requested video sessions as the query result. The video server then can locate the requested video recordings and deliver them to the user.

This VOD prototype reduces the users' response time by using the request-on-demand technique. The video server sends chunks of the video file to the user; the user displays the data that has been received while the rest of the video data is being transmitted simultaneously. This pipelining of data transmission and data display operations results in shorter turnaround times for video display. By properly choosing the buffer size and allowing it to inflate or deflate dynamically, the system performance increases by about 10% [3]. Furthermore, our system is capable of supporting multiple client requests at the same time.

This VOD prototype allows arbitrary distribution of users on the network. Any internet users are able to replay and selectively retrieve the video recordings of video conferencing sessions by making database queries to the video server responsible for maintaining the video data.

4 Experiments

4.1 Experimental Setup

4.1.1 Testbed

We chose Network Video (NV) [6], a popular video conferencing tool developed at XEROX PARC for the Internet, as our testbed to implement the adaptability and session recording features, and to conduct performance studies. We have extended the system to provide adaptability to network characteristics while at the same time maintaining the highest permissible video quality. The system's hardware-independent nature allows us to compare the overheads involved in software implementation of some algorithms of various codec schemes without having to exclude the performance benefits due to the hardware realization of certain codec schemes.

4.1.2 Platform

The platforms for the experiments included a Sun Sparc 10 station and a Sun Sparc 5 station connected in a LAN environment, and a couple of video cameras. The Sun workstations were running Solaris 2.3 operating system.

4.2 Experiments, Observations, and Discussions

4.2.1 Experiments on Color Depth Compression

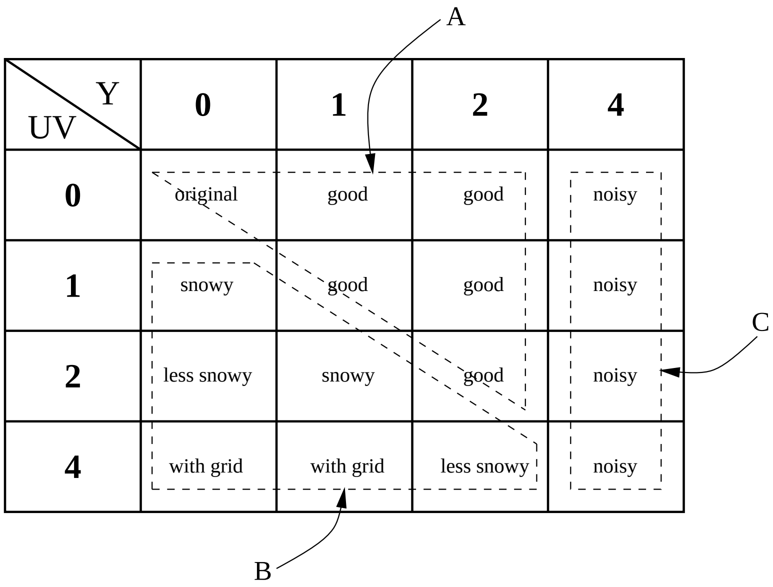

Figure 7: Image Qualities for Different Compression Factors for Y and UV Components

We have experimented with various configurations set up by separately compressing Y and UV components each by 1, 2, and 4 bits. For example, compressing UV components by 2 bits means they are represented with only 6 bits. Figure 7 shows the effect of reducing the number of bits in Y and UV components of a pixel on the quality of a video frame. We divide figure 7 into three regions: A, B, and C, as described below:

1. **Region A:** Very good image quality can be achieved when either Y components alone or both Y and UV components are compressed by a small number of bits (up to 2 bits). Thus, up to 4 bits of combined Y and UV component compression can be achieved when a very good image quality is being maintained.

2. **Region B:** Human eyes can tolerate (accept) image quality in this region when the number of bits compressed for Y components is less than 4 bits. This can be attributed to more compression in UV components than in Y components. Thus, more bits in encoding UV components *can* be reduced than those in encoding Y components as long as the compression for Y components does not reach a threshold value (4 bits).

3. **Region C:** The image quality in this region drops dramatically (though the image is still recognizable) when the number of bits compressed for Y components reaches a threshold value (4 bits in this case). Thus, we *have to* compress more bits for UV components if the bandwidth becomes too small, because the number of bits in encoding Y cannot be reduced any further.

Therefore, we can draw some conclusions as follows:

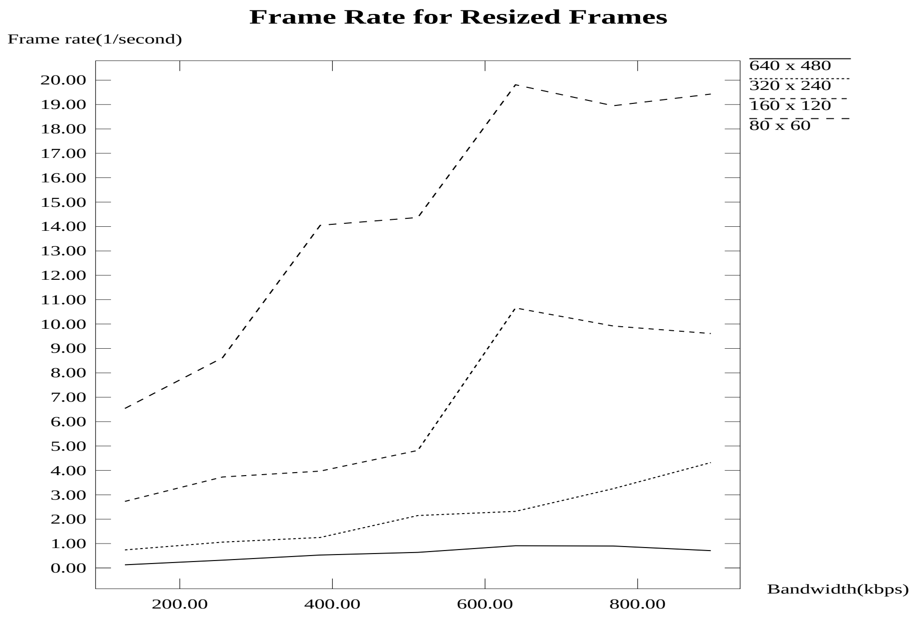

Figure 8: Frame Rates for Resized Frames

1. We stay in Region A and obtain really good image quality if the combined number of bits of compression for Y and UV components of a pixel is not expected to exceed 4 bits.
2. We enter Region B where we can get not-so-bad image quality when the combined number of bits of compression for Y and UV components of a pixel is greater than 4 bits, but is no larger than 6 bits.
3. We go to Region C when we need a higher compression rate in combined number of bits to encode Y and UV components of a pixel. But image quality is degraded substantially. Due to the relative importance of luminance components, we should *not* compress Y for more than 4 bits if we want to maintain a recognizable image quality.
4. Although the adaptability to bandwidth decrease generally means sacrifice of video/audio quality in order to maintain frame rate, it is not always the case, as evidenced by our experimental results. For example, figure 7 shows that the video quality is nonlinear to the number of bits compressed for Y and UV components.

4.2.2 Experiments on Resolution Reduction and Frame Resizing

To see how the frame resizing can be used for adaptability for video conferencing, we conducted some experiments to measure the frame rates for frames of different sizes (Figure 8). For a particular frame size (e.g., 640 x 480), as the available network bandwidth decreases, the corresponding frame rate also decreases. This will result in loss of continuity and smoothness of video presentation at the sender as well as receiver sides. In such a situation, the video conferencing system would adapt by changing to smaller frame size to maintain (or even

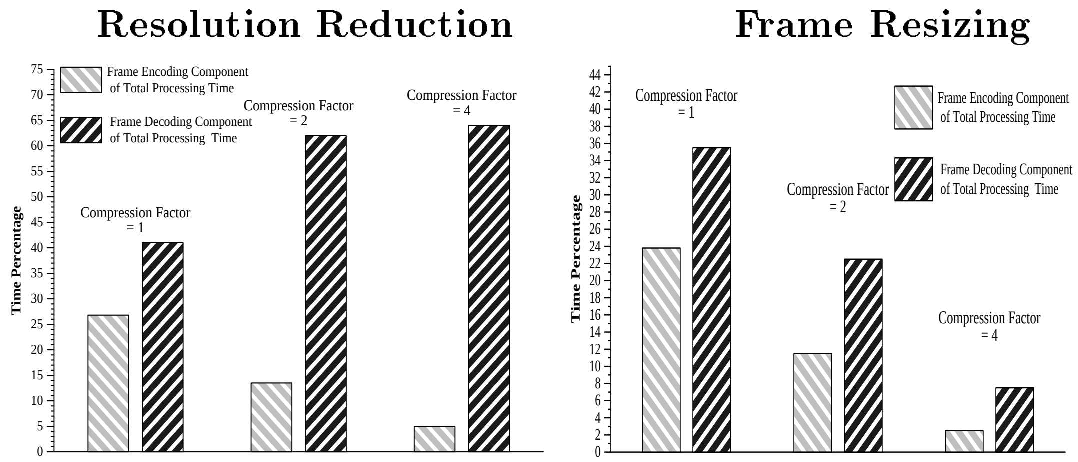

Figure 9: Impacts of Resolution Reduction and Frame Resizing on Computation Costs

improve) the original frame rate. At present, the system supports only four discrete levels of frame sizes. Thus, the frame rate may be changed (improved) when the system changes to a level of service with less bandwidth requirement. Supporting more levels of frame sizes would allow the system to adhere (very closely) to the current operating frame rate while reducing the network bandwidth requirements at the same time.

Dynamic frame resizing and resolution reduction have a great impact on computation costs. We measured the average time percentages in encoding and decoding one frame over its total processing time (figure 9). For frame resizing, as the *compression factor* (CF) increases, the combined time percentage in encoding and decoding over the total processing time goes from 60% (for CF = 1) down to less than 40% (for CF = 2) or even less than 20% (for CF = 4). For resolution reduction, as CF increases, the time percentage in encoding over total processing time goes from more than 25% down to less than 15% (for CF = 2) or even less than 5% (for CF = 4). The time percentage in decoding over the total processing time, however, goes from 40% up to near 65% (for CF = 2 and 4). This is due to the extra computation overhead involved in frame size reconstruction necessary for resolution reduction operation.

This experiment shows that although frame resizing and resolution reduction take extra CPU time, this overhead is tolerable or even negligible. For frame resizing, as CF increases as a result of bandwidth decreases, the time spent in frame transmission becomes more and more dominating in the whole process since the encoding and decoding become less expensive. This may at first seem a little contradiction with the intention that the adaptation to the network bandwidth descrease should result in a saving in network bandwidth consumption. A more careful inspection of the experimental data reveal that the network bandwidth consumption is actually reduced despite the fact that the frame tranmission time dominates the overall time in processing a single frame. The reason is stated as follows.

According to figure 8, the frame rate increases as the frame size is reduced (as CF increases), therefore, the overall time for processing one frame (including encoding, decoding, and transmission) is reduced as the frame size is decreased, considering the fact that this time is the reciprocal of frame rate. Although the frame transmission dominates the time percentage of the overall processing time (figure 9), the absolute time spent in frame transmission and in encoding and decoding decrease as CF increases as a result of the decrease of the network bandwidth. This is due to the fact that the computation time for encoding and decoding is proportional to $(m*n)*\log(m*n)$ [], whereas the time for frame transmission, including the time spent in packetizing the frame, is proportional to $m*n$, where $m*n$ is the size of a frame. When m and n are each reduced by half, the saving of computation time in encoding and decoding is much more than that in frame transmission, causing frame transmission contributes to a larger percentage of total processing time for one frame than before, though the absolute values of both decrease. Note that the amount of increase in frame rate as CF increases is smaller than the amount of decrease in the number of packets belonging to one frame, the number of packets sent over the network per time unit is actually decreased. Usually frame packetization produces packets of almost equal size. This actually shows that frame resizing (as frame size is reduced) does reduce the network bandwidth consumption.

For resolution reduction, as CF increases, the combined time percentage in encoding and decoding over the total processing time is slightly bigger than that in original NV (CF = 1), which means frame transmission is not a dominating factor in overall processing time. Similar to frame resizing, however, the average time spent in processing one frame reduces as CF increases. Therefore, the absolute time spent in frame transmission is also reduced. Using the same analysis above, we can conclude that frame resolution can also actually reduce the network bandwidth consumption.

However, figure 9 also tells us that we cannot arbitrarily reduce the frame size in order to save network bandwidth. For frame resizing, while the combined time percentage of encoding and decoding is less than 15% when CF = 4, this number will be further reduced as CF = 8 in which case the frame is hardly recognizable. Therefore, we have to seek other techniques to further reduce the network bandwidth requirement. Similarly for resolution reduction, we cannot arbitrarily reduce the frame resolution in order to save network bandwidth. Although the combined time percentage of encoding and decoding is still well above 60% as CF =4, the time percentage of encoding is only about 5% and this number will be smaller as CF = 8 in which case the frame is also hardly recognizable. Therefore, other techniques should be used in order to accommodate the reduced network bandwidth requirement.

4.2.3 Experiments on Session Recording

Session recording involves both the computation overheads and secondary storage requirements. One important parameter that we consider is *recording frequency*, the frequency at which the video frames are stored. This must be differentiated with *display frequency*, the frequency at which the video frames are updated on a user terminal. The latter determines

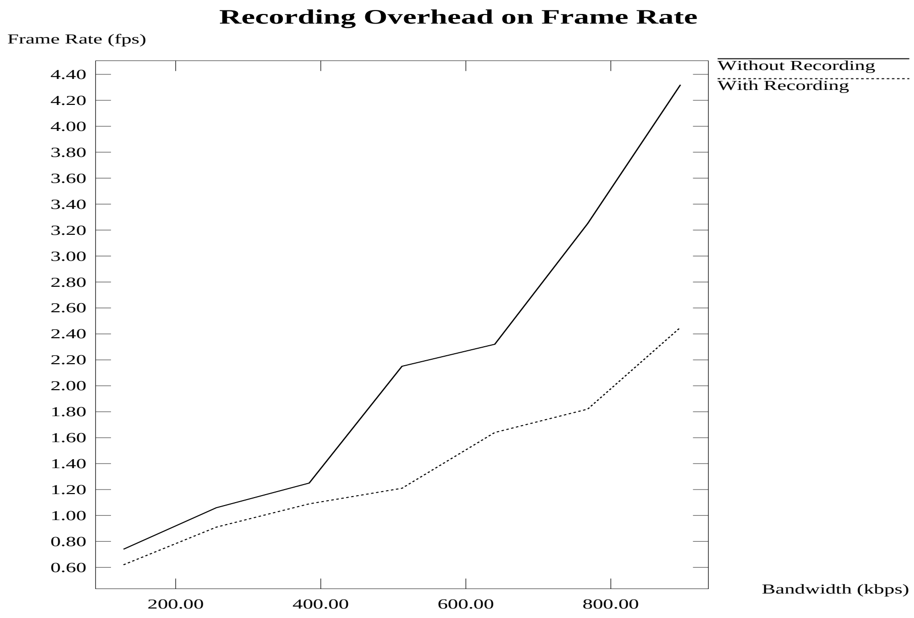

Figure 10: Effect of Recording Overhead on Frame Rates

the continuity and smoothness of the video session presentation. Storing every video frame generated during a video session is sometimes not necessary because many consecutive frames may have almost the same contents (for example, because of lack of scene changes) and because it would consume a lot of disk space. On the other hand, recording frames at a very low recording frequency may hurt the smoothness of video sessions and degrade the video quality during replay.

Recording video frames imposes computational overheads which can affect the rate at which video frames are generated for display (i.e. the display frequency). Since the system records the video conferencing sessions only when an anamoly occurs, for example, when the available network bandwidth decreases dramatically, imposing computation and network (e.g. due to storing of frames in a distributed file system) overheads may affect the frame rate (display frequency) even more significantly. Figure 10 shows the comparison between the frame rates without recording and the frame rates with recording (recording every frame) under different available network bandwidths. We observe that the overhead (signified by drop in frame rates) imposed by recording the whole session is very small when the available bandwidth is small. In other words, recording the entire session does not contribute to large performance degradation when network bandwidth becomes a bottleneck. This is counter intuitive to our reasoning presented earlier. However, this can be explained as follows. When the network bandwidth is low, the system generates and transmits fewer frames per second to remote recipients. The low frame rate results in residual computation cycles on the system which are *under-utilized.* These computation cycles can be utilized for storing the frames in the database without affecting the frame rate. On the other hand, high available network bandwidth allows the system to generate

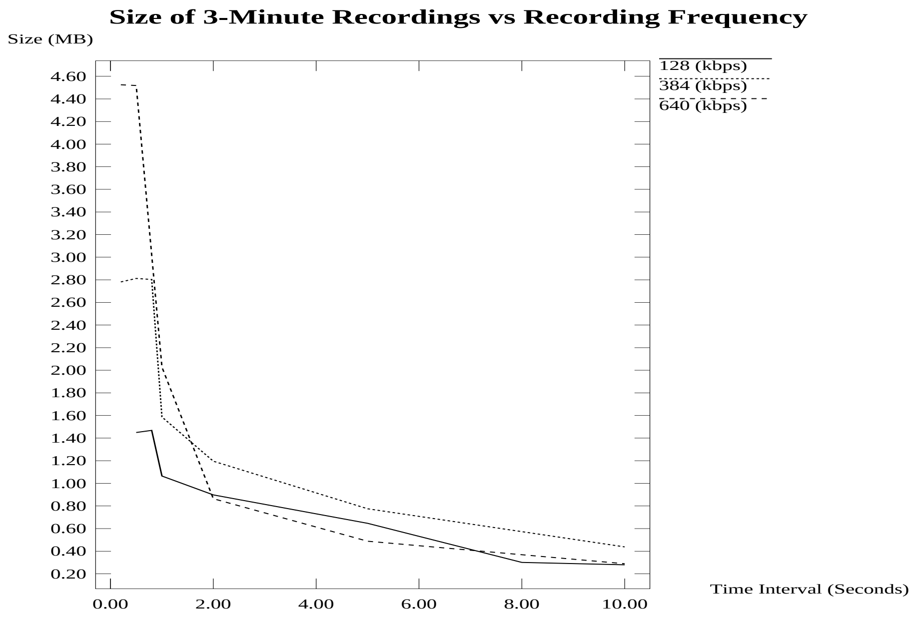

Figure 11: Effect of Changing Recording Frequencies on Sizes of Recordings

and transmit larger number of frames to remote recipients. The process for storing the frames now have to compete with the process for generating-transmitting-displaying the video frames, resulting in drop in frame rate to almost half of the original, as shown in figure 10.

Another concern in recording video sessions is the storage bandwidth requirements for recording the video frames. It is possible not to record every frame into the database, and still maintain the quality of video replay. We have conducted experiments to observe the effects of changing recording frequencies on sizes of recorded video sessions (Figure 11). As the recording frequency decreases, there is an exponential decay in size of the stored video conferencing session data. As the available network bandwidth increases, the rate at which the frame can be delivered to the remote sites also increases. Therefore, more frames are stored with the same recording frequency (time interval), thereby increasing the size of the recordings. Here, the recording frequency takes the form of time interval between two consecutive frames to be stored, hence is actually the reciprocal of the time interval. This treatment is only for convenience because time interval is easy to implement; it means certain amount of time must have elapsed before next frame can be recorded.

For a particular network bandwidth, we observe that the size of the stored video session is a constant up to a certain value of time interval for recording frames (recording frequency), beyond which it falls exponentially as the time interval between consecutive frames to be stored increases. We call this a turning point t_d for a particular network bandwidth d (see figure 11). The value of t_d actually indicates that on average a frame is generated every t_d seconds. For time interval less than t_d, the recording frequency is higher than the display frequency, i.e. the frames are stored at a rate greater than those at which they are generated. This will result in duplicating the frame being stored. To avoid storing duplicated frames, in our implementation of the recording module, one frame is stored only once. This is achieved by recording a unique timestamp in the database for each encoded frame and storing a frame only if its timestamp is different from that of the most recently stored frame. To summarize, each frame is stored without duplicates for any time interval between two frames to be stored less than t_d. Beyond this point, the recording frequency is smaller than the display frequency, and hence a subset of the generated frames is stored in the database.

Several approaches exist for reducing recording overheads and storage spaces while at the same time improving the quality of the recordings. In our current implementation, recording frequency, once specified by users, is a constant and independent of the session content. This may not be appropriate, however, because we may record more frames with fewer scene changes and less frames with more scene changes. This can be improved by recording those frames more frequently whose contents change rapidly and those frames less frequently whose contents change slowly, based on the values of interframe differences computed in encoding phase. This adaptive recording scheme will lead to better playback quality and more efficient use of storage space.

Reducing the recording frequency can save the storage spaces of the video conferencing session recordings on one hand, but may hurt the smoothness of the replay of the recorded sessions on the other hand. To seek a compromise between the storage cost and the quality of the recordings, another approach that we can use is to record the frames with their sizes reduced by, for example, a factor of 2 (i.e., CF = 2). The replay will be performed with the frames' resolution reduced after the frames of the retrieved recordings are reconstructed to the original size. This will significantly reduce the storage requirement of the video conferencing session recordings while at the same time maintaining the smoothness of the recorded sessions as well as reasonably good quality for each recorded frame.

5 Related Work

In the original NV software, the transmission rate is controled by the value of *maximum bandwidth*, which, once specified by the user, is never changed even if network is congesting until a user explicitly changes it. So this introduces the fairness problem as the network is shared by many applications running different protocols. Suppose a NV application is running in an environment where many TCP applications are running. If the network becomes congested, TCP will explicitly backs off the transmission exponentially by giving up some of the network bandwidth. However, the original NV software still consumes the same network bandwidth since it is not aware of any network congestion except that the loss rate is increased. So actually NV software effectively *steals* the bandwidths given up by other applications. In our prototype adaptable video conferencing tool, once the network congests, as evdienced by the decrease of the available network bandwidth, it will back off the transmission implicitly by reducing the network bandwidth requirement using the adaptability features describved earlier. The transmission rate is thus constrained by the reduced bandwidth.

Indexing for video conferencing has been an important research topic. [12] explores four broad classes of indexes.

- *Intentional annotations.* They are created during an activity for the purpose of marking particular time points or segments of activities.
- *Side-effect indexes.* These are activities whose primary purpose is not indexing, but which provide indexes because they are automatically timestamped and logged.
- *Derived indexes.* They are produced by automated analyses of detailed timestream records. Speaker identification indexes and scene change indexes are such examples.
- *Post hoc indexes.* They are produced by anyone who later accesses the activity records – an intentional annotation, but after the fact.

[8] discusses four paradigms for indexing video conferencing:

- Indexing by intended content. This is similar to annotation-based indexing. Both annotation and intended content which is expressed as conference agenda are some sorts of descriptions of the conference; they are created and inputed by human interactively. They can be created before, during, or after the conference. The linkage between them and actual contents of video conference sessions is achieved through some relations in databases.
- Indexing by actual content. Actual content means what actually happened in a conference, rather than what was planned for the conference. Instead of using video parsing technique directly to index video conference scenes, however, [8] developed a speech recognition system and applied it to the audio track stored with the video conference sessions to create text-based records of the sessions. The semantic clusters of words, viewed as conference topics or themes, are recovered from a discourse using a technique known as lexical chaining after the words have been recognized. Indexes can thus be built to support theme-based information retrieval through searching or browsing.
- Indexing by temporal structure. The semantics of a conference is also related to some temperoal information, for example, the points where topics are changed, the lengths of discourses and pauses, and the degree of overlay between speakers. Rather than using timestamps, [8] builds indexes by constructing an augmented transition network with appropriate transition conditions and by using simple signal processing which helps determining the lengths of speaking and pause for each participant.
- Indexing by application record. This paradigm logges any activities associated with a conference and uses them as indexes back into the audio/video streams. The events logged are application-specific operations such as object creations, deletions, and modifications, as well as other common operations such as changing focus, groups, and undoing. However, [8] doeS not present any techniques for supporting this kind of indexing.

Adaptability has been studied extensively in literature. [4] presented a model for adaptable systems for transaction processing. [5] described a portable and robust software mechanism for adaptive frame rate control for real time packet video transfer and viewing in workstation environment. However, none of the previous work on adaptability was directly applied to video conferencing systems for the purpose of dealing with resource constraints and anamoly management in distributed multimedia systems. Our work is a first step toward performing Quality of Service (QOS) control in distributed multimedia systems [10].

6 Conclusions and Future Work

We have been modifying NV video conferencing software for conducting our adaptability study. We have implemented color depth compression, resolution reduction, frame resizing, and the recording module for video conference sessions. We are now designing and implementing a new transport protocol that incorporates the adaptability parameters.

Storing video frames is expensive in terms of both the storage requirements and the I/O costs. Thus, the judicious use of stored frames is required. We are working on a computational model to determine without user intervention when to store video frames; and the frequency and quality for storing these frames based on the load and storage capacity at the sender site. We are currently doing experiments for finding out a frequency optimal for both storage and the video quality.

We are also studying mechanism for automatic and semi-automatic indexing on video frames during recording. Specifically, we are devising techniques for storing some attributes describing the video streams along with the video frames in the database. This will allow users to make high-level queries on stored video sessions.

In future, the Internet will be replaced with ATM networks [1]. With ATM technology, it is possible to guarantee network bandwidths for some applications via resource reservation. Admission control mechanism can be used to accept applications submitted earlier and to reject those submitted later. Whereas this can certainly guarantee QOS for some applications, it is too strict in the sense that sometimes the network bandwidths may be wasted because the applications may not use full bandwidths allocated to them all the time. Besides multiplexing multiple applications, we can also combine the resource reservation mechanism with adaptability. That is, for each application, the system will guarantee a minimal bandwidth with which the basic QOS can be guaranteed. Additional bandwidths for more services or better quality can be obtained by adapting the application to network conditions. That way, the system will always guarantee QOS while at the same time utilize the bandwidths to the most extent.

References

[1] B.G.Kim and P.Wang. ATM Network: Goals and Challenges. *The Communication of the ACM*, 38(2), Feb. 1995.

[2] Bharat Bhargava, Shunge Li, Shalab Goel, Melliyal Annamalai, Puqi Tang, and Yihong Zhang. Impacts of Codec Schemes on Multimedia Communications. In *Proceedings of the International Conference on Multimedia Information Systems (MULTIMEDIA 96), New Delhi, India*, pages 94–105. IETE, February 1996.

[3] Bharat Bhargava, Shunge Li, Shalab Goel, and Jin Huai. A Distributed Video-on-Demand System for Video Conferencing. In *Proceedings of the International Conference on Multimedia Information Systems (MULTIMEDIA 96), New Delhi, India*, pages 83–93. IETE, February 1996.

[4] Bharat Bhargava and John Riedl. A Model for Adaptable Systems for Transaction Processing. *IEEE Transactions on Knowledge and Data Engineering*, 1(4), December 1989.

[5] Soumen Chakrabarti and Randolph Wang. Adaptive Control for Packet Video. In *Proceedings of the International Conference on Multimedia Computing and Systems*. IEEE Computer Society, 1994.

[6] Ron Frederick. Experiences with Real-Time Software Video Compression. In *Proceedings of the Packet Video Workshop*, Portland, Oregon, September 1994.

[7] Winston William Hodge. *Interactive Television: A Comprehensive Guide for Multimedia Technologies*. McGraw-Hill, 1995.

[8] Rick Kazman, Reem Al-Halimi, William Hunt, and Marilyn Mantei. Four Paradigms for Indexing Video Conferencing. *IEEE Multimedia*, 3(1):30–40, Spring 1996.

[9] Anders Klemets. The Design and Implementation of a Media on Demand System for WWW. In *WWW'94*, 1994.

[10] Shunge Li. Quality of Service (QOS) Measurement and Control for Distributed Multimedia Systems. Preliminary Proposal, Department of Computer Sciences, Purdue University, Apr. 1996.

[11] Shunge Li, Shalab Goel, and Bharat Bhargava. VC Collaborator: A Mechanism for Video Conferencing Support. In *SPIE Photonics East '95 Symposium – First International Symposium on Photonics Technologies and Systems for Voice, Video, and Data Communications, SPIE Proceedings Vol. 2617*, pages 89–99, Philadelphia, Pennsylvania U.S.A, October 1995.

[12] Scott L. Minneman, Steve R. Harrison, Bill Janssen, Gordon Kurtenbach, Thomas Moran, Ian Smith, and Bill van Melle. A Confederation of Tools for Capturing and Accessing Collaborative Activity. In *ACM Multimedia 95*. ACM, 1995.

[13] Ronald J. Vetter. Videoconferencing on the Internet. *IEEE Computer*, 28(1):77–79, January 1995.

[14] Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann, and Jan Gecsei. Distributed Multimedia and QOS: A Survey. *IEEE Multimedia*, 2(2):10–19, Summer 1995.

[15] Carsten Vogt. Quality-of-Service Management for Multimedia Streams with Fixed Arrival Periods and Variable Frame Sizes. *Multimedia Systems*, 3(2):66–75, May, 1995.

[16] Hongjiang Zhang, Chien Yong Low, and Stephen W. Smoliar. Video Parsing and Browsing using Compressed Data. *Multimedia Tools and Applications*, 1(1):89–111, Mar. 1995.