1. Introduction

The rapid Internet expansion in resent years brought completely new means of communication and information distribution. Internet connected world in a way so much richer and more interesting, providing possibilities to express people' thoughts simply and efficiently, totally ignoring their geographical location. It allowed humans living on the opposite sides of the globe to collaborate on the same projects, combining theirs skills and knowledge.

However, paradoxically the most common methods of human communication turned out to be most difficult to accomplish in the Internet environment. Audiovisual techniques of information exchange still cause many problems to network application developers. This very advanced network structure just seems not to be designed for real-time applications purposes. Moreover, its modification is so strenuous to achieve that probably for a next few years we will have to do without it.

Those difficulties provided huge motivation to develop technologies and products to overcome the obstacles. Many compression algorithms, transport and control protocols emerged on the market. Videoconferencing products, having been rare and expensive attraction a few years ago, now appear in variety of offered capabilities and options. Yet, Internet community has a lot of work to do. None of existing products presents features that fully meet the conditions of simple and effective, audiovisual communication.

Therefore, we decided to create a videoconferencing tool that offers functionality going beyond what is available on the market, thus far. Partially, our solution is based on existing technologies and standards. Nevertheless, many problems were solved in a unique way, that based on the analysis of the difficulties met and our experience seemed to be most appropriate.

This thesis presents results of this work. It is divided into three parts. First one briefly describes available technologies, that are currently most important for real-time, network applications. Second part shows existing videoconferencing products with the analysis of their strengths and weaknesses. Based on that we created system goals presented in part three. Part three also includes system design and implementation stage description. This stage objective was to meet the assumed requirements. Final analysis of created product concludes the thesis.

2. Technology

The purpose of this chapter is to briefly present technologies and standards used currently for videoconferencing applications. It will allow to show the environment, in which our system was designed and justify utilization of some technologies and rejecting others.

This chapter is divided into two parts: data compression and multimedia standards. The former describes the most crucial element of modern multimedia applications, without which creating any videoconferencing system would be impossible. The latter presents several existing multimedia standards, which definitely facilitate design and implementation of audio-visual applications, creating foundation for such applications in Internet environment. However, the primary goal of those standards is to create network components common for every application, allowing wider interoperability and easier collaboration.

2.1 Data compression

Data compression is the reduction in the amount of space that must be allocated for the information. It allows decreasing the transmitted signal bandwidth, what is particularly important in current network environments where bandwidth is the most crucial constraint.

Variety of compression techniques may be divided into two classes: redundancy reduction and entropy reduction. Redundancy reduction techniques remove or reduce the redundancy in the information; however, enabling these modifications to be inverted and data structures to be reconstructed. Thus, redundancy reduction does not introduce any signal distortions or information losses. Term entropy refers to average information in data. Therefore, entropy reduction algorithms reduce information, which cannot be recovered. Hence, this method is irreversible and introduces signal distortion and losses.

There exist many general-purpose compression algorithms. Unfortunately, they do not work very efficiently, as far as audiovisual data are concerned. Accordingly, special methods dealing with these types of information were introduced. They take into consideration specific characteristics of audiovisual data, utilizing them to remove redundant or unnecessary information.

Following two sections address those issues, analyzing audio and video formats and presenting techniques to reduce the allocated space. They describe the existing standard solutions, chosen according to their effectiveness and performance. Most of them are widely used in variety types of multimedia applications.

2.1.1 Audio compression

Transporting uncompressed digital audio takes significant bandwidth. Therefore, in most videoconferencing system various compression algorithms are used. Some of them are applicable to general audio signals; others are specifically designed for compression of human speech.

2.1.1.1 Digital audio

Frequencies perceived by human ear as sounds are typically between 20 Hz and 20 kHz. Human voice can produce frequencies between 40 Hz and 4 kHz. Sounds to be digitized must be converted from the continuous time and value domain into the discrete one.

Process of converting analog sound in time domain is called sampling. According to Nyquist's theorem, the signal has to be sampled with rate at least twice higher than its maximal frequency. Typically computer audio cards offer sampling rates from 8 kHz to 44.1 kHz depending on required quality. 8 kHz sampling rate is sufficient for the human voice sampling.

Then audio signal is quantized (converted to discrete values). During this process some information is lost, which results in appearance of quantization noise. Its influence on audio quality depends on the number of quantization levels. Increase of the number of quantization levels leads to necessity of coding audio sample with more bits. Usually this number varies from 8 bits (256 levels - voice quality) to 16 bits (65536 levels - CD quality). Because human ear is not equally sensitive to signal changes at different levels, some quantization methods (e.g. PCM) are not uniform, but logarithmically spaced. It improves audio quality without changing the sample size.

2.1.1.2 ADPCM

Adaptive Differential Pulse Code Modulation (ADPCM) belongs to the class of predictive coding methods. [2][1] Predictive coding, based on previous signal samples, predicts the value of the current sample and encodes only the difference between the prediction and the real value. This approach takes advantage of the fact, that for most data sources the variance of the original data is higher than the variance of the difference. Thus, range of compressed data is lower and it is possible to encode it with less number of bits. In stationary, highly correlated signals, this operation allows to remove data redundancy.

Methods in this class differ in the algorithm of finding the predictor. The simplest predictor is just the previous sample value. Usually, linear prediction techniques are used. In this case the predictor is a sum of a number of previous samples weighted by appropriate coefficients. Coefficients are chosen to minimize the error between original samples and decoded values. Usually, the Mean Square Error (MSE) function is used to calculate compression error:

$$MSE = E [(S - \Psi)2]$$

Where: E - expected (mean) value

S - actual signal value

Ψ - predicted value

Predictive coding with fixed coefficients works very well only for stationary signals. If signals characteristics change (signal is not stationary), the set of coefficients is not optimal anymore. Audio signals are rather locally than generally stationary. It means that for a short terms their characteristics are constant, but they do change on long-term bases. Therefore, for audio signals it is more suitable to use techniques, where coefficients are regularly updated. This solution imposes another difficulty connected with an algorithm of coefficient modification. In terms of computational intensity the best solution is an adaptive method, where there is a fixed coefficient and an adaptive quantizer, which added to the coefficient, gradually converts it to the optimal value.

Disadvantage of those methods is an error propagation – error in a transmission of one sample affects all consecutive samples, since this sample value is used to reconstruct the value of the next one and so on. It may result in unwanted effects in decoded signal. There are a few ADPCM standards such as Intel/DVI or G.721. The Intel/DVI algorithm is not very computationally intensive, and the quality of compressed audio is reasonable good even for music.

2.1.1.3 GSM 06.10

GSM (Global System for Mobile Telecommunication) is telephony standard defined by the European Telecommunications Standards Institute (ETSI). The GSM 06.10 is a voice coding technique that utilizes a priori information about the human speech, in particular the mechanism that produces it. [3][1]

Human voice is created by vibrations of the vocal cords. Those vibrations before leaving the mouth go through the vocal tract containing the throat, the tongue, the teeth and lips. During short period of time a sound made by the human voice remains constant. These atomic speech sounds are called phonemes. There are two types of phonemes: voiced and voiceless. Voiced phonemes are pulses of given pitch created mainly by vocal cords. Voiceless phonemes appear as a result of blowing air through the vocal tract. Vocal tract is shaped according to the type of sound we want to make.

Vocal tract can be physically modeled as a system of connected lossless tubes. At the tube boundaries a wave is reflected and interferes with other waves. Given the coefficients of those reflections we can simulate impact of the vocal tract on the speech. However, one has to be aware of some differences that affect the accuracy of this model: vocal tract is not lossless, its walls resonate with voice waves, and there is also nasal tract.

We can realize this model according to the Figure 1. White noise generator simulates the voiceless phonemes, impulse generator – voiced phonemes, adaptive filter – vocal tract. We need to provide following information:

- type of phoneme (voiced or voiceless)
- pitch
- signal strength
- filter coefficients

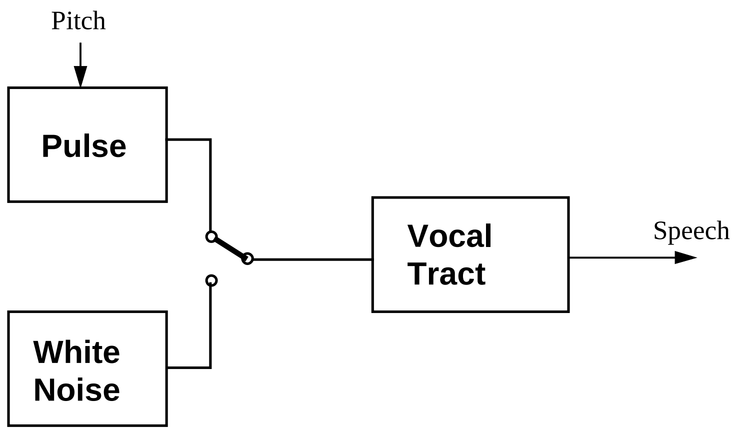


Figure 1 The source-filter model of speech production

Based on this information (instead of the signal waveform), it is possible to replicate input signal, simulating the speech production. Still, there is a problem connected with design and implementation of vocal tract filter. The solution is long-predictive coding filter (LPC) or actually its modification used by GSM standard – RPE-LTP (regular pulse excitation long-term predictor).

Architecture of GSM codec is presented on Figure 2. Short-term analysis calculates the residual signal – the signal which going through vocal tract filter produces the sequence closest to the original samples. Long term analysis determines the LTP gain and lag. LTP gain is a scaling factor for the samples – according to the scaling factor during the RLE encoding stage they are scaled down and packed into 3 bits units. LTP lag decides of the size of the window in which sequence will be analyzed, filtered and retrieved. In order to do that, long-term analysis calculates the signal correlation (to find the sequence that matches best the current samples). Finally, to be able to put 40 audio samples into 47 bits, RPE encoding removes every two out of three samples (since there is no need to transport precise signal – especially voiceless phonemes), first subtracting the long-term predictable values (we are able to retrieve them on the decoding side).

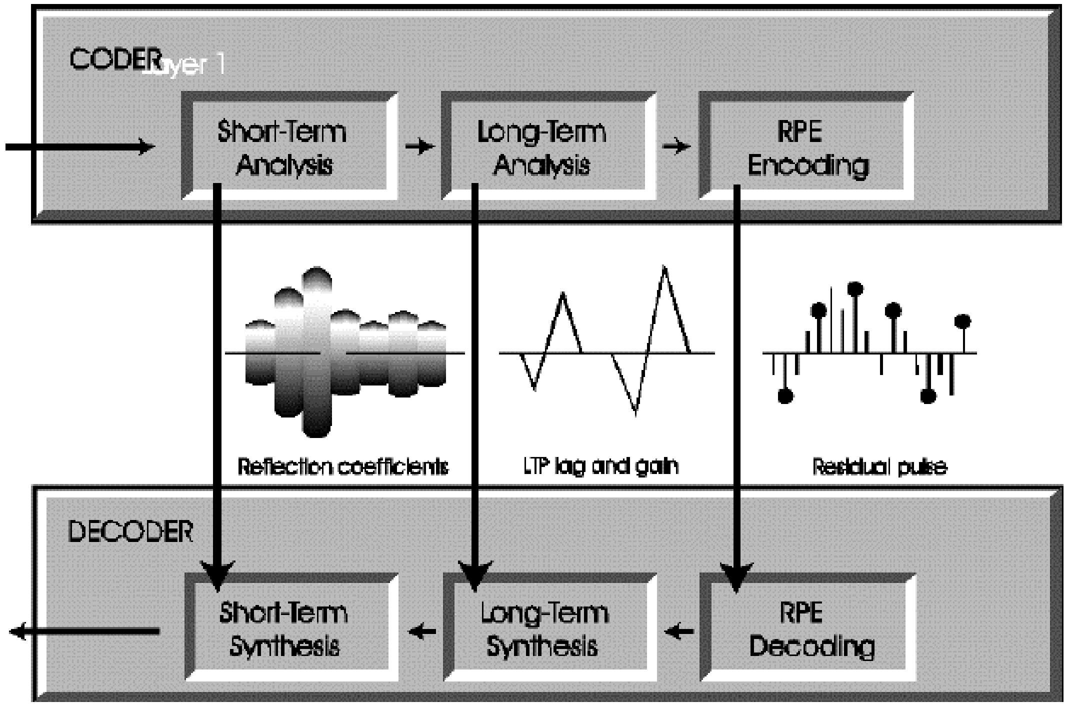


Figure 2 GSM codec architecture

Decoding proceeds in the opposite directions. First, encoding samples are expanded to full size 40 samples (RPE decoding). Then, they are multiplied by the gain factor and added to the incoming pulse (long-term synthesis), which is in turn filtered by the short-term synthesis.

GSM encoder compress 160 16-bit voice samples into 264-bit GSM frame. GSM 06.10 is faster than codebook lookup algorithms such as CELP. It offers 13 kbps bandwidth.

2.1.2 Video compression

An uncompressed digital image is usually too large to effectively store, treat or transmit through the network. Many compression techniques have been devised to compress digital video.

Requirements for an efficient video codec are very demanding: it must have a high compression ratio, keep the image quality and be fast. Additionally, lossless compression methods such as Huffman, Arithmetic Coding or LZW do not work well on the images. It was hence necessary to create compression techniques designed specially for video. At present, the most popular algorithms for low bandwidth video compression are based on the ITU H.261 recommendation.

2.1.2.1 Video formats

To represent any color distinguished by a human eye three primary components are necessary. This phenomenon is widely used in television and video technology. By mixing three-color components, red, green and blue (RGB), it is possible to display all other colors. In RGB representation, a certain number of bits is assigned to represent each color component. The most demanding applications call for an 8-bit representation for each R, G, and B, for a total of 24 bits (3 bytes) per pixel. This representation is known as "True Color".

Color images and videos are notoriously demanding in their storage requirements. For practical networked applications, images and video must be compressed. We will discuss basics of the image and video compression later. The physiology of human eye suggests that, as a first step towards compression, a more efficient color representation is possible by isolating the luminance signal responsible for brightness and the two chrominance components describing colors. Since the human eye is less sensitive to color than to brightness changes, it is possible to reduce the amount of information in chrominance components without affecting the image quality. Another advantage of this format is concentration of the image information in the luminance component, which results in reduced element correlation. Hence components can be compressed separately without much efficiency loss. Image size reduction is achieved by subsampling chrominance components at the rate of 2, 4, or even 16 times lower than the luminance component.

Television standards use different types of color spaces. In European's PAL and SECAM, YUV color space is used, while YIQ representation is used by NTSC standard. In both representations the Y component (luminance) is the same. For digital applications, the equivalent of YUV is YCbCr where Cb and Cr correspond to U and V components respectively. This representation allows normalization of color components ranges. If the luminance range is between <0; 1> the UV range is undefined but Cb and Cr ranges are between <-0.5; 0.5>.

The conversion between RGB and YCbCr formats can be represented by the following matrix transform:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix}$$

Additionally, H.261, H.263 and MPEG conversions perform slightly different scaling. According to CCIR Recommendation 601, color space is re-scaled and shifted according to the following formula: Y' = 219/255 * Y + 16, Cb' = 224/255 * Cb + 128, Cr' = 224/255 * Cr + 128.

2.1.2.2 H.261 compression algorithm

H.261 is the International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) low bandwidth video compression algorithm. [6][7] H.261 compression uses two different techniques enabling effective video compression:

- spatial redundancy removal – both INTRA and INTER frames coding

- temporal redundancy removal – INTER frames coding.

Spatial redundancy removal

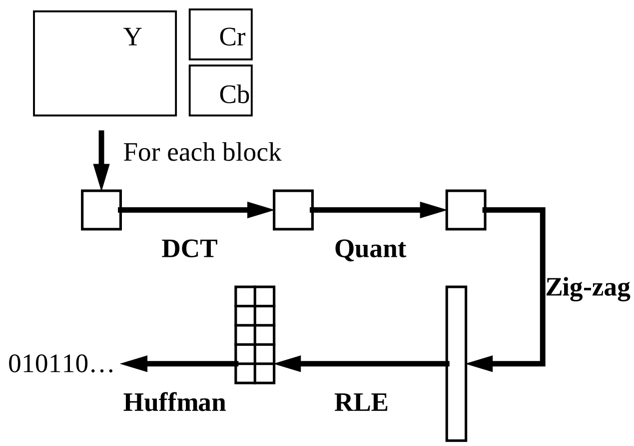


Figure 3 INTRA frame coding

Spatial redundancy removal is a method based on the Discrete Cosine Transform (DCT). The YUV image is divided into blocks 8x8 pixels each. DCT is performed on each block according to the following equations:

$$F[u,v] = \frac{1}{4}\sum_{i,j} \Lambda(u)\Lambda(v)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}\, f(i,j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for \xi = 0 \\ 1 & otherwise \end{cases}$$

After DCT, subsequent block values represent information for increasing signal frequency. Frequency coefficients could also be obtained using FFT (Fast Fourier Transform) but FFT is slightly less stable as a computational method.

Since the human eye is less sensitive to higher frequencies, it is possible to eliminate them without losing much of the image quality. Such elimination, or actually reduction, of the component information is achieved during the coefficient quantization step. Each block value is divided by the proper constant collected in the quantization table. There are different quantization tables for luminance and chrominance signals. Scaling quantization tables by a certain factor increases or decreases image quality. Additionally, custom quantization tables can be used and put into the image header.

The next step of the compression process is the Zig-Zag scan. It converts a two-dimensional 8x8 block into a 1x64 vector, grouping low frequencies at the top of the vector. On such a prepared vector the standard RLE (Run Length Encoding) and Huffman compression algorithms are performed.

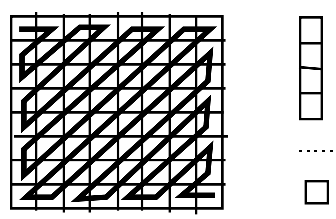


Figure 4 Zig-Zag Scan

Temporal redundancy removal

The temporal redundancy removal technique uses the correlation between two consecutive frames in a video stream to reduce their compressed size. To achieve this goal, the encoded H.261 stream consists of two types of video frames:

- INTRA (I-frames) -- coded only with the spatial redundancy removal technique

- INTER (P-frames) – “predicted”, coded based on the “pseudo-difference” from the previous frame.

The first stage of INTER frame coding is the motion vector estimation process. For each macroblock in the coding frame (a macroblock is a structure consisting of four luminance blocks and two chrominance blocks, Cr and Cb), the area of the previous frame (reference frame) that matches it best is found. Then, instead of coding the macroblock, it is enough to code the difference between the best match area and add the information about the motion vector to the compressed stream. Such information can be put into the macroblock header. The difference is then coded using spatial redundancy removal, the only difference being separate quantization tables for INTER frames.

Bitstream structure

As we mentioned above, to efficiently compress a video stream, apart from the video signal data we also need some additional information. Thus, it was necessary to create an H.261 video stream structure to enable efficient information retrieval by the decoder. A simplified scheme of the H.261 bitstream structure with a brief description of its fields is presented below.

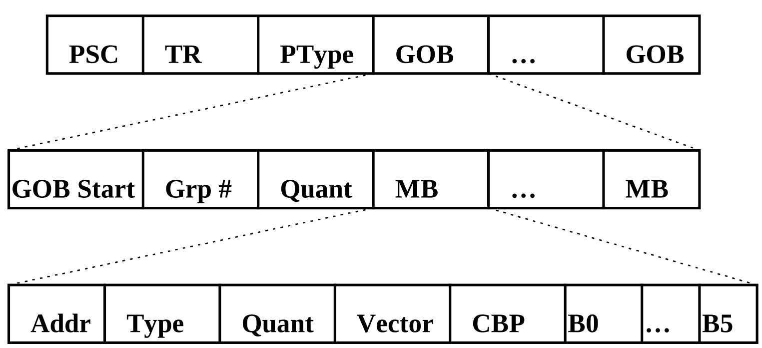


Figure 5 H.261 Bitstream structure

Table 1 H.261 Bitstream fields description

Field	Description
PSC	Picture Start Code, unique sequence to delineate boundaries between pictures
TR	Temporal Reference, timestamp used for audio synchronization
Ptype	Picture type (INTRA / INTER)
GOB	Group of Blocks
Grp #	Group number, enables skipping the whole group indicating its number
Gquant	Group Quantization Value **-** common value used for quantization of whole group
Addr	Address of macroblock in case it is exact match
Type	If good match cannot be found block may be coded in INTRA mode
Quant	Quantization Value for macroblock
CBP	Coded Block Pattern, bitmask indicating which blocks are present in case they matched poorly

Motion vector search

Motion vector estimation is the most computationally intensive part of the H.261 encoding process. Hence, it is very important to implement an effective method of search. Thus far multiple algorithms have been used for this purpose. Usually a faster method means lower video quality. The two most important motion estimation algorithms are:

- **full spiral search --** checks all the positions in the searching area starting from the middle and moving spirally outward

- **two-dimensional logarithmic search** – similar to a binary search, checks only a few locations in the area then continues searching around the location that matched best. A logarithmic search is the least computationally intensive; however, it does not provide the highest quality

2.1.2.3 H.263 compression algorithm

H.263 is an improved version of H.261 compression algorithm. [4] The idea of image coding is the same but some additional elements were added which significantly increased the compression efficiency. At the same percepted image quality it was possible to reduce bit stream size by 30-50%. The main factors that enabled this significant progress are:

- half pixel motion vector prediction
- negotiable options
- improved variable length coding (VLC) tables

The most important of those new features is introducing the negotiable options. [5] Those options should be negotiated between the encoder and the decoder at the beginning of the media transportation process. The encoder should be capable of utilizing all of them during the compression stage to take a full advantage of the H.263 effectiveness. Decoder may implement only a subset of the negotiable options, requesting data streams with only those options, it is capable to decode. Negotiable options are briefly described below.

- **Unrestricted Motion Vector (UMV) mode**. In this mode motion vectors are allowed to point outside the picture. “Non-existing” pixels from outside of a picture are reconstructed based on the edge pixels. This mode offers extensive advantage during movements along the edges of a picture (including camera movements)
- **Advanced Prediction mode**. In addition to 16x16 motion vector for some macroblocks four 8x8 vectors are used. Encoder decides which type of vectors to use. Four vectors take more

bits but offer better prediction.

- **Syntax-based arithmetic coding mode**. Instead of VLC coding this mode utilizes arithmetic coding. It improves compression ratio of 3-4%, keeping SNR at the same level.
- **PB-frames mode**. In this mode two consecutive pictures are coded as one unit, similarly to the MPEG compression. There is one frame predicted from the last decoded frame (P) and one predicted bidirectionally (B) from the last decoded frame and currently decoded P-frame. For simple video sequences it allows doubling the frame rate without increasing the bandwidth.

2.2 Multimedia standards

Modern network environments are a perfect field for applying standards. Eventually, computer networks are all about connecting people and standards provide the means for that. Unfortunately, there are several problems with the existing standards and their applications. First of all, many standards do not define the issues detailed enough to allow the real interoperability. In addition, the extremely fast network evolution makes some standards outdated. Another group of problems connected with standards is the vendors' approach. They are often more interested in products differentiation than interoperability. Many products are "partially" compliant to the standards, what really makes them unusable in the multi-applications systems.

This section briefly presents existing multimedia standards, which may provide videoconferencing applications interoperability. The standards presented below deal with the general issues in real-time multimedia applications. Other standards describing particular audio and video compression algorithms are presented in Chapter 2.

2.2.1 Real-Time Protocol (RTP)

RTP is a transport protocol for real-time applications. [14] It provides functions to transport real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not cover resource reservation and does not guarantee quality of services. Services provided by RTP include payload type identification, sequence numbering, timestamping and delivery monitoring. To perform the latter RTP is augmented by Real-Time Control Protocol (RTCP). RTCP conveys information about the participants and on-going sessions enabling loosely coupled sessions – without membership control and session set-up.

RTP presents a new type of protocol, rather integrated in application processing than implemented as separate layer. RTP is intended to be modified according to application needs. Therefore, for a particular application it requires the specific profile and payload format definition.

RTP is designed to allow users the maximum flexibility. If both audio and video media are used in conference, two separate RTP channels are created. The reason for this separation is ability for some participants to receive only one medium. Real-time protocol addresses also issue of mixers and translators. Mixers and translator are used whenever needs of some users are different then others. One example is a low bandwidth participant connected to a high-bandwidth conference. Instead of reducing the bandwidth (and quality) for whole conference, it is possible to place a mixer on the way to the low-bandwidth user, which translates media streams reducing its bandwidth. Other examples of mixers and translators applications are firewall funneling, different network protocol translating and group scene video mixers.

2.2.1.1 RTP header structure

RTP header format is presented in Table 2. First twelve octets (up to SSRC) are present in every RTP packet. CSRC fields are optional.

Table 2 RTP header structure

V=2 **2**	**P** **1**	**X** **1**	**CC** **4**	**M** **1**	**PT** **7**	**sequence number** **16**
timestamp **32 bits**						
Synchronization source (SSRC) identifier **32 bits**						
Contributing source (CSRC) identifiers **32 bits** **…**						

Short description of RTP header fields:

- **version (V):** 2 bits

  identifies version of RTP (current version is 2)

- **padding (P):** 1 bit

  indicates if packet contains additional padding octets at the end (they are not the part of the payload), padding octets are needed for some applications

- **extension (X):** 1 bit

  indicates if packet contains the header extension, specified by the standard

- **CSRC count (CC):** 4 bits

  a number of CSRC identifiers following the fixed header

- **marker (M):** 1 bit

  an additional bit enabling marking of special events (e.g. frame boundaries), its interpretation is defined by the profile

- **payload type (PT):** 7 bits

  identifies a packet format enabling its proper interpretation by an application, there are several standard payload types defined by the RTP companion profile

- **sequence number:** 16 bits

  a number of packets in a sequence, enables the packet loss detection and restoring packets sequence

- **timestamp:** 32 bits

  the clock value at the moment of packet sending, allows media streams synchronization and network jitter calculation, the timestamp resolution is the application dependent and specified by the profile

- **synchronization source (SSRC):** 32 bits

  a randomly chosen identifier for the synchronization source, intended to be unique

- **contribute sources (CSRC) list:** 0-15 items, 32 bits each

  a list of contributing sources, number specified by CC field, inserted by mixers to specify sources that contributed to the packet content

2.2.1.2 RTP control protocol (RTCP)

RTCP is created for periodical transmission of control packets to all the participants in the session. RTCP uses the same distribution mechanism as data packets. Control protocol channel is separated from the data channel. As a result, we obtain data and control independence, allowing more application flexibility.

The primary function of RTCP is to provide a feedback on the quality of data distribution. Each session participant receiving data sends reports to all the participants. It allows detecting and locating problems and evaluating if they are global or local. RTCP carries the persistent identifier for RTP source – CNAME (based primarily on user IP address). The difference between SSRC and CNAME is that SSRC is changed each time session is restarted or conflict has occurred. Also, there may be several SSRC for one user (CNAME) identifying many sessions in which the user participates. CNAME in connection with other, optional information about participants convey session control identification data and may be used to control the number of participants in session and simple session management. To obtain this goal RTCP packets must be sent by all participants at the same, controlled rate.

RTCP specification defines several RTCP packet types:

- **Sender Report (SR)** – transports statistics from active senders
- **Receiver Report (RR)** – transports statistics from all other participants
- **Source Description Items (SDES)** – convey source description items (in particular CNAME)
- **BYE** – indicates session leaving
- **APP** – reserved for application specific functions

SR and RR packets should be sent as often as bandwidth constraints will allow. SDEC is sent as soon as CNAME identifier is obtained.

RTCP packets allow controlling the number of session members, allocating the required bandwidth and controlling transported data based on the session reports.

2.2.2 ITU-T H.323

The H.323 standard describes visual telephone systems and equipment for local area networks without guaranteed quality of service (QoS). [15] H.323 presents systems and devices capable to carry real-time audio, video and data, or any combination of these. H.323 uses logical channel signaling procedures described in recommendation H.245, including capabilities negotiations.

This recommendation defines components of H.323 system: terminals, gateways, gatekeepers, multipoint controllers, multipoint processors and multipoint control units. It addresses also issues of components interactions and communication. H.323 components are divided into two classes: endpoints and entities. Endpoints participate directly in call set-up procedures. Entities are contacted to provide additional functionality.

2.2.2.1 Terminals

A terminal is an endpoint capable of creating two-way, real-time communication channel with another terminal, gateway or multipoint control unit. The terminal provides means for communication control, indications, audio, video and data transportation.

Terminals consist of user interfaces, video and audio codecs, telematic equipment, H.225.0 layer, control functions and network interface. Their structure is shown on Figure 6. Elements within the scope of H.323 are briefly described below.

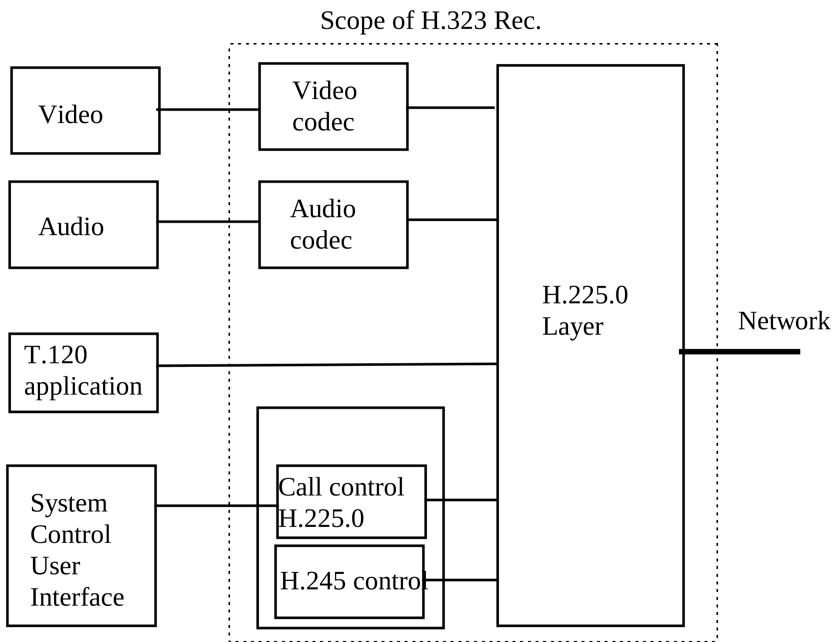


Figure 6 H.323 Terminal equipment

Audio codec

Audio codec is a mandatory element of every H.323 terminal. It supports at least the G.711 audio formats, both A-law and u-law. Optionally, it can be capable of encoding and decoding audio using other formats (e.g. G.722, G.728, MPEG 1). The coding option is set during capability exchange stage. The audio codec should be able to operate asymmetrically i.e. send and receive audio in different formats. Additionally, it may send more than one audio channel at the same time (e.g. speech in two languages). Audio stream is packed according to the recommendation H.225.0.

Video codec

Video codec is an optional element of H.323 terminal. All terminals supporting video should be capable to encode and decode H.261 QCIF video. Additionally, they may support other H.261 and H.263 formats. Similarly as for the audio codec, the video option is set during the capability exchange stage and a terminal can send more than one video channel.

Data channel

H.323 provides also support for other than audio and video data exchange. Particularly, it allows integrating applications implemented according to the T.120 standard with the H.323 terminal. T.120 connection establishing procedures are performed either during H.323 call as its inherent part or prior to the call. T.120 operation after connection establishing is outside the scope of H.323 recommendation.

H.245 control function

H.245 defines control protocol for multimedia communication. It is utilized to manage operation of H.323 endpoints. H.245 provides following functionality:

- master/slave determination
- capabilities exchange
- logical channel signaling
- mode request
- round trip delay determination
- maintenance loop signaling

The most important task of the H.245 procedures is logical channels establishing. Logical channels are created for audio, video, data and control information. They are unidirectional or bidirectional. Terminal can open many logical channels for each media type except control channel that must be one per call.

H.225 layer

Except logical channels, terminal uses signaling channels for call control and gatekeeper related functions. Those channel formats are defined in recommendation H.225.0

2.2.2.2 Gateways

A gateway is H.323 entity that is responsible for conversions between data formats and control protocols. In particular gateway provides a bridge between local area networks and switched circuit networks (SCN e.g. ISDN). In this case gateway performs all the call set-up and clearing procedures on both LAN and SCN side and is responsible for all data formats translations. Functionality provided by gateway is transparent for the endpoints.

2.2.2.3 Gatekeepers

A gatekeeper is an optional H.323 system component, which provides call control functionality to the endpoints. It is a logically separate element; however, it can be physically combined with the terminal, gateway or multipoint control unit. A gatekeeper performs following functionality:

- **address translation** – translates alias address to transport address
- **admission control** – authorizes sessions access
- **bandwidth control** – sets sessions bandwidth limits
- **zone management** – creates and manages separate zones within which sessions take place
- **call control signaling** (optional) – takes over some call control functionality from

terminals

- **call authorization** (optional) – decides if a call is accepted or rejected
- **call management** (optional) – keeps information about on-going calls and takes appropriate actions

2.2.2.4 Multipoint controllers (MC)

A multipoint controller is responsible for creating and managing conferences between three or more endpoints. It performs capabilities exchange on behalf of every participating endpoint. All capabilities may be common for all nodes or some of them may have different set. MC is able to change the capabilities during session. MC may be located in multipoint control unit, terminal, gateway or gatekeeper.

2.2.2.5 Multipoint processors (MP)

MP is a multipoint system component that converts audio, video and data streams. It may change their formats and mix or switch among streams from different endpoints. It communicates with the MC; however, this process is not a subject to standardization. Usually MP resides in the Multipoint Control Unit (MCU) together with the MC.

2.2.2.6 Multipoint Control Units (MCU)

MCU is an endpoint that supports multipoint conferences. It consists of MC and several MPs. Functionality of MCU is based on recommendation H.243. MCU provides centralized, decentralized or hybrid multipoint capabilities. It may perform additional operations such as conference rate matching or lip synchronization.

2.2.3 ITU-T T.120

T.120 is series of recommendation describing protocol integrating multiple data formats in multipoint conference. [9] T.120 has a support for real-time audio-video applications. It provides:

- support for establishment of a conference
- participant identification and capabilities exchange mechanism
- mechanism for setting up flexible communication channels

Table 3 T.120 series of recommendations

T.121	Generic Application Template (T.GAT)
T.122	Multipoint Communication Service for Audiographic Conferencing: Service Definition
T.123	Protocol Stacks for Audiographic and Audiovisual Teleconference Applications
T.124	Generic Conference Control
T.125	Multipoint Communication Service Protocol Specification
T.126	Multipoint Still Image and Annotation Protocol
T.127	Multipoint Binary File Transfer
T.128	Audio Visual Control for Multipoint Multimedia Systems

2.2.3.1 Structure

T.120 has a layered structure presented on Figure 7. Each layer provides services to the layer above, using functions performed by the layer below. Communication between layers is performed by sending Protocol Data Units (PDU). It allows extendibility, network and platform independence and interoperability. T.120 does not impose any constraints on conference topology, however the most common is star topology with single Multipoint Control Unit

(MCU).

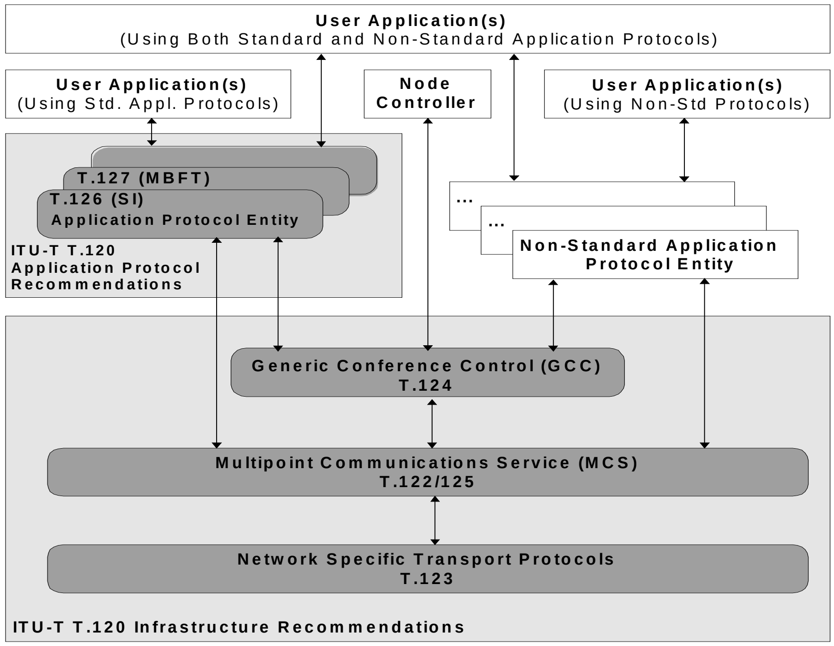


Figure 7 T.120 Structure

Top-level layer is the **user application** with its protocols both standard and non-standard. The series includes a set of protocols for the most popular applications. It also provides the Generic Application Template that distinguishes and describes the functionality common for all collaborative applications. At the same level we have also the **node controller** existing in all systems, performing conference management functions and providing an interface over Generic Conference Control (GCC) layer.

Layers below create communication infrastructure that provides multipoint connectivity with reliable data delivery. It includes three standardized components: Generic Conference Control (GCC), Multipoint Communication Service (MCS) and Transport Protocol Profiles for each of the supported networks.

Generic Conference Control (GCC)

GCC, described in T.124, provides set of functions for creating, managing and terminating conferences. It combines independent MCS channels into one multipoint domain. GCC serves also as a central point that identifies application channels, maintain applications database and exchange application information and capabilities. Nodes can query GCC about on-going conferences, and perform all the operations necessary to join it.

Multipoint Communication Service

MCS (T.122/T.125) uses point-to-point connections provided by the network layer and collects them to form a multipoint channel. It is independent of underlying network connections. MCS organizes the conference nodes into a tree structure. This way it acts as a resource (channels and tokens) provider to layers above. It transports data stream to other nodes without any knowledge about its content.

MCS distinguish two kinds of channels: *static* and *dynamic*. Static channels have preassigned definition and are open in a sense, that any node can join it. Dynamic channels are created on request. They come in two types: *multicast channels* that have open access similar to static channels, and *private channels*. Private channels have their owner (the node that created the channel) and are joined by an invitation only.

MCS data are sent in two ways: *ordinary data* – sent to destination by the shortest route, hence streams from different sources may come in different order for different users; and *uniform sequenced data* – sent through a common point and received by all users in the same order.

2.2.4 MBone

MBone is the Internet backbone for distributing real-time multimedia data to multiple destinations. MBone was created in 1993 by Internet Engineering Task Force (IETF). Its name was inspired by the name of the European backbone network “EBONE”. In recent years it became very popular in the Internet community.

The basic idea behind the MBone is the concept of multicast. Normally, Internet packages are transmitted in the point-to-point unicast mode. It means that to achieve the one-to-many transmission, separate copies of data must be sent to each destination. This process is very bandwidth consuming. The multicast introduces a more efficient way to deliver data to multiple destinations. Sources send data to the multicast IP addresses (a special range of IP addresses reserved for multicast purposes), instead of sending them to each destination. Multicast address represents a group of hosts that are willing to receive all the data sent to this address.

The implementation of multicast requires the specific routers, which determine what groups are active on particular subnet and which hosts belongs to them. Multicast packets are encapsulated inside the standard IP packets and are sent between the multicast routers like normal unicast messages. This process is know as “tunneling” and is presented on Figure 8.

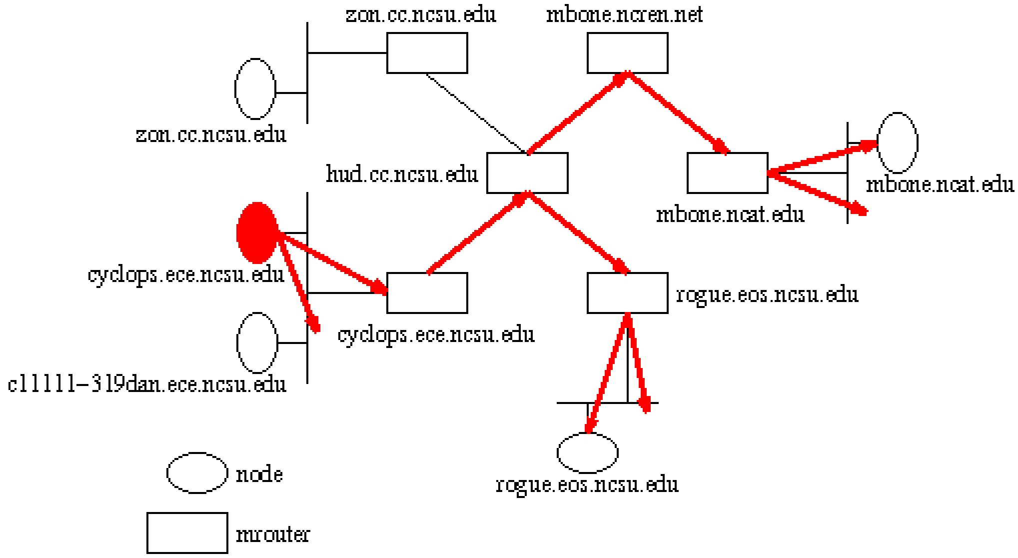


Figure 8 Muticast delivery

Most of the newer Unix operating systems support the multicast. The special software modifying older ones is available free via ftp. Many routers are already equipped with the multicast functionality. Although, MBone is still an experimental technology, it is very likely that soon it will become a popular standard.

3. Videoconferencing products

During the last few years many desktop, videoconferencing products emerged on the market. However, most of them are simple point-to-point videophones. We chose a few of them based on offered quality and standards compliance. It should be emphasized that the situation in videoconferencing changed significantly since our system was designed. This is actually one of the areas that evolve fastest in the Internet environment. Therefore, many features that were innovative during the initial implementation phase of our system, are now elements of many other applications. Yet, we believe that in many fields our system offers capabilities going beyond what is currently available.

3.1 Intel Internet Video Phone

Internet Video Phone is the application created by Intel. It is implemented only for one platform – Windows 95. Intel Video Phone is compliant to the RTP/RTCP protocol described in RFC 1889/90 and the H.323 conferencing standard. Nevertheless, audio and video compression algorithms are proprietary. Audio-video data are distributed only in the unicast mode, since this application offers only point-to-point connections using an approach similar to a standard telephone. It is possible to call someone giving his IP address. A call may be answered or ignored. At any moment the connection can be terminated. Intel Video Phone has a simple and friendly user interface. However, its bandwidth equals 30-40 kbps for the QCIF video mode what makes it difficult to use with modem connections. Additionally, lack of the multipoint option and only the single platform implementation substantially limit its usage.

3.2 IBM BambaPhone

BambaPhone is part of the IBM Bamba suite of research technology demonstrations, which includes Web browser plug-ins for audio and video streaming. It is very similar to the Intel application. Implemented exclusively for Windows 95/NT is also only a point-to-point unicast system. BambaPhone includes modular components for call setup, multimedia control, and video and audio support. The application itself was written with high-level API of those components.

The telephone paradigm is moved even further – BambaPhone is equipped in dialing telephone-like keyboard. It is possible to call someone dialing his IP address number. It makes its user interface to look very impressive but is not very convenient.

The IBM application also uses the RTP protocol for audio-video data exchange. IBM claims only partial compliance to the H.323 standard, what results in lack of interoperability with other videophones. Compression algorithms are proprietary and do not comply with any existing standards. The advantage over the Intel product is the ability to choose a connection type. Based on that BambaPhone automatically negotiates the compression technique, what definitely makes it more flexible. However, its main disadvantages are single platform implementation and only point-to-point mode.

3.3 CU-SeeMe

CU-SeeMe is definitely the most advanced product available on the market. It is created by Cornell Research Foundation. Unlike presented thus far, it is a multi-user system. Moreover, it offers several features that facilitate conference establishment and management such as phone book and directory service. The system is fully based on the H.323 standard. It also supports the

H.263 video compression. It has an additional whiteboard application supporting T.120 and a text chat. It provides functionality to fully control the transmission bandwidth from the 28.8 modem to LAN connection. The audio offers some kind of echo reduction system.

The CU-SeeMe multi-user architecture is based on the central control point called reflector. Reflectors are implemented by White Pine Corporation. Each conference participant sends their data streams to the reflector, which distributes them to all other session members. The latest version provides both broadcast and multicast packets distribution modes. Reflectors also manage conferences, and control their access, having even capabilities for billing and tracking system users.

Unfortunately, CU-SeeMe is available only for the Windows 95/NT 4.0 platform. The reflector approach to multi-user conferences, although simplifying system operation, creates an intermediate point that may affect transmission delays and network jitters. The product does not provide any audio standards making its interoperability impossible to achieve.

3.4 Vic/Vat

Videoconferencing Tool (VIC) and Visual Audio Tool (VAT) are developed by Lawrence Berkeley Laboratories. These are separate applications that may run independently. However, Berkeley Laboratories created also the Session Directory (SD) that coordinates the operation of those two, providing also additional functionality such as advertising of on-going conferences. The system was implemented on the top of the MBone architecture, taking advantage of the multicasting. Hence, this environment structure is fully open, not imposing any kind of restrictions on conferences participation. VIC supports several standard compression algorithms: MPEG, JPEG, NV, CellB and its proprietary intraH.261. Both VIC and VAT are based on the

RTP standard, allowing the loosely coupled session scheme.

Big advantage of the Berkeley package is a variety of platforms on which system is available: DEC, FreeBSD, HP, Linux, SGI, SunOS, Solaris. Nevertheless, all of them are Unix environments. Thus far, VIC/VAT does not support the most popular one – Windows 95/NT (there is only a beta version). The system provides a variety of compression and bandwidth modes, what makes it very flexible. The separate components architecture is not very convenient neither to install, nor to use. The SD functionality provides only open session model, which is not appropriate for many applications, lacking security and access protection. VIC/VAT is an example of a non-commercial environment, offering state-of-the-art technologies but neglecting user convenience and necessary robustness.

3.5 Summary

The products presented above present a variety of different features and miscellaneous functionality. However, each of them possesses several disadvantages and constraints, making their use difficult or impossible in certain situations. Most of them are only videophones, without any support for multi-user conferences. Usually, they cover only one operating system – Windows 95/NT, what significantly hinder their general and flexible usage. Majority of audio and video compression algorithms is proprietary, affecting the interoperability of the available products. Many architectures are not optimal in terms of system performance, extendibility and flexibility.

Therefore, next section is to propose a system free of presented above drawbacks with completely new functionality, which may be very useful for a videoconferencing application.

4. Design and implementation

4.1 Introduction

In the recent years many videoconferencing systems emerged on the software market. They offer different audio and video quality and require different hardware capabilities. However, none of them is moving toward enriching its functionality beyond the audio-video data exchange and integration into multiplatform, collaborative tool, enabling variety of applications in different domains.

Therefore, we undertook some activities in order to create such a system in Internet and Web environment. The conclusions presented in this thesis are results of specific approach based on idea of creating independently videoconferencing, real-time system and Web-oriented collaborative environment, which were integrated afterwards. In real-time applications we paid particular attention to archiving capabilities including random access to stored conference data.

All the solutions were implemented and tested in NPAC both on SGI Indy and PC Windows 95/NT platforms. Our product was called BuenaVista. It was released on the WWW as a public domain product. BuenaVista was used during the virtual course CSC 499 “Programming for the Web” held from Syracuse University to Jackson State University in the fall semester of 1997.

4.2 System objectives

Analyzing the existing conference products in Chapter 3, we enumerated several disadvantages they own. In addition, we found several videoconferencing systems features not implemented thus far, though making them much more convenient to use. The most important of them is the ability to archive the content of audiovisual meetings. Based on that knowledge we can define the objectives of our system. It should possess following features:

- **flexibility** – ability to adapt to different circumstances, in particular network options; BuenaVista should operate both in the Internet environment and on high bandwidth Intranets
- **interoperability** – this feature covers actually two aspects of system operation: *platform interoperability* – capacity to operate on different platforms and *implementation interoperability* – capability to collaborate with other existing videoconferencing products
- **extensibility** – ability to add new components and to modify the system, meeting future needs of videoconferencing products
- **archiving capability** – capacity of store and retrieve the content of audiovisual sessions

To achieve the goals above we decided to adopt solutions described briefly below:

- several compression options both for the video and the audio tool, characterized with different compression ratios, CPU performance requirements and offered quality (e.g. H.263, YUV9, GSM, PCM)
- the full control over the compression algorithms and the offered bandwidth through a simple connection mode settings
- an implementation of the system on two currently the most popular multimedia platforms: SGI workstations and PCs (Intel based Personal Computers); in addition, this choice provides ability to port the system easily on other similar platforms such as the Sun

workstations or Macs

- the compliance with the existing standards (e.g. RTP, H.261, H.263, ADPCM)
- a modular structure enabling a component modification or addition of a new one without necessity of any changes in other system parts
- a separate archiving subsystem, easy to integrate but optional, allowing to store and retrieve conferences content

4.3 System architecture

Designing the system architecture our primary concern was its modularity. Modular system structure enables quick and easy modifications and improvements of some of its parts without affecting the others. Based exclusively on the core modules, it is possible to add new collaborative applications, utilizing existing system mechanisms. This property is especially important in the Internet environment, where the system context evolves dynamically.

BuenaVista modularity covers both the physical system components and the logical layered structure. Physically, BuenaVista consists of two main modules. The first one is the Conference Engine (CE) which contacts the remote hosts, processes control messages and passes indications of events to the Conference Manager (CM). Conference Engine also processes local requests from the Conference Manager. To create and control the Conference Manager, the environment provides a set of API functions. They facilitate and organize the process of implementing arbitrary conference management components.

Other system components supported by an API are applications. Applications are the components defining only what data are to be transported and how to present them to a system user. All other aspects of collaboration are responsibilities of the CE and CM. Hence, it is possible to add a new application without any concern about the conference control and multipoint communication.

An optional, element of the system is a directory service, which provides advertisement of users and conferences functionality and facilitates sessions management. Figure 9 presents the physical system components and their interactions.

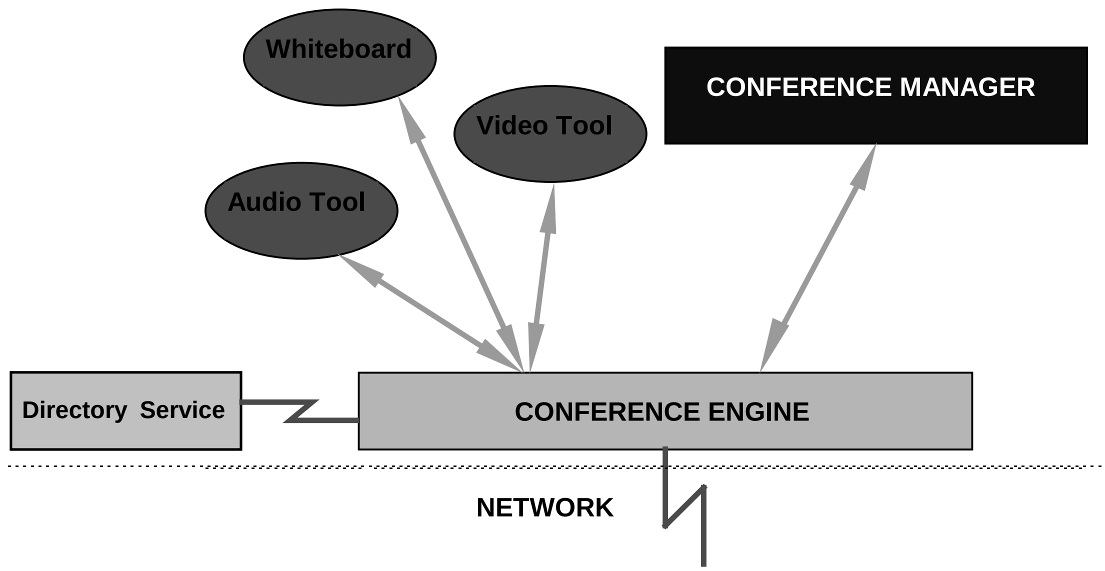


Figure 9 System components

Apart from splitting the system into the physical components, it is also divided into the logical layers, responsible for providing different functionality. The logical structure of BuenaVista can be viewed on Figure 10. The logical layers are closely related with the physical components. Usually, a physical component implements one logical layer. However, one-to-many relation between a component and layers is also possible. The following sections describe the logical layers with their relations to components.

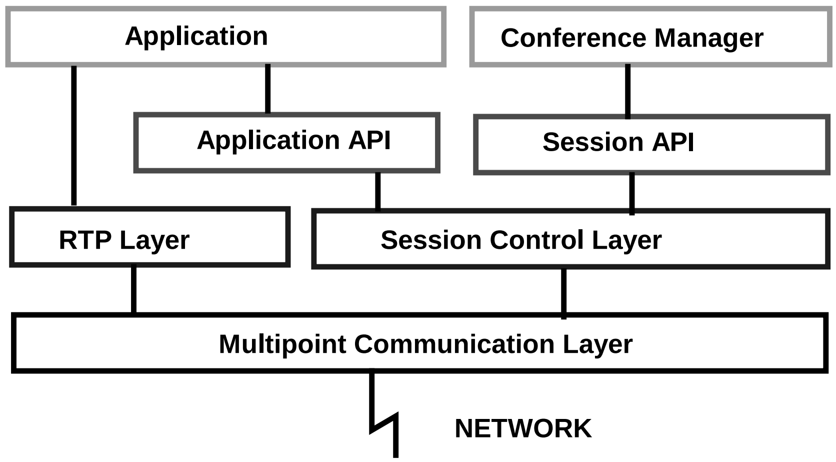


Figure 10 Logical system structure

4.3.1 Multipoint Communication Layer

Multipoint Communication Layer (MCL) provides the means for transporting conference data to all the session participants. MCL uses point-to-point transmission methods specific for the network type and combines them into multipoint channels. For MCL, content of transported data is fully transparent. Since both Session Control Layer and applications use multipoint channels MCL is an element of both Conference Engine and application API.

BuenaVista MCL is able to arrange two types of multipoint channels: reliable and non-reliable. Reliable multipoint channels are based on TCP (Transmission Control Protocol) and are used primarily for conference control purposes. All the requests from Session Control Layer (SCL) are served with reliable channels. Control messages are not very sensitive to network delays but are extremely prone to data losses or transmission errors. Therefore, reliable channels are the only reasonable way to distribute control messages.

Unlike control messages, most of data transmitted by real-time audiovisual applications are severely dependent on network delays, having some tolerance for data losses or errors. Thus, the best suitable protocols for those purposes are non-reliable protocols. They do not posses time-consuming packets resending procedures, offering better network performance. In particular, the BuenaVista MCL implements non-reliable channels based on the UDP (User Datagram Protocol).

However, not all application data have the characteristics described above. Data oriented applications, such as whiteboard or text chat, transport information in a way more similar to control messages. Hence, they could take advantage of the reliable channels. Moreover, even real-time applications usually have some kind of a control protocol (e.g. for capabilities exchange), that work more efficient with reliable communication. Therefore, the applications have a choice of a channel option, being even capable of using both channel types at the same time.

4.3.2 Session Control Layer

Session Control Layer (SCL) is the central element of the conferencing system. SCL is physically implemented inside the Conference Engine. Both Conference Manager and system applications are connected to the CE. SCL is responsible for events distribution and handling both with local entities (CM and applications) and remote hosts. Hence, it is the only system component that has full knowledge about conference status and is capable to control it.

Functionality of the SCL can be divided into two groups: events distribution and handling and conference information management. Using services provided by MCL, Session Control Layer distributes control messages to other conference participants. This process does not include conference data, which are responsibility of conference applications and utilize different distribution mechanisms. SCL also handles messages received from other BuenaVista nodes. The control communication scheme is organized into the Session Control Protocol (SCL) described in section 4.4. Moreover, SCL interacts with the local entities: Conference Manger and the applications. The protocol used for this communication is briefly described in API section 4.3.4. Based on received requests SCL makes the decision about the event distribution and its destinations. For example receiving notification about a new user joining the session SCL send the indication both to the Conference Manager and the applications.

However, SCL is not only the events distribution module. It manages all the information needed throughout the session. The Conference Engine keeps information about the session participants and applications. Based on that, it controls the session growth and its modifications. Conference Manager and applications can query SCL about this information. Many decisions made by SCL are based on current conference status.

4.3.3 RTP Layer

Multipoint, non-reliable channels do not satisfy all the requirements for transporting real-time, multimedia data. Audiovisual applications should be capable of determining a variety of additional informations to effectively process received media streams. For this purpose we implemented the Real-Time Protocol (RTP) Layer.

The RTP protocol is an existing standard for transporting real-time media streams. It is briefly described in section 2.2.1. For more information refer to [14]. RTP is an application level protocol, which needs additional profile specification.

The RTP packet fields convey useful information, enabling extensive analysis of the network and the connection status and initialize appropriate correction activities if necessary. A brief description of the RTP header fields, with the methods they are used in audio-video applications, is presented below:

- **time-stamp** – allows to determine the network delay and jitter and set appropriate bandwidth constraints

- **sequence number** – enables the lost or misplaced packet detection and initialize the recovery procedures

- **source ID** – determines the source of the packet, allowing to demultiplex media streams from different participants (e.g. displaying video streams from different users on separate windows)

- **message type** – describes the content of the packet, enabling its appropriate processing (e.g. choosing the decompression option)

In addition RTP packets are used to maintain statistics about the communication process, that can be viewed and analyzed by a user, offering measurable estimation of the session quality and difficulties.

4.3.4 Application Programmer’s Interfaces (APIs)

BuenaVista APIs were created to facilitate development of system-based applications. Conference Engine and APIs provide the backbone for the conferencing system. They allow flexible improvement of BuenaVista, creating a variety of new applications and enriching of their functionality. According to the type of an application that is implemented, there are two different kinds of API functions:

- **session API -** for development of the conference management applications (e.g. Conference Manager)
- **application API -** for development of the ordinary conference applications (e.g. Video Tool, Whiteboard)

API functions perform all the message exchange operations and offer easy and straightforward callbacks mechanism to proliferate system events. They also enable acquisition of all kinds of conference information that may be useful for an application.

4.3.4.1 Session API

Session API provides convenient functions for conference creation and management. However, the session API functions do not perform those operations directly. Actually, they only send appropriate requests to the Conference Engine, which is the main element responsible for all the session control. Session API is rather an implementation of the internal protocol between Session Control Layer and Conference Manager. Nevertheless, handling SCL indications, session API also control several data structures, which may be useful for the CM.

The session API functions can be divided into four classes: initialization and cleanup, events processing, requests generation and information acquisition. The first step of the Conference Manager is the conference initialization. This operation is performed via `ncsSessionInitialize`() function, which returns an error value upon the initialization failure. The initialization opens the control socket, prepares all necessary data structures and informs the Conference Engine that a managing application is active. Similarly the manager should perform all necessary cleanup invoking the `ncsSessionQuit`() function.

The events processing functions rely on the system callback mechanism. Callbacks are a simple and convenient way to inform applications of conference events. A developer writes the procedures handling events and adds them as callbacks to the conference system. Upon the event reception, the appropriate callback is invoked automatically without any additional developer's effort. A conference callback is set with `ncsSessionAddCallback`() function.

To properly react to system events, the application developer must implement a checking loop. The checking loop is a simple sequence of code that constantly and regularly checks the connections with SCL for new messages and initializes their processing. Depending on the developing platform, it may be an X-Windows working procedure, a Windows thread, or an ordinary loop. Inside the loop one can obtain the application file descriptors set, which may be then used as a parameter for the standard socket `select`() command. If at least one of the file descriptors is active, a function processing incoming messages (`ncsProcessIncoming`()) should be invoked.

The request generation functions are probably the biggest group in the session API. They are responsible for converting local user actions into requests to the Conference Engine. We will present only a small subset of those functions, which provide the most basic functionality. User may start a new session invoking the `ncsSessionCreate`() function. After that, the SCL is informed that the user is a member of an existing conference and automatically rejects any invitation received. The next step in conference set-up is inviting a new participant. This operation is performed via the `ncsSessionAddMember`() function. Receiving the invitation the user may accept it with `ncsSessionAccept`() or reject it with `ncsSessionReject`().

The last group of session API is the information acquisition functions. Conference Manager may query the system about conference participants or active applications. Some of this information are stored by the session API and do not require sending requests the CE. Other must contact CE and receive the appropriate indication.

The session API provides all the necessary functionality for creating Conference Manager. Therefore, it is possible to create different types of managers fitting best the user preferences. This is one of the methods providing BuenaVista flexibility.

4.3.4.2 Application API

The application API is probably the most important element in terms of system extensibility. It provides all the necessary means to create a new conference application without any concern of session control or data distribution mechanisms. The application developer is responsible only for the choice of data transported and their view for the application user.

In its structure application API is similar to session API. It can be divided for the same function classes. To collaborate within the current session, each application must attach to the Conference Engine with the `ncsAppAttach`() command, which returns the application handle used by all other conference operations. Application detaches from Conference Engine using a call to `ncsAppDettach`().

Application API takes advantage of the same callback mechanism. It also handles the incoming messages in a similar way. Additional elements of the application API are functions connected with data distribution. In the check loop function `ncsAppGetData`() must be invoked to retrieve incoming messages from other participants. Conference data can be distributed to all the session participants or the selected ones via the `ncsAppSendData()` function. A variety of conference informations, in particular about session participants, can be retrieved with the API functions.

4.4 Session control protocol

The BuenaVista Session Control Protocol (SCP) is the control protocol for tightly coupled conferences. The design of this protocol was, in part, guided by the ITU-T recommendation T.120 (section 2.2.3) and H.323 (section 2.2.2). SCP was created to provide the following functionality:

- conference participants management
- conference applications management
- maintaining additional conference information

All those elements must be controlled in such a way, that a consistency of the conference state is ensured. SCP utilizes messages passing mechanism to maintain the conference status. Every received message is checked, if it meets the consistency requirements. SCP is based on the TCP protocol to allow maximum reliability.

During the normal operating mode, from the end-user point of view, all the session nodes are treated in the same way. However, SCP does distinguish one special type of node – the statekeeper. The statekeeper is a session node that is responsible for resolving any conflict connected with concurrent operations. Usually, the conference initiator is the statekeeper. In some situations it is possible to transfer the statekeeper status to another session member (e.g. when the initiator leaves).

4.4.1 Participants management

SCP conference creation paradigm is based on the invitation model. A user who wants to start the conference with particular people sends them invitations, which they may accept or reject. After a conference creation each session participant is allowed to invite new users. This model ensures the tight conference structure – no one from outside is allowed to join a conference without a direct request. This pattern is accomplished with following message types:

- INVITATION

  This message informs a user that he is invited to participate in a conference. In addition, the invitation message conveys information about the caller ID to allow the user to accept only the invitations from certain sources.

- ACCEPT

  An accept message confirms the user will to participate in a conference. It includes accepting party ID, allowing the identification of a joining participant in a possible set of invited users.

- REJECT

  Informs the inviting party, that a user is not interested in joining the conference. As a result the conference status will not change and other participants will not be notified about the invitation.

- STATUS

  A joining user needs to initialize the conference status information. A status message enables this update, carrying all the necessary status data, in particular conference participants. This message is sent upon receiving the invitation acceptance. A reception of this message is also a switching point from the “free” (willing to join a conference) to the “busy” (automatically rejecting any invitations) state.

- NEW_USER

  Thus far the whole joining process took place only between two participants: inviting and accepting. Now, all other session participants need to be informed about a new user. This operation is performed via NEW_USER message, sent by a new participant after the conference status initialization.

- LEAVE

  Informs all conference participants that the sending user is leaving the conference. Upon reception of this message all participants modify their session status by deleting the leaving user. Any conference participant is allowed to quit at any time. Therefore, all the participants should be ready to process this message immediately.

Conference participants are identified within the conference by unique and simple participants IDs. A participant ID is assigned dynamically upon joining the session, remains constant only for the time of participation in the conference, and is independent of the participant network address. SCP IDs are used for the primary participant identification by all the conference model logical layers.

During the normal system operation conference initiator is the statekeeper. However, when a statekeeper leaves, conference system should transfer his functions to another session node. This operation is based on the SCP IDs. Initially, the conference initiator (and statekeeper) has ID equal to zero. When his functions are being transferred, the statekeeper status is assigned to the conference participant with the lowest ID. This mechanism provides the simple and robust method for maintaining consistent the statekeeper status.

4.4.2 Application management

Because of the system modular structure, SCP has to manage the conference applications as well. The goal is to obtain the same subset of conference applications, running on all participants' machines. This functionality could be simplified, keeping all the application running all the time. However, often participants want to use only some applications (e.g. only audio) and running all of them would be a waste of resources. In addition, it is possible that a participant does not wish to receive or send a particular type of data (e.g. his CPU is not able to process the video stream). Thus, application management should leave participants a level of flexibility with choosing their applications.

In BuenaVista this issue was solved by forcing participants to start all the application in the session, leaving them freedom of closing them without any effect on the conference. However, SCP was designed to leave the solution of this problem to particular implementation of the higher logical layer (Conference Manager). SCP provides exclusively means of indicating such events to SCL layer.

The BuenaVista applications are identified inside the system by their unique IDs. In particular these are the application BSD socket port numbers. However, for the SCP, the strategy of assigning numbers is immaterial as long as they are unique. The application management is performed with two message types:

- START_APP

  Includes the application ID. Informs that sender of this message started the application.

- CLOSE_APP

  Similarly to previous message it includes the application ID. Informs that sender of this closed the application

Both those messages are distributed by the SCL layer to all session participants.

SCP provides also convenient functionality of transporting the application data through the control channel. These services may be useful for applications based primarily on non-reliable multipoint channels, sending infrequently important, need-to-know, short information. An example of such an application is the Video Tool, which sends all the multimedia, video data through the non-reliable channels but sporadically informs the participants about the beginning or the end of a transmission, what allows them to prepare or cleanup appropriate display devices. The message type performing those operations is:

- SEND_APP_DATA

  Includes application ID and conveyed data. Receiving this message, SCL may distribute this event to appropriate application, which can handle it accordingly. For SCP, a content of this message is fully transparent. All the data inside are simply passed to the higher level layer.

4.4.3 Maintaining additional conference information

SCP is equipped with the additional functionality, which may useful in some situations but is not part of the main conference control process. Thus far, it allows contacting and obtaining information from a simple user directory, keeping data about all the active system users. To accomplish this goal SCP includes the following message types:

- SRV_REGISTER

  Informs the server (user directory) that a user started the videoconferencing system and may be reached through the invitation mechanism. Includes user IP address and operating system user name. Allows the user directory to update its structures adding the user to the active list.

- SRV_UNREGISTER

  Informs the server that the user is quitting his videoconferencing application and will not be available anymore. Includes user IP address and operating system user name. Allows the user directory to update its structures removing user from the active list.

- SRV_USERS_REQ

  Requests list of the active users from the server.

- SRV_USERS_IND

  The indication from the server including the list of active users. Upon reception, this message should be distributed to higher control layers, allowing displaying the information to the end-user.

In addition, we intend to include the token control functionality into the SCP. Tokens manage concurrent access to the limited resources. They allow a user to have the special status within the session. An example of the application of tokens may be the conductorship assignment (teacher-student mode), which may be particularly useful in education.

4.4.4 Example scenario

This section presents an example scenario, showing how SCP is applied to the conference creation and management. This is only one possibility, aiming to illustrate the process of SCP operation, not to document its functionality.

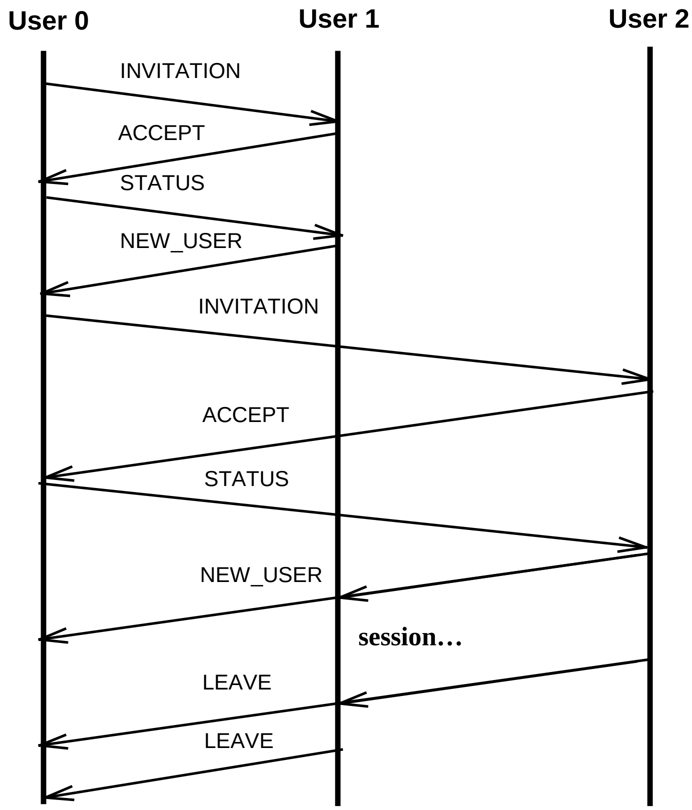


Figure 11 Example session control scenario

Figure 11 presents the simplest possible scenario of a conference creation with SCP. The User 0 which is the conference initiator invites first the User 1, and after successful session join, the User 2. It should be noted that the User 1 could also invite the User 2 without any consequences for the future conference course. All the INVITATION messages are acknowledged with the ACCEPT packet. Passing the STATUS indication is confirmed with the NEW_USER message sent to all the participants. For simplicity Figure 11 presents only the elements connected with the participants management. Users signal leaving the conference with the LEAVE message.

4.5 Applications

4.5.1 Audio Tool

The Audio Tool is probably the most important application of the videoconferencing system. It enables the real-time voice communication between the conference participants. Mainly, the application was designed to transport the speech signals, however transmission of low quality music is also possible. To achieve this goal, the application utilizes 8 kHz sampled, 16 bits/sample, mono audio signal as a base for compression and distribution. In addition, following compression algorithms are adopted:

- **GSM** – low bandwidth (13 kbps), low quality codec with the highest CPU performance requirements. It offers capabilities to compress exclusively speech signals.

- **Intel/DVI ADPCM** – medium bandwidth (32 kbps), good quality codec with relatively low CPU performance requirements. Designed for general audio signals, it is capable to compress low bandwidth music.

- **PCM** – raw 8 KHz, 16 bits/sample, mono signal; high bandwidth (128 kbps), good quality, general purpose

4.5.1.1 Application structure

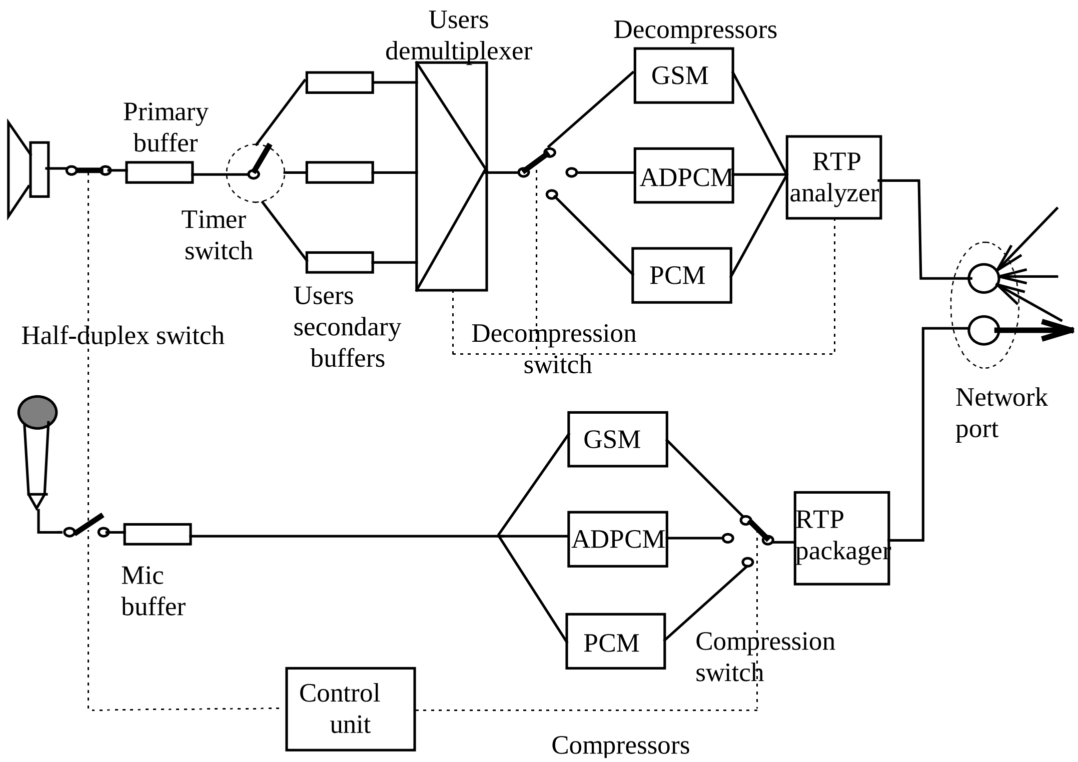


Figure 12 Audio application structure

The audio application structure can be divided into two main subgroups: sending path and receiving path. The sending path is responsible for capturing an audio signal from microphone, compressing it and distributing to other participants. The receiving path handles the incoming audio packets, eventually directing them to the speaker.

Sending path

First, the sending path subsystem captures audio samples, controlling the process by the half-duplex switch described in section 4.5.1.1.3. Then, captured audio samples are stored in the microphone buffer. Actually, to provide continuous audio samples supply, the double-buffering scheme is used – when first buffer is processed, capturing operates on the other one. Afterwards, the audio signal is directed to the appropriate compressor. The compressor selection depends on the application settings, which in this case are determined through the connection user menu. Employment of the application settings is handled through the control unit. Besides the compression option, control unit is responsible for the half-duplex switching. Finally, the compressed audio chunks are packed into the RTP structures and sent to the network port.

Receiving path

Receiving path has significantly higher complexity due to necessity of handling multiple audio streams. Its structure highly depends on the hardware platform and the available audio libraries. On SGI Indy the structure is definitely simpler. There are several reasons explaining this property. The SGI Indy workstation equipment includes standard audio card and drivers. Thus, audio libraries are created specifically for this hardware type, providing several mechanisms facilitating audio applications development. The most important of them is the internal mechanism of multiple audio streams buffering and multiplexing. Another convenient feature of the SGI audio device is the full-duplex mode, which totally removes necessity of half-duplex switching.

A PC audio application must take into consideration several factors connected with variety of available hardware audio devices. First, it has to query available devices about the supported formats. Then, it has to decide which of the devices is most suitable for the videoconferencing purposes. Since multiplexing and buffering mechanisms are not accessible, they have to be implemented inside the application. PC Audio Tool must also adapt to the half-duplex mode, which requires additional switching mechanism.

Structure on Figure 12 presents the most general option. First, RTP audio packets from different sources are analyzed in order to retrieve the necessary information about their content. In particular, data about packet source and content type are obtained. The content type is used to determine, which decompression option should be used.

After decompressing, the decoded stream consists of packets from different sources. It has to be demultiplexed using the RTP packet source information and directed to the appropriate secondary buffer. The secondary buffers are locations where data from the particular user are stored. Then, decoded and sorted packets must be multiplexed again to create the final audio stream sent to the speaker. To properly multiplex incoming data and to make playing operation independent of the receiving process, the Audio Tool utilizes the two-level buffering scheme. The application puts the data into secondary buffers according to their rate of coming from the network. The main playing loop adds the content of all the active secondary buffers to the primary buffer and send the primary buffer data to the speaker with a constant rate dependent on audio stream properties. Final access to the speaker is controlled by the half-duplex switch.

Half-duplex switching

A convenient property of a conferencing audio device is the full-duplex mode i.e. ability to capture and play audio streams in parallel. Unfortunately, many audio cards do not support this option, allowing either capturing or playing audio streams, exclusively. This operation mode is called half-duplex.

The half-duplex mechanism in a videoconferencing system requires switching the audio device between capturing and playing. Since during normal conversation, it is unusual to speak in parallel with another participant, the switching tool may provide functionality to simulate the full-duplex mode, detecting silence periods and switching accordingly.

The simplest possible solution is to leave the switching to the user. The application GUI is equipped with the “push-to-talk” button, that changes the active device either to the microphone or to the speaker. Unfortunately, this solution is inconvenient for the user, requiring constant attention for efficient conversation.

Therefore, another option is the automatic silence detection and switching. However, there are several problems connected with the efficient automatic half-duplex system. First of all, it is difficult to determine how long the silence periods should be before changing the mode. In addition, sometimes participants talk at the same time. Hence, usage of the half-duplex mode requires a discipline in adapting the conversation to the applied solution. The BuenaVista AudioTool offers the choice between both the options to ensure maximum flexibility.

4.5.2 Video Tool

Video communication enhances the quality of a conference. It introduces a sense of presence and allows conveying the information through gesturing. Unfortunately, process of effective video transportation in network environment causes several serious problems. First of all, two-dimensional video data bandwidth is usually significantly higher than of an audio signal. Therefore, efficient compression algorithms are extremely important for video applications.

In order to provide flexible solution, offering different bandwidth options we adopt the following video compression algorithms:

- H.263 – state-of-the-art video compression standard, offering very low bandwidth but extremely CPU intensive

- H.261 – the popular H.263 predecessor, less bandwidth efficient and less CPU intensive, Video Tool utilizes also modified H.261 Intra mode. Sending exclusively I-frames, it offers better performance, yet introducing significant bandwidth increase

- YUV9 – Intel video format, essentially without any compression, extremely bandwidth inefficient but requiring very little CPU power

Implementing Video Tool we took advantage of H.261/H.263 codecs available on WWW public domain for non-commercial use. However, in their existing forms those solutions were not suitable for videoconferencing purposes. Hence, we undertook activities, described in following section, to overcome those obstacles.

4.5.2.1 Adapting H.261/H.263 codecs for the videoconferencing purposes

The source code for the Video Tool compression algorithms was obtained from the Stanford University H.261 and Telenor's H.263 video codecs. Both of them were fully compliant with the existing standards and very similar in their structure. Thus, below we present only the modifications introduced to Stanford H.261 codec, since we utilized similar procedure to adapt the other one.

The Stanford University source code was organized into the file-to-file streaming codec, encoding whole video sequences in one step. Applying this solution directly to the conferencing tool would require buffering strategies, introducing a substantial delay to images transmission process. Hence, the following modifications were interposed:

- converting the existing program into a library, allowing to invoke separate procedures rather than coding the whole video sequence

- retrieving the initialization procedures and the single frame encoding/decoding functions

- converting the I/O functions – thus far they communicated only with the files, they were converted to pass the data in argument and return data as a memory pointer

- multiplexing all the reference frame information for multi-stream decoding (streams from different participants)
- inserting regularly the INTRA frame into the encoded sequence – necessary to prevent application crash if one frame is lost

Moreover, the performance offered by the Stanford implementation was unacceptable for the real-time, conferencing purposes. Improving the situation required extensive source code analysis and testing to determine the weakest points and optimize them.

The H.261 algorithm is asymmetric – the image encoding is much more time consuming than the decoding. Therefore, we focused on optimizing the compression algorithm. Testing confirmed the theoretical hypothesis that the most crucial encoder component in terms of performance was the motion estimation procedure. It was estimated that this procedure took about 70 % of compression time. Stanford codec adopted full spiral motion vector search algorithm, offering good quality but very inefficient. Since quality issues are not very important for slowly changing videoconference sequences, it was decided to replace the spiral search algorithm with faster logarithmic search. Those improvements reduced the time of the motion vector estimation to 30 % of the whole compression process.

4.5.2.2 Application structure

The video application structure is very similar to the Audio Tool structure presented on Figure 12. The hardware devices in this case are a camera and a monitor. In addition, there is no multiplexing unit, since separate video streams may be displayed on different application windows. Both operations: video capturing and displaying are handled through separate hardware cards, thus they may be performed in parallel eliminating any switching between those two. Certainly, different types of data codecs are used. The rest of the application is exactly the same: receive path with RTP analysis, decompression and streams demultiplexing and sending path with capturing, compression and the RTP packager. Actually, those units are the components of a generic application model, common for all the real-time system applications.

4.5.2.3 Other issues in designing and implementing video application

Frame rate control

We already mentioned that the encoding of video images is a very CPU intensive process. Therefore, a rate of compressing video is limited not only by the network bandwidth, but also by the CPU performance. The application requires the effective CPU utilization, allowing other application to work (in particular Audio Tool). This constraint imposes the necessity of implementing a frame rate control mechanism, which analyses the processor usage and sets the rate limit. Once again, the solution of this problem is hardware dependent. Hence, the implementations are different on SGI Indy and PC.

SGI provides the developer with the internal capturing buffer. It captures the images with a preset rate, but allows getting the pointer to the current frame at any time. The control mechanism measures the time needed to decode the frames for all other users and the rates with which they are sent. Based on this information the interval required for the decoding process is calculated. After capturing and encoding each frame the process waits the calculated interval and continues to obtain the next frame. This method allows the adaptive adjustment to the circumstances and leaves the computer and the conferencing system operation undisturbed.

Unfortunately, adaptive control is not possible on PC. The capturing process invokes the callback procedure, for every new frame. The rate adjusting requires time consuming reset of the whole capturing module. Thus, different approach was employed. Initially, the rate is set at the relatively low level, which can be maintained by majority of modern PCs (with at least the Pentium processor). During the several first images capturing, the application analyses the compressing and decompressing times and calculates the maximum rate that the machine is able to support. Finally, the value is set and from that moment capturing operation is uncontrolled. There are several events that may force to recalculate the value (e.g. new participant starts to send the video), but during relative long periods of time this limit is constant. This leaves several possibilities, when system will not work optimally (e.g. a new time consuming application is started). However, they are very difficult to control.

Video formats conversions

Modern video capturing cards come in a variety of the offered video formats. But even the most popular format – RGB24 is not supported by all of the cards. Additionally, the available image displaying libraries have a limited choice of possible formats. Therefore, the video application requires making several video conversions to meet the constraints of capturing devices, displaying procedures and compression algorithms. It deals primarily with two major types: RGB and YUV, which are often implemented in several variants. This problem is particularly important on PCs, which may be equipped with several different video cards. The Video Tool completed conversions among the most popular formats: YUV22, YUV11, YUV9, RGB24 and RGB16. This set was sufficient for most of the video cards tested during the development stage.

4.5.3 Other collaborative applications

4.5.3.1 Text chat

Text chat is a simple application enabling participants to distribute short text messages. Its functionality seems to be redundant, having superior audiovisual communication. However, under some circumstances text chat is very useful. First of all, it is possible to transport complicated char strings such as addresses, names etc. without complicated and error prone spelling process.

Moreover, audio and video applications are very sensitive to several configuration and environment problems. Changing an audio card or a driver sometimes causes serious problems with a device stable operation. Even the most advanced networks often experience bandwidth constraints making the audio and video applications unusable. This is particularly severe in the Internet environment. For all those cases users are left with only one solution – a text chat.

Design and implementation of the text chat is so simple, that it will not be described in this thesis. This section is only to mention the importance of chat applications in a videoconferencing system. The text chat for BuenaVista was initially implemented in Java, but since Java applications require an interpreter, what significantly complicates the system installation process, we implemented simple chat in C++.

4.5.3.2 Whiteboard

Whiteboard is a very convenient collaborative tool that allows the sharing of drawings among conference participants. It is possible to perform the following operations:

- drawing lines and basic geometrical shapes
- choosing drawing colors

- choosing line width
- displaying text
- saving and retrieving a drawing from a file.

Implementation

Each drawing element is stored in memory as an “graphic object”. Object consists of following information:

- object type
- object position
- additional location
- object properties (color, line width etc.).

The application creates and manages a list of objects that can also be saved and retrieved from a file. Each object type has its own drawing procedure. Objects are exchanged between applications using the BuenaVista distribution mechanisms. Objects received are handled the same way as any other user input.

4.6 Directory Service

A directory service is an additional, optional element of the videoconferencing system. One of the assumptions for the system design was its ability of operating independently of any servers or network components except the local clients. However, for the user convenience it is very important to have information about the active BuenaVista nodes and on-going conferences. Therefore, we created directory service.

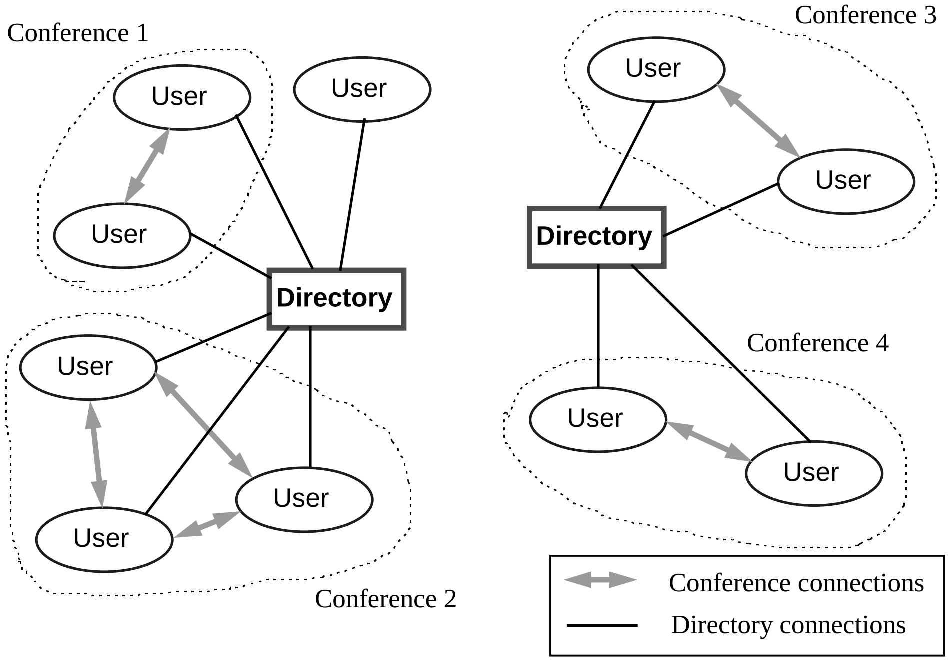


Figure 13 Location of Directory Service in conference structure

Main purposes of the directory service are:

- to keep information about the active nodes (users willing to participate in conferences)
- to keep information about on-going conferences
- to facilitate creating, joining and managing conferences
- to ensure sessions protection and security

This functionality requires storage and management of a variety of data regarding conferences and users. Initially, they will be stored internally by directory service; eventually we intend to create database system for the information control. The structure of data stored in directory service is presented in Tables 4 and 5.

A user that created the conference becomes its ‘master’ – he defines conference properties and may control the conference access. One of the most important conference property is its type. We distinguish three types of conferences:

- *open* – accessible without any restrictions
- *password protected* – access to conference controlled with password
- *master controlled* – access to conference by master acceptation

Normally conferences are created dynamically by BuenaVista users. However, there is also a set of predefined static sessions, which by definition have the open character.

Upon starting, the application automatically connects to the directory service. Its location is the element of system configuration and can be changed whenever convenient. It is predicted to implement also automatic directory service detection. The directory may accept or reject a new user. If the user is accepted the directory registers him and if necessary creates appropriate entity in the database.

Table 4 Data structure for conference information

Field name	Field type	Short description
Name	String	Unique name by which conference will be identified and which will be presented to system users
Type	Implementation dependent	Conference type identifier, described above
Participants	List	List of conference participants

Password	String	Password controlling access to conference (applied only for password protected conferences)
Properties	Implementation dependent	Additional conference properties e.g. audio and video capabilities of conference participants

Registered users can query the service about active users and existing conferences. They can also send requests for a new conference creation, termination, joining existing session or information about an active user. A user may create new conference if he has appropriate rights. He must be accepted to join the open session. He is asked to type the password to join the password-protected session. When he tries to join the master controlled session, a special indication is sent to session master, who can accept or reject it. A user may terminate only conference that he created.

Table 5 Data structure for user information

Field name	Field type	Short description
Name	String	User name which will be presented to other users
Address	IP address	IP address of host from which user is connected
e-mail	String	User e-mail address
Info	Text	Additional info, comment etc. (defined by user)
Rights	Implementation dependent	User rights (conference creation etc.)

4.7 Archiving

Employing a videoconferencing system, very convenient capability would be to store the sequence of events occurred during a session. Then, the conference content could be retrieved, replayed or searched, enriching the collaboration process. BuenaVista includes such archiving functionality, allowing retrieving the session events in the real-time mode.

4.7.1 Synchronous vs. Asynchronous archiving

Normally, multimedia data are stored in a synchronous way. Data stream has the constant set of properties, which enables efficient interpretation of retrieved data. Throughout the whole storing process data are homogenous – they have the same type and structure. In particular, multimedia data have specifically defined its timing constraints.

Videoconferencing data have completely different profile. The data come from different applications (audio, video, whiteboard etc.), having characteristics that often change. For example video stream has usually the changing frame rate and the compression type (depending on available bandwidth or CPU performance). In addition, we usually deal with data from different users that could be combined together within the current session.

There are two solutions to the problem above. First one is to store data of different formats, from different users in separate streams and then convert each stream to the homogenous format. Another solution is based on the generic storing and retrieving mechanism handling different types of data with different properties (including different timing) simple neglecting their content and encapsulating them in additional containers.

The first solution provides data streams in the standard formats, allowing their retrieval with the commonly available applications. Therefore, it offers the standard content that can be easily transformed or modified. However, it introduces high complicity connected with data converting and managing. All the formats must be standardized; information about each session must be created and controlled. It creates huge overhead on the application development.

The second solution stores data in a proprietary format, that can be retrieved, viewed or modified only by the applications created specially for this purpose. In that sense it is much less flexible. Nevertheless, a big advantage of this implementation is its simplicity. This type of archiving system is application independent. Internal system mechanisms provide the means for storing and retrieving application data seamlessly for conference application developer. Therefore, any new collaborative application supports archiving functionality without any extra effort from its developer. Since archiving capabilities were not our primary concern designing the system, we adopted this solution for BuenaVista.

4.7.2 Architecture

The archiving system structure can be divided into two logical and physical components: recorder and retriever. It is presented on Figure 14. The recorder is the element responsible for the conference content storage, first converting it to the appropriate format. The retriever reads the stored content and performs operations enabling its replaying.

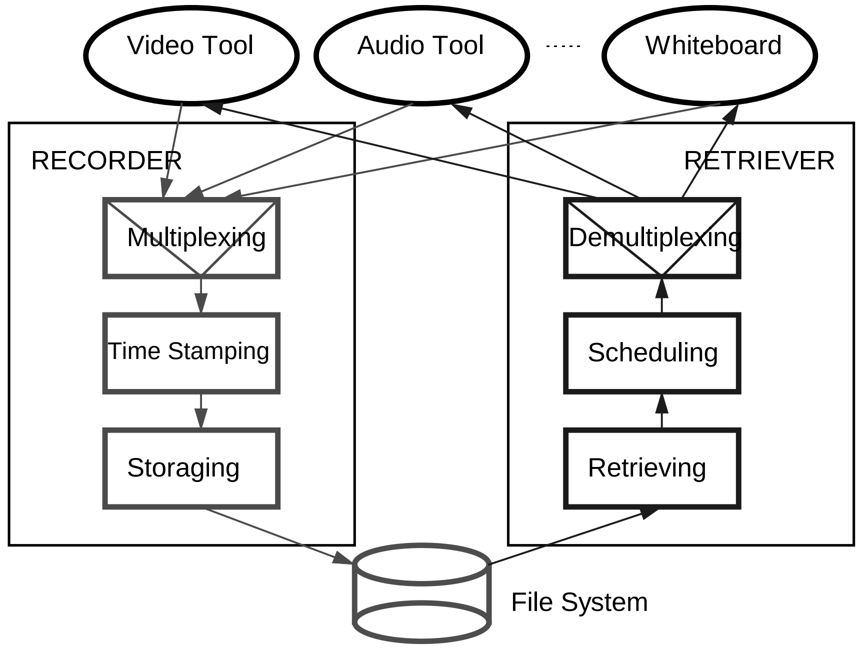


Figure 14 Archiving System Structure

Both of those modules are characterized below:

- **Recorder**—an intermediate point between BuenaVista applications and file storage. Recorder multiplexes messages from different applications adding their unique identification numbers, inserts time stamps and store messages to a file.

- **Retriever**—performs similar functions in the opposite direction. After retrieving a message it schedules its sending time based on the timestamp waits until scheduled moment and send data to the appropriate application using its identifier.

During the normal operation, archiving capabilities are not active. Conference Manager is responsible for starting this functionality upon a user request.

4.7.3 System integration

The archiving system components are optional but tightly integrated parts of the BuenaVista videoconferencing system. To support their functionality several existing system elements must be adapted:

- **BuenaVista application API** — improved of communication with the recorder capabilities and switching on/off database mode upon the BuenaVista engine request (sending messages to the database all the time would be wasteful and would affect application performance)

- **BuenaVista engine**—enriched in receiving and handling Conference Manager archiving requests procedures. Handling those requests includes starting recorder or retriever, requesting archiving mode from applications, transmitting reliable application data to recorder.

- **BuenaVista session API**—enables switching between recording, retrieving or normal working mode.

Location of both modules in the BueanVista structure is presented on Figure 15. Archiving functionality is activated by a request from Conference Manager. Having received the request, Conference Engine starts appropriate components (recorder or retriever) and distributes the indications to conference applications. When the applications are notified, they commence to send all the conference data (both received from remote participants and local) to the recording module. The activated retriever reads the stored data from the disk and distributes them to the appropriate applications, controlling the time constraints.

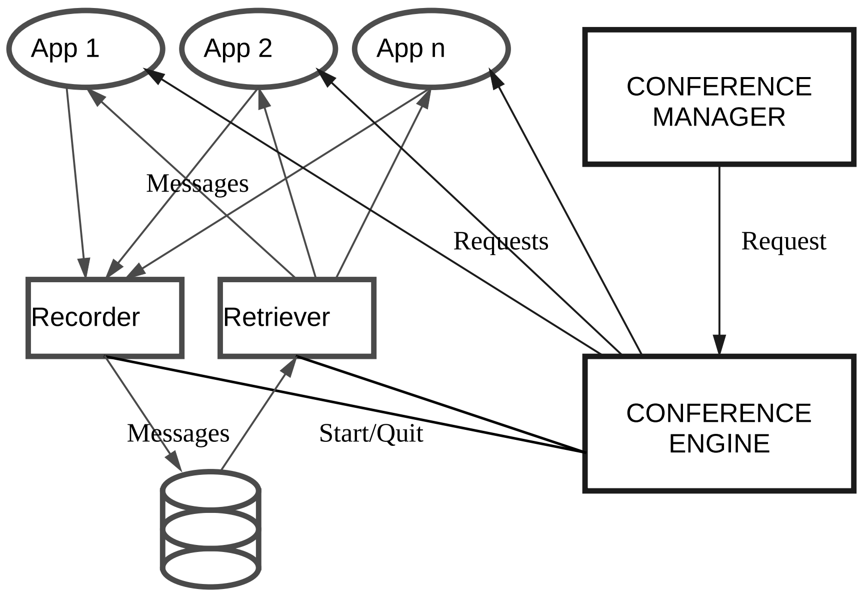


Figure 15 Archiving capabilities management

4.7.4 Database integration

Simple conference content archiving offers very limited functionality to the system users. Stored conferences should be organized and managed, enabling random access to session events, acquiring information about session participants, date, discussed subjects and search of all the stored sessions for a particular information.

Therefore, we intend to integrate the archiving component with a database system. A database can perform all the described above operation, creating a powerful tool for people communication.

4.8 Other issues in system design

4.8.1 Java API

Our conferencing system was written in C++. However, the only constraint for a conference application is to meet the internal protocol requirements. Hence, it can be created in any language supporting TCP/UDP network capabilities. A huge advantage for the application developer is the existing API (described in section 4.3.4), which enables creating applications without getting into complicated protocol issues. Initially such API was implemented only for the C++ and C applications. Nevertheless, to prove that it is possible for any other language and to provide a tool to create such an application in winning enormous popularity in recent years, multiplatform Java programming language – we created API for this environment.

As an example of BuenaVista collaborative application written in Java, we successfully ported the Web based chat application to our conferencing system. This way, we achieved the goal of language and platform independence.

4.8.2 Session security

The BuenaVista session model provides support for the conference security. Tightly coupled, close for the users from outside session model meets the requirements for the secure system. The only remaining issue is the confidentiality of distributed data.

Thus far, confidentiality components were not included in BuenaVista. However, during the design stage those issues were taken into consideration, specifying place of the confidentiality elements within the system. Addressing the confidentiality of conference data requires implementing an encryption layer, placed between RTP layer and MCL. An arbitrary encryption algorithm can be utilized, disregarding content and structure of higher level packets (RTP).

Security issues were also analyzed, designing the Directory Service. The goal of introducing the password protected and master controlled session types was to provide the privacy mechanisms to directory users. This way it is possible to take a full advantage of the system functionality without any concern about the privacy issues.

4.9 Standards compliance

System standard compliance issues can be divided into three classes: compression standards, data transportation standards and session control standards. First group is fully covered by the system implementation. All the compression algorithms both audio and video are fully compliant to the existing standards. At the application level the system is fully open – it can communicate with any other standards compliant videoconferencing application.

BuenaVista uses RTP protocol for the application data exchange. This solution provides standardization at the transport level. All application packages can be received and properly interpreted by other open systems. This structure altogether with the application level compatibility offers the full application data exchange standards compliance. The only feature of RTP not covered by BuenaVista is the RTCP. However, this protocol was created primarily for the lousily coupled conferences. Since BuenaVista adopts the tightly coupled session model most of this functionality is unnecessary. The only exceptions are the participant reports, offering useful information about transmission problems. Thus, we intend to include those elements into the future version of the system.

Thus far, the only area not supporting any standards is the session control. Currently, the best solution for the session control protocol is definitely the H.323 standard. Adopted already by many software vendors probably will prevail as the most popular conference control standard. Although, BuenaVista does not implement H.323, our control model is very similar to the one offered by the standard. Therefore, we predict to port the system to the H.323 platform easily, effectively and quickly.

4.10 Integration with Web-oriented Collaboration System

In recent years, World Wide Web (WWW) has become the most popular and attractive area of the Internet. Millions of people browse the Web every day, accessing tons of information. But modern Web is much more than a tool to display text and images. Many interactive applications transformed the Web into the biggest multimedia, collaborative tool available. Therefore, we attempted to combine our system with this powerful environment. As a connecting element, facilitating the integration, we chose the Web collaborative application developed in NPAC – Tango, presented briefly below.

4.10.1 Overview of Tango collaborative environment

Tango is an integration platform that enables building Web-based collaborative environments. The system provides the means for fast integration of Web and non-Web-applications into a multi-user collaborative environment. The main functionality provided by the system consists of session management, communication between collaborating applications, user authentication and authorization and event logging. Application of this Java/WWW-based collaborative framework is focused on military command and control, Internet distance education and remote collaboration.

Tango has client-server architecture, seamlessly integrated into the World Wide Web. Key server and client components are written in Java for multi-threaded Java collaborative server management, session control and a Web user interface. System is integrated with the Netscape Navigator browser, which serves as a client collaboration interface and working environment. The most important client-side component is the Netscape plugin, which allows placing the system inside the browser. For inter-applet communications Tango system utilizes Netscape LiveConnect technology. It is anticipated that Tango will be ported also to Internet Explorer platform.

The environment provides simple APIs for Java and C applications, which can be easily integrated into the system. Tango is the multi-user and multiplatform system, currently available for SGI, Sun and PC. It offers several collaborative applications like whiteboard, chat, collaborative browser, presentation tool and various educational applications and demos.

4.10.2 Tango vs. BuenaVista

Both presented above systems have many unique features enabling collaboration in the Internet environment. However, none of them covers the full scope of possible collaboration aspects. Tango is Web-oriented system based on the idea of connecting applications through a central server. This solution gives better control over management and maintenance of the collaboration process and enables the multisession mode. It also allows adding new applications easier, with less work overhead (e.g. Java applets). Nevertheless, it does not support efficient enough the real-time applications. The central server that is a big advantage concerning mentioned above features, becomes a bottleneck and decreases performance of such applications, not meeting their needs. The Web-independent, “out-of-browser” system is also more reliable and more suitable for these specific requirements.

Taking into account all those features, we concluded that the best solution would be the integrated system consisted of both Tango and BuenaVista. Certainly both systems can work separately matching the particular needs of some users, but combined hybrid is a powerful tool based on state-of-the-art technologies, offering rich functionality necessary in modern desktop collaboration environment.

Obviously, reaching this goal is connected with a variety of different problems. Both systems are based on the opposite approaches. Their architecture, communication channels and protocols are totally distinct.

4.10.3 Structure of integrated videoconferencing application

To obtain seamless integration following activities were undertaken:

- removing the existing graphical user interface
- creating the BuenaVista-Tango control protocol
- installing the Tango communication unit instead of GUI

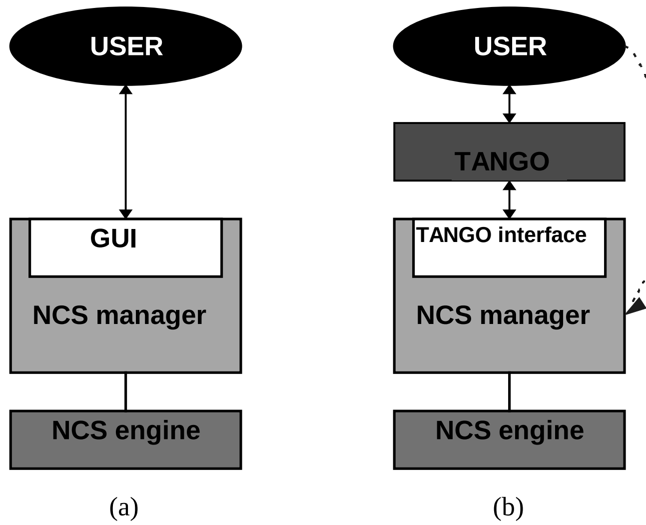


Figure 16 (a) BuenaVista-user interaction (b) BuenaVista-Tango integration

The main problem of the integration was linking the multi-session Tango with the one-session BuenaVista. It was overcome allowing just one BuenaVista session per machine and creating a simple graphical user interface enabling the BuenaVista applications management.

5. Conclusions

The conferencing system described in this paper goes beyond functionality offered in other systems created thus far. First of all, it is multiuser environment enabling fully distributed conference participation. System provides all necessary functionality to effectively create and control multiuser sessions, providing ability to participate in the audiovisual meetings.

Its functionality is accessible from different platforms. It covers two currently the most popular multimedia environments: the SGI Unix workstations and PCs. Additionally, covering the 32 bits Windows platform and the Unix environment, it allows to be easily ported to Macs, Suns and other popular operating systems.

Another important advantage of the system is its extensibility. One of the assumptions during the design stage was the ability to modify the existing environment or add new components. Therefore, we decided to create modular system, where new components do not interfere with the existing ones and one element modification does not result in changes in whole system. We implemented the development APIs for different programming languages (C, C++, Java). Thus, it is possible to port an application written in any of those languages. Java API enables to create the universal applications working on all platforms.

BuenaVista presents flexibility needed for modern collaborative tool and the abilities not just to exchange information but also to manage them. The audio-video applications offer several operating modes, allowing the adaptation of the system to different circumstances. It is particularly convenient to be able to control the application bandwidth and quality, based on the accessible network connections. The system can also be integrated with other collaborative environments currently available. It was verified linking BuenaVista with the Web-based Tango collaborative environment.

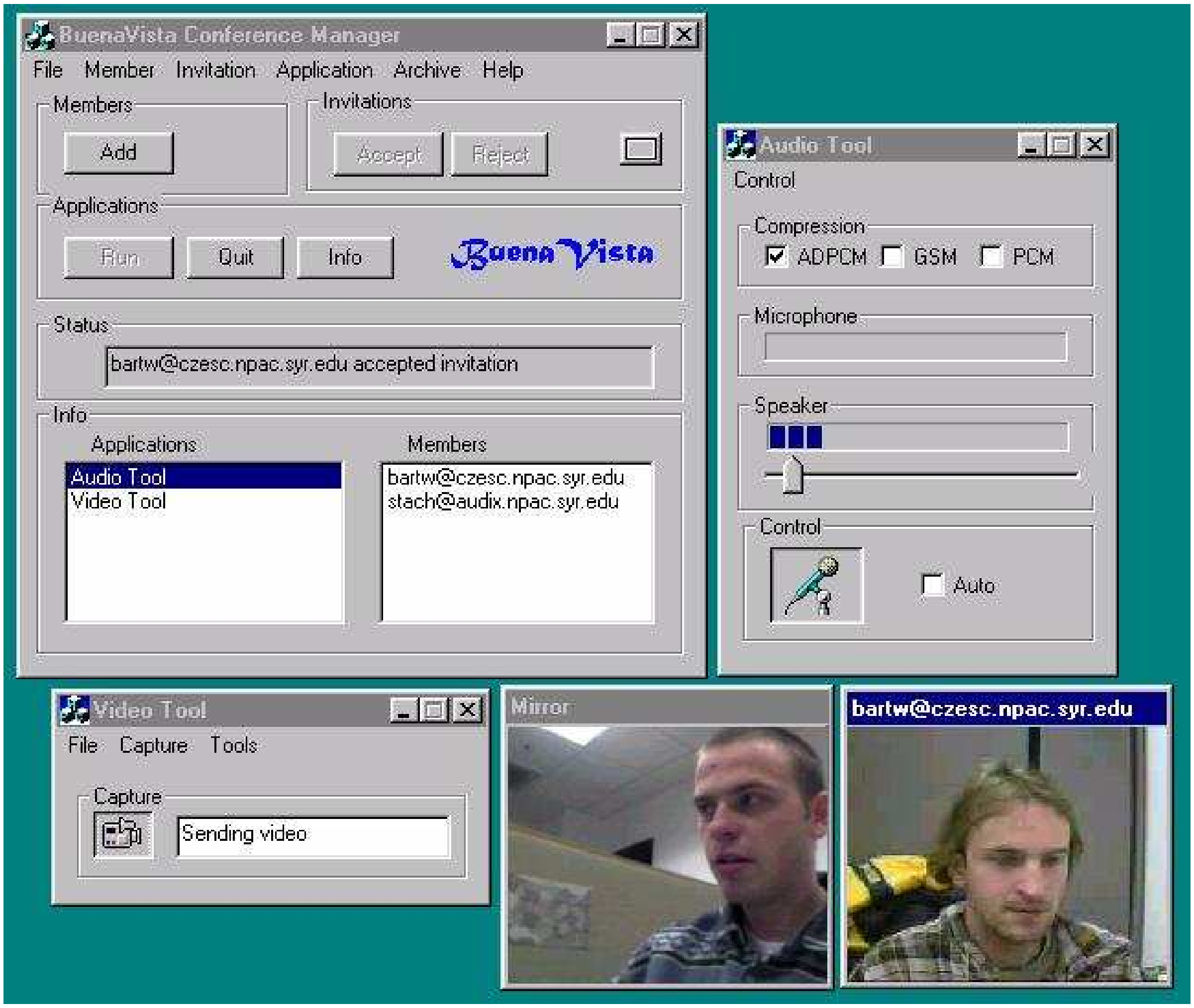
Our implementation addresses another issue that has not appeared anywhere else, thus far. It is the support for archiving of audiovisual conference content. Archiving of real-time, multiuser, distributed sessions if extremely complicated problem. BuenaVista offers such capabilities. For the time being those are only the local storage and retrieval of conference content. However, we intend to further develop that functionality, eventually integrating the system with a multimedia database.

Finally, applying a variety of multimedia standards such as RTP, H.263, H.261, GSM, ADPCM, we provided system interoperability. Although, for the time being some of the standards are implemented only partially and it is not possible to cooperate directly with other systems, we intend to make it fully compliant. Additionally, none of the products existing on the market does offer full interoperability, either. Thus, the issue of interoperability for videoconferencing products is still open.

Certainly, our implementation has also several disadvantages. As we stated, we still lack full standards compliance, especially regarding H.323. In spite that the system was extensively tested, there are still some problems connected with its robustness. Importance of this factor caused even creating a simplified, combined version of BuenaVista – simpler and much more stable but not so flexible and extensible. Another issue is some drawbacks in the control paradigm such as only one session per host and the session joining model. Finally, especially from commercial point of view, the problem of graphical user interface (GUI) seems to be very important. However, since this implementation was primarily concerned about creating a system infrastructure and solving technological problems, the issue of GUI was of secondary importance for us.

6. Appendices

6.1 BuenaVista for PC screenshots



7. References

[1] Lynch, Thomas J. “Data Compression Techniques and Applications”, Van Nostrad Reinhold Company Inc., 1985

[2] Clarkson, Peter M. “Optimal and Adaptive Signal Processing”, CRC Press Inc., 1993

[3] Degener, Jutta “Putting the GSM 06.10 RPE-LTP algorithm to work”, December 1994, http://www.ddj.com/ddj/1994/1994.12/degener.htm

[4] ITU-T Recommendation H.263 “Video Coding For Low Bitrate Communication”, July 1995, http://www.fou.telenor.no/brukere/DVC/h263_wht/

[5] Telenor “H.263 Advanced Negotiable Options”, December 1995, http://www.fou.telenor.no/brukere/DVC/h263_options.html

[6] ITU-T Recommendation H.261 “Video Codec For Audiovisual Services at P x 64 kbits”, March 1993

[7] Hung, Andy C. “ PVRG-P64 Codec 1.1”, November 1993

[8] Insoft Inc. “OpenDVE Architectural Overview”, 1996

[9] ITU-T Recommendation T.120 “Data Protocols for Multimedia Conferencing”

[10] VTEL Corporation “H.320: A Quality Requirement Guide”, 1996

[11] Bulawa, Janusz “Integration of multimedia collaboratory environment with Web browser” – Master Thesis, September 1996

[12] Rettinger, Leigh Anne "Desktop Videoconferencing: Technology and Use for Remote Seminar Delivery" (Under the direction of Dr. Thomas K. Miller III.), 1995, http://www2.ncsu.edu/eos/service/ece/project/succeed_info/larettin/thesis/

[13] C. Bormann, J. Ott, C. Reichert "Simple Conference Control Protocol", Internet Draft, June 1996

[14] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996

[15] ITU-T Recommendation H.323 "Visual Telephone Systems and Equipment for LAN which Provide a Non-guaranteed Quality of Service", November 1996

[16] M. Handley, H. Schulzrinne, E. Schooler "SIP: Session Initiation Protocol", Internet Draft, August 1997

VITA

NAME OF AUTHOR: Tomasz Stachowiak

PLACE OF BIRTH: Poznan, Poland

DATE OF BIRTH: July 25, 1973

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

Franco-Polish School of New Information And Telecommunication Technologies, Poznan, Poland

Technical University of Poznan, Poznan, Poland

PROFESSIONAL EXPERIENCE:

Research Assistant, Northeast Parallel Architectures Center, Syracuse University 1997