# Integration of multimedia collaboratory environment with Web browser

*Janusz Bulawa*

Master thesis prepared under
supervision of prof. Czeslaw Jedrzejek, EFP.

**Abstract**

This document describes design and implementation of the Web-integrated multimedia collaboratory

system.
The presented system provides collaborative services from within a World Wide Web browser. It allows for automatic multiparticipant teleconferences establishment and management, along with multimedia data distribution using the world's largest wide area computer network resources - the Internet network.
It integrates many newest Web technologies and advanced video compression schemes into one system and is integrated with Netscape Web browser acting as a *plug-in*.
Some preliminary adjustment of the component technologies had to be made to integrate them in one collaboratory multimedia system.
We believe that after some additional extensions our system will be advanced enough to be incorporated as an attractive teleconferencing tool in Web environment.

# Résumé

Ce document décrit la création et l'installation d'un système collaborative de multimèdia integré au Web.
Le système prêsenté propose des services de collaboration à l'interieur du Netscape dans le World Wide Web. Ce système autorise l'etablissement automatique et le gèrement de téléconférences à participations multiples ainsi que l'utilisation de distribution de données multimèdiatiques entre les participants utilisant les ressources les plus importantes mondialement en ce qui concerne les réseaux d'ordinateurs - le réseau Internet.
Ce système incorpore plusieurs nouvelles technologies pour le Web ainsi que des méthodes avancées de compression de vidéo et il fonctionne comme *plug-in* pour Netscape.
Quelques ajustements préliminaires sur les composants technologiques ont du être faite pour les integrer dans un système collaborative multimèdiatique.
Nous pensons qua après certaines extensions aditionelles notre système sera assez avancé pour être incorporé dans le Web en tant qu'outil intéressant pour les téléconférences.

# Streszczenie

Ponizszy dokument zawiera opis fazy projektowej oraz implementacji multimedialnego systemu telekonferencyjnego zintegrowanego ze srodowiskiem WWW (World Wide Web).
Przedstawiony system udostepnia uslugi telekonferencyjne uzytkownikowi przegladarki WWW. Pozwala na automatyczne zestawianie i zarzadzanie telekonferencja wielu uczestnikow oraz dystrybucje multimedialnych danych pomiedzy nimi uzywajac zasobow najwiekszej swiatowej rozleglej sieci komputerowej - sieci Internet.
System ten integruje w sobie wiele najnowszych technologii dla srodowiska Web oraz zaawansowane algorytmy kompresji video oraz jest zintegrowany z przegladarka WWW firmy Netscape stanowiac tzw. *plug-in* ("wtyczka").
Jako poczatkowy etap pracy, konieczne bylo wstepne dopasowanie skladowych technologii w celu integracji ich w jeden system do zarzadzania telekonferencjami multimedialnymi.
Mamy nadzieje, ze po wprowadzeniu kilku dodatkowych rozszerzen nasz system bedzie wystarczajaco zaawansowany aby zostac zaakceptowany jako atrakcyjne narzedzie do pracy grupowej w srodowisku WWW.

# Table of contents:

# I.Introduction

Recent years have brought an overwhelming growth of interest in use of world's information resources and computer networks for knowledge exchange and collaboration of large groups of people. The largest world-wide computer network, *Internet*, became a potential platform for *information highway* development. Development of the World Wide Web in 1989 was a milestone in the process of modern information highway creation. Billions of people, companies and organizations share their information resources via WWW. Number of data transfers over Internet grows steadily and rapidly. At the beginning of the WWW development most of data was in textual form, at present more and more of it takes form of audio, video, 3D graphics, etc. This increases significantly attractiveness of the WWW environment as a place when people meet and communicate with each other. Seeing many new potential communication possibilities, people are not any more satisfied with telephone, fax, and electronic mail communications.
Up to now, companies have been spending large amounts of money on their employees travels to meet their parties in different places all over the world. Numerous meetings of people from different countries are held every day. This consumes money and time, and also limits significantly theoretical possibilities for collaboration.
Therefore, the natural need for true multimedia communication services over Internet arose since the idea of

collaboration was mentioned for the first time. Businesspersons need tools that will prevent them from wasting their money and time on travels while providing true real-time multimedia collaboration services to any group of people in wherever all over the world.
Many researchers focused on the area of collaboration services. These services had to be first defined and then implemented and offered to people. Many collaborative systems have been designed over last years and also many new technologies have been developed to enable real-time multimedia collaboration of people. Currently available collaboration systems differ in the offered features, required network bandwidth and costs of installation. More sophisticated systems deploy proprietary solutions and require extensive hardware installations that obviously increase the costs of their use and eliminate a possibility of inter-system operation.
Commercial systems are usually not able to operate over Internet for the reason of limited and not sufficient network resources. Therefore, the number of potential users of such single system is quite limited. The access to commercial collaboratory system is expensive and no third parties can participate in a given teleconference.
Non-commercial systems are designed to operate in Internet environment (thanks to the MBone development) but the drawback here is a relatively poor quality of service and rather low reliability. Advantage of these systems is such that they are widely available and everyone willing to participate in a given teleconference can do so without any extensive preparation.
In spite of these, many people and companies use successfully today's commercial and non-commercial collaboratory systems for multimedia teleconferencing.
While many collaboratory systems are in use in today's business world, the researchers developed some new technologies that potentially enable teleconferencing over Internet with much better quality of service and more attractive features.
These enabling technologies are: *network protocols* (RTP, RTCP, RSVP), *multicasting* (IMCP, IGMP), *hypertext processing* (HTML language, HTTP protocol), *Java* language, *LiveConnect* capability and *LiveMedia* framework, advanced *video compression schemes* (H.263, wavelets), *collaboratory platforms* (such as OpenDVE technology), *virtual world modeling* techniques (such as VRML) and many others.

Being aware of many limitations and inconveniences of currently available we decided to understand some of the newest technologies and integrate them into one collaboratory system.
We therefore decided to design and build collaboratory system which will feature the following:

- will be integrated with a Web browser to provide a user with convenient interface
- will be modular in terms of teleconferencing tools for user data distribution (audio, video, graphics, files etc.)
- will be build on top of OpenDVE platform which is supposed to be a major component of the future LiveMedia framework
- will deploy advanced data compression/processing schemes to be able to operate with limited network bandwidth when providing satisfying quality of service (QoS)
- will be compliant with current standards for teleconferencing (H.320, T.120 ITU-T Recommendations) to the best possible extent

In the course of entire system implementation we integrated many separate technologies which sometimes required some additional research and adjustments of existing solutions. Finally, we reached our goals and the preliminary operational version of the Web-integrated collaboratory system has been created.
As for the time of redaction of this document, the collaboratory system featured the following:

- it operates within Web browser window and automatically establishes a teleconference when a file with participants addresses is downloaded by user
- session management is provided by OpenDVE library routines and Conference Engines
- H.263 video coding scheme is deployed for videoconferencing and this results in a very low bit rate (of about 25 kbps for a typical "talking heads" sequence at 10 QCIF frames per second)
- several teleconferencing tools for different media (audio and video included) are available and any other tool can be added to the system in the future (these tools are OpenDVE plugins and can be provided by third party)

We also encountered some obstacles and difficulties during the implementation of our system:

- multicast support in OpenDVE environment remains out of control of our conference manager. It is believed to be exposed in LiveMedia framework.
- use of Real-Time Transport Protocol and Resource ReSerVation Protocol is not possible in OpenDVE environment since there is no direct network interface in OpenDVE libraries. It is also believed to be covered in LiveMedia framework.
- H.263 video encoding is a very computationally extensive process which results in low frame rates on end-systems. Decoding is much faster but along with encoding the two processes load heavily a CPU. Therefore, in multiparticipant videoconferencing with several video streams more powerful CPUs are required on end-systems.

While some of encountered limitations are to be overcome by already announced technologies, there is still certain amount of effort needed to tune the entire system so that all its components will be efficient enough to provide real-time multimedia teleconferencing service on the Internet.

In this document we describe the design and implementation of our multimedia collaboratory system for the Web. Component technologies are also presented.
Section 2 contains description of two major standards for teleconferencing: H.320 and T.120 ITU-T Recommendations.
In section 3 features of some of the most important teleconferencing systems available on the market (commercial and non-commercial) are briefly described.
In section 4 the major component technology used in implementation of our system, the *OpenDVE - Digital Video Everywhere* is presented in details.
Section 5 contains description of World Wide Web environment and state-of-the-art Web technologies presentation.
In section 6 the design, architecture and implementation of our collaboratory system are presented and some possible further extensions are proposed.
Section 7 presents H.263 video coding scheme and contains description of Netscape H.263 video player plug-in implementation. It also describes modifications and adjustments to H.263 codec which were necessary to adopt it to our collaboratory system.
Finally, in section 8, short conclusions resulting from our work are presented.

To avoid confusion during the reading of this document we would like to explain difference between two similar notions used in this document: *plug-in* is assumed to be an application written with use of the Netscape Plug-In API and cooperating with Netscape browser; *plugin* is assumed to be an application created with use of the OpenDVE SDK and running in the OpenDVE environment.

# I.Current standards for teleconferencing systems

Desktop collaboration encompasses a number of existing and emerging technologies. These technologies are subject to several standards and recommendations concerning collaborative applications development. Presented below standards are mainly designated for applications for telephony environment but their extension to LAN and B-ISDN platforms are expected in the near future. However, employment of those standards and recommendations will help to ensure a compatibility and interoperability of the different collaborative applications making them easier to use also in the Internet environment. We have to remember that presented standards <u>do not guarantee</u> any compatibility since they provide some requirements in an abstract form without any detailed specifications.

In following sections the most important current standards for teleconferencing are discussed.

## *A.T.120 series of protocols*

The T.120 series of recommendations collectively define a multipoint data communication service for use in multimedia conferencing environments. They also provide facilities to establish and manage interactive communications involving two or more participants an and between a variety of different networks. It provides a multipoint data communication service for those participants, which is independent of the underlying network. Within a conference T.120 allows communication to be established between any combination of participants. The 120 series also provides support for applications and their associated protocols. It defines start-up mechanism, a means of identifying resources, and procedures to compute a common set of capabilities. The T.120 protocols provide a means of telecommunicating many forms of Data/Telematic information between two or more multimedia terminals and of managing such communication. They provide a multipoint data communication service that has a particular application in multimedia conferencing. The T.120 protocols are suitable for use on many types of network: PTSN, ISDN, CSDN, PSDN, B-ISDN, LAN. They provide the capability for seamless internetworking of applications between terminals connected to different networks.

The T.120 protocols provide:

- support for conference establishment among a group of network nodes (such as conference terminal and MCUs)
- mechanism to identify the participating nodes and a comprehensive roster and capability exchange mechanism
- flexible management of communication between any combination of these elements.

The T.120 protocols can handle one or more simultaneous conferences, a terminal may participate in more then one of these if authorized to do so. The *conveyor* of a conference may control the participation in that conference and the information that flows in the conference. No constraint is placed on the rate or volume of information transmitted within the various media; the T.120 protocols have the capability to organize different rates of information flow, within the constraint imposed by the type of network and connections established thereon. Not all of the T.120 protocol provisions are mandatory: T.123, T.122/125, and T.124 are mandatory for conferencing and group-working environments. The remainder are conditional: where functionality covered by the standards is provided, the standard protocols of the T.120 series <u>must</u> be implemented. This ensures that it is always possible to achieve a basic level of interworking, and does not prohibit customized enhancements and negotiation of proprietary modes if (and only if) all participating elements are able to support such modes.
Traditionally, telephony services have been constrained to point-to-point operation. In order to support group activities such as meetings, conferences (involving physically separated participants) there is a requirement to join together more than two locations. The term *multipoint communication* simply describes the interconnection of multiple terminals. Normally a special network element, known as a Multipoint Control Unit (MCU), or more simply a bridge, is required in order to provide this function.
A *conference* refers to a group of geographically dispersed nodes that are joined together and that are capable of exchanging audiographic and audiovisual information across various communication networks. Conference participants may have access to various types of media handling capabilities such as audio only (telephony), audio and data, audio and video, or audio, video and data.
The T.120 series of recommendations define the component which is used to provide both a data communications service, and a management service for any other media services present.
T.120 protocols provide the infrastructure required to provide data services for many types of conferencing and groupworking, making it suitable for a diverse range of application areas. It is expected to find use in videotelephony and audio-graphic conferencing as well as other forms of multipoint multimedia communication.
T.120 regards point to point connections as the simplest form (a degenerate case) of multipoint connection, and are therefore fully supported by the T.120 protocols. Terminals with multiple communication ports (each with an appropriate T.120 transport stack) can act as T.120 data bridges and allow multipoint connections to be established involving three or more nodes.
MCUs are nodes that do not normally support terminal functionality. They act as bridging nodes, bridging data and other media streams present in the connections.
Figure 2.1-1 show an example of how three MCUs may be connected to bridge a group of terminals.

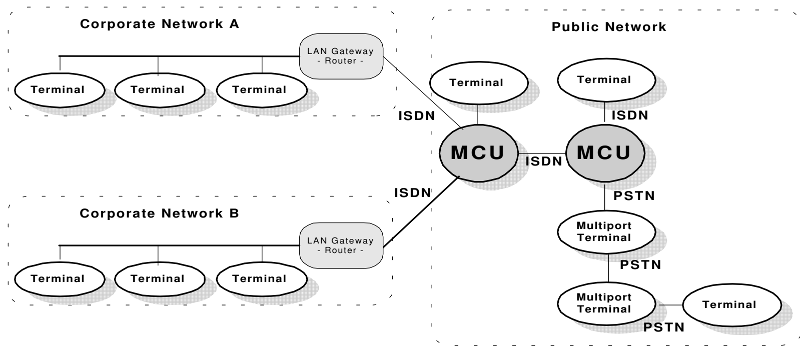



Figure 2.1-1 Example of a mixed-network conference topology

The T.120 model is comprised of a communications infrastructure and the application protocols that make use of it. Figure 2.1-2 shows the full model with both standardized and non-standardized applications. The model serves to show both the scope of the T.120 suite of recommendations (indicated by the shaded background) and the relationship between each of the recommendations and other components in the system. Generally, each layer provides services to the layer above and communicates to its peer(s) by sending Protocol Data Units (PDUs) via services provided by the layer below. This description will address each of the major functional levels in Figure 2.1-2: User Applications, Application Protocols, Node Controller, Communications Infrastructure and Networks.

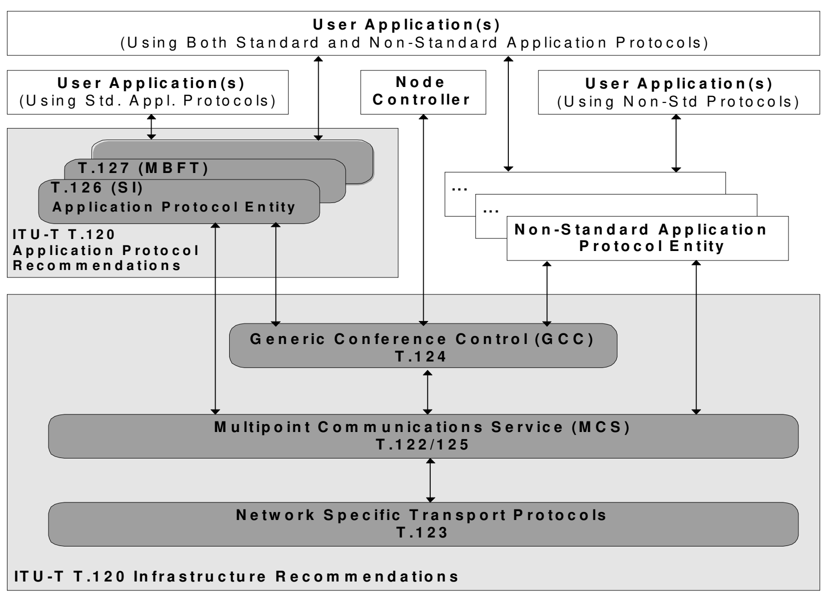



Figure 2.1-2 T.120 System Model

Applications *per se* are not the subject of standardization in the T.120 series. Applications that use the services offered by the T.120 series will generally be multipoint aware and designed to use the T.120 services provided by GCC and MCS. These applications are termed *User Applications* and they may use any combination of standardized and non-standard protocols to communicate with peer user applications. The T.120 environment supports multiple user applications concurrently operating in the same conference by providing mechanisms for the applications to co-ordinate the use of communications resources. The Generic Application Template (T.121) provides guidance to user application developers on how to utilize the T.120 infrastructure in a coherent and consistent way. The T.120 series includes a set of application protocols designed to meet the needs of multipoint conferencing. These protocols define minimum requirements in order to ensure interworking between different implementations. T.127 provides simultaneous Multipoint file transfer. T.126 provides still image viewing and annotation, shared whiteboard, and facsimile. A given application may use any combination of standardized and non-standard application protocols.

The *Node Controller* is the element that provides the T.120 management role at a terminal or MCU. It issues primitives to GCC (see below) which start and control the communication session. The node controller itself is outside the scope of the T.120 recommendations, and only where it communicates to GCC are its interfaces defined.
The communications infrastructure provides multipoint connectivity with reliable data delivery. It can accommodate multiple independent applications concurrently using the same multipoint environment. Connections between nodes can be any combination of circuit switched telecommunications networks and packet based LANs and data networks. The T.120 infrastructure is composed of three standardized components: Generic Conference Control (GCC), the Multipoint Communication Service (MCS) and Transport Protocol Profiles for each of the supported networks.

*Generic Conference Control* (T.124), GCC, provides a set of services for setting up and managing the multipoint conference. It provides access control and arbitration of capabilities. GCC facilities are used by applications to co-ordinate the use of MCS channels and tokens within the same multipoint Domain. Nodes can join and leave meetings at any time and GCC facilities can be used to query an MCU or multiport terminal node to find a desired conference. Multiple applications can be running on any given node and can be dynamically launched, used and shut down during a meeting. As part of the management role, peer GCC providers exchange information about the applications present and their capabilities. GCC also makes a centralized registry facility available to applications in order to identify dynamically assigned channels and tokens.

*Multipoint Communications Service* (T.122/125), MCS, provides a general Multipoint connection-oriented data service. It collects point-to-point Transport Connections and combines them to form a Multipoint Domain. Within that Domain a large number of logical channels are provided that can provide one-to-one, one-to-many and many-to-one data delivery. Nodes within an MCS Domain are hierarchically organized in a tree structure. Data delivery normally follows the most efficient path to the nodes that are to receive the data, but a mechanism is provided to guarantee that data originating from different nodes is received in the same sequence at all nodes. MCS acts as a resource provider to the layers above, independent of the underlying network; providing channels and token resources on demand. A large number of tokens are provided for applications to use for coordinating events and processes.
MCS expects its underlying *Transport Connections* to provide reliable point-to-point sequenced data delivery of its PDUs and to segment that data if necessary. T.123 (*Transport Protocol Stack Profiles*) specifies a protocol stack for each particular network supported. T.123 presents a uniform OSI Transport Service interface to the MCS layer above.

The T.120 suite provides for operation over the following networks:

- PSTN - Public Switched Telephone Network (or compatible service).
- ISDN - Integrated Services Digital Network as defined in ITU-T I-series Recommendations.
- CSDN - other digital circuits using H.221 framed signals.
- PSDN - Packet Switched Data Network using X.25.
- Other (switched or permanent) digital circuits using H.221 framed signals.
- Use of T.120 on other networks such as B-ISDN and LANs is currently under study. Alternative protocol stack profiles may be defined in the future.

The approach taken with the T.120 architecture leads to the multipoint routing information being located in MCS above the transport stacks, this is the key to the network independence that can be achieved with T.120.
The T.120 protocols are designed to operate over a wide range of networks and indeed to facilitate communication between end-points on a mixture of networks. The differences in T.120 operation for the various networks are confined to the lowest layers as detailed in recommendation T.123. Recommendation T.123 defines the <u>network specific transport stacks for each supported network</u>. Generally, existing link layer protocols appropriate to each network are selected and then mapped into a common interface layer, thus defining a transport profile for a given network. At Transport level the conference is seen as a group of point to point connected pairs ( different pairs may be on different

networks). Multipoint Communication Service (MCS), takes the transport pairs from the layer below it and maps them into a multipoint domain.

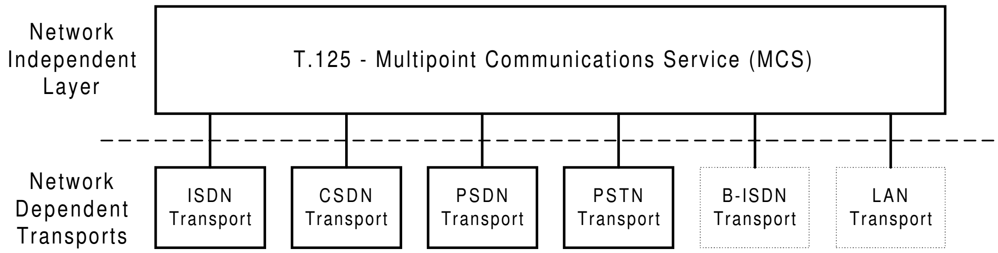



Figure 2.1-3 Protocol Stacks - defined in T.123

T.120 series includes application protocols that provide commonly required functionality to user applications, in a way that ensures a guaranteed level of interworking across a diverse range of terminals with differing capabilities.
Provision of non-standard protocols may be made by using the extension capabilities to provide enhanced functionality along side the existing T.120 standardized protocols or it may be made by provision of additional proprietary application protocols which use the resources of MCS and GCC.

For use within a multimedia conferencing environment T.120 requires:

conformance with the Transport Protocol stack profile (T.123) for the selected networks
conformance with the Multipoint Communication Service protocol (T.125)
conformance with the mandatory parts of Generic Conference Control (T.124)
conformance with the mandatory parts of any standardized application protocols that have scope covering functionality supported by the user applications.

Conformance to T.120 is specified in the table 2.1-1. Options within each item are clearly set out within the requisite standard; thus, except where otherwise stated, in this table conformance means to the minimum requirement of the standard.

Table 2.1-1 T.120 Conformance Table [17]
Item Status Conditions
T.123 Mandatory Basic mode profile, according to network type
T.124 (GCC) Mandatory Mandatory protocol elements - as specified in Table 6-1/T.124, according to node type.
Conditional protocol elements - Requirement dependent on the needs of application protocols to be supported at the node.
In addition: a terminal attempting to enter an existing conference via GCC-Conference-Join request shall specify exactly the minima and maxima of the MCS domain parameter values defined in Annex B as its allowed range of negotiation.
T.125 (MCS) Mandatory The defined protocol is mandatory. The domain parameter values selected for an MCS domain shall lie within the ranges defined in Annex B of this recommendation.
T.126 (SI) Conditional

Mandatory when a user application requires one or more of the following features:
exchange of soft copy still images
exchange of hard copy still images(including FAX)
shared whiteboard functionality
exchange of annotated soft copy still images
Note- T.126 Annex A defines those parts of T.126 that are mandatory for each of the above functions.
T.127 (MBFT) Conditional
Mandatory for general purpose file transfer applications.

The following information has been prepared on the basis of document [16] provided by VTEL Corp. H.320 is an overview document that has set the standard for videoconferencing, mainly with use of ISDN links. As the ITU-T develops standards, one goal is to define the minimum requirements all videoconferencing vendors must support. This minimum requirement ensures that all H.320-complaint systems will communicate with each other. The optional requirements affect the quality of the audio and video and their implementation is left to the manufacturer. The factors that affect the system quality are:

- Picture Resolution
- Frame Rate
- Pre- and Post Processing
- Motion Compensation
- Audio.

A videoconferencing system can belong to one of the following classes:

- Class1 - minimum level of support (required by H.320)
- Class 2 - support of some optional features
- Class 3 - support of all optional features

In H.320 system there are two types of **picture resolutions**: *Quarter Common Intermediate Format* (QCIF) and *Common Intermediate Format* (CIF). Class 1 systems will only support QCIF, Class 2 systems may support CIF, and Class 3 will support CIF. The ability to support CIF resolution is critical to the quality of the picture. CIF capable systems provide a picture resolution of 352 x 288. QCIF provides quarter the resolution -- 176 x 144. A QCIF picture will look like an unfocused blocky image. A high-quality system will be forced to degrade its resolution if it connects to a low-quality system. For example, if a Class 1 and Class 3 systems connect in a call, the Class 3 system will be forced to display a QCIF resolution.

**Frame Rate**, also referred to as the Number of Frames Per Second (fps), is the number of times the picture image is refreshed. H.320 systems can support frame rates of 7.5, 10, 15, or 30 fps. A low frame rate will cause the motion to appear jerky. The higher the frame rate, the smoother the motion will be. Class 1 systems are required to support a frame rate of 7.5. Support for the higher frame rates are left to the manufacturers and the ones with a desire to deliver high quality products will support higher frame rates. Typically Class 2 systems will support up to 15 fps. Class 3 will support up to 30 fps. Unfortunately even the manufacturers who deliver Class 3 systems are affected by manufacturers who are not driven to produce high quality products. This means if a Class 1 system connects to a Class 3, the Class 3 system will have to downgrade its capabilities to communicate to the Class 1system. The Class 3 system would be forced to use 7.5 fps for the call to work.

**Pre-processing** is a complex process to reduce the amount of re-coding the background. This is not required for Class 1 systems. If pre-processing is not used the video encoder can spend a lot of time encoding 'noise' caused by the poor lighting of the camera. These situations may make the system think that there is motion in the background when in fact there is none. Pre-processing ensures that only real motion gets encoded. **Post-processing** will compensate for picture degradation's caused by fast motion. Post-processing can reduce the blocking and noisy effects caused by H.320 video coders. Post-processing can also be used to enhance the frame rate and reduce the jerky motion effect caused by low frame rates.

There are two aspects of **motion compensation**: motion estimation and the actual motion compensation. *Motion estimation* is performed at the encoder to determine what the motion vector should be. *Motion compensation* is performed at both the encoder and the decoder, and consists of moving blocks of data around based on the motion vector. Essentially, motion compensation only encodes the section of video where motion has occurred rather then encoding the entire video area for every frame. This is especially important at lower bit rates. All H.320 systems must have the ability to decode a motion compensation signal. Providing encoded motion compensation is optional. The encode ability is where video quality improvements are made. A Class 1 system will not support the encode compensation, a Class 2 system may support a limited form of motion compensation, and a Class 3 system will support a higher form of encode motion compensation.

H.320 specifies three types of **audio**:

- G.711 -- 48-64 Kbps Narrowband
- G.722 -- 48-64 Kbps Wideband
- G.728 -- 16 Kbps Narrowband

A Class 1 system only has to support G.711. G.711 is telephone quality audio (narrow-band, 3khz). G.722 produces stereo-quality audio (wideband, 7Khz). A higher data rate, typically 256 Kbps and above, Class 3 systems will support G.722 offering the user the best audio quality available. Class 1 systems will not support G.722, and Class 2 typically will support G.722. G.728 (16 Kbps) is narrow-band audio but it is important for low bit-rate calls, typically less then 256 Kbps. For example, in a 128 Kbps videoconference (typically via BRI) a high-quality manufacturer will offer the user to select between G.722, 64 Kbps, or G.728, 16 Kbps. If audio quality is more important then video quality (64 Kbps for the audio will only leave 64 Kbps for the video), the user can choose to use G.722. If video quality is more important the user will use G.728. The audio will use 16 Kbps leaving 112 Kbps for the video. High-quality manufacturers will allow the user this choice, not dictate it to them. Class 3 systems again are limited by the lesser Class systems. If a Class 3 system establishes a call with a Class 1 system the audio will be G.711. The Class 3 system is forced to use lower-quality audio in order for the call to work.

**Data rates** have an impact on the perceived quality. The higher the data rate the better the video quality will be. At T1 speeds (1536 Kbps) the video quality will be optimal. Realistically most users will not run at this speed, this is mainly a cost issue. In today's market many users are using 768 Kbps as a trade-off between high quality video and cost. 384 Kbps is probably the most common data rate in use today, 128 Kbps is becoming more popular with the availability of ISDN. As desktop videoconferencing becomes more mainstream 128 Kbps will become one of the more common data rates.

Many factors affect the overall quality of an H.320 system. The ability of vendors to implement optional features will greatly impact the overall quality. The user should understand their environments and the application the H.320 system will be used in. These factors can help make a decision on what the requirements for the video conferencing system will be. Applications can range from desktops to boardrooms and each will have its own requirement. Not only will the system the user purchases determine the quality but the capabilities of the system on the other end will have a major impact to the overall quality. A lesser capable system will always force the high quality system to downgrade its quality in order for the conference to be held. The best solution for a user is to require a system that is flexible and software upgradable. Flexibility means the system can be configured to the requirements of the user, which may change over time or even may change from conference to conference. Software upgradability provides investment protection to the user by allowing the user to take advantage of new features without being required to buy new hardware.

Conformance to H.320 is specified in the table 2.2-1.

Table 2.2-1 H.320 Conformance Table [16]

| **Item** | **Class 1** | **Class 2** | **Class 3** |
| --- | --- | --- | --- |
| Picture Resolution | QCIF mandatory | CIF optional | CIF mandatory |
| Frame Rate | 7.5 fps | up to 15 fps | up to 30 fps |
| Motion Compensation | must be able to *decode* motion compensation signal | limited form of *encode* motion compensation | higher form of *encode* motion compensation |
| Audio | G.711 | G.722 | G.722, G.728 |

More information on H.320 can be obtained from ITU-T Web home pages.

# I.Description of available videoconferencing systems

Today's multiparticipant conferencing systems help people to conduct real-time desktop-to-desktop meetings with a transmission of a multimedia data. Almost all such systems support file transfer capability, whiteboard, shared applications, text tools, etc. However, the most challenging issue in a teleconferencing system is its capability of live video transmission. This feature is the one that differs the most from system to system. Potential customers and users are mostly interested in the quality of video versus the costs of installation.

We can find two major categories of teleconferencing products: commercial systems and those for the World Wide Web environment. In the following sections we discuss some of the products from both categories focusing on their functional features, required network bandwidth and additional hardware as well as on the costs of installation.

## *A.Commercial systems*

Commercial teleconferencing systems are designated mostly for use in business, i.e. in companies, enterprises, co-working groups etc. These systems usually feature a better video quality but, on the other hand, they often use proprietary solutions and therefore are not able to interoperate with each other. The other issue is the cost of such commercial systems which is usually rather high, especially when some additional hardware components are required.

Commercial systems are always well evaluated and robust which make them be attractive from the point of view of the customers. On the basis of information gathered in "Video Conferencing Buyers Guide" published in [15] we analyzed typical features of today's commercial teleconferencing systems and we present our conclusions below.

There exists at least several decents of commercial teleconferencing systems from different manufacturers. Communique! from InSoft Inc., LIVE PCS products family from PictureTel Corp., InPerson from SGI, ShareVision from Creative Labs Inc., ProShare from Intel Corp., Person-to-Person from IBM, ShowMe from SUN Microsystems, VTEL 115/117 from VTEL Corp. are few examples of such systems.

Some of commercial teleconferencing systems  are entirely implemented in software whereas some requires quite costly additional hardware. Typically a video and/or audio compression card, a network interface card, camera and microphone are required for a commercial product. This increases significantly the costs of installation, even if not very sophisticated hardware is needed. Furthermore, it also improves the quality of video and audio data which is rather important to the user.

Most of those systems, but not all of them, offer file transfer, echo cancellation, whiteboard, app sharing, multipoint connections and self-view feature.

The coded video picture size is typically of 176 pels per 144 lines (QCIF format) or 160 pels per 120 lines. Some systems offer 320x240, 352x288 or even bigger picture formats.

As far as the network interface is concerned, the commercial systems use Ethernet, ATM, FDDI, TokenRing and ISDN links, depending on the coding scheme and the picture size. Usually TCP/IP protocol is used which, in our opinion, is the simplest and the least efficient solution.

Video coding is typically multioptional. A user can choose between standardized encoding schemes such as MPEG, Motion JPEG, H.261 or CellB. Some proprietary solutions are also available.

Framerate varies from few (!) frames per second to 15, 20 or even 30 fps. What was surprising is that some commercial systems only offer 3-4 or 6 frames per second speed which in our opinion is far to few for a commercial product using dedicated hardware.

Network bandwidth required by offered commercial teleconferencing systems is far too large to think about using them over today's Internet. They usually require hundreds of kilobytes per second, some of them however can work with two ISDN Basic Rate channels (64 kbps each) giving a rather poor video quality.

There is not much said about audio coding in those systems but it seems to be usually one of the G-series audio compression algorithm employed (G.728, G.711 etc.)

According to the presented information we concluded that the offered functionality of the commercial conferencing systems is not particularly good and that very often the manufacturers do not pay attention to more efficient and more advanced solutions.

Furthermore, not all of the available systems are fully complaint with H.320 recommendation (see section 2.2 in this document) and some of them require a really expensive network infrastructure (ATM links, multichannel ISDN lines, etc.)

For detailed comparison of the most popular commercial systems for teleconferencing from thirty nine different vendors, please refer to [15].

## *A.Conferencing systems for the Web*

The teleconferencing systems for the World Wide Web environment differ significantly from the commercial ones. The major reason is that here one can not assume a large network bandwidth available. This in turn impacts a video coding scheme, a picture size and a number of transmitted frames per second. Usually these systems are fully implemented in software and use more sophisticated video coding schemes in order to cope with insufficient network bandwidth available. There also has been a lot of work done in order to develop appropriate transport protocols (see section 5.4) which would be able to transfer the increasingly demanding amount of data in videoconferencing. As a result, an entire suite of protocols and applications have been developed which seek to alleviate the many problems that exists in video conferencing.

In the World Wide Web environment these are a multicast transport scheme and the real-time transport protocols (such as RTP, described in section 5.4.1) that enable a real-time multipoint multimedia conferencing. Along with the development of the MBone and multicast protocols (see section 5.4.3) it became feasible to run conferencing applications on the Internet. All conferencing tools for WWW use those technologies without which they would simply not be operational.

In the following paragraphs we describe the most popular teleconferencing packages currently in use.

- **Network Video** (NV), developed by Xerox Park was one of the first network video only conferencing packages. The entire system is implemented in software and makes use of RTP and IP multicasting. This package uses a proprietary video coding scheme which gives a compression factor of about 20-35. The encoding of a one frame takes about tens of milliseconds and typical framerate equals to 10-15 frames per second (reasonable for a local area network with a typical workstation). The Haar wavelet transform and discrete cosine transform are used to encode those 8x8 pixels blocks which have changed significantly as compared to the previous frame. No motion estimation is employed. A user can increase a framerate by lowering the quality of coded pictures. This is mainly imposed by the network's inability to handle a large amount of video data. NV is capable of video capturing from an X11 window and therefore making itself compatible with many platform-dependent video grabbers. For more details on NV, please refer to [19] or [20].
- **Inria Videoconferencing System** (IVS), developed by INRIA researchers was one of the first video and audio conferencing systems. The audio coding scheme in IVS gives a quite good quality and generated bitstream takes up to 64 kbps. Video data uses a mere 128 kbps. The images are usually a quarter screen or smaller. IVS uses H.261 video compression algorithm which due to its advanced motion estimation and compensation techniques reduces significantly the amount of data to be transmitted. The drawback is that due to its complexity H.261 encoding and decoding takes more time. Therefore, on a lower end its performance in video encoding is rather not sufficient. On the other hand, IVS system is the only one available supporting synchronized audio and video in one application. The popularity of IVS package may grow with the appearance of the faster workstations. For more details on IVS, please refer to [19] or [20].
- **Videoconferencing Tool** (Vic), developed by Lawrence Berkeley Laboratories, is perhaps the most flexible conferencing system available. It combines all that was gained by its predecessors, NV and IVS along with adding some new features of its own. It supports many more compression algorithms and has a user interface that make it particular useful. It is a video only tool and has to be used in

conjunction with an audioconferencing tool like VAT (described below). Vic supports the following video compression schemes: MPEG, JPEG, NV, CU-SeeMe, CellB and its proprietary intraH.261. Therefore, using the Vic tool, one can join many different conferences as long as they use one of the aforementioned compression algorithms. The innovation here is that instead of encoding the block changes in a picture, which depends on previous frames, this version of H.261 simply encodes the entire changed block. Vic takes advantage of the RTP protocol to coordinate many different compression schemes (RTP header can have any application specific extensions). A new feature called *conference bus* is implemented in VIC. This allows all participants to receive critical information concerning a course of a teleconference (floor assignments, different conference formats). Another big advantage of VIC is that it is able to support different hardware codecs. Vic reaches a sending framerate of 30 fps at a bandwidth of 1.1 Mbps. Unlike NV, VIC is not able to capture video data directly from an X11 window and therefore is dependent on a video grabber type in use (Vic supports a very limited set of video grabbers). For more details on Vic, please refer to [19], [20] or [8].

- **Visual Audio Tool** (VAT) is the compliment to Vic that carries an audio portion of a teleconference. Like IVS, It supports several audio compression schemes with the maximum bandwidth of 64 kbps. It also uses RTP protocol that allows to mix video and audio streams into one according to 32-bit sender identifiers implemented by the RTP. For more details on VAT, please refer to [19] or [20].
- **Session Directory** (SD) is the glue that makes the whole idea of teleconferencing work. Developed by the Lawrence Berkeley Laboratories, SD advertises on-going conferences as well as coordinating the different tools for video, audio and sketch pad. Whenever a group of participants decides to take part in a teleconference, they have the choice to advertise the conference to the rest of the MBone community via the SD tool. Anyone else running the SD tool will be presented with a list of on-going conferences. In order to join a conference, all that one has to do is to select a particular conference from the list. SD takes care of selecting the proper multicast address and port number to transmit and receive on. SD is also able to manage the session tools as well as it takes care of synchronization. The inconvenience we observed here is that there is no restriction on who can join a multicast group which is not a very good solution with regard to the security issues. Cryptography can solve that problem but not all of presented tools are able to encrypt and decrypt the data. For more details on SD, please refer to [19] or [20].

Presented tools for the WWW are less sophisticated than commercial ones but they bring a real teleconferencing to the desktop through the Internet and therefore are available for the widest spectrum of users.

## A.Proposal of an improved conferencing system for the WWW

In the above sections we described merely some of the most popular teleconferencing systems for commercial use and for the World Wide Web environment.

First group of products, the commercial one, lacks an efficient transport capability and this limits their use to the local area networks or dedicated links. They are also relatively expensive and usually do not interoperate.

Second family of products, those for the WWW environment are much more efficient in terms of the data transfer across the network but they lack a sufficient quality, mainly for video. In spite of having all state-of-the-art concepts implemented they suffer much from the fact that there is only limited and unstable network bandwidth available. They do not deploy the most advanced compression techniques, and are usually limited to one or two different media.

Having in mind all conclusions concerning the currently available teleconferencing systems we decided to work on a new system which will feature the following:

- will be integrated with a WWW browser to make a session initialization be fully automatic
- will feature high-compressed video giving an extremely low bit rate for video data
- will be capable to run both video and audio on ISDN links
- will be compliant with current H.320 recommendation
- will integrate all state-of-the-art Web technologies
- will be modular in terms of used conferencing tools (separate tool for different media)

To achieve our goals we decided to use the following technologies:

- **OpenDVE collaboratory environment**, that is supposed to be an integral part of LiveMedia framework announced recently by Netscape Communications Corp. and adopted by more than twenty leading in teleconferencing and multimedia technology companies (see section 5.4.3)
- **H.263 video coding scheme**, that will give us a superior compression ratio and picture quality (see section 7.1 in this document)
- **Netscape Plug-in Application Programmers' Interface**, that will allow us to integrate the entire system with Netscape WWW browser (see section 6.1 in this document)
- **LiveConnect** technology, that will allow us to integrate the system with Java and Java Script giving a whole spectrum of possible innovations to the system with the use of Java language capabilities (see section 5.4.5 and 7.3 in this document).

These were our goals that we made effort to achieve through the course of the research work.

# I.OpenDVE™ - Digital Video Everywhere technology

**Digital Video Everywhere**, DVE, is an open software architecture developed by InSoft Inc. which currently is a part of Netscape Communication Corporation. The OpenDVE™ API provides the application developer with productive environment to create distributed multimedia and collaborative applications. The developers of InSoft's collaborative conferencing and distributed digital video applications (Communique!™ and INTV!™) have had the benefit of four generations of product development leading to a maturity of product design reflected in the OpenDVE architecture. The latest generations of InSoft's products (collaborative computing, desktop conferencing, distributed digital video development tools and business applications) are built using the OpenDVE environment [1].

In following sections we describe features, architecture, user data flow and session management in OpenDVE environment. Presented information is based on our own experience gained by analysis of exemplary OpenDVE plugins and undocumented header files in OpenDVE SDK. Some basic details come from InSoft's documentation.

## *A.Main features*

DVE architecture encompasses the following principles:

- collaboration between multiple desktop systems including Intel-compatible PCs, Sun, Hewlett-Packard, Digital Equipment Corp., IBM and Silicon Graphics workstations
- transparent network connectivity across Ethernet, Token Ring, ATM, Frame Relay, FDDI, ISDN and SMDS
- media interoperability using different coding and compression algorithms (implemented in plugins)
- scalability (in terms of number of participants and number of plugins being used simultaneously)

Using programming environment for collaborative applications based on OpenDVE technology, developers can efficiently:

- create customized collaborative environments with the appropriate access and membership policies
- add collaborative features and/or distributed multimedia functions to specialized applications
- exploit the latest state-of-the-art audio, video, and network technologies
- enable open collaborative applications

One of the most important features of OpenDVE from the application developer's point of view is its modularity which allows to create different tools, different versions of the same tool and integrate them all into one collaborative environment which, among the others, was taken advantage of in our work.

For more detailed information on the OpenDVE technology features, please refer to [1] and [2].

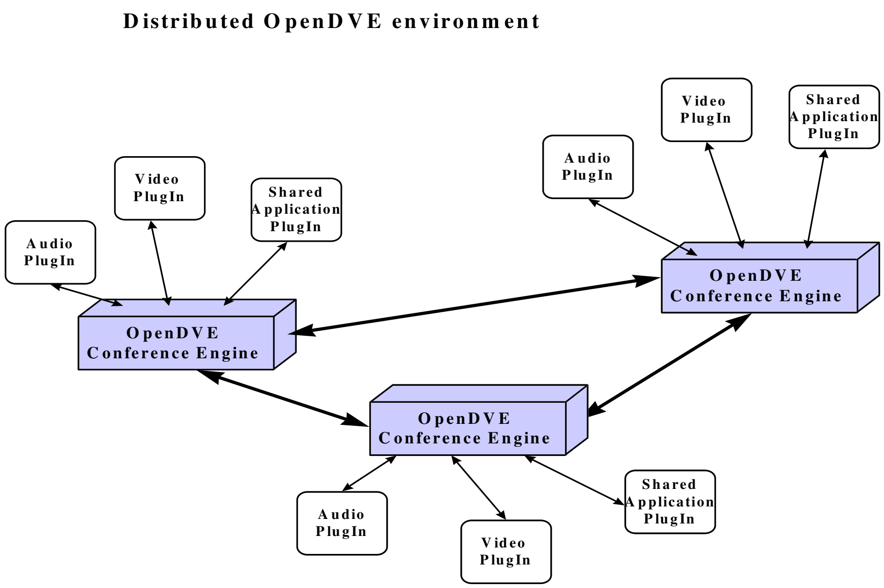
## A.Architecture of OpenDVE

Basically, OpenDVE environment is compliant to recent recommendations concerning multimedia collaborative environments. Its operability is based on the following elements:

- session management: teleconference members join or leave a given session
- event distribution: some actions require event distribution to all teleconference members
- data distribution: different types of data need to be distributed among participants

Graphical representation of the OpenDVE environment architecture is presented in the Figure 4.2-1

Figure 4.2-1 Architecture of distributed OpenDVE environment [1]




OpenDVE provides all services mentioned above through so called **Conference Engine** modules. On each end-system using OpenDVE technology there must be an OpenDVE Conference Engine module running. This module registers itself (through the **Registrar** module) with other ones running in the network and provides a full intercommunication service via OpenDVE library calls. The Registrar module provides network identification for each Conference Engine running on the system. Conference Engine module acts as a conference manager and a psychical network control mechanism. While transparent to the user, Conference Engine serves as a traffic coordinator, maintains data structures, keeps track of the users membership. Upon request it distributes data and events to all, or some of the participants. Conference Engines communicate with each other using TCP(control data) and UDP(user data) network protocols. A single Conference Engine creates a high-level network interface layer, above the actual network's transport layer (in OSI Network Architecture Model). The Conference Engine's internal interface that creates network independence to the API developer, is the **Network Abstraction Layer** (NAL). This proprietary layer allows efficient use of the underlying network (Ethernet, ATM, ISDN, etc.) while protecting the developer from the complexities of network programming. This solution is fully compliant with T.123 protocol stacks recommendation (see section 2.1 in this document).

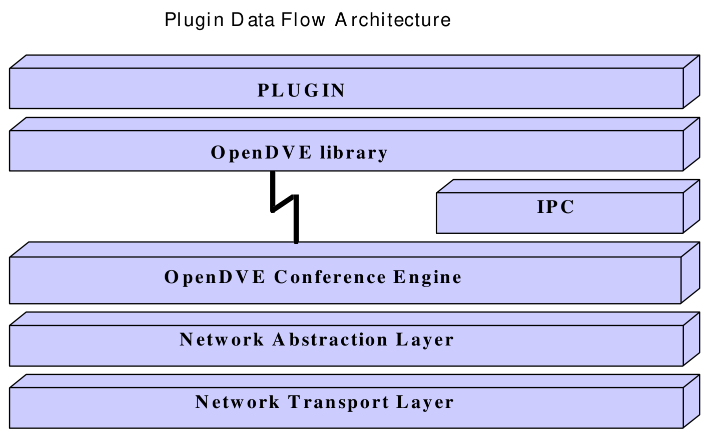
Other important element of the OpenDVE architecture are so called *plugins*. An OpenDVE **plugin** can be any application that is attached to the Conference Engine and uses its services. Each plugin is connected to the Conference Engine through an independent Inter-Process Control (IPC) mechanism. Plugins of the same type communicate among themselves using the services and connections provided by the conference

Engine.

Plugins provide the user-interface with special functions making up the substance of a given collaborative application and are responsible for the actual data processing as well as presenting the conference to the user. A plugin communicates with a Conference Engine via domain sockets (UNIX environment).

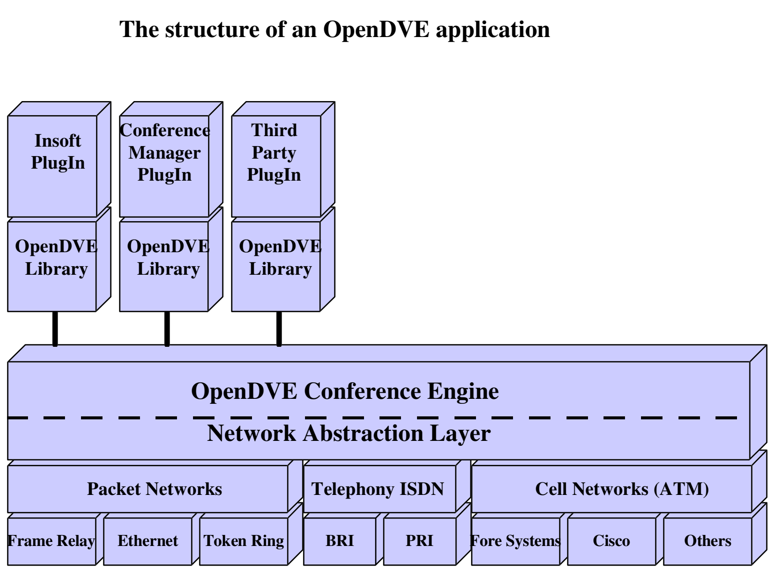
A data flow between an OpenDVE plugin and the network is shown in the Figure 4.2-2

Figure 4.2-2 An OpenDVE plugin data flow architecture [1]




A single application can be built from multiple plugins which fulfills a modular approach paradigm. An architecture of example OpenDVE application is presented in the Figure 4.2-3

Figure 4.2-3 Architecture of an OpenDVE application [1]




Communication between a single plugin and the Conference Engine occurs via so called OpenDVE **ports** which are integer numbers (see paragraph 4.4). Each plugin has its own unique OpenDVE port number that distinguishes it from the others. Some OpenDVE port number values has already been reserved for InSoft's commercial products, some of them are assigned to the OpenDVE environment functions such as session management (port number 1), plugins management (port number 5) etc. A Plugin can use a number of OpenDVE ports at the same time provided that there are no collisions between them and other plugins connected to the Conference Engine.

For more details about OpenDVE architecture, please refer to [1] and [2].

## *A.Session management in OpenDVE*

A collaborative application requires an extensive session management mechanism to be able to distribute data and conference events among participants of a given teleconference.

In OpenDVE environment the entire conference management is built on the base of the core component of the OpenDVE architecture, Conference Engine. It maintains the dynamically updated database containing information which identifies all teleconference participants, list of plugins in use etc.
In this section we present how a conference is managed in the OpenDVE environment. We have to remark, that there has been no consistent and detailed documentation on this issue provided by InSoft. Therefore, the presented information is based on our own experience gained during the research work.

To actually manage a conference, a *conference manager* plugin application has to be created on the basis of OpenDVE. This particular plugin orchestrates other available plugins, which are in charge of capturing and presentation different kinds of data to the user.

Conference Manager Plugin communicates with Conference Engine module via OpenDVE library routines and performs all tasks concerned with session management:

- session initialization
- membership management (invitation, joining given conference, leaving given conference)
- plugins management: identification, start/terminate commands
- session termination

Each separate plugin can obtain current information on the teleconference status, membership etc. through the appropriate calls from the OpenDVE library.

Session management mechanism in OpenDVE is based on event distribution among participants. In OpenDVE session management a user can generate and is notified about the following OpenDVE *events*[3]:

- **EngineGreeting** - this event is sent when a plugin has successfully connected to the Conference Engine.
- **EngineRejected** - this event is sent whet the Conference Engine rejected plugin's connection request. This usually means that there is already another OpenDVE plugin connected to the Conference Engine on the same port.
- **EngineClosed** - this event is sent when the Conference Engine closes the connection to a given plugin, or a plugin closes its OpenDVE session.
- **EngineShutdown** - this event is sent when the Conference Engine module terminates
- **State** - this event is sent when the Conference Engine transitions to a new state. This, for example, can notify a plugin when a given teleconference has terminated or started
- **Starting** - this event is sent when the Conference Engine starts or when a plugin first connect to the Conference Engine. This event carries an additional data holding information on conference members and the current teleconference state
- **NewMember** - this event is sent when a new user is added to the conference (after the invitation has been sent) but does not mean that he/she is already active
- **MemberLeft** - this event is sent when a user has left the conference
- **MemberAccepted** - this event is sent when a new user has accepted an invitation to the conference.

After reception of this event one can communicate with that user.

- **MemberRejected** - this event is sent when a user has declined to join the conference
- **LocalMemberInfo** - this event is sent when the local user's membership information or Do Not Disturb flag changes (see DND Changed event)
- **DNDChanged** - this event is sent when the local user's Do Not Disturb flag changes

Each event notification and/or generation can provide some additional event-specific information which in turn can be interpreted by a given plugin and appropriate action can be undertaken. All events concerned with session management in OpenDVE environment are distributed on OpenDVE port number 1.

One of the important issues is that a user can not join the conference unless he/she is invited by someone who is already active in the conference. This approach prevents from unexpected users listening to the conference and is compliant to T.120 protocol series by ITU-T (see section 2.1 of this document)

## *A.Data distribution within OpenDVE environment*

In OpenDVE environment a data is being distributed upon the requests from conference participants sent to the Conference Engine module.

The Conference Engine module distributes all data in the same manner without any regard for a type of processed data since the OpenDVE environment provides the means of delivering data between plugins, but it does not specify the data format (see below). A data to be distributed is generated by a plugin application (see paragraph 4.2) which interacts with a user, captures digital data (audio/video), presents incoming digital data to a user etc.

Interaction between the OpenDVE Conference Engine and a plugin is conducted in the context of the **OpenDVE session**. Each OpenDVE session is identified by a *handle* returned when the OpenDVE session is opened using the OpenDVE library function (dveOpenSession()). The handle is valid until the session is closed by dveCloseSession() call. The valid session handle is meaningless for other conference participants and even other local plugins. The OpenDVE session is also identified by the **OpenDVE port** (see paragraph 4.2) specified when the session is opened. This parameter can be used to identify both local and remote OpenDVE sessions. It is also used to multiplex traffic to and from different plugins via Conference Engine.

A plugin may conduct multiple simultaneous OpenDVE sessions (each with a unique port and handle) during its execution.

InSoft Inc. reserves all OpenDVE port numbers in the range from 0 to 32767 for its own use. Every developed plugin can be registered with InSoft Inc. in order to prevent any possible collisions with other plugins (developed by third parties).

The plugin's message data format is referred to as its **protocol**. The protocol **id** is typically the same as the original plugin's OpenDVE port number. In order to receive and process incoming data a plugin should establish a **user callback** routine for a particular protocol id. A plugin is free to define its own protocol (it resides in the *Application* layer of the OSI model) to use to exchange data with remote plugins. OpenDVE development platform provides a routine (dveSendDataPacket()) to send any type of data in a given protocol from one plugin to the other participants in a given teleconference.

A plugin's protocol definition includes:

- the plugin's OpenDVE port number that will be used as a protocol id
- a set of tags used to identify custom messages
- a definition of the messages arguments (if any)

After having its protocol defined, a plugin can communicate transparently with remote ones by sending data messages with indicated OpenDVE port number for a given protocol.

The data distribution scheme in OpenDVE environment is convenient in many ways for a plugin developer but it has also some very unpleasant limitations.

Firstly, the transport layer of the underlying network is entirely hidden from a plugin application. Therefore, a single plugin can not much influence the way its data is being distributed. This concerns the following issues:

- delays introduced by data sent/receive operations
- direct accessibility of different network interfaces (Ethernet, ATM, ISDN)
- data packetization
- use of uni- and multicast protocols

Despite of what InSoft Inc. claims in their marketing information, we were not able to determine how to deal with mentioned issues inside our proprietary plugin applications. Having some details retrieved from InSoft's products configuration files we could assume that one can only determine the network interface to use by the Conference Engine by setting users' network addresses as appropriate before initiating the conference.
As for now we did not find any possibility to influence data encapsulation, its packetization nor the delays in transmit/receive operations. This in turn complicates new transport protocols (such as RTP or RSVP) use on OpenDVE development platform. Having user data encapsulated by the Conference Engine, some protocol-specific information becomes invisible in the network routers which in turn disables a given protocol functionality (for example in RSVP protocol). It seems to us that this problem can be solved only by implementing separate (additional) network connectivity between the conference users (by Conference Engine bypassing or access to the network layer API in OpenDVE exposed by InSoft).

For more details on user data distribution, please refer to [1] or [2].

## *A.Software Developers' Kit for OpenDVE*

InSoft Inc. provides the **OpenDVE Software Developers' Kit** which is supposed to fulfill the needs of collaborative applications developers. The package contains several libraries along with header files and few simple exemplary plugins which expose the principles of programming in the OpenDVE Development Environment.

As it was mentioned above, an OpenDVE application consists of the Conference Engine and at least one plugin. The application interface in the OpenDVE library is the OpenDVE API. The interface allows the application programmer to include Conference Engine services in the customized application, thereby creating a new OpenDVE plugin. This in turn allows to create a completely original collaborative application from scratch.

Since the Conference Engine handles all of the complexities of data distribution and session management, the programmer who uses the OpenDVE API is free to concentrate on adding a special functionality, instead of low-level details of conferencing and network support.

The OpenDVE API includes the following documented set of libraries:

- **session management** - contains routines for OpenDVE session initialization, registration with Window Manager, standard input processing
- **event handling** - contains routines which allow to set, add and remove user callbacks invoked by incoming OpenDVE events and user data
- **plugin messages** - contains routines allowing to prepare and send data in a given protocol
- **member information** - contains routines which allows to obtain information on local and remote users (their network addresses, network interface type etc.)
- **locating plugins** - contains routines allowing remote plugins scanning in order to determine whether a

remote user receives our data or not

- **conference management** - contains routines for conference management such as: sending invitations, accepting/rejecting incoming invitations, leaving conference etc.
- **memory management** - contains routines allowing memory allocation and freeing in OpenDVE environment
- **user interface tools** - contains routines helping in user interface building (esp. for conference manager where the information on participants is presented)

In spite of encountered lack of documentation we managed to understand OpenDVE SDK enough to deploy OpenDVE technology in our Web-integrated collaboratory system. However, a lot of work had to be done and a lot of time has been spent on analyzing and understanding functionality of OpenDVE library routines.

For more detailed description of OpenDVE programming, please refer to [2] or [3]

### *A.Relation of OpenDVE to LiveMedia framework*

In January 1996 InSoft Inc. has been integrated into Netscape Communications Corporation. Along with the **LiveMedia framework** announcement, Jim Barksdale, president and CEO of Netscape, confirmed that LiveMedia will be based partially on InSoft's OpenDVE architecture: "This will extend existing Netscape's software platform into a collaborative, real-time Internet communications".

Therefore, programming in OpenDVE is perceived as a unique experience which can be of a great value when LiveMedia framework appears on a market. We believe, this will allow to develop easily and quickly a collaborative application based on the state-of-the-art technologies integrated seamlessly in future WWW browsers. As for now there is no signs of LiveMedia appearance on the market in the nearest future and of course there is always a risk that it will never happen, or that OpenDVE technology will not be deployed in this framework.

For more details on LiveMedia framework, please refer to section 5.4.4 of this document.

# I.World Wide Web technology and Internet browsers

The *World Wide Web* (WWW) was invented by Tim Berners-Lee at CERN Physics Laboratory in Switzerland in 1989. The National Center for Supercomputing Applications (NCSA) of the University of Illinois later produced a browser client for the Web called *Mosaic*. Mosaic revolutionized the use and acceptance of Internet by the non-academic population all over the world. The growth of the World Wide Web has been phenomenal so far and it is still growing.

The architecture of the World Wide Web as it exists today is based on a *client/server* model. A client/server model in WWW context is one in which the server becomes a repository of information, and the client is a vehicle to view and access that repository. An information provider can create documents as files and store them on a server, which is also connected to the Internet or the internal company LAN. Any potential consumer of the information on the Internet can request and view those documents using a client or browser software such as Mosaic or Netscape browser.

In the World Wide Web, information is delivered to clients only when they request delivery - the servers themselves can not decide to deliver to a client. This model is similar to the newspaper or magazine model, where one goes to the store to view the newspaper or magazine, reads it, and later buys it.

Development of the HTML language and HTTP protocol (see section 5.4.5 in this document) even increased the growth of popularity of the WWW in the entire world.

There has been also a sophisticated WWW browser client application developed (for example Netscape browser and Microsoft's Internet Explorer) which exposes an attractive user interface and make it extremely easy to navigate through the World Wide Web in order to retrieve any kind of information in databases all over the world. They operate using HTTP protocol and can interpret HTML language. Recent versions of these applications support many other attractive technologies such as Java and VRML making the entire World Wide Web environment much more sophisticated and rich in different advanced data-to-user presentation techniques.

In the following sections the role of WWW technology in the collaborative multimedia applications development, its constraints and limitations as well as some new enabling technologies are discussed.

## A.Availability of WWW in the today's world

Popularity and accessibility of the World Wide Web technology have been increasing steadily and very rapidly for the last years. Internet has reached almost every place on the planet Earth, and so did the Web. In today's world the majority of companies, enterprises, universities, schools and institutions as well as millions of individual users are connected to the Internet and have therefore an access to all its resources. Obviously, the number of WWW users varies from country to country but, in general, one can certainly say that the Internet and World Wide Web are everywhere. Such situation implied a growing interest in using the World Wide Web as a convenient environment for collaboration, teleconferencing and multimedia data distribution. The Internet brings a complete, mature and widely spread computer network infrastructure on which these applications can be implemented and be accessible for the majority of the population without any significant additional costs.

All of these make the Internet and the World Wide Web two major technologies in collaborative multimedia applications development.

## A.Network resources

Network resources in the Internet consist basically of the nodes called *routers*. A router is a node that accepts incoming data and forwards it to the other routers in the Internet according to the addressing information in data packets. Routers compose a *mesh* structure which is spanned all over the world and connects thousands of small computer subnets. A subnet is connected to a router via a link characterized by its *capacity* in terms of the maximum amount of data that can be transmitted over this link within certain period of time. This link's capacity is also referred to as a *bandwidth*. Once a subnet is connected to an Internet router, it gains an access to all other subnets connected to the Internet. Communication between end-systems from any separate computer subnets becomes possible.

Two end-systems in the Internet exchange data on a basis of IP (Internet Protocol) protocol which is basically in charge of the data path determination and data delivery from the source to the destination. IP protocol resides in the network layer of the OSI network model and it operates on a basis of the *IP addresses*. Each end-system connected to the Internet has its own unique IP address that distinguishes it from all others users connected to the Internet.

On top of the IP protocol, different transport protocols can be used. The most widely used in today's Internet are TCP/IP and UDP protocols. TCP/IP protocol assures an ordered data delivery while when UDP is employed, data packets can be disordered.

We observe a rapidly growing data traffic in the Internet along with the needs for additional bandwidth. None of the existing transport protocols for the Internet provides a quality-of-service guarantee. They all operate in "best-effort" mode which means that no guarantees concerning delivery time, delays, jitter and so on can be made. Therefore, in case of the network congestion the delivery of data can suffer from large unexpected delays, or in critical cases can be even impossible.

"Best-effort" approach was efficient, simple to implement and fair with low-bit rate and non real-time communications. Since for the last years we observe growing interest in and needs for collaborative real-time multimedia applications, such functionality of the transport protocols became to be rather insufficient.

These were the reasons for an intensive research work focused on development of new enabling technologies which could turn the Internet into a modern information highway before a new infrastructure based on different technologies (such as ATM) will be created.

For a description of some of the new concepts and products, please refer to section 5.4 in this document.

## *A.Needs for multimedia data distribution in Internet*

The term *multimedia* refers to different types of data being transmitted and processed simultaneously. With today state-of-the-art in technology, people are not any more satisfied with simple communication tools such as a telephone, facsimile, electronic mail, etc.

New advanced technologies in consumer electronics, desktop computer systems, audio/video processing make it possible to communicate with the use of several different media at the same time. This in turn raised new critical requirements:

- creation of new transport protocols, well suited to the real-time multimedia transfer
- development of new compression schemes, mainly for video data
- creation of techniques to manage multiparticipant sessions

These challenging issues were focused on during the last years and the results of those efforts are presented in section 5.4 of this document.

## *A.New enabling technologies and their relation to OpenDVE*

In order to enable multimedia teleconferencing on today's Internet there has been done a lot of research projects during past few years. Researchers focused on transport protocols' improvement, interoperability issues and integration of the existing technologies. Some very important results have been obtained and new technologies started to be used in the state-of-the-art collaboratory applications designated for Internet environment. In this section we discuss some of them, which we believe to be the most important from the point of view of the teleconferencing application development

### 1.Real-Time Transport Protocol (RTP)

Development of the **Real-Time Transport Protocol** (RTP) is perceived as a milestone in the overall research activity in the field of the real-time collaborative, multiuser, multimedia applications for Internet environment. The following paragraphs describe functionality, features and possible applications of the RTP protocol.

Real-Time Transport Protocol provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, timestamping and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality. However RTP may be used with other suitable underlying network or transport protocols. RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network. RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, without necessarily decoding packets in sequence.

While RTP is primarily designed to satisfy the needs of multiparticipant multimedia teleconferences, it is not limited to that particular category of applications. Storage of continuos data, interactive distributed simulation, active badge, control and measurement applications may also find RTP applicable.

RTP consists of two closely-linked parts:

- real-time data-carrying part of the RTP
- the RTP control protocol (RTCP) to monitor the quality of service and to convey information about the participants in an on-going session.

RTP presents a new style of protocol following the principles of application level framing and integrated processing proposed by Clark and Tennenhouse. That is, RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer. RTP is a protocol framework that is deliberately not complete and is intended to be tailored through modifications and/or additions to the headers as needed. Therefore, a complete specification of RTP for a particular application requires some additional documents:

- a profile specification document which defines a set of payload type codes and their mapping to payload formats
- payload format specification documents which defines how a particular payload, such as an audio or a video data, is to be carried in RTP packets.

Some of needed for multimedia applications profiles (audio, video) has been already defined and their description can be found in [11].

Possible applications of the RTP protocol include:

- simple multicast audio conference
- audio and video conference
- and many others configurations for collaboration

RTP is also designed to cope with heterogeneous networks in terms of bandwidth and terminal equipment. To adapt a given data stream to different capabilities of receivers so called "mixers" and "translators" are used.

Having monitoring data available through the RTCP operation, an application based on RTP can adapt itself to the conditions in a network and react efficiently to any kind of traffic congestion, unexpected delays etc. (for details on this issue, please refer to [13]).

For more details on RTP protocol, please refer to [10].

## 1.Resource ReSerVation Protocol (RSVP)

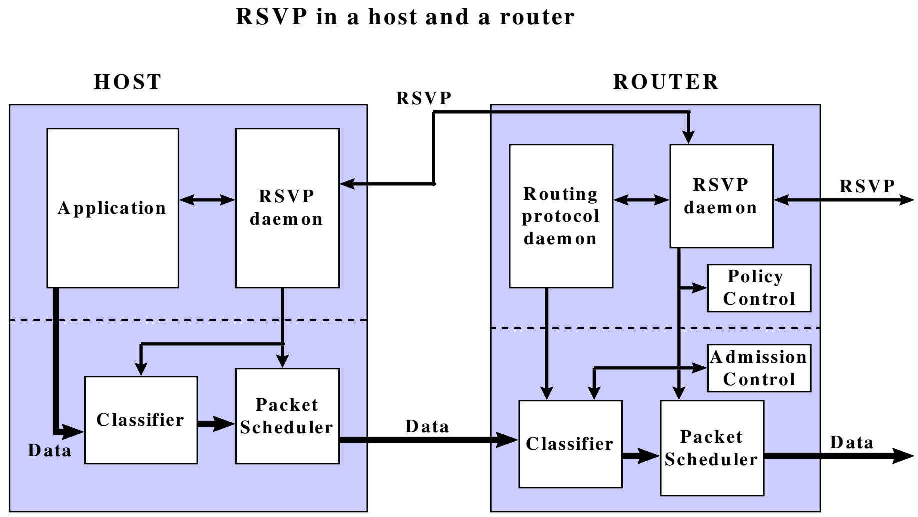
Another milestone in the real-time multimedia collaborative applications development was the specification of RSVP, which stands for **Resource ReSerVation Protocol**. This protocol was intentionally specified for the Internet which lacks of any *quality of service* (QoS) provision capabilities. The issue of QoS is critical for all real-time multimedia collaboratory applications designed for Internet and therefore the RSVP significance is relatively large.

In the following paragraphs the features and functionality of RSVP are described.

The RSVP protocol is used by a host, on behalf of an application data stream, to request a specific quality of service from the network. The RSVP protocol is also used by routers to deliver QoS requests to all nodes along the path(s) of the data stream and to establish and maintain state to provide the requested service. RSVP requests will generally, although not necessarily, result in resources being reserved along the data path.

RSVP requests resources for simplex data streams, i.e. it requests resources in only one direction. Therefore, RSVP treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and a receiver at the same time. RSVP operates on top of IP (either IPv4 or IPv6), occupying the place of a transport protocol in the protocol stack. However, RSVP does not transport application data but is rather an Internet control protocol, like ICMP (Internet Control Message Protocol), IGMP (Internet Group Management Protocol) or routing protocols. Like the implementations of routing and management protocols, an implementation of RSVP will typically execute in the background, not in the data forwarding path as shown in the Figure 5.4.2-1

Figure 5.4.2-1 RSVP host and router [12]




RSVP is not itself a routing protocol, it's designed to operate with current and future unicast and multicast routing protocols. An RSVP daemon consults the local routing database(s) to obtain routes. In the multicast case, for example, a host sends IGMP messages to join a multicast group and then sends RSVP messages to reserve resources along the delivery path(s) of that group. Routing protocols determine where packets are forwarded, RSVP is only concerned with the QoS of those packets that are forwarded in accordance with routing.

Each node that is capable of resource reservation passes incoming data packets through a *packet classifier*, which determines the route and the QoS class for each packet. For each outgoing interface, a *packet scheduler* then makes forwarding decisions for each packet to achieve the promised QoS on the particular link-layer medium used by that interface.

In order to efficiently accommodate heterogeneous receivers and dynamic group membership, RSVP makes receivers responsible for requesting QoS. A QoS request, which typically originates from a receiver host application, is passed to the local RSVP implementation shown as a *daemon process* in the Figure 5.4.2-1. The RSVP protocol then carries the request to all the nodes (routers and hosts) along the reverse data path(s) to the data source(s).

At each node, the RSVP daemon communicates with two local decision modules, *admission control* and *policy control*. Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permissions to make the reservation. If both checks succeed, the RSVP daemon sets parameters in the packet classifier and scheduler to obtain the desired QoS. If either check fails, the RSVP daemon returns an error notification to the application process that originated the request.

RSVP is designed to scale well for very large multicast groups. Since both the membership of a large group and the topology of large multicast trees are likely to change with time, the RSVP design assumes that router state for traffic control will be built and destroyed incrementally. For this purpose, RSVP uses "soft state" in the routers. That is, RSVP sends periodic refresh messages t maintain the state along the reserved path(s). In absence of refreshes, the state will automatically time out and be deleted.

RSVP provides several reservation models or styles:

- Wildcard-Filter WF) Style (shared link, all upstream senders)
- Fixed-Filter (FF) Style (distinct reservations for all explicitly chosen upstream senders)
- Shared-Explicit (SE) Style (shared link, explicitly chosen several upstream senders)
- Detailed description of those styles of reservation in RSVP along with examples can be found in [12].

RSVP operates transparently through those routers that do not support it. In this case the usefulness of the RSVP can be heavily impaired by a single bottleneck (a router which does not support QoS) in the entire data path, which is obviously of a critical importance.

For more detailed description of RSVP, please refer to [12]

## 1.MBone - multicast backbone network

One of the most widely successful and robust pieces of Internet technology existing today is the *Virtual Multicast Backbone On the interNEt*, otherwise known as the MBone. This technology forms the backbone of the Internet for distributing real-time multimedia information to millions of computer desktops worldwide. Computer networks today are more important than ever - and with the arrival of fast multimedia desktop computers, coupled with computer networks, some users believe that the "cable-ization" (as in cable TV) of the computer networks and the computer itself is not too far away.

The MBone is a technology that enables distribution of and access to real-time interactive multimedia on the Internet. Distributing such *isochronous* media (i.e. requiring delivery from source to destination within a certain bounded time) in a large-scale manner over packet network such as the Internet was not feasible before the MBone was developed. Typical examples of isochronous media types are real-time voice and video. The MBone enables delivery of such media on the Internet. Network scalability of multimedia data distribution technologies is very important, especially when large-scale community use of resources is involved.

The MBone itself consists of a subset of Internet routers that understand the Internet class D addressed packets and their routing. The class D Internet addresses (first byte value between 224 and 239) are used for Internet-wide IP multicasting. A small subset of this class D address space (all IP addresses matching 224.2.*.*) has been set aside for multimedia conferencing over the Internet and this subset constitutes the MBone. Multicasting of packets prevents the network from being congested by multiple copies of the same data packets, which constitutes the major advantage over a traditional unicasting scheme. A multicast address is no tied to any specific physical network interface at a certain site. This is logical group address that is dynamic in nature - it exists as long as there is any Internet site interested in sanding or receiving multicast data.

A site can become a member of a certain multicast group by sending an IGMP join message. Only the sites that have joined a multicast group are able to send and receive data addressed to this particular group. However, a site does not have to join a group to transmit data to this group.

In order to prevent multicast data to travel too far from the sender site (and thus generating useless traffic in the network) each multicast address has a *Time To Live* (TTL) value associated with it. TTL value specifies how far, in terms of number of IP router hops, the multicast traffic should o on the Internet.

The MBone has been appreciated all over the world and the number of MBone subnets is growing very rapidly. Plenty of software tools allowing to manage and monitor MBone sessions have been developed. Currently, the MBone is used for hosting public events, workshops, seminars, classroom courses, meetings, a few news and entertainment events.

Potential uses for the MBone are numerous:

- television (basic, pay-per-view)

- interactive games
- video on demand
- home shopping
- banking and financial transaction services
- digital audio
- electronically delivered newspapers and magazines
- long distance education

It is worth mentioning that the MBone infrastructure provides duplex connections that, as opposed to today's cable TV installations, allows a user to interact with service provider entities.

For more detailed information on MBone, please refer to [14]

## 1.LiveMedia framework for Internet collaborative multimedia applications

In January 1996 Netscape Communications Corporation announced Netscape **LiveMedia**, a standards-based framework for bringing real-time audio and video to the Netscape open software platform. As the cornerstone of its new framework, Netscape also announced the signing of a definitive agreement to acquire InSoft, Inc., a leader in network-based communications and collaborative multimedia software for the enterprise. InSoft's applications include Communique! for desktop collaboration and videoconferencing, InSoft Network Television or INTV! for distributed digital video, and CoolTalk and CoolView for Internet audio, video and data communications on Windows, Windows95 and UNIX based platforms. Netscape plans to use InSoft's technology to create the Netscape LiveMedia framework, which it plans to make a standard component in future Netscape clients, servers and tools. Netscape LiveMedia framework will enable users to have easier access to new Internet applications such as audio and video on-demand, real-time video conferencing and Internet telephony. In addition, eleven technology-leading companies announced plans to support Netscape LiveMedia, which is based on open standards and interfaces that will enable Netscape and third-party real-time audio and video products to interoperate. As for now, the companies that announced LiveMedia support are: Progressive Networks, Adobe Systems, Digital Equipment Corp., Macromedia, NetSpeak, OnLive!, Precept, Silicon Graphics, Inc., VDOnet, VocalTec, Xing, Adaptive Media Technologies, Apple Computer, @ONCE, Farallon Communications, Four 11, IBM, RADVision, SoftCom, Starlight Networks, OZ Interactive, Vivo Software and Winnov.

The Netscape LiveMedia framework will be based on the Internet Real-time Transport Protocol (RTP), RFC number 1889, and other open audio and video standards such as MPEG, H.261 and GSM to enable products from these and other companies to work together seamlessly, providing users with a range of real-time audio and video capabilities on the Internet. Netscape will publish the LiveMedia framework on the Internet, openly license key technology components of it, and work with the Internet standards bodies to facilitate the adoption of this technology as a formal Internet standard. By integrating InSoft's technology within their existing software architectures, Netscape will be able to extend its software platform into a foundation for real-time Internet and Intranet communications. The Internet is perceived as a strategic component of Netscape's open, enterprise approach to collaborative multimedia InSoft's OpenDVE Collaborative Multimedia Framework enables cross-platform collaboration between PCs and UNIX workstations, transparent connectivity between a wide range of networks, and interoperability between a range of video offerings. Netscape plans to integrate InSoft's products in two phases. In the first phase, the combined companies will develop the LiveMedia framework and continue to promote InSoft's OpenDVE software architecture and development toolkits; Communique!, INTV!, CoolTalk and CoolView applications, and the Global Conference telecommunications gateway. In the second phase later in 1996, Netscape plans to integrate InSoft's real-time audio and video capabilities into future versions of Netscape Navigator and Netscape servers. In addition, Netscape expects that third-party developers will offer a wide variety of add-on audio and video products based on the LiveMedia framework. Netscape also signed an agreement with Voxware, Inc., to license that company's compression/decompression (codec) technology to plug into the Netscape LiveMedia framework. Codecs from other companies can also be incorporated into the extensible framework. This technology will help to provide widespread, reliable voice delivery to any user with a Web connection. "Enabling users to communicate and work together from any location -- and in real time -- extends the Internet's power and reach to a new level," said Marc Andreessen, vice president of technology at Netscape. "The Netscape LiveMedia framework will make a number of real-time audio and video applications available to Internet users. InSoft and Voxware are important to Netscape's strategy to deliver products based on LiveMedia, and we will work with the technology-leading companies who are supporting this interoperable approach to make real-time audio and video applications an immediate reality."

Netscape LiveMedia represents a commitment to open Internet-supported protocols and content definitions that transcend any single vendor, including Netscape. Vendors adopting the LiveMedia Internet standards will be able to offer products that interoperate with other LiveMedia compatible products, making it easy for users to take advantage of real-time audio and video capabilities on the Internet.

For more details on LiveMedia Framework, please refer to Netscape Communications Corp. Web home pages.

## 1.The Internet Application Framework

The **Internet Application Framework** is a comprehensive set of open protocols, standards, technologies and application programmers' interfaces (APIs) for building and deploying applications for the worldwide Internet or internal corporate networks (intranets). We describe in the following paragraphs the components of the framework, its key features and benefits, some sample applications, and future directions.

The Internet's exponential growth has created new exciting opportunities for application developers both inside and outside the corporation. Indeed, a new category of application is emerging. These new applications are network-centric, media rich, and tailored to the end user; these applications enable meaningful collaboration between people across the worldwide Internet.

Corporate developers are already building these applications to make their organizations more effective in communication and collaboration, including:

- Multimedia electronic mail, discussion groups, etc. that allow teams to share the latest plans, design documents, and project schedules and so on
- Electronically published marketing information
- Electronic forms that automate all manner of business communication

Developers both inside and outside corporations are also creating applications that deliver product and service information to customers and partners in an engaging, interactive, and customized way, such as:

- Electronically published catalogs and on-line order forms that enable customers to view and order the latest products and services for delivery either instantly (for soft goods such as publications, software, and stock market data) or by mail. In short, the World Wide Web is an inexpensive new distribution channel for goods and services of all kinds.
- General purpose productivity tools including spreadsheets, graphing packages, and presentation software are emerging rapidly, offering users small, optimized tools whose files can be instantly published and shared worldwide. File formats cease being a barrier for sharing rich data.
- Multimedia titles created exclusively for the World Wide Web are growing as client software becomes increasingly adept as displaying media ranging from pictures to movies, audio clips to 3D worlds.

The frenetic growth of the Internet and these Internet-based applications owe a large part to open Internet standards such as TCP/IP, HTTP, HTML, CGI, and SSL. As protocols grow richer and new protocols, languages, and APIs come into existence (such as Java, JavaScript, and VRML), a need grows for defining an open platform standard that encompasses these standards. We discuss below that open standard called the **Internet Application Framework**.

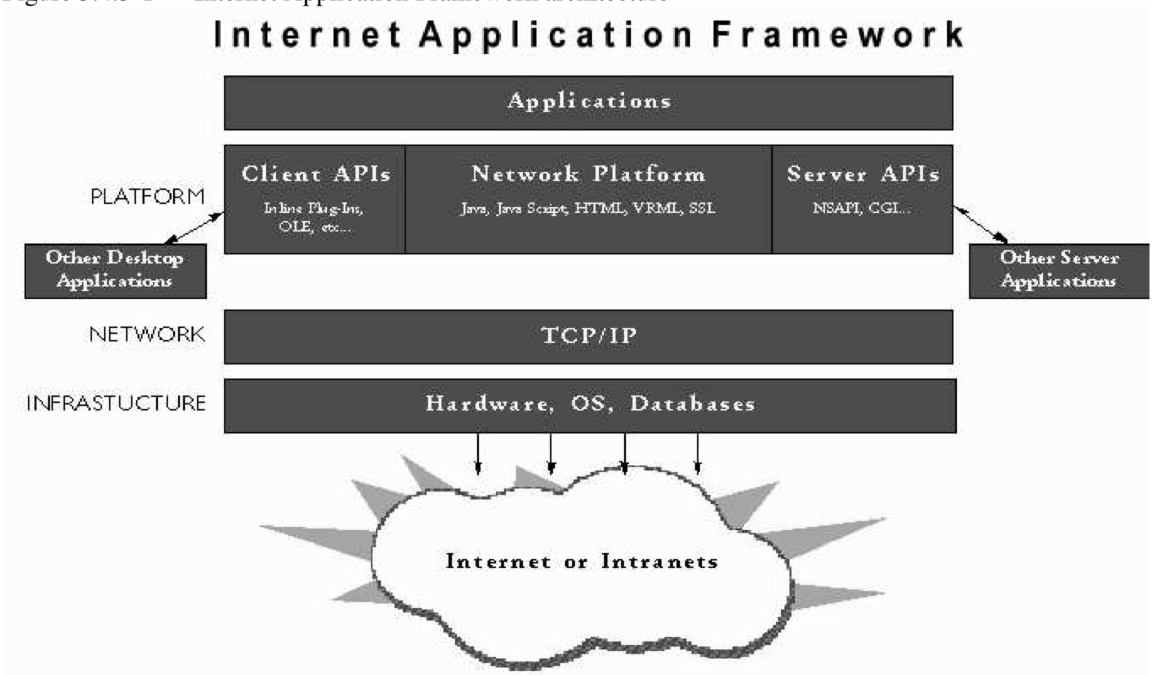
The Internet Application Framework is a comprehensive set of technologies - including protocols such as HTTP and SSL, languages and APIs such as Java and JavaScript, and architectures such as LiveMedia - for building and deploying these new applications for the Internet or for internal corporate intranets. The

Internet Application Framework has three components:

- Network Platform
- Client APIs
- Sever APIs

The architecture of IAF is illustrated in the Figure 5.4.5-1

Figure 5.4.5-1 Internet Application Framework architecture




The **Network Platform** is a collection of cross-platform, open standards that enable developers to quickly create and deploy network-centric applications. From our point of view, as developers of Web-integrated multimedia collaboratory system, the following components of the Network Platform seem to be important:

- **Java**, a complete programming language that allows true platform-independent application development. It was developed by Sun Microsystems and has been submitted to the open standards process. The Java White Paper offers this concise description of its characteristics: a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multi-threaded, and dynamic language. Of particular significance to developers of network applications is the ability of a Java application, called an "applet", to be securely downloaded from any node in a network to any other. The application can then be loaded dynamically and executed immediately. It is simple to place references to Java applets into HTML or VRML documents. This means that HTML documents can be enriched with virtually unlimited dynamic content that may have originated anywhere in the network. The enormous interest in Java as an emerging standard is due in part to the fact that Java was designed with network applications in mind. It provides solutions to a number of issues - platform independence, location independence, versioning, security and multi-threading, to name just a few - that are crucial to effective networked application architectures, yet difficult to address in other languages. For this reason, Java appears likely to become the dominant language for network-centric applications. We used Java language to implement LiveConnect feature in our video-player Netscape plug-in, which is described in section 7.3 in this document.
- **JavaScript**, a scripting language that allows dynamic behavior to be specified entirely within HTML documents. It is derived from Java, but is considerably simpler. This means that HTML authors who are not professional programmers will be able to incorporate a rich variety of dynamic behaviors into their documents without the need to write separate support programs such as CGI scripts or Java

applets. A simple, but enormously useful capability illustrates the power of JavaScript. JavaScript provides access to a substantial portion of the power of Java itself. JavaScript plays an especially significant role on the server side of Internet Application Framework applications. The Internet Application Framework includes JavaScript facilities to provide high-performance native access to standard relational databases such as Oracle, Informix and Sybase. We used JavaScript capabilities to implement LiveConnect feature in our video-player Netscape plug-in, which is described in section 7.3 in this document.

- **HTML** (*HyperText Markup Language*), is enormously well known and widely used. It is an international standard whose specification is maintained by the Internet Engineering Task Force (IETF). HTML is one of the fundamental elements underlying the World Wide Web and has, with remarkable rapidity, gained acceptance within corporate intranets as well. It has a position of comparable prominence in the Internet Application Framework. An HTML document is the fundamental means by which users interact with Network Applications. It is the vehicle through which an application designer can provide the user with access to resources available anywhere on the network. It has the flexibility to allow the presentation of content ranging from static text to multimedia presentations, and the invocation of applications of virtually any purpose or complexity
- **HTTP**, (*HyperText Transport Protocol*), is the standard protocol for communications between clients and servers on the World Wide Web. It plays the same fundamental role in the Internet Application Framework. HTTP is a stateless protocol which specifies how a client and server establish a connection, how the client requests a specific service from the server, how the server issues a response, and how the connection is terminated. Indeed, the terms "client" and "server" in the Internet Application Framework are defined primarily in terms of their roles in an HTTP interaction. HTTP connections over the Internet are implemented using the TCP/IP protocol. Secure connections, necessary in applications supporting commercial or otherwise sensitive transactions, are implemented in the Internet Application Framework by interposing the Secure Sockets Layer (SSL) between the HTTP and TCP/IP layers.
- **LiveMedia**, an open architecture proposed by Netscape Communications for creating interoperable real-time multimedia data services over the Internet. LiveMedia is an open architecture for handling real-time data such as audio and media over the Internet (see section 5.4.4 in this document).
- **LiveConnect**, proposed by Netscape, *LiveConnect* lets you integrate Java, JavaScript and Navigator plug-ins. This allows you to create plug-ins that can be controlled by Java and JavaScript, or use Java and JavaScript to implement portions of your plug-in. Fundamentally, Netscape LiveConnect allows you to: call Java methods from plug-ins, call native methods implemented in plug-ins from Java, call Java methods from JavaScript, call JavaScript from Java methods (see also paragraph 7.3 in this document)
- **Database Access APIs,** JavaScript defines objects that allow developers to access databases. Netscape is also working with leading database vendors to define standard ways for efficiently accessing cross-platform databases such as ODBC and Sun's emerging Java Database Connectivity (JDBC)technology.
- **Internet protocols**, the Internet's exponential growth has been fueled by a collection of fundamental protocols and standards such as TCP/IP, SMTP, POP3, NNTP, FTP, IRC, Telnet, and MIME. These protocols form the foundation of the Internet Application Framework. New protocols are constantly being proposed such as the Real-Time Transport Protocol (RTP) (see paragraph 5.4.1 above in this section) and the Lightweight Directory Access Protocol (LDAP), and the Internet Application Framework will incorporate these emerging protocols as products and services become available to support them.

The **Client-Specific APIs** allow developers to extend the capabilities of the client. The technologies in this component of the Internet Application Framework include:

- **Inline Plug-in APIs** provide a means of incorporating dynamically loadable modules into the client process itself. Together with Java and JavaScript facilities, Plug-ins enable high-performance application delivery of performance-intensive rich content, such as sound, graphics and video.
- **Netscape Client APIs** allow developers to write applications on native operating system platforms such as Windows, MacOS, and UNIX that can communicate and remotely control a client application such as Netscape Navigator. These APIs allow applications to communicate via Apple Events, OLE Automation, DDE (Dynamic Data Exchange), and X Events.

The Internet Application Framework is comprised of open standards that are neutral with respect to hardware architecture, operating system and windowing system.
The open nature of these standards allows any vendor to write a product embodying the standard, giving the customer a rich marketplace of products, tools, and services from which to choose. These open standards have been, and will continue to be, central to the Internet's explosive growth. Another advantage of open standards is that customers can deploy applications incrementally.
From a developer's point of view, open standards translate into platform freedom. Writing applications on top of the Internet Application Framework allows developers to write their applications with a variety of development tools and to deploy those applications on a wide variety of hardware platforms, operating systems, and databases (cross-platform).
As technologies emerge in response to new needs and opportunities, the Internet Application Framework will continue to incorporate open standards that create opportunities for a wide variety of vendors and application developers.
Network applications based on the Internet Application Framework are being successfully deployed by organizations ranging from individual publishers, small businesses and corporate work groups to multi-national corporations, universities and governments.
Furthermore, applications based on the Internet Application Framework scale from the LAN to the Internet. Whereas the 1980s saw network and application standards such as TCP/IP and Simple Mail Transport Protocol (SMTP) unify an organization's internal infrastructure, the 1990s see a new layer of standards such as the Internet Application Framework unify computing infrastructures *across* organizations. This unification around open standards will enable workers to collaborate with their colleagues, partners, and customers in rich new ways. The Internet Application Framework makes applications easy to deploy for a variety of reasons:

- A single, universal client program can display data from a myriad of different servers that may speak different protocols and deliver documents in a variety of format. Because the Internet Application Framework encompasses open Internet standards, a client has ready access to HTML documents or Java applets via HTTP, file downloading via FTP, mail access using SMTP or POP, news access using NNTP, 3D landscapes via VRML documents, live video via LiveMedia documents, and many other kinds of network resources. This universal client eliminates many of the hidden costs of client software licenses associated with most of today's client-server applications.
- The universal client viewer lets users interact with all Internet Application Framework applications in a consistent way, reducing support and training costs.
- Network-centric applications are stored centrally on a server and are downloaded to the client on demand. This eliminates support headaches of maintaining multiple versions of the same application.
- Help documents written in standard languages like HTML allow users on different platforms to view help documents.

Applications written on the Internet Application Framework are typically media-rich, deliver dynamic data, and support communication and collaboration. These applications can incorporate such elements as sound, graphics, animation, video and virtual reality explorations:

- Media-rich applications allow users to share text, sounds, pictures, movies, 3D landscapes, and other rich data. The explosive growth of HTML owes much of its success to its ability to define rich, multimedia, linked documents
- Dynamic data is tailored by a particular user. That data might be a stock portfolio with up-to-date prices, an electronic catalog of goods (presented, of course, in the user's chosen language), or the current weather at an upcoming vacation destination. File formats woes disappear; if a client does not have the appropriate software to allow the user to interact with the data being delivered, the software accompanies the data in the form of a Java or JavaScript applet.
- Communication and collaboration form the foundation of the Internet Application Framework. Applications of all sorts may be written to enhance the communication and collaboration of people around the world

The last months have seen unprecedented opportunity, growth, and excitement in the Internet marketplace.

The Internet Application Framework will accelerate the growth of this market by providing an open foundation for developing dynamic, multimedia and collaborative applications:

- Enterprise Communication (for example collaborative applications, such as CAD systems)
- Electronic Commerce (marketing, sales over Internet etc.)
- General purpose applications (teleconferencing etc.)

For more details on Internet Application Framework, please refer to Netscape Communications Corp. WWW home pages.

# I.Integration of a WWW browser with collaboratory environment

Growing needs for collaboratory systems create a serious challenge for all researchers dealing with multimedia and Web technologies. Because of its wide-spread availability, the WWW is perceived to be an ideal environment for collaborative systems. Naturally, for the convenience of potential customers we would like to have a multimedia collaboratory system which is able to operate on today's Internet (in terms of transport bandwidth) and be integrated with the Web. One possible solution is to develop a collaboratory system which will encompass advanced data compression schemes and therefore generate reasonable traffic so that it could be transferred with the use of Internet resources. Other issue is that it could also be integrated with a Web browser making itself be easy to use.

In the next section we discuss a concept of plug-in application which enabled true integration of customized applications with existing Netscape Web browser. Then requirements' specification for our collaboratory system and its architecture are presented, implemented features are described, encountered limitations are defined and finally, possible further extensions are proposed.

## *A.Plug-in applications for Internet browsers*

One of the major extensions to the Netscape WWW client browser was the **Netscape Plug-In Application Programmers' Interface**. The concept of a plug-in application was introduced which extended significantly the capabilities of a WWW browser.

Before this concept was developed, a client browser could handle only two families of types of data:

- a data that could be interpreted (understood) properly by a browser itself (graphics, text)
- a data that could be passed to a stand-alone application according to its MIME type and established by a user associations of a kind: MIME-type - application

Furthermore, a data that was to be presented to the user had to be first entirely downloaded from the Web. This imposed a serious inconvenience when any video or audio sequences were to be downloaded. For a file of several megabytes it could take a long time to download or be even impossible to do so.

With the idea of a **plug-in**, those limitations appear to be overcome. Firstly, a plug-in can handle any type of data (according to its functionality), which allows to handle many more different kinds of digital data retrieved from the Web. Virtually, any kind of incoming data can be handled provided there is an appropriate plug-in installed. The advantage of a plug-in over a stand-alone application is such, that a plug-in can run inside a browser's window making the entire Web interface more user-friendly. But the major advantage of a plug-in we observed is that the date to be presented to the user does not have to be downloaded before it can be presented. This is especially very convenient feature when video sequences are being retrieved from Web.

We found the Netscape plug-ins capabilities very useful from the point of view of multimedia data retrieval from Web. With a Netscape plug-in functionality we are able to decode video data on-time, i.e., every single video frame is decoded and displayed as soon as it arrives from the network. In this way all unpleasant delays for data download are completely eliminated.

To create a Netscape plug-in, one has to use Netscape Plug-In API, which is freely available on Netscape Communications Corp. Web home pages. A plug-in application is a shared-objects library containing all mandatory and plug-in specific routines. These mandatory routines are divided into two groups:

- **Netscape methods** (*NPN_*()*), used by plug-in to request some services from Netscape browser (data read, data send, new URL etc.)
- **plug-in methods** (*NPP_*()*), called by Netscape browser, should be implemented within a plug-in's source code

*Plug-in methods* are called by Netscape browser when it starts and when certain events occur . By plug-in methods, a plug-in application can do the following:

- register itself with a certain MIME type
- initialize all its private variables
- build its user interface inside a browser's window (a plug-in's interface can occupy the entire browser's window or its part)
- register its interface to Java and JavaScript (new feature in LiveConnect, see section 5.4.5 in this document)

A plug-in is started by Netscape browser when a data with registered MIME type is requested by a user click on a URL. Therefore, one can register as many plug-ins as it is needed and de facto all kinds of data can be handled.

As for now, many companies engaged in creation of their own Netscape plug-ins to handle different kinds of data available on Web. Many plug-ins that handle video, audio, different image formats and 3D graphics have been implemented since Netscape Plug-In API became available.

A Netscape plug-in can be placed within any HTML document. It is loaded when Netscape browser detects an URL with the appropriate MIME type in a document to download. To run a plug-in within an HTML document one have to use the *EMBED* tag in the HTML language:
<EMBED src="http://naos.npac.syr.edu/jabu/heads.263" width=500 height=400 PARAM1=*value* PARAM2=*value* …>
where:
*src* indicates an URL of the file to be downloaded and handled by a plug-in, *width* and *height* specify the size of a plug-in window within the browser's window, *PARAM#* are the plug-in -specific parameters. Inside *EMBED* statement there can be any plug-in -specific parameter specified. Their names and values will be all presented to the plug-in by Netscape browser when loading the plug-in.

To make a plug-in use the entire browser's window, an URL indicating to required data file has to be opened in a browser (using for example *open location* command) or a *HREF* tag has to be used within an HTML document as in the following example:
<A HREF="http://naos.npac.syr.edu:1972/jabu/heads.263"> </A>

A plug-in can also run in the background which can be specified by *mode* parameter in *EMBED* statement.

There exists two modes of data provision to a Netscape plug-in:

- **client-pull**, in this mode each portion of data is explicitly demanded by a client (Netscape plug-in). There exists the *NPN_RequestRead()* Netscape method in Netscape Plug-in API [4] which is supposed to allow a client plug-in to ask for particular piece of data from the remote server. Generally, in this mode delays incorporated by client-server communication are more significant and therefore this mode is not advised for multimedia real-time data streams. NPN_RequestRead() routine did not seem to function properly as for time of our research.
- **server-push**, in this mode a server attempts to provide a client with next portion of data at any possible moment of time. There exist two plug-in methods that are called automatically by a browser whenever a plug-in goes into *idle* mode (no operations are performed):

- NPP_WriteReady(), which should return a number of bytes that a plug-in is able to accept
- NPP_Write(), which provides a plug-in with data retrieved from the network

Both routines must be implemented within a plug-in source code.

The actual plug-in functionality is implemented in its private routines which can be called from within *plug-in* or *netscape* methods provided in Netscape Plug-In API.

For our collaboratory system integration with Web browser the Netscape Plug-In technology was found to be fundamental. We decided to implement our collaboratory system as a set of separate pieces of software:

- conference manager based on OpenDVE technology (Netscape plug-in)
- several separate OpenDVE plugins to handle different kinds of data in a conference

Since Netscape Communications Corporation announced LiveMedia framework (see section 5.4.4 in this document) and decided to build its collaboratory features with the use of OpenDVE technology, we believe our system to be very advanced and up to date with all what we can expect in Web technology in the nearest future.

To better understand how to implement a Netscape plug-in application, please refer to the listing of routines that have to be implemented within a plug-in source code provided in Annex C at the end of this document, or to the sample plug-in source code provided with Netscape Plug-In SDK (can be downloaded from Netscape Communications Corp. Web home pages).

## A.Requirements specification for collaboratory plug-in

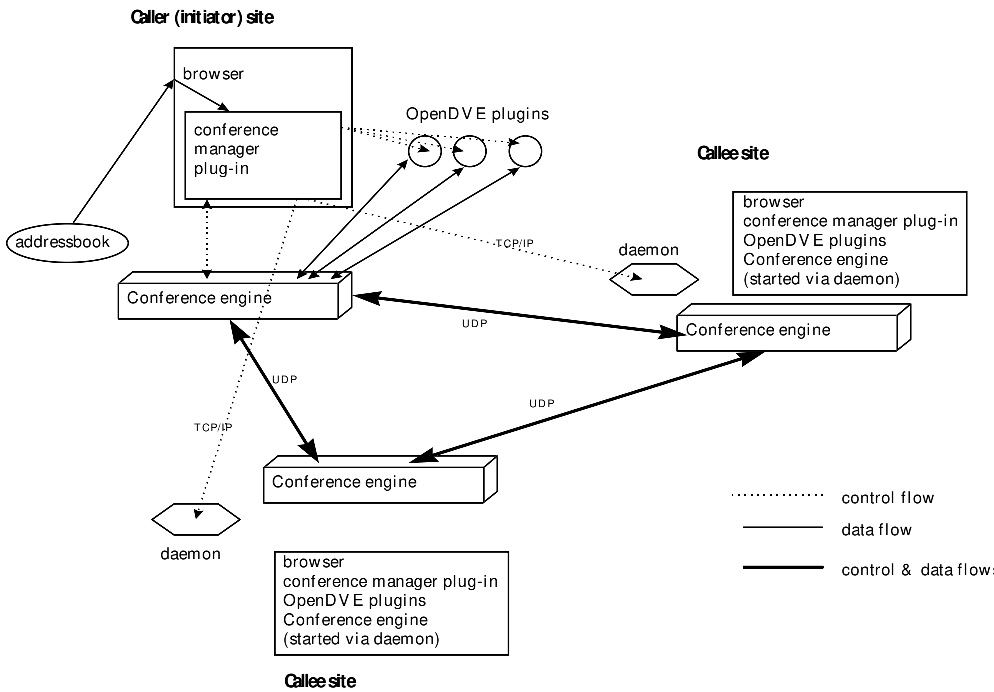
Being aware of the present state-of-the-art in Web technology we defined the requirements for our collaboratory system as the following:

- it has to be integrated with Netscape Web browser, this should allow to start and manage a teleconference directly from within a browser
- it has to be able to establish a teleconference, invite new members and discontinue a teleconference
- it has to manage OpenDVE plugins which will distribute among participants all needed kinds of digital data
- the OpenDVE plugins must be independent from each other and be able to communicate with the use of Internet (limited resources and no quality of service guarantee)

## A.Architecture of the Web-integrated collaboratory system

As it was mentioned above, the collaboratory system we implemented was to be based on OpenDVE technology and act as a Netscape plug-in. Therefore, we designed its architecture in a manner which is presented in the Figure 6.3-1.

Figure 6.3-1 Architecture of the Web-integrated collaboratory system




The presented scheme contains all major system components and explains all dependencies between them. The architecture of our multimedia collaboratory system is quite complicated and comprises of many separate components interacting with each other:

- **OpenDVE Conference Engines**, these modules provide the entire set of session management services and are in charge of user data distribution. The functionality of the Conference Engine module is described in the section 4.2 in this document
- **OpenDVE plugins**, these are in charge of the user data of different kind distribution. As for now, our plugins handle video, audio, graphics but also third-party plugins can be used (for example, those from InSoft's Communique!). For more details on implemented OpenDVE plugins, please refer to section 6.5 in this document
- **daemon** applications, these are in charge of the software components start on remote end-systems (OpenDVE registrar, OpenDVE conference engine and Netscape browser)
- **Netscape browser**, this module acts as a conferencing panel for all conference participants (it starts a conference manager Netscape plug-in)
- **conference manager** Netscape plug-in, this module is the "heart" of the entire system. It manages a teleconference and all available OpenDVE plugins
- **addressbook files**, these files contain a list of conference participants, and indication of the local user status in the conference (*conveyor* or participant)

Presented scheme also explains data and control flows within the entire collaboratory system. Control flow for OpenDVE plugins management occurs between *conference manager* and *OpenDVE plugins* and also for conference management between *conference manager* and *Conference Engine*. All Conference Engines communicate with each other to distribute either control or user data.
According to documentation provided by InSoft Inc., control information (OpenDVE events) is distributed with use of TCP/IP protocol to ensure the reliability. User data is distributed with use of UDP protocol which is more convenient when large amounts of bytes are generated (especially for video data).

The assumed architecture is entirely distributed across the network. There is no central entity coordinating a session and data distribution. This is the paradigm of OpenDVE environment. All participating entities communicate via several conference engines. Such architecture allows for multicast communication between participants which in turn is crucial for all multimedia applications for Web environment.

The other paradigm here is the *daemon* application we implemented and included in our system. A single *daemon*, running on each end-system of our collaboratory environment, listens to a TCP/IP socket and waits until a request in form of URL string arrives from a network. Upon receiving a request, daemon starts on the local host the Conference Engine and Netscape browser with received URL. This automatically activates the conference manager Netscape plug-in inside started Netscape browser. A user receives therefore notification that someone in the network wants to establish a teleconference with him. A conference initiator should always send to remote sites the URL to a *default addressbook file* containing his own network address (see also below).
This mechanism is very similar to a simple phone call, where a party is called to communicate with caller. Use of *daemon* applications allows to establish a teleconference at any point of time without any special announcements provided the called user is logged in and the *daemon* is running in the background.
We implemented a very simple and basic daemon functionality which should be extended to provide some additional features such as backward notification about successful/unsuccessful remote software start. This was not implemented since our daemon's functionality is to be replaced (and extended) by a system designed by Konrad Olszewski (for details, please refer to [21]).

An *addressbook file* idea was created to make a teleconference establishment as easy as possible. An addressbook file is used to provide the conference manager plug-in started on conference initiator (conveyor) site with other teleconference participant's network addresses when a given teleconference is to be established. This is done without any interaction with user and allows to create a kind of notebook with different groups of addresses in separate files. Such *notebook* could be a simple HTML document containing the URLs of the files with addresses.

A single *addressbook file* has to be prepared in the following format:

#initiator_tag#
#number_of_addresses#
#participant_1#
.
.
.
#participant_n"

where:
*intitiator_tag* determines whether a user downloading this file will be a conference conveyor
*number_of_participants* determines a number of participants to be invited (without the initiator)
*participant_1* to *participant_n* are the Internet addresses of the participants in the form of:
login_name@host_name

This approach allows to have single copy of all addressbook files stored in one place in the network and to access them by all members in the system via appropriate URLs.

The *default addressbook file* has the same structure, but should contain a local user's network address and the *initator_tag* set to zero since it is loaded by a remote site when invitation to the conference comes from outside. The contents of this file can be used to provide remote users with coordinates of the conference initiator. Therefore, each user of our system has to create his/her private default file and always send its URL to the remote sites when initiating a teleconference.

## A.Limitations imposed by OpenDVE environment

In course of the collaboratory system architecture design we encountered a number of limitations imposed by OpenDVE technology.

These limitations are mainly due to the fact that we can not directly access the network resources (it is covered by the Network Abstraction Layer of the OpenDVE Conference Engine) in the OpenDVE environment. Therefore, we are not able to react efficiently to different kinds of traffic disturbances nor we can influent the delays in data transfer. We have to rely entirely on the OpenDVE data distribution scheme (see section 4.4 in this document). From our position this was not a very comfortable situation, but having in mind that LiveMedia framework is to be built on the OpenDVE technology and will also use all the newest protocols RTP, RSVP, multicast) we can assume that this part of our system is provided by the lower layers. For conference management these limitations are not crucial since the conference events' distribution does not require much bandwidth and is implemented in TCP/IP protocol to be reliable.

Other issue we were aware of, was a proper synchronization of the remote software start with the beginning of a teleconference. This was primarily to be solved by more sophisticated daemon applications use but then was left to be provided by the system to replace our simple daemons (see [21]).

## *A.Implemented features*

In our multimedia collaboratory system we made a distinction between a *conference conveyor* and a *conference participant*. A conference conveyor is a person who starts given teleconference and therefore invites other people to participate in it. A conference participant is a person who has been invited to join given teleconference. Some of the implemented features are only available to conference conveyor.

The overall user interface for a conference manager Netscape plug-in is presented in the annex A (Figure A-1) at the end of this document. It consists of several buttons and two scrolled windows containing a list of available OpenDVE plugins and a list of active members in the teleconference.

The implemented features of our collaboratory system are the following:

- conference manager starts when a document (addressbook file) with a MIME-type of *application/x-opendve* is requested by a user
- content of the addressbook file is loaded by browser and presented to the conference manager plug-in (server-push mode, see section 6.1 in this document)
- collaboratory environment is being prepared on the remote site(s) by daemons. Daemons are provided through a TCP connection with a URL pointing to a default addressbook file (required to start a Netscape plug-in)
- all participants mentioned in an addressbook file downloaded by the conference initiator can be invited to the teleconference by pressing the "Invite" button on conference manager's interface.
- upon reception of Member Accepted or Member Rejected OpenDVE events (see section 4.3 in this document) a list of active members is created and displayed in a scrolled window of the conference manager's interface
- all available OpenDVE plugins are identified and their names are put on the list which is displayed in a scrolled window of the conference manager's interface. All OpenDVE plugins to be used by the collaborative system need to be listed in a special configuration file pointed by *DVE_PLUG_CFG* UNIX environment variable. Sample content of such file is presented in annex B to this document.
- a new participant can be invited by the conference conveyor, this is done by clicking on the "AddMem" button and typing the new participant's Internet address inside a prompt box.
- OpenDVE plugins can be started or terminated after a choice is made from the plugins' list and the "Plugin Start" or "Plugin Stop" button is pressed. Some additional information about a chosen plugin can be obtained by pressing the "Plugin Info" button.
- when the conference conveyor starts a plugin, on all remote sites the same plugin is started. In similar way plugin termination occurs.
- there is no limits on a number of available OpenDVE plugins since these are a completely independent modules.
- when a user leaves the teleconference all remaining participants are notified about it and her or his name disappears from the list of active participants
- when the conference conveyor leaves the teleconference, all participants are notified and advised to terminate their conference managers by exiting from a Web browser. This is a condition to be able to "see" next incoming invitation.
- every participant can start any available OpenDVE plugin but this not implies that the same plugin will

be started on remote sites.

- data distribution occurs via separate OpenDVE plugins for each type of data
- our system is fully interoperable with InSoft's Communique! providing some new features which can be useful, such us remote OpenDVE plugin start and stop etc.

As for now we implemented three OpenDVE plugins that we used in the collaboratory system evaluation:

- **audioconference** plugin: (implemented by Janusz Bulawa and Tomasz Stachowiak) provides multipoint audioconference capability, transmitted audio data can be encoded either in GSM (8 kHz, around 13 kbps) or ADPCM (8 kHz, 32 kbps) format (user's choice), format of incoming audio data is automatically detected, incoming audio streams are mixed in the output audio device, G series codecs to be implemented soon, echo cancellation not yet implemented
- **whiteboard** plugin: (implemented by Janusz Bulawa and Tomasz Stachowiak), provides multipoint whiteboard sharing capability, users' actions on local whiteboards are distributed to all conference participants
- **videoconference** plugin: (implemented by Janusz Bulawa), provides multipoint videoconference capability, video data is H.263 encoded which results in a very low bit rate and good picture quality, pictures are in QCIF format, local monitor capability included, H.261 video codec to be included soon

Evaluation of the functionality has shown a reliable and efficient session management in OpenDVE environment, and also a relatively good performance of implemented OpenDVE plugins. Especially, video transmission did not suffer from insufficient bandwidth with the rate of 10 frames per second (pictures were coded and transmitted in QCIF format and extrapolated on display to CIF format). Here we have to mention that a framerate of 10 fps were obtained with a transmission of already encoded sequences, since we were not able to encode this video sequence in-time with H.263 encoder (see sections 7.2 and 7.4 in this document).

With in-time H.263 encoding (pictures were grabbed from a desktop camera) we reached a framerate of 3 frames per second on the slowest SGI machines. This could be increased to 10 or even 15 fps on the more powerful CPUs.
During the experiments with video data distribution we noticed also, that in case of high CPU load on the low end-system, OpenDVE Conference Engine does not call the *user callback* for plugin's protocol immediately after a user data arrives. This introduces enormous delays which in particular can reach several seconds or even minutes for video data (!). We think the reason is that Window Manager's events in UNIX which notify a plugin about incoming data are handled without high enough priority causing them to be queued when CPU is heavily loaded. This problem does not exist when more powerful CPUs (for example, SGI's processor R4600 with 200 MHz clock) are used.

Current version of OpenDVE video plugin was our first implementation of a multipoint videoconferencing tool and it needs a lot of simple improvements to become an acceptable tool for real videoconferencing.
To make the implementation of those improvements be easy we modified source code of H.263 codec so that its picture encoding/decoding routines can be extended to incorporate other video coding schemes (such as H.261). Also video grabbing routine can be easily extended to handle different video inputs (camera, VCR etc.).
Unfortunately, we need to incorporate public domain codecs since, in spite of what InSoft Inc. claims in their marketing materials, OpenDVE SDK does not provide any access to video/audio codecs. They are implemented within OpenDVE environment and used by InSoft's products but can not be directly accessed from third party OpenDVE plugins.
This was one of the reasons for modification of H.263 software codec in order to extend it to support different coding schemes (for example H.261) so that we could used them later on in our teleconferencing tools (see also section 7.4 in this document).

When implementing our audio OpenDVE plugin for collaboratory system we struggled with a serious problem concerning audio data stream continuity. Along with H.263 video coding we could not manage to obtain acceptable audio quality on the receiving end-systems. The reason for this lies in the complex video encoding algorithm which consumes the most of CPU time. On low end-systems we were not able at all to

use video and audio tools simultaneously. The idle time of CPU between consecutive video frames encodings was so short that Conference Engine was not able to activate plugin's callback responsible for audio data reception. The received audio was full of breaks, unpleasant noises, etc. We did some modifications to audio frame grabbing and encoding and we deployed two audio coding formats: GSM and ADPCM. The GSM format gives better compression (13 kbps) but lower quality and is more computationally complex. The ADPCM is a simple algorithm that gives better quality and higher bit rate (32 kbps). The size of single audio packets was also adjusted to provide the best possible quality of service. Afterwards, we tested our video and audio plugins on high end-systems (more powerful CPUs) and we reached several video frames per second along with good audio quality. The observed delays were acceptable and the overall quality of service was good enough for teleconferencing.
We were not forced to reduce generated audio bandwidth as much as possible since video data has been H.263 encoded and the overall bandwidth of even one ISDN link (64 kbps) is sufficient for both audio and video conferencing with use of our system.
Furthermore, preliminary experiments with our audio and video plugins running within our collaboratory system has shown their better subjective performance when compared to well-known commercial products such as InPerson from SGI.

### *A.Relation to current teleconferencing standards*

The components of our collaboratory system were supposed to be fully compliant with major standards concerning collaboratory and teleconferencing.

The compliance with H.320 recommendation [16] has been reached by:

- **format of coded pictures** (QCIF)
- **framerate** of the video tool (too low on low-end systems but can reach more then 7.5 fps on powerful CPUs or after some additional and not yet implemented modifications of the H.263 codec source code)
- **video coding scheme** (very advanced H.263 coding algorithm, both encoder and decoder motion compensation capability)
- **audio coding scheme** (G series codecs easy to deploy as a separate audio plugins or integrated into one tool)
- **data rate** (much less then expected 128 kbps, actually of about 50-60 kbps for both audio ADPCM and video H.263)

As far as T.120 series recommendations [17] are concerned, our system has the following features:

- **Network Abstraction Layer** is implemented within the OpenDVE Conference Engine (see section 4.2 in this document), which is compliant with T.123 Protocol Stacks for Audiographic and Audiovisual conferencing recommendation (see section 2.1 in this document)
- **mutlipoint communication** is available through services provided by OpenDVE Conference Engine, which is compliant with T.125 Mutlipoint Communication Service recommendation
- **session management** is provided by OpenDVE Conference Engine (see section 4.3 in this document), which is compliant with T.124 Generic Conference Control recommendation.

Additionally, our collaborative system can be easily extended to support T.126 *Still Image Transfer* protocol and T.127 *Multipoint Binary File Transfer* protocol which can be supported optionally by a conferencing system (see section 2.1 in this document). These extensions require two additional simple OpenDVE plugins to be implemented.

According to what we presented above, the collaboratory system for Web that we designed and implemented is fully compliant to major current teleconferencing standards and therefore has the advantage on many commercial systems, if not to mention all systems for the WWW environment (see sections 3.1 and 3.2 in this document).

### *A.Possible further extensions*

The Web-integrated collaboratory system has its basic functionality implemented and some simple conferencing tools are already functional. Anyway, there exists a lot of possibilities to make this system more attractive and sophisticated. Also many extensions to the current OpenDVE plugins set would be of a great value.

In our opinion the following improvements could be done to make our collaboratory system much more attractive:

- implementation of more sophisticated remote software start mechanism, this is believed to be provided to our system by an integration system designed by Konrad Olszewski (for details, please refer to [21])
- implementation of new OpenDVE plugins to make the entire system more flexible
- possibly also RSVP protocol could be included to the system, this should be done separate RSVP-specific connections among all participants that will overpass the Conference Engines
- teleconference log (events and/or user data) in a database can easily be implemented. Object-DataBase Connectivity (ODBC) or Java-DataBase Connectivity (JDBC) along with LiveConnect technology can be used (see sections 5.4.5 and 7.3 in this document).
- integration with the existing Java based distributed application, this can be done with use of LiveConnect technology that we tested successfully with our H.263 video player Netscape plug-in (see section 7.3 in this document)
- implementation of some useful features such as *notebook* feature

These are the major extension that can be easily introduced in our collaboratory system. Further ideas are found to be dependent on the particular application of this system.

# I.Adaptation of H.263 video coding standard for a collaboratory systems in Internet/WWW environment

Because of the numerous attractive features of the H.263 video coding scheme, we decided to focus on this algorithm in order to deploy it in our collaboratory system.
In following section the basics of H.263 coding algorithm are described. Then we discuss briefly the features of H.263 codec in software provided by Telenor, Norway. Next, our Netscape H.263 video player plug-in that was our first step to deploy H.263 in Web-integrated collaborative system is presented Further, we present how we adopted H.263 coding scheme to fulfill our needs in terms of coding speed and possible use over the Internet. Finally, the last section contains a short list of suggested extensions to our Netscape plug-in.

## *A.Main features of H.263 coding scheme*

The basic configuration of the H.263 video source coding algorithm is based on the ITU-T recommendation H.261 and is a hybrid of inter-picture prediction to utilize temporal redundancy and transform coding of the remaining signal to reduce spatial redundancy. The source coder can operate on five standardized **picture formats**: sub-QCIF(128x96), QCIF(176x144), CIF(352x288), 4CIF(704x576) and 16CIF(1408x1152).

The decoder has **motion compensation** capability, allowing optional incorporation of this technique in the coder. **Half pixel precision** can be used for the motion compensation, as opposed to Recommendation H.261 where full pixel precision and a loopfilter are used. Variable length coding (VLC) is used for the symbols to be transmitted. In addition to the basic video source coding algorithm, four negotiable coding options are included for improved performance:

- **Unrestricted Motion Vectors**, in this optional mode motion vectors are allowed to point outside the picture. The edge pixels are used as prediction for the "non existing" pixels. With this mode a significant gain is achieved if there is movement across the edges of the picture, especially for the smaller picture formats
- **Syntax-based Arithmetic Coding**, in this mode the arithmetic coding is used instead of variable length

coding. The SNR and reconstructed frames are the same, but significantly fewer bits is produced

- **Advanced Prediction**, in this optional mode overlapped block motion compensation is used for the luminance part of P-pictures. Four 8x8 vectors instead of one 16x16 vector are used for some of the macroblocks in the picture. The encoder has to decide which type of vectors to use. This results in less blocking artifacts giving a subjective gain in quality
- **PB-frames**, a PB-frame consist of two pictures being coded as one unit: one P-picture which is predicted from the last decoded P-picture and one B-picture which is predicted from both the last decoded P-picture and the P-picture currently being decoded. With this coding option, the picture rate can be increased considerably without increasing the bitrate much.

All these options can be used together or separately except for the Advanced Prediction mode which requires the Unrestricted Motion Vector mode to be used at the same time.

The **video bit rate is variable** and no constraints on it are given.

The pictures are coded as **luminance and two color difference components** (Y, Cb and Cr). These components and the codes representing their sampled values are as defined in CCIR Recommendation 601: black=16, white=235, zero color difference=128, peak color difference=16 and 240. For each of the picture formats, color difference samples as sited such that their block boundaries coincide with luminance block boundaries.

Each picture is divided into *groups of blocks* (GOBs), a group of blocks (GOB) comprises of k*16 lines, depending on the picture format (k=1 for sub-QCIF, QCIF and CIF; k=2 for 4CIF and k=4 for 16CIF). Each GOB is divided into *macroblocks*. A macroblock relates to 16 pixels by 16 lines of Y and the spatially corresponding 8 pixels by 8 lines of Cb and Cr.

The prediction is inter-picture and may be augmented by motion compensation. The coding mode is called INTER when prediction is applied or INTRA when no prediction is used. This can be signaled at the picture level (I-picture for INTRA, P-picture for INTER) or at the macroblock level in P-pictures.

Motion compensation refers to bi-directional *motion vectors* which determine a relative position of the current macroblock versus the corresponding macroblock in the previous frame. Motion estimated block is not encoded entirely but only the difference value is coded and transmitted.

Encoded blocks are DCT-transformed and resulting coefficients are subsequently quantized. Finally, the VLC or SAC is applied.

There are several facilities provided to support switched multipoint operation:

- *freeze picture request*, causes the decoder to freeze its displayed picture until a freeze picture release signal is received or a time-out period of at least six seconds has expired. The transmission of this signal is by external means (for example ITU-T Recommendation H.245)
- *fast update request*, causes the encoder to encode its next picture in INTRA mode with coding parameters such as to avoid buffer overflow. The transmission of this signal is by external means (for example ITU-T Recommendation H.245)
- *freeze picture release*, a signal from an encoder which has responded to a fast update request and allows a decoder to exit from its freeze picture mode and display decoded picture in the normal manned. This signal is transmitted in the picture header of the first picture coded in response to the fast update request.
- *Continuos Presence Multipoint* (CPM), in a multipoint connection, a Multipoint Control Unit (MCU) can assemble two to four bitstreams into one video bitstream, so that at the receiver up to four different video signals can be displayed at the same time. In Continuos Presence Multipoint mode four independent video bitstreams are transmitted in four logical channels in one H.263 video bitstream. Capability exchange for this mode is done by external means (for example ITU-T Recommendation H.242). Logical channel indicators shall be transmitted in the picture and GOB headers of each of the H.263 logical channels.

Described features of the H.263 make it a very attractive video coding scheme as far as the picture quality and generated bit rate are concerned (for experiment results, please refer to [6]). It seems to be a very convenient solution to limited network bandwidth in today's Internet. This made us concentrate on H.263 video coding when designing and implementing our Web-integrated collaborative system.

For more detailed information on H.263 coding, please refer to [9]

## A.Software H.263 codec's features

In order to fulfill our goals from the section 3.3, we decided to find and use the public domain H.263 codec in our collaboratory system. The source code of the H.263 codec was obtained from the public ftp site of the Telenor Co., Norway. This code was written by Karl O.Lillevold and is available as a public domain software.

The very first step was to analyze and understand the architecture of the encoder and decoder. The source files were separated for encoder and decoder parts. According to the notes, the codec is compliant with H.263 Recommendation and has all negotiable coding options implemented. It also provides statistics about generated bit stream such as number of generated bits, Signal-to-Noise Ratio of the encoded frames, etc.

Only first frame is INTRA encoded, every next picture can be encoded only as P or PB frame. This obviously reduces the generated bit rate but is very inconvenient when any data loss occurs during transmission in the network. Every P or PB frame is decoded on the basis of the previous frame (see section 7.1 in this document) and if any frame in the video sequence is lost, the rest of it can not be decoded properly which is unacceptable for our collaborative system for Web, where data losses can often occur. This issue has been taken care of and undertaken counter measures are described in section 7.4 in this document.

The motion estimation is based on *full search* which is very CPU time consuming but gives a very good picture quality.

A problem raises with the encoding speed. The original version of the H.263 encoder provided by Telenor, Norway was so slow that it seemed to be useless for any real-time applications. We decided therefore to try to increase the encoding speed by the source code modifications (see section 7.4 in this document).

## A.Video player plug-in for the Netscape's WWW browser

As the very first step to deploy H.263 video coding scheme in Web we decided to implement a video player plug-in for Netscape Web browser that will handle a video data coded with use of this algorithm.

The basic requirements for a video player plug-in were the following:

- video player should decode incoming data on-time, i.e. as soon as a single frame arrives from the network
- it has to provide basic control capabilities such as *play*, *pause* and *replay* functions
- video player should display video sequence within the browser's window

We implemented our video player plug-in with use of Netscape's Plug-In API (see section 6.1 in this document) and H.263 codec source code provided by Telenor (see section 7.2 in this document). Current version of video player is running only under IRIX operating system and with Netscape browser version 3.0 or higher.

The video player plug-in handles data with MIME type of video/h263 which is assumed to be H.263-coded video data. The plug-in has the following features implemented:

- can handle QCIF and CIF picture formats (QCIF can be expanded to CIF when requested by user)

- can *pause* and *replay* video sequence
- video sequence can not only be decoded and displayed but also be stored in a local file (when requested by user)
- it can be controlled by a Java applet via its simple *aa()* method (LiveConnect simple feature implemented together with Konrad Olszewski), see also below for more details

The decoding speed is set to 10 frames per second. A capability to modify it by a user can be easily added to the existing video player plug-in (the maximum decoding speed reached was of about 25 frames per second).
The user interface of video player plug-in is presented in the Annex A (Figure A-2) at the end of this document. The H.263 video player plug-in itself along with downloading instructions and sample video sequences is available on WWW at *http://naos.npac.syr.edu/jabu/HTML/demo.html* .

With our video player plug-in we are able to retrieve video sequences from the Web with a very low bandwidth consumption (typically 25-30 kbps at 10 frames per second for "talking heads" sequences) which proves the extraordinary features of the H.263 video coding scheme and its suitability for all kinds of collaborative systems for Web environment.

The implementation of our video-player Netscape plug-in needed a lot of effort to fully understand the Netscape plug-in paradigm. There were no examples provided with continuous data streaming from network implemented nor with even simple user interface. We had to insert video window into Netscape browser's interface which was not that simple as one could imagine.
But the most challenging difficulty during video-player Netscape plug-in implementation was to adopt a continuous data streaming mechanism to existing H.263 decoder. In original version of H.263 decoder, video data is read from file when needed. With Netscape Plug-in API we could use *client-pull* (see section 6.1 in this document) mode to ask for video data from the network, but this approach would introduce unexpected delays. Furthermore, the routine to be used in client-pull mode was not operational.
We thus deployed a *server-push* data transfer mode (see section 6.1 in this document). This required to made many important modifications to the original source code of H.263 decoder. We changed control flow scheme of the decoder so that it keeps data in a buffer, and after each frame it goes into *idle* mode. This activates plug-in methods responsible for data provision (see section 6.1 in this document) and allows to fill this buffer with new video data.

A very important issue here was our very first experiment with LiveConnect technology. As we mentioned in section 6.3 of this document, we decided to replace the *daemon* application of our teleconferencing system with a system designed by Konrad Olszewski [21]. This system is written in Java language (see section 5.4.5 in this document) which provides many convenient means for implementation distributed applications for Web.
To integrate our system with a system written in Java language, we could use Netscape's LiveConnect technology which is one of the newest features available today. This allows us to establish communication with our *conference manager* Netscape plug-in (see section 6.3 in this document) with Java applets belonging to the system to replace the daemons in our system.
In order to understand LiveConnect technology, which has not yet been used by many software developers, with cooperation of Konrad Olszewski we implemented a simple communication scheme between a Java applet and our H.263 video player Netscape plug-in.
In order to establish communication between our plug-in and an applet we implemented a special method in the plugin called *aa()* which is used by an applet to control the video player plug-in.

Because LiveConnect technology is completely new, we present below how we implemented it in our video-player plug-in.
First, we had to build a Java class corresponding to our Netscape plug-in. This class inherits from *Plugin* class provided with Netscape's Plug-In API ([4]). The definition of Java class (we named it *Pl* class) for our video-player plug-in was the following:

```
import netscape.plugin.Plugin;
class Pl extends Plugin {
  public native int aa();
```

```
}
```

Declared here *aa()* method will be defined in plug-in source code and also will be used by Java applets to communicate with plug-in (see below).
Upon video-player plug-in downloading by Netscape WWW browser a JavaScript object *Pl* is created and can be accessed by Java applets (see below).
Having Java class defined we could bound it into the source code of the video-player plug-in. We did it in the following way:

```
JRIEnv* env;
…
jref
NPP_GetJavaClass()
{
   env = NPN_GetJavaEnv();
   return use_Pl(env);
}
…
void
NPP_Shutdown(void)
{
                unuse_Pl(env);
}
…
void native_Pl_aa(JRIEnv* env, Pl* self, int option)
{
   if (option)
     {
       timerId=XtAppAddTimeOut(This->appContext,1,start,NULL);
     }
   else
     {
       timerId=XtAppAddTimeOut(This->appContext,1,pause,NULL);
     }
   return;
}
```

The two first methods are used to register/unregister Java environment in our plug-in and its corresponding Java class.
The method `native_Pl_aa()` the one which is called from Java applet and performs some actions on video-player plug-in. This method is also declared in Java class definition for plug-in (see above).

To communicate with a plug-in, a Java applet has to get appropriate pointer to a JavaScript *Pl* object representing Netscape plug-in. This is done in the following manner:

```
public class AA extends Applet
{
    private Button start, stop;
    private Pl o;
    public void init()
    {
      add(start =new Button("Start"));
      add(stop =new Button("Pause"));
            JSObject document=getDocument();
            o=(Pl)document.getMember("vplayer");/* pointer to plug-in
named "vplayer" inside HTML document is obtained here */
    }
    public JSObject getDocument() {
      return (JSObject)JSObject.getWindow(this).getMember("document");
    }
    public boolean action(Event e,Object ob) {
      if (ob.equals("Start")) {
```

```
            System.out.println("Plugin START");
          o.aa(1);/* applet invokes "start" command in plug-in*/
            System.out.println("Done");
          return true;
    }
    if (ob.equals("Pause")) {
            System.out.println("Plugin PAUSE");
          o.aa(0); /* applet invokes "pause" command in plug-in */
            System.out.println("Done");
          return true;
    }
    return super.action(e,ob);
  }
}
```

Both, the Java applet and video-player plug-in are embedded in the same HTML document and they interact with each other. APPLET tag within HTML document has to contain *mayscript* option to make Java applet be able to interact with JavaScript. EMBED tag for Netscape plug-in has to contain *name* option with the same value as indicated in *document.getMember(name)* method int the above example of Java applet.

The exemplary HTML document containing the interacting Java applet and Netscape H.263 video-player plug-in can contain the following:

```
<HTML>
<BODY>
<H1>Netscape plug-in with Java applet LiveConnect example</H1>
<HR>
This page contains two interacting objects: video-player plug-in and
simple Java applet which interacts with plug-in. You can control video-
player either via its own or applet's buttons.
<HR>
Java applet is running right here:
<br>
<APPLET name=app code="AA.class" width=200 height=30 mayscript></APPLET>
<HR>
Embeded H.263 video-player plug-in runs here:
<br>
<EMBED name=vplayer src="http://naos.npac.syr.edu:1972/jabu/heads.263"
width=500 height=400 FORMAT=CIF></EMBED>
</BODY>
</HTML>
```

In a similar way, the Netscape Plug-In API and LiveConnect features allow to control a Java applet from within a plug-in application. For details, please refer to [5] and [4].

## A.Adaptation of the H.263 codec for collaboratory systems in Internet

According to H.263 ITU-T Recommendation (see section 7.1 in this document or refer to [9]), its outstanding performance in terms of the quality of coded pictures and the number of generated bits is due to a very complex video coding algorithm. We analyzed all H.263 features and available software performance and the following conclusions were retrieved:

- encoding speed of the available codec in software had to be increased in order to make its use for real-time applications possible
- some error recovery mechanisms had to be implemented in the codec in order to allow video data losses when transmitted over the Internet
- modifications of source code had to be made in order to build a library with interface routines to H.263 codec such as *codeFrame()*, *decodeFrame()*. This would allow to incorporate many different coding algorithms within the same architecture of video codec.

To fulfill our needs presented above we decided the following:

- periodical I frames (INTRA coded, see section 7.1 in this document) have to be generated by H.263 encoder. This will allow to recover video decoding after any P-type video frame loss.
- source code of the H.263 codec has to be optimized and motion estimation algorithm has to be replaced to increase the encoding speed.

We realized that too many I frames in coded video stream increase significantly required bandwidth and that new motion estimation algorithm has to provide sufficient picture quality along with reduction of the computational complexity.

To optimize the source code of H.263 encoder the IRIS Developer Magic Performance Analyzer was used. With help of this tool the following modifications were found to be useful and therefore were implemented:

- two new options to the motion estimation have been implemented: *logarithmic search* and *subsampling pixel patterns* for SAD calculation (for details please refer to [7])
- the most CPU time consuming routines have been partially optimized
- periodical I frames has been implemented. The gap between two consecutive I frames in a video sequence is to be specified by a user (default value is 100000)
- interfaces to some useful routines have been created. This will allow to use H.263 software codec as a library when videoconferencing tools are being implemented

<u>After having these modifications implemented our H.263 encoder was able to encode several QCIF pictures within a second which make it possible to deploy in videoconferencing tools for the Web</u>. The encoding speed varies from 3 to 15 frames per second depending on a CPU used.
Comparison of the encoding speed for original and modified software encoder on different CPUs is presented in the table 7.4-1 below.

Table 7.4-1 Comparison of the encoding speed for original and modified H.263 encoder (in QCIF frames per second)

| CPU/clock freq. | R4600/100MHz | R4400/150MHz | R4400/200MHz | R10000/195MHz |
|---|---|---|---|---|
| original encoder | 0.8 | 1.3 | 1.8 | 3.5 |
| modified encoder | 3.4 | 5.0 | 7 | 12 |

Presented results are not satisfying if the cost of an end-system with powerful CPU is taken into consideration. We are aware that the encoding speed of H.263 software encoder has to be yet increased by a factor of at least 2-3. This can be done by further modifications of the source code, especially by elimination of time-consuming memory read/write operations, *for* and *while* loops optimization etc.

Periodical I frames increase the overall generated amount of bits by a quite significant factor (about 40% when every 10$^{th}$ frame is INTRA coded) but it gives also two very important advantages:

- *random access* capability to the video sequence
- *error recovery* possibility after data loss in the network

This two issues are important from the point of view of modern Internet services such as video on demand, videoconferencing etc. To decrease the excessive bandwidth consumption by periodical I frames, we did experiments with **wavelet encoding** of these frames. The wavelet encoding and compression methods are very efficient in terms of generated amount of data and quality of encoded pictures. The computational complexity of these methods is relatively high and as for now encoding in wavelets is rather slow. However, due to enormous research effort all over the world, the algorithms used for wavelet compression are getting faster reaching the boundaries of the real-time processing. We decided to integrate our H.263 encoder with wavelet compression scheme to obtain the very <u>first hybrid video encoder</u>. The results for QCIF pictures are only slightly better (several bytes per I frame) when wavelets are used but with larger picture sizes wavelet compression is believed to give a significant gain in number of generated bits with a very good quality of picture preserved. This work was supposed to give us a first experience with hybrid video coding which can be then deployed in the future videoconferencing systems for Web. As for now the advantages of wavelet compression scheme can be used in video-on-demand services with H.263 video coding.

Presented efforts resulted in having H.263 encoder to operate in real-time domain which allows to deploy it in collaboratory applications. This in turn, in conjunction with a very low bit rate and relatively good quality of encoded pictures makes such solution be one of the most advanced in the current state of Web technology.
For H.263 encoded video and ADPCM encoded audio the overall required bandwidth does not exceeds 64 kbps for typical sequences in videoconferencing (*talking heads*). This allows to use only one ISDN B channel (64 kbps) to participate in a teleconference with audio and video which is a very promising feature and gives the advantage over all currently used systems. For other sequences (used in video-on-demand systems) such as video clips, movies etc. two ISDN B channels (128 kbps) will be sufficient.

We deployed modified H.263 video encoder in our Web-integrated collaboratory system by integrating it with our video plugin for OpenDVE environment (see section 6.5 in this document).

### A.Possible further extensions and improvements

Time limitations did not allow to implement all features of Netscape H.263 video player plug-in. Only basic functionality has been assumed and some further extensions can be worth of implementation:

- random access to a video sequence feature, this could be implemented with the use of one of the *netscape methods* (NPN_RequestRead(), see section 6.1 in this document or refer to [4]) but it was not operational by the time of our work
- several colormap support mechanism, in current version of video player plug-in, a private colormap is created and installed which is not always allowed on some window manager systems
- binaries (executables) for different operating systems (SunOS, AIX etc.) should be compiled and linked
- support for more picture formats (4CIF and 16CIF) could be implemented, as for now it is not reasonable due to not sufficient H.263 software decoder performance

## I.Conclusions

We designed and implemented an experimental and unique Web-integrated collaboratory system for multimedia data distribution. Many state-of-the-art technologies have been deployed to make this system be as sophisticated as possible. We integrated several distinct technologies and concepts into one system for teleconferencing in the Internet environment. We also considered future trends in Web technology and we tried to base our design on them so that all further extensions to the preliminary system will be compatible with the most advanced technologies.

To summarize the results of our efforts we can state the following:

- teleconferencing and collaboration with use of Web browsers became possible with our system
- multimedia teleconferencing can be handled by Internet resources due to the state-of-the-art data compression schemes deployed in our system (mainly very low-bitrate video compression)
- developed system can operate on the Internet in spite of many resources availability limitations
- additional teleconferencing tools (modules) in form of OpenDVE plugins can be added to the existing collaboratory system
- our teleconferencing system encompasses technologies to be used in LiveMedia framework (see section 4.6 in this document)
- teleconferencing system we designed and implemented is compliant with major teleconferencing standards such as H.320 recommendation and T.120 family of protocols
- along with development of more new technologies for teleconferencing, the number of systems and different solutions will grow rapidly and they will assure better and better quality of service

Due to the time constraints we were not able to implement all useful features in our collaboratory system. Possible extensions to the existing version of our system are presented in section 6.7 in this document.

We encountered also some limitations and inconveniences imposed by technologies we integrated. We presented them in section 6.4 in this document.

The most unpleasant experience was with InSoft's OpenDVE Software Developers' Kit. Unfortunately, this product is provided by InSoft in form of undocumented files along with some very simple examples. Provided documentation covers only a small part of the entire OpenDVE technology features and the most important issues are not covered in examples. The marketing information from InSoft Inc. said far too much comparing to the actual form and contents of their SDK for OpenDVE.

The actual OpenDVE SDK lacks the following, crucial for collaboratory system development, features:

- there is no conference management scenarios in terms of conference events provided (we had to determine which conference events and when are generated).
- plugin management remains completely undocumented (using only undocumented header files we managed to incorporate OpenDVE plugins' management into our conference manager)
- there is no directions for a customized conference manager design (we built our conference manager without any guidance from InSoft)
- components of OpenDVE environment such as audio and video codecs are claimed to be accessible from any application but they are not
- some library files are missing while corresponding header files are provided (for example, *Help Viewer* tool which was claimed to be available to the programmers) eliminating their use in our own applications
- provided documentation lists routines which are never used in provided examples and there is no way to understand their functionality

We realized that the reason for such scandalous inconsistencies in what InSoft Inc. claims and what actual OpenDVE SDK offers to programmers, lies in the fact that InSoft does not really want people do develop their own stand-alone collaboratory systems but to use InSoft's products such as Communique!

In spite of that, we managed to deploy OpenDVE technology in the way we wanted to except for limitations imposed by this technology itself.

We hope that our multimedia collaboratory system can be easily adjusted to the upcoming LiveMedia framework for Internet and therefore become one of the very first truly multimedia collaboratory systems available on the market.

However, we have to stress that many extensions should be done to make this system be attractive and sophisticated enough to be accepted by the Internet society ...

# I.Acknowledgments

The author would like to express thanks to professors Czeslaw Jedrzejek of EFP and Marek Podgorny of NPAC for their helpful ideas, supervision and contribution to the work presented in this document , to Konrad Olszewski and Tomasz Stachowiak for their cooperation, to all those people who supported him during the research work and redaction of this document and to Northeast Parallel Architectures Center, Syracuse NY, USA where the entire research has been done.

The research work presented in this document was funded by US Department of Defense grant, Rome Lab. Contract No. F30602-95-C-0273, PR No. C-5-2293/4

# I.Abbreviations

| | |
|---|---|
| 3-D | Three - dimensional |
| ADPCM | Adaptive Differential Pulse Code Modulation |
| ANSI | American National Standards Institute |
| API | Application Programmers' Interface |
| ARM | Application Resource Manager |
| ASE | Application Service Element |
| ATM | Asynchronous Transfer Mode |
| B-ISDN | Broadband ISDN |
| CGI | Common Gateway Interface |
| CIF | Common Intermediate Format |
| CPU | Central Processing Unit |
| CSDN | Circuit Switched Digital Network |
| DCT | Discrete Cosine Transform |
| FDDI | Fiber-Distributed Data Interface |
| FTP | File Transfer Protocol |
| GCC | Generic Conference Control |
| GOB | Group of Blocks |
| GSM | Global System for Mobile Communication |
| GUI | Graphical User Interface |
| H.263 | one of the latest video-coding standards, emerged from h.261 |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| IAF | Internet Application Framework |
| IETF | Internet Engineering Task Force |
| IGMP | Internet Group Management Protocol |
| IMCP | Internet Message Control Protocol |
| IP | Internet Protocol |
| IPC | Inter Process Control |
| ISDN | Integrated Services Digital Network |
| ITU | International Telecommunication Union |
| IVS | Inria Videoconferencing System |
| JDBC | Java-DataBase Connectivity |
| kbps | kilobit per second |
| LAN | Local Area Network |
| Mbps | megabit per second |
| MBFT | Multipoint Binary File Transfer |
| MCS | Multipoint Communication Service |
| MCU | Multipoint Control Unit |
| MIME | Multipurpose Internet Mail Extensions |
| MPEG | Moving Pictures Expert Group |
| NAL | Network Abstraction Layer |
| NNTP | Network News Transfer Protocol |
| NV | Network Video |
| OpenDVE™ | Open Digital Video Everywhere |
| OSI | Open Systems Interconnection |
| PDU | Protocol Data Unit |
| PSDN | Packet Switched Digital Network |
| PSTN | Public Switched Telephone Network |
| QCIF | Quarter CIF |
| ODBC | Object-Oriented DataBase Connectivity |
| QoS | Quality of Service |
| RFC | Request For Comments |
| RSVP | Resource ReSerVation Protocol |
| RTCP | RTP Control Protocol |
| RTP | Real-time Transport Protocol |
| SAC | Syntax Arithmetic Coding |
| SAD | Signal Absolute Difference |

| | |
|---|---|
| SD | Session Directory |
| SDK | Software Developers' Kit |
| SET | Secure Electronic Transactions |
| SIT | Still Image Transfer |
| SMDS | Switched Multi-Megabit Data Service |
| SMTP | Simple Mail Transport Protocol |
| SNR | Signal to Noise Ratio |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TTL | Time To Live |
| UDP | User Datagram Protocol |
| URL | Unique Remote Locator |
| VAT | Visual Audio Tool |
| Vic | Videoconferencing Tool |
| VLC | Variable Length Coding |
| VRML | Virtual Reality Modeling Language |
| WWW | World Wide Web |

# I.References


1. "OpenDVE Architectural Overview", InSoft Inc., 1996
2. "OpenDVE Programming Overview", InSoft Inc., 1996
3. "OpenDVE Library Reference", InSoft Inc., 1996

4. "Netscape Plug-in API User's Guide", Netscape Communication Corporation, 1996
5. "LiveConnecting Plug-ins with Java", Netscape Communication Corporation, 1996

6. "Very Low Bit Rate Video Coding Using H.263 Coder", King N.Ngan, Douglas Chai, Andrew Millin, IEEE Trans. on Circuits and Systems for Video Technology, Vol.6, no.3, June 1996
7. "New Fast Algorithms for the Estimation of Block Motion Vectors", Bede Liu, Andre Zaccarin, IEEE Trans. on Circuits and Systems for Video Technology, Vol.3, no.2, April 1993
8. "vic: A Flexible Framework for Packet Video", Steven McCanne, Van Jacobson, Lawrence Berkeley Laboratory
9. Draft ITU-T Recommendation H.263, July, 1995

10. "RTP: A Transport Protocol for Real-Time Applications", H.Schulzrinne, S.Casner, R.Frederick, V.Jacobson, January 1996
11. "RTP profile for Audio and Video Conferences with Minimal Control", H.Schulzrinne, January 1996
12. "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", R.Braden, L.Zhang, S.Berson, S.Herzog, S.Jamin, February 1996
13. "Dynamic QoS Control of Multimedia Applications based on RTP", Ingo Busse, Bernd Deffner, Henning Schulzrinne, GDM Fokus, May 1995

14. "MBone, Interactive Multimedia on the Internet", Vinay Kumar, New Riders Publishing, Indiana, USA, 1996
15. "Video Conferencing Buyers Guide", Digital Video Magazine, September 1995
16. "H.320: A Quality Requirement Guide", VTEL Corporation, 1996
17. Draft Recommendation T.120 - Data Protocols for Multimedia Conferencing
18. Interactive Multimedia Association (IMA) Web Home Pages (http://www.ima.org)
19. "Videoconferencing: A Comprehensive Look", Jack Chen, Yidong Ding, 1996
20. "Audio and Video on the Internet - application comparison", PC Kurier 17/96 (in Polish)
21. "Building collaborative applications with database back-end with use of Web technologies", master thesis by Konrad Olszewski, September 1996


# I.Appendices

## *A.Annex A*

Figure A-1 The *conference manager* Netscape plug-in user interface


Figure A-2 The *H.263 video player* Netscape plug-in user interface


## *A.Annex B*

Figure B-1 Sample contents of a file *filename* used by collaboratory system to identify OpenDVE plugins available (DVE_PLUG_CFG = *filename*)

```
[Plugin Configuration]
```

```
SearchDir0 = /home/R11A/bulawa/DVE/localplugins
NumSearchDirs = 1
NumPlugins = 3
Plugin0 = JBVideoPlugin
Plugin1 = TSAudioPluginMotif
Plugin2 = TSBoardPluginMotif
```

## *A.Annex C*

Sample code implementing a Netscape plug-in application:

```
#include "npapi.h" /* Header file for Netscape Plug-In API */
…
/* You include here all needed libraries (Motif, Xlib etc.) */
…
/* Here is a structure containg mandatory and optional (plug-in's
private) variables and structures */
typedef struct _PluginInstance
{
    Widget  netscape_widget; /* The widget provided by netscape */
    Display    *display;         /* Netscape's display connection */
    uint16  mode;            /* NP_EMBED, NP_FULL, or NP_BACKGROUND */
    uint32 width, height;

    /*--- Start custom code */
    Window inWindow;
    GC gc;
    XtAppContext appContext;
  *--- End custom code */
} PluginInstance;

JRIEnv* env;
PluginInstance *This;

/* This method is called when browser starts and loads identified plug-
ins, allows to register MIME type corresponding to this plug-in:
MIME type:        video/h263
file suffix:      263
plug-in's name:   Video Player */
char * NPP_GetMIMEDescription(void)
{
  return("video/h263:263:Video Player");
}

/* This method is called when browser starts and loads identified plug-
ins, allows to register plug-in's interface in browser */
NPError NPP_GetValue(void *future, NPPVariable variable, void *value)
{
      NPError err = NPERR_NO_ERROR;

      switch (variable) {
            case NPPVpluginNameString:
                  *((char **)value) = "H.263 player plug-in";
                  break;
            case NPPVpluginDescriptionString:
                  *((char **)value) =
                        "This plugins handles h.263 video files and
plays "
                         "them in a window with some controls.";
                  break;
            default:
                  err = NPERR_GENERIC_ERROR;
      }
```

```
      return err;
}

/* This method is called when new plug-in instance is created by browser
*/
NPError NPP_Initialize(void)
{
  return NPERR_NO_ERROR;
}

/* This method is called to register a Java environment (used  by
LiveConnect) when new plug-in instance is created */
jref NPP_GetJavaClass()
{
   env = NPN_GetJavaEnv();
   return use_Pl(env);  /* Java object of the class Pl is used */
}

/* This method is called when a plug-in instance is deleted (after
NPP_Destroy() */
void NPP_Shutdown(void)
{
                unuse_Pl(env);
}

/* This method is called when new plug-in instance is created by
browser. You can parse here all arguments passed within EMBED tag in
HTML documents that contains a plug-in */
NPError NPP_New(NPMIMEType pluginType,
      NPP instance,
      uint16 mode,
      int16 argc,
      char* argn[],
      char* argv[],
      NPSavedData* saved)
{

      if (instance == NULL)
            return NPERR_INVALID_INSTANCE_ERROR;

      instance->pdata = NPN_MemAlloc(sizeof(PluginInstance));

      This = (PluginInstance*) instance->pdata;

      if (This != NULL)
      {
              int i;
            This->netscape_widget = NULL;
            This->display = NULL;
            This->mode = mode;    /* mode is NP_EMBED, NP_FULL, or
NP_BACKGROUND (see npapi.h) */
            This->width = 0;
            This->height = 0;
            This->inWindow = NULL;

            /* *Developers*: Initialize fields of your plugin
             * instance data here.  If the NPSavedData is non-
             * NULL, you can use that data (returned by you from
             * NPP_Destroy to set up the new plugin instance.
             */
                /* get any args we recognize from the html file */

            if (mode==NP_EMBED)
            {
                      for(i = 0; i < argc; i++)
```

```
                   {
                              if(!strcmp(argn[i], "FORMAT"))
                              {
                                          if(!strcmp(argv[i], "QCIF"))
                        {
                                                This->winPels = 176;
                                                 This->winLines = 144;
                        }
                                          if(!strcmp(argv[i], "CIF"))
                        {
                                                This->winPels = 352;
                                                This->winLines = 288;
                        }
                              }
             }
                  } /* if EMBED */
            return NPERR_NO_ERROR;
      }
      else
            return NPERR_OUT_OF_MEMORY_ERROR;
}/* NPP_New */

/* This method is called by browser when a plug-in instance is destroyed
(user changed URL,  HTML page etc.) */
NPError NPP_Destroy(NPP instance, NPSavedData** save)
{

      if (instance == NULL)
            return NPERR_INVALID_INSTANCE_ERROR;

      This = (PluginInstance*) instance->pdata;

      /* PLUGIN DEVELOPERS:
       *    If desired, call NP_MemAlloc to create a
       *    NPSavedDate structure containing any state information
       *    that you want restored if this plugin instance is later
       *    recreated.
       */

      if (This != NULL) {
            NPN_MemFree(instance->pdata);
            instance->pdata = NULL;
      }
      return NPERR_NO_ERROR;
}/* NPP_Destroy */


/* This method is called by browser when a plug-in instance is created
by NPP_New() */
/* It allows to build plug-in's private user interface and put it into
browser's window */
NPError NPP_SetWindow(NPP instance, NPWindow* window)
{
      if (instance == NULL)
            return NPERR_INVALID_INSTANCE_ERROR;

      This = (PluginInstance*) instance->pdata;
      /*
       * PLUGIN DEVELOPERS:
       *    Before setting window to point to the
       *    new window, you may wish to compare the new window
       *    info to the previous window (if any) to note window
       *    size changes, etc.
       */
```

```
      This->display =((NPSetWindowCallbackStruct *)window->ws_info)-
>display;
      This->inWindow = (Window)window->window;
      This->width = window->width;
      This->height = window->height;

      display=This->display;

      This->netscape_widget = XtWindowToWidget(This->display,
(Window)window->window);

      /*--- Start custom code */

      /* here you add all widgets you want to appear on user interface
*/

      /*--- End custom code */
      return NPERR_NO_ERROR;

}/* NPP_SetWindow */


/* This method is called by browser when a new stream is created (new
URL opened) */
/* You can initialize here your plug-in: variables etc. */
NPError NPP_NewStream(NPP instance,
            NPMIMEType type,
            NPStream *stream,
            NPBool seekable,
            uint16 *stype)
{
      if (instance == NULL)
            return NPERR_INVALID_INSTANCE_ERROR;

      This = (PluginInstance*) instance->pdata;

      /*--- Start custom code */
      /* We want a stream of data */

      *stype = NP_NORMAL;

      /* initialize variables */
      …..
      …..
      /*--- End custom code */

      return NPERR_NO_ERROR;
}/* NPP_NewStream */


/* PLUGIN DEVELOPERS:
 *    These next 2 functions are directly relevant in a plug-in which
 *    handles the data in a streaming manner. If you want zero bytes
 *    because no buffer space is YET available, return 0. As long as
 *    the stream has not been written to the plugin, Navigator will
 *    continue trying to send bytes.  If the plugin doesn't want them,
 *    just return some large number from NPP_WriteReady(), and
 *    ignore them in NPP_Write().  For a NP_ASFILE stream, they are
 *    still called but can safely be ignored using this strategy.
 */

int32 NPP_WriteReady(NPP instance, NPStream *stream)
{
      if (instance != NULL)
            This = (PluginInstance*) instance->pdata;
```

```
	/* We return number of required bytes and after they all (or part
of it) come from the network the NPP_Read() is called automatically by
browser */
	return required;
}


/* This method is called by browser when after NPP_RequestRead() a date
arrived from the network and can be presented to a plug-in */
int32 NPP_Write(NPP instance, NPStream *stream, int32 offset, int32 len,
void *buffer)
{

	if (instance != NULL)
	  {
		This = (PluginInstance*) instance->pdata;
		}
	/* Here we accept a data from buffer and return number of bytes
accepted */
	/* control flow returns to NPP_RequestRead() */
	return len;	/* The number of bytes accepted */
}


/* This method is called by browser when user opens new URL (new stream
of data is created)*/
NPError NPP_DestroyStream(NPP instance, NPStream *stream, NPError
reason)
{

	if (instance == NULL)
		return NPERR_INVALID_INSTANCE_ERROR;
	This = (PluginInstance*) instance->pdata;

	/* here "clean" everything the previous data stream: memory,
structures etc. */

	return NPERR_NO_ERROR;
}

void NPP_StreamAsFile(NPP instance, NPStream *stream, const char* fname)
{
	/* implementation not actually required */
	return;
}

void NPP_Print(NPP instance, NPPrint* printInfo)
{
	/* implementation not actually required */
	return;
}

/* Here you implement all your private routines which provides plug-in's
functionality: data decoding, displaying, calculations and so on. They
can be called from within any plug-in methods according to plug-in's
functionality */
…
void play(..)
{
..
}
…
…
void pause(..)
{
```

```
..
}

etc.
…
/* Here LiveConnect method used by Java applets to control plug-in is
implemented */

void native_Pl_aa(JRIEnv* env, Pl* self, int option)
{
   if (option)
     {
       timerId=XtAppAddTimeOut(This->appContext,1,start,NULL); /* plug-
in's private routine play() is called */
     }
   else
     {
       timerId=XtAppAddTimeOut(This->appContext,1,pause,NULL); /* plug-
in's private routine pause() is called */

     }
   return;
}
```

For more details on how to implement a Netscape plug-in application, please refer to [4]