SYNCHRONOUS COLLABORATION OVER THE INTERNET

- *INCORPORATING SYNCHRONOUS COLLABORATION TOOLS INTO WEB BASED GROUPWARE SYSTEMS*

MAGNUS INGVARSSON

SWEDISH INSTITUTE FOR SYSTEMS DEVELOPMENT

AUGUST 1997

ABSTRACT:

The document describes architectures and methods that can be deployed for implementing support for synchronous collaboration via the Internet in general, and in particular in combination with Web based cooperative environments, or virtual workspaces. The document contains a thorough investigation of different architectures and protocols for synchronous communication over the Internet, after which there's an in-depth discussion of how these techniques can be incorporated into an existing Web based collaborative application, BSCW (Basic Support for Cooperative Work). However, the results and conclusions are quite general and can easily be applied to other systems as well. BSCW is a system which is being jointly developed by SISU and the German national research institute GMD, in the CoopWWW project with funding from the European Commission under contract TE 2003 of the Telematic Applications Programme. This paper is the final result of a Master's thesis carried out at SISU in 1996.

CONTENTS

1 INTRODUCTION

The development of the World Wide Web and its wide availability offers a great opportunity to develop and implement systems for collaborative information sharing for widely-dispersed working groups. One of the main benefits of the World Wide Web is its platform independent nature. Another, is the lack of need for installing special client applications. These features of the Web can be effectively utilized to develop collaborative systems that potentially could enable group collaboration anytime, anywhere, and anyhow.

The BSCW (Basic Support for Collaborative Work) project at GMD has developed a system, which draws on the benefits of the WWW architectures and tries to realize the visions above. When this paper was initiated, the BSCW system was a simple research product from the FIT-CSCW group at GMD, in Bonn/St.Augustin, Germany. A consortium consisting of SISU, GMD, Nexor, RWTH Aachen, and horz informatik has since received funding from the Commission of the European Union, within the Telematic Applications Programme, for a project under the name CoopWWW. The main building block, or basic system kernel, of the CoopWWW system is the BSCW shared workspace system from GMD.

The BSCW system is heavily focused on support for asynchronous collaboration, e.g. document sharing, discussions, group management, and event notification. Up until very recently, the system lacked any type of more synchronous collaborative support, such as audio-, video-, and data-conferencing. Various user evaluations of the BSCW system suggest there is in fact a need for more synchronous modes of cooperation, and also that the system provides for a smooth transition between asynchronous and synchronous modes of operation.

The main aim of the work to be carried out by SISU within this project is the integration of synchronous communication tools into the BSCW architecture. The purpose is not to develop new software for real-time communication, but to integrate third-party tools into the BSCW system. The purpose of this document is to investigate possible architectures that can be used for the integration aspects of this work, but also to investigate different architectures for the synchronous communication itself, so that appropriate software for integration can be chosen. To fully understand the issues in this document, some experience with groupware systems, the Internet in general, and the World Wide Web in particular is needed. Experience with tools for synchronous communication, such as video-conferencing applications will also facilitate the reading.

This document is structured as follows: in Section 2, the concepts of CSCW and groupware systems in general are introduced. Section 3 covers relevant standards and protocols necessary to implement synchronous communication. Extensions of the WWW architecture for collaboration are discussed in Section 4. The specific extensions with synchronous communication tools are covered in section 5. Section 6 introduces the BSCW shared workspace system, and section 7 the integration of synchronous collaboration facilities into the BSCW architecture. Conclusions are found in Section 8, and suggestions for further work within the area in section 9.

2 OVERVIEW OF CSCW AND GROUPWARE

Computer Supported Cooperative Work, or *CSCW*, is a relatively large field of research. There is no exact definition of the term CSCW, but the general consensus is that CSCW is the scientific discipline that motivates and validates the design of *groupware*. CSCW is also concerned with theory of how people work together and how groupware systems affect group behavior and organizations.

Groupware is the hardware and software that enables and supports group work. Groupware itself is also a very vague term, and its definition could be stretched to include any product that is used by more than one person. A more practical definition is to restrict the term groupware to include only products that are focused directly on groups and group processes. Groupware then becomes any system that is designed to enable and support groups to work together electronically.

2.1 Time and Space Dependencies

Traditional group activities can be placed into a time/space matrix, according to their respective time and space dependencies, as seen in Figure 2.1 below. The time/space matrix is often used as a reference when categorizing different characteristics of group collaboration.

		Space	
		Same location	**Different location**
Time	**Synchronous**	Face-to-face meeting	Telephone call
	Asynchronous	Post-it-note Bulletin board	Letter

Figure 2.1: *CSCW time/space matrix, categorizing traditional group activities, with respect to time and space dependencies.*

Different features and characteristics of groupware can be categorized into the time/space matrix as follows:

		Space	
		Same location	**Different location**
Time	**Synchronous**	Meeting room facilities	Video/audio-conferencing Chat systems Application sharing
	Asynchronous	Co-authoring tools	E-mail Mailing lists Newsgroups File sharing

Figure 2.2: *Different group activities supported by groupware in the time/space matrix.*

Groupware systems are usually developed to enable and support group activities in the two right squares in Figure 2.2 above. Common groupware systems like Lotus Notes, Novell GroupWise, and Microsoft Exchange mainly support distributed asynchronous activities, as in the bottom right box of the figure. However, there are also often facilities for scheduling and planning activities on the left half of the time-space matrix, for example using group calendars and automatic scheduling facilities.

Same time - same location collaborative methods, such as meeting room facilities are out of scope of this document. The focus of virtual workspaces is on the right half of the figure above, and the focus of this document mainly on the upper right box, i.e. systems supporting synchronous collaboration between people at different locations, or put another way, synchronous collaborative services for widely dispersed working groups.

3 COLLABORATION ON THE INTERNET AND WWW

3.1 Background

A number of Web-based systems to provide support for group collaboration have been developed during the last couple of years. Almost all of these systems suffer from the same severe problems, namely the usage of customized servers and clients. While this approach may have been realistic in the past, when both server and client software had a fairly easy and simple structure, it now suffers from the fact that this is no longer true. It is especially untrue for client software, where, for example, extensions of HTML and new technology like Java have made writing a WWW client (or modifying an existing one) anything but a "quick fix." Furthermore, the development of WWW into an integrated platform with support not only for the HTTP protocol complicates matters even more. Any such customized client would have to provide similar support, and thus this task becomes impossible to achieve.

The increased complexity of WWW client technology has made relying on third party software for this purpose a must, which immediately rules out the possibility of providing extended functionality by ways of non-standard HTML tags and non-standard client APIs.

In previous systems for extended WWW functionality, customized clients have often been used to provide support for non-standard HTML tags and non-standard client APIs. A modified WWW client can of course be programmed to perform numerous additional tasks based on these non-standard HTML tags. However, if such a client does not provide support for newly standardized HTML and new client technology, such a browser is only useful in specialized tasks, requiring the user to use another browser for other tasks. This approach to provide extended functionality no longer seems realistic.

The market for WWW clients is currently dominated by Netscape, with its Communicator suite and the old Navigator, and by Microsoft with its Internet Explorer. In order to build a successful Web based collaborative environment, one has to chose one of these platforms as the main target, or alternatively develop for the greatest common denominator between the two dominating browsers thus guaranteeing support for a vast majority of all browsers.

The definition of *WWW extensions for collaboration* is very vague and can actually incorporate virtually any Internet facility that supports collaboration between people in any way.

3.2 Overview

Today's Web pages are typically single-user applications. CSCW systems, or groupware, are multi-user applications. So far, the technology has not allowed much support for collaborative functionality within the WWW framework. However, new and emerging technology like Java, JavaScript, extended HTML functionality, support for Plug-ins, VRML, etc., have opened new possibilities for extending the Web to support group activities. Actually, HTML, which was originally a hypertext markup language, is rapidly developing into a platform independent resource description language to assemble applications from different components (embedded objects, JavaScript, Java applets, etc.).

Existing systems supporting group activities via WWW are, with a few exceptions, limited to asynchronous functionality. Typical extensions include electronic messaging systems and shared workspaces. The functionality of these systems is usually quite restricted and thus the benefits of using such a system are relatively small. There are a lot of other groupware systems on the market, which do not build on WWW, but include support both for asynchronous and synchronous activities.

This kind of reasoning makes it clear that there is an obvious need to extend the CSCW support via WWW to include synchronous activities, such as shared whiteboards, on-line discussions, and audio- and video-conferencing.

An asynchronous working group, over the Internet, is formed by the people who are authenticated to access the contents of a defined set of Web pages and services. A synchronous working group can be formed by a subgroup of these people, and possibly include other external resources or people, currently cooperating in a synchronous facility such as a video-conferencing system.

In order to create user awareness of who are currently participating in a group, events like members leaving or entering the group must be broadcast or in some other way made available to everyone in a group. Other interesting information of this kind can be that certain members are currently busy with other work, or temporarily away from their computers.

A user must be able to "see" who are currently participating in a group. This can be done e.g. by providing names, pictures, etc., of the active members. This list of members could be used to get another members attention in order to engage in a conversation of some sort, e.g. a chat-session, or an audio- or video-conference.

In the following sections different aspects of collaborative work are briefly described, including some reflections on the state of the current market situation. There is a division into audio-, video-, and data-conferencing. However, in reality, many applications include functionality from all of the following different methods to communicate.

3.3 Audio-conferencing

Audio-conferencing, or Internet telephony, has gained enormously in popularity since the first applications started to appear on the market. Today, there are a large variety of Internet telephony applications, differing in both the underlying technology and in targeted user groups. Almost all of the available applications support both full-duplex and half-duplex operation, and works with standard PC sound hardware.

Most Internet phones only support point-to-point communication, though there are a few exceptions that can handle either point-to-multipoint or true multipoint transmission.

Almost all Internet phones uses proprietary codecs, though some of the applications let the user choose between a selection of different codecs. However, the new ITU H.323 standard for low-bandwidth communications is quickly being adopted by most vendors, so interoperability between different vendors' products is imminent.

In addition to simple voice communication, most Internet phones on the market offer a set of useful services, voice mail, integrated e-mail, flexible call blocking, call transfer, and

answering machines, sometimes even with different outgoing messages depending on who is calling. Some audio-conferencing applications also include extended functionality in form of a shared whiteboard, text-based chat, and file transfer. An application that includes these extended services should probably be categorized as a general Internet collaboration tool, instead of a simple Internet phone. However, this is no coincidence; there is currently a trend in the market towards general collaborative functionality among Internet phones, many of which are now, or in the near future, being extended with advanced collaborative services and video capability.

The scheme for finding users and calling someone works quite different between different products. There is no general strategy, though most vendors have set up *User Location Servers* (ULSs) listing active users, either publicly available or hidden requiring the caller to know the address of the person to call. Some vendors have also set up white pages listing also those users that are not currently active.

The general trend within the market is a move toward H.323 compliance and compatibility with third-party "White Pages" providers, or online directory services, like Four11, WhoWhere?, etc. These online directory services are further explained in Section 4.7.

3.4 Video-conferencing

Internet video-conferencing takes the concept of audio-conferencing, as described in the previous section, one step further by adding video capability to the client applications. As with audio-conferencing products there is a large variety of different applications, but most of the applications only support point-to-point transmission. Some products offer some sort of point-to-multipoint transmission, and a few true multi-point transmission.

Since bandwidth constraints are problematic for audio-conferencing, this problem is even worse for video-conferencing. Video data is much denser than audio, and since video-conferencing applications use both audio and video, they require bandwidth for the output of both the audio and the video codecs concurrently.

Most video-conferencing applications work quite well inside a LAN, where bandwidth is generally not a problem. The performance on ISDN connections is quite good as well, typically delivering about 4-7 frames per second (fps). With modem connections and poor Internet connectivity the overall quality of audio and video drops to an almost unbearable level, at most delivering about 2 fps video, an quite often 0 fps. On low-bandwidth connections the video is usually sacrificed in order to keep up a good audio connection. The best products on the market use advanced scaleable codecs that automatically adjust transmission to available bandwidth. In general, different CODEC algorithms have different levels of compression, quality and speed.

The market for Internet based *conferencing* tools is currently dominated by NetMeeting from Microsoft, closely followed by Enhanced CU-SeeMe from WhitePine, and Netscape Conference distributed with the new Netscape Communicator suite. The latest versions of these tools all comply with (or intend to comply with) the ITU standards for IP-based video and audio communication (H.323) with integrated application sharing (T.120), and they are thus interoperable.

3.5 Data-conferencing

Data-conferencing can be defined to include a variety of different communication facilities. However, the major different areas most often associated with data-conferencing are:

- *text-conferencing*
- *shared whiteboards*
- *application sharing*

The three different applications above are briefly explained in the following sections.

3.5.1 Text-conferencing

The Web architecture can quite easily be extended with functionality to support real-time text-based conferencing. Most Internet systems for text-based conferencing use the *Internet Relay Chat* (IRC) protocol, which is further described in Section 4.5.3. Text-based systems may be implemented using a Java client or platform specific clients. Using the IRC protocol, interoperability between different vendors can easily be achieved.

Text-based chat systems are very popular on the Internet. Most audio- and video-conferencing applications come with built-in support for text chat. With the current reality of poor bandwidth on the Internet causing frequent audio and video break-ups, a chat window is really quite useful as a complement. Furthermore, text chat can be very useful in explaining or clarifying certain issues in a conference. A stand-alone text chat system could also be useful for "manually" negotiating between users to take further steps to initiate a more advanced synchronous session using e.g. audio or video.

3.5.2 Shared Whiteboards

Both audio- and video-conferencing applications often come with the extended functionality of a shared whiteboard. A shared whiteboard reflects the annotations of participants to all clients in real time. In addition to basic drawing support, most whiteboard applications support a broader range of more advanced functionality, such as screen capture devices, screen synchronization, and image compression to speed up transmission.

3.5.3 Application Sharing

There are a few products on the market that provides some sort of application sharing. Application sharing, as the term suggests, lets conferees in a conference view, and in some cases control applications on a remote computer.

The responsiveness of shared applications is very much dependent on the type of application and the quality of the connection, i.e. the available bandwidth. Generally, text-based applications such as word processors or spreadsheets require less bandwidth than e.g. image editors, and are therefore also the ones that work best when sharing applications over the Internet.

Currently, there is no widely accepted standard for application sharing, so the products available use a variety of different proprietary protocols and thus there is currently little or no interoperability between products from different vendors.

3.6 Scalability of Multipoint Conferencing

Multipoint conferencing is technically problematic in several different ways. One of the main problems is scalability, the increased communication resulting from a growing number of participants in a conference. This problem is by no means specific to computer conferencing. In fact, this simple mathematical phenomenon appears in many other contexts as well. The basic problem is illustrated by the following figure.

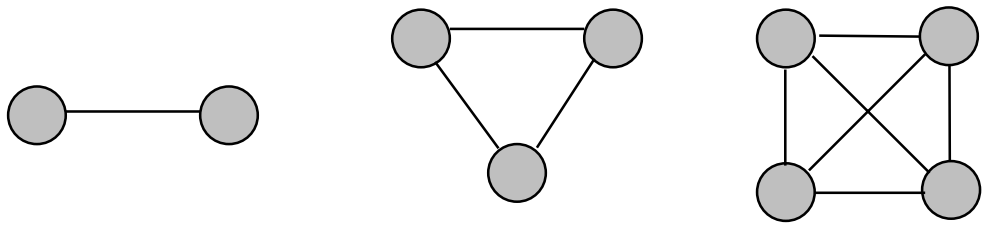

Figure 3.1: *In a server-absent environment, the number of connections between clients increases faster than the number of participants.*

The number of connections, and thus the communication overhead, increases more rapidly than the number of participants in a server absent environment. A solution to this problem is to route the traffic between the participants through some central point. In the case of four participants, this would reduce the number of connections to four, instead of six as in Figure 3.1 above. This reduction makes a larger impact as the number of participants grows. With five participants, not using a server would require ten connections, while using a server would only require five connections.

However, a server introduces other scalability problems, like performance constraints as the number of clients connected to the server increases. With point-to-point conferencing there are few or no advantages of routing the traffic through a server. However, with multipoint conferencing there are often great advantages with routing all traffic through a server. The server software could be programmed to dynamically adapt the traffic sent from and to participating clients to utilize the available bandwidth as effectively as possible. Figure 3.2 below shows a typical environment in which a multi-point conference may take place.

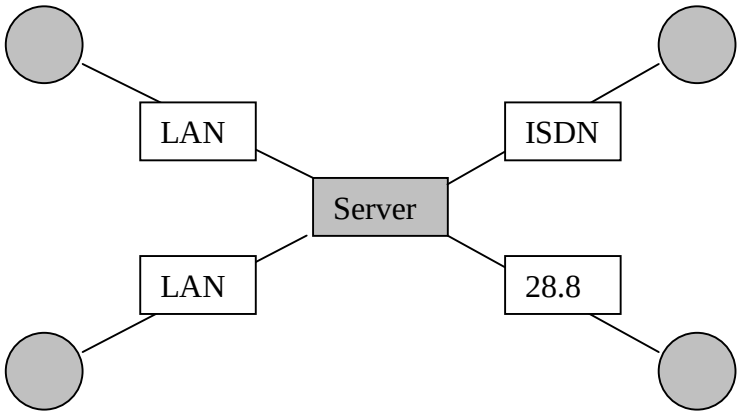

Figure 3.2: *A typical environment for a multi-point conference with clients connected on varying bandwidth conditions.*

One way of adapting the traffic sent to different clients is by using a technique which is commonly known as *bandwidth pruning*. A server for routing multimedia traffic, such as video and audio, can have built-in intelligence about the available bandwidth to different clients connected to the server. This way, the server can automatically adapt the amount of data and prioritize what data is being sent to each client, instead of flooding all clients with all data which would result in the network arbitrarily dropping random packets for clients on low-bandwidth connections. Users on high bandwidth connections will also benefit since the traffic between high bandwidth users need not be cut down because there are users on low bandwidth connections in the conference. Also, communication between users on high and low bandwidth connections will be smoother. Thus, pruning can enhance the quality for all participants in a conference.

Another way to further enhance the quality and effective use of bandwidth in multi-point conferencing is to connect multiple servers for distributing traffic and use some sort of multicasting technique. Multicasting is further described in Section 4.3.

3.7 Dynamic IP Address Allocation Schemes

Several different methods to assign IP addresses to clients connected to a network exist. The ones that are problematic in terms of synchronous communication, are those that have a dynamic addressing scheme, which assigns different IP address every time a computer is connected. For example, dynamic IP address assignment is very common amongst public Internet service providers (ISPs) that provide dial-up access for the public. However, dynamic addressing is also used sometimes within local area networks.

Proxies and firewalls are also problematic in terms of synchronous communication. Here the problem is two-fold:

- Some communication applications will not function correctly unless the firewall is configured appropriately. Some applications will not work at all through a firewall.
- There is no general method, via the Web, of asking for a client's IP address if the client is connected through a proxy.

In addition to dynamic addresses, proxies, and firewalls, there are also a number of related problems with using IP addresses to identify people on the Internet. By definition, an IP address identifies a computer connected to the network, and not the person sitting behind it. It is thus possible, and even likely, that a person will access the system from a number of different computers and IP addresses. A one-to-one mapping between a user and an IP address is therefore unrealistic, and only possible in special cases or during a limited period of time.

The problems as described above cannot easily be solved. The approach should probably be a combination of manual and automatic IP address configuration. For example, a user that usually accesses a system from the same computer using the same IP address (and being aware of this fact) should be able to specify this. As another example, a user that connects to a system via a public ISP, that uses dynamic IP address allocation, should be able to specify this and have the system automatically retrieve the current IP address.

3.8 On-line Directory Services and User Location Servers

Some of the problems with dynamic IP address allocation are partly solved by some of the major on-line directory systems available on the World Wide Web. Some of the more well known are Four11, WhoWhere?, Bigfoot, Infospace, and IAF (Internet Address Finder).

While most of the vendors of synchronous communication tools are running their own User Location Servers (ULSs), some of the online directories are running separate ULSs that keep track of the users currently using a particular conferencing application.

A User Location Server is basically a program that keeps track of the current users of a particular application and the Internet addresses of these users. Thus, it is not the same as services that are known as Internet "white pages", which list users independently of whether or not they are currently using a particular application.

All Internet communication facilities that use some sort of User Location Service more or less operate in a similar way. Upon start-up, the client application sends a message to the specific ULS that the application has been configured to use, giving the ULS such necessary parameters as current IP address, e-mail address, etc. The ULS then keeps a list of all the current users of the application. This list is usually presented to the user with some built-in facility in the client application, and/or as an ordinary Web page. If a Web page is used for presenting the list of active users there is usually some support of calling another person simply by clicking on that person's name or an icon. The ULS will then have a CGI program create an appropriate response of a specific (non-standardized) MIME-type together with some connection parameters to automatically spawn the client application and have it connect to the specified user.

In addition to the necessary Internet addresses, a user of a ULS usually has the possibility of entering some additional parameters that will be displayed in the ULS list. This could be, for example, a physical location, some comment, a URL, etc.

Today, most tools for Internet audio-, and video-conferencing (e.g. Enhanced CU-SeeMe, VDOnet VDOPhone, Microsoft NetMeeting, etc.) come with a field where the user is supposed to type in a preferred ULS to be used to find other users and to list oneself.

4 STANDARDS AND PROTOCOLS

In this section, an overview of the most important standards for video-, audio-, and data-conferencing over the Internet is given. In general, there are a lot of standards defined by various standardization organizations and a tremendous amount of proprietary standards for Internet conferencing. However, only standards and protocols usable over the Internet are mentioned in this section. Standards applicable solely to other transmission mediums such as ISDN, POTS and high-speed Ethernets are out of scope of the issues of this document.

4.1 Internet Video-Conferencing Standards

For video-conferencing, the most important standards are those handling the compression and decompression of audio and video. In addition, special standards are needed to handle multipoint conferences, to provide security and to control data and application sharing.

Currently, there is a big lack of widely accepted standards. There is a tremendous proliferation of proprietary standards, but attempts to define winning standards are in progress. As a result of this, systems from different vendors are still incompatible. There is also no interoperability across platforms or different transmission mediums.

The International Telecommunications Union, ITU (formerly known as CCITT), defines the most widely accepted standards in the area of video-conferencing.

4.1.1 H.323

ITU has defined the H.323 standards suite for audiographic conferencing over packet switched networks. H.323 has emerged as the most important standards suite for video-/audio-conferencing over the Internet. H.323 is really a set of different standards. The components of H.323 are:

- G.723 for audio communication,
- H.263 for video communication,
- T.120 for multi-point data-conferencing,
- H.245 and Q.931 for call control.

H.323 defines e.g.:

- how calls are set up,
- negotiation of capabilities,
- wire transmission of data,
- default audio and video codecs.

4.1.2 H.263 - Video

The H.263 standard video codec is a variation of H.261 optimized for low bandwidth connections.

4.1.3 G.723 - Audio

The G.723 component of H.323 defines the standard (default) audio codecs to be used within H.323. These are 5.3kbps GSM based and 6.4 TrueSpeech based modes.

4.1.4 H.245 and Q.931

The H.245 and Q.931 protocols can be seen as the "heart" of H.323 interoperability. These protocols define the standards for establishing the audio and video connections. They also have built-in support for capability negotiation, e.g. codec selection.

4.1.5 T.120

T.120 is a standard defined by ITU for *data-conferencing.* To be exact, it is really a series of protocols: T.122, T.123, T.124, T.125, ... The T.120 protocol standard enables products from different vendors to interoperate over data-conferencing. The standard defines e.g.:

- network interfaces and wire formats,
- a session model for conferencing,
- data transmission facilities.

4.2 Streaming Protocols

On the Internet, the network service provided is on a *best effort* level. This means that packets sent over the network may be lost on the way, and thus never reaching its intended receiver. The inherent loss off packets is handled by the Transmission Control Protocol (TCP), by re-sending packets that have been lost. This re-transmission process takes a lot of time, and may introduce delays of several seconds, and sometimes much longer than that. These delays are often small enough not to cause any major problems for *on-demand* type of transmission, e.g. between a Web server and a Web client. However, it is quite obvious that these delays are unacceptable for audio- and video-conferencing applications. Therefore, most real-time audio and video applications do not re-transmit lost packets, but only play the packets that arrive within a specific time-frame. The RTP protocol was designed to make the additional control information useful in a situation where packet loss is accepted.

4.2.1 RTP - Real-time Transport Protocol

The Real-time Transport Protocol (RTP) has been under development by IETF for about four years, and has recently been standardized as RFC-1889. Several leading vendors of real-time Internet applications have since declared support for the protocol, e.g. Netscape and Microsoft.

RTP specifies a set of headers to facilitate transmission of real-time data. However, RTP does not include any specifications on the particular codecs used and format of the data that is transmitted. Thus, RTP can be used when transmitting standardized formats like H.261, H.263, GSM, etc., as well as many proprietary formats.

The RTP protocol contains two different types of packets: RTP packets that contain the audio and video being transmitted, and RTCP (Real Time Control Protocol) packets, which contain mainly feed-back information about the quality of the transmission that arrives at the receiver.

When using RTP in a video-conferencing application between two computers, each source, i.e. camera, microphone, etc., is assigned its own RTP stream. Thus, in a point-to-point video-conference with audio, four separate RTP streams are used, two for video in each direction and two for audio in each direction. The alternative would be to use multiplexing, and combing all the packets from one computer into a single stream. However, this is not very practical and not often used, since this makes it impossible for a client to select only the audio or the video, e.g. in a video-conference, to be received.

RTP packets contain several important fields, e.g. *sequence number*, *time stamp*, *payload type*, the specific format of the transmitted media, and a *synchronization source identifier* which identifies the source of the RTP packet.

The RTCP packets contain extensive information about the status of an RTP stream transmission, which can be used by the application to dynamically adapt the transmission of the data to minimize loss and improve overall quality. For example, if packet loss reported in RTCP packets exceeds a certain threshold, the sending application can automatically switch to an encoding that requires lower bandwidth. This is usually called *automatic rate adaptation.*

4.3 IP Multicast and MBONE

When a packet is *broadcast*, it is delivered to *all* attached hosts. In contrast, when a packet is *unicast*, it is delivered from its source to a *single* destination. *Multicasting* refers to a technique by which a single packet can be delivered to a set of *selected* destinations.

The MBONE, or Multicast Backbone, is a virtual network consisting of a communication layer on top of parts of the Internet. The idea is to support multicasting of IP packets over the Internet, a function that is currently not incorporated into most routers on the net. However, more and more routers support this technology. The MBONE network consists of islands that support multicast packets. In order to send multicast packets between different parts of MBONE, the packets are encapsulated as to appear as normal unicast packets and then sent trough “tunnels” (virtual point-to-point links) to another system that supports multicasting. Besides the MBONE, IP Multicast is currently used within enterprise networks for audio and video distribution.

Lately there has been some movement in the market by networking hardware vendors and industry leading software companies to kick off the technology by encouraging software vendors to develop software that operates over MBONE.

MBONE, and IP Multicast in general, is certainly an interesting technology, but its development has been very slow so far. In terms of collaboration for a widely dispersed working group, MBONE can currently not be seen as a viable communication technology for several reasons, the most obvious, of course, the lack of commercial applications that support it. In addition, there are several technical difficulties and security issues acting as further obstacles. Interoperability between different vendors’ hardware and software is still a big problem. Also, the increasing use of firewalls is problematic, since these are usually configured to block UDP (User Datagram Protocol) traffic, which is the protocol normally used by IP Multicast.

4.4 Some Relevant Proprietary Standards

Mainly due to the lack of widely accepted standards, there has been an enormous proliferation of proprietary standards for Internet conferencing. However, recent development point to that there is in fact a certain convergence of standard efforts into using protocols defined by various standardization bodies, mainly ITU.

Product vendors, whilst fierce competitors, have an interest in making there products comply with international standards, and thereby also opening the door for potential interoperability between products from different vendors.

4.4.1 CU-SeeMe

One of the most widely used applications for video-conferencing on the Internet, is CU-SeeMe. CU-SeeMe was first developed as a research project at Cornell University, and was actually one of the first video-conferencing applications for the Internet. The software has since been commercialized by WhitePine Software.

CU-SeeMe is currently not based on open communication protocol standards. The video codec in CU-SeeMe is proprietary and very different from other codecs used in similar applications.

CU-SeeMe can handle multipoint conferencing through the use of a CU-SeeMe reflector. A reflector is a special application that reflects the traffic between the users in a particular video-conference. Theoretically, the traffic in a multipoint conference can be sent between all the clients involved in a conference. The idea of a reflector is to reduce the communication overhead by reducing the number of communication channels and to be able to more effectively control the transfer rate of data to and from participating clients.

4.5 Other Collaboration Relevant Protocols

Besides the protocols and standards used for the synchronous communication itself, there are also other protocols that have a large impact on the ways in which it is possible to collaborate over the Internet. After all, synchronous communication by itself would add little value if there was no way to effectively collaborate and negotiate about e.g. the date and time a synchronous session will occur, who will participate, and so on.

In this section, some protocols and standards that have a central role as support mechanisms for synchronous collaboration are presented.

4.5.1 NNTP - Network News Transfer Protocol

NNTP is a protocol that specifies how news articles may be distributed, retrieved, and posted from and to central databases. The protocol is designed so that a user can select only those articles he wishes to read. The protocol also includes mechanisms for cross-referencing articles, indexing, and expiration of old articles.

One central component of a powerful groupware system is the ability to have threaded discussions. NNTP has played an important role in implementing this functionality in Internet and Intranet settings. However, there are also many Internet based systems that implement threaded discussions in completely different manners. NNTP, however, is a very strong mechanism for transferring and replicating threaded discussions between databases around the world, a fact proven by the enormous popularity of Network news, or Usenet news. The standard also incorporates powerful mechanisms to keep track of new versus old messages, e.g. the *newsrc* file on the client sides.

4.5.2 LDAP - Lightweight Directory Access Protocol

LDAP, which has been around for sometime, got a boost in May 1996, when Netscape announced an LDAP-based directory server. This action spurred other vendors to quickly declare support for the standard, which eventually promises to give users a single method to search and retrieve information from directory systems.

LDAP is basically a subset of the retrieve mechanisms found in X.500, and was developed as a streamlined method to access data from a X.500 directory. A client implementation of X.500's DAP (Directory Access Protocol) was believed to require too much of a PC's processing time and there is no inherent support for running it over TCP/IP. LDAP, on the other hand, has been specifically designed to run over TCP/IP. In addition, it uses network bandwidth more conservatively than DAP.

Since its arrival, and its recent boost, LDAP has already made a big impact on the market, and most vendors active in the directory market are working fast to make their applications and systems LDAP-compliant. However, even though LDAP delivers a common method of accessing the information in diverse directory systems, it does not solve the problems of the tremendous amount of different and proprietary implementations of the directories themselves. The LDAP standard is further described in RFC-1823.

4.5.3 IRC - Internet Relay Chat

IRC stands for "Internet Relay Chat", and was originally designed by Jarkko Oikarinen, Finland in 1988. Originally, IRC was designed as a replacement of the UNIX *talk* program, but it has since become much more than that. IRC is basically a multi-user chat system where users around the world can hold group or private discussion on different *channels.* (IRC gained international fame during the Persian Gulf war in 1991, where updates from around the world instantaneously came across the Internet.)

4.5.4 vCard and vCalendar

vCard and *vCalendar* are two closely related standards defined by a consortium called versit. versit is a multivendor initiative founded by Apple, AT&T, IBM, and Siemens. The rights to the vCard and vCalendar specifications have very recently been turned over to the Internet Mail Consortium, IMC. One of the main goals and visions of versit is to enable diverse communication devices, applications and services from competing vendors to interoperate in all environments. Among the main areas that versit has looked at is Personal Data Interchange (PDI), conferencing and messaging, and wired connectivity. The rights to the vCard and vCalendar specifications have very recently been turned over to the Internet Mail Consortium, IMC.

Personal Data Interchange, by definition, occurs every time two or more persons communicate. This communication can be both in a business and personal context, face-to-face, or across space and time. Interchanges like these often include information such as telephone numbers, business cards, date and times of appointments and meetings, etc. Computer systems can be specifically designed to support this form of interchange to make it quick and reliable, and to ensure that the information is properly stored and available when needed. Currently, there is no common standard or set of standards to collect, structure and communicate PDI information across different communication channels, a problem that was recognized by versit. Among the most interesting technologies originating with versit are the vCard and the vCalendar specifications.

4.5.4.1 vCard

vCard is a specification of a format for an electronic, virtual business card, or vCard. The format of a vCard is defined independently of the method used to transport it and suitable for interchange between different applications and systems. The standard is specified in (versit 1996).

The vCard standard has been off to a slow start, but has recently gained in leading industry vendor support. To date, applications with support for vCard include e.g. Netscape Communicator, and future versions of Microsoft Internet Explorer. The specification has reached a stage of maturity to be quickly accepted as an industry standard, and more applications with support for vCard are likely to appear soon.

4.5.4.2 vCalendar

vCalendar, which is described in (versit 1996), is a specification of a format for electronic calendaring and scheduling. The vCalendar format is suitable for interchange of calendar entries between applications and systems, and independent of the particular method used for transport.

The purpose of the vCalendar format is to enable transfer of information about *event* and *todo* types of calendaring and scheduling entries. An *event* is basically the information that makes up a meeting that is to take place at a particular point in time, at some location, etc. A *todo* is an entry that represents an assignment or an action that has to be carried out. In terms of scheduling synchronous sessions in a virtual workspace, *event* type information is the most relevant. The specification also includes recommendations on how to transfer vCalendar objects in MIME encapsulated messages, using the non-standardized MIME type text/x-vCalendar, and also for presenting vCalendar objects in a viewable format on the World Wide Web.

The vCalendar specification is very new, and there are to date only a few commercially available applications that support it, e.g. the new Communicator suite by Netscape. There is, however, a tremendous amount of support for the format by leading industry vendors, and the format is likely to be included and supported in both new software and newer versions of existing calendaring and scheduling applications.

4.6 Application Sharing Standards

One of the most interesting areas in terms of synchronous collaboration is the ability to share applications across the network. Unfortunately, this is one of the technologically most difficult tasks to realize, a fact which becomes quite obvious when reflecting upon currently available applications and systems for this particular purpose.

Application sharing is not currently part of the T.120 standard for data-conferencing. However, submissions for the inclusion of application sharing in T.120 have been made, and widely accepted standards are likely to be pinned down in the near future. One has to realize, however, that total platform interoperability in terms of application sharing is unrealistic and currently far from technically possible to implement. Even though some limited platform interoperability can be achieved, most of the application sharing protocols are designed for collaboration on the same infrastructure environments.

5 IMPLEMENTING SYNCHRONOUS FUNCTIONALITY

5.1 Integration of Synchronous Tools into the Web

Synchronous communication tools can be integrated into WWW-based collaboration systems using a large variety of approaches. The most important difference between various approaches is the *level of integration*. Generally speaking, the level of integration can be categorized into three different groups:

- *Close integration*, making the synchronous tools indistinguishable from the WWW client. This can be achieved e.g. by embedded Java applets, ActiveX components, or client plug-ins.
- *Semi-close integration,* running the synchronous tools as separate applications, making them launchable by using an unstandardized MIME type and with built-in support to interact with the WWW client, e.g. "kicking" a running WWW client into loading specific URLs.
- *Loose integration*, running the synchronous collaboration tools as different applications, but enabling automatic launching by using an unstandardized MIME-type.

The following sections present different architectures and techniques and that can be used for extending the functionality of the basic Web client-server architectures and protocols.

5.2 Limitations of the Web Architecture

The Web client-server architecture does not directly support any sort of collaboration. However, the CGI interface (further described in Section 5.3) does provide a simple way of extending the functionality without requiring any modifications to client applications. There are, however, some constraints of the Web architecture, the protocols, and the clients themselves, that limit the level of interactivity that can be achieved using this standard architecture.

The HTTP protocol is a stateless protocol that cannot guarantee any level of transmission rate between server and client. HTTP is therefore not suitable for transmission of real-time data like audio and video, or any continuous media. On the contrary, HTTP transmissions in general tend to be bursty and uneven.

The HTTP protocol was designed for client-to-server communication initiated by the client and subsequent server-to-client transmission of data, i.e. as a protocol optimized for *data retrieval*. Any other type of communication using the HTTP protocol is more or less problematic. In particular client-to-client communication or server-to-client initiated communication is hard to achieve. In terms of developing interactive client-server applications this creates a significant problem, since interactivity usually requires the server to actively communicate with the clients.

For the reasons stated above, attempts to integrate real-time audio and video into the Web have not been very successful. Today, however, live audio and video can be integrated into Web pages, using new technology like the plug-in architecture, Java, and ActiveX. Most of these approaches replace the basic protocols for Web transactions (HTTP over TCP), with

streaming protocols that have been specifically designed for transmission of real-time data over packet-switched networks, e.g. RTP (Real-Time Protocol) or other protocols based on UDP (User Datagram Protocol).

Applications that use streaming protocols (e.g. audio/video viewers or audio/video conferencing tools) can by some simple methods be rather elegantly used in combination with Web browsers using HTTP and TCP. The application that uses the streaming protocol may be *closely* integrated into the Web running as a plug-in or *semi-closely* integrated running as a helper application. The differences between the two approaches are only visible on the user level. An application running as a plug-in is in fact a separate application, just like a helper, though it executes within screen-space inside the Web client. Thus, what we have done is to add another client application. The following figure illustrates the general architecture:

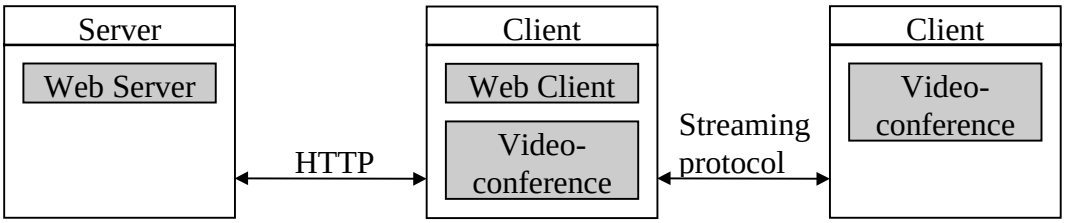

Figure 5.1: *The integration of helper applications and plug-ins using streaming protocols.*

Thus the helper or plug-in application is responsible for the transmission of real-time data, hence completely relieving the Web client itself from this responsibility. Furthermore, the helper (or plug-in) can be launched and automatically "kicked" into contacting a specific computer by returning a MIME-type associated with the application and the necessary parameters for telling the application what to do (usually an IP address of another client or host to contact). For this to work, the Web client has to be properly configured to launch the right application upon receiving a specific MIME-type, and the helper application has to be able to interpret the parameters being sent to it. This scheme can successfully be used both for integrating *on-demand* applications (viewers), or more interactive applications like Internet phones and video-conferencing utilities.

5.3 The Common Gateway Interface

The *Common Gateway Interface*, or *CGI*, is a standard interface between external applications and Web servers. Contrary to HTML files, which are static and fetched by the Web server upon receiving a request, CGI programs are executed in real-time and thus capable of producing dynamic information.

A common use of CGI is for "hooking up" databases to the World Wide Web. Since the information in a database changes continuously, it would not be possible to pre-produce HTML pages that reflect the contents of the database correctly. A CGI program, on the other hand, can create the HTML pages "on-the fly", so that their contents always correctly reflect the database structure.

There are really no limits on what can be executed through a CGI program. However, the process should not take too long to execute since the WWW client which issues a request is put in "waiting mode", until the CGI program returns the results of the request.

CGI programs are usually put in special directories for security reasons, and the WWW server has to be configured correctly so that the programs will be executed.

5.3.1 Different CGI Script Languages

CGI programs can be written in any language that is allowed to be executed on the system on which the WWW server runs. The most common languages for writing CGI programs are (in no particular order):

- PERL
- C / C++
- Python
- TCL
- Fortran
- Java
- any Unix shell.

5.3.2 Compiled vs. Interpreted

As seen in the list in the previous section, CGI programs can be written using programming languages that need compilation, like C and Fortran. However, they can also be written using a scripting language like some shell, PERL, TCL, or Python. There are advantages and disadvantages with both. Compiled programs will usually execute faster than interpreted scripts. However, scripts are easier to modify and maintain than compiled programs, though they will execute slower since they have to be interpreted in real-time.

5.4 Java Applets and Applications

5.4.1 General Characteristics

Java is a language intended to be used both for stand-alone applications and for embedded applets that are executed on a virtual machine on the client side. Java *byte-code* can be transferred over the net and executed on the client machine. Java source code is compiled using a platform specific Java compiler that produces the intermediate byte-code. This byte-code is transferred over the net to the client where it is interpreted using a platform specific interpreter and executed on a platform independent virtual machine. The fact that the programs are platform independent makes Java very well suited for such a diversified environment as the Internet.

Up until now, the contents of Web pages have been static. Java opens new possibilities for providing interactive content in WWW pages. Java moves the user interaction from the server to the client machine, where it should be. It eliminates the need to send information from the client to the server in order to provide interaction. Using Java, data objects can be encapsulated to take care of themselves on the client side.

Unfortunately, the discussion of Java as a platform to support synchronous communication quickly turns into a discussion of Java security management and policies.

5.4.2 Java Security Layers

The basic problem when transferring code over the net to be executed on the host is that those programs must have access to certain resources on the host machine. One cannot simply

forbid downloaded programs to access resources all together. For a program to be useful it has to access certain resources. Obviously, such access has to be carefully controlled.

The security in Java basically operates at two layers. First there is the security enforced at compile-time when Java source code is compiled into byte-code. The Java compiler performs an extensive compile-time checking for potential errors. However this is not enough, since it is conceptually possible to develop "hostile" compilers to produce malicious byte-code.

The second layer is at the client where applets loaded over the net have to pass through the *byte-code verifier*, which checks that the class conforms to the Java language specifications.

5.4.3 Basic Security Dilemma

The basic problems with applets loaded over the net are:

- In general, applets loaded over the net are prevented from reading and writing files on the client file system, and from making network connections except to the originating host. The security policy chosen by Netscape is to prevent reading and writing of files all together.
- Applets loaded over the net are also prevented from starting other programs on the client, load libraries, or to define native method calls. An applet that would be allowed define native method calls would gain direct access to the underlying computer.

There are other restrictions on what an applet can do, but what is mentioned above cover most cases. There is no foreseeable solution to overcome any of the properties stated above.

There are two different ways an applet can be loaded into the Java system, through the *applet class loader* or through the *file system loader*. The way an applet enters the Java run-time system decides what is allowed to do. Applets loaded via the file system loader do not go through the byte-code verifier. Applets loaded via the file system loader are allowed to read and write files on the client. All applets that are loaded via the applet class loader have to pass through the byte-code verifier.

The security policy for stand-alone applications is very different and more relaxed than applet security policies. Stand-alone applications are loaded by the file system loader, and thus will not pass through the byte-code verifier. Stand-alone applications are for example allowed to read, write, delete and execute local files (not allowed by Netscape applets). They can also load libraries, connect to ports on client and on 3rd hosts (also allowed by Netscape applets loaded from the local file system but not by applets loaded from the network).

5.4.4 Good use of Java

The Java language is well suited to provide interactive content via WWW, due to its platform independent nature. Since Java byte-code is executed on the client side, this eliminates the need to communicate with the server, as would be the case executing a CGI script. In addition, Java byte-code can be executed effectively and in a secure fashion on the client. As a comparison, a CGI script always runs the risk of a slow or failing execution, due to network congestion or server failure.

5.5 JavaScript

JavaScript can be seen as a complement to the Java language. While Java is used by to develop stand-alone applications and applets, JavaScript is intended to be used to dynamically script the behavior of objects running on either the client or the server.

Many industry-leading companies have already agreed to adopt JavaScript as an open standard scripting language and intend to provide it in future products. This broad industry support will undoubtedly speed up the development of JavaScript.

JavaScript is very well suited for enhancing the user interface of a Web page by moving some computation to the client, and thus lowering the number of required server requests. JavaScript is typically used for dynamically computing and updating WWW forms, and for writing event handlers that reacts to certain client events such as the clicking of a button or a hypertext link.

JavaScript resembles the Java syntax, but does not use the static typing and strong type checking found in Java. JavaScript scripts are embedded in HTML pages and interpreted at run-time by the client, i.e. there is no similarity to compilation of Java code into byte-code. In general, JavaScript can, at best, be viewed as a subpart of Java, and thus restrictions in Java will also apply to JavaScript. Presently, the weakness of the language prohibits any complex, or advanced usage.

JavaScript is best suited for writing *event handlers*, to take care of actions that result from something that the user does. In general, JavaScript can be used to implement functionality that would otherwise be implemented by CGI scripts and for enhancing the user interface of a Web application. Since JavaScript is interpreted on the client side, fast response times for user actions are possible, making user interaction smoother. JavaScript can also be used to communicate with applets and execute plug-ins.

5.6 Client Plug-ins

The support for Plug-ins in the latest versions of Netscape’s and Microsoft’s browsers, allows third parties to extend the clients with additional functionality. These added capabilities, when installed and properly configured, act indistinguishable to the user from the basic functionality.

Plug-ins can be embedded, full-screen, or hidden. An embedded plug-in appears as a rectangular subpart of an HTML document. One feature that a plug-in can implement is progressive viewing of data that is being processed as it arrives as a stream from the network.

A plug-in is associated with a MIME type that the Web client has no native support for. When the Web client encounters data of an unknown MIME type, it checks whether there is a plug-in associated with that particular MIME type and loads it.

There are numerous vendors currently developing plug-ins for the Netscape clients. Most of these support playback or real-time playback of different media and multimedia formats, mainly on Windows platforms.

A basic problem with plug-ins is that they are native to the platform on which the Web client runs. This undoubtedly causes “major head-ache” in cross-platform development projects where portability is a big issue. The goal of e.g. the Netscape plug-in API, however, is to be functionally equivalent across all platforms.

5.7 ActiveX

ActiveX is a set of technologies from Microsoft, to enable interactivity on the World Wide Web. The ActiveX infrastructure includes both server and client technology and components. ActiveX is basically a slimmed down variant of OLE, which is optimized for size and speed to fit into the constraints of the WWW architecture.

Currently, ActiveX is only available on Windows platforms. However, Microsoft is working with other vendors to implement versions for Macintosh and UNIX based systems. Even though Microsoft has developed the technology, it is not completely proprietary. An independent organization, the ActiveX Working Group, has recently been formed to “provide stewardship for the ActiveX standard as it is implemented on multiple platforms and operating systems”. In other words, the main purpose of the group is to promote and manage the evolution of ActiveX across platforms and to increase the interoperability with other environments. Nevertheless, Microsoft does of course have tremendous influence on the direction of the ActiveX technology.

5.8 VRML - Virtual Reality Modeling Language

VRML (*Virtual Reality Modeling Language*) is a language for describing multi-participant interactive simulations - virtual worlds within the World Wide Web. VRML is an open, platform-independent file format for 3D graphics. So far, a separate VRML viewer has usually been used to view VRML worlds. However, the latest beta version of Netscape Navigator with support for inline plug-ins allows VRML worlds to be embedded into Web pages. VRML is described in (Bell *et al*. 1995).

6 THE BSCW SHARED WORKSPACE SYSTEM

6.1 System Overview

The *BSCW* (*Basic Support for Cooperative Work*) project at GMD has developed a system that provides a set of collaboration services that use existing Web technologies. The system provides widely dispersed groups of users on different platforms and network environments, with easy ways to collaborate and share information in a Web environment.

The BSCW Shared Workspace system (Bentley *et al*. 1997) is an extension of the World Wide Web, implemented as set of CGI programs using the standard Common Gateway Interface (CGI) of Web servers, and accessible by an unmodified Web client, such as Netscape Navigator or Microsoft Internet Explorer. No modifications to servers, clients or protocols are needed. The current version of the system (BSCW 3.01, released in June 1997) has been extensively modified and re-designed since the release of the first system (Bentley *et al*. 1995).

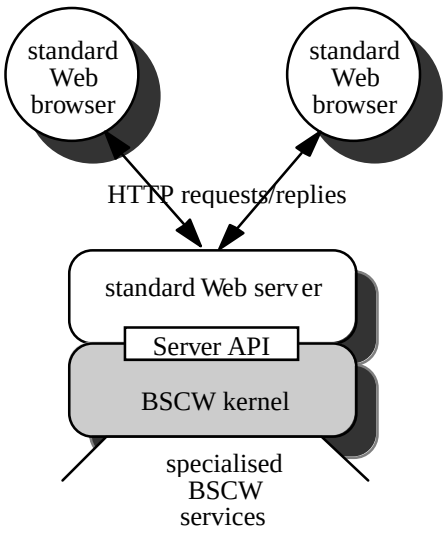

Figure 6.1: *The overall architecture of the BSCW system as an extension of the Web.*

The BSCW system can be broken down into three different layers, which deal with different aspects of the system functionality, as seen in the figure below. The *request handling layer* parses the Web client request and creates a *request object.* The request object is then dispatched to an appropriate *operation handler*, which is the implementation of the functionality that was requested. The operation handlers interacts with the *persistent object store* (the BSCW database) and finally creates a response object which is sent back to the request handling layer and formatted as an HTML (or possibly e-mail) response and sent back to the client.

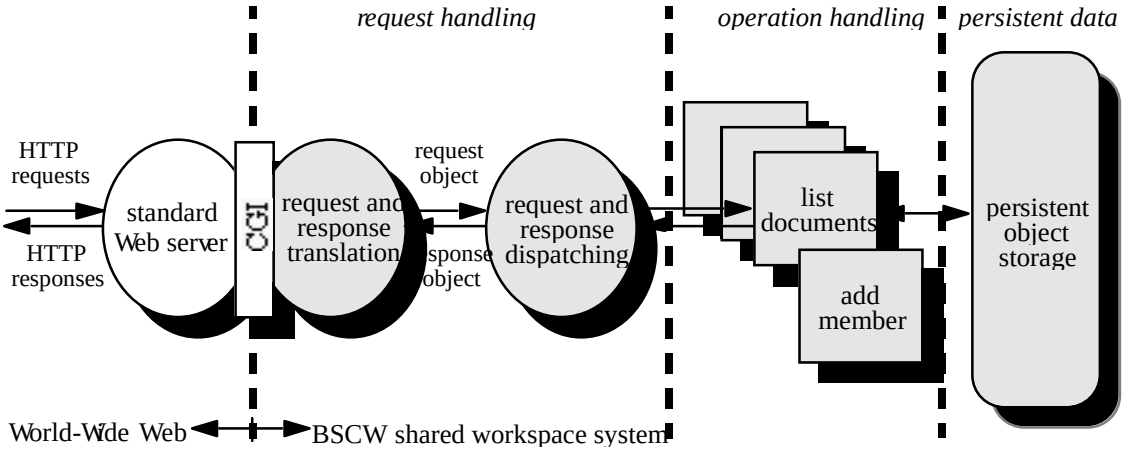

Figure 6.2: *The structure of the BSCW system.*

6.2 User Interface

The user interface in the BSCW system currently conforms strictly to the HTML 2.0 specification, and none of its features rely on special or proprietary client capabilities such as Java, JavaScript, tables and frames. However, today a vast majority of the Web clients that people actually use (Netscape Navigator and Microsoft Internet Explorer) posses all these more advanced features. Recent drafts of the HTML specifications also include most of the current de-facto standards. Thus, the baseline requirements for using the BSCW system are set at a very low level, but they are likely to be higher in future versions of the system.

The latest version of the BSCW system, however, uses JavaScript to enhance the interface for JavaScript enabled browsers. Nevertheless, there is still a strict requirement that all functionality should be available using a baseline browser.

Figure 6.3 below shows the user interface of the BSCW shared workspace system.

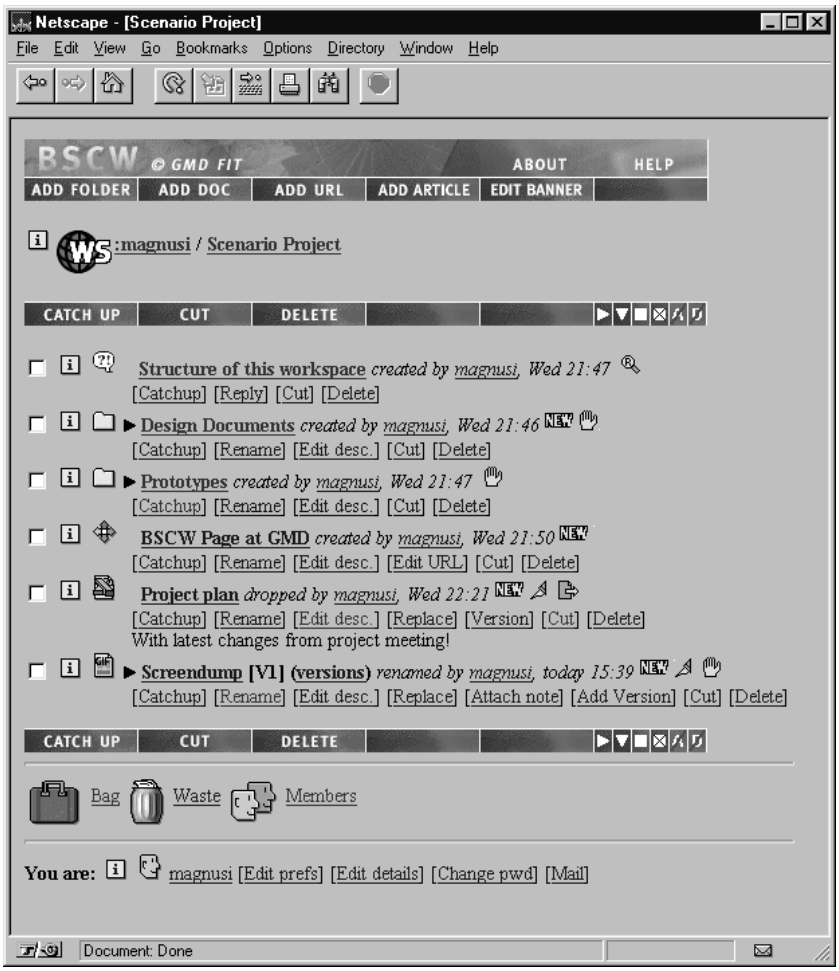

Figure 6.3: *The user interface of the BSCW shared workspace system.*

6.3 System Implementation and Infrastructure

The extended functionality of BSCW is provided by Python scripts using the standard CGI interface to communicate with the HTTP demon. Using a scripting language like Python has the advantage that the scripts can easily be modified and maintained, since no compilation is necessary.

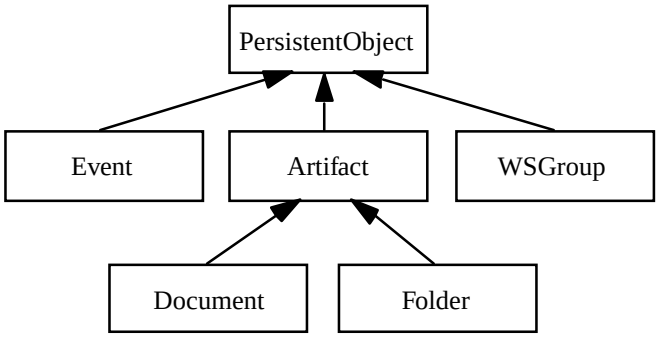

Figure 6.4: *Fragment of the object hierarchy (arrows indicate inheritance relationships).*

The BSCW system is built around a simple object-oriented architecture, as shown in Figure 6.4 above. The system can be extended relatively easy, by means of implementing new object types that inherit the properties of the top-level class types in the object hierarchy.

6.3.1 Use of the Python Language

The BSCW server is implemented completely using the Python programming language. Python (named after Monty Python's Flying Circus) is a language developed by Guido van Rossum, at Centre for Informatics and Computer Science (CWI), in Amsterdam. The language is described in (Rossum 1996).

Python is a very high level language (VHLL), that is syntactically similar to ABC, C, C++, and Modula-3. The language is object-oriented and interpreted. It has dynamic typing and is extensible. Python mainly supports interactive use, scripting and prototyping. Python has become quite popular for implementing CGI scripts, a purpose for which it was not originally designed, but for which inherent support has been added to later version of the Python library. In fact, the language has a very large set of libraries for performing all kinds of tasks, including many functions that can not be found in other comparable scripting languages like e.g. PERL.

Maybe the most serious problem with Python is that of performance and scalability. The interpreter does not perform adequately well on all platforms. However, this is not a problem specific to Python; all interpreted languages suffer from this to some extent. Also, there have not been sufficient tests to determine how many simultaneous requests to execute Python scripts will affect performance.

Another problem has been that of usage and spread. Previously, there was very little commercial use of Python. However, this problem is rapidly disappearing. It is safe to assume that the language is gaining in usage. Today, there are in fact many commercial organizations using Python. For example, Python is used extensively by the Infoseek search engine (http://www.infoseek.com) and the Four11 directory service (http://www.four11.com). Other high profile organizations using Python include the Johnson Space Center at NASA, the US National Bureau of Standards, Silicon Graphics Inc., and many more.

The language has reached a level of maturity, where is relatively stable. The formation of The Python Software Activity, PSA (http://www.python.org/psa/), ensures some kind of support in the event that its creator, Guido van Rossum, would suddenly disappear from the scene.

6.3.2 Helper Applications for File Upload

Later releases of Netscape Navigator (2.0, 3.0, 4.0, …) include support for file upload, using the POST method and a scheme based on RFC-1863. Microsoft Internet Explorer also supports this in an update to their 3.02 browser. However, previous WWW client software lacked this support. Therefore, platform-specific helper applications were developed to support file upload to the BSCW server. These helper applications provide a simple interface to the local file system, and are launched automatically by a correctly configured WWW client, when a specific MIME-type application/x-bscw is received.

The BSCW system relies heavily on MIME types to support file upload and download. The current focus with distinction of types based on file suffix places some severe and not easily overcome problems on the system with respect to cross-platform functionality.

6.4 Possible Extensions

Version 2.1 of the BSCW system lacked the following functionality which is often found in comparable products (not necessarily Web-based):

- calendaring and scheduling,
- video-, audio- and data-conferencing,
- directory information services,
- integration with existing enterprise systems,
- replication of data.

6.5 Current Development Status

Version 2.0 of the BSCW system was released to the pubic domain in June 1996. A bug-fix release (2.1) was released in September 96. Version 3.0 was released in May 1997, and a subsequent bug-fix (3.01) in June 1997. All code is offered free of charge for non-commercial use. The current focus of development efforts is to produce an open implementation of the basic mechanisms, which makes the system easy to customize and suitable for extensions.

The BSCW system is currently being developed and extended with funding from the European Union, within the Telematic Applications Program. Within this project, other specialized components like interfaces to enterprise information systems, decision support systems, and advanced versioning and data conversion utilities are also being added to the basic BSCW kernel.

7 SCENARIO FOR SYNCHRONOUS COLLABORATION SERVICES IN THE BSCW ARCHITECTURE

The main aim of SISU's plans in terms of synchronous communication for BSCW is to select, evaluate and integrate third party tools for collaboration into the BSCW system, providing extended collaboration support to the shared workspace. The purpose is not to develop new real-time collaboration technology, but to implement facilities that enable a user to initiate real-time communication and collaboration within the shared workspace with the help of third party software.

One of the main goals of the integration of synchronous components into BSCW is to provide a clear separation between the representation of synchronous capabilities in the BSCW system, and the actual synchronous application software used to support these sessions. Thus, the focus of development efforts will be on the supporting mechanisms for synchronous collaboration and not so much on the synchronous communication itself.

In light of recent industry developments and the fast technological developments within the area, the services for synchronous collaboration to be developed for BSCW will have to assume very little in terms of different third party solutions, and thus the services should be easy to re-design to work with other third party tools. In fact, one of the main aims should be to reduce the dependence on any particular conferencing technology. The core functionality of the synchronous communication components of BSCW should thus not be linked in any irrecoverable way with particular client or server software solutions. On the contrary, the core functionality should be specifically designed to facilitate the extension of additional tools.

7.1 General Requirements

The integration of a meeting service into the BSCW kernel is associated with a number of requirements on this service to provide:

- a seamless integration with the BSCW kernel, allowing deployment as a single system,
- a flexible architecture that allows the BSCW system to be used as an access point to meetings of different types,
- a set of user interface mechanisms allowing presentation and interaction with meeting objects in a consistent manner,
- access from a consistent user interface across different computing and network infrastructures,
- conformance to basic BSCW services, such as access control, event notification, and group management.

In terms of synchronous collaboration, the BSCW system should be seen as the "access point" to synchronous collaboration sessions, or meetings. Synchronous collaboration can be either point-to-point (from one person to another) or *multi-point* (between a group of people). Both point-to-point and multi-point communication should be inherently supported by the system.

A point-to-point session is more likely to be *ad-hoc* than a multi-point session, since a meeting where a whole group of people will participate generally requires more planning and pre-scheduling. However, point-to-point meetings may of course also be scheduled in

advance. Hence, the system should make no assumptions whatsoever about the characteristics of scheduled meetings. However, for ad-hoc sessions, the system should initially be designed to facilitate point-to-point communication, since this will be the most likely scenario.

Thus, the main services to be provided are:

- the ability to schedule, access details, and join meetings with workspace members (or a subset) or with an arbitrary set of users,
- the ability to quickly initiate ad-hoc meetings with another user.

These requirements lead to a model that results in the introduction of new object types to be incorporated into the BSCW object hierarchy, of which the important ones are objects to represent meetings and user calendars.

7.2 Active Notification Services

In BSCW, the user interface to the events, that take place in a workspace, is essentially a passive one: unless the user logs in to the BSCW server, there is no way to receive events that have occurred in a workspace. The details of how a more active event notification service, likely to use e-mail, will be supported and implemented within the BSCW system are yet to be determined. Any event notification concerning events that take place with respect to meeting objects should obviously be coordinated with the rest of the event notification service.

Nevertheless, it is quite obvious that an event notification scheme to inform users that they have been scheduled to participate in a meeting would be very useful. Initially, a meeting notification using e-mail could be implemented as a separate function. Future user evaluation and later implementation decisions should inform a possible re-design and a more active and advanced notification system in the future.

7.3 Representation of time

In the BSCW system, the time of events and actions are represented by local time, i.e. the time given by the operating system by a specific time request. This approach creates a potential problem in terms of scheduling meetings. Participants of a meeting can be located in different time zones of the world. Depending on the type of the meeting, different representations of time may be required.

For example, if the meeting is an event scheduled to take place at a certain geographic location, the starting time is best represented by the local time of that location. On the other hand, if the participants are located in different time zones during the actual meeting (which could be the case for e.g. a video-conference), a different approach may be necessary.

When creating a meeting, the creator specifies the date and time when the meeting will take place. The system could automatically check and select the local time zone (of the server), but the time zone for the meeting should be open to alteration by the creator. With this approach, all meetings are automatically assigned the local time zone of the BSCW server, in case the creator does not choose to override this selection.

7.4 Scheduled Meeting Services

The following sections describe in detail how a meeting service could be integrated with the user interface of the BSCW kernel and scenarios for basic operations. All user interface details in the following section are based on version 3.01 of the BSCW system, which of course are subject to change extensively in the future. Screen dumps in this section are from version 3.01 or from prototypes currently under development.

7.4.1 Extension of the Basic Workspace View

The basic *user-view* is extended with a personal *Meetings icon* that is placed in the same area as the *Bag* and the *Trashcan* (this makes sense because the meeting list is a "personal area" just like the bag and the trashcan). The meetings-icon itself can look different if meeting invitations exist or not. Clicking the icon results in the *personal meeting list view* (as represented by the Calendar object) being loaded and scheduled meetings displayed, just like clicking the trashcan will display the contents of the trashcan.

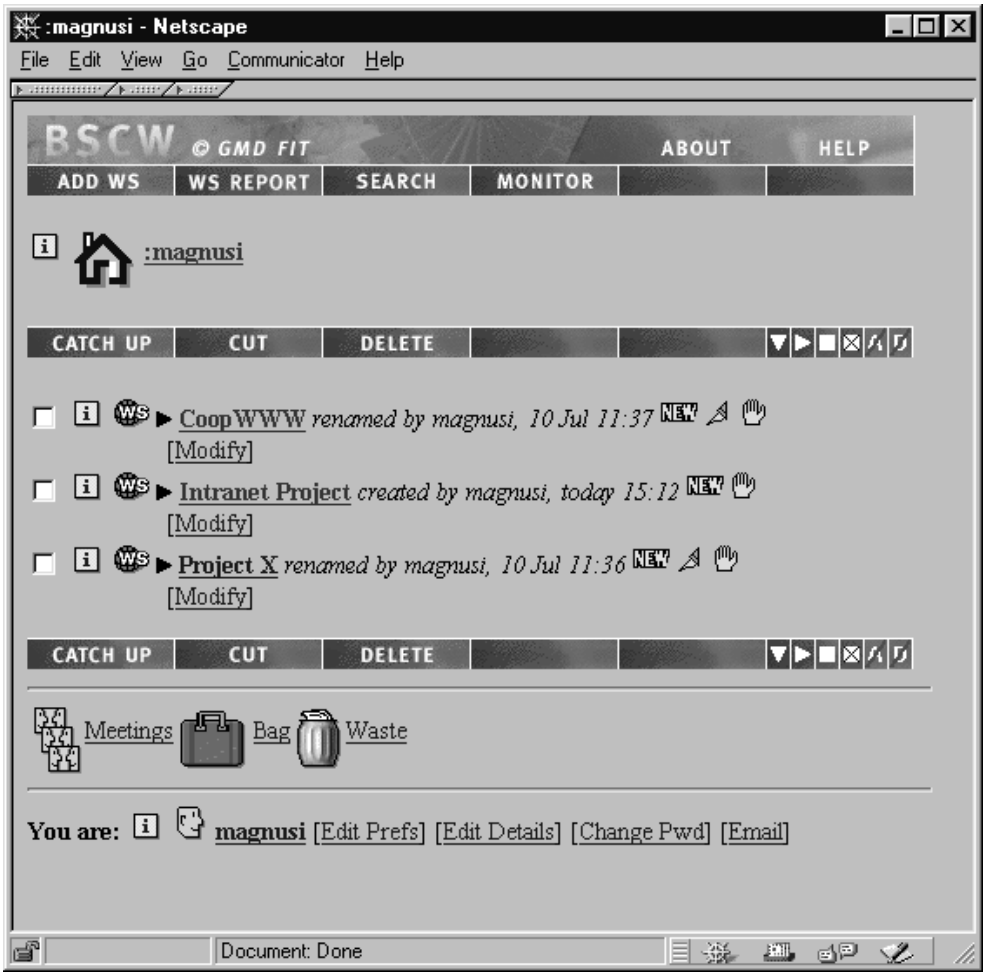

Figure 7.1: *Extension with a "Meetings" icon. Clicking the icon will display the user's personal list of meetings.*

In the BSCW 3 implementation, when a user is examining the contents of a workspace or some folder, the bottom navigation row contains a mix of personal areas (*bag* and *waste*basket), non-personal-areas (*members*). The addition of the calendar icon, and also possibly other future extensions, highlights the potential problem with this approach in terms of user navigation, but in the short run this approach seems adequate.

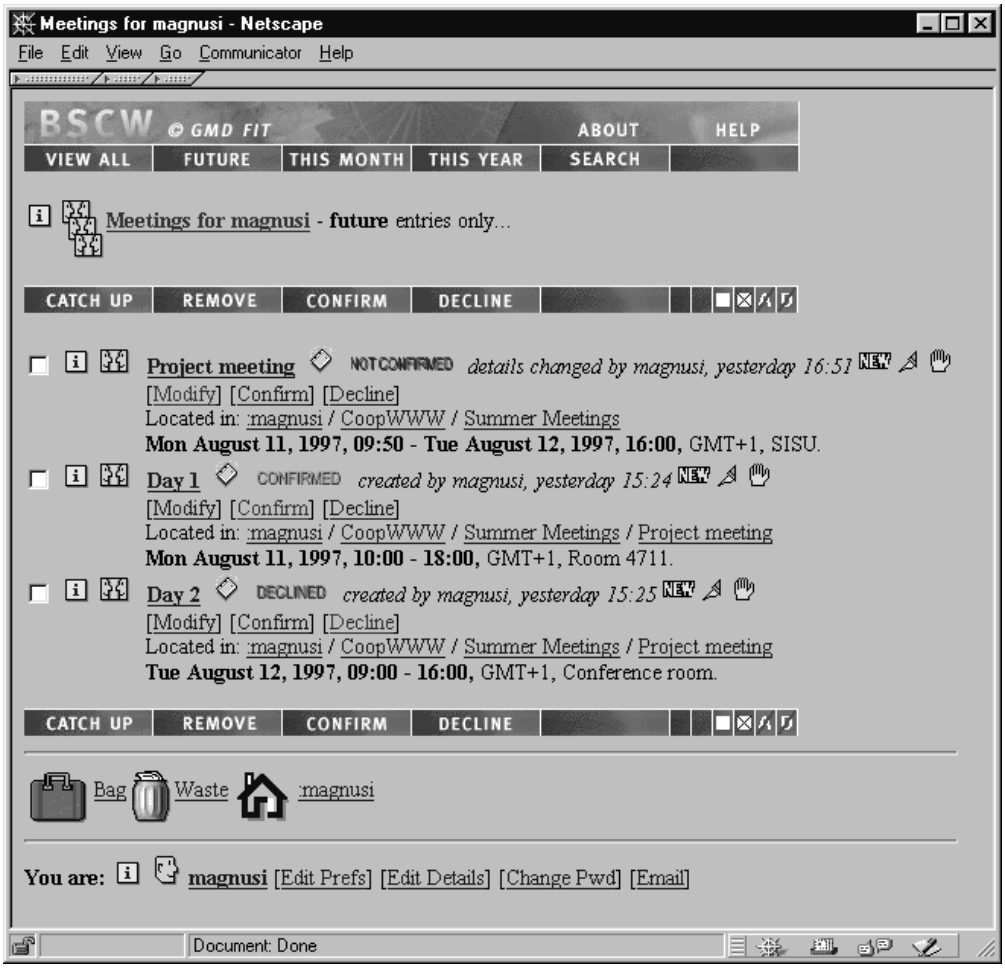

Figure 7.2: *The personal list of meetings as it appears upon clicking on the "Meetings"-icon.*

7.4.2 Representation of Meetings in a Workspace

Since a meeting is an event that is scheduled to take place at a particular point in time, the presentation of meetings is best implemented by a sequential list where meetings are sorted according to their scheduled starting times.

However, in the proposed model, a meeting can be created anywhere in the workspace hierarchy. To overcome the potential lack of overview of scheduled meetings, these are listed in the personal meeting list, in a time sequential order, or a more calendar-like fashion. The initial design of the meeting list should be fairly simple and straight-forward.

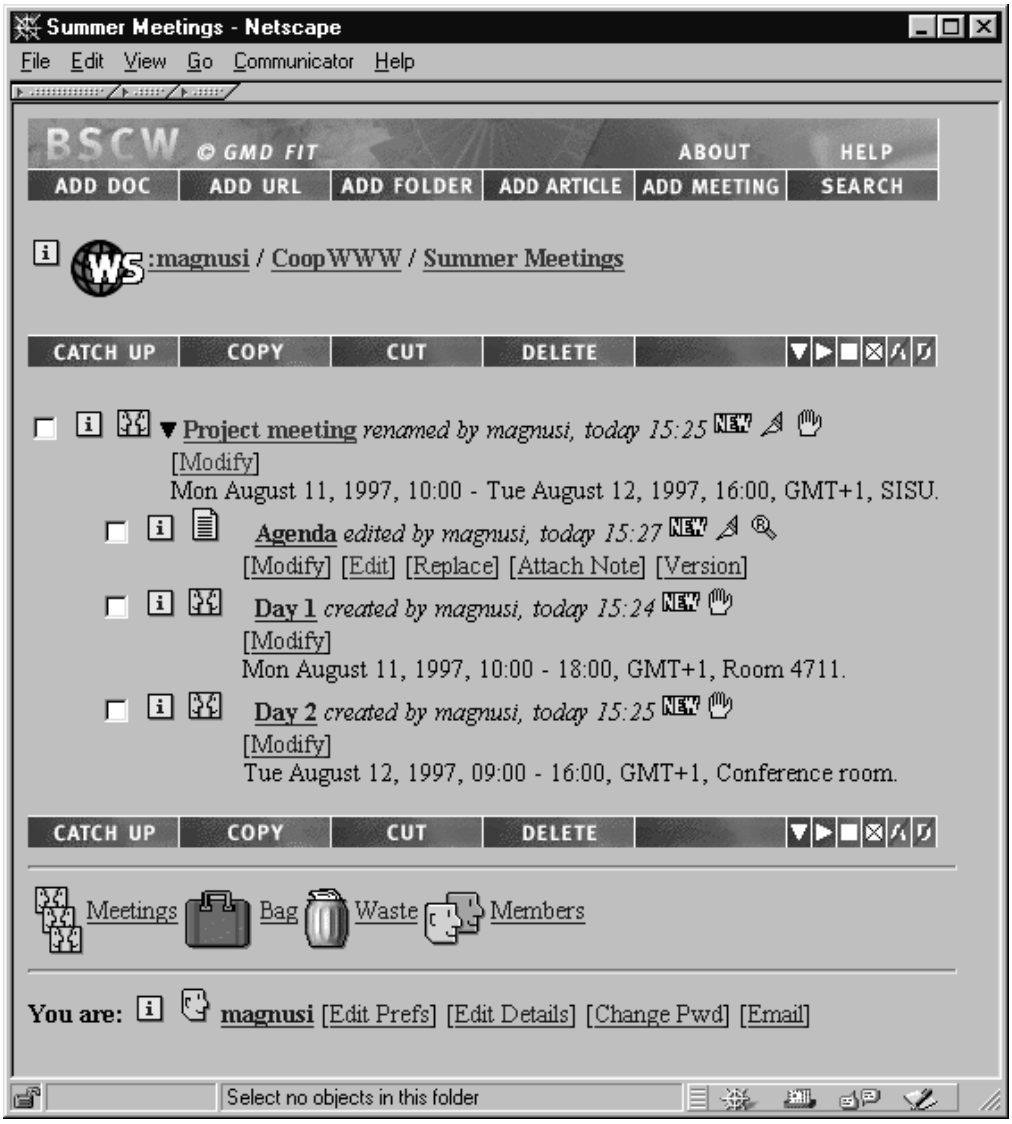

Figure 7.3: *A workspace with meeting objects. Note that there are objects inside the top-level meeting object (Project Meeting).*

Figure 7.3 above shows a workspace with a number of meeting objects. Note that a meeting can be located inside another meeting, and that a meeting can be folded out, just like a folder or an article. Clicking on a meeting object in the workspace listing results in a view of the meeting and its contents, as shown in the figure below.

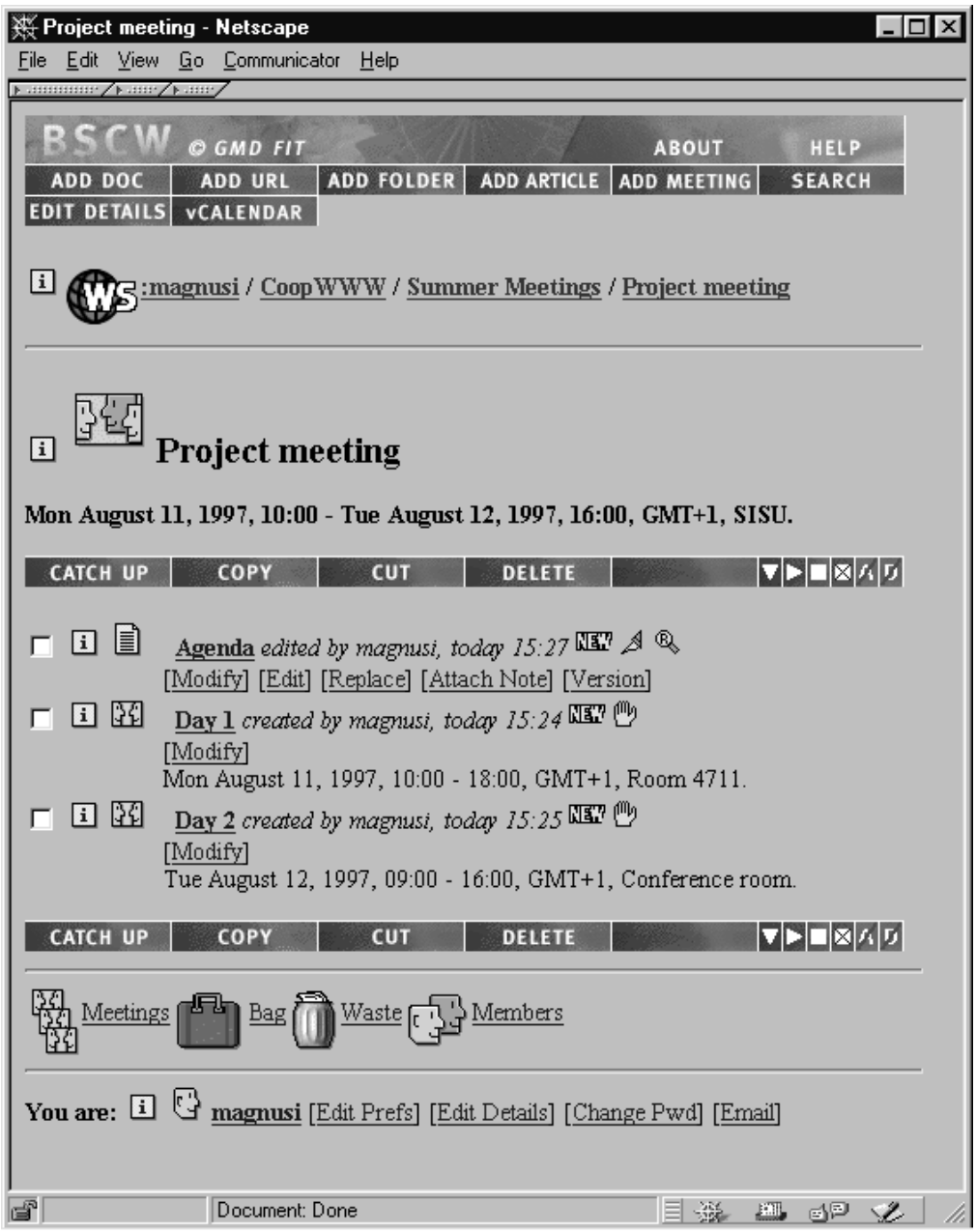

Figure 7.4: *Displaying a meeting object and its contents in a workspace.*

7.4.3 Creating a Meeting

In this implementation, a meeting is created anywhere in a workspace by pressing the *Add Meeting* button. When creating a meeting, the following information may be specified by its creator:

- start date and time (required),
- end date and time (required),
- time zone (required),
- location (required, e.g. WhitePine Reflector, IP address, geographical location, etc.),
- type (required, e.g. telephone, face-to-face/physical, video-conference, etc.),
- participants (optional, should be possible to add later).

All details concerning a meeting, as listed above, can be changed at a later stage, but only by someone who has the right to do so (by default the creator, i.e. the owner). In general the

default access control for meeting objects should be the same as for any other workspace objects. Note that it is not necessary to specify the participants of a meeting upon the actual creation. These can be added at any stage subsequent to the creation.

The figure below shows how the “add meeting” form can be implemented. This form will have a relatively large number of fields, since the details required for a meeting object are quite numerous. Nevertheless, the design of the user interface should aim to facilitate the process of creating a new meeting as far as possible, e.g. by pre-specifying some fields with expected values. The figure below only shows the top part of the prototype form; the bottom part is shown in a separate figure following this one.

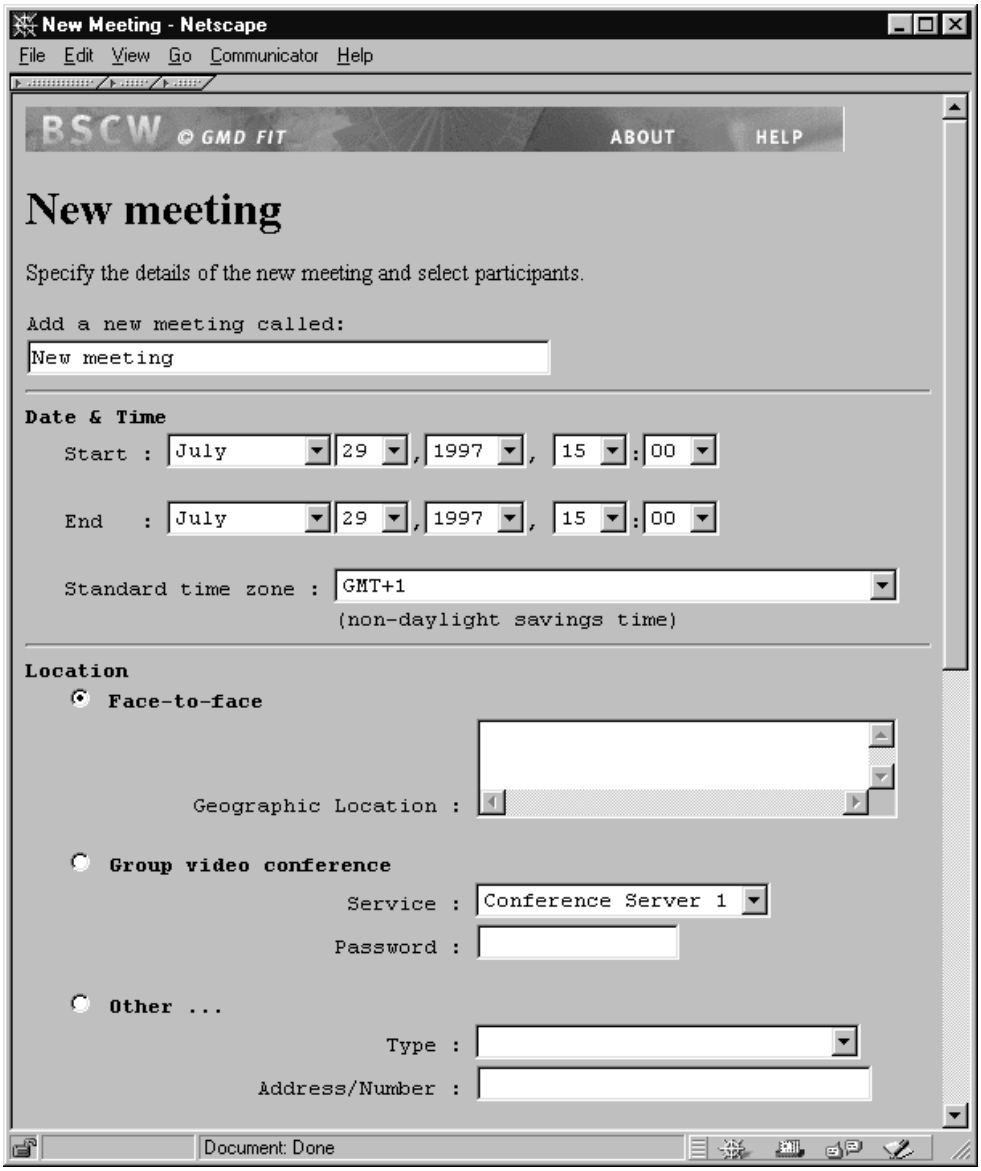

Figure 7.5: *Creating a meeting by specifying meeting details. See next figure for the bottom part of the add meeting form.*

The previous figure shows how the user may specify what type of meeting is created. The next figure shows how participants may be added. There is also a check box for automatic email notification and a field for adding some comments.

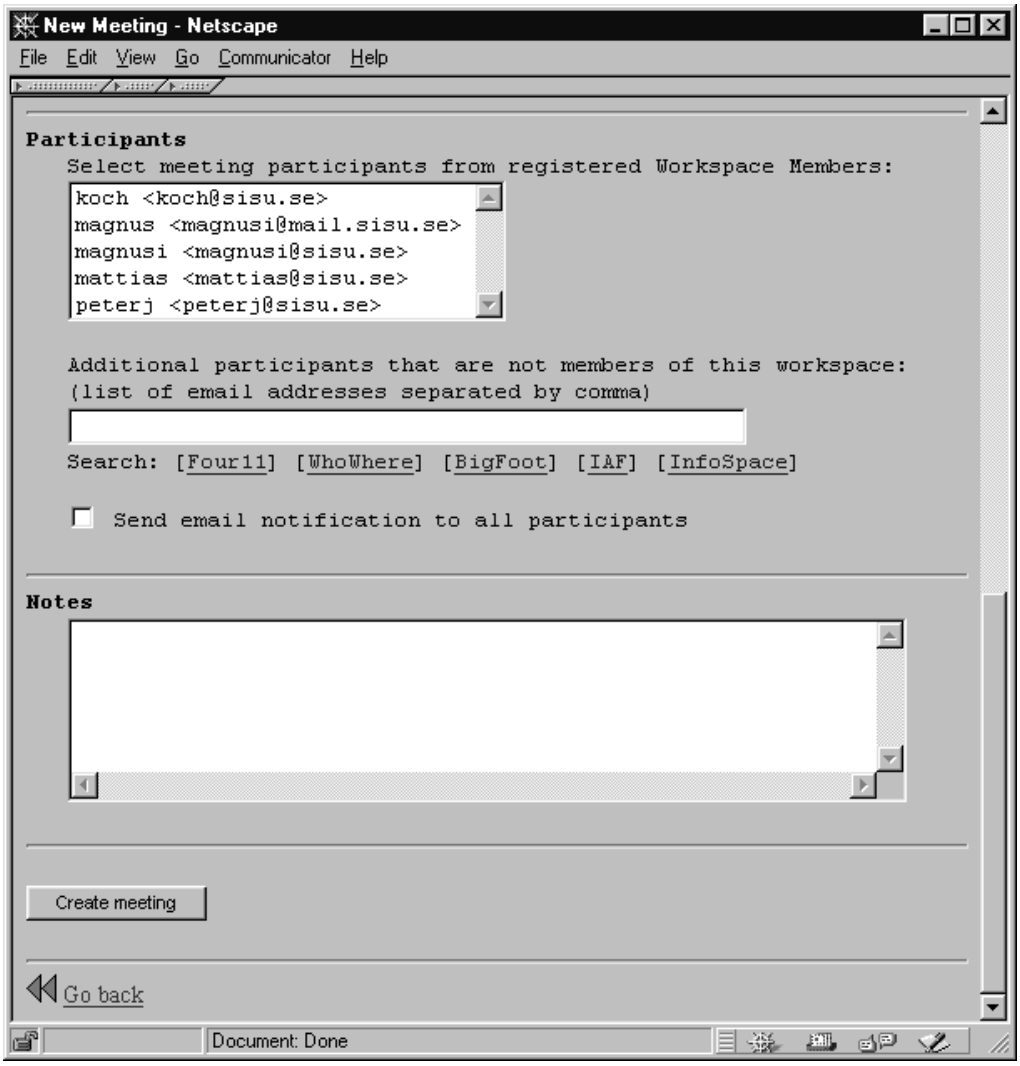

Figure 7.6: *The bottom part of the add meeting form.*

It is important to understand that the service being offered through the meeting scheduling does not aim to solve the general problem of scheduling events and guaranteeing that all participants will be available at a particular time. Therefore, no automatic scheduling methods (based on participants' available free time) should be provided. It is the author's personal belief that automatic such automatic scheduling does not work well in reality. However, there are organizations using this and claiming that it works well, but it is rather safe to assume that this is more an exception than a rule.

7.4.4 Inviting Participants to a Meeting

Participants can be invited to a meeting in the following manners:

- by the creator of a meeting upon the actual creation of the meeting;
- by the creator or an invited participant an unlimited number of times subsequent to the creation.

For meetings created inside a workspace, only members of that workspace can be invited, due to restrictions in the implementation of the BSCW access control scheme.

7.4.5 Joining a Meeting

Depending on the type of meeting there may be ways supported by the BSCW system to join that meeting, e.g. for video-conference meetings. For traditional meeting types such as a telephone conference over POTS or a regular face-to-face meeting, the status of such a meeting and the process of joining it is clearly out of control of the BSCW system. However, for a meeting that is to take place over the network there should be built-in support for the actual process of joining a meeting.

The join operation on a meeting object is only allowed on a meeting, in the case where a special MIME type can be associated with the meeting. The "Join" option should thus not be present in the user interface for "non-joinable" meetings.

For the rest of this paragraph, we will assume that the meeting is a video-conference to be hosted by an integrated CU-SeeMe reflector, to illustrate the general methodology. However, the meeting could be any other type of point-to-point or multi-point meeting. The procedures are exactly the same for other types of applications, where it is possible to spawn the client program and send parameters (such as an IP or host address), while doing so.

Provided that the client machine has the appropriate software installed (WhitePine Enhanced CU-SeeMe), and that the WWW browser has been configured to launch this software upon the reception of a MIME type of application/x-cu-seeme, the following will happen:

1. CU-SeeMe is automatically launched at the client.
2. A connection attempt to the integrated reflector is automatically made.
3. If successful, a list of *conference IDs* will be presented.
4. The user selects and joins the appropriate conference.

In addition, there may be a password associated with the conference in question, that the user has to have knowledge of and that has to be submitted before joining the conference.

7.4.6 Removal of Meetings

In this implementation, a meeting object is handled in a similar manner as any other object in a workspace. Thus, it can be deleted and later destroyed, but only by someone who has the rights to do so (by default its creator).

Since it is possible to add any object inside a meeting object, the meeting object may still be useful even after the meeting has taken place. For example, an agenda, meeting minutes and other documents can be placed inside a meeting, and the meeting object therefore serves as the access point to these documents. Meetings should thus not be automatically removed after having occurred.

However, the user interface representation of a meeting could change slightly after the meeting has taken place, to make the distinction between future and past meetings more visible to the user. This applies both to the meeting objects themselves as well as to their representation inside a user's personal meeting list.

7.5 Ad-hoc Meeting Services

All the objects and operations described up to this point are support mechanisms for pre-arranged, scheduled meetings. However, there should also be mechanisms to quickly initiate a synchronous communication session with another user, without the need for scheduling or advance planning. These operations are described in further detail in this section. Generally speaking, the provision and implementation of support for ad-hoc meetings is not as complicated as pre-scheduled meetings and calendars. However, there are several design decisions that have to be made, which will lead to a number of required modifications to the system.

The basis for ad-hoc meeting support should initially be provided be through the member's page. The member page scheme could be extended in several ways to support quick initiation of synchronous collaboration. These extensions are described in more detail in the following sections.

7.5.1 Extension of the User Details

The BSCW kernel provides the user with a number of basic facilities to update and change such properties as password, preferences, and other useful personal information. The User details page, where the user can change the personal details, could be extended with fields to let the user specify a preferred IP address (or domain name) and available communication capabilities. A number of different third party tools currently exist that support automatic launch and connect, e.g.:

- WhitePine Enhanced CU-SeeMe,
- Netscape CoolTalk,
- Connectix VideoPhone,
- VDOnet VDOPhone.

These tools are automatically launched by returning a specific MIME-type associated with the software and by passing certain parameters on a certain format, which is different for every application.

These and similar tools should be supported by the system, but it could also be possible for the user to specify basically any software, provided that there is a MIME-type associated with it and that it can be kicked into automatically connecting by passing it parameters on a certain format.

This extra information about a user could be entered into the system by clicking the "Edit details" from the member page or from any workspace page. This action results in a form being displayed where the user can enter all personal information. This form could be extensively enhanced to incorporate the extra information as described above.

Every user in the BSCW system is represented by a User object in the persistent object store. The User object has to be modified and extended to hold this additional information.

7.5.2 Extension of Member Information

Clicking the info button of a user or the hyper-linked username in a workspace, results in the member information page of that user being displayed. The member information page contains details such as the user's real name, phone number, address, etc. This page could be

extended with details about a user's communication capabilities, IP address, etc, as specified by the user through the user details page.

The member information page could also contain links to automatically launch and connect different third party tools to the specified user.

In the following sections, a possible user interface design for an ad-hoc meeting service is presented. The screen dumps in the following sections are from a simple prototype implementation.

7.5.3 Specifying Personal Communication Capabilities

In the prototype, the personal communication capabilities are specified by clicking the "Edit details" button from a user's own info page or by clicking "Edit details" displayed on the bottom of every workspace page. This action will result in the member details page being displayed, with all the necessary fields for specifying the personal communication capabilities, as seen in the figure below.

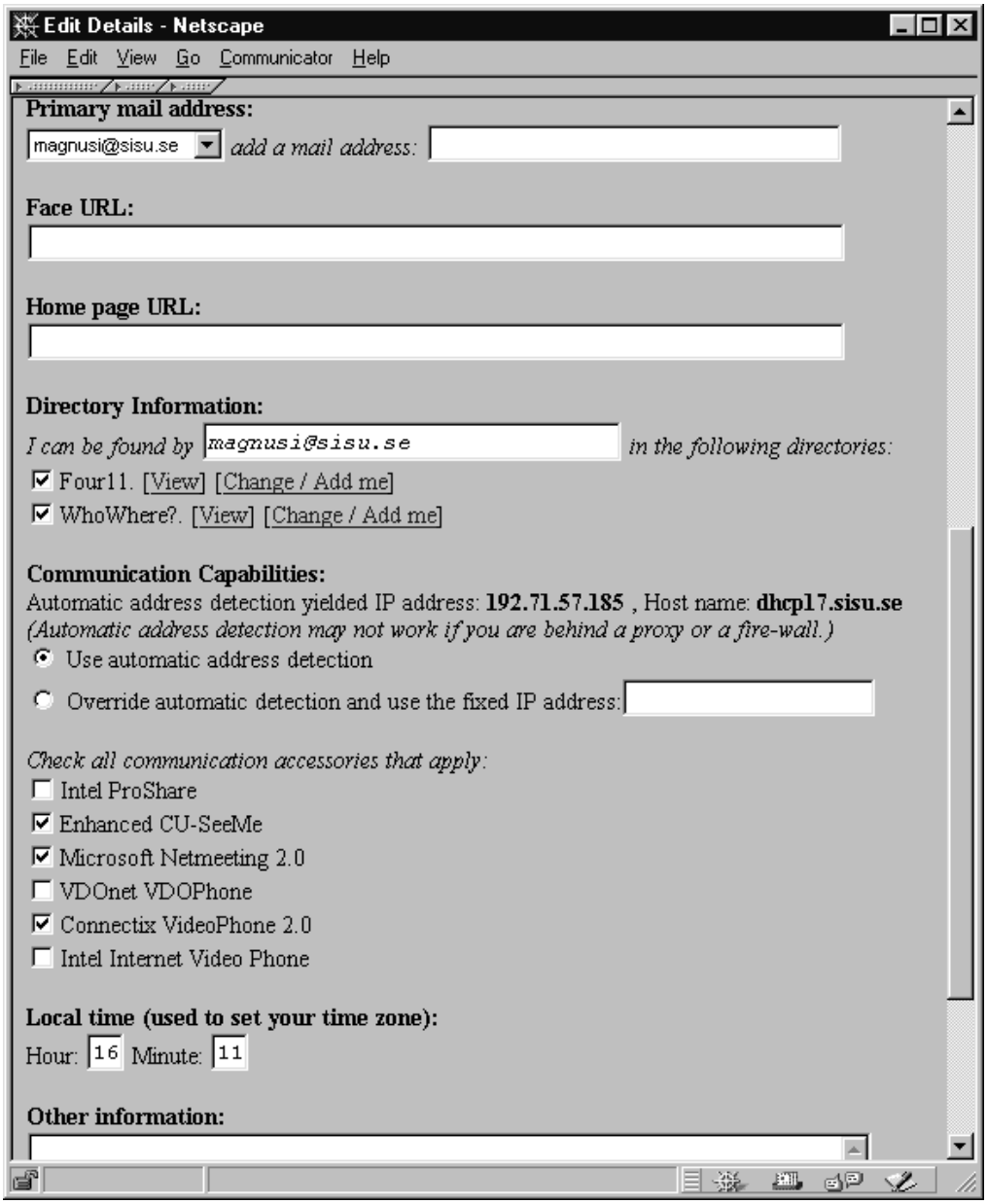

Figure 7.7: *The proposed extension of the "Edit details" page. A user can specify an IP address and all available communication accessories that are supported by the system.*

7.5.4 Accessing a User's Communication Capabilities

In the prototype, a user's communication capabilities can be accessed via that user's member info page. The member info page is also the access point to ad-hoc synchronous collaboration sessions. On this page it should be possible to just click on a link and automatically launch and connect the appropriate software tools, as seen in Figure 7.8 below.

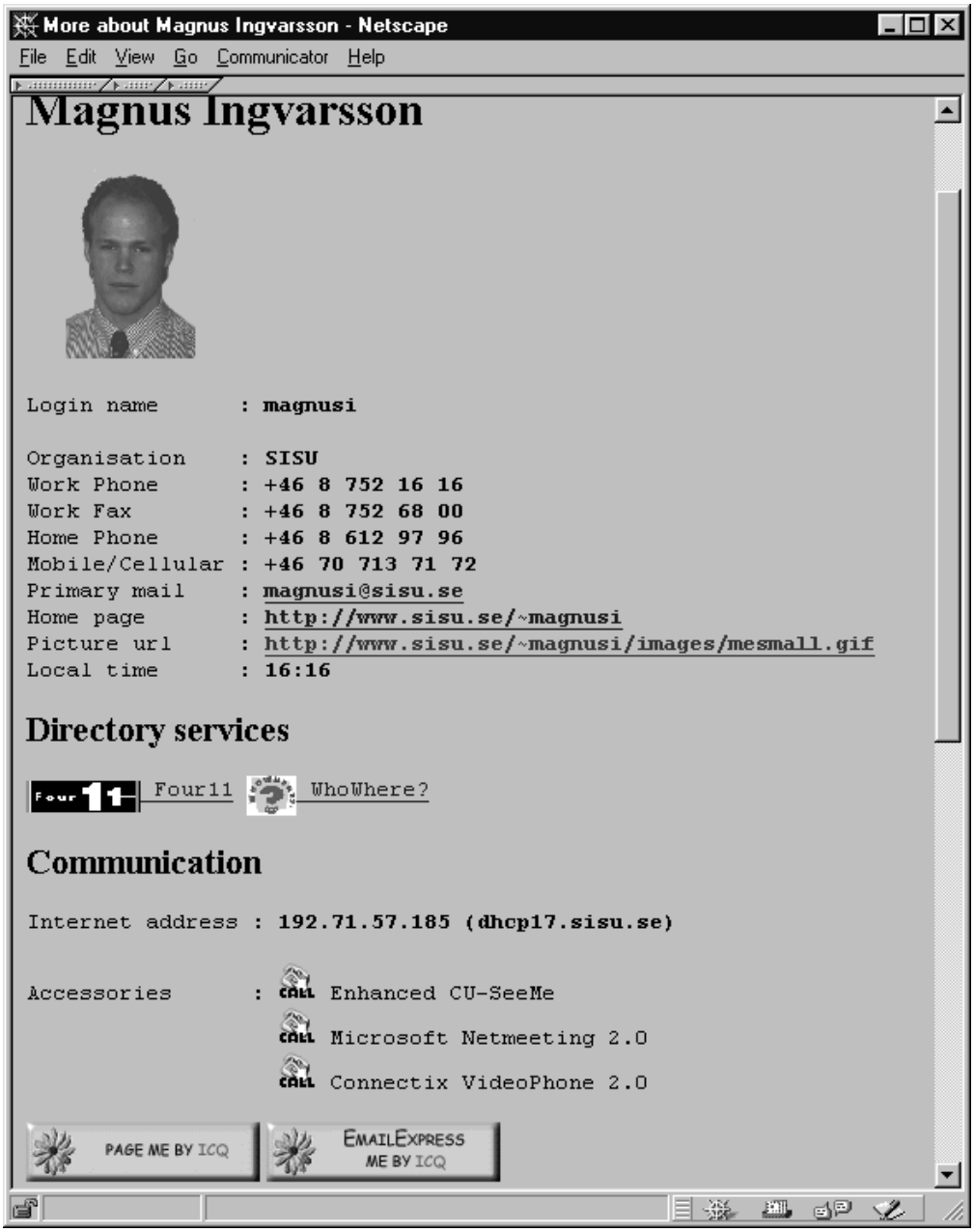

Figure 7.8: *Accessing a user's communication capabilities and methods to launch and automatically connect different communication tools.*

7.5.5 Launching a Synchronous Session with Another User

The actual launching of the correct client applications and the connections to the specified user are made by clicking on a link on a particular user's member info page, as in the previous figure. This will make the system return an appropriate MIME type, and if the client WWW browser is configured to handle this MIME type correctly, the right client applications will be automatically launched and connected to the user.

7.6 Creating Presence Awareness

The BSCW user interface needs to be extended to make the user aware of who is currently "in" or "online" and likely to be available for communication. The concept of "in" in this context is hard to define. In this section we assume, however, that such a function is indeed implemented (likely in Java), and that the basic user interface consists of a list of users and

their current status (*available*, *busy*, *away*, etc.), as in the figure below showing an actual prototype for this particular purpose.

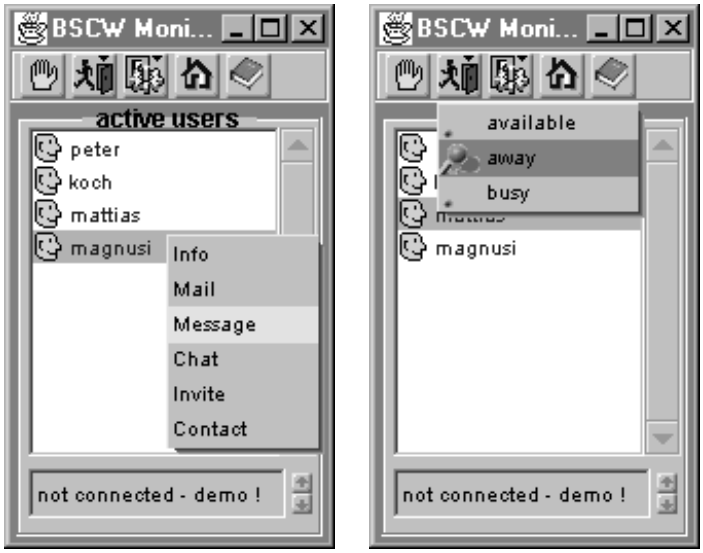

Figure 7.9: *A Java prototype of a simple presence awareness service.*

Previous work in this area has assumed that there is a need for being able to tell who else is currently on the same Web page. Upon reflecting on this approach for some time, it becomes obvious that this is a rather dubious concept. First of all, there is technically no simple way of extracting this information. Furthermore, people do not often stay on the same page for a very long period of time and even if they did, what can this information be used for? Even if a user is not on the same page (or in the same workspace for that matter), this person might still be sitting at his computer and is indeed available for some sort of communication.

A useable approach to this problem apparently needs to be separated from the Web interface of the shared workspace service itself. Some consequences may be that the user online service will require some extra login procedures and active participation by the user. However, this need not be a disadvantage, since this is approach is more likely to conform to users' current mental models of how online notification services normally work. Examples of other applications or services trying to solve this problem include the rather well known ICQ™ by Mirabilis Ltd., iPage by ichat Inc., and Excite PAL by Excite and Ubique Ltd. However, all these applications operate on a global scale, and thus, do not naturally fit into a closed environment like a shared workspace.

Assuming that such a service is indeed implemented, clicking on a user's name or picture in a list should provide a link to display a presentation of that particular user's communication capabilities.

7.6.1 Presenting User Communication Capabilities

User communication capabilities can be integrated into the current user interface in a number of different ways. Independently of implementation, the additional information storage is likely to include:

- IP address for CU-SeeMe, InternetPhone, or other TCP/IP-based communication tools,
- ISDN telephone number for Intel Proshare, PictureTel or other conferencing tools.

7.6.2 Initiating User Communication

Independently of where and how user communication capabilities are presented, they should be extended with behavior; i.e. clicking on a call button for a certain accessory should result in actions being taken to try to initiate that type of communication. How this will be implemented is discussed in Section 4.

7.6.2.1 Example Scenario

User A is currently doing something in the workspace. A notices that user B has recently been active in the workspace. A wishes to discuss a certain topic with B. A clicks on B's picture and is presented with B's communication capabilities. A notices that B has CU-SeeMe capability, which A also happens to have. Therefore, A clicks on the *call* button for CU-SeeMe to initiate a CU-SeeMe session with B. CU-SeeMe is launched on A's computer with an attempt to connect to the IP-address specified by B. If B happens to be available, he can take adequate actions to respond.

The steps (user actions) involved in this scenario can be summarized as follows:

1. Is the person likely to be available for communication?
2. What are his/her communication capabilities?
3. What are my communication capabilities?
4. Initiate communication.
5. Wait for response (success or failure).

This scenario launches the client communication software automatically without any additional input by the user. The user control and interference is therefore minimized. The scenario will be similar for other types of communication.

7.7 Possible Implementation Architectures

The implementation of the meeting object, the personal meeting list and the integration with the BSCW kernel assume very little in terms of the structure of the user interface. Operations on meeting objects can be performed independently of the access method to the meeting services.

As a result of this, the meeting service could be implemented using the following technology:

- CGI programming and generation of user interface presentations in HTML,
- the inclusion of one or more Java applets and/or JavaScript scripts.

Independently of which technology is incorporated, any used metaphors and the user's mental model of the system and its functionality should be similar. This partly allows the general mechanisms of the user interface to be designed independently of the final implementation technology.

The meeting object is implemented like any other object in the system, and the user interface to interact with meeting objects is basically the same as the rest of the system. Thus, no additional client capabilities will thus be necessary for basic interaction with meeting objects.

However, the personal meeting list could be implemented and presented to the user in a more "calendar-like" fashion, to facilitate browsing calendar entries. The basic interaction scheme of the BSCW system, with the file-system browsing metaphor, may not be suitable for a calendar presentation. Initially, the design should be very simple and straight-forward. The

initial goal should be to make clear to the user what the meeting list is, and to design the look-and-feel of the list to make it distinct from workspace listings.

Special tasks, like booking a meeting could be implemented using more advanced technology such as Java and JavaScript. This will require a modern browser such as Netscape 2.0 (or higher) or Microsoft Internet Explorer 3.0 (or higher). However, by the time of the planned release of the extended BSCW system, a vast majority of the potential users will possess this required client technology. In a subsequent re-design, a majority or all of the functionality may be implemented using more advanced features. The usage of this technology should however be limited initially, due to immature performance and differences between different browsers (both within the same browser line and across different vendors, e.g. Netscape and Microsoft). However, the user should be able to turn off these advanced features by a simple toggle function, and still be able to access the basic workspace services.

In general, the conditions for implementing extended functionality in WWW, e.g. support for synchronous communication, constantly changes. The development of WWW client technology, in particular, is very intense and rapid. This results in ever-changing conditions and architectures to exploit and comply with. A study of different approaches to implement extended support for synchronous communication yields the following list of different architectures available:

- *Client plug-ins,*
- *Helper applications configured as viewers,*
- *Java,*
- *JavaScript,*
- *ActiveX,*
- *Server CGI scripts.*

Note that all other architectures are ruled out, based on the restrictions set forth in the previous section. Of the architectures listed above, the last three or four are best suited for specialized tasks, such as extensions to the BSCW user interface.

8 CONCLUSIONS

The most important conclusion to come out of this paper is the importance of *integration*. The choice of conferencing technologies is of course not unimportant, but the integration of these tools into other collaborative systems is the key factor to successful deployment.

The market for audio-, video-, and data-conferencing is developing so fast that any commitment to a particular conferencing technology is currently unrealistic. The general consensus within the industry for the need of standards and protocols is very positive, and long overdue, but this fact alone cannot justify a commitment to conferencing architectures that comply with the proposed standards. The immense proliferation of proprietary standards and the diversity of open standards undoubtedly causes major “head-ache” with IS managers looking to invest in technology for their corporate needs, or ISPs looking to provide additional services to their customers. Most of these people will today choose a “wait-and-see” approach, to reduce the risk of investing in technology that will be obsolete within a year.

The impact of this reasoning with respect to the integration of synchronous communication tools into the BSCW system is significant. In the same way as the IS managers cannot make a definite choice of technology, the integration of synchronous communication facilities into BSCW cannot make this choice either. Therefore, the mechanisms for integration of synchronous tools into the BSCW system, or any Web-based collaborative system, should be designed with a very generalized approach, so that that particular synchronous tools can be replaced without the need for a total re-design. In fact, the integration of synchronous facilities into the World Wide Web should probably be designed to specifically support the incorporation of different conferencing technologies and applications as they become available.

In addition, there is still controversy whether transmission of real-time audio and data over the Internet is really a viable way of doing things. The first generation of audio- and video-conferencing tools is often experienced as providing very low quality transmissions. The basic dilemma is that packet switching is inherently unreliable, and there is no way to guarantee when or if packets will arrive to their intended receiver. This makes implementing applications where real-time transmission of data is necessary a very difficult task. There is also a potential risk of severe network congestion problems when the technology is made available to a larger public and the popularity of these tools take off in a major way.

9 SUGGESTIONS FOR FURTHER WORK

One of the main stumbling blocks of synchronous communication tools is the complex task of keeping track of users' current location, IP addresses, and available communication facilities.

Provided that a system can be designed to keep track of this information, and that there are effective mechanisms of presenting this information to other users, there are several other functions that are needed in order to have a truly effective and comprehensible infrastructure for easy-to-use collaboration.

Areas that have not been investigated in depth or publicly documented in any significant way include, for example:

- *Communication facility negotiation*. In a perfect world, a person wishing to communicate with another person somewhere in the world, should not have to be concerned about that particular person's available hardware, software, and ability to communicate. A client-to-client negotiation, possibly handled via some server, of available hardware and software could automatically pick the appropriate and most suitable tools for the specific synchronous session at hand and the task to be performed.
- *Generalized User Location Services.* Today, almost every vendor of Internet-based audio- and/or video-conferencing applications are running one or more ULSs of their own, in order for the users' of their software to find people to call. This approach, while surely effective for the particular conferencing tool it supports, rapidly becomes unbearable when using half a dozen or more different conferencing applications. There are also severe scalability problems with this approach, since these ULSs are likely to be heavily overloaded when the number of users increases.

REFERENCES

Note: Most references below refer to online documentation. In most of these cases the references are available exclusively online. In cases where the reference is available both in print and on-line, this is specifically stated. In those cases where the reference is available at multiple online locations, the one listed is believed by the author to be the most reliable. If not, a simple search query on an Internet search facility is likely to produce alternative locations.

Appelt, W., Busbach, U. (1996), *The BSCW System: A WWW based Application to Support Cooperation of Distributed Groups*, presented at WET ICE 96: Collaborating on the Internet: The World-Wide Web and Beyond, Stanford University, June 19-21, 1996, http://orgwis.gmd.de/~busbach/wetice.ps

Bank, J. A., (1995), *Java Security*, 1995, http://www-swiss.ai.mit.edu/~jbank/javapaper/javapaper.html

Bell, G., Parisi, A., Pesce, M. (1995), *The Virtual Reality Modeling Language, Version 1.0 Specification*, November 9, 1995, http://www.vrml.org/VRML1.0/vrml10c.html

Bentley, R., Appelt, W., Busbach. U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J., Woetzel, G. (1997), *Basic Support for Cooperative Work on the World Wide Web*, in International Journal of Human-Computer Studies: Special issue on Innovative Applications of the World Wide Web, Spring 1997, Academic Press, http://bscw.gmd.de/Papers/IJHCS/IJHCS.html

Bentley, R., Horstmann, T., Trevor, J. (1997), *The World Wide Web as enabling technology for CSCW: The case of BSCW*, in Computer-Supported Cooperative Work: Special issue on CSCW and the Web, Vol. 6, 1997, Kluwer Academic Press, http://bscw.gmd.de/Papers/CSCWJ-WWW/CSCW_journal.html

Bentley, R. and Appelt, W., *Designing a System for Cooperative Work on the World-Wide Web: Experiences with the BSCW System*, in Proceedings of HICSS'30: The Hawaii International Conference on the System Sciences, Maui, Hawaii, 7-10 January 1997, IEEE Computer Society Press, http://bscw.gmd.de/Papers/Uni-Twente/BSCWEval.ps

Bentley, R., Busbach, U. and Sikkel, K. (1996), *The architecture of the BSCW Shared Workspace System*, in Proc. 5th ERCIM/W4G Workshop "CSCW and the Web", GMD, Sankt Augustin, April 1996, pp. 31-42, http://orgwis.gmd.de/W4G/proceedings/bscw.html

Bentley, R., Horstmann, T., Sikkel, K. and Trevor, J. (1995), *Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System*, in The World Wide Web Journal: Proceedings of the 4th International WWW Conference, Boston, Massachusets, Issue 1, December 1995, pp 63-74. O'Reilly & Associates, http://bscw.gmd.de/Papers/WWW4-Boston/WWW4-Boston.html

Berners-Lee T., Fielding R., and Frystyk H. (1996); *Hypertext Transfer Protocol -- HTTP/1.0*; Network Working Group; May 1996, ftp://ds.internic.net/rfc/rfc1945.txt

Bigfoot Partners L.P., http://www.bigfoot.com

Bolot J., Hoschka P. (1996), *Sound and Video on the Web*, INRIA-Sophia Antipolis, in Tutorial at the 5th WWW Conference, Paris, May 6-10, 1996, http://www.inria.fr/rodeo/personnel/hoschka/WWW5tutorial.ps.gz

Borenstein N., Freed N. (1993); *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*; Network Working Group; September 1993; ftp://ds.internic.net/rfc/rfc1521.txt

BSCW Project, GMD-FIT, Sankt Augustin, Germany, http://bscw.gmd.de

BSCW Project (1997), *BSCW User Manual, 1997*, GMD-FIT, Sankt Augustin, Germany, http://bscw.gmd.de/Help/

CoopWWW Project, http://orgwis.gmd.de/projects/COOPWWW/

Dix, A., Finlay, J., Abowd, G., Beale, R. (1993), *Human-Computer Interaction*, 1993, Prentice Hall International (UK) Limited, University Press, Cambridge.

Excite, http://www.excite.com

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T. (1997), *Hypertext Transfer Protocol -- HTTP/1.1*, January 1997, RFC-2068, Network Working Group, ftp://ds.internic.net/rfc/rfc2068.txt

Four11, http://www.four11.com

Freed, N., Borenstein, N. (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC-2045, Network Working Group, ftp://ds.internic.net/rfc/rfc2045.txt

Frivold, R., Lang, R. and Fong, M. (1995), *Extending WWW for Synchronous Collaboration*, in Electronic Proceedings of Second WWW Conference: Mosaic and the Web, http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/CSCW/frivold/frivold.html

Gosling, J., Joy, B., Steele, G. (1996), *The Java Language Specification Version 1.0*, August 1996, Sun Microsystems Inc. http://java.sun.com/docs/books/jls/html/index.html

Gosling, J., McGilton, H. (1996), *The Java Language Environment: A White Paper*, Sun Microsystems Inc., May 1996, http://java.sun.com/docs/white/langenv/

Gosling, J., Yellin, F. (1996), *Java API Documentation Version 1.0.2*, The Java Team, Sun Microsystems Inc., April 1996, http://java.sun.com/products/jdk/1.0.2/api/

Hill J., Ozer J., Mace T. (1996); *Collaboration on the Web, Real-Time Collaboration*; PC Magazine Online; October 1996, http://www.pcmag.com/iu/features/1517/rt-intro.htm

Howes T., Smith M. (1995); *The LDAP Application Program Interface*; Network Working Group; August 1995; ftp://ds.internic.net/rfc/rfc1823.txt

ichat, http://www.ichat.com

IETF, Internet Engineering Task Force, http://www.ietf.org

IMC, Internet Mail Consortium, http://www.imc.org

ITU, International Telecommunication Union, http://www.itu.ch

Javasoft, http://www.javasoft.com

Kantor B., Lapsley P. (1986), *Network News Transfer Protocol*; Network Working Group; February 1986; ftp://www.internic.net/rfc/rfc977.txt

Kessler, G., Shepard, S. (1997), *A Primer On Internet and TCP/IP Tools and Utilities*, RFC-2151, Network Working Group, June 1997, ftp://ds.internic.net/rfc/rfc2151.txt

Lotus, http://www.lotus.com

Lotus (1997), *Lotus Domino 4.5, Powered by Notes: Part I -- Technical Overview*, May 1997, http://www2.lotus.com/worktheweb2.nsf/d0824a6016946414852564520070 8c99/887f9d826375c4b7852564790050aa93?OpenDocument

Lutz, M. (1996), *Programming Python*, First Edition, October 1996, O'Reilly & Associates, Inc., California.

Microsoft Corporation, http://www.microsoft.com

Mirabilis Inc., http://www.mirabilis.com

NCSA, University of Illinois at Urbana - Champaign, IL, USA, *NCSA HTTPd Documentation*, http://hoohoo.ncsa.uiuc.edu/docs/

NCSA, University of Illinois at Urbana - Champaign, IL, USA, *The Common Gateway Interface Specification*, http://hoohoo.ncsa.uiuc.edu/cgi/interface.html

Netscape Communications Corporation, http://www.netscape.com

Netscape Communications Corporation, *JavaScript Authoring Guide*, http://home.netscape.com/comprod/products/navigator/version_2.0/script/script_info/index.html

Netscape Communications Corporation (1995), *Plug-in Design Specifications*, 1995, http://home.netscape.com/comprod/development_partners/plugin_api/plugin_design.html

Oikarinen J., Reed D. (1993), *Internet Relay Chat Protocol*; Network Working Group; May 1993; ftp://ds.internic.net/rfc/rfc1459.txt

Opper S., Fersko-Weiss H. (1992), *Technology for Teams - Enhancing Productivity in Networked Organization*, Van Nostrand Reinhold, New York, N.Y.

Python Software Activity, http://www.python.org

Rossum, van G. (1996), *Documentation for Python: Tutorial, Library Reference, Language Reference, Extension Manual*, 1996, http://www.python.org/doc/

Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V. (1996), *RTP: A Transport Protocol for Real-Time Applications*, RFC-1889, Audio-Video Transport Working Group, Network Working Group, January 1996, ftp://ds.internic.net/rfc/rfc1889.txt

Schulzrinne, H. (1996), *RTP Profile for Audio and Video Conferences with Minimal Control*, RFC-1890, Audio-Video Transport Working Group, Network Working Group, January 1996, ftp://ds.internic.net/rfc/rfc1890.txt

Sun Microsystems Inc., http://www.sun.com

Sun Microsystems Inc., *Java™ Language Overview*, http://java.sun.com/docs/overviews/java/java-overview-1.html

Trevor, J., Koch, T., Woetzel, G., *MetaWeb: Bringing synchronous groupware to the World Wide Web*, in Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW'97), Lancaster, September 1997, Kluwer Academic Publishers, http://bscw.gmd.de/Papers/ECSCW97-MetaWeb/MetaWeb.html

versit Consortium (1996), *vCalendar, The Electronic Calendaring and Scheduling Exchange Format, Version 1.0*, September 18, 1996, http://www.imc.org/pdi/vcal-10.doc

versit Consortium (1996), *vCard, The Electronic Business Card, Version 2.1*, September 18, 1996, http://www.imc.org/pdi/vcard-21.doc

VRML Consortium, http://www.vrml.org

VRML Consortium (1997), *The Virtual Reality Modeling Language Specification, ISO/IEC DIS 14772-1*, VRML/ISO Draft for International Standard (DIS), April 4, 1997, http://www.vrml.org/Specifications/VRML97/DIS/

Walther, M. (1995), *Multi-User Web Browser*, 1995, http://www.dstc.bond.edu.au/~walther/groco/multiweb.html

Walther, M. (1995), *Implementation Options for EMS in Java*, 1995, http://www.dstc.bond.edu.au/~walther/groco/javaEMSArchitectures.html

Walther, M. (1995), *Registration Issues in Shared Interactive Pages: Run-time Architecture and Generic Registration Protocol*, 1995, http://www.dstc.bond.edu.au/~walther/groco/registration.runtime.html

Walther, M. (1995), *EMS in GroCo*, 1995, http://www.dstc.bond.edu.au/~walther/groco/grocoems.html

Watters, A. R. (1995), *UnixWorld Online: Tutorial Article No. 005, The What, Why, Who, and Where of Python*, http://www.wcmh.com/uworld/archives/95/tutorial/005.html

Watters, A., Rossum G.van., Ahlstrom, J. C. (1996), *Internet Programming with Python*, M&T Books of MIS:Press Inc., 1996, New York, N.Y.

WhitePine Software, http://www.wpine.com

WhoWhere? Inc., http://www.whowhere.com

Yellin, F. (1995), *Low Level Security in Java*, 1995, http://www.w3.org/pub/Conferences/WWW4/Papers/197/40.html

APPENDIX: ABBREVIATIONS AND TERMINOLOGY

Computer communication technology in general and Internet communication in particular use many abbreviations, some very well known but others not so wide-spread. For this reason, this rather long list of abbreviations and terminology used in the text is provided to facilitate the readability of this document.

ACL	Access Control List.
ActiveX	The ActiveX architecture from Microsoft.
Apache	Wide-spread WWW server, an extension of NCSA httpd.
API	Application Programming Interface.
Applet	Java applet embedded in HTML document.
AWT	Abstract Window Toolkit.
Browser	WWW Client Application, e.g. Netscape Navigator or MSIE.
BSCW	Basic Support for Cooperative Work.
codec	Compression/Decompression algorithm, e.g. for audio and video communication.
CGI	Common Gateway Interface.
CMC	Computer Mediated Communication.
Collabra	Groupware application by Netscape Communications Inc.
Communicator	The Netscape Communicator application suite including Navigator.
CoopWWW	The European research project CoopWWW.
CORBA	Common Object Request Broker Architecture.
CSCW	Computer Supported Cooperative Work.
CU-SeeMe	Video-conferencing software from Cornell University and WhitePine Software.
Demon	Server process running in the background.
DHCP	Dynamic Host Configuration Protocol.
fps	Frames per second, a measure of video transmission rate.

Frame rate	Frequency of picture update, i.e. frames per second.
FTP	File Transfer Protocol.
GET	The HTTP GET method.
GMD	German National Research Centre for Computer Science.
Helper	Client-side helper application that is launched by WWW client upon receiving data of a particular MIME type.
HTML	HyperText Markup Language.
HTTP	HyperText Transfer Protocol.
HTTPd	HTTP demon.
IETF	Internet Engineering Task Force.
IP	Internet Protocol.
IRC	Internet Relay Chat protocol.
ITU	International Telecommunication Union, formerly known as CCITT.
Java	The Java programming language developed by Sun.
JavaScript	The JavaScript scripting language by Netscape and Sun.
JDK	Java Developer’s Kit.
Lotus Domino	Web-based version of Lotus Notes (see below).
Lotus Notes	Wide-spread Groupware application by Lotus/IBM.
MIME	Multipurpose Internet Mail Extensions.
MSIE	Microsoft Internet Explorer WWW client.
Navigator	Netscape Navigator WWW client.
NCSA	National Center for Super Computing Applications.
Netscape	Netscape Communications Corporation.
PDI	Personal Data Interchange.
Plug-in	Embedded program running inside HTML document in a WWW client application.

POST	The HTTP POST method.
POTS	Plain Ordinary Telephone Service, Plain Old Telephone System.
PUT	The HTTP PUT method.
Python	The Python programming language.
RFC	Request for Comments, working documents of IETF.
RTP	Real-time Transfer Protocol.
RSVP	Resource Reservation Protocol.
SISU	Swedish Institute for Systems Development.
SSL	Secure Sockets Layer.
Subtype	MIME Subtype.
Sun	Sun Microsystems Incorporated.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
ULS	User Location Server, User Location Service.
URL	Uniform Resource Locator.
VB	Visual Basic.
VM	Virtual Machine.
VRML	Virtual Reality Modeling Language.
W3, Web, WWW	World Wide Web.