An Analysis of Seamlessness

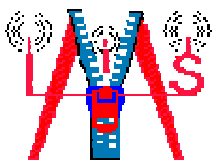
Norsk Regnesentral
ANVENDT DATAFORSKNING
Norwegian Computing Center/Applied Research and Development

RAPPORT / REPORT

Report nr. 931

Peter D. Holmes
Frode Løbersli
Wolfgang Leister
Lars Aarhus

Oslo
September 1998



IMiS Kernel

Norsk Regnesentral
ANVENDT DATAFORSKNING

Rapport/Report

Tittel/Title:
An Analysis of Seamlessness

Dato/Date: September
År/Year: 1998
ISBN: 82-539-0414-2
Publikasjonsnr/
Publication no: 931

Forfatter/Author:
Peter D. Holmes, Frode Løbersli, Wolfgang Leister, Lars Aarhus

Sammendrag/Abstract:

The IMiS-Kernel project is defined as a three-year project. It is comprised of six Work Packages (WP1-6). This purpose of this report is to describe the first-year work which has been carried out within IMiS-Kernel's Work Package 1 (WP1).

The overall goal within WP1 is to understand and exploit the demands that next generation multimedia applications pose upon the underlying infrastructure and how these demands influence on the design of the applications. As its primary goal in the *first* year, WP1 focused its work upon the issue of seamlessness, in order to establish a concrete foundation for the informed selection and characterization of a communication service/application suited to the needs of the project.

The focus of the report therefore deals with the results of an analysis conducted in order to help attain a more concrete understanding of the nature of and requirements for achieving seamless service/application performance within a variety of work contexts. The report closes with the motivation for WP1's recommendation of a specific application framework, for use as the foundation for application development and experimentation within the IMiS-Kernel project.

Emneord/Keywords: seamlessness, communication services and applications, work contexts, email, news, WWW, conferencing, video streaming, DBMS

Målgruppe/Target group: software industry, research institutions, NFR

Tilgjengelighet/Availability: Open

Prosjektdata/Project data: Project participants are Ericsson, NR, UNINETT and IFI UiO.
Funded 50% by the Research Council of Norway (NFR).
Prosjektnr/Project no: 28006

Antall sider/No of pages: 66

Norsk Regnesentral/Norwegian Computing Center
Gaustadalléen 23, Postboks 114 Blindern, N-0314 Oslo, Norway
Telefon (+47) 22 85 25 00, telefax (+47) 22 69 76 60

An Analysis of Seamlessness

Peter D. Holmes
Frode Løbersli
Wolfgang Leister
Lars Aarhus

Norsk Regnesentral
September 1998

Table of Contents

Chapter 1

Introduction

1.1 Background

The acronym IMiS stands for *Infrastructure for Multimedia in Seamless Networks*. IMiS Kernel is a long-term research project where the goal is to understand what demands next generation multimedia applications pose upon the underlying infrastructure and how that infrastructure should meet these requirements. On a national basis, work within the project will help develop high-level competence within strategic areas of information technology, as well as enrich cooperative efforts between research organizations and industry. The current actors within IMiS Kernel are Ericsson, NR, UNINETT and IFI UiO. Further details about the project can be found on its WWW pages [1].

The IMiS-Kernel project is defined as a three-year project. It is comprised of six Work Packages (WP1-6). This purpose of this report is to describe the first-year work which has been carried out within IMiS-Kernel's Work Package 1 (**WP1**).

1.2 WP1 Goals and Activities

The overall goal within WP1 is to understand and exploit the demands that next generation multimedia applications pose upon the underlying infrastructure and how these demands influence on the design of the applications. Within the project's three year span, the broad activity plan defined for WP1 includes activities aimed to:

1. analyze aspects of seamlessness in communication services and applications;
2. investigate application-level requirements for achieving seamlessness within a select (set of) communication-based work-context(s);
3. determine the characteristics of a service/application which satisfies these requirements;
4. design, develop and prototype the technical foundation of this service/application;
5. create and, when possible, evaluate prototype applications built upon this foundation, within well-defined pilot environment(s); and,
6. create both conceptual and technical results (i.e., architectures, component designs, etc.) which can be published within the academic literature, as well as promoted for further development, productification and spin-offs within the industry.

As its primary goal in the *first* year, WP1 was to focus its work upon the first three activity areas described above; that is, to *select* and *characterize* a communication service/application suited to work context demands from IMiS Veritas. The purpose in choosing Det Norske Veritas (DnV) as a case organization was grounded in the fact that the ongoing IMiS Veritas project [2] had already established concrete results concerning functional

requirements for various mobile work contexts — contexts in which the need for seamless applications and network services is very great.

In addition, the service/application to be selected and characterized was required to have the capacity of (ultimately) being run in the IMiS Kernel experimental network, in order to demonstrate the advantages gained through use of the advanced, infrastructural features being developed and implemented within **WP2**[1].

Given these criteria, WP1's application selection and characterization work can be perceived as having been carried out within the context illustrated in figure 1.

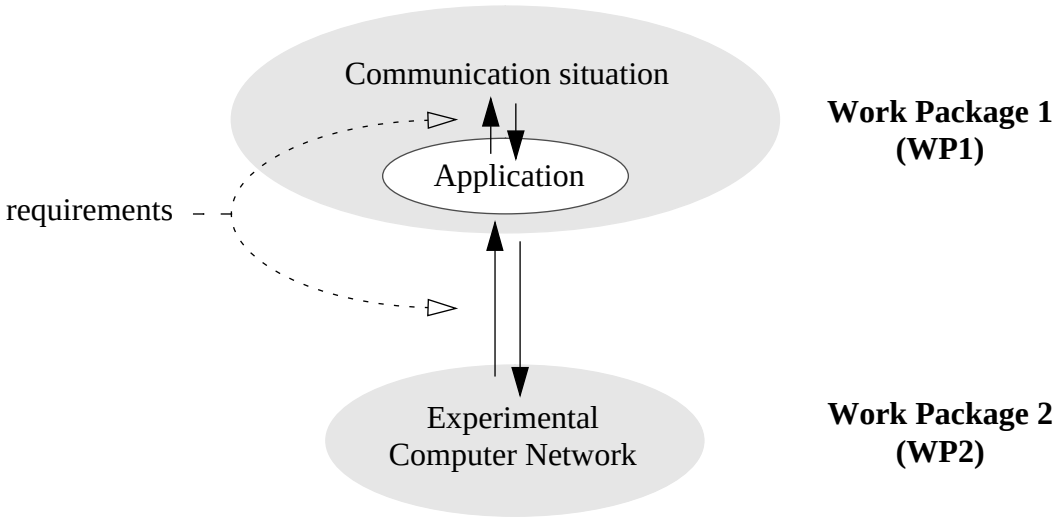


Figure 1 : *Context for application selection and characterization*

1.3 Seamlessness

Possibilities and requirements within the area of personal and business communication are evolving at an unprecedented pace. *Mobile workers and mobile devices* are central forces which currently propel the industry. Some trends which characterize this rapid evolution are:

- an increase in the volume of mobile workers;
- an increase in both the volume and diversity of mobile devices; and,
- a "breakdown" in the traditional association of a person with a specific number, device and/or network.

Some may be contend that the need for seamlessness arises from the requirements inherent to supporting these trends. That is, there is a clear need to empower mobile workers with applications which provide "seamless" access to the data they require, regardless of the

1) Most of the work aimed to develop the experimental network is being carried out in IMiS-Kernel's Work Package 2

kind of device they happen to have available. This new generation of applications must be explicitly designed for and/or adaptively tuned to the capabilities of the device and available infrastructure, in order that work can be performed effectively.

Within the context of computer networking, there is still no international agreement upon the exact meaning of the term ‘seamless’. The usage of the term is therefore often very loose, and highly subject to the context in which it is applied; it has several definitions, depending upon which perspective that is used. In this report, we will emphasize the concept of seamlessness as defined from the user’s point of view. Thus, we will employ the definition put forward within the IMiS pre-project [3]:

> *The ultimate goal for the user is that the “system” will recognize the user wherever she logs on, on any network, with any equipment, at any time, with the applications in a given state and have them adapt in the best possible way given these surrounding conditions.*

1.4 Focus and Outline of the Report

This report describes the first-year work carried out within WP1. The focus of the report deals with the results of an analysis conducted in order to help ground the term ‘seamlessness’ — to attain a more concrete understanding of the nature of seamlessness in a variety of contexts.

In Chapter 2, WP1’s initial work process is described. Thereafter, certain contextual dimensions of seamlessness are identified and described. Chapter 4 continues with an examination of a range of services and applications, and analyzes them with respect to the contextual dimensions which have been described. Based upon the analyses performed, the report concludes with the motivation for DP1’s recommendation of a specific application framework — a framework to be used as the foundation for application development and experimentation for continued work within the IMiS-Kernel project.

Chapter 2

Initial Work Process

The purpose of this chapter is to briefly describe the initial, exploratory work carried out by the DP1.0 project team. It intends to help clarify the process which has led to the results described in this report.

From the project's outset, the DP1.0 activity within the IMiS-Kernel project was subject to initial project specifications and goals. As its primary goal in the first year, DP1.0 was to identify the characteristics of an application for investigation, (possible) development and further use within the IMiS-Kernel project's second year work. If possible within the first project year, DP1.0 was to even go so far as to specify the application.

2.1 Characteristics Required of the Application

As part of the initial project specifications for DP1.0, certain characteristics were required of the application. These included that the application should:

a) focus upon establishing a communication service which meets demands within the IMiS-Veritas project [2];
b) support cooperative work for some given work situation; and,
c) enable testing of mechanisms for achieving seamlessness, as provided by DP2.0.

From DP2.0, some of the characteristics required of the application were that it should:

d) enable experimentation with QoS; in particular, experimentation with bandwidth reservation and bandwidth management (e.g., IPv6, RSVP, etc.);
e) enable experimentation with technology for achieving mobility (e.g., MobileIP);
f) allow for experimentation with heterogeneous networks, systems and terminals;
g) handle and operate with multiple media types; and,
h) transmit audio and/or video streams, in order to generate a high volume of network traffic.

In characterizing the application, it was judged advantageous to consider the features of existing, in-house prototypes on equal footing with those of applications emerging on the market. When selection of the application was ultimately to be made, one of the primary considerations was a need to have access to the application's source code. Without the source code, it would be difficult to meet some of the requirements from DP2.0.

It was also recognized that both access to and *knowledge of* the source code could provide an even better early position for the project. For this reason, in-house prototypes in which there existed ongoing or planned activity were to enjoy special favor during the consideration process.

Given these initial conditions, DP1.0 started out its work with efforts to further articulate the criteria, and to seek priorities amongst the requirements listed above. This led to an initial focus upon points (a) and (b) above; that is, to begin considering what kind of system could support cooperative work within one of the mobile work settings identified during the IMiS-Veritas effort.

2.2 Assistance-on-Demand Study

The IMiS-Veritas project had spent time studying and describing a set of work situations and contexts encountered by certain members of DnV's mobile workforce. Some of the problems and requirements of the mobile workers include:

- the need to contact co-workers "there-and-then", for information exchange or problem solving;
- the need to schedule meetings with co-workers and customers from the field;
- the need to produce and transmit documents;
- the need to retrieve information from DnV's shared files and intranet (perhaps also from personal/private files), and
  - collate it with data from the field, then
  - re-distribute the coordinated information to geographically-distributed co-workers and/or synchronize it with DnV's central resources.

Within DP1.0, discussion of these studies led to the suggestion of pursuing an examination into the characteristics of a possible Assistance-on-Demand (AoD) system for DnV. The idea was to consider the characteristics of a system which could provide a single application interface to the fundamental set of applications and services used by a large part of DnV's Inspection and Consulting personnel. The fundamental set of applications and services used by the mobile workforce studied within IMiS-Veritas include:

- **Companion**: a shared electronic database / archiving system, including:
  - information about customers
  - incoming / outgoing correspondence with customers (including copies of email)
  - courses and course information (e.g., course participants and their status);
- **Microsoft Outlook**: a system for coordinating email, contact information, (shared) agendas, etc.;
- **Nauticus**: DnV's system for storage and access to all documentation concerning a ship's life-cycle, including text files, Microsoft Word documents, images, photos, CAD drawings, digitalized video, references to analog video, etc.
- (remote) access to **personal / private files**
- (remote) access to **DnV's intranet and shared files**.

Some time was spent by the DP1.0 group looking into the different kinds of architectural features which would be required by this AoD system, and considering how these features addressed some of the characteristics required of the IMiS-Kernel application. Soon, the focus was narrowed down to examining whether and how NR's LAVA technology [15] for video streaming could be used within an AoD architecture.

After some discussions and deliberations along these lines, efforts in this direction were terminated. The decision to terminate this angle of approach was based upon the following reasons:

- it was judged that video streaming for mobile workers at DnV was not a primary concern for DnV;
- the AoD architecture was still extremely young in its conception and formation, with most parts only vaguely defined;
- at that time, the future level-of-activity for the LAVA technology was uncertain; and,
- it was judged that insufficient progress was being made toward the goal of application characterization.

2.3 Examining Seamlessness

Having terminated one line of pursuit, it was decided to direct efforts toward one of the other requirements for application characterization. The requirement selected for focus was point (c), from section 2.1; that is: “the application should enable testing of mechanisms for achieving seamlessness”. The decision was therefore to conduct a study which would help ground the term ‘seamlessness’ — to attain a more concrete understanding of the nature of seamlessness in a variety of contexts.

The approach devised was to examine one region within the “space” of seamlessness. The region was divided into a set of **contextual dimensions**:

- bandwidth
- connectivity
- centralized vs. de-centralized (architecture / data)
- terminal type
- homogeneous vs. heterogeneous environment, and
- real-time demands.

These contextual dimensions were adopted from work carried out in the IMiS-Veritas project[1]. There, observations and studies were made of the different kinds of situated work carried out by certain members of DnV’s mobile workforce. It was determined that the contextual dimensions listed above reflect some of the fundamental, yet *variable* aspects of the mobile workers’ electronic environments.

With these contextual dimensions in hand, the examination of seamlessness proceeded using a selected, yet basic set of applications and services. Each application / service was independently examined with respect to each of the dimensions. The results of this study are presented and discussed in chapter 4.

1) Further clarifications of each of these dimensions are found in chapter 3.

Chapter 3

Contextual Dimensions of Seamlessness

In the previous chapter, certain dimensions of seamlessness were mentioned. These were:

- bandwidth
- connectivity
- centralized vs. de-centralized (data)
- terminal type
- homogeneous vs. heterogeneous environment, and
- real-time demands.

Here, these different dimensions are called **contextual dimensions**, since they reflect some of the fundamental, yet variable contextual aspects of mobile workers' electronic environments.

The purpose of this chapter is to provide further clarifications for each of these contextual dimensions, prior to the presentation of the studies of seamlessness found in chapter 4.

3.1 Bandwidth

This dimension of seamlessness refers quite simply to the amount of bandwidth available during use of the service/application. In the study, variation along this dimension has been loosely characterized as being either "low bandwidth", "medium bandwidth" or "high bandwidth". The interpretations of these difference classes can be *(very)* roughly conceived of as:

- "low bandwidth": bandwidth commonly available over a GSM or modem-type connection (ca. 9600 - 56000 bits/sec)
- "medium bandwidth": the range of bandwidth available over a 2x64 kbps (kilobits per second) ISDN connection, up to that commonly available on a moderately loaded 10BaseT Ethernet connection (ca. 128 kbps - 1++ megabits per second)
- "high bandwidth": the range of bandwidth available over a moderately loaded 100BaseT Ethernet connection, up to that commonly available on a moderately loaded 622 Mbit/s ATM connection (ca. 10 - 200++ Mbit/s).

3.2 Connectivity

This dimension of seamlessness ranges from "No connection / (connected now and then)" to "Constantly connected". The former of these conditions represents situations in which

the user may be employing portable equipment which, for example, he occasionally connects to the internet. The use of the connection at that time may be to retrieve, transmit and/or synchronize information. The latter condition, "constantly connected", refers to a situation in which the user's equipment is connected such that no hard application failures occur (i.e., crashes) due to loss of connection[1].

Between these extremes are conditions in which the user experiences "poor or partial connection" with the systems supporting the service/application. For example, poor or partial connections can sometimes be experienced when connections are established through GSM, infrared and other kinds of wireless links[2].

3.3 Centralized vs. Decentralized (Architecture / Data)

For each service/application examined in chapter 4, this dimension of seamlessness intends to focus upon either the centralized vs. decentralized nature of that service/application's architecture, the nature of its data storage model, or both. Exactly which of these is addressed per service/application is made clear in the respective discussions.

The categories established for this dimension are:

- centralized
- partial / proxy / hybrid, and
- de-centralized.

In some cases, these headings may be modified to more closely fit the nature of the service/application.

3.4 Terminal Type

Here, we have chosen to focus this dimension of seamlessness upon *standard types of terminals*, rather than specific *terminal profiles*. To look into and examine seamlessness with respect to specific terminal profiles would result in a degree of combinatorics and complexity that is probably unmanageable, and certainly far from serving the principal aim of this study. Still, we must be explicit about some of the characteristics of what we call "standard terminal types".

By "standard types of terminals", we mean the basic kinds of computing terminals upon which today's services and applications can be employed. The terminal types considered in this study are:

1) Note: equipment may be considered "constantly connected" in cases where a wireless link is providing an extremely stable connection.
2) For example, very-high frequency radio (ca. 2.4 GHz) as employed by products such as BreezeNET, WaveLAN, etc.

- Palmtops / PDAs
- Laptops
- Desktops (e.g., PC, UNIX workstations), and
- Borrowed desktops.

When acquired, these terminals are usually configured with some "standard" equipment and devices. Nowadays, the laptops and desktops are often configured with CD drives, microphones, speakers and soundcards. Some of the desktops are even configured with video cameras and video processors, as well.

These terminals can also be configured with other, more specialized equipment and devices (e.g., scanners). Here, we assume that such specialized devices are not part of a standard configuration.

When it comes to communication links, what we mean by "standard terminal types" must also be clarified. It must be made explicit that although palmtops and laptops are portable terminals, it is here assumed that these terminal types have available — either as internal or external devices — the necessary equipment to become "connected now and then".

For the palmtops, it is assumed that a standard connection is made using a modem and a telephone, thereby achieving a "low bandwidth" connection. It is of course technically possible for palmtops to achieve medium- and high-bandwidth connectivity; alternatives include use of an Ethernet card and a (high-speed) Ethernet connection, as well as through a very-high frequency radio link (e.g., BreezeNET, WaveLAN). In the context of this study, however, these latter kinds of connections are *not* assumed to be the standard kind of way in which palmtops achieve connectivity — it is simply not the way most users will be connecting their equipment today.

Like the palmtops, it is assumed that the laptops can be connected via modem and phone. In addition, it is also assumed that the laptops can also be connected via Ethernet or a very-high frequency radio link. Thus, for the laptops the bandwidth available to the machine can range anywhere from low to high.

For the desktops, it is assumed that a standard connection is made using an Ethernet, ISDN and/or ATM link, thereby constituting a "constantly connected" computing terminal offering relatively steady medium- to high-bandwidth connectivity.

In this study, we also include a terminal type called "borrowed desktop". This type is included since it factually represents a computing and communication environment often faced by members of DnV's mobile workforce. When it comes to addressing seamlessness, the interesting situation about using another person's machine is that one's rights as a guest user — rights which are specified and controlled in a variety of different parts of the total computing environment — can have an overwhelming impact upon whether or not one is able to usefully employ a specific service or application.

3.5 Homogeneous vs. Heterogeneous Platform / Environment

In general, this dimension of seamlessness ultimately intends to focus upon the degree to which the different parts of a given service/application — as well as the underlying systems and infrastructure which support it — are *compatible* with one another. Here, the combined term 'platforms/environments' refers to the entire set of systems, subsystems, protocols, transmission media, etc. which *support* the application layer — essentially, the platform/environment is everything *but* the application layer.

As an aspect of seamlessness, compatibility must be considered both in terms of performance and functionality[3]. Here, a "*perfectly compatible*" set of (interoperating) systems and subsystems is meant to denote a set of systems and subsystems in which:

- the set of functions provided by the various (sub)systems provides complete coverage of the functional requirements across all elements of all (sub)systems, and
- the performance of the various (sub)systems is within the lower bounds required across all elements of all (sub)systems.

A set of perfectly compatible systems and subsystems should be observed as interoperating flawlessly, with respect to specifications and expectations.

At this point, it is important to remember that compatibility and compliance are two different conditions. A (sub)system can *comply* to a given set of standards by ensuring that:

- the mandatory aspects of the standards are implemented and satisfied, and that
- the implementation does not violate any mandatory aspect of any of the standards.

As long as it satisfies these two conditions, an implementation is free to include whatever other functionality it wishes, while still being classified as being compliant to the given set of standards. For many aspects of (sub)systems, this freedom includes the performance of its implementation. In other words, **the overall performance of a standards-compliant implementation can be very good or extremely poor, as long as any performance characteristics which are explicitly mandatory in the standards[4] are satisfied.**

Theoretically, perfect compatibility can be achieved within electronic contexts in which the interoperating platforms/environments have either homogeneous or heterogeneous implementations. Realistically, the platforms/environments we employ often consist of:

- heterogeneous components, which are
- more-or-less compliant to some *evolving* set of standards, and thereby
- compatible to varying degrees.

Returning to the dimension of seamlessness concerning homogeneous vs. heterogeneous platforms/environments: we find that though different (sub)system implementations may *comply* to some given set of standards, many — if not most — such (sub)systems are not perfectly compatible. One (sub)system implementation offers a certain range of functionality, while another offers a similar — though not wholly overlapping — range of functionality. In addition, significant differences in (sub)system performance can also exist.

3) In some cases, performance can be viewed as a functional parameter.
4) Timeout values are one kind of performance characteristic which may be subject to standardization.

Consequently, is it precisely these conditions which lead to differences in service/application behavior when employed across heterogeneous platforms/environments. This is the area of focus addressed by this dimension of seamlessness.

3.6 Real-time Demands

Most everyone would prefer to have a service or application perform and operate as fast as possible, all of the time. This is not the intended point of “real-time demands” in this study.

This dimension of seamlessness concerns the degree to which a specific service/application has an implicit requirement that it operate and perform in real-time. Person-to-person conferencing and video-streaming are examples of applications which share this nature.

Both of these applications are similar in that once a stream begins to be played/displayed for a recipient, that the recipient most likely prefers that the stream continue to be played/displayed without disturbing gaps and pauses. The two applications are different, however, in that for video-streaming, the user may be willing — perhaps even required — to tolerate the fact that the stream cannot be played/displayed as soon as desired. Such a situation is not tolerable in a person-to-person conference: it would simply break down the attempt to carry out natural dialogue.

Other services/applications, such as email and news, are clearly different when it comes to real-time demands. For these applications, there is less of a demand for quick, smooth delivery of the data — gaps and pauses during data delivery does not radically degrade the quality of the user’s experience.

Other issues related to real-time can be directly or indirectly relevant to the user as well. For instance, in services/applications such as video-streaming, WWW, email and news, there is always a greater or lesser temporal distinction between the time at which the content is created and the time at which the content is absorbed by a receiver.

For each service/application examined in chapter 4, the discussion under “real-time demands” will emphasize and elaborate more upon *real-time related, end-user needs and expectations for that service/application.*

Chapter 4

Services, Applications and Seamlessness

In this chapter, a range of services and applications are examined with respect to some different contextual dimensions of seamlessness, as described in section 2.3 and chapter 3. The services and applications examined here are:

- email
- news
- World Wide Web
- conferencing
- video-streaming, and
- database management systems.

For each service/application, a reference model for the underlying, supporting architecture is provided. Following each reference model, the results of each study and a short discussion are provided.

Within each aspect of this study, the emphasis is upon the end-user effects experienced when trying to use the service, or to employ the application. Where relevant, some mention is also given to how different architectural configurations may limit or enable the capacity to achieve more seamless behavior of the service/application.

4.1 E-mail

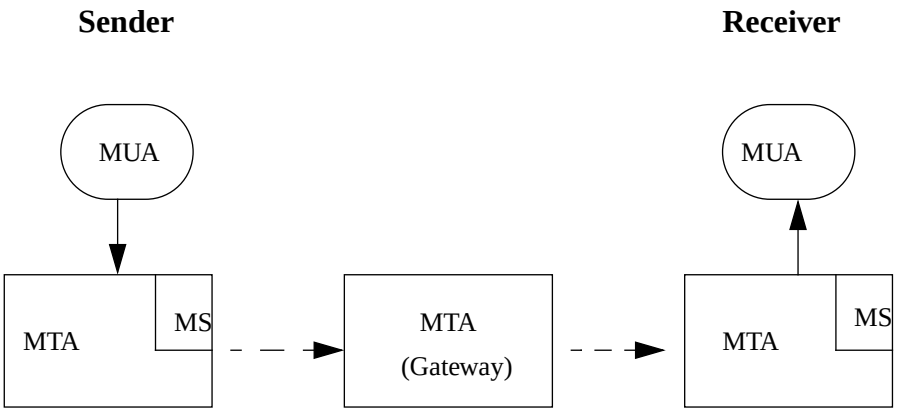
This section discusses seamlessness and electronic mail, or *e-mail*, which is perhaps the most traditional and popular data network service today.

4.1.1 Reference model

The generic reference model for e-mail is called MOTIS (Message-Oriented Text Interchange System), and is part of the OSI reference model (ISO 10021)[7].

The classic model consists of two entities, the Message Transfer Agent (MTA), and the Message User Agent (MUA). The MTA provides the intelligence needed to move messages between recipients. The MUA is responsible for formatting messages and transfer to the MTA.

Often, two additional components are added to the MTA, the Message Store (MS) and the (mail) Gateway. The MS is an intermediate data structure for holding messages, and traditionally resides on a (mail) server. The Gateway is responsible for conversion of messages between proprietary e-mail systems. An example of MOTIS is given in figure 2.



MUA = Message User Agent
MTA = Message Transfer Agent
MS = Message Store

Figure 2 : *E-mail reference model*

This study is primarily concerned with Internet mail, where:

- *sendmail* is usually the implementation of MTA,
- SMTP (Simple Mail Transfer Protocol) [11] is the communication protocol between all the entities — especially the MTAs — and
- POP (Post Office Protocol) [10] or IMAP (Internet Message Access Protocol) [5] is used when retrieving message information from the MS to the MUA.

Traditionally, Internet mail messages have consisted of text only, and in a format as specified in RFC 822 [6]. Nowadays, however, non-textual mail attachments are possible, and MIME (Multipurpose Internet Mail Extensions) [4], [9] is the standard for supporting this.

Other e-mail systems, such as X.400, are only occasionally considered and discussed here; that is, they are mentioned only when they offer genuinely different functionality. For the interested reader, Lovett and Skogseth [8] supplies a good introduction to X.400.

4.1.2 E-mail and aspects of seamlessness

The purpose of this section is to provide a short study of e-mail and seamlessness along the contextual dimensions given in chapter 3.

Table 1: *E-mail and* BANDWIDTH

Low Bandwidth	Medium	High Bandwidth
Generally few problems with text only messages; handling large MIME attachments increases complexity and, with it, the chances of encountering new problems. Lack of adaptivity in message-retrieving: · little support for downloading of message parts; note: retrieval of header plus first 10 lines is possible in IMAP. · no support for *not* downloading attachments; or rather, the functionality is not implemented yet, but technologically possible (IMAP).	No problems.	No problems.

Table 2: *E-mail and* CONNECTIVITY

No connection (i.e., connected"now and then")	Poor / partially connected	Constantly connected
Synchronization problems with leave-mail-on-server functionality can be experienced; requires special configuration in IMAP to maintain consistency.	Practical use can be disturbed.	No problems.

E-mail and CENTRALIZED VS. DE-CENTRALIZED

For each administrative domain, the e-mail reference model has a centralized intermediate mail storage (MS). However, when an MUA accesses the mail server, the messages are usually transferred and stored locally on the end-user's machine (though they can be left on server, as explained above under connectivity), which means that the data storage is in fact de-centralized.

Globally speaking, the e-mail architecture (like most other network service architectures) is highly de-centralized; logically, it can be seen to consist of all the different administrative domains. The main technical problem with e-mail is the transformation between different message formats within the Gateways. An example is converting from an external message format, such as RFC-822, to a proprietary internal solution e.g. X.400. However, such transformation is usually transparent to the user, unless the conversion is incomplete (e.g., due to incompatibility problems between the formats). This is further elaborated below under homogeneous vs. heterogeneous platforms.

Table 3: *E-mail and TERMINAL TYPE*

Except for "Borrowed desktop" below, all cases presume the existence of the required applications (MUA) with MIME support.

Palmtop	Laptop	Borrowed desktop	Desktop
Should be OK. Support for conversion of MIME attachments is required on the mail server, as this should not be handled by the client application. Memory size can limit downloading.	OK.	Cannot succeed at all unless required applications are (or can be put) in place, and remote access (e.g. telnet) to mail server and local host is provided. Also, applications (or "plug-ins") for MIME support required for handling attachments. Mail via web access is not supported by most mail servers, although technically possible (e.g. through accounts such as *hotmail*)	OK.

E-mail and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS

Regarding the homogeneous vs. heterogeneous platforms issue, this is lifted from the platform/environment level up to the application layer. Thus when discussing e-mail in this context below, the platform is assumed to be the same, but the applications may vary.

Generally, MUAs exist for almost every possible environment. Interoperability between users with different clients within the same administrative domain is usually uncomplicat-

ed. A mail server is usually able to support many different MUAs, as most clients communicate using SMTP and POP/IMAP.

However, the case of a single user alternating between different MUAs, or changing from an old client to a new one is made difficult by proprietary local message storage formats. The main problem concerns old messages tucked away in mail folders, as these are archived in a non-standardized manner.

As mentioned above, compatibility between different domains may sometimes not be achievable, even when the same hardware platforms are in use. This can occur when the two sets of functions supported by the internal mail architectures are vastly different. One example is the X.400 receiver-notification, which is sent to the sender once the message is retrieved by the receiver from the Message Store. Such functionality simply doesn't exist in Internet mail.

A final problem experienced by users is platform-dependent MIME support between the sender and receiver MUA. For instance, receiving a Microsoft Word document attachment on a UNIX mail client is cumbersome. More than anything else, however, this "problem" has to do with proprietary document formats, coupled with the fact that Microsoft Word is not supported on UNIX.

E-mail and REAL-TIME

E-mail has no true real-time demands, since the communication form is asynchronous by nature. However, the user has some *expectations* about maximum delay in message delivery, based on past experience with the service. The delivery time will vary depending on the global network load, but an upper limit is usually assumed. For instance, a message sent from Norway to USA is expected to reach the recipient within minutes, rather than hours.

4.1.3 Discussion

Electronic mail is probably the most widely employed and used data network service in the world today. The reasons are simple. The communication form is asynchronous, but still very fast. It consumes little bandwidth, apart from attachments. It has no real-time demands. It works on every platform and terminal type. It is usually found to be easy to use, even for the novice.

Often, the main problems with e-mail have to do with poorly-connected sessions, where there is a danger of losing mail because of inconsistency between the mail server and the local mail storage. Also, MIME attachments can be difficult to handle, especially on a borrowed desktop, since the necessary support must be available.

All in all, this study seems to indicate that e-mail is in fact already a very seamless service, and only marginal improvements can be accomplished. For this reason, electronic mail is not an immediately interesting candidate application for developing and testing new and experimental mechanisms for achieving (even greater) seamlessness.

4.2 News

This section discusses seamlessness and network *news*, which is another traditional and quite popular data network service.

4.2.1 Reference model

The generic reference model for news is a traditional client/server model, which consists of two entities, the News Server (NS) and the News Client (NC). The NS provides the storage of a collection of news groups and corresponding articles (which are either local or common with other news servers), and the intelligence needed to allow NCs to connect and access news groups according to server policy. The NC is responsible for downloading and presenting the articles to the user, for formatting and transferring new articles by the user to the NS, and maintaining local subscription information.

Often, different news servers are connected, and propagate articles and establishment of new news groups between themselves. The best example of such a system is USENET news. Such a global system has a distributed model, as seen in figure 3. In USENET news, NNTP (Network News Transfer Protocol) [13] is by far the most common communication protocol between all the entities. Traditionally, USENET articles have consisted of text only, and in a format as specified by RFC 1036 [12] and its successor [14]. Like e-mail, however, inclusion of non-textual news attachments (e.g., HTML) is possible, though not encouraged since there is no standard for supporting this.

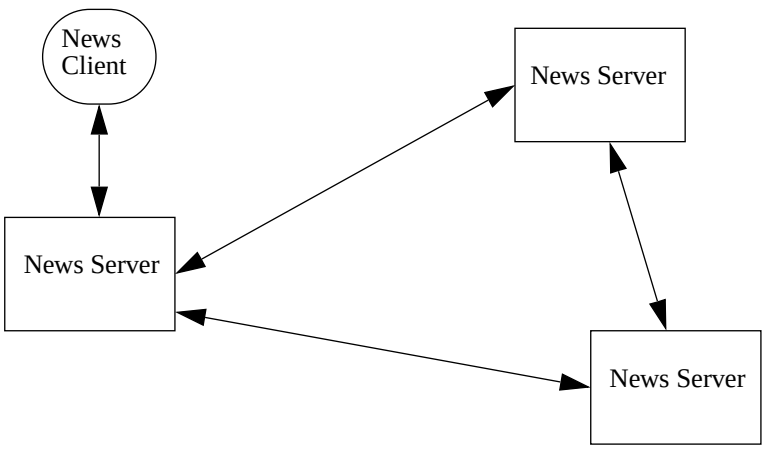


Figure 3 : *News reference model*

4.2.2 News and aspects of seamlessness

The purpose of this section is to provide a short study of news and seamlessness along the contextual dimensions given in chapter 3. Generally, news has many similarities to e-mail, but there are also some differences.

Table 4: *News and BANDWIDTH*

Low Bandwidth	Medium	High Bandwidth
Generally few problems (text only articles), except for binary groups, and when handling HTML attachments. Lack of adaptivity in article-retrieving from server: · only support for downloading of headers	No problems.	No problems.

Table 5: *News and CONNECTIVITY*

No connection (i.e., connected"now and then")	Poor / partially connected	Constantly connected
Requires intermediate local storage of news groups and articles.	Practical use can be disturbed.	No problems.

News and CENTRALIZED VS. DE-CENTRALIZED

As explained above, the global news reference model has a distributed and de-centralized data storage. There is no single, centralized storage of news articles. Instead a copy of the same article is kept on all News Servers connected to USENET news. With the increasing use of the service, this places a more and more heavy burden on the servers in terms of storage and propagation capacity.

Locally speaking, the only information stored in the News Clients is a (set of) resource file(s), (e.g., `.newsrc` on UNIX hosts), containing which news groups the user subscribes to, and which articles the user has read. There are usually separate resource files for each News Server accessed by the client, which might be confusing if the user chooses to subscribe to the same news group on different servers.

Table 6: *News and* TERMINAL TYPE

Except for “Borrowed desktop”, all cases presume the existence of the required applications (e.g., a News Client).

Palmtop	Laptop	Borrowed desktop	Desktop
Should be OK, though to our knowledge, no such client is available yet.	OK	Can succeed if required applications are (or can be put) in place. Proper user-specific behavior also requires that the user’s resource file(s) are also available. If local resource files need to be updated, remote access (e.g. telnet) to news server and local host is required. News via web access is not supported by most news servers, although proprietary solutions exist.	OK

News and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS

As with e-mail, the focus of the homogeneous vs. heterogeneous platforms issue is lifted from the platform/environment level up to the application layer.

Generally, News Clients exist for almost every possible environment. Interoperability between users with different clients is usually uncomplicated, since all servers and most clients communicate using NNTP.

Problems can arise in cases where a given user alternates between different News Clients, or changes from an old client to a new one; these problems usually arise from the proprietary resource file formats employed by different clients. A user is normally not able to switch directly from one client, e.g. Xrn, to another, e.g. Netscape News, since the local subscription information is maintained in different, non-standardized ways.

News and REAL-TIME

News has really no real-time demands, since the communication form is asynchronous by nature. Like e-mail, however, the user has some *expectations* about maximum delay in article propagation. The propagation time will vary depending on the global network load. Due to the distributed architecture of news, replies to an article from users close to a news server might also arrive earlier in time than the original article itself, when the original article is posted by a user farther away. Behavior of this can be somewhat confusing to unexperienced news users (and perhaps even irritating to the experienced ones). Generally speaking, the real-time expectations are lower for news than for e-mail.

4.2.3 Discussion

Network news is a widely used data network service, which has many similarities to e-mail. The communication form is asynchronous, but still reasonably fast. It consumes little bandwidth, though the increasing use of HTML attachments could affect that dimension. It has no real-time demands. It works on almost every platform and terminal type. News is fairly easy to use, even for the novice user.

Few problem areas have been revealed by the study. The most important ones are concerned with server storage and propagation capacity, and updating of local resource files during remotely connected sessions (especially on a borrowed desktop).

All in all, the study seems to indicate that news is in fact already a seamless service, with only little room for improvements. As with e-mail, the conclusion is that network news is not an immediately interesting candidate application for developing and testing new and experimental mechanisms for achieving (even greater) seamlessness.

4.3 World Wide Web (WWW)

4.3.1 Reference model

The WWW has in principle a simple architecture. It may be perceived as a client and a server program which communicate via HTTP (Hypertext transport protocol). The following figure shows the WWW architecture in its purest form.

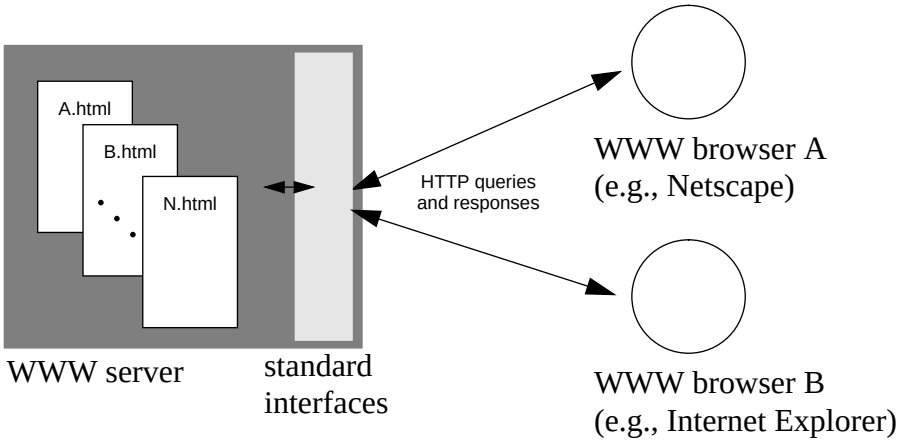


Figure 4 : *Original WWW reference model*

The figure shows two clients (WWW browsers) and a WWW server. In this figure, the server contains hypertext documents written in the HTML formal. Upon request from a client, the server transfers an HTML document to the client.

HTML documents can be stored as static elements; alternatively, they can be generated dynamically depending upon input contained in the request which arrives from the browser. Dynamic generation of HTML documents is often performed using Common Gateway Interface (CGI) scripts, which are invoked by the server. Other mechanisms for dynamic generation of HTML documents also exist (e.g., CORBA-based method invocations).

When an HTML document is returned to the client[1], the client presents the document for the user. HTML documents may include embedded information which is of a specific data type (e.g., `gif` image files, `wav` audio files, etc.). Browsers may employ any number of plug-ins and helper applications in order to help present the data to the user.

The great majority of HTML documents are hypertext documents: they include links to other documents. The linked documents may be stored upon the same server or upon any other server in the WWW.

Since the advent of JAVA, both small and large applications (“applets”) can be stored at the server and delivered to the browser. Some applets are simply for presentation and viewing, and offer the user no input channel. Larger applets can be developed, and delivered to the browser through precisely the same mechanisms. Larger applets can be highly complex, and can even offer the user full-scale application services.

1) The documents are transferred via the HTTP protocol. HTTP is a protocol implemented at the application level. It is built upon the TCP protocol.

Between the server and the WWW-browser there may exist one or several proxies and caches. The purpose of the caches and proxies are to reduce network traffic. The web-browser (or the proxy) compares the cached document to the network document and shows the most recent one. A proxy can be viewed as a cache used by several web-browsers.

4.3.2 WWW and aspects of seamlessness

Table 7: *WWW and* BANDWIDTH

Generally speaking, HTTP delivers best-effort performance. This is due to the fact that:

1. HTTP is implemented upon TCP
2. TCP is implemented upon IP
3. the overwhelming majority of the Internet — the network upon which most WWW traffic is routed —employs a network layer (e.g., IP) in which no performance guarantees are provided.

The consequence of this condition is that WWW content cannot be transferred any faster than the maximum speed of the weakest link (i.e., no faster than the *slowest* link along the routing path between server and browser). This means that for most of the media-types used in the WWW today, there are seldom any problems when medium-to-high bandwidth is available. An exception to this is the transfer of video data (see section 4.5 on video-streaming, for further details).

HTTP has mechanisms for checking whether the content on the server has changed compared to the contents in the cache or proxy. These mechanisms help reduce network traffic, since only changed or new documents are transferred. These functions help make the systems more seamless, since documents can be rapidly displayed at times.

In a low bandwidth situation, the users may experience long response times and downloading times. Some browsers available today do not have any automatic mechanism for prioritizing contents. This means that such browsers do not distinguish different content-types such as text, images, sounds, etc. Every media stream has the same priority, though they may have different information value.

Other browsers have functions which fully block image downloading or, alternatively, postpone image downloading until the rest of the document has been loaded. These functions help make the system more seamless for users. The IMiS-Kernel project group has not yet seen any browsers which automatically prioritize content-type based upon a function of the bandwidth and terminal type. We believe that such functions will help make WWW systems more seamless from a user's perspective.

Low Bandwidth	Medium	High Bandwidth
· Little support for adaptation: - Some support on the server - Some support on the client - Depends on the knowledge of the user · Proxy/cache solutions are often necessary. · Based on best-effort. Flow control done by TCP · Needs mechanism for prioritizing contents	· No problems except for video	· OK, no problems

Table 8: *WWW and* CONNECTIVITY

WWW technology was originally designed for situations with a stable network condition. However, some clients have tried to bypass this assumption. This is done by implementing mechanisms for off-line browsing. Off-line browsing is done by downloading predefined documents to a local proxy when connected. The downloading is done after predefined rules. An example of such rule is to download all new soccer results every morning at six o'clock to the local disk. The user may then view the results off-line. Still, this mechanism is not optimal. To get an optimal solution the system should have mechanisms for optimal cost, prioritizing of content-type and change notifications.

No connection (i.e., connected"now and then")	Poor / partially connected	Constantly connected
· Need mechanisms for off-line browsing. Few browsers have yet this possibility · Needs mechanisms for optimal cost reduction · Needs change notification mechanisms	· Needs proxy/cache · Depends on TCP connections	· WWW is design for this condition

Table 9: *WWW and* CENTRALIZED VS. DE-CENTRALIZED

WWW is in principle a centralized architecture, from a technical perspective. The documents are stored on a centralized server. These documents are accessed from the clients. In many situations we want not to have the documents in a centralized server. This due to for example organizational issues, network traffic reduction, security issues, etc. In these circumstances the data may be decentralized on different servers. This may be done with either hyperlinks or with mirroring.

Using hyperlinks to decentralize means that the information is stored on several servers. An example may be that each branch of an organization has their own web-server. Each different branch may then link related documents on the different servers. In this way the information on the servers can be perceived as a single information source. The problem with this solution is to update the hyperlinks when documents are changed. Another problem may be to implement a strategy for search engines by which to index decentralized web-sites. Most readers of this report know how difficult it may be to narrow a search to reduce the number of hits. Many companies implement their own search engines to index just their own server. When decentralizing, the search engine must index several servers. Again, this may lead to the problem of massive hits when searching.

Mirroring a web-site means that a copy of the documents is stored on several servers. This is done for instance to reduce the load on popular web-sites. Mirroring may also cause problems with hyperlinks and for search engines as described above. Some sites that are mirrored have built-in logic within the servers to send request to the server that fits best. The server that fits best may either be the one with lowest load or the one that is geographically nearest. From a user perspective, this functionality helps make the system seamless.

Centralized	Partial / proxy / hybrid	De-centralized
· Design for centralized data	· May be solved by hyperlinks. · It may be hard to update hyperlinks · Search engines may have problem indexing web-site	· May be solved by hyperlinks or site mirroring. · It may be even harder to update hyperlinks · Search engines may have problem indexing web-sites

Table 10: *WWW and TERMINAL TYPE*

When developing web-sites, most developers have the assumption that the users are using a desktop computer. This assumption creates certain problems for people using other terminal types. Most of these problems are rooted in the fact that the screen quality, battery capacity and security issues can limit usability.

In the PDA (Palmtop) case today, we often need specially-designed HTML-pages. These HTML pages are designed to be viewed on small monochrome devices. Another problem with PDAs is that many of the WWW-clients do not support all the functionality found in ordinary, desktop clients. For example, WWW-clients for PDAs currently lack Java-support. Another problem with PDAs is that they have highly restricted storage capacity. This implies that the browsers cannot use much space for caching.

Battery use is also a problem for PDAs and laptops. Especially in situations where the WWW pages have information targeted for different devices, such as sound devices. This may reduce the time the PDA or laptop may be used before it has to be recharged.

Another issue that may cause problems is use of plug-ins. This is especially true for PDAs and for borrowed PCs. It may be a problem for PDAs since support for plug-ins is still missing. And when support for plug-ins is implemented, it will take some years before the different plug-ins are implemented. In the case of a borrowed PC, plug-ins may be a security issue. If a plug-in is missing at a borrowed PC, it must be installed in order to view the associated content-type: otherwise, that content-type cannot be presented. The problem is that both the user and the plug-in must be trusted before the plug-in can be installed. In many situations this will not be allowed because of the IT- security strategy.

Palmtop	Laptop	Borrowed desktop	Desktop
· No Java support yet · No plug-in support yet · Needs specially-designed HTML pages · Often based upon specially designed clients which do not support all functionality · Small disk, i.e., small cache · Power limitations · Ergonomics may limit usability	· Screen quality may be a problem · Processor capacity usually weaker than desktops. · Power limitations · Ergonomic may limit usability	· Plug-ins may be a problem	· OK

Table 11: *WWW and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS*

In the case of WWW, there are few problems regarding homogenous vs. heterogeneous platforms. Client and server software exists for most relevant platforms. A growing problem, however, is that plug-ins are often implemented for only the most popular platforms. This results in situations in which certain pages' WWW content cannot be viewed on a less-popular platform. Another emerging problem is that of different functionality in the browsers. Here, the problem with homogenous vs. heterogeneous platforms has been lifted from the operating system level up to the application level.

Homogeneous	Mostly homogeneous	Heterogeneous
· OK	·	· There exist WWW-clients and servers for most of platforms. · Plug-ins may be a problem

WWW and REAL-TIME

The most pressing real-time demand for a WWW service is that of minimizing the delay between a browser request initiated by the user and presentation of content. When long delays are experienced, some users cancel requests rather than waiting. When a WWW server takes too long to acknowledge a request, an automatic timeout can occur which also leads to request cancellation.

4.4 Conferencing

Many associations can be made in regard to the term *conferencing*. Some of the dimensions of conferencing include:

1. the types of media being shared (e.g., audio, video, data, etc.),
2. the architectural design of the applications which enable the conference,
3. the properties of the infrastructure supporting communication,
4. the types of terminals amongst which the media is being shared,
5. whether the streams involved are generated from live or stored sources,
6. whether or not the streams involved are being directed to live sinks, and more.

4.4.1 Reference models for sharing data applications

Quite simply, there is no single reference model for conferencing. To help articulate the problem space, the focus here will begin with points 1 and 2 above. Specifically, we will first focus upon conferences in which two or more users ("**conference parties**") experience that they are sharing a consistent view of *one* given application; this situation is commonly called **application sharing**. The keyword here is 'consistent': at any given time, the semantic content underlying each party's view of the application is consistent.

To further limit the discussion at the outset, the application being shared is a **non-streaming data application**. Common conferencing examples here include shared document editing, shared spreadsheet (e.g., shared budgeting) work, etc.

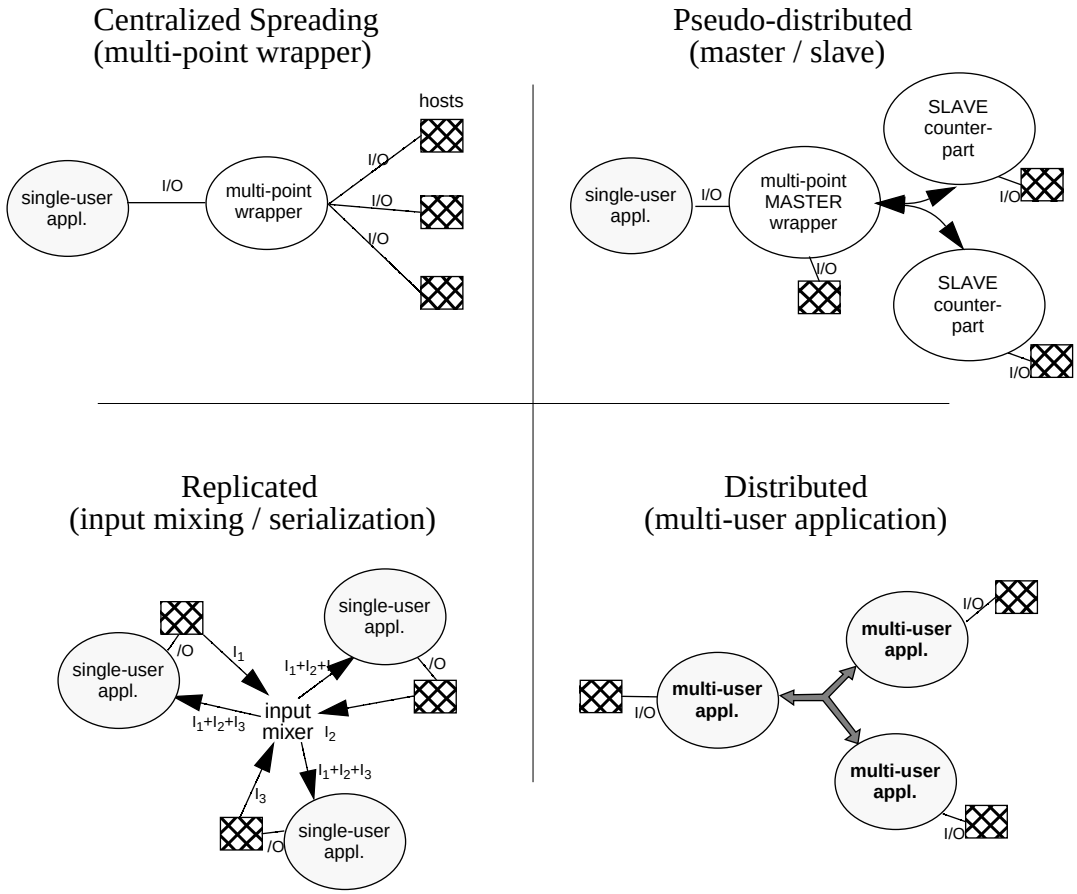


Figure 5 : *Traditional architectural approaches for sharing data applications*

Figure 5 depicts some traditional architectural approaches for sharing (non-streaming) data applications. Here, these different approaches have been given the names:

- centralized spreading (multi-point wrapper)
- pseudo-distributed (master / slave)
- replicated (input mixing / serialization)
- distributed (multi-user application)

The goal of all of these approaches is the same: to create the experience that the parties in the conference are sharing a consistent view of the same application. Some of the mechanisms employed in order to maintain consistency include data transfer, control protocols (e.g., for application updating and process synchronization), internal application state information and reasoning about shared states. Different constellations of these kinds of mechanisms are used for application sharing, depending upon the architectural approach. These different approaches are discussed further below.

4.4.1.1 Centralized spreading

This approach is often used when sharing *one* instance of single-user application process. Here, a **single-user application** is one which is designed to receive only one input stream, and deliver only one output stream. In the centralized spreading approach, a **multi-point wrapper** is used between the single-user application and the host(s). The wrapper serves two primary purposes for the application:

- it serializes multiple channels of incoming input control signals into a single input stream, and
- it spreads the application output to each of the participating hosts[2].

In the centralized spreading approach, both the application state information as well as the data being operated upon within the application are usually local to the application itself; the participating hosts do not have individual copies of the data nor state. This approach therefore usually requires copious output from the wrapper, in order that the participating hosts can present a consistent view of the application. In contrast, however, there is no need for advanced control protocols between the application and the participating hosts.

4.4.1.2 Pseudo-distributed

This approach much likens the centralized spreading approach, and is therefore also centralized to a very great extent. One difference usually found, however, is the difference in the complexity of the multi-point wrapping mechanism.

In the centralized spreading approach, one multi-point wrapper process is required for *each* single-user application process to be shared. In the pseudo-distributed approach, only one **multi-point MASTER wrapper** is needed in order to share a *set* of single-user application processes amongst participating hosts. The **SLAVE counterparts** designs employed in this approach are usually one of two kinds:

- designs in which only one SLAVE counterpart process instance is required (per participating host) *for the entire set* of applications being shared, or,
- designs in which one SLAVE counterpart process instance is required (per participating host) *for each* application being shared.

Since it a highly centralized approach, the characteristics of the pseudo-distributed approach are much like those of the centralized spreading approach. The multi-point MASTER serves two primary purposes:

- it serializes input control signals directed to the application, and
- it spreads application output to each of the participating hosts[3].

In this approach, both the application state information as well as the data being operated upon within the application are usually local to the application(s) themselves; the participating hosts do not have individual copies of the data nor state. Like the approach above, this approach usually requires copious output from the multi-point MASTER wrapper, in order that the participating hosts can present a consistent view of the application. In con-

2) When spreading application output, some wrappers employ IP multicast.
3) Again, some wrappers employ IP multicast when spreading application output.

trast to the centralized approach, however, there exist somewhat advanced control protocols between the multi-point MASTER wrapper and the SLAVE counterpart(s).

4.4.1.3 Application sharing via replication

This approach has a variety of related forms. The basic characteristics, however, are that:

- there exists an instance of the shared, single-user application process upon *each* participating host
- there exist individual, "replicated" copies of the application state information, as well as the data being operated upon, within each such application process
- (consistent) application sharing is achieved through centralized serialization and spreading of the input control signals generated by each application process instance.

Since there exist replicated copies of application state information and data, this approach requires only the serialized sharing of control information. However, the approach often assumes that the data to be shared has been copied and distributed in advance to the conference parties.

It should also be observed that independent, post-conference use of replicated data can possibly lead to inconsistent data conditions at a later time.

4.4.1.4 Distributed approach

This approach also has a variety of related forms. Compared to the three forms mentioned above, however, the basic difference is that the application being shared is *designed* as a **multi-user application**: it is designed to simultaneously receive input from more that one participating host, as well as to distribute its output to more that one participating host.

Many distributed application designs involve the use of individual instances of *identical*, inter-communicating process types upon *each* of the participating hosts. Other distributed application designs are hybrid in nature: involving the execution of individual instances of identical process types upon *some* of the participating hosts, while the remaining hosts are included via processes which liken some form of MASTER / SLAVE configuration. For the sake of simplicity here, we shall refer to all such processes as **peers**, whether they are truly identical or not.

In distributed approaches for achieving application sharing, there is no defacto design as to where the application state and application data reside. Application state and data may be localized to a single peer, or it may be distributed across all peers.

When application state and data are distributed, both of these kinds of information may require transmission amongst the peers. In order to preserve a consistent view of the application, distributed designs often require quite advanced control protocols for application updating and process synchronization. In order to synchronize state correctly, one or more or the peer processes may need to reason about its own state, perhaps even its peers' states. The mechanisms for achieving application synchronization are based upon a combination of the application's control protocol complexity, along with application reasoning about shared internal state.

Despite certain complexities in the distributed approach, it is by far the most flexible when properly implemented. Implementations which minimize and streamline transmission and "hand-off" of application state and data can be exceptionally robust in the face of unstable/ unpredictable network conditions.

4.4.2 Reference models for sharing audio and video

Throughout section 4.4.1, the focus is upon the approaches for sharing non-streaming data applications. In this section, we will examine some traditional architectural approaches for engaging in real-time **audio / video conferences**[4], thereby addressing the points 1, 2, 5 and 6 mentioned in section 4.4. Explicitly, the discussion here will concern person-to-person conferences — conferences in which both audio / video streams are generated by live sources.

With regard to the discussion of architectural approaches presented in *this* subsection, the most significant characteristic is that there are live sources[5] on each end of the conference; less emphasis will be placed upon the fact that there are live sinks in the conference. Discussion about the aspects of seamlessness for *receivers* of live streams is offered in section 4.4.4. Discussion concerning stream generation from *stored* sources is provided in section 4.5 *("Video streaming")*.

Here, it is assumed that an audio stream is generated by an audio application. A interface for user-control of the application is provided. Appropriate audio devices (e.g., microphone and speakers/headset) are controlled by the application. These same principles apply for the video stream and its associated application; the video devices instead include a camera and a display screen. For simplicity's sake, it is assumed that both the audio and video applications are run upon the same host.

4.4.2.1 Basic functional requirements

In order to achieve person-to-person audio / video conferences, some basic functionality must be in place; other kinds of functionality may be required, depending upon the total configuration (i.e., application *and* infrastructural configuration taken together). These areas of basic functionality include:

- control protocol transcoding (may be required)
- agreement upon stream format (i.e., "capability negotiation")
- stream format conversion (may be required)
- stream mixing or switching
- determination of conferencing configuration (may be required)
- master/slave determination (may be required)

Logically, these functional components can be implemented within arbitrary parts of the infrastructure, the terminals, the audio / video applications, or any of combination of these

4) Here, the notation 'audio / video' should be read as "audio and/or video". It refers to conferences and contexts which are audio-only, video-only, or both audio and video. This term is later distinguished from the term 'audio-video'.

5) I.e., the user is the focus of stream content, and can additionally act as a dynamic stream controller.

elements. To simplify this discussion, however, we will group these functional components into higher-level components, and denote them with terms which intend to reflect — though neither specify nor restrict — their roles. It must be remembered that the grouping and naming used here is simply a matter of convenience.

The higher-level components employed in the following discussion are:

- **gateways** (for control protocol transcoding and/or stream format conversion),
- **multipoint controllers**, or "**MC**s" (for capability negotiation, determination of conferencing configuration, and/or master/slave determination); and,
- **multipoint processors**, or "**MP**s" (for stream mixing and/or switching).

As before, it should be clearly understood that these higher-level components can be resident within either common or independent physical elements of the infrastructure, perhaps even resident upon the terminal or within the application itself.

At this point, we return to a more thorough discussion of the basic functional components.

Control protocol transcoding: This functional requirement arises when two participating applications (and/or terminals) employ incompatible control protocols. In such cases, a gateway can be employed to transcode the control signals, thereby enabling compatible control signalling between those applications / terminals.

Capability negotiation: This functional requirement may arise when applications can send and/or receive more than one stream format per media type. What must be resolved in such cases is exactly which audio (and/or video) stream format each participating application should send and/or receive. The task of negotiating and determining the formats to be sent and received amongst participating applications can be:

- resolved by (some set of) the participating applications, or
- delegated to an MC, to resolve on behalf of the participating applications.

Stream format conversion: This functional requirement arises when there is no single, common stream format which can be sent and/or accepted by all participating applications. In such cases, two or more stream formats (for one or more media types) must be employed within the conference. In order to complete such transmission, gateways can be employed to convert one stream format into another.

Stream mixing or switching: This functional requirement arises when engaging in audio / video conferences with three or more parties. Multipoint processors (MPs) are employed for stream mixing or switching in such cases. As a working definition here, we employ (parts of) the description of an MPs functionality as cited in [16]:

> *The MP may process one or more media stream types.*
>
> *An MP which processes video shall provide either video switching or video mixing. Video switching is the process of selecting the video that the MP outputs to the terminals from one source to another. The criteria used to make the switch may be determined through detection of a change in speaker (sensed by the associated audio level) or through [other] control. Video mixing is the process of formatting more than one video source into the video stream that the MP outputs to the terminals. An example of video mixing is combining four source pictures into a two by two array in the video output picture. The criteria for which sources and how many are mixed is determined by the MC [or] other controls...*

> *An MP which processes audio shall prepare N audio outputs from M audio inputs by switching, mixing, or a combination of these. Audio mixing requires decoding the input audio to linear signals (e.g., PCM or analog), performing a linear combination of the signals, and recoding the result to the appropriate audio format. The MP may eliminate or attenuate some of the input signals in order to reduce noise and other unwanted signals. Each audio output may have a different mix of input signals providing for private conversations.*
>
> *The MP terminates and sources the media channels.*
>
> *from "Draft ITU-T Recommendation H.323V2 - Packed Based Multimedia Communication System" [16], p.31*

Determination of conferencing configuration: Here, the term 'conferencing configuration' is used to refer to the configuration of the control and media streaming channels between participating applications (e.g., centralized vs. distributed control, centralized vs. distributed stream spreading, etc.). This functional requirement arises when there exists more than one configuration by which to establish/organize these channels. The task of determining the conferencing configuration amongst participating applications can be:

- resolved by (some set of) the participating applications, or
- delegated to an MC, to resolve on behalf of the participating applications.

Master/slave determination: This functional requirement may — though not necessarily — arise when the conferencing configuration is such that more than one multipoint controller (MC) becomes involved in the establishment and/or maintenance of the control and media streaming channels between participating applications. Depending upon the manner in which the MC functionality is implemented and integrated with the conferencing application, the presence of two or more MCs in the conferencing context may require that some form of master/slave determination be carried out amongst the MCs. As one might expect by the name, the result of such determination is that one MC becomes the "master" MC, while the others MCs subordinate their control to that master.

4.4.2.2 Architectural issues

Some of the basic functionality described above constitutes essential parts of the conference set-up and establishment phases (e.g., capability negotiation, determination of conferencing configuration, master/slave determination). Some of the other functionality (e.g., control protocol transcoding, stream format conversion) concerns signal and data transformations required in order that the participating applications exchange compatible information.

Conferencing configurations

In this section, the focus is upon the *conferencing configuration* of the "conference-in-progress" — that is, the configuration of the control and media streaming channels between the participating applications. Conferencing configuration is one of the primary architectural characteristics which limits or enables the capacity to develop and deploy seamless conferencing applications.

The alternative conferencing configurations depicted in figure 6 are the result of the combinations of possible configurations allowed and/or preferred by the media streaming application architecture, together with the architecture of the supporting infrastructure. For each of the audio / video control and media streaming channels, the figure illustrates the two fundamental alternatives available: centralized spreading vs. distributed spreading[6].

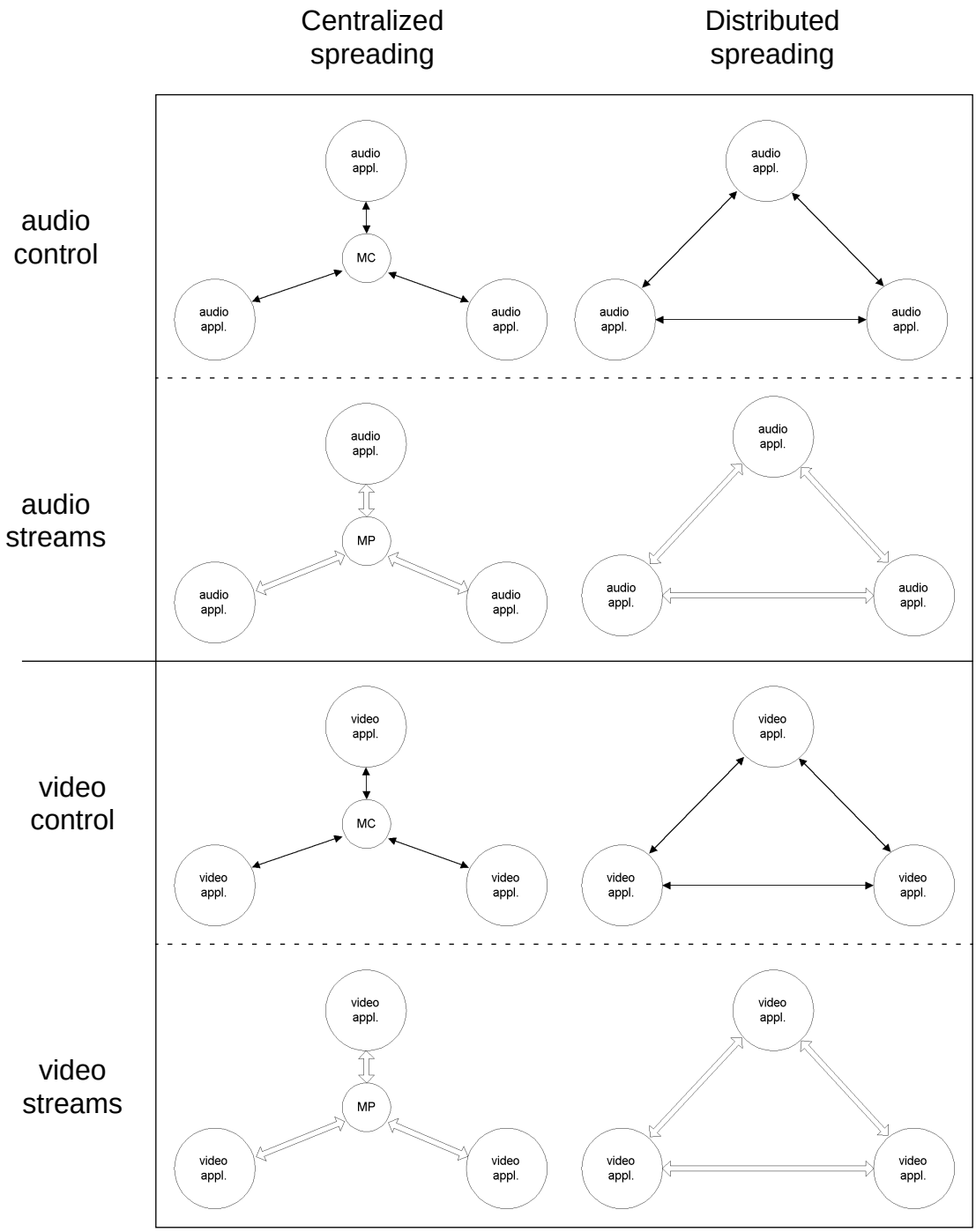


Figure 6 : *Alternative configurations of control and media streaming channels*

All of the centralized spreading models involve the use of an MC-like or MP-like process, depending upon whether it is the spreading of control or stream information, respectively.

6) It should be understood that in order to increase efficiency, **multicast** can be used within either of the spreading approaches; it is not a technical requirement for multi-point spreading, however.

The figure makes no assertions as to where these MC / MP processes are resident; they could be hosted upon a terminal which hosts one of the audio / video application processes or, alternatively, upon some other non-terminal device.

When spreading control information using a distributed spreading approach, certain MC-related functionality is required within the terminal and/or audio/video conferencing application. The same is true for streams: when spreading streams using a distributed approach, certain MP-related functionality is required within the terminal and/or audio/video conferencing application.

For a conference including both audio and video media types, it is possible that the conferencing configuration is very complex. For each row in figure 6, one of two fundamental spreading approaches is possible. *This yields 16 basic, alternative conference configurations for a conference including both audio and video media types.*

Other hybrid forms of conference configurations are possible to construct, as depicted in figure 7. The figure illustrates a configuration in which an MC and an MP are co-located within some element, and a fully distributed spreading model is joined with a fully centralized spreading model via that element. Still other hybrid forms can exist: for example forms in which more than one MC / MP is involved in connecting the participating hosts within the conference.

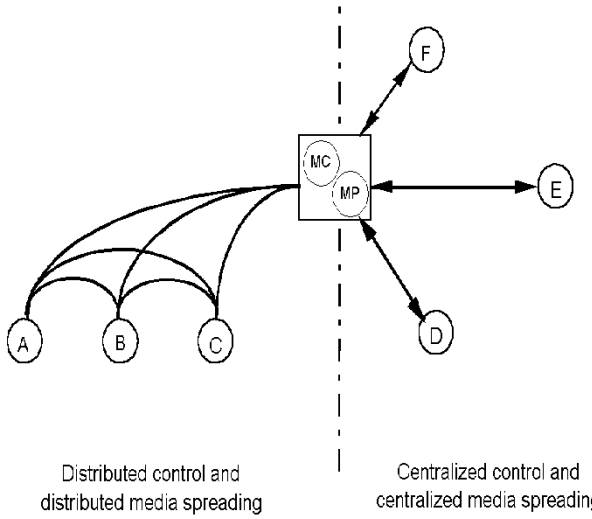


Figure 7 : *Example of a hybrid conference configuration*

Audio-video conferencing applications

Most video conferencing applications are, in fact, not limited to handling video only; most handle audio as well. That is, the application itself handles both the media types[7]. When considering the architecture of such applications, it is necessary to attend to the manner in which each of these media types is treated and managed by the application,

One vital consideration is the format of the media stream(s) sent and received by such applications. Some employ stream formats in which the audio and video information is woven together into the same stream. Others employ stream formats which are audio-only

7) Here, these kinds of applications are denoted as **audio-video conferencing applications**.

and video-only. Usually, the more well-developed audio-video conferencing applications can send and receive a variety of media stream formats (e.g., several types of audio-only formats, several types of video-only formats, certain types of combined audio-video formats, etc.)

It should be noted that when conferencing applications can send and receive a variety of media stream formats, the capability negotiation phase of conference set-up can have a very significant impact upon the final conferencing configuration the system settles upon.

4.4.3 Reference models for sharing audio, video and data

From a logical perspective, a **conference session** includes the totality of audio / video control and media stream channels employed amongst the participating hosts, along with all control and data channels used to achieve shared data applications within the conference.

When it comes to assembling the technology required to carry out a conference involving audio, video and data, there exist three basic options:

1. use of an audio-only application, a video-only application, and sharing one or more data applications[8] in parallel;
2. use of an audio-video application, while sharing one or more data applications in parallel;
3. use of an audio-video application and one or more data applications, *all* of which have been developed from the same underlying **conference framework**.

Options 1 and 2 suffer the unfortunate consequence that the applications involved in the conference have little to no knowledge of one another with respect to the conferencing session. This leads to a situation in which the coordination of the applications within the session becomes cumbersome for the user[9] and, at times, completely impossible to manage.

With respect to the third alternative, a conference framework is a set of objects and classes which realizes a conferencing architecture. This kind of framework and architecture provides a uniform set of mechanisms and means by which to create applications which "understand" how to engage in multi-user operation. The purpose and motivation for using applications created from the same conferencing framework is that all such applications employ a common set of mechanisms through which they participate, interoperate and remain coordinated within a conferencing session.

Of course, it would be naive to think that all applications will one day be created from one, unique conference framework. Still, when a given conference framework is based in widely employed standards, there will exist a better chance that applications developed from that framework can interoperate with other standards-based applications, even when those other applications are developed outside the given framework.

8) See figure 5 concerning traditional approaches for sharing data applications.
9) For instance, when a new participant is added to the conference, all of the applications involved must be independently informed and updated to reflect this new condition.

In chapter 5, the MEDIATE Application Framework will be presented [17, 18]. MEDIATE is a conference framework from which “session-aware” applications can be developed.

4.4.4 Conferencing and aspects of seamlessness

The purpose of this subsection is to look into some of the different contextual dimensions of seamlessness, as described in chapter 3. The emphasis here is upon the end-user effects experienced when trying to achieve and carry out conferencing within each these contextual foci. Where relevant, some mention is also given to how different architectural configurations may limit or enable the capacity to achieve more seamless conferencing.

In the tables found within this subsection, three types of conferences are designated and addressed:

- **A**: real-time, multi-point audio-only conference
- **V**: real-time, multi-point video-only, or audio-video conferences
- **D**: real-time, multi-point sharing of a non-streaming data application

In these tables, nothing is assumed about the conferencing architecture and spreading models, unless noted explicitly.

Table 12: *Conferencing and* BANDWIDTH

Low Bandwidth	Medium	High Bandwidth
A: low quality should be achievable and usable V: video streaming is essentially achievable / unusable, though emerging compression and data formats (e.g., MPEG-4) are beginning to approach the capacity to support an "acceptable" level of performance. In contrast to streaming, frame capture and exchange could be possible and of value to end-users. Adjustment of frame capture parameters would be required in this case, and users may find benefit in disabling audio-send during frame transmission. D: only applications having "minimal" levels of output (e.g., *chat*) are likely to be tolerated in this context. To function satisfactorily here, all conference types above would benefit from use of multicast. Use of a centralized architecture here could well result in unacceptable delays.	A: high quality shouldn't be a problem, but it's likely that delays will still be experienced during conversation (depends on distance and connection type) V: streaming can be achieved, but frame size, resolution and / or color must be limited. Choice of compression and data format will also affect performance. Start-up delays, jitter and lack of lip-sync may well be experienced. D: Most shared applications should work satisfactorily, though some lack of synchrony may appear during use. Use of replicated or distributed architectures which interchange limited control and data packages is an advantage here. Again, multicast will boost performance.	A: no problem. V: high quality, smooth streaming should be the rule (rather than exception) here, though start-up delays may still occur. D: Most shared applications should work quite well. Type of application architecture should not have a great impact. Multicast is always a plus.

Table 13: *Conferencing and* CONNECTIVITY

No connection (i.e., connected "now and then")	Poor / partially connected	Constantly connected
A, V, D: conferencing not possible when not connected. Note that connections "now and then" will require the presence of some kind of "location" / directory service, in order that potential conference participants can be contacted / addressed. In addition, it is technically possible to create recordings of conference content which could be played back at a later time.	A, V, D: depends upon stability of the connection. If connection fails/drops too frequently, it is likely that conferencing attempts will be aborted.	A, V, D: no negative effect upon quality of conferencing.

Conferencing and CENTRALIZED VS. DE-CENTRALIZED

It should be apparent from sections 4.4.1 to 4.4.3 that the simple terms 'centralized' and 'de-centralized' are not sufficiently specific on their own; in fact, the term 'de-centralized' has purposefully been discarded from use within this discussion about conferencing. Instead, the focus has been upon alternative approaches for achieving data application sharing, along with focus upon the basic models available for spreading control and stream information.

Regarding seamlessness, some discussion of the end-user effects experienced as a result of different conferencing configurations has already been provided above.

Table 14: *Conferencing and* TERMINAL TYPE

Except for “Borrowed desktop” below, all cases presume the existence of the required devices and drivers. A homogeneous or *compatible* execution environment is also assumed (i.e., a compatible execution environment entails, for example, that any shared applications in use are not affected by a conferencing context which includes desktops, laptops and PDAs within the same conference).

Palmtop	Laptop	Borrowed desktop	Desktop
A: should be usable, presuming the required audio devices are available (e.g., mic, speaker). The *quality* of the audio devices themselves can also be the most significant factor here. V: possible, but screen size and display clarity may limit effective use. D: possible, but screen size may limit effective use.	A: should be OK. V: should be OK, but display clarity may limit effective use. D: should be OK.	A, V, D: cannot succeed unless required devices, drivers and applications are (or can be put) in place. D: security issues and implementations may hinder use of shared applications, especially when the architecture employed requires pre-distribution of data-to-be-shared, and *storage* of that data on the borrowed machine / environment)	A, V, D: OK.

Conferencing and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS

Regardless of platform, conference types A and V cannot wholly succeed unless:

- each sender, or some gateway, is able to produce a stream type which each receiver can handle
- each sender, or some gateway, can generate control packets which can be understood by each of the receivers (e.g., signals based upon the same standard).

The shared applications within conference type D cannot wholly succeed unless:

- each sender, or some gateway, can generate control and data packets which can be understood by each of the receivers.

In cases where the conditions above are only partially satisfied, certain participants may experience lack of audio and/or video, and/or experience unpredictable/unstable behavior of the shared applications.

For conferencing situations, running identical applications on compatible platforms will always deliver the most stable and best performance. Use of standards (e.g., T.120) is

required in order that interoperation of different conferencing applications on different platforms will have any chance whatsoever of success.

Conferencing and REAL-TIME

All conference types are defined to be real-time. For all conference types, the presence of delay can lead to lesser or greater disturbances within the conferencing context.

A considerable amount of literature emphasizes that the existence and quality of the audio channel is the most important factor within a conferencing context. Research has also been carried out which demonstrates that the amount of delay tolerated within an audio channel is quantifiable.

Within type V conferences, it is reported that lack of lip synchronization is also a highly disturbing problem.

For shared applications within type D conferences, real-time demands again lead to a need for minimal delay between the appearance of synchronized, consistent views for all participants.

4.4.5 Discussion

Research has shown that the most essential ingredient within conferencing is the presence of a satisfactory audio connection; here, the connections must have sufficient quality, including medium-to-good average voice quality, and a (very) low drop-out frequency. Audio-only conferences can be achieved using only limited bandwidth, and carried out amongst most types of terminals. However, to achieve audio conferencing amongst a set of heterogeneous environments, applications and/or terminal types requires the use of standards and, in some cases, the use of gateways[10].

Video conferencing requires significantly more bandwidth than audio conferencing; this is especially true when considering person-to-person conferences as we have done here. Certain exceptions do exist, however, but many of these fall outside the use contexts we intend to focus upon in this discussion. Very low quality person-to-person video conferencing may be possible upon palmtop terminals, though again, the areas of use are limited.

The area of security can have a highly variable impact upon conferencing. The impact can be anything from “no impact” to a “full stop” in the effort to achieve conferencing.

In regard to architecture, we find that the best architectural approach is:

- to select a standards-based conference framework as the basis for development of all conferencing applications
- to select a framework which employs distributed spreading models for both control and stream information
- if possible, select a framework which implements distributed spreading using multicast technology where appropriate
- if possible, select a framework which implements and treats both audio and video as primitive data types.

10) The same is true for video and data sharing conferences, as well

We believe that this architectural approach is the one which enables the capacity to achieve more seamless conferencing in the greatest variety of cases and contexts. It is worth paying the price for more complex control protocols, etc., in exchange for the flexibility and robustness of the applications which can developed from this approach.

In chapter 5, we will see that these characteristics have formed some of the motivating criteria in selection of the MEDIATE Application Framework for further pursuit, study and development within the IMiS-Kernel project.

4.5 Video streaming

4.5.1 Reference model

Video streaming applications will likely cover a wide range of applications in the future. Two emerging applications are **Digital Video Broadcasting** (**DVB**) and Video on Demand (VoD) services. In the home market, these will cover the functionality of today's TV and Text-TV, in addition to multimedia services. Scenarios and more information on digital television can be found at the DVB web pages [20].

Other applications will be established in professionally-oriented video applications, where video content is used in production, documentation, research, archiving and surveillance. Hyperlinks will be used to integrate streaming video with other data types, such as text, still images, graphics, sound documents, animation, etc.

Video streaming is more than just playing a video on a computer screen: Instead of playing a film sequence from a file or a CDROM, the data stream comes from a remote server through a data network. The data source is called **video server**, which contains a database of video information, a coding system, and a communication system for sending the video stream. On the other side the **video player** receives the data stream; it is decoded, then presented to the user.

The original definition of video streaming includes some kind of **rate control** and real time facility. Video streaming is then either a rigid application (i.e., the application works well only when the quality of service (**QoS**) exceeds a certain level); or, an adaptive one (i.e., the application can adjust to variations in quality of service).

Transfer of video data without rate control is often called **data piping**, **asynchronous data streaming**, or **http-streaming**. On the receiving end, the stream is buffered to a certain degree, and the video stream is played as long as there is content available. In this situation, the application behaves as a sort of "elastic" application. For example, many Java Media Framework (JMF)-based players are founded on the http-streaming paradigm.

The protocols used for video streaming are specially designed for real time demands. These include RTP (Real Time Protocol) and RTCP (Real Time Control Protocol) [21] and RTSP (Real Time Streaming Protocol) [22]. Other related protocols can be found on the IETF Home Pages [23].

Here we use **MPEG** as a reference [24], though other streaming formats exist (e.g. *RealVideo*, see overview [27]). It must be noted that although video contents can be coded and stored in many different formats, not all of these formats are suited for streaming. For example, some formats contain important information at the *end* of the stream. When using such formats, the *entire* content must be downloaded (e.g., via http-streaming) before playing is possible. Other formats — such as MPEG — are designed for streaming; MPEG can also be used as a storing format. DVB will use the MPEG-2 [24] format.

The perceived quality of a video streaming session is influenced by several factors; these include hardware, screen size and type (different technologies), the machine-external environment, operating system, network issues (bandwidth, jitter, delay), coding of the video stream, etc.

In the Lava Project [15], a Video on Demand system including the Elvira video server and the LAVA player was developed. Its components are described in [28], while a closer look at the video server implementation is given in [19].

4.5.2 Video streaming and aspects of seamlessness

Table 15: *Video streaming and* BANDWIDTH

Video streaming is a very bandwidth-intensive application. Bandwidth demands for a full-screen MPEG-2 video stream reach from 1Mb/s up to ca. 10 Mb/s. While MPEG-1 is designed for video streams up to 1.5 MB/s, the MPEG-2 definition supports video streams in a higher bandwidth range, typically 10MB/s. In addition, strong demands on other QoS parameters (e.g., jitter) must be supported to satisfy the user's demands.

In applications or areas where the high bandwidth demands are not fulfilled, adaptation techniques are necessary. Proxy techniques, pre-loading, and carousel access are some of the techniques which can be used. Depending upon performance, these mechanisms may or may not have a significant effect upon the user's perceived quality of the video presentation.

In cases where these kinds of mechanisms are not available/sufficient, other approaches for application adaptation can be employed. These include: selective presentation of content (e.g., audio only), reduction of video quality (e.g., smaller size, less color, less resolution, more jitter, lower frame-rate), etc. Obviously, these kinds of adaptation techniques affect the user's experience of video quality. In more extreme cases, further degradations of content presentation might need to be employed (e.g., still images and text instead of a multimedia presentation). As the reader can easily deduce, bandwidth demands are quite critical when it comes to video streaming.

Low Bandwidth	Medium	High Bandwidth
· proxy solution with intermediate storage necessary (huge disk capacity required). · methods for data compression necessary. · audio-only when bandwidth is too low.	· stamp-size video (30 kb/s) · needs good compression algorithms. · proxy techniques often required.	· 1-10Mb/s · Digital Video Broadcasting (DVB) designed for this situation. · Multicast techniques can further improve performance.

Table 16: *Video streaming and* CONNECTIVITY

Using the term ‘video streaming’ in the pure sense, it seems useless to discuss other than a constantly connected system. In this “pure” context, the client must be connected to the network while playing a multimedia stream. The connection could be achieved through an internet connection (with enough bandwidth), cable modem, satellite link, etc. For broadcasting via a satellite link the communication is one-way only (no return channel from client to server), and thus synchronization is neither possible nor necessary.

It must be pointed out that certain hybrid solutions can be used in order to achieve video streaming for a partially connected system. In such cases, it is possible to download the contents of a video stream to the local disk, and then stream[a] or play the contents locally. In such cases, the end-user might experience that the contents (e.g., a news broadcast) are not the latest available. In addition, the user may have to wait until some part of the stream is downloaded. This solution also has a disadvantage in that it requires huge disk capacity.

Carousel methods open up for periodically repeated transmission of multimedia data. This can be used in a partially connected environment, where the newest version of a periodic part of a transmission is kept on the client. Using this kind of solution, the user may experience certain undesirable effects which liken those described above. The user may also experience an additional delay while waiting for the next video segment to be broadcast (i.e., in cases where broadcasts are not downloaded in the background).

No connection (i.e., connected“now and then”)	Poor / partially connected	Constantly connected
· proxy solutions (implying file storage) is the only solution; this requires huge disk-capacity.	· maintenance of video session can be a problem. · proxy solutions and file storage will be necessary.	· use of QoS mechanisms will yield better quality.

a. Streaming video locally requires that a video server is locally installed.

Table 17: *Video streaming and* CENTRALIZED VS. DE-CENTRALIZED

The question of centralizing/decentralizing a video streaming application is a question of both system design and data storage. A centralized architecture is simpler to implement. However, such a solution can create bottlenecks with respect to data transfer, and data storage. To avoid bottlenecks, a database can be used to organize a distributed scheme. However, distributed architectures must address and solve potential synchronization problems, when data are delivered from servers at different locations.

For end-users, the type of architecture and data storage model employed (i.e., centralized vs. de-centralized) should not affect the user's perception of the streaming quality.

Centralized	Partial / proxy / hybrid	De-centralized
· Video stream usually originates from one central source on one central server.		· Synchronizing problems must be handled when using several decentralized sources or decentralized servers.

Table 18: *Video streaming and* TERMINAL TYPE

Terminal types and profiles are important factors which influence how the recipient experiences a streaming document. Not only the terminal characteristics (size, colors, technology) are crucial, but also the processing hardware on the client computer. Decoding a video stream requires computer power, and it is of advantage to employ hardware decoders; a number of such decoders are available on special-purpose boards which can be easily installed within an existing terminal. Using today's processor technology, software decoders only allow smaller sized video streams in an acceptable quality.

Applications should be able to detect and accommodate themselves to the technology which serves as an interface to the user. Capability negotiation techniques can help find an optimal solution for a given application, device and connection context. There exist a variety of possible devices with different terminal characteristics. **Multimodal** systems can support a variety of interfaces, transforming the contents with respect to the terminal characteristics. Transformation routines can be placed in the network (e.g., when there is a need for a low-bandwidth black-and-white representation of the contents for a small screen).

Multimodality and adaptation techniques are two of the main areas which can help guarantee user satisfaction with regard to the quality of the video streaming application.

Palmtop	Laptop	Borrowed desktop	Desktop
· streaming not available with today's technology. · still, there exist some specially developed solutions for terminals of this size (e.g., MPEG-cam).	· some restrictions exist due to screen technology.	· client must be installed.	· graphics card / MPEG card is an advantage

Table 19: *Video streaming and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS*

In some cases, especially for proprietary systems, the client software is not available for all platforms. However, by using coding standards (e.g., MPEG) in connection with standardized formats and protocols, it is possible to implement clients for all platforms in question. *RealVideo* is one example of a video streaming system which is available for many relevant platforms.

For end-users, the type of platform environment (i.e., homogeneous vs. heterogeneous) should not affect the user's perception of the streaming quality.

Homogeneous	Mostly homogeneous	Heterogeneous
· OK	· availability of clients could be a problem.	· availability of clients could be a problem.

Video streaming and REAL-TIME

The real time demands manifest themselves mostly in terms of delay, i.e. the time for the user must wait for the video stream to be presented on the screen. For some application technologies (e.g. VoD or Near-VoD, material distributed by carousel) delays up to a few minutes can arise. In some cases, such delays *can* be accepted; the determining factor here almost always depends upon the context of use and its associated costs.

Certain other QoS parameters can be eliminated and/or satisfactorily managed within the network and/or application (e.g., jitter can be eliminated through proper buffering). Some such techniques lead to greater delay, however, which can be a point of frustration for the end-user.

Some video streaming applications have extremely demanding real-time requirements. One example is that of simulators which use streaming technology to create the visual and auditory surroundings for the user.

4.5.3 Discussion

Video streaming is a very challenging application, as it places high demands upon hardware capabilities and the network. Seamlessness is rather hard to achieve, and use of hybrid solutions is necessary in many cases. Caching of data, smaller image size, and reduced quality must be tolerated when the infrastructure is suboptimal.

High bandwidth demands (i.e., >1-10 Mb/s) are necessary for a full-quality video which compares with TV quality. Demands upon the control of jitter and certain other QoS parameters are also quite high. However, as demands to delay are not always as harsh, buffering can help avoid the most severe kinds of quality degradation. Users are familiar with the quality of television, and usually won't tolerate a level of quality with is significantly less than that.

To achieve satisfactory quality when presenting the video stream, special hardware is often necessary (e.g. MPEG decoding cards). This kind of "solution" is not so much against the

goals of seamlessness as it is orthogonal to them: an adequate, *seamless* solution should not rely upon the use of specialized hardware. Problems with achieving satisfactory quality also occur also when employing smaller handheld devices; characteristics such as size, weight, energy consumption, availability of a network, etc. all set a limit for the level of quality which can be attained.

Other problems with achieving seamless video streaming arise since as there is no standard framework for streaming video yet. A number of video applications are optimized for streaming via CDROM or file, but not through a network. Other existing solutions can only deliver low quality video to an end-user using a PC.

In conclusion, demanding real-time and bandwidth requirements, as well as processing capacity, make it difficult to develop seamless video streaming solutions for the end-user.

4.6 Database management systems (DBMS)

4.6.1 Reference model

The following figure shows a reference model for DBMS systems. It contains a DBMS server and several client applications. The server contains one or more tables which contain data. Several applications may then use the data.

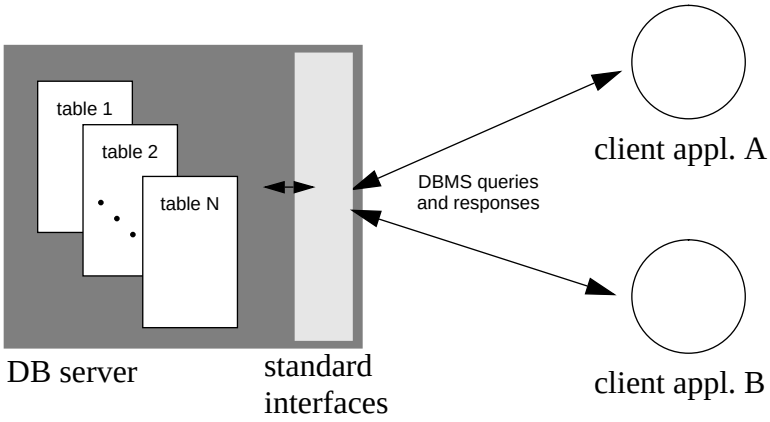


Figure 8 : *DBMS reference model*

The DBMS server may deliver data which is stored at several places. The DBMS is then called a distributed database.

4.6.2 DBMS and aspects of seamlessness

Table 20: *DBMS and BANDWIDTH*

Most of the applications of DBMS do not require much bandwidth. Still, there may be some problems when using low bandwidth. Support for adaptation must be implemented in the clients. One problem is to predict the result of a query. Some queries may return a huge amount of data. In these circumstances the DMBS system should offer mechanisms by which clients can stop/abort the query. Another problem related to low bandwidth is that it may be necessary to lock records or tables for longer periods. This may cause problems for other users, since they may have to wait for the client to unlock the record or table. It is also possible to use proxy or cache solutions when aiming to serve low bandwidth conditions.

Low Bandwidth	Medium	High Bandwidth
· Little support for adaptation - Hard to predict the result of a query. · Locking of records for longer periods · Proxy/cache solutions are often necessary · Based on best-effort. Flow control done by TCP	· No problems, except for response times	· No problems

Table 21: *DBMS and* CONNECTIVITY

This dimension is the hardest to solve from a seamlessness perspective. DBMS is built around the concept of short transactions. That is, every update of the database should be done as one short operation. If that is not possible (records/tables are locked), the transaction should not be done. The problems appear when DBMS systems should work when network connection is not available or not stable. In these situations the DBMS system can not lock the records or tables. This can cause the "lost update problem". To solve this problem we need mechanisms for off-line use. This means to replicate (part of) the database to the local disk, and synchronize the databases when connected. The system must then handle conflicts if the same record is updated by several users. No general solution exists for handling conflicts, since correct resolution depends upon the semantics of the application. Conflicts must be solved by the users in many situations.

No connection (i.e., connected"now and then")	Poor / partially connected	Constantly connected
· Need mechanisms for off-line use. · Missing mechanism for cost optimization. · Change notification missing · Synchronization mechanisms are necessary. · Need support for long transactions · Approaches for conflict resolution must be explicit and understood by users; methods and tools are also required · Conflict resolution requires user knowledge	· Need proxy/cache mechanisms · Need support for long transactions · Conflict resolution may sometimes be required	· DBMS is designed for this situation

Table 22: *DBMS and CENTRALIZED VS. DE-CENTRALIZED*

Original DBMS system designs were based upon centralized storage of data. Still, it is often necessary to distribute the data storage. A goal for this distribution of the database is to achieve seamlessness, such that the storage architecture is completely transparent to the user; this is done to make the system more efficient in use. Distributed databases complicate the problems for the other aspects of seamlessness discussed in this chapter, and especially the bandwidth and the connectivity dimension.

Centralized	Partial / proxy / hybrid	De-centralized
· DBMSs were originally designed for centralized storage of data	· Replication may be necessary. · Horizontal or vertical[a] dividing of the database? Depends on use. · Checking and updating foreign keys may be a problem	· Solved by replicating or dividing the data on several servers · Hard to update foreign keys · Synchronization/mirroring/replication

a. For example, each department in a company can have their own customer database. This is horizontal division of the database, since all of the tables are replicated; still, only the relevant data are stored in each local database. Vertical division of the database is when tables are divided amongst several distributed databases. For example, the customer-relevant tables can be stored at the marketing department, while the employee-relevant tables be stored in the administration department.

Table 23: *DBMS and TERMINAL TYPE*

Regarding terminal types, most of the problems rather general for each type. The biggest challenge is for the PDA (Palmtop). The challenge is to find a solution by which to display big tables in a usable manner. PDAs require specially-designed clients to solve this problem.

Palmtop	Laptop	Borrowed desktop	Desktop
· How to presents big tables? · Special designed clients	· OK	· Needs client software (e.g., WWW-interface)	· OK

DBMS and HOMOGENEOUS VS. HETEROGENEOUS PLATFORMS

A database server can serve clients operating upon different platforms. The most common approach, in such cases, is that:

- the server offers a set of standard interfaces towards the client (e.g., ODBC, JDBC, SQL, etc.), and
- each client architecture employs that interface within its own implementation.

When database clients and servers are constructed in this manner (i.e., the client and server are *compatible*), the user's experience of the application is *not* affected by the homogeneous vs. heterogeneous nature of the platforms. What may instead affect the user's experience of the application is the application's user interface, its features, the type of connection and/or bandwidth, etc.

DBMS and REAL-TIME

The real-time demands for a DBMS are directly related to the application area in which the system is to used. It is not hard to imagine applications at each end of the spectrum: for instance, a DBMS could be designed and employed as part of a hard-critical, guaranteed-response real-time system (e.g., within a power plant). Alternatively, a DBMS could be employed as part of an off-line request-handling system (e.g., as part of a mechanism for searching for and retrieving literary references, articles and abstracts within a library).

Like most other services and applications, most users wish that a database system respond as fast as possible to every request. As suggested above, however, the time when the user actually requires the data can vary greatly.

Chapter 5

Characterization and Selection of the Application

5.1 Revisiting the Application Requirements

The initial work process described in chapter 2, followed by the work and results reported in chapter 4, put the DP1.0 team in a new position. From that point, two major alternatives were proposed by which to further the effort toward characterization and selection of the application for iMiS-Kernel.

When considering these alternatives, it was judged by the team that both options required analysis work which would likely be too ambitious to achieve within the project's prevailing time-constraints. Facing this situation, it was necessary for the DP1.0 team to limit the focus of their further studies. It was therefore decided to consider only one of the service/application types which had been examined in chapter 4 — other service/application types could be pursued later. The team therefore reviewed once again the characteristics required of the application. As listed in section 2.1, these included that the application should:

a) focus upon establishing a communication service which meets demands within the IMiS-Veritas project;
b) support cooperative work for some given work situation;
c) enable testing of mechanisms for achieving seamlessness, as provided by DP2.0;
d) enable experimentation with QoS; in particular, experimentation with bandwidth reservation and bandwidth management (e.g., IPv6, RSVP, etc.);
e) enable experimentation with technology for enabling mobility (e.g., MobileIP);
f) allow for experimentation with heterogeneous networks, systems and terminals;
g) handle and operate with multiple media types; and,
h) transmit audio and/or video streams, in order to generate a high volume of network traffic.

5.2 Focus upon Conferencing and Recommendation of MEDIATE

In re-considering the needs of the mobile workers at DnV, it was judged that an application which supported conferencing could be of significant use for those personnel. As mentioned in section 2.2, the IMiS-Veritas project had identified a need for mobile workers to be able to contact co-workers on-the-spot, for information exchange and problem solving.

The decision to focus upon conferencing applications also matched well with respect other project requirements. In addition to meeting points (a) and (b), most conferencing applications also satisfied points (g) and (h).

The decision to focus upon conferencing was also influenced by another, at least equally significant factor. At that point in the DP1.0 effort, an in-house system began to emerge as a strong candidate for the IMiS-Kernel application. The MEDIATE Application Framework [17, 18] was brought to our attention, a system designed and developed by Dr. Steinar Kristoffersen as part the doctoral work he was completing at that time.

The MEDIATE Application Framework includes a conferencing framework through which it is possible to develop a variety of interoperable, “session-aware” applications. The MEDIATE architecture also realizes some of the desirable design characteristics for conferencing application, such as a distributed spreading model for both control and stream information (see section 4.4.5).

Other important factors also promoted MEDIATE as an application candidate. Perhaps most significant of these was that we had, in-house, both access to and knowledge of the MEDIATE software. This condition created a situation in which requirements (c), (d), (e) and (f) could also be addressed within the project.

With the requirements for application selection so thoroughly addressed, the DP1.0 team has thereby recommended that the MEDIATE Application Framework be used as the foundation for application development and experimentation for continued work within the IMiS-Kernel project.

References

[1] WWW Homepage for IMiS-Kernel:
http://www.nr.no/imis/imis-k/

[2] WWW Homepage for IMiS-Veritas:
http:// www.nr.no/imis/veritas/

[3] Alvestrand, H., Børseth, H., Lovett, H., Ølnes, J., Final report for IMiS feasibility project: Potential for a main program with focus on mechanisms for and use of multimedia applications in seamless networks, NR Note IMEDIA/04/97, Oslo, January 1997.

[4] Borenstein, N. & Freed, N.: *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC 1521, September 1993.

[5] Crispin, M: *Internet Message Access Protocol - version 4rev1*, RFC 2060, December 1996.

[6] Crocker, D. H.: *Standard for the Format of ARPA Internet Text Messages*, RFC 822, August 1982.

[7] Henshall, J. & Shaw, A.: *OSI Explained. End to End Computer Communication Standards*, Chichester, England: Ellis Horwood, 1988.

[8] Lovett, H. & Skogseth G.: *Tjenester i Datanett - en innføring i OSI applikasjoner*, Universitetsforlaget, 1992.

[9] Moore, K.: *MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text*, RFC 1522, September 1993.

[10] Myers, J & Rose, M: *Post Office Protocol - version 3*, RFC 1939, May 1996.

[11] Postel, J.B.: *Simple Mail Transfer Protocol*, RFC 821, August 1982.

[12] Horton, M. & Adams, R.: *Standard for Interchange of USENET Messages*, RFC 1036, December 1987.

[13] Kantor, B. & Lapsley, P.: *Network News Transfer Protocol*, RFC 977, February 1986.

[14] Spencer, H.: *News Article Format and Transmission*, Internet Draft (son of RFC 1036), June 1994.

[15] WWW Homepage for LAVA:
http:// www.nr.no/lava/

[16] Draft ITU-T Recommendation H.323V2 - Packed Based Multimedia Communication System, March 27, 1997.

[17] Kristoffersen, S., Multimedia-Enhanced Collaborative Situations, Proceedings of the 20th IRIS, August, 1997, Hanko, Norway. See also:
http://www.ifi.uio.no/iris20/proceedings/32.htm

[18] Kristoffersen, S., MEDIATE: An Object-Oriented Framework For Developing Collaborative Multimedia, Proceedings of ECSCW`97 Workshop on Object-Oriented Groupware Platforms, September 1997, Lancaster, UK.

[19] Sandstå Olav, Langørgen Stein og Midtstraum Roger: “Video Server on an ATM Connected Cluster of Workstations”, Institutt for datateknikk (NTNU), http://www.idi.ntnu.no/IDT/grupper/DB-grp/tech_papers/SCCC97_elvira/sccc97.ps.gz

[20] Web pages for Digital Video Broadcasting (DVB): http://www.dvb.org

[21] RFC1889: RTP / RTCP

[22] RFC 2326 RTSP

[23] IETF Web Pages http://www.ietf.org/

[24] MPEG Web Pages http://drogo.cselt.stet.it/mpeg

[25] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall. MPEG Video Compression Standard. Chapman&Hall, New York, 1996.

[26] Seminar Multimedia und Electronic Publishing, Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe, 1997, http://i31www.ira.uka.de/docs/mm+ep/

[27] B, Foyn, B. Johansen, H. Lovett, T. Sørensen. Technical Survey Video-on-demand systems. Norwegian Computing Center, http://www.nr.no/lava/kjerne/docs/tech_survey/tech_watch.html

[28] Solvoll Dag and Sørensen Tryggve: LAVA - Communication system, Norwegian Computing Center, Notat IMEDIA/03/97, 1997. http://www.nr.no/lava/kjerne/docs/doc/index.html