A CORBA-based Proxy System for Distributed, Mobile Multimedia Applications

Diploma Thesis at the Institute of Telematics
Prof. Dr. Dr. h.c. mult. Gerhard Krüger
Fakultät für Informatik
Universität Karlsruhe (TH)

by

cand. inform.
Michael Ebner

Advisers:
Prof. Dr. Dr. h.c. mult. Gerhard Krüger  Dr. Nigel Davies
Dipl.-Inform. Elmar Dorner  Dr. Adrian Friday
Dr. rer.nat. Jochen Seitz

Fakultät für Informatik  Computing Department
Universität Karlsruhe (TH)  Lancaster University, UK

Day of registration: 1. April 1998
Day of delivery: 30. September 1998
Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 30. September 1998
Abstract

High data transfer rates in modern networks allow the transfer of multimedia data. The increasing data transfer rates in wireless networks made it also possible to transfer multimedia data to mobile computers. However, the wireless link has its own characteristics which makes special handling necessary.

The proxy concept is one solution to handle the special constraints on wireless links. The idea is to filter the multimedia stream if the capacity on wireless link decreases. This leads to the transfer of multimedia streams with less quality. It is better to have an ongoing video with low picture quality than a stagnating video with good picture quality. Therefore a proxy system must be installed which inserts filters if necessary and deletes filters if possible.
Acknowledgement

First, I would like to thank Dr. Jochen Seitz who offered me to do this thesis in Lancaster and Dr. Nigel Davis who accepted me as visiting student in Lancaster. The regularly discussions with Nigel, Jochen and Dr. Adrian Friday were passionate but each time productive for my work.

Furthermore, I wish to thank Matthew Storey and Keith Mitchell (he is cool!) which were my office mates. All the time they helped me with my English and it was always an amusing atmosphere. I would also thank Rui for spending our spare time together. I will miss the cake time each Friday morning where we ate sweet cakes and talked about all and nothing.

I would also thank the DMRG for their kindly welcome, especially all people who have played football with me on each Thursday even if the weather was rainy, cold and stormy.

Finally, I have to thank my parents for providing so much support all the years for my unusual career path.
Contents

1 Introduction 1
  1.1 Overview ............................................. 1
  1.2 Aims ................................................. 3
  1.3 Thesis Outline ..................................... 4

2 Multimedia and Mobility 5
  2.1 Multimedia ......................................... 5
    2.1.1 Definition and basic concepts .................. 5
    2.1.2 End systems .................................... 7
    2.1.3 Communication ................................ 8
    2.1.4 Quality of Service (QoS) .................... 11
    2.1.5 Moving Picture Experts Group (MPEG) ....... 14
  2.2 Mobility ........................................... 19
    2.2.1 Basic concepts .................................. 19
    2.2.2 End-systems .................................... 21
    2.2.3 Communication ................................ 23
  2.3 Adaptation for Varying Quality Of Service ....... 29
  2.4 Summary ............................................. 30

3 Common Object Request Broker Architecture (CORBA) 33
  3.1 The CORBA Standard ................................. 34
    3.1.1 The Object Model and Interface Definition Language 35
    3.1.2 ORB Architecture ................................ 39
    3.1.3 Interoperability and Portability ............... 42
  3.2 CORBA services, CORBA facilities and CORBA domains .................................................. 44
  3.3 Summary of CORBA ................................ 45
## Table of Contents

4 **Proxies** 47

4.1 Design Pattern Proxy ........................................... 47

4.2 Proxies in network area ........................................... 48

  4.2.1 Proxies in mobile computing ............................... 50

  4.2.2 Shortcomings of approaches ............................... 52

4.3 Reactive and Adaptive Proxy Placement (RAPP) ............... 52

  4.3.1 Architecture overview ..................................... 53

  4.3.2 Classification of Proxy Objects ........................... 53

  4.3.3 Proxy Selection Process (PSP) ............................ 58

  4.3.4 Proxy Installation Process (PIP) ......................... 58

4.4 Summary ............................................................. 61

5 **Implementation** 63

5.1 Application constraints ........................................ 63

5.2 RAPP Architecture Elements ................................... 64

  5.2.1 QoS Service Module ....................................... 64

  5.2.2 CORBA Adapter ............................................. 65

  5.2.3 Factories .................................................... 71

5.3 Prototype ............................................................. 74

  5.3.1 Existing programs ........................................... 74

  5.3.2 RAPP Architecture Elements ............................. 75

  5.3.3 Additional Elements ....................................... 78

  5.3.4 Functionality ............................................... 79

5.4 Summary ............................................................. 81

6 **Evaluation** 83

6.1 Evaluation Environment ........................................ 83

6.2 Proxy Insertion Evaluation ..................................... 86

6.3 Proxy Removal Evaluation ..................................... 88

6.4 Summary and Conclusion ....................................... 90

7 **Conclusion** ........................................................ 91

A **Appendix** ............................................................ 93
1. Introduction

This diploma thesis was supervised by the Institute of Telematics at the University of Karlsruhe, Germany in cooperation with the Computing Department at the Lancaster University, United Kingdom and carried out in Lancaster.

1.1 Overview

Nowadays, enabling technologies like high-speed networks and improved data processing power lead to the use of distributed multimedia applications. Additionally, multimedia hardware for playing audio and video is supported in end-systems. Furthermore, high data transfer rates in modern networks allow the transfer of multimedia data over networks and not only direct from CD-ROMs. However, the downloading of high-quality audio and video files from the Internet can lead to very long, unacceptable transfer times and playback latency. Therefore, streaming applications are preferred to avoid these disadvantages wherefore the client plays received multimedia data immediately requiring real-time capabilities. This enables distributed multimedia applications like multimedia conferencing feasible.

The restrictive time requirements and mass volume of data arise new problems which have to be considered leading to the introduction of Quality of Service (QoS) that has to be negotiated and monitored and may be modified. This is a complex task in the Internet because it interconnects sites with widely varying bandwidth capabilities and it only offers a best-effort service.

The emerging market of mobile end-systems and mobile networks with increased data transfer rates and interconnection to static networks also demands distributed multimedia applications. Mobile end-systems have heavily increased in variety and performance ranging from mobile phones to higher performance laptops. Therefore, a slogan for the new trend is:

Multimedia goes mobile.

Nowadays, users of mobile computers want to execute the same applications and to have access to the same information as they would when connected to a fixed network. Therefore, mobile networks should support the same QoS as static networks.

In the future the Internet will see the rollout of ATM, RSVP with the ability to control Quality of Services (QoS) and mobile networks with widely varying QoS (Hunter, Witana & Antoniades 1997).
However, mobile networks using wireless links which have their own characteristics and this makes a special handling of data necessary. These differences can be expressed with parameters like delay, throughput and error rate which have a significant impact on demanding applications especially on multimedia applications with its special time and volume requirements. Therefore, to provide the same QoS as in static networks an adaptation of the transferred data, used protocols, etc. is necessary to reflect the special characteristics of wireless links (Katz 1994).

The most common adaptation techniques used are delay of sending, filtering data or the caching of data. For example, multimedia streams could be filtered to reduce required data volume which leads to the transfer of multimedia streams with slightly less quality. However, it is better to have an ongoing video with low picture quality than a stagnating video with good picture quality. A few images of the real-time transmission of Bill Clinton's testimony video in the Internet from the 21st September 1998 (Broadcast.com 1998) can be seen in figure 1.1. The top row shows three good images, whilst row two shows three bad images and the last row shows the used application RealPlayer suffering from net congestion.

Figure 1.1: Images of Clinton’s Grand Jury Testimony Realvideo

Congestion messages occurred many times and the longest time was about 150 seconds in an observation time of two and a half hours. This and the bad images show that in static networks transmission of real-time video is not perfectly solved. If multimedia streams are still a problem in static networks then they are especially critical in mobile networks.

The proxy concept is one solution to realize adaptation techniques for handling the special constraints in wireless links. The idea is the insertion of a proxy between client and server which can filter, cache or transform data and may use other
protocols to adapt to the wireless link. In current solutions a proxy is statically inserted therefore the stream establishment is between server-proxy and proxy-client. Thereby, the proxy placement has to be decided prior to communications between client and server and insertion of further proxies or removal of a proxy is not possible. Hence, the static insertion lacks in extensibility, flexible behaviour, flexible placement and scalability. Therefore, a dynamic proxy system has to be developed which inserts proxies if necessary and deletes proxies if possible. The proxy system should use CORBA for all remote signaling messages which provides location and system architecture independence.

1.2 Aims

The main aim is a running demo to prove the feasibility of the CORBA-based proxy concept for the mobile computing environment with multimedia streams. This should be a basic work for the integration of filter proxies in any application which deals with the specified problems.

The following items are fundamental:

- **Concept for proxy-based approach with special constraints.** This includes basic solutions for
  - the overall behaviour of the proxy system,
  - the location problem for each proxy,
  - the communication between the distributed elements,
  - the integration of filters.

- **Specification and design of interfaces with IDL.** This applies to the interfaces between proxies and the interfaces between a proxy and its environment. The aim is the development of APIs to support easy integration in applications.

- **Prototype implementation of the system,** supporting filtering of MPEG streams.

The result is a basic proxy system with fundamental functionality. The items

- security aspects (authentification, secure data transfer, . . .) or
- accounting

will be considered in other works.

The existence of the design and specification of the following points is assumed:

- QoS-Parameter for multimedia streams
- multimedia stream types

The existence of filters is also assumed.
1.3 Thesis Outline

This remainder of this thesis is structured as follows. Chapter 2 contains a discussion of multimedia and mobility with their special requirements for end-systems and both wired and wireless communications, the remainder of the chapter then introduces some adaptation techniques. Chapter 3 introduces an overview of CORBA with main focus on the required features for the prototype. Chapter 4 explains the concept of proxies, analyses existing proxy approaches to show their shortcomings and defines the Reactive Adaptive Proxy Placement (RAPP) architecture to overcome the shortcomings. In chapter 5 the implementation aspects of RAPP will be shown and the implemented prototype is explained. To evaluate RAPP the proxy insertion/removal process is discussed in chapter 6. Finally, the thesis is summarized, conclusions are given and future work is listed.
2. Multimedia and Mobility

Nowadays, computer technology is capable of handling multimedia data, especially audio and video. The hardware, storage and communication aspect of multimedia is widely deployed for professional applications. The success of multimedia communication is a consequence of the success of the Internet with its special tariffing, architecture and technology. For commercial success the Internet and public telecommunications with their totally different basic approaches have to grow together. Furthermore, the world-wide, seamless and independent use of wired and wireless networks will provide a new era of mobility. However, mobility should not lead to the abandonment of multimedia data and especially of multimedia communication. Therefore, the link between multimedia and mobility shall be described in this chapter.

Firstly, the communication aspect of multimedia shall be illustrated. Secondly, the mobility approach and its problems will be explained. Finally, to combine multimedia and mobility together the adaptation of the communication is necessary.

2.1 Multimedia

The use of computers for multimedia applications lead to a widespread discussion, and made computers interesting for new areas like entertainment and has also lead to use in daily life. It is not sufficient to consider the processing of multimedia data at the end-system since people want to transfer multimedia, too, both end-systems and underlying networks must consider the demands.

However, what is meant by multimedia and what special constraints have to be considered in computer systems? To clarify this in the following section multimedia is defined and basic concepts are described. This section also outlines the requirements for end systems and multimedia communication. The Quality of Service (QoS) consideration to meet the new requirements is then introduced and finally the video compression standard MPEG is described to show a data format for audio and video data.

2.1.1 Definition and basic concepts

Firstly, the word multimedia shall be defined. Since there is a lack of a common definition a few definitions are cited. The first definition derives from the composition of the word multimedia:
**Multi** means many; much; multiple; numerous

**Medium** An intervening substance through which something is transmitted or carried on *(Ame)* 1991.

This definition is not very exact and contains no computer specific meanings. In the sense of this definition each system with more than one medium is a multimedia system. However, this means each system which uses text and graphics is a multimedia system. Therefore, a second more precise definition shall be given:

A multimedia system is characterized by computer-controlled, integrated production, manipulation, presentation, storage and communication of independent information, which is encoded at least through a continuous (time-dependent) and a discrete (time-independent) medium *(Steinmetz & Nahrstedt)* 1995, p. 17.

This definition narrows the functionality to five basic tasks which have to be integrated by computers. The *creation*, *processing* and *storage* tasks are fundamental functions of any computer system whereas *presentation* is only necessary for user interaction. The *communication* aspect is determined by the high interconnection of todays computers. Through the use of computer independent information the representation can be chosen freely and processing is independent of other media. The additional requirement of a distinction between continuous and discrete medium is the most important point. Consequently, the media kind is the *really* new thing in the whole multimedia boom *(Steinmetz & Nahrstedt)* 1995. Therefore, the following lazy definition is possible:

Multimedia means, from the user’s perspective, that computer information can be represented through audio and/or video, in addition to text, image, graphics and animation *(Steinmetz & Nahrstedt)* 1995, p. 1).

Nevertheless, it was realized that it is not enough to handle the multimedia concept in one part, like processing, representation and communication, without observing all parts together. Hereby, the cooperation of all participants is necessary. Continuous data is the new problem which imposes *time requirements* to the underlying infrastructure. Most approaches divide the system into a continuous and non-continuous part. To process multimedia data additional, specialized hardware is mostly necessary which leads to the loss of flexibility.

A second problem is the different *representation* of multimedia data which consists of analog and digital parts. The analog representation was the main technology for audio and video and, therefore, new integrating approaches between analog and digital representation are needed. The aim is to use digital systems in all involved parts, starting with the production and ending with the consumption of multimedia data. However, until this aim is not reached the use of hybrid consumption systems is necessary.

Another problem is the synchronization aspect between media which shall be used together (audio and video). This is not detailed here.

This all leads to new requirements for computer systems whereas the *real-time* constraint is the main point for all new requirements *(Steinmetz & Nahrstedt)* 1995,*(Blair & Stefani)* 1998. This can be seen in the sections 2.1.2 and 2.1.3.
2.1. Multimedia

2.1.2 End systems

The development of more powerful computers in the low cost range made the use of multimedia for these systems possible. However, how does it differ from a normal computer? Well, the main hardware differences lie in processing power especially through special multimedia equipment, storage capacity and storage format. The software differences lie in support of special hardware, time requirements and resource reservation. These differences are shown now, separated into hardware and software aspects.

**Hardware** The central unit consisting of Central Processing Unit (CPU) and Central Memory (CM) is the main part of the computer. The high amount of memory and CPU power is essential for processing without delay which is necessary to meet time requirements. Multimedia data require a lot of storage capacity, therefore, large hard disk are necessary to support this fast access. Compact Disks (CD) and Digital Audio Tapes (DAT) are used for longer storage and transportation. To get the multimedia data into the computer new multimedia peripherals are necessary. This includes audio and video digitizers for microphones, video recorder, and video cameras which work with analog signals. If no processing of audio and video data is necessary, the data have to be transferred directly to the corresponding output device. The temporary storage in the central memory, which is the normal procedure, is inefficient and leads to time requirements violation. Only if the data have to be changed, the memory is used. However, the high video data amount makes the use of compression techniques necessary (see table 2.1 and 2.2). Therefore, the support by additional hardware is necessary to meet the time requirements. Furthermore, the memory bus and peripheral bus are a bottleneck in each computer system, but in multimedia systems it is crucial for the overall performance. The bus systems transports data between different devices. Figure 2.1 shows an overview of a multimedia system (Fluckiger 1995, p. 54).

**Software** The hardware changes and the use of continuous media data with its special time critical requirements forces changes in the operating system. Operating systems, databases and communication systems are the middleware between application and devices. Furthermore, the applications themselves have to consider the following aspects.

Due to the time requirements *resource reservation* has to be introduced. This requires the reservation of all involved resources that deal with continuous data. Therefore, process, memory and device management is necessary and new file system structures, too.

*Resource management* is required for the manipulation and communication of data streams over the network. This means the reservation of all required resources on the whole network data path. Therefore, resource management has to deal with all parts of the data stream which involves the communication network and the used computers. Quality of Service (QoS) parameters are introduced to manage network resources which are explained in section 2.1.4.

*Process management* has the aim to fulfill the real-time processing requirements on a computer. Therefore, special scheduling algorithms are used which have to consider
time restrictions of multimedia systems. Furthermore, interprocess communication and synchronization have to be changed to provide relations among different media. To use audio and video together both media types have to be processed continuously and alternately.

Memory management has to provide efficient access to the data for manipulation and has to fulfill the delay requirements, therefore, copying data should be avoided and buffers should be used.

Device management is necessary through the use of additional multimedia devices and shared use of devices. For example, network access is essential to transmit continuous video streams to another end system and therefore requires access time guarantees.

File systems (Databases) contain code and data for the whole system. This information has a longer lifetime than processes and is therefore very important. To satisfy the time constraints and to handle the big size of multimedia data multimedia file systems require new file organization and structures (Steinmetz & Nahrstedt 1995).

### 2.1.3 Communication

Due to the use of distributed systems for multimedia applications, the communication aspect, especially the network, has to be considered.

The success of the Internet is visible in the widespread use and interconnection of networks which let people communicate with each other all over the world. Nowadays, it is usual to have computers with network access and the capability to process audio and video. This makes it easy to overcome time and space and to use multimedia for this. However, to follow the high demand of data transmission capacity
the network infrastructure will need to improve. The improvements in speed and quality of networks make the widespread use of large quantities of data between multiple points possible. Thereby, the transmission of multimedia data is practicable but demands restrictive requirements to the network. Multimedia applications for Computer-Supported Cooperative Work (CSCW), for example, require high data rates and lightweight transport protocols. The data stream characteristic of multimedia streams will be detailed on page 2.1.3.

To support multimedia communication the gap between a generic, broadband and service driven network model, like the ISO/OSI Reference Model, and a specific, narrowband and application-driven network model, like the Internet model, has to be closed. This is necessary to bring public and private networks together. As example, public networks of telecommunication companies work with the ISO/OSI Reference Model whereas private networks of companies work with the Internet model. Communication without a standard is not possible and therefore the standards have to grow together or bridges between standards must be defined (Rose 1996).

The base of the success of multimedia depends of the digital representation of the content. The digital representation has three big advantages:

- **Data Replication**
  It is easier to create copies of digitally stored data.

- **Integration**
  The use of different data types together is easier.

- **Bandwidth**
  Digital data requires less bandwidth than the same data in analog form.

This are the economic reasons for the use of digital data in industry. For example, the use of digital transmission for satellite communication permits the use of 5 to 10 more television programs as with analog transmission if the MPEG-2 compression algorithm is used. The MPEG video compression standard is explained in section 2.1.5 and details about other media types can be seen in Steinmetz & Nahrstedt 1995, Chiariglione 1996).

The user desires interoperability from end to end. If this is not satisfied by standards and products the user will not buy it. Therefore, digital video and audio is not synonymous with multimedia. It is more necessary to build a multimedia system (Chiariglione 1996). As example, the time requirements in connection with the amount of data and new applications, like CSCW, with high interactions lead to new requirements.

The communication aspect can be divided into the transport and application aspect. Transport concerns the direct handling of the network access whereas application concerns the management of the data streams. The application aspect is not described here and the transport aspect is explained in the next paragraph.
**Data Stream Characteristics**

Multimedia data streams have many common and special requirements to the underlying networking infrastructure. Therefore, only special multimedia requirements are discussed below (Steinmetz & Nahrstedt 1995, p. 401-402):

- **Data Throughput**
  Multimedia streams require high data throughput, especially if several streams co-exist, even if the data is compressed. Further, the required real-time data processing might lead to a bottleneck on a local end system.

- **Fast Data Forwarding**
  Fast data forwarding from the network layer to the application avoids high buffers. This supports better use of resources and reduces data delays.

- **Service Guarantees**
  To guarantee a service is absolutely fundamental for the success of distributed multimedia applications. If a service cannot be guaranteed the user would not use it. Therefore, the use of QoS guarantees is necessary (see section 2.1.4).

- **Multicasting**
  Support of multicast is necessary to reduce resource consumption and to provide an easy communication mechanism for the maintenance of groups.

The combination of inconsistent requirements make multimedia transmissions difficult. For example, high data volume is in contradiction to real-time. This can be summarized to the following sentence:

**Multimedia streams require distributed real-time transmission of continuous, high data volume.**

To satisfy these requirements the following four quantitative performance criteria are important:

- **Throughput**
  Throughput is the exchangeable data rate between to end systems and, therefore, is a basic criteria for transmission success.

- **Latency**
  Latency is the transit delay of packets over the network. It requires time to transmit from sender to receiver. This is important for synchronous applications.

- **Jitter**
  Jitter is the delay variation which says something about the range of the latency.

- **Error rate**
  The error rate is an indicator about the integrity of the received data. Data can be changed by alteration, loss, duplicate or out of order delivery.
The requirements comprise changes on each layer of the ISO/OSI Reference Model ([Halsall 1996](#)). This includes resource reservation and guarantees to comply the timing constraints.

A *multimedia networking system* allows for the data exchange of discrete continuous media among computers. This communication requires proper services and protocols for data transmission ([Steinmetz & Nahrstedt 1995](#) p. 313).

Therefore, QoS parameters (see section 2.1.4) are defined.

### Table 2.1: Bandwidth requirements of different media types

<table>
<thead>
<tr>
<th>Media</th>
<th>Transaction Type</th>
<th>Format</th>
<th>Sampling dimensions</th>
<th>Uncompressed bit rate</th>
<th>Compressed max. bit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech and music</td>
<td>Telephony</td>
<td>PCM</td>
<td>8000 sample/s × 8 bit/sample</td>
<td>64 kbit/s</td>
<td>8 – 32 kbit/s</td>
</tr>
<tr>
<td></td>
<td>CD-audio</td>
<td></td>
<td>44100 sample/s × 16 bit/sample</td>
<td>70.56 kbit/s</td>
<td>128 kbit/s</td>
</tr>
<tr>
<td>Business video</td>
<td>Videophone</td>
<td>MPEG-4</td>
<td>176 pixel × 144 line × 12 bit × 10 frame/s</td>
<td>3.04 Mbit/s</td>
<td>64 kbit/s</td>
</tr>
<tr>
<td></td>
<td>Video conferencing</td>
<td>MPEG-1</td>
<td>352 pixel × 288 line × 12 bit × 25 frame/s</td>
<td>30.4 Mbit/s</td>
<td>1.15 – 2 Mbit/s</td>
</tr>
<tr>
<td>Entertainment video</td>
<td>Broadcast television</td>
<td>MPEG-2</td>
<td>720 pixel × 576 line × 12 bit × 25 frame/s</td>
<td>124.4 Mbit/s</td>
<td>15 Mbit/s</td>
</tr>
<tr>
<td></td>
<td>High quality televisions</td>
<td>HDTV</td>
<td>1920 pixel × 1080 line × 16 bit × 30 frame/s</td>
<td>994.3 Mbit/s</td>
<td>135 Mbit/s</td>
</tr>
<tr>
<td></td>
<td>High quality televisions</td>
<td>MPEG-3</td>
<td>1920 pixel × 1080 line × 12 bit × 30 frame/s</td>
<td>745.8 Mbit/s</td>
<td>20 – 40 Mbit/s</td>
</tr>
</tbody>
</table>

### Table 2.2: Space requirements of images

<table>
<thead>
<tr>
<th>Media</th>
<th>Transaction Type</th>
<th>Format</th>
<th>Sampling dimensions</th>
<th>Uncompressed bit rate</th>
<th>Compressed max. bit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td>Normal resolution</td>
<td>JPEG</td>
<td>720 pixel × 576 line × 16 bit/pixel</td>
<td>26.636 Mbit</td>
<td>104 k – 830 kbit</td>
</tr>
<tr>
<td></td>
<td>Very high resolution</td>
<td>JPEG</td>
<td>1280 pixel × 1024 line × 24 bit/sample</td>
<td>31.46 Mbit</td>
<td>300 k – 3 Mbit</td>
</tr>
</tbody>
</table>

Because of the high amount of data (see table 2.1 and 2.2) for graphic, video, audio, etc., representation, data compression techniques are used to reduce the amount for storage and transmission ([Halsall 1996](#) p. 561). Less bandwidth consumption is a fundamental aspect and is one of the first requirements to make a satisfactory transmission possible ([Hunter, Witana & Antoniades 1997](#)). As example, the video compression technique MPEG is explained in detail in section 2.1.5.

### 2.1.4 Quality of Service (QoS)

As explained in the sections before multimedia requires real-time and safety critical services and applications. QoS is introduced due to manage this environment and
to give guaranteed services. QoS mirrors the special characteristics, like physical properties, used protocols, current load, etc., of the underlying system in a well defined manner. Therefore, the use of QoS for multimedia applications is essential to fulfill the requirements and furthermore the acceptability (Wolf 1998).

QoS encompasses different aspects like static and dynamic parameters and is used on different levels of the system. Therefore, the layers have to work together. Furthermore, especially the difference between network, transport, operating system and application QoS is important. The use of distributed systems require the cooperation of end systems and the network to provide an end-to-end QoS. The customer is not interested in internal details but in the performance he gets at the end (Blair & Stefani 1998).

Distributed environments use a set of functions to express a service which are exported by the system and accessible to its clients. However, what happens with the non-functional properties? The non-functional properties have also to be considered and therefore QoS was introduced. QoS provides the parameterization of communication aspects into common parameters which can be used for the communication between layers, especially between transport and application layer.

To give an overview the ISO/IEC QoS Framework is introduced. Afterwards, some multimedia aspects shall be shown.

Concepts of the QoS Framework

ISO/IEC 13236 have defined a QoS Framework ([ISO 1995b]) to provide concepts and terminology for further developments in the QoS area. Furthermore, it gives an overview about relevant areas of QoS. The framework defines QoS as

A set of qualities related to the collective behaviour of one or more objects ([ISO 1995b]).

The framework defines several parts which can be divided into QoS concepts and modeling, Information and Management function. To give an overview the parts are described together and not separately whereas the focus is on the concept for use.

![Figure 2.2: Relationship between QoS concepts](image-url)
To introduce the concept (see figure 2.2) the base for QoS management has to be explained. QoS management requires the definition of QoS characteristics as fundamental information bases. They are an identifiable and a quantifiable aspect of the QoS of a system, service or resource. QoS characteristics represent an actual behaviour independent of their interpretation or use.

Furthermore, QoS requirements are necessary to specify what is required by the user. QoS requirements can be described by QoS parameters and a QoS context. QoS parameters are dynamic parameters for the current use whereas a QoS context contains static information. These information may be of different kinds, including

- a desired level of characteristic — e.g. a target of some kind;
- a maximum or minimum level of a characteristic — e.g. a bound;
- a measured value, used to convey historical information;
- a threshold level;
- a warning or signal to take corrective action;
- a request for operations on managed objects\(^1\) relating to QoS, or the results of such operations [ISO 1995b].

An example for QoS parameters in Asynchronous Transfer Mode (ATM) is given in [Chen, Liu, Procanik, Wang & Casey 1998].

QoS requirements can be expressed in deterministic or stochastic terms. Furthermore, the two extremes guaranteed and best effort service can be used. QoS categories are used to divide requirements into discrete groups. This can be used for distinction of different kinds of applications with their special requirements. This is explained for multimedia below.

QoS management functions use this information to administrate and control the system. They use QoS mechanisms to perform their actions, as an example the reservation of resources, to meet QoS requirements. It can be divided into static and dynamic functions. Static functions are specification, negotiation, admission control and resource reservation and are carried out during connection establishment. Dynamic functions are monitoring, policing, maintenance and renegotiation and are used during run time [ISO 1995b, Blair & Stefani 1998, Sluman, Tucker & Wood 1997, Hutchison, Coulson, Campbell & Blair 1994].

**Multimedia specific QoS**

Distributed multimedia systems especially require QoS for three parts (see page 10) which are explained below.

- **Timeliness**
  Timeliness concerns end-to-end delays of either continuous or discrete interactions. Therefore, the values of latency and jitter are important.

\(^1\)Representation of any resource in an Object oriented way
• **Volume**
  The volume refers to the throughput which is essential for multimedia communication.

• **Reliability**
  Reliability for multimedia interactions considers terms like bit error rate and loss of data.

Other categories, like cost, criticality\(^2\) and quality of perception, are important, too, but not further discussed here. Examples for some media types and corresponding QoS requirements are given in table 2.3 (Hehmann, Salomy & Stuttgen 1990).

### Table 2.3: Typical quality of service requirements

<table>
<thead>
<tr>
<th>QoS</th>
<th>Maximum latency (s)</th>
<th>Maximum jitter (ms)</th>
<th>Average throughput (Mbit/s)</th>
<th>Acceptable bit error rate</th>
<th>Acceptable packet error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice</td>
<td>0.25</td>
<td>10</td>
<td>0.064</td>
<td>&lt; 10(^{-4})</td>
<td>&lt; 10(^{-4})</td>
</tr>
<tr>
<td>Video (TV quality)</td>
<td>0.25</td>
<td>100</td>
<td>100</td>
<td>10(^{-2})</td>
<td>10(^{-3})</td>
</tr>
<tr>
<td>Compressed video</td>
<td>0.25</td>
<td>100</td>
<td>2 – 10</td>
<td>10(^{-9})</td>
<td>10(^{-9})</td>
</tr>
<tr>
<td>Data (file transfer)</td>
<td>1</td>
<td>–</td>
<td>2 – 10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Real-time data</td>
<td>0.001 – 1</td>
<td>–</td>
<td>&lt; 10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Image</td>
<td>1</td>
<td>–</td>
<td>2 – 10</td>
<td>10(^{-4})</td>
<td>10(^{-9})</td>
</tr>
</tbody>
</table>

As example for compressed video the Moving Picture Experts Group (MPEG) standard will be described in the next section, because it is relevant for the prototype implementation described in chapter 5.

### 2.1.5 Moving Picture Experts Group (MPEG)

The Moving Picture Experts Group (MPEG) was founded in 1988 and they currently work under the name *ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio*. They elaborated several standards for video and audio compression. MPEG-1 defines video and audio compression for medium quality and medium bit rate. With MPEG-1 the current CD-ROM players technique should be usable for video and audio storing in VHS-quality. Therefore, the given data transfer rate of 1.5 Mbit/s of CD-ROM players was used. This allowed commercial use of video and audio. Due to the low data rate MPEG-1 can be used for transmission over wired computer networks, too. The MPEG-1 specification (ISO 1995a) consists of the five parts system, video, audio, conformance testing and software simulation. The first three parts reached *International Standard* status in early 1993 and part 4 in 1994. Part 5 is a technical report about an implementation of the parts 1-3 whereat the source code is not publicly available.

Furthermore, MPEG-2 was defined achieving high-quality video playback at higher data rates for the professional use (digital television). The parts system, video and

\(^2\)“relates to the assignment of relative priority levels between activities” (Blair & Stefani 1998, p. 69)

MPEG-3, initially thought for High Definition TeleVision (HDTV), was dropped. During definition of MPEG-2 it was realized that with slightly higher scaling it could be used for HDTV, too.

In September 1993 the work at MPEG-4 started which initially concerns very low bit rate coding of audio-visual programs and shall reach the status *International Standard* by January 1999. To achieve this new methods have to be developed. The aim of MPEG-4 was the use of the European Mobile Telephony System (Global System for Mobile telecommunications GSM), for example. MPEG-4 concerns the representation of audio and video data as objects to provide the integration of the production, distribution and content access paradigms of the fields digital television, interactive graphics applications and the World Wide Web. Therefore, MPEG-4 video is optimized for low (< 64kbit/s), intermediate (64 – 384kbit/s), and high (384kbit/s – 4Mbit/s) bit rates.

It can thus be expected that MPEG-4 will become the enabling technology for multimedia communications as much as MPEG-1 has become the enabling technology for digital video and MPEG-2 of digital television (Chiariglione [1996]).

The further text covers the video part of MPEG-1 only because the applications of the implemented prototype (see section 5.3) use MPEG-1 video. To explain the MPEG-1 video part the requirements are described first. Secondly, the chosen methods are explained. Thirdly, the structure of the bit stream is described.

**Requirements**

The video compression algorithm of the MPEG standard had the following features to comprise:

**Random Access** Random access means that each frame[^3] in the compressed video bit stream is decodeable in a limited amount of time. Therefore, access points are necessary.

**Fast Forward/Reverse Search** Scanning a bit stream is an important feature for video. Therefore, it should be possible to use random access points to show only selected pictures and not all pictures.

**Reverse Playback** Reverse playback is necessary for interactive applications and should not lead to high additional cost.

**Audio-Visual Synchronization** Audio and video signals have to be synchronizable to use them together.

[^3]: A frame is the lowest MPEG structure for a whole picture
**Robustness to Errors** An errorless transmission cannot be assumed and therefore the standard has to deal with the appearance of errors. A transmission error should not lead to fatal behaviour.

**Coding/Decoding Delay** The MPEG applications are mostly interactively used and this leads to time constraints which have to be met.

**Editability** Insertion of independent correction units should be possible.

**Format Flexibility** For using videos in window-based operating systems, the presentation format, like size, shall be flexible.

**Cost Tradeoffs** An MPEG real time decoder must be implemented with a small amount of chips and the technology of 1990.

**Video Compression Methods**

MPEG-1 was defined after the ISO-standard JPEG [ISO 1997] and CCITT-recommendation H.261 [CCITT 1990] had been established. JPEG allows the compression of still images and H.261 is a standard for visual telephony. There was the attempt to use JPEG also for motion compression named Motion-JPEG (M-JPEG). The idea of M-JPEG was to compose the video of a sequence of still images. However, the compression of still images is not good enough and audio data is necessary, too. The problems of H.261 were the inflexibility through the restrictive specification. Hence, MPEG-1 especially combines the idea of M-JPEG and the motion compensation of H.261. Therefore, MPEG-1 uses JPEG for still images, whereas all compression parameters are available, and tried to be compatible with H.261. Additionally, MPEG defined new features for more flexibility. Hereby, MPEG is a *generic* standard. This means that not all features have to be used at all time for each application. Furthermore, only the syntax to create an MPEG stream and the decoding process, not the encoder and decoder itself, are part of the standard. Thus, three frame types (I-, P- and B-Frames) (see page 17) and a layered structure for the bit stream (see page 17) are defined. MPEG allows symmetric as well as asymmetric compression methods. Asymmetric compression requires more time for encoding and less time for decoding [BMR 1998, Gall 1991, Schroiff 1995].

**Motion Compensation** The motion compensation is the central point of the video compression algorithm and makes use of the fact that many times successive images are similar. Mostly, the image areas are shifted to another place and do not change at all. Therefore, only the difference has to be encoded for reducing the temporal redundancy.

Mostly, an object inside an image moves from one point to another and therefore only the new location of the object has to be encoded. Therefore, a vector encodes the movement from the old to the new position. Due to the difficulty to determine a moving object MPEG defines macroblocks (see page 17). A macroblock consists of 16x16 pixels and corresponds to four JPEG blocks. The macroblocks are compressed with JPEG methods. The moved macroblocks have not always to be identical. It is conceivable that some changes inside the macroblock have occurred. Hereby, the
changes have to be encoded, too. How to compute the vectors and macroblocks is not part of the standard and consequently the quality depends of the used encoder \cite{Schroiff1995}.

**Frame Encoding** To use the idea of motion compensation and to compress still images the MPEG standard distinguishes three types of images:

- **I-frames** *(Intra-coded images)* are still images with no references to other images and compressed with JPEG. Since the need for real time compression I-frames have the lowest compression rate inside MPEG. Through the independence of other images I-frames are the entry points for random access in MPEG streams and a reference point for P- and B-Frames.

- **P-frames** *(Predictive-coded frames)* contain the motion compensation information as described above. Therefore, information of the previous I-frame and/or all previous P-frames is required.

- **B-frames** *(Bi-directionally predictive-coded frames)* are encoded relative to the past and/or to the following P- or I-frame. B-Frames use a similar encoding as P-Frames except that the motion vector references can refer to the following frame, too. This leads to the highest compression rates inside MPEG and makes it impossible to use B-frames for random access.

Only I- and P-Frames are used as reference points for motion vectors. Hereby, B-Frames cannot propagate errors. The encoder must consider the necessary data buffer for decoding. Hence, it is not possible to use any order and number of frames. However, it is not necessary to use a static pattern (always the same order and number) for the use of frame types. For a real time performance, 30 frames per second are necessary \cite{BMR1998}.

**Video Stream**

The above mentioned methods and frame types for compression are mapped into a layered structure to build the bit stream.

The goal of a layered structure is to separate entities in the bit stream that are logically distinct, prevent ambiguity and facilitate the decoding process. The separation in layers supports the claims of *genericity, flexibility* and *efficiency* \cite[p. 55-56]{Gall1991}.

The layered structure of syntax (see figure 2.3) for the MPEG video bit stream consists of the following six layers \cite{Schroiff1995, Gall1991}:

**Sequence Layer** The sequence layer contains context information and especially information for the data buffer handling (width, height, format, refresh rate, data rate, buffer size). Thereby, the storage capacity and constant bit rate is used to determine the decoding time to check the delay. By using this layer the sequences are independent of each other and different parameters can be used. Therefore, the decoder initializes itself each time a sequence layer starts and it can be used as random access point.
Group of Pictures Layer (GOP) The GOP layer consists of several different frames containing at least one I-frame which has to be at the first position. GOPs can be used as random access points.

Picture Layer The picture layer contains the information for a whole frame. The header consists of the image number inside the GOP, the frame type and information about the used motion vectors.

Slice Layer Slices are a collection of macroblocks. They can be decoded without access to other slices. This avoids the propagation of errors. The slice layer contains several macroblocks.

Macroblock Layer A macroblock consists of six blocks. It contains the location of the macroblock inside the image and the macroblock type (I, B or P).

Block Layer Blocks are the smallest unit in the video stream and consist of $8 \times 8$ pixels. It uses the same compression methods as JPEG.

One important point is that the stream must be decodable with a buffer of an appropriate size. Therefore, some additional constraints are necessary for building the stream. MPEG specifies the data stream requirements which is divided into single packets. If communication networks are used, the packets always contain the header information first. For synchronization, time stamps are defined. Some information, like the maximal data rate, is transferred only at setup of the data stream (Steinmetz & Nahrstedt 1995). This can lead to problems.

In a conferencing application, using an MPEG stream might be inconvenient because a new user might like to join an existing conference after the data streams have already been setup. Therefore, the necessary header information would not be directly available to her/him (Steinmetz & Nahrstedt 1995 p. 159).
As we could see above the MPEG data stream consists of sequences. The sequence packet which is transmitted at the beginning of a sequence contains all context data to interpret the following data correctly. If a new user wishes to join an existing stream it is not guaranteed that the user gets a sequence packet first. Therefore, the user must wait until a new sequence arrives or he gets the sequence packet information in advance. This problem also occurs when trying to reroute the data stream to a new client and is discussed further in chapter 5.

This section introduced multimedia and the difference between non multimedia and multimedia end and communication systems was given. This showed the requirement for QoS support for multimedia systems which was then described. MPEG was also introduced as example for a video and audio compression standard which is used in the implemented prototype (see section 5.3).

The next section introduces the mobility aspect of mobile end systems.

2.2 Mobility

Due to the wide deployment of fixed network for voice and data communication in the 80’s, the next logical step was mobile communication. The mobile trend is a solution to our needs for individuality and unlimited communication. Therefore, the ability to communicate anywhere and anytime in personalized terms and conditions with access to a wide range of services shall be possible. The services contain the whole range of multimedia data to be useful for the users.

Another reason for the increased popularity of mobile communication is the fact that communication is the base of today's industry and therefore essential for the success of a country. Wireless (mobile) infrastructure is cheaper than wired and therefore suitable for poor countries to support their own industry.

To introduce the mobile and wireless aspect some basic concepts are detailed first. Secondly, end-system characteristics for mobile environments are illustrated. Thirdly, the communication aspect is described.

2.2.1 Basic concepts

What is meant by mobility? In order to explain this, a differentiation between mobile and personal communication is necessary. Mobility and personal communication are close together because of the demand of individualized services. Mobility is required for location independence and personal communication for the individual use of it. Furthermore, mobility is not a synonym for wireless communication. The table shows examples for the different aspects of wireless versus mobile [Dorner & Schiller 1998]. Mobility means terminal mobility, like mobile phone, which can be described as follow:

The ability of a terminal to access telecommunications services from different locations and while in motion, and the capability of the network to identify and locate that terminal or the associated user [Rose 1996, chap. 2].
Table 2.4: Wireless versus Mobile

<table>
<thead>
<tr>
<th>Wireless</th>
<th>Mobile</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>desktop computer</td>
</tr>
<tr>
<td>×</td>
<td>√</td>
<td>notebook in hotel</td>
</tr>
<tr>
<td>√</td>
<td>×</td>
<td>radio LAN in not wired house</td>
</tr>
</tbody>
</table>
| √        | √      | mobile telephone (see section [2.2.2])

Therefore, telecommunication services have to be available in all areas and at all time, ideally. Besides, the complete dissociation of terminals from a specific customer is necessary. For example, today’s mobile phones are linked to the telephone number and the owner and not to the actually user of it. Personal mobility is the second part of mobility, which can be described as follow:

The ability of a user to access telecommunication services at any terminal on the basis of a personal telecommunication identifier, and the capability of the network to provide those services according to the user’s service profile (Rose 1996, chap. 2).

This includes problems like addressing, routing and charging of the users’ calls. Besides, service profile and user interface portability shall be possible. If both together are established, real mobility is reached. For this, Universal Personal Telecommunications (UPT) is defined to roam between terminals in fixed and mobile networks. Universal Mobile Telecommunications System (UMTS) defined by European Telecommunications Standards Institute (ETSI) is the next step which has the following objective:

UMTS is intended to provide personalized, globally accessible high quality mobile communication services [...] (Rose 1996)

To realize UMTS a new infrastructure is necessary wherefore the information technology, broadcast entertainment and public/private fixed/mobile telecommunication networks will grow together. They work together to get the advantages of digital communication and to make the project economically possible. The new infrastructure shall work as one global network whereas it internally consists of many networks with different types and characteristics.

Hereby, among other things the use of mobile and fixed networks together leads to new problems. Mobile networks require new transport protocols to match the new constraints, for example. The use of fixed and mobile networks in voice communication is realized. However, this was only possible due to the special characteristics of voice communication. Voice communication only requires low and constant data rate whereas video data requires high and flexible data rates (Rose 1996).

The availability of wireless techniques for Wide Area Networks (WANs) allowed the use for mobile data services. One of the well known standards for digital cellular services in WANs is *Global System for Mobile telecommunications* (GSM) which is introduced on page 28. After the successful introduction in WANs, wireless technique
was introduced in Metropolitan Area Networks (MANs) and Local Area Networks (LANs), too. If a seamless connectivity through handovers between different networks is achieved when the users roam, the whole potential of wireless networks is visible.

To understand the new problems through mobility the enabling technologies for mobile computing are introduced. They are separated into end-system and communication technology.

Firstly, the characteristics of end-systems for mobility are explained. Secondly, the use of wireless communication technology for the exchange of information and the use of remote resources is described.

### 2.2.2 End-systems

To provide mobile end-systems new requirements must be fulfilled. The technology and concepts for desktop computers are not adequate for the use in a mobile environment. For example, to be mobile the end-system has to be carried by the user which leads to limitation in size and weight.

Therefore, different kinds of end-systems are shown first. After this, their special requirements will be discussed.

The following enumeration gives examples for mobile end-systems whereas the sequence mirrors the system’s performance [Dorner & Schiller 1998]:

- **Pager** A pager receives simple text messages and displays it on a small display.

- **Mobile Telephone** Mobile telephones send and receive voice data. A small display for text messages is available, too.

- **Personal Digital Assistant (PDA)** PDAs have a graphical display which allows the use of pens to make inputs. The software implements hand-writing recognition and sometimes simple access to the World Wide Web (WWW).

- **Palmtops** Palmtops consist of a small keyboard and display and are able to process simple applications, like diaries. They are a smaller version of laptops but with the size of PDAs and, therefore, they compete with PDAs.

- **Laptop** A laptop provides equivalent functionality and performance as a desktop computer but is portable and, therefore, limited in his size. The usual size is of standard A4 paper and laptops provide a standard keyboard and usually a trackball for user input [Mitchell 1997].

The techniques for pager and mobile telephones is simple and they only provide a small capacity for mobile computing. Therefore, we can divide the end-systems into three classes, simple end-systems (pager and mobile telephones), advanced end-systems (PDAs and palmtops) and portable computers (laptops). However, the development of a palmtop with integrated mobile phone shows that end-system functionality of all three classes converge. The premise for portable computers is the compatibility to desktop computers whereas PDAs have to support a specific set
of organizational tools which are easy to use. To be accepted by the users the PDAs must be convenient and as reliable as existing tools wherefore they have to be very portable, rugged and exhibit long battery life [Friday 1996]. PDAs and portable computers differ in their capability to provide applications and in the programming environment. PDAs provide an architecture independent programming environment whereas portable computers are compatible with their desktop counterparts. Besides, the compatibility to desktop computers requires the provision of the same or equivalent input/output devices, like a trackball instead of a mouse for the use of windowed environments.

Through limitations in size and portability end-systems require specialized hardware elements which make portable end-systems more expensive. These specialized elements are explained below are not necessarily important for each mobile end-system. The chosen subset for an end-system depends on the environment constraints.

- **CPU** The heart of each computer system is the Central Processing Unit (CPU). The rapid deployment of fast processors and the integration of many components, like cache and coprocessor, in one chip made the use for mobile end-systems possible. Nevertheless, due to the limited amount of power and space it is necessary to use specialized versions with power save support and hence, sometimes limited performance.

- **Memory** The access to high memory capacity is essential for each computer system, especially for multimedia. Hence, mobile end-systems also require high memory capacity with a small size and low power consumption.

- **Hard Drive** The use of hard drives with high storing capacity is limited by the lack of power. Due to the constant motion the use of maneuverable elements is restricted.

- **Flash Disk and Memory Card** The basic configuration of an end-system does not always contain the desired disk and memory capacity due to the limited size and weight. Therefore, the additional use of memory and disk cards, e.g. via PCMCIA, should be possible to fulfill the users special requirements.

- **Smart Cards** Smart cards are application specific cards which provide user specific information, for example, authentication or account information. Hereby, personalized data for authentication and accounting can be used.

- **Display** The display is a very important factor for the new environment. Because of the limited size, special format and the outdoor use it is necessary to use specific display technology. Pen-sensitive displays are essential since they do not require a keyboard and mouse. Mostly, the display quality is not as good as with standard displays. Nevertheless, the display of video should not be restricted by the new technology.

- **User Interface** The user interface is limited by size. If a keyboard is provided its size is a compromise between finger size and the ability to carry it.

Power management is absolutely essential for mobile end-systems. The lack of continuous power access and the restrictions of battery technique necessitate energy
management in software and hardware. The loss of data due to lack of power or damages because of mishandling is always present when using mobile end-systems.

The application user interfaces have to be changed considering input (keyboard, mouse) and output (display). Many GUIs do not work well with small displays and the input via keyboard is not always possible. Besides, the use of pen-sensitive displays has to be considered leading to recognition of of hand written text, too.

Due to mobile environments the capability to add wireless communication technology, for example as a PC card, to the end-system becomes important. To get access to the mobile network the mobile end-system especially requires a small antenna with low power consumption, in order to save energy. Besides, studies have shown that the use of radio technique close to the head is not harmless \textsuperscript{[Rose 1996, Dorner \& Schiller 1998, Friday 1996]}. The wireless communication aspect is explained in detail below.

### 2.2.3 Communication

In order to permit mobile communication new technology for data transmission has to be developed. To support mobility within distributed computing the use of \textit{wireless} technologies must be considered. Existing radio communication technology was mainly used for analog voice communication and not for digital data communication, and therefore, new techniques had to be developed.

Networks are classified into several groups which differ in the physical area of coverage. A Local Area Network (LAN) covers a campus or company area, a Metropolitan Area Network (MAN) covers a city and a Wide Area Network (WAN) covers a country, for example. Another distinction is the ownership, LANs usually are private housed and maintained and WANs are public. The concepts for wireless LANs (WLAN) and wireless WANs (WWAN) are fundamentally the same. The differences lie in the special requirements and environments. WLANs are used to cut cost in laying of cables and allow easier mobility of a computer inside a building. Besides, the quick connection of visitors mobile is also advantageous \textsuperscript{[Davies, Mitchell \& Cheverst 1998]}. WLANs offer a relative good quality of service in comparison to WWANs because of the low range between sender and receiver. However, the use close to electrical equipment lead to higher disturbance as in WWANs. WWANs are mainly used to provide communication services for moving end-systems and to cover large areas. Commercial use of WWAN must consider accounting, security and availability items which have influenced the system design.

The base station antenna location for WWANs is mostly on vantage points such as hilltops and the roofs of buildings. Orbiting satellites are a second approach for the location of antennas.

The commercial use started with the satellite mobile communications which were initially intended to serve large vessels on the high sea. Now, there is nearly no part of communication without satellite communication. The existing satellites are geo-stationary which means they are at a far and fixed point in the sky. Therefore, satellite communication is more affected by the atmosphere and the distance. The
next phase of satellites shall be able to fly lower and therefore they are non-geo-
stationary. Until the end of this millennium the first systems shall be used.

The advantage of non-geo-stationary satellites is better signaling performance due
to the shorter distance, however, the disadvantage is the fact that these satellites
are not fixed on the sky for the user on earth. Therefore, it is not possible to
permanently use the same satellite even if the user on earth does not move.

The use of satellites allows cheap and easy access to the global communication
infrastructure which is especially important for undeveloped areas. Furthermore,
satellites are good for the covering of difficult to reach areas where a normal mobile
infrastructure is not possible to install, which is important especially for the use in
case of emergency. Therefore, the fusion of fixed networks and mobile networks is
essential.

At the moment, only a few services are provided directly using satellites. Satellites
are normally used by providers and not by the users themselves. This will change
in future: satellite personal communication networks (SPCN) and services (SPCS)
will have been installed.

One example is SkyDSL from ALOHA Networks:

SkyDSL is the only two-way satellite system able to economically ex-
tend Internet connectivity to large populations of business end-users.
SkyDSL reshapes the economics of unicast satellite services by com-
bining a patented multiple access protocol called SAMA, a proprietary
TCP/IP proxying and link layer solution, and a specifically tailored hard-
ware implementation. Together these technologies significantly improve
the throughput of carrying bursty Internet access traffic over satellites
(ALO 1998).

Another example is the Iridium satellite system (Iri 1998) which is a step to SPCN:

![Iridium system overview](image)

Figure 2.4: Iridium system overview

The Iridium system is a global wireless communications network that
will combine the worldwide reach of 66 low-earth-orbit satellites with
land-based wireless systems to enable subscribers to communicate using handheld telephones and pagers virtually anywhere in the world (Iri 1998).

Iridium uses land-based telephone systems of roaming partners if customers are inside these local cellular coverage. Otherwise, the Iridium satellite network is used. A new technique of Iridium is the use of crosslinks between satellites. Furthermore, all calls to a user are automatically rerouted to its current location whilst the telephone is switched on. This is a step to the realization of UPT and UMTS as described in section [2.2.1] The commercial service commencing of Iridium is planned for late 1998 but failures and damages of satellites lead to delays for the commissioning of the whole system.

The media characteristic, access and structure of wireless networks are detailed below. To explain the structure the GSM standard was chosen as example.

The special characteristics of the media air and the used transmission techniques influence the quality of service and are therefore important for the user. Besides, the media access methods influence the QoS, too.

**Media characteristic**

The use of wireless media leads to new characteristics. The differences to fixed networks are

- due to interferences, the error rates are 100–1000 times higher,
- the available frequency range for communication is limited which requires frequency control, furthermore most frequencies are actually used,
- the available data rates vary between several Mbit/s and 9.6 kbit/s with GSM (see page [28]),
- the range to the receiver and the power used for transmitting influence the quality,
- connection establishment takes longer (GSM: range of seconds),
- the use of security mechanisms is absolutely important when accessing the public interface air,
- the used media is always shared and therefore requires access mechanism.

Knowledge about signal spreading is necessary to understand the characteristic of the received signals and the corresponding problems. Basically, signals spread in a line. Signal attenuation is the loss of signal intensity through the resistance of the molecules in the air. Attenuation is higher in rainy or foggy conditions which therefore effects QoS. Shading through obstacles makes it impossible to receive signals. Therefore, it requires an efficient placement of the sending station to reach receivers and the receivers must give attention to be inside the scope of a sending station.
A special problem are tunnels which require a sending station inside the tunnel to allow the use of wireless communication. Signals spread in a line as long as they do not hit an obstacle. If a signal hits a big and flat surface it will be reflected at it. Scattering at little obstacles divides the signal into several signals. Therefore, it is possible to receive the same signal at different time intervals, directions and intensity. This also happens if the receiver gets the same signals of two sending stations offering broadcasting services. Diffraction at sharp edges leads to little redirections of the signal. Each time a signal touches an obstacle it loses intensity whereby the amount depends on the signal phase.

While an end-system moves the signal’s way changes and therefore the signal’s phases and delays change, too. This leads to short interruptions called fast fading. If the end-system moves away from a base station or a signal is redirected by an obstacle the signal power changes slowly what is called slow fading. The signal power is reduced by proportion \( \frac{1}{d^2} \) (d = distance between sender and receiver). This leads to the dependency of distance to error rate and throughput. Additionally, the used signaling technique influences the distance limit, too. Therefore, mobility and throughput show an inversely proportional relationship which can be seen in figure 2.5.

During the motion the change to another network with other quality of services is perhaps necessary. An example is the move from a WLAN with a data rate of 2Mb/s to a WWAN with 9.6 kb/s data rate. Therefore, the connectivity varies in throughput, error rates or dial-up delays. Applications are typically aware of their environment and therefore they make assumptions about the resources of the environment. In a mobile environment it is normally not possible to fulfill the same assumptions like in fixed networks. As example, the availability of a remote file system is not as good as in fixed networks. The effects of interrupted connectivity and blackspots\(^4\)4 have to be concerned. Another part of resources is their location. By the mobile host moving the routing of data to and from the host is influenced, too.

**Media access**

The access methods to the media air is determined by the limited frequencies and the characteristic above described. Therefore, efficient use of available frequency is

\[^4\text{blackspot} = \text{area with no access to a wireless network}\]
paramount essential, so techniques such as individual scheme and the open channel scheme are introduced.

**Individual channel** The provided bandwidth is divided into separate radio spectrums which are called channels. A channel consists of time or access patterns (time, frequency or code division multiplexing), which permits the use of several mobiles with one base station and is called trunking. To decide which mobile is associated with which channel, a protocol is required. Therefore, the protocol divides the channels into a control channel and data channels. A mobile has to use the control channel to request a data channel.

**Open channel** The open channel scheme uses shared frequencies or channels whereas each mobile uses the same open channel. Therefore, a mobile may listen to any transmission within range. A problem is the signal attenuation which prefers the strongest, mostly nearest, transmitter. Besides, it is possible that a mobile cannot hear another mobile because of signal attenuation. To control channel access a social protocol is required to allow each mobile the access. The concept is the same as in ALOHA (University of Hawaii) or Carrier Sense Multiple Access (CSMA).

The cost factor of using data network resources was not important due to the fact that most users had free access to it. However, this changes with commercial network providers which charge network resources, connection establishment, etc.

An important factor of the usability of end-systems is their power consumption. This has to be considered by the communication part of the end-system because each transmitted piece of data reduces battery life. Therefore, protocols have to be optimized and applications must carefully decide which data to send or receive.

**Structure of a wireless network**

The structure of a wireless network defines the location between and roles of each element. Basically, the structure for the use of mobiles consists of base stations and mobile hosts. Stations are the access points for all mobile hosts communication. Base station provide communication transparency. To deal with the limited number of frequencies and signal attenuation (see page 25) the number and location of the base stations is important. Therefore, multiple base stations are used, which work together, too. Each base station covers its own area (cell) which adjoins to other areas (cells). To guarantee correct functionality each cell uses different frequency ranges as its neighbour. The density of base stations and the cell size may reflect the user density to provide better services. If a user leaves the range of a cell or another cell provides better transmission quality the base stations automatically have to handover the user to the new cell.

*ad hoc* networks are an extension to this where each mobile routes traffic to other mobiles itself which extends the range of each mobile. This builds a temporal network which requires cooperating mobiles\(^5\) and only supports a limited range. However, it is useful for meetings to exchange data between the participants, for example.

\(^5\)battery power is a valuable resource
To show a structure of a wireless network, GSM is chosen.

**GSM** The most important European standard for mobile data transmission is the Global System for Mobile telecommunications (GSM). GSM was developed for voice communication and to interact with the existing voice networks. Therefore, GSM is the mobile part of the telephone system. Data services are explicitly specified in GSM but the low data rate of 9.6 kbit/s is the big disadvantage of GSM which makes it not suitable for data communication and especially multimedia communication. However, the video and audio compression method MPEG-4 (see section 2.1.5) and H.263 [ITU-T 1996, Yeadon, Davies, Friday & Blair 1998] has been specialized for this environment. Besides, it will change in the near future because of the new development of a GSM standard with a data rate of 64 kbit/s. This promotes GSM to the mobile ISDN.

Nevertheless, GSM is the most feature-rich digital system for mobile communication. It permits telephony, facsimile, short message service, circuit switched data services, forwarding and barring services. The added features advice of charge, line identification services, call hold/waiting, multi-party and closed user group services are all compatible with ISDN. The GSM standard covers not only radio aspects but also service and network aspects like open interfaces and automatic roaming.

GSM is a Public Land Mobile Network (PLMN) and the structure (see figure 2.6) consists of the three main parts

![Figure 2.6: General architecture of a GSM network](image-url)

- **Mobile Station** The mobile station comprises the mobile equipment (the terminal) and the users’ smart card to identify him. The smart card is called Subscriber Identity Module (SIM) and contains the International Mobile Subscriber Identity (IMSI), a secret key for authentication and other information. The terminal is identified by International Mobile Equipment Identity (IMEI).

- **Base Station Subsystem** The Base Transceiver Station (BTS) and Base Station Controller (BSC) form the base station subsystem. The BTS exchanges signals with mobile stations and consequently builds a cell. A BSC controls several BTS and manages setups, handovers, etc. between the BTSs. Besides, the BSC forwards the data of the mobiles to the network subsystem.
• **Network Subsystem** The network subsystem switches the data like normal switches in fixed networks. The central element is the Mobile services Switching Center (MSC). Additionally, the MSC handles the registration, authentication, handover and location updating of each mobile. The call routing to a roaming subscriber is also done by the MSC. The Home Location Register (HLR) and Visiting Location Register (VLR) are databases to support the call-routing and roaming. The HLR contains all necessary information about each subscribed mobile whereas the VLR only contains the necessary information for the control and provision of services for each mobile inside the control area of the VLR. The Equipment Identity Register (EIR) stores all valid International Mobile Equipment Identity (IMEI) numbers to test the validity of a mobile. The Authentication Center (AuC) stores the secret keys of each subscriber and therefore requires special protection (Scourias, 1998; Dorner & Schiller, 1998).

As described in this section mobile systems have special requirements and limitations, and furthermore, communication with mobile systems over wireless links has new media characteristics which have to be considered by the used protocols and applications. Finally, GSM as an example for the structure of an mobile networks was explained. To handle QoS variations the use of adaptation techniques is necessary. Therefore, the next sections introduce some techniques to adapt protocols and applications for the use in mobile environments.

### 2.3 Adaptation for Varying Quality Of Service

As we could see in sections 2.1 and 2.2 a mobile multimedia system operates in a permanently changing environment and requires specific quality of services. This requires dynamic adaptations to achieve the required services and what includes reduced services which allow the basic functionality.

The use of mobile systems leads to a new environment for the system itself and applications on it if they communicate over a wireless connection with other systems. The fluctuations in the environment, quality of service, cost for service, physical location, capabilities of supporting hardware, has to be considered by system and applications (Katz, 1994). Therefore, the systems have to change from information hiders to providers:

Mobility requires *adaptability*. By this we mean that systems must be location and situation-aware, and must take advantage of this information to dynamically configure themselves in a distributed fashion (Katz, 1994).

Not all of the required information should be created inside of the advanced applications. Therefore, applications require environmental support to get information about changes in the infrastructure. The changes consist of add or remove operations of infrastructure elements, like special cards, or QoS changes. This allows feedback by application or user which can adapt own actions, mode changes, etc.
The use of knowledge about low level infrastructure information like communication and processor conflicts with propagated transparency paradigms like CORBA, DCE or RM-ODP. Transparency works good in static case but not in highly dynamic case. Therefore, service location, cost, penalties and availability, QoS, physical location and capabilities of supporting hardware have to be considered.

The transmission of video is the most demanding media type but it is time-critical and requires high bandwidth. As described in section 2.1.5 the bandwidth can be reduced by the use of compression techniques. MPEG compression is a static adaptation technique which prevents information loss. Filtering also reduces the required bandwidth but reduces the content of information, too. Therefore, filtering allows using a poorer connection through transmission of lower quality of multimedia data.

By changing from a fixed to a wireless LAN the link characteristic changes (see section 2.2.3), too. This leads to QoS changes, less bandwidth, other number and type of errors, temporarily disconnections through blackspots, etc. In WWANs the latency of connection establishment is in the range of seconds.

The processing of many problems in hardware is replaced through software wherefore better adaptation of protocols, download of new software and adaptation for inserted or removed cards is possible (Davies 1996).

Common adaptation techniques are summarized in table 2.5 (Friday 1996). The proxy concept can be used to adapt applications without internal changes of applications. This concept is detailed in chapter 4.

2.4 Summary

In section 2.1 four key issues of multimedia communication are identified (Blair & Stefani 1998), namely:

- support for continuous media,
- real-time synchronization,
- multiparty communication, and
- quality of service management.

To adapt to the available bandwidth multimedia data can be reduced wherefore the compression technique MPEG (see section 2.1.5) is used for video transmission.

In section 2.2 the special problems of mobile end systems were displayed. Furthermore, the wireless communication characteristic was explained.

Section 2.3 introduced the requirement of adaptation to react on the multimedia requirements and special mobile communication characteristic.
<table>
<thead>
<tr>
<th>Level</th>
<th>Technique</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Change of working practices</td>
<td>The user can alleviate demands on the network, e.g. change task, swap from synchronous to asynchronous collaboration or specify which tasks are most important to them.</td>
</tr>
<tr>
<td>Application</td>
<td>Restructure using agents or delegation of processing</td>
<td>Processing/network intensive tasks can be offloaded to remote sites or pre-processing or filtering applied to remote data (reducing bandwidth requirements and freeing host for other tasks/dozing to save power).</td>
</tr>
<tr>
<td></td>
<td>Use substitute services</td>
<td>The application can use local substitute services based on cached information (often with reduced functionality) while disconnected.</td>
</tr>
<tr>
<td></td>
<td>Change model of interaction</td>
<td>Interactions can be adjusted from polling to event-based structures or from RPC to an alternative (perhaps asynchronous) paradigm.</td>
</tr>
<tr>
<td></td>
<td>Reorganise application structure</td>
<td>One example of application restructuring is to change from using distributed state to a centralised architecture to simplify consistency management in unreliable conditions.</td>
</tr>
<tr>
<td></td>
<td>Re-bind to new services</td>
<td>The application may be able to rebind to equivalent services which are easier/cheaper to access. Alternatively, it may be possible to migrate the service or application component.</td>
</tr>
<tr>
<td></td>
<td>Change application demand</td>
<td>New quality of service requirements can be negotiated or non-essential communication streams dropped. Alternatives may be possible, e.g. lossy encoding.</td>
</tr>
<tr>
<td></td>
<td>Adjust consistency requirements</td>
<td>Considering multicast, groups may be able to tolerate weaker consistency or adjust operations to achieve quorum, yet avoid hard to reach members.</td>
</tr>
<tr>
<td>Middleware</td>
<td>On-demand cache management</td>
<td>Information can be fetched only when needed, instead of speculatively, e.g. opening the first page of a document and transferring successive pages later, or retrieving e-mail headers before message bodies.</td>
</tr>
<tr>
<td></td>
<td>Prefetching into the cache</td>
<td>The application can fetch information while the link is good, in case it is required when the link degrades or becomes expensive.</td>
</tr>
<tr>
<td></td>
<td>Apply filtering and compression</td>
<td>The volume of information to transfer can be reduced by compression or filtering non-essential frames from hierarchically encoded data.</td>
</tr>
<tr>
<td></td>
<td>Efficient protocol utilisation of the channel</td>
<td>The transport mechanisms can be adjusted to match channel characteristics, e.g. retransmission/backoff strategies, header compression, error control and handling of asymmetric channels.</td>
</tr>
<tr>
<td>Transport and below</td>
<td>Change or introduce new protocols</td>
<td>New protocols can be selected which suit the characteristics of a particular network or appropriate protocols can be introduced (e.g. injecting a reliable data link layer).</td>
</tr>
<tr>
<td></td>
<td>Optimise network data</td>
<td>Protocols can adjust their packet sizes to suit different networks. The operating system can adjust the queue sizes onto the network interfaces which impacts on latency, particularly of multimedia streams.</td>
</tr>
<tr>
<td></td>
<td>Optimisation of multicast</td>
<td>Multicasts can be mapped onto the network technology, particularly those with partial or full hardware multicast support.</td>
</tr>
<tr>
<td></td>
<td>Optimise for the characteristics of the network</td>
<td>There are a number of cost and network structure optimisations. For instance, batching data to spread the dialing delays, or transferring additional information while the time is already paid for.</td>
</tr>
<tr>
<td></td>
<td>Reordering of data</td>
<td>The priority or urgency of data may require that it is handled preferentially in scarce bandwidth situation.</td>
</tr>
<tr>
<td></td>
<td>Demultiplexing to multiple networks</td>
<td>If multiple technologies are available simultaneously, it may be advantageous to use several at once.</td>
</tr>
</tbody>
</table>
3. Common Object Request Broker Architecture (CORBA)

The Common Object Request Broker Architecture (CORBA) is part of the Object Management Architecture (OMA). The OMA is the framework defined by the Object Management Group (OMG). The OMG is the world’s largest computer industry consortium and a nonprofit organization with about 700 corporations financing it. It started in 1989 with eight members.

The organization remains fairly small and does not develop any technology or specifications itself. It provides a structure for its members to specify technology and then produce commercial implementations (Vogel & Duddy 1997, p. 244).

The main aim of the OMG is to support integrated software systems especially developing and applying them. Therefore, the OMA framework (see figure 3.1) has been developed. The created infrastructure shall be a solution for portability, reusability and interoperability problems. The basic approach for solving the problems is the object management paradigm which we also see in the name of the OMG. Therefore, it consists of two fundamental models: the Core Object Model (see section 3.1.1) and the OMA Reference Model (OMG 1997c, Mowbray & Ruh 1997, Vogel & Duddy 1997).

The OMA Reference Model contains the following basic elements:

Components the Object Request Broker (ORB),

Interfaces there are four categories defined:

Object Services are general services,
Common Facilities are end-user oriented facilities,
Domain Interfaces are domain-specific interfaces,
Application Interfaces are non-standardized application-specific interfaces,

Protocols like the general inter-ORB protocol (GIOP).

The OMA is a Distributed Object Management (DOM) architecture. It gives an overview and a base for discussion about the expected components. The heart of the OMA is the ORB which is the communication infrastructure of the whole framework (see section 3.1) for the distributed environment. It allows access to distributed
objects. The protocols and interfaces (see above) are common possibilities to communicate with the ORB and to use standardized services, facilities and domain (see section 3.2). The implementation aspects are not specified because to be OMG conform it is only necessary to implement the interfaces. This makes it possible to choose non object-orientated programming languages and to interconnect different languages across the ORBs. That leads to three important theoretical fundamentals for the OMA:

- Object-oriented model
- Open distributed computing environment
- Component integration and reuse

The most discussed and important part of the OMA is the ORB. The name of the ORB framework is CORBA which will be discussed in detail in the next section. This is followed by an overview of the CORBA services and CORBA facilities which are not part of the CORBA core standard but which are necessary to understand the possibilities of CORBA (OMG 1997c; Mowbray & Ruh 1997).

### 3.1 The CORBA Standard

If people are talking about CORBA they mostly mean the OMA. However, the CORBA standard (OMG 1998a) mainly contains the ORB specification and all things related to it. This includes among other things the object model, interfaces, communication, etc. The OMA interfaces like CORBA services are a separate standard which uses CORBA as platform and adds new functionality to it. To give a little introduction to CORBA the reason for some components follow now. In the sections thereafter they will be explained in detail.
3.1. The CORBA Standard

One important aspect of CORBA is to provide a high-level communication infrastructure. The ORB routes requests and responses from one object (see section 3.1.1) to another. This includes objects of other components of the architecture which are distributed over the whole system. Besides, it shows that in CORBA the client-server approach is being replaced by a client-ORB-server approach. Therefore, the applications are provided with common services, like basic messaging and communication, and they are independent of all underlying differences.

To integrate objects it is necessary to specify standard interfaces. This may be done with the interface definition language (IDL) (see section 3.1.1). This allows to hide the implementation behind the interfaces. Therefore, the usage of the object is independent from the exact implementation, physical location, platform type, networking protocol and programming language. The ORB itself also has an IDL-based architecture which will be explained later (see section 3.1.2).

The communication between the objects is per default peer-to-peer and synchronous in an at-most-once manner. Asynchronous communication is only available in using one-way and best-effort manner. CORBA itself uses the General Inter-ORB Protocol (GIOP) (see section 3.1.3) for communication which allows interoperability between ORB implementations. GIOP is only a transfer syntax which makes a real protocol necessary. Until now, only the Internet Inter-ORB Protocol (IIOP) is defined which uses the Internet technology. The ORB itself controls the whole communication. (OMG 1998a, Mowbray & Ruh 1997)

The CORBA Standard shall be detailed in the following sections. First, IDL is explained that is a base part of the whole CORBA standard. Second, the CORBA architecture with the interface definitions is introduced. Finally, the interoperability between ORBs is discussed with focus on GIOP and IIOP.

3.1.1 The Object Model and Interface Definition Language

To understand IDL one must know the object model of CORBA. All features of the object model must be captured by IDL because it shall describe these features. Therefore, the object model and IDL can be described together. All that is valid for the object model should be valid for IDL and vice versa except the special restrictions of a model and a language. An example for the exceptions is the module concept of IDL.

The object model of CORBA is the elementary part of the whole standard. It is derived from the abstract Core Object Model of the OMA. It describes the view, called interface, on each object. This view is described using the interface definition language (IDL) of the OMG for CORBA (OMG 1998a, chap. 3). It is a language to describe interfaces in a language independent manner and is also adopted by the International Organization of Standardization (ISO) under the name ISO DIS 14750. It shall not describe implementation characteristics like behaviour, instances or relationships. Therefore, the following constructs are defined:

- **Constants** to assist with type declarations
- **Data types declarations** to use for parameter typing
Attributes which allows getting and setting of a value of a particular type

Operations which take parameters and return values

Interfaces which group data type, attribute, and operation declarations

Modules for name space separation (Vogel & Duddy 1997)

Thus, each client can invoke remote operations on an object without knowing about the implementation details.

An IDL interface acts as a contract between developers of objects and the eventual users of their interfaces (Vogel & Duddy 1997).

Therefore, IDL is a base of the whole CORBA standard and an important point in developing distributed systems with CORBA. It allows the reuse and interoperability of objects in a system. Therefore, it is necessary to keep the specification of IDL stable. The mapping between IDL and the programming language is defined in the CORBA standard (OMG 1998a).

IDL is very similar to C++ containing pre-processor directives (include, comments, etc.), grammar as well as constant, type and operation declarations. There are no programming language features like e.g. if-statements.

Data Types. IDL supports the most basic data types from C++ but there are no int and references in IDL (see section 3.2) (Baker 1997). Instead exists string, boolean and any. Some data types have a different specification like char which is a real character type in CORBA. The constructed types typedef, enum, struct, array and union are similar to C++. The template type sequence is a variable length array of elements of one, but any, IDL type. The type string is like sequence but it only supports ASCII ISO-Latin characters. Any is like Object in JAVA a placeholder for any possible IDL type which is often used by CORBA services.

The most important part of IDL are operation specifications. The operations are a part of interfaces for an object. Interfaces may be summarized to modules. This leads to hierarchical composition with naming scopes.

The next sections explain the module and interface concept and operation declarations.

Modules and Interfaces

The main concept behind IDL are the interfaces. Each interface may contain constants, types, attributes, exceptions and operations (see section 3.1.1) for one object. A module is a method to separate name spaces and may contain any IDL construct. Each IDL construct is automatically public according to the object orientated concept. (Multiple-) Inheritance is only permitted for interfaces and not for modules. It is not forbidden to derive interfaces in a cycle. The concepts may be seen best in the example below which shows a module called Proxies that contains a basic interface proxy which is refined for filtering proxies and this is refined for filtering MPEG streams.
module Proxies {

    interface Proxy {
        // basic proxy operations and attributes
    };

    interface FilterProxy : Proxy {
        // Inherits all basic proxy operations and attributes
        // adds FilterProxy definitions
    };

    interface MPEGFilterProxy : FilterProxy {
        // Inherits all FilterProxy operations and attributes
        // adds MPEGFilterProxy definitions
    };

};

Concrete language mappings between C, C++, SmallTalk, Cobol, Ada, Java and IDL can be found in the standard (OMG 1998a). Forward declarations of interfaces permit their use before they are defined. Each IDL file must end with .idl. OMG IDL definitions may be found in orb.idl.

Attributes and Operations

Each client knows the IDL interface specification of each object containing all information about the object. Attributes are like variable definitions but they behave like operations in CORBA. Each read-write attribute gets a set- and get function and each read-only attribute gets a get function. The main part are the supported operations of an object. An operation declaration consists of

- an operation attribute that specifies the invocation semantic,
The default operation attribute is synchronous invocation and one-way stands for asynchronous invocation. Each operation may return exactly one value or must be void. The parameter list contains the mode, type and name of each operation parameter.

Exceptions are an easy programming concept to handle the distributed system’s characteristics, especially errors. The context expression allows the client to transfer context specific information, like security context information for the security service.

An example containing declarations of interfaces with operations and attributes is given below. It was taken from the NamingService of CORBA services (see section 3.2) [OMG 1997a].

```idl
// OMG IDL for the Common Object Services.
// Naming Service V1.0, 3/94
module CosNaming {

  typedef string Istring;

  struct NameComponent {
    Istring id;
    Istring kind;
  }

  typedef sequence <NameComponent> Name;

  enum BindingType {nobject, ncontext};

  struct Binding {
    Name binding_name;
    BindingType binding_type;
  }

  typedef sequence <Binding> BindingList;

  interface BindingIterator;

  interface NamingContext {

    enum NotFoundReason { missing_node, not_context, not_object};

    exception NotFound { NotFoundReason why; Name rest_of_name; }

    exception CannotProceed { NamingContext cxt; Name rest_of_name; }

    exception InvalidName{};
    exception AlreadyBound {};
    exception NotEmpty{};

    void bind(in Name n, in Object obj)
      raises(NotFound, CannotProceed, InvalidName, AlreadyBound);

```
void rebind(in Name n, in Object obj)
    raises(NotFound, CannotProceed, InvalidName);

void bind_context(in Name n, in NamingContext nc)
    raises(NotFound, CannotProceed, InvalidName, AlreadyBound);

void rebind_context(in Name n, in NamingContext nc)
    raises(NotFound, CannotProceed, InvalidName);

Object resolve (in Name n)
    raises(NotFound, CannotProceed, InvalidName);

void unbind(in Name n)
    raises(NotFound, CannotProceed, InvalidName);

NamingContext new_context();

NamingContext bind_new_context(in Name n)
    raises(NotFound, AlreadyBound, CannotProceed, InvalidName);

void destroy( )
    raises(NotEmpty);

void list (in unsigned long how_many, out BindingList bl,
          out BindingIterator bi);

};

interface BindingIterator {
    boolean next_one(out Binding b);
    boolean next_n(in unsigned long how_many, out BindingList bl);
    void destroy();
};


3.1.2 ORB Architecture

The ORB architecture (see figure 3.3) consists of

**IDL Stubs** the clients static interfaces to invoke objects,

**Dynamic Invocation Interface** one dynamic interface to invoke objects which are unknown at compile time,

**ORB Interface** the interface to the ORB core,

**Static IDL Skeleton** the analogue to the client’s IDL stubs on the object implementation side (server),

**Dynamic Skeleton** the analogue to the client’s dynamic invocation interface on the object implementation side (server),
Object Adapter  a special interface for an object implementation to communicate with the ORB,

ORB Core  the ORB core.

**Figure 3.3: The CORBA Architecture**

This shows that the ORB core only offers standardized interfaces to any application. Therefore, the core must not be specified and only the interfaces are a topic for standardization. The interfaces may be distinguished by predefined interfaces, object adapters and user-defined interfaces. Besides, the static or dynamic manner is important.

The ORB interface is static and provides common basic services for each application. The static interfaces, stub and skeleton, are IDL generated and consequently user-defined. The Dynamic Invocation Interface (DII) and Dynamic Skeleton Interface (DSI) are predefined interfaces for dynamic invocation. The Object Adapter (OA) gives an object implementation the possibility to interact with the ORB to offer special features.

In the next sections these interfaces will be explained in depth.

**ORB Interface**

A direct communication with the ORB core is possible through the ORB interface. It offers common functions for all objects and is independent on the special ORB implementation, the objects interfaces and the object adapter. Because of this, only few operations are defined. They operate on the ORB or on object references and can be invoked from clients and implementations.

The main problem solved by the ORB interface is the handling of object references, especially for storing purposes. As the object references depend on the ORB implementation they cannot be used for storing. Therefore, they can be converted to strings and vice versa. If the ORB supports the Interoperable ORB Protocol (IOP)
3.1. The CORBA Standard

the stringified reference can be correctly transformed to object references by each ORB which uses IOP. Furthermore, the ORB interface supports operations on object references, like comparison and existence.

The second aspect of the ORB interface are ORB initialization and information about supported services, like Naming Service (see section 3.2). This includes methods to resolve initial references for solving the bootstrapping problem.

User Defined Interfaces

The IDL stub and Static Skeleton Interface (SSI) are created by the user IDL definitions. The IDL stub contains the methods which the client must invoke for the operations it requires. The SSI defines the methods which an object must implement to make an invocation from the ORB possible. Therefore, the SSI is the analogue of the IDL stub. The important point by user defined interfaces is that they must be known at compile time. It is not possible to use this technique with new interfaces at run time. For this, the dynamic interfaces are defined (see section 3.1.2).

![Diagram of Object Request](image)

For an operation request (see figure 3.4) the client invokes an IDL stub method and blocks until the method returns. The stub passes the call to the ORB core and the core connects to the server ORB core. The server ORB invokes the suitable skeleton method and executes the object. The invocation should return with a return value which is passed by the server ORB core to the client ORB core and then to the IDL stub method. Now, the IDL stub method can return with a value and the client can process further. If an error occurs at any point in the procedure an exception must be returned to the client. The client should catch exceptions for special handling. Every return value that comes with an exception is per definition invalid.

The method invocation seems like a normal method invocation for the client. The client should not see any difference to a normal method invocation. The client or the server side have no knowledge about the interface type of the request or object invocation. A client request obtains the same semantics if it is invoked with an IDL
stub or a DII (see section 3.1.2). The same is valid on the object side, where it is the same semantics whether the invocation comes from an SSI or DSI (see section 3.1.2).

**Dynamic Interfaces**

The problem of static interfaces, like the ORB interface, is the knowledge about the interfaces at compile time. It should be possible to use interfaces which will be detected first at run time. Therefore, the *Dynamic Invocation Interface* (DII) and the *Dynamic Skeleton Interface* (DSI) have been defined. The DII gives the methods to create requests and to invoke requests to an object. The DSI is the analogue of the DII at the server side. The DSI permits an object to get *unknown* parameters at compile time. This is useful for debuggers, for example.

Generally, static interfaces are used for object invocation. The dynamic handling is very intensive in programming and should be only used if it is really necessary.

**Object Adapter**

Object implementations use *Object Adapters* (OA) to access services which are provided by the ORB. They are specialized for their task and provide a common interface to the ORB core to handle the wide range of services. OAs are also a middleman between the ORB core and the DSI and SSI.

Object adapters are responsible for the following functions:

- Generation and interpretation of object references
- Method invocation
- Security of interactions
- Object and implementation activation and deactivation
- Mapping object references to the corresponding object implementations
- Registration of implementations ([OMG 1998a](#), p. 2-15)

It is not specified which tasks must be implemented in the OA and which in the ORB core. Only the interface of the OA to the object implementation says what is possible and what not. Therefore, one OA, the *Portable Object Adapter* (POA), formerly the *Basic Object Adapter* (BOA), is specified by the CORBA specification. It supports common functions which are widely necessary and independent of vendor ORB implementations. Hereby, the interoperability should be supported between different ORB implementations.

**3.1.3 Interoperability and Portability**

Interoperability stands for an ORB-to-ORB communication between different vendors. The applications which use the ORB shall see nothing of this. For this, three elements of interoperability are defined:
3.1. The CORBA Standard

- ORB interoperability architecture
- Inter-ORB bridge support
- General and Internet Inter-ORB Protocol (GIOP and IIOP)
- additional, Environment-Specific Inter-ORB Protocols (ESIOPs)

The ORB Interoperability Architecture consists of three requirements (OMG 1998a, p. 11-1):

- Ability of two vendor’s ORBs to interoperate without prior knowledge of each other’s implementation.
- Support of all ORB functionality.
- Preservation of content and semantics of ORB-specific information across ORB boundaries (for example, security).

It uses a domain and bridge concept to fulfill the requirements.

The General Inter-ORB Protocol (GIOP) was defined for the ORB to ORB communication.

The General Inter-ORB Protocol (GIOP) element specifies a standard transfer syntax (low-level data representation) and a set of messages formats for communications between ORBs (OMG 1998a, p. 10-3).

Because of the lack of a transport protocol specification further protocols are necessary. Therefore, the Internet Inter-ORB Protocol (IIOP) was defined for a direct mapping of GIOP to Transmission Control Protocol/Internet Protocol (TCP/IP) of the Internet.

The Internet Inter-ORB Protocol (IIOP) element specifies how GIOP messages are exchanged using TCP/IP connections. The IIOP specifies a standardized interoperability protocol for the Internet, providing out of the box interoperability with other compatible ORBs based on the most popular product- and vendor-neutral transport layer (OMG 1998a, p. 10-3).

The heart of the IIOP is the Interoperable Object Reference (IOR) that contains the internal object reference, which is the ORB reference to the object, the Internet host address and port numbers. GIOP and IIOP are mandatory for each ORB implementation. There is also an architecture for interworking between CORBA and Microsoft Component Object Model (COM) (OMG 1998a, chap. 15). A comparison of both concepts can be found in Tallman & Kain 1998.

The interoperability between different ORB implementations leads to a widespread use and the use of different ORB implementations together, too. Hence, this is not enough to use the same applications without changes on different ORBs. Therefore, it is not good to use vendor specific features and it is recommended to use only standard features (see also 3.2). This is a recommended aim for each software design and it may be an important point for the survival of a company. Hereby, portability is also an important point which leads to independence from vendors and specific products.
3.2 CORBAservices, CORBAfacilities and CORBAdomains

CORBAservices and CORBAfacilities are part of the OMA standard and allow the possibility to provide common services to an application. In (Vogel & Duddy 1997, p. 256) CORBAservices can be seen as

This set of interface specification provides fundamental services that application developers may need in order to find and manage their objects and data, and to coordinate the execution of complex operations.

and CORBAfacilities can be seen as

Common Facilities are those end-user-oriented interfaces which provide facilities across application domains.

CORBAdomains are also part of the OMA standard. They can be seen as vertical-market areas that consider only their own interoperability needs and not common needs across the domains. This is a distinction to services and facilities where common things are provided. Examples are financial services, healthcare and telecommunications.

![Object Request Broker](image)

Figure 3.5: Object Management Architecture Technical Reference Model

Everything outside of CORBA and CORBAservices can be found in CORBAfacilities or CORBAdomains. A CORBA service should be primitive and not depend on other services. CORBA facilities should extend services to provide more capabilities and CORBAdomains should do the same with facilities. That leads to a hierarchical order of services, facilities and domains. However, there is no exact distinction between the layers. Therefore, it is a point for discussion and must be decided for each interface.

The published services include all services of table 3.1. Only the Naming Service and Trading Service are used in chapter 5 wherefore they are described below.
Table 3.1: CORBA services

<table>
<thead>
<tr>
<th>Information</th>
<th>Task</th>
<th>System</th>
<th>Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collections</td>
<td>Events</td>
<td>Lifecycles</td>
<td>Security</td>
</tr>
<tr>
<td>Persistence</td>
<td>Transactions</td>
<td>Naming</td>
<td>Time</td>
</tr>
<tr>
<td>Relationships</td>
<td>Concurrency</td>
<td>Query</td>
<td></td>
</tr>
<tr>
<td>Properties</td>
<td></td>
<td>Licensing</td>
<td></td>
</tr>
<tr>
<td>Externalization</td>
<td></td>
<td>Trading</td>
<td></td>
</tr>
</tbody>
</table>

**Naming Service**  The naming service is a generic directory service which realizes a *white pages* service. This allows the user to get object references by using the name of an object. Thereby, the naming service supports the distribution of object services between different participants. This is especially useful to distribute object references at booting a whole CORBA system.

To use a naming service CORBA objects can *bind* to the service by providing a name and an object reference. Names of objects are hierarchically ordered and therefore build a naming tree. The tree provides a unique name for each object to identify it. This name can be used to *resolve* an object by using its name ([OMG 1997a](#), [Mowbray & Malveau 1997](#), [Baker 1997](#)).

**Trading Service**  The trading service is a *yellow-pages* service. Therefore, the trading allows registering of service offers and querying them. The querying service uses service characteristics and qualities which have to be provided by each offer to identify them. Thereby, matching offers can be identified and delivered ([OMG 1997b](#), [Mowbray & Malveau 1997](#), [Baker 1997](#)).

### 3.3 Summary of CORBA

In this chapter the CORBA specification was introduced and how it provides an object model and a distributed communication infrastructure was explained. For it, IDL was specified to permit the specification of software interfaces. The interfaces are independent of the programming language or the platform which is used. Hereby, a universal applicable infrastructure for distributed communication is given. However, until now each application implementation is not vendor independent because of special features or no exact conformity with the standard.

For better reuse, infrastructure capabilities and application interoperability CORBA services and CORBA facilities have been defined. But until now, not all CORBA services or CORBA facilities are available on each platform.

CORBA is a *de-facto* industry standard and there are many ORB implementations available. Besides, there exists an architecture for interoperation between CORBA and COM. Hereby, the ongoing success of CORBA should go further and become a widely accepted standard for distributed communication.
4. Proxies

As we could see in chapter 2 multimedia applications require special QoS which must permanently be monitored and adapted to guarantee transmission. To be used for wireless communication the unpredictable variations of QoS in latency and error rate have to be considered. Resource reservation is almost impossible due to the special media characteristics of wireless communication (see section 2.2.3). Therefore, adaptation through decrease of data or delayed transmission is necessary. Data can be decreased by filtering or requesting lower quality multimedia data. One adaptation approach is the use of new applications for the mobile environment like described in the Rover-Project (Josephs, deLispinasse, Tauber, Gifford & Kaashoek 1995). However, to be more flexible and to use existing applications without greater modifications another approach is necessary. Therefore, proxies are used in this work to adapt mobile, distributed multimedia applications.

In the remainder of this section the definition of proxies and the basic concept behind them is introduced and their use for mobile applications is shown in a few examples of developed and used proxies. Thirdly, the new developed proxy concept called Reactive and Adaptive Proxy Placement (RAPP) is introduced.

4.1 Design Pattern Proxy

A software design pattern describes a family of solutions to a software design problem (Tichy 1998).

In (Tichy 1998) an approach for categorizing design patterns is made. It uses solved problems as categorizing scheme. The proxy pattern is contained under the subtree “Decoupling → Layers”. The purpose of decoupling is

Decoupling patterns divide a software system into several independent parts in such a way that they can be built, changed, and replaced independently as well as reused and recombined in unforeseen combinations (Tichy 1998).

and the purpose of layers are:

A (software) layer provides an interface and an implementation of this interface (Tichy 1998).
The *proxy* or *surrogate* design pattern is described as

> Provide a surrogate or placeholder for another object to control access to it ([Gamma, Helm, Johnson & Vlissides 1995](#) p. 207).

with the purpose to

> add or withdraw unplanned functionality transparently ([Tichy 1998](#)).

This leads to the fact that “The Proxy design pattern makes the clients of a component communicate with a representative rather than to the component itself.” ([Buschmann, Meunier, Rohnert, Sommerlad & Stal 1996](#)). A component can be a whole application, too. Furthermore, an important point is that a “Proxy defines a representative or surrogate for another object and does not change its interface” ([Gamma, Helm, Johnson & Vlissides 1995](#) p. 150).

The advantage of proxies are the possibility to let client or server, or both, remain unchanged or only be changed very little. This makes the introduction of new features for legacy systems easier and more flexible. Furthermore, developing and widespread use is easier through independence of concrete applications and location. Proxy aware applications like web proxies are commonly used. Therefore, web browsers can redirect their stream to proxies to give control to a proxy. This is typically done for caching or for use with firewalls.

### 4.2 Proxies in network area

Proxies in the network area are used to handle network heterogeneity, which can arise especially in mobile computing. A list of examples is given below:

- Adapting changes by use of mobile hosts, for example, when changing from high to low speed connection, changing hardware/software, etc.
- Using with firewalls to support security restrictions.
- Processing of data on another more powerful machine with special hardware/software if necessary.
- Caching for better performance.

In general, a proxy is an intermediate system that acts on behalf of a client by receiving data from a data source, processing the data and then relaying it to the client.

The tools for adaptation are among other things optimized protocols (application and network), dropping data, compressing data, delaying data, re-segmenting data, acting on behalf of a connected client, trading off computation and communication, etc. ([Zenel & Duchamp 1997a](#)). This proxy functionality can be divided into the three areas network or transport layer, application protocol or application data as you can see in table 4.1 ([Seitz & Davies 1998](#)).

The insertion of proxies arises new problems. A few problems and their solution are discussed below ([Zenel & Duchamp 1997a](#)).
### 4.2. Proxies in network area

<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network or transport</td>
<td>The proxy uses a modified protocol and/or changes the data stream. Thereby, the proxy can react instead of the mobile client.</td>
</tr>
<tr>
<td>layer</td>
<td></td>
</tr>
<tr>
<td>application protocol</td>
<td>The proxy uses a modified application protocol to adapt to the wireless link, for example, delay of packets in case of a bad wireless link.</td>
</tr>
<tr>
<td>application data</td>
<td>The proxy modifies certain application data to reduce required bandwidth. Furthermore, processing of computation intensive tasks instead of the mobile is possible.</td>
</tr>
</tbody>
</table>

**Routing**  It has to be guaranteed that all traffic for a host is routed through the corresponding proxy (proxies) if one is installed. If a proxy server exists, which performs as a platform for proxies, using a static route is possible. If a mobile host is used the protocol has to be changed if necessary. If there is no server proxy then a handover mechanism between server-proxy and proxy-client is necessary (see section 4.3).

If participants of a connection can move as it is the case by mobile hosts the connection between proxy and mobile can be interrupted by moving. In this case, a new proxy has to be installed at the new location or the connection has to be rerouted over the old proxy.

**Connection semantics**  The insertion/removal of a proxy should not lead to a connection interruption. By using a (static) null proxy this is no problem because of the insertion during connection establishment. If no null proxy is used the handover process is more critical through the establishment of new connections during installation of proxies.

If a proxy is inserted a new connection to this proxy is necessary. By using a connectionless transport protocol like UDP this can easily be realized because of lacking a “static” connection. A datagram transport protocol uses no connections and therefore only the receiver address of each packet has to be changed. In case of a connection-oriented transport protocol like TCP a new connection has to be established from server to proxy and proxy to client. After the successful establishment of the new connections the stream can be rerouted over them.

The end-to-end semantics at transport layer is not given if a proxy is used leading to two connections instead of one. In this thesis the focus is of application level and therefore the end-to-end semantics is not discussed further.

**Proxy placement**  The proxy placement should not lead to unnecessary increase of latency and therefore it is recommended to use a proxy host which lies on the route. However, which position in comparison to server and client is the best? Well, there is no general solution because of the security, robustness and application context. If a mobile host is used the most preferable locations are the server or the mobile host.
itself. If filtering or compression is used the reduced bandwidth should be used on the whole link and not only partly and therefore, the server or mobile host is a good choice. This is discussed further in section 4.3. The security and robustness aspect is not further discussed in this thesis.

Proxy control The proxy control introduces the questions of what is controlled, who controls and what features are offered? Control of the proxy itself (insertion/remove) and capabilities are the two answers for what is controlled. Internal and external control is the answer for who controls. The internal case reflects control by application and the external case reflects control by environment (QoS monitoring) or user input. The range of capabilities depends on the degrees of freedom which are inherent in the stream type which is controlled. Furthermore, to be effective using a proxy capability should lead to a “visible” change. Only small changes are not satisfactory to adapt to a bad situation in the environment.

4.2.1 Proxies in mobile computing

Using proxies for mobile computing and wireless communication permits an integration with fixed networks and distributed applications through adaptation mechanisms. Therefore, proxies for mobile applications normally divide the communication between client and server into two parts (see figure 4.1). As we saw in section 4 the link is normally divided in wireless and wired parts. Therefore, base stations are the interface between both parts. It is made the assumption that the wireless link creates problems and not the wired network. Thereby, proxies can be used on the mobile and base station to handle the wireless link problems or the location problem through moving. The wired network does not recognize the communication with a mobile and therefore requires no changes (Badrinath, Bakre, Imielinski & Marantz 1993).

![Figure 4.1: Proxies in mobile computing](image)

A few examples of existing proxy systems are described below.

Network- and transport layer proxies

Indirect TCP (I-TCP) Indirect TCP (I-TCP) divides a TCP connection into two differently handled TCP connections. The connection is split near a base station to handle the wireless and wired link with own connections. Thereby, the proxy
4.2. Proxies in network area

confirms all data to reduce packet retransmission between client and server in case of failure. Packet retransmission now only take place between server to proxy and proxy to client necessary. However, the end-to-end context gets lost. Furthermore, the slow-start-mechanism of TCP is not suitable for wireless links. Packet losses on wireless links are not an indication for a congestion but an indication of bad quality. Therefore, the proxy can use another mechanism when noticing high packet loses (Bakre & Badrinath 1995).

Snoop-Protocol The snoop protocol divides a TCP connection like I-TCP, too. However, the end-to-end context is maintained wherewith confirmations guarantee the delivery of packets to the receiver (Balakrishnan, Seshan & Katz 1995).

Application protocol proxies

The most used application protocol for proxies is the Hypertext Transfer Protocol (HTTP) of the World Wide Web (WWW). They are used in firewalls and to cache data (Padmanabhan & Mogul 1995). Another modification of an application protocol is Low Bandwidth X Protocol (LBX) for X-applications (Fulton & Kantarjiev 1993).

Application data proxies

Computing intensive applications like in Computer Aided Design (CAD) or network management can shift their data to more powerful computers for processing. Furthermore, the data volume for transmission can be reduced by compressing, filtering or combining data (Citrin, Hamill, Gross & Warmack 1997).

Mixed proxy examples

The above described examples specialized for one functionality area. However, the following examples show how it is possible to mix proxy functionality.

• OreOs (Brooks, Mazer, Meeks & Miller 1995) modify HTTP to allow redirections to other servers if there is a network congestion. Furthermore, transmitted data formats can be changed to save bandwidth.

• The GloMop-Project (Fox & Brewer 1995) allows real time lossy compression of images in HTML pages to reduce required bandwidth. Furthermore, image details with better quality can be reloaded. The Pythia-Project (Fox & Brewer 1996), successor of GloMop, additionally allows to reduce colour depth, resolution, etc. to adapt the mobile client capabilities.

• A range of proxy functionality was developed in Bruce Zenel’s proxy architecture (Zenel 1998, Zenel & Duchamp 1995, 1997). The filters include Internet Control Message Protocol (ICMP) or TCP for network and transport level, Network File System (NFS) or Simple Mail Transfer Protocol (SMTP) for application protocol filter and HTTP or MPEG filter for application data.
4. Proxies

- The approach “service proxies” in (Hokimoto, Kurihara & Nakajima [1996]) and the proxies in “WebExpress” in (Housel & Lindquist [1996]) use two proxies between client and server. One proxy on client side and one on server side to adapt to the wireless link.

4.2.2 Shortcomings of approaches

The solutions described in the section above are too specialized and inflexible for mobile, distributed multimedia applications. They mostly use specific applications, especially WWW access for mobiles over wireless links. They require pre-determined nodes for proxy instantiation and provide flexible run-time configuration with personal profiles. This limits the scalability and potential usefulness of these systems.

A summary of this shortcomings is given in the following description:

- **Extensibility** Most approaches developed specific proxies for certain applications or for special data types, but it is often hard to extend the architectures to deal with new applications and data types.

- **Flexible Behaviour** In most systems the behaviour of the proxy is statically defined. In addition to this, in general it is not possible to fine-tune proxy operations to cope with variations of the wireless link’s QoS.

- **Flexible Placement** The location of the proxy is an important factor in determining the performance of the system.

- **Scalability** Finally, there is the issue of scalability. All the described approaches work well, if you have one or two mobile hosts, each being represented by its own proxy, or if you scale to the number of mobile nodes by installing a high-performance computer or cluster of computers to host the proxies (e.g. (Fox & Brewer [1996])). However, for the system to scale proxies must be distributed throughout the system.

Therefore, a new approach called Reactive and Adaptive Proxy Placement (RAPP) was developed to permit a more general and flexible adaptation mechanism. RAPP is described in the next section.

4.3 Reactive and Adaptive Proxy Placement (RAPP)

The new generic architecture developed to treat the shortcomings described in section 4.2.2 permits proxy insertion/removal as an interpose in a connection and dynamic control of them to adapt to the network quality. This all gives tools to manage the whole proxy life cycle. It was designed to handle the wireless part between mobile and fixed network but it can be used for each communication endpoints located anywhere.

Firstly, the architecture overview is showed. Secondly, proxy objects as fundamental elements of the architecture are described. Thirdly, the proxy object selection process is explained. Finally, proxy object installation is detailed.
4.3 Reactive and Adaptive Proxy Placement (RAPP)

4.3.1 Architecture overview

The RAPP architecture (Seitz, Davies, Ebner & Friday 1998) can be used to extend an existing distributed multimedia application consisting of a client, a server and a specific application protocol. The extension adds CORBA adapters to an application and implements two additional processes (see sections 4.3.3 and 4.3.4). This all can be seen in figure 4.2.

4.3.2 Classification of Proxy Objects

To determine a suitable proxy object for installation a classification is necessary. The classification is hierarchically organized in order to allow flexibility in definition and use (see section 4.3.2). The levels of distinction are classes and types which are described below.
Classes of Proxy Objects

Three classes of proxy objects for data filtering, data type modification and data caching have been identified:

- **Filtering Proxies** modify the input stream without changing the stream type and caching data. Hence, the amount of data decreases but the input and output stream types are still the same. An example is an MPEG filter to reduce the required bandwidth by dropping colours or less relevant picture frames [Yeadon 1996].

- **Transforming Proxies** process the incoming data without caching and not only changes the amount of data to be sent but also its stream type. Such a proxy might compress an ASCII stream into a compressed binary stream in order to save bandwidth.

- **Caching Proxies** store data so that the client can retrieve them later. This is useful for decoupling a mobile client from a wired network. One example is a mail cache which saves sent e-mails until the mobile client is connected to a wired network.

Hence, these proxy classes can also be combined. Because of the realization of each class as an interface (see section 4.3.2) a combined proxy object has only to implement the combination of the interfaces.

Types of Proxy Objects

A further distinction are the received and sent stream type. Streams are classified based on the content types and subtypes of the Multipurpose Internet Mail Extensions (MIME) standard [Freed & Borenstein 1996]. The content types are narrowed to the five types: “text”, “application”, “image”, “audio” and “video”. Each of the five stream types can be further subdivided using fine-grain sub-types shown in figure 4.3.

![Figure 4.3: Stream Types and Subtypes](image)

Additional to the stream type and subtype an enhanced characteristic like “encrypted” or “compressed” can be used.
Interfaces of Proxy Objects

Each proxy object offers an interface for requesting the different services which the object has implemented. To be generic proxy objects supply a generic interface. This makes it easy for applications to request services independently of the proxy objects’ implementation. Additionally, applications have not to be especially customized to use a service offered by a proxy object. To show the different tasks and purposes of each proxy object four levels of interfaces for proxy objects have been defined. They mirror the class and type classification as seen above.

1. The root of this hierarchy is the generic proxy object interface which has to be implemented by each proxy object.

2. Underneath it, the class-dependent proxy object interfaces offers interfaces according to its class to influence the behaviour in a general way.

3. To gather type dependent operations the type-dependent proxy object interfaces are defined.

4. Finally, any proxy object implementation might offer an implementation-dependent proxy object interface. The operations contained in this interface are not restricted in any way. However, the applications must know the implementation if it wants to use these operations.

```cpp
//-------------------------------
// Generic Data
//-------------------------------
typedef string IPAddress;
struct StreamData {
    IPAddress sender;
    long port;
    IPAddress receiver;
    long receiverPort;
    long streamID;
};
//-------------------------------
// Generic Proxy Object
//-------------------------------
interface ProxyObject {
    readonly attribute string id; // proxy object identificator

    // reroute stream to new sink
    boolean reRouteStream(in StreamData streamData);

    // remove proxy object (does not care about stream
    // -> reroute stream before calling deleteProxy())
    oneway void deleteProxy();
};
//-------------------------------
// Class-dependent Proxy Objects
//-------------------------------
```
interface FilterProxyObject : ProxyObject {
    boolean filterOn(); // enable filter
    boolean filterOff(); // disable filter
    boolean increaseFiltering(); // reduce required bandwidth of data stream
    boolean decreaseFiltering(); // loose the degree of filtering
};

//========================================================================
// Type-dependent Proxy Objects
//========================================================================
interface MpegFilterProxyObject : FilterProxyObject {
    boolean dropBFrames(); // erase all B-Frames in MPEG-Stream
    boolean dropBandPFrames(); // erase all B- and P-Frames
    boolean dropColor(); // change colors to black and white
    boolean requantize(); // requantize resolution
};

//========================================================================
// Implementation-dependent Proxy Objects
 //========================================================================

Location of Proxy Objects

At last, the location of a proxy object has to be considered when installing a proxy. Determining the place of a proxy instance is a problem that has not been addressed very thoroughly in current proxy-based research projects. In most cases, the proxy instance is installed on a pre-determined machine in the client’s domain. However, this approach does not scale very well if an environment including WANs such as the Internet and world-wide mobile clients are used. In an idealized world the optimal location would be either on the server (for filters), or on the client (for caching), or partly on both ends (for compression). However, there are the following two restrictions that have to be considered:

1. The proxy object must be installed on a node where
   (a) the required resources (hardware, software, qualitative requirements) can be fulfilled and
   (b) the object may be installed.

2. The installation of one proxy object may result in the installation of another proxy object as it is the case by compression.

Due to complex network structures and different network domains, it is generally very complex to compute the optimal location for proxy installation. Therefore, a list of location preferences is attached to each proxy object. The list determines the preference sequence and may consist of the following five options (see figure 4.4):

1. The proxy should be located on the server.
2. The proxy should be located within the server’s domain (e.g., the same subnet).

3. The proxy should be located in the client’s domain.

4. The proxy should be located on the client.

5. The proxy should be located anywhere in the network.

![Diagram of proxy placement options](image)

Figure 4.4: Options of proxy placement

The first option in the list is the best choice for installing the proxy, the other follow according to their suitability. If an option is not contained in the list, this option is not permitted for proxy placement, since it might be senseless. A typical sequence for filter proxy objects looks like [1, 2, 3, 5].

After all proxy features are explained a proxy description in IDL for use by other services can be given:

```idl
typedef sequence<long> IntArray;

// Interface Definitions for ProxyDescription
// =================================================
interface ProxyDescription{
    attribute IntArray locationSequence;
    attribute long inStreamType;
    attribute long inSubType;
    attribute long inSubTypeEnhancement;
    attribute long outStreamType;
    attribute long outSubType;
    attribute long outSubTypeEnhancement;
    attribute boolean lossProneCompression;
    attribute Factory proxyFactory;
    string proxyShortDesc();
};
```

More details can be seen in appendix A.
4.3.3 Proxy Selection Process (PSP)

The proxy selection process monitors the QoS for each communication data stream (Seitz, Davies, Ebner & Friday 1998). If a decrease in QoS is recognized the PSP chooses one of the current data streams for proxy insertion dependent on user preferences and the provided stream classifications (see section 4.3.2).

User preferences User preferences may consist of

- a generic priority for different data streams to decide which stream should be modified if necessary,
- allowance of loss-prone compression techniques depending on content type,
- consider check of mobile system’s characteristics because this could also effect the QoS,
- decision between totally automatic, application-specific or user-specific proxy selection and placement mechanism.

QoS Monitoring To monitor each stream the communication protocols and the distributed applications provide QoS information. To decide if a proxy should be inserted the PSP requires not only the incoming QoS but also outgoing application related QoS characteristic of streams on the server side. Therefore, the CORBA extension on the server side regularly sends messages containing this information. The calculated difference between incoming and outgoing QoS controls insertion and modification of proxies oriented on defined thresholds. The installation process itself is described in section 4.3.4.

4.3.4 Proxy Installation Process (PIP)

After determining in the PSP (see section 4.3.3) that a proxy must be installed and what kind of proxy to install the installation has to be executed. Therefore, a hierarchy of instances was defined:

- **Proxy Objects** They implement the proxy functionality like filtering or caching. See section 4.3.2 for details.
- **CORBA Adapter** The CORBA adapter is the added CORBA code to an application to get access to the proxy system (Mowbray & Malveau 1997).
- **Proxy Factories** They manage a repository of several proxy objects and are able to install a proxy object according to the request of a client.
- **Proxy Trading Service** The Trading Service manages a list of proxy offers exported by the proxy factories. Hence, clients address the Trading Service supplying the kind of proxy objects they are looking for and receive a list of proxy factories offering that kind of proxy object (Mowbray & Malveau 1997).
4.3. Reactive and Adaptive Proxy Placement (RAPP)

To install the proxy object into a communication data stream it has to be determined where to place it and how to instantiate it. This questions and their solution are discussed in the following sections (Seitz, Davies, Ebner & Friday [1998]).

**Determination of proxy objects**

During the PSP, the client requests a proxy object from the proxy trading service to be installed for a given data stream. The proxy trading service gets the proxy service description which contains all parameters as described in section 4.3.2. All proxy factories have to export the service description to the proxy trading service for all the proxy objects they are responsible for. The descriptions and the corresponding proxy factory are stored in a proxy object repository by the proxy trading service.

![Proxy Trader Diagram](image)

Figure 4.5: Proxy Trader

After receiving a request the client’s trading service looks in his local repository for suitable proxy objects. To determine the suitability it looks for the functional requirements like stream types and the positional requirements. Additional to his own repository it queries the server’s trading service because it could offer suitable proxy objects, too. The client’s trading service merges his own results with the results of the server’s proxy service. If the resulting list is empty other proxy trading services have to be asked by the client’s trading service. The enquiry for a certain proxy service will thus lead to a list of proxy factories being returned by the proxy trading service. The returned list is ordered by the proxy trading service according to the suitability of the proxy factories compared to the location sequence of the proxies. The client can now choose a proxy factory from this list. Some proxy trading service interface definitions can be seen below. The whole definition is shown in appendix A.

```
typedef sequence<Factory> ProxyFactoryList;

// Interface Definitions for ProxyTrader

interface ProxyTrader{
    // The export procedure is used to export proxy offers to the
```
// proxy trader. If the export was successful, the procedure returns
// an identifier used for manipulating this object, otherwise it
// returns 0.
// ===================================================================
long export( in Factory proxyFactory,
in long proxyClass,
in ProxyDescription proxy );

// The withdraw procedure is used to withdraw an offer from
// the trading service. One has to specify the ID returned as
// the result of the export function. If the withdrawal was
// successful, the procedure returns TRUE otherwise FALSE.
// ===================================================================
boolean withdraw( in long proxyOfferID );

// To query the trading service, the query procedure is used.
// This procedure returns a list of handles to proxy factories
// offering the desired proxy objects. If this returned list is
// empty, no suitable proxy offer could be found.
// ===================================================================
ProxyFactoryList query( in long clientID,
in ProxyTrader serverProxyTrader,
in ProxyDescription proxy,
in long preference,
in long howMany );

All functions for using the proxy trading service are conform to the CORBA service trading service (Baker 1997, OMG 1997b, Mowbray & Ruh 1997). However, to keep the trading service simple, not all functions are implemented.

Installation of proxy objects

After a list of suitable proxy objects was determined by the trading service, as describe in the section before, the client chooses one proxy factory from the returned list. The client then requests the proxy factory to install the proxy object. As described on page (see section 4.2) new connections have to be established to and from the proxy.

It is possible to install several proxies into one stream. Therefore, proxy objects must eventually connect to another proxy instead of server or client. However, the proxy object should not care about this. For the use of, for example, compression techniques the installation of two proxies or more at the same time is necessary. Once a proxy is installed, it can be controlled by the CORBA control mechanism conceived for the RAPP architecture. Thus, the proxy provides general modification operations and specific operations for his kind. Once the filter is not required anymore, it can be uninstalled. Thereby, the predecessor of the proxy object has to reroute the stream to the successor of the proxy object.
4.4 Summary

The first section of this chapter introduced the proxy design pattern to give an overview about the proxy concept. This was detailed for using proxies in the network area. The next section provided an overview about existing proxy systems and their shortcomings. This justified the development of a generic and general proxy management system. The developed Reactive and Adaptive Proxy Placement (RAPP) architecture was described in the last section. Some implementation aspects and the implementation of a prototype to demonstrate its feasibility is described in the next chapter.
5. Implementation

This chapter deals with the implementation aspects of the Reactive and Adaptive Proxy Placement (RAPP) architecture as described in section 4.3. Furthermore, a prototype of RAPP was implemented. The main focus was set on the realization of CORBA adapters and the insertion/removal process of proxies into streams to demonstrate the feasibility.

Firstly, some application constraints are explained. Secondly, the proxy system elements will be discussed. Finally, the implemented proxy system prototype is introduced.

5.1 Application constraints

The use of the applications client and server together with the RAPP architecture postulates some constraints and normally modifications. Thereby, the use of an CORBA adapter (see section 4.3.4) for each application is necessary. The CORBA adapter is the application interface to the RAPP architecture. Therefore, the application constraints shall be described below.

The following concerns all types of applications as described in section 4.3. Which basic assumptions must the applications fulfill to allow the use of the RAPP architecture? The general assumption for the applications are the sending and receiving of a specified data and signaling stream. The constraints for the data communication concern the packet order during hand over and the not necessarily stateless data content of the data stream. Therefore, the following constraints apply:

Packet order If a connectionless oriented transport protocol is used the data stream must contain a mechanism to distinguish data packets. This is necessary to sort the received packets into the correct sequence.

Context Through the hand over of the data stream the proxy objects may have not enough context information for using data immediately. This happens, for example, with MPEG streams because a proxy object requires the sequence information to interpret data correctly. To solve this the proxy objects have to pass forward all packets without modifications until they have enough information for doing their work. This makes the server independent of the installed proxy object and gives more flexibility to the proxy object side. However, it might lead to temporarily unnecessary packet redirections to the proxy object. Nevertheless, it is possible to
wait at server side until a *stream specific point*, like the sequence start in MPEG streams, is reached but this means the server knows which proxy type is inserted. If, for example, a compressor/decompressor proxy object pair is inserted no delays are necessary.

**Application instantiation**  Instantiation of applications like client and server is out of scope of the RAPP architecture and this study, only instantiation of proxy objects and adapters is part of it. Thereby, no part of the specification directly deals with server and client application starting. There is only the constraint that some RAPP architecture methods require the server and/or client adapter (see section 5.2.2) reference. Thereby, the client and server CORBA adapter have to be instantiated depending highly on the adapter type (see section 5.2.2) if the application is instantiated. Furthermore, the corresponding QoS module must be instantiated and has to get the adapter reference to have access to the application. The client QoS module additionally requires the reference of its proxy trading service (see section 4.3.4) to request proxy objects. The server CORBA adapter reference has also to be available to the client QoS module to exchange signaling messages. The ways to get this references depend on the adapter implementation.

Instantiation of proxy objects is the task of proxy factories and therefore discussed in section 5.2.3.

### 5.2 RAPP Architecture Elements

The RAPP architecture contains some general elements which are independent of the used applications and, therefore, are generally valid.

As described in section 4.3 each application is equipped with a QoS Service module and a CORBA adapter. After this, the proxy object factories shall be described.

#### 5.2.1 QoS Service Module

The QoS service module on top of each adapter is generic and only accesses QoS data through the adapter. Therefore, providing the required QoS data is the task of the application CORBA adapter which is application specific.

The QoS service module provides QoS information and suggests and executes the insertion or removal of proxies and their parameters. This includes the report of QoS changes, especially QoS decrease, to the user. Therefore, the server QoS module which can be implemented inside of the adapter, sends QoS information to the client adapter. To execute the decisions the QoS module has the overview of the data stream and all inserted proxies. Consequently, the QoS module manages the data stream, too.

Currently, only the data rate as QoS information is used which can be seen in the next section.
5.2.2 CORBA Adapter

The task of the adapter is to provide the specified IDL interfaces (see below) for the RAPP architecture to access the programs.

To give the RAPP architecture access to a program like the application server and client (see below) and proxy objects (see section 4.3.2) the program has to provide a specific CORBA interface. The section above specified the assumptions for the use of the interface. To support the interface the programs have to provide a specific set of functionality. Therefore, a program may or may not require modifications to provide the interface. The implementation of this interface is called adapter because of adapting a program to a specific interface. By using CORBA the knowledge about the concrete realization of the adapter is not necessary. Therefore, only the interface itself, for example the proxy object interfaces (see section 4.3.2), is part of the RAPP specification. However, without programs which can provide the adapter the RAPP architecture becomes useless.

The application CORBA adapter interfaces shall be introduced, first. In the rest of this section the concrete realization of adapters for different programs shall be introduced wherefore structure design patterns and their realization are discussed. This gives an overview over the feasibility of adapting programs for RAPP.

CORBA Adapter Interfaces

To integrate a program with the RAPP architecture the program has to provide a specific CORBA interface. Therefore, the common interface aspects shall be described here. Comments to the realization of specific program adapters itself can be found in section below.

Each program type has to implement the corresponding basic program adapter type. Additionally, each program can implement further interface types. Server and client can provide extra functionality with this interfaces. Normally, only proxy objects implement additional interfaces to provide their special functionality on different ways. Since proxy object interfaces were described in 4.3.2 the following text only concerns adapters for server and client.

The interface for server and client defines methods for the possibility of rerouting the signaling and data stream to allow transparent proxy insertions, accessing the signaling communication and the QoS data information. If a program supports signaling functionality which overlaps with the RAPP architecture the signaling communication has to be captured by the CORBA adapter to check and perform consistency between RAPP and program. Checking the consistency can lead to modifications or suppression of the signaling communication. This concerns for example the data rate which has to be known. If a client application forces the server to send faster but the QoS is not sufficient the RAPP architecture has to prevent this. Therefore, a program interface provides a generic method to transmit signaling messages. This method uses named values to provide transmission of generic messages. Thereby, there is an agreement about the used names and their corresponding value is assumed to allow the use of it. This can be done program specific which leads to the dependence of the server and client program adapter.
Generic methods can be used for common tasks but are not defined at the moment. The defined interfaces are shown below:

module Applications {

    //========================================================================
    // Named Values and Arguments
    //========================================================================
    typedef string Identifier;
    typedef unsigned long Flags;

    interface NamedValue {
        readonly attribute Identifier name; // argument name
        readonly attribute any value; // argument
        readonly attribute Flags arg_modes; // argument mode flags
    };

    typedef NamedValue Argument;

    interface Arguments {
        typedef sequence<Argument> Args;
        typedef sequence<Identifier> Identifiers;

        oneway void put(in Argument argument); // insert argument
        Argument get(in Identifier name); // get argument with identifier
        oneway void remove(in Argument argument); // remove argument
        Args arguments(); // returns all arguments
        Identifiers getIdentifiers(); // returns all identifiers
        long size(); // number of arguments
    };

    //========================================================================
    // Stream Data (same as in proxy object interface)
    //========================================================================
    struct StreamData {
        IPAddress sender;
        long port;
        IPAddress receiver;
        long receiverPort;
        long streamID;
    };

    //========================================================================
    // Basic Application Interface
    //========================================================================
    interface Application {
        readonly attribute string id; // application identifier
        oneway void signalingMessage(Arguments arguments);
    }

    interface Server : Application {

Another problem is the distribution of the CORBA adapter reference. This point is important for the proxy factories and thereby discussed in section 5.2.3 and for client and server which is discussed on page (see section 5.1).

However, what happens if programs do not offer an interface which may simply be converted to another interface and if the required functionality is not given? Then the interface and functionality has to be added in any way. A few possibilities of realization shall be explained in the following text wherefore structure design patterns are discussed.

Structure Design Patterns

The adapters implement the adapter (or wrapper) design pattern (Gamma, Helm, Johnson & Vlissides 1995, Mowbray & Malveau 1997). The adapter or wrapper design pattern is described as follows:

> Convert the interface of a class into another interface clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces (Gamma, Helm, Johnson & Vlissides 1995, p. 139).

This description is based on an object-oriented paradigm but it is also usable to add an CORBA IDL interface to an existing program which operates as a front end for it. “The wrapper communicates to the legacy system using its native communication facilities” (Mowbray & Malveau 1997). This is mostly used to adapt legacy systems to the object-oriented paradigm or for remote access of programs like it is necessary for the RAPP architecture (Mowbray & Malveau 1997). Therefore, the purpose of the adapter or wrapper design pattern is more generically described as

> convert a given interface into another given interface (Tichy 1998).

Since the assumption that all functionality is given by the program cannot be upheld the adapter may have to implement additional functionality. Beside the benefit of a CORBA IDL interface the adapter allows the existing functionality of the program to be unchanged. However, it requires detailed knowledge about the program and no general solution is possible. Additional, performance problems are possible. Despite the above wrapper definition, the following text uses the name transducer for it. The
name \textit{wrapper} is used as in [Nwana 1996] where an object wrapper is integrated in the program as a special module (object) which has direct control of data structures.

An adapter may be realized by using the proxy pattern (see section 4.1). However, in contradiction to a proxy “An adapter is meant to change the interface of an \textit{existing} object.” (Gamma, Helm, Johnson & Vlissides 1995, p. 149). This leads to the important fact that \textit{existing} programs are used and have to be \textit{adapted} to provide another interface which results in mostly additional or withdrawn functionality. Thereby, the use of the proxy and adapter pattern together is possible to provide a proxy with another interface as the original for which the proxy was designed.

Another design pattern called \textit{bridge} also addresses the use of a different interface for a given object. However, “The Adapter pattern makes things work after they’re designed; Bridge makes them work before they are” (Gamma, Helm, Johnson & Vlissides 1995, p. 219). The RAPP architecture has no influence on the interfaces and functionality of given programs. Therefore, the bridge pattern is not usable for existing programs.

\textbf{Discussion of Realization}

Because of the nature of different programs the required functionality is not always given and the kind of access of the provided functions and information can differ. Therefore, the adapters have to be program-specifically implemented to call functions and to get and interpret the information. There are several ways of program adaptation to provide the CORBA interface for the RAPP architecture. Based on the former discussion of design patterns a few solutions for program adaptation are named below and are illustrated in figure 5.1(b).

1. Requirement: program modifications are possible
   
   - \textbf{Rewriting}
     Integration of the CORBA interface into the program by rewriting the whole program.
   
   - \textbf{Object Wrapper}
     Add an object wrapper to the program. A wrapper is an afterwards added module and has direct control over the data structures and can therefore modify them.
5.2. RAPP Architecture Elements

- **RAPP Library**
  Supporting library methods which programs have to use. The library supports all network access methods and instantiates the CORBA interface itself.

- **Standard Library**
  Modifying the standard library methods to realize RAPP functionality. Therefore, network access methods have internally to be rewritten and the CORBA interface has to be instantiated.

2. Requirement: program modifications are not possible or not wanted

- **Surrogate**
  A surrogate is a separate piece of software which realizes the CORBA interface. The surrogate catches all program communication and forwards it depending on the RAPP signaling.

- **Transducer**
  A transducer is a separate piece of software \cite{Nwana1996} which realizes the CORBA interface and bridges this to the program with its native communication protocol.

The big arrows in figure 5.1 show the program communication and the small arrows the RAPP architecture specific communication. Figure 4.2 only shows functionality of RAPP and no implementation details which depends on used programs. Therefore, adapters may receive the data stream to redirect it.

What are the differences between the approaches? As we could see the realization first depends on the possibility to modify the program. The further choice depends on the supported functionality. Nevertheless, performance aspects are important, too, which could be a problem by the surrogate approach. This can be seen in table 5.1 and is detailed below.

<table>
<thead>
<tr>
<th>Names</th>
<th>Pattern</th>
<th>Program modification</th>
<th>Existing program functionality</th>
<th>Complexity</th>
<th>Maintenance</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>Module</td>
<td>Internal</td>
<td>—</td>
<td>not full</td>
<td>Low</td>
<td>Average</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Wrapper</td>
<td>Interface</td>
<td>—</td>
<td>full</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Surrogate</td>
<td>Adapter, Proxy</td>
<td>not possible</td>
<td>not full</td>
<td>—</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Transducer</td>
<td>Adapter, Proxy</td>
<td>not possible</td>
<td>full</td>
<td>—</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Transducer**  A transducer can be used if the program supports external control to redirect data streams and get required QoS information. The transducer can be implemented externally and the external program interface can be used. This is an easy case but requires one transducer implementation for each program. If the program interface changes the transducer has to be changed, too. Another problem is that it is not supported by the most programs until now.
The complexity should be low because there is only a conversion necessary between the CORBA interface and program interface. Therefore, the maintenance is also low because only changes in the program interface effects the transducer. The performance is only effected by the signaling transmission between transducer and program.

**Surrogate** If no program changes are possible or wanted and using a transducer is not possible the data stream could be forwarded to an autonomous process which can reroute the data stream again. The QoS information must then be calculated on the basis of the data stream or another source has to be found. This is a difficult case but it should be possible to use it for many (or all) programs.

The complexity is high because of the data stream handling. The maintenance is low because the surrogate should be mostly independent of the program which depends on using program specific features. However, the additional data stream handling with its time constraints might lead to performance problems.

**Wrapper** A *wrapper* can be used if all required functionality can be accessed by no or only small changes in the existing code. Additionally, if not supported functionality can be added easily a wrapper should also be possible to use. Therefore, only a new module has to be added which uses existing functionality and may provide new functionality to realize the required CORBA interface.

The wrapper approach is usable if most functionality is supported by the program and, therefore, is not a complex task to realize. However, it highly depends on internal functionality which makes maintenance difficult. Additionally, each modification inside the wrapper leads to a new linking of it to the program and in worst case the program has to be changed, too. By using internal functionality and direct access the performance should be no problem.

**Library** The CORBA adapter can be directly implemented into the program if it is possible to extend the program. How to implement this adapter depends on each program and used programming language. If an own library is provided then all method calls have to be changed to the new methods. However, this is not always possible to the internal structure, for example, if there is no real differentiation between signal and data stream methods. In this case the standard library has to be changed. This is an easy case because the QoS information and the data communication can be accessed directly inside the library. However, the program has to be changed.

An own library is easy to implement wherefore the complexity is low. The modification of a standard library depends on the source code and therefore, existing code has to be changed what increases complexity. The maintenance is average because program changes does not effect the library if all assumptions are still valid but the new library may have to be linked to the program. However, a standard library approach depends on existing code and its availability and therefore, increases the cost for maintenance. The performance should be no problem because of direct access to all required data.
The transducer approach works fine if a program is used which is implemented for remote control and supports the required features which is not a common kind of program. The surrogate approach is very important through the fact that commercial programs like browsers cannot be modified. A wrapper can be used if most features are supported by program and the other features can easily be added but the remote control feature is not provided. The library approach gives the possibility to provide a general interface which can be used by building new programs.

5.2.3 Factories

Factories are used to hide concrete object instantiation which permits using different implementations. The abstract factory software design pattern for object oriented concepts is defined as

Provide an interface for creating families of related or dependent objects without specifying their concrete classes (Gamma, Helm, Johnson & Vlissides 1995).

Furthermore, the factory design pattern is defined as

Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory methods let class defer instantiation to subclasses.

This makes instantiation independent of classes, isolates concrete classes, makes exchanges easier and gives better consistency.

Therefore, the factory concept is used to hide the specifically used proxy object from the requested proxy object description. Each proxy object which is suitable can be chosen for instantiation by the factory. Clients request a proxy object and get the object reference of the proxy object back to make further calls on it. Therefore, the factory provides a requestProxy method which requires the stream data as defined in section 4.3.2 to set initial values of the adapter and program. This values can be given to the adapter on an adapter dependent way which is described below. This makes getting the initial values, the location of each object and the chosen proxy object independent of each other. For management purposes the factories store the reference to each created proxy object and the proxy objects have to unregister with the factory if they remove themselves.

Each factory knows its proxy objects and which services they provide. Thereby, it must have knowledge about the provided interfaces. Furthermore, the proxy object description (see section 4.3.2) is offered to the trader system (see section 4.3.4).

To start a proxy object, to exchange initial values and to get the proxy object reference is an important point for factories. Furthermore, the location of the factory is not determined. This points shall now be investigated.

```java
interface Factory {
```
typedef sequence<string> ProxyObjectTypes;

readonly attribute string id;
readonly attribute ProxyObjectTypes types;

ProxyObject installProxy(
    in proxyTradingService::ProxyDescription proxyDescription,
    inout StreamData streamData);

oneway void deleteFactory();

// getting data for an library adapter
// arguments are adapter specific data
void getInitialData(out string id, out StreamData streamData,
    Arguments arguments);
};

Creating Proxy Objects  The possibilities of creating proxy objects which consist of the adapter and program depend on the adapter type. Thereby, the factory has to use different methods. The creation of adapter and/or program can be divided into process starting or instantiation and starting as thread requiring the same programming language for factory and adapter or program. The further text uses the word starting for process starting and instantiation for instantiation and starting as thread. Instantiation is useful if an interpreted language is used. Hereby, only one interpreter must be started. However, instantiation always requires the availability of it at compile time of the factory.

In case of a transducer or surrogate the factory can start or instantiate adapter and program (see 1a. in figure 5.2) or only the adapter if the adapter starts or instantiates the program (see 1b. in figure 5.2). The program should not start or instantiate the adapter because it should not be aware of the adapter. By using a transducer or surrogate the normal case should be program starting because the main reason for using this adapter type is the fact that the program code is not modifiable which forbids instantiation.

It is recommended to choose an approach which lets the factory only interact with the adapter and not with the program. This keeps all knowledge about the concrete program inside the adapter which leads to a better encapsulation.

In case of a wrapper or library the adapter is integrated into the program which requires only starting one process by the factory (see 2a. in figure 5.2). If the program is written in the same language as the factory and a wrapper is used as adapter the adapter can be instantiated which instantiates the program (see 2b. in figure 5.2). In case of a library the adapter is instantiated by using the library methods by the program.

CORBA Reference and Initial Values  Distributing initial values like stream data, adapter identification and factory reference to the adapter and getting the adapter’s CORBA reference to the factory depends on the used starting method as described above.
If the adapter or program is instantiated by the factory the values can be given with the constructor and if it is started as a process the values have to be given as command line arguments.

In case of a standard library adapter it is not possible to use this two approaches because the program is not aware of the adapter. Therefore, a standardized way is necessary to get the factory reference to the adapter. An easy solution is the use of a persistent factory or adapter reference which never changes. The adapter or factory have to provide a method to exchange initial values. The use of persistent references limits the flexibility but is not unusual. Another solution is the use of a well known IP address and port for the use of sockets. Furthermore, a well known file can be used to get the factory reference and all other informations. However, how should the factory know which adapter is responding? The factory could block until it gets response or the file is modified and contains the adapter reference but this decreases the performance of the factory.

The CORBA adapter reference has to be known by the factory and client to permit method invocations for controlling. Therefore, the adapter has to register its CORBA reference with the factory which can be done with the register method. This allows the factory to give the reference to the client, too.

During the registration process the requestProxy method remains invoked wherefore a concurrency model for the factory is necessary which allows invocation of at least two methods by different clients at the same time. A discussion of concurrency models for CORBA can be found in (Laukien, Seimet, Newhook & Spruiell 1998).
**Location**  The RAPP architecture says nothing about the concrete location of a proxy factory. Therefore, it is possible to use one factory for each host (see 1. in figure 5.3) or several hosts or even a domain (see 2. in figure 5.3). If the factory is competent for more than one host then it needs a helper (daemon) on the other hosts to instantiate objects what makes the use of *internal* factories necessary. Furthermore, an algorithm is necessary for deciding which internal factory is used. The use of one factory for several hosts is useful if for example security, accounting or performance are important. However, each host requires a factory and each additional factory for a hierarchical organization does not effect the above explained problems and solutions.

![Diagram of factory location](image)

**Figure 5.3: Factory location**

In this section implementation aspects of the general RAPP architecture elements have been discussed to explain design decisions in the implemented prototype which is shown in the next section.

### 5.3 Prototype

A prototype was developed to demonstrate the feasibility of the proxy concept as described in chapter 4 and above. The prototype shows the insertion/removal of a filter for an MPEG-1 video stream. Thereby, all signal messages are implemented with CORBA (see chapter 3). The implemented elements and the used programs and their hierarchic order is summarized in figure 5.5.

![Diagram of prototype](image)

**Figure 5.4: Legend for all following pictures**

Firstly, a description of the existing programs is given. Secondly, the RAPP architecture elements and some additional elements are described in detail. Finally, the cooperation between the elements is shown to explain the functionality.

### 5.3.1 Existing programs

The existing programs for the filtering of a MPEG-1 video stream consist of the parts server, client, filter and the corresponding daemons for server and filter or
Graphical Users Interfaces (GUI) for client and filter. Both GUIs are written in TCL/TK and server, client and filter are written in C and taken from the work of N. Yeadon for his PhD thesis about “Quality of Service Filtering for Multimedia Communications” (Yeadon 1996). For this work, server and client are adapted from the MPEG tools of the Berkeley Multimedia Research Center (BMRC) (BMR 1998). The filter program has been implemented by N. Yeadon himself. For demonstration purposes the original filter can only be inserted \textit{statically} into the MPEG-1 video stream. Therefore, the filter does an own connection request to the preceding server or filter which allows cascades of filters. The stream establishment can be seen in figure 5.6.

The transport protocol for the MPEG-1 video stream and the control signals is the User Datagram Protocol (UDP) (Postel 1980, Stevens 1990).

### 5.3.2 RAPP Architecture Elements

To implement the RAPP architecture the following elements were necessary. Firstly, the existing programs were extended with a CORBA adapter which also includes the QoS module. Furthermore, the factories for the instantiation of proxy objects, the proxy trader to register and to query proxy objects and a infrastructure consisting of the Graphical User Interfaces (GUI) to demonstrate the RAPP architecture are implemented.
CORBA adapters The prototype uses the transducer adapter type for all three programs. This was chosen because all necessary information can be accessed by catching the signaling communication and remote control for filter operations was implemented. Furthermore, it leads to language independence and this makes the use of Java and a Java ORB for the adapter possible. CORBA with Java or vice versa is explained in [Vogel & Duddy 1997]. The chosen Java ORB “OmniBroker” [Laukien & Seimet 1997] was used in another diploma thesis which we want to use [Wolf 1998]. A wrapper adapter for the C programs with the Java native language interface to C was not chosen because it had too many drawbacks. The drawbacks consists of splitting methods or using global values to get the information and of too complicated memory management between the two languages. An adapter in C or C++ as additional library or direct in the program was not chosen because of the difficult thread and memory handling under C++ in comparison to Java and the difficult use of CORBA under C. Additionally, the programmer had more experience with Java and prefers it because of the independence of low level programming problems.

The implemented adapters have many similarities wherefore a basic adapter class was defined. To extend this basic class the CORBA part has to be implemented with the TIE-concept which uses delegation to allow extension (Baker 1997). Some methods of the basic adapter class are shown below.

```java
// ************************************************************************
// interface classes to work together with factory
// ************************************************************************
public ProxyObject getCorbaObject(); // get CORBA reference of adapter

// start adapter after instantiation
abstract public Arguments startAdapter(FactoryImpl factory, String id);

// ************************************************************************
// IDL methods
// ************************************************************************
public String id(); // adapter ID

// send reroute stream to predecessor
public boolean reRouteProxy(ProxyObject predecessor, StreamData streamData);

// reroute data stream to new sink
public boolean reRouteStream(StreamData streamData);

public void deleteProxy(); // remove proxy

// get control message
public void controlMessage(Arguments arguments);

// ************************************************************************
// helper methods for subclasses
// ************************************************************************

// more reliable socket creation
DatagramSocket createSocket() throws SocketException;
```
The reason why not one adapter for all programs is possible lays in the fact that the programs are built for a static behaviour as shown in the last section. The programs send or receive connection requests not in the same way wherefore the special handling is necessary.

To fulfill the RAPP architecture constraints the program had to be changed a little bit. The reroute mechanism was internally added by the filter and server. Therefore, a new command line argument for the signaling host and the handling of the stream reroute request was added. It was necessary to split the signaling and data host information into own variables because they are now not necessarily the same. Except of the context constraint the server reroutes the data stream not immediately. The server reroutes when coming to a packet with a sequence header in it. Therefore, the filter program had only to be changed a little bit. To reroute the stream and wait for the right packet the new clients address and port is temporarily stored in a global value and substituted in the used variables if the right packet is reached. For the packet order constraint no changes were necessary because the packets contain time stamps.

The program code for changing the servers sink is shown below. The variable corbaChangeSink indicates the claim of a change to another sink. If this is true and a sequence header has to be transmitted the new address from variable corbaNewAddress is copied into the internal used address variable ctx->addr from type struct sockaddr * which is accessible under the format struct sockaddr_in * by using the variable dummy. Both types are part of the system network API (Stevens [1990]). After changing the network address the watch dog variable has to be reset and the data has to be sent to the new sink.

```c
#ifdef CORBA
    /* ask watch dog and pdu header */
    if(corbaChangeSink && (ntohs(pdu->header)== seqHeader))
    {
        /* copy new address into ctx structure */
        dummy = (struct sockaddr_in *) ctx->addr;
        dummy->sin_port = corbaNewAddress.sin_port;
        bcopy((char *)&corbaNewAddress.sin_addr, (char *)&dummy->sin_addr,
              corbaAddressLength);
        /* reset watch dog for changing sink */
        corbaChangeSink=0;
    }
#endif
    return sendto(ctx->sock, start, len, 0, ctx->addr, *ctx->addrlen);
```
Additional changes were necessary by using the adapters to start the programs. This only concerns the command line arguments evaluation and has little effect on the program. The existing daemons were not used because the factories made them obsolete.

All changes are inside of `#ifdef CORBA .. #else .. #endif` statements and thereby the changes can be switched on and off during compilation time.

**Factory**  
The factory is implemented using Java and thereby the adapters can be instantiated directly as Java objects. Because of the easy use and for a better test environment the factory can instantiate the player and server, too. It is not foreseen to use other adapter types but it is not difficult to add these. Each participating host is equipped with one factory which is sufficient for demonstration purposes. Hereby, no additional helper processes on the hosts are necessary.

**QoS Service Module**  
The QoS service module is integrated into the server and client adapter. The server QoS module regularly sends the actual sending rate to the client adapter. The client QoS module gets the actual sending rate from the server adapter and the actual receiving rate from the underlying program. The values are regularly checked for QoS violation which may lead to a proxy object insertion. Additionally, the client QoS module manages data about all participants of a data stream involving server, client and all inserted proxy objects. This offers easy access for controlling and monitoring.

### 5.3.3 Additional Elements

In addition to the above described elements the CORBA service *naming service* and a *Proxy System Control GUI* was implemented which are described below.

**Naming Service**  
As a central point for the distribution of CORBA references the CORBA *naming service* was chosen. Each factory and trader has to register themselves with the naming service. Therefore, a factory can get the trader reference to offer proxy objects (see section [4.3.4](#)). The references have to be registered with a corresponding name which identifies the reference (see section [3.2](#)). The names consist of naming contexts which build a path to the reference like in a file system. The used path names have to be defined in advance to be used by the factories and trader. At the moment only the trader reference for the factories is defined. However, as we can see in section [5.3.4](#) the factory registration is necessary for testing.

For better usage of the CORBA naming service a separate naming class was written which makes using the CORBA naming service easier by handling common cases and providing easier naming facilities. Names are defined as strings which are build like a file path in file systems and not as an array of naming context.
Proxy System Control GUI  For controlling and monitoring the whole system the Proxy System Control GUI (PSC GUI) was implemented. The GUI allows the establishment of a data stream by starting client and server and the insertion and removal of filter proxies into the stream. Furthermore, parameters like increase filtering can be set by proxy objects. It is possible to handle more than one stream with the GUI.

To get user inputs, like the video name or filter parameter, the Tool Command Language/TK Toolkit GUIs (TCL/TK GUIs)\(^1\) of N. Yeadon are used. This has the advantage to use familiar programs. Slight changes in the GUI and additional shell scripts were necessary to use the programs in conjunction with the PSC GUI. The problems mainly consist of calling TCL/Tk scripts from a java program which requires the setting of the display environment because using a TCL/Tk GUI from a Java program is not supported as far as the programmer knows.

5.3.4 Functionality

The RAPP architecture elements described above must interact together to realize the whole RAPP architecture. Therefore, the interaction shall be shown by three examples. Firstly, the whole RAPP architecture has to be booted. Secondly, the stream establishment is shown which is not part of the defined RAPP architecture but is necessary to build an easy to use demonstration prototype. Thirdly, the handling of proxy objects is illustrated.

Booting  To boot the RAPP architecture the proxy system name service has to be started first. The proxy name service starts the CORBA name service and opens a well known UDP port at a well known IP address. Secondly, the trader and after it the factories must be started. Each factory and trader can now get the name service reference by sending a UDP packet to the well known port. The proxy name services responses with a UDP packet containing the reference. Now, factories and trader can register itself with the name service and factories can get the trader reference to offer proxy objects. That is all for the RAPP architecture but to control the elements for a demonstration the PSC GUI is necessary, too. The PSC GUI gets all trader and factory references from the name service. This all can be seen in figure 5.7.

The socket approach was chosen because of the independence of other systems like a web server and because only one name service instance has to be instantiated. By using files the distribution over locations which have independent file systems requires more than one name service which results in unnecessary complexity for a prototype.

By using a well known file the location of the name service itself is independent but not the location of the well known file to find the name service reference. Another approach would be the use of a Uniform Resource Locator (URL)\(^2\) to get access to the well known file. This makes the use of a web server necessary, for example. However, using a web server leads to another system on which the prototype depends. The prototype should be independent of

\(^1\) Ousterhout, J. 1994

\(^2\) Berners-Lee, T., Masinter, L. & McCahill, M. 1994
Figure 5.7: Booting of the proxy system

A standardized bootstrap protocol has until now not been defined but the OMG got
submissions for an improved bootstrapping service for CORBA [Henning & Neville
1998, Spirn, Frantz, Mischkinsky, High, Stewart & Cox 1998, Chapman & Stringer
1998].

Stream establishment The stream establishment (see figure 5.8) is invoked by
the user input in the PSC GUI which contains all data for the establishment. The
PSC GUI requests a client adapter from a factory. The client adapter starts the
client program and requests a server adapter using a factory. The server adapter
starts the server program, too. Now, the data stream is established.

Figure 5.8: Stream Establishment with RAPP Architecture

Proxy handling The proxy handling consists of the insertion, setting of param-
eters and removal of proxy objects. All these methods are implemented in the QoS
service module which shall be explained in detail, now.

Insertion For the insertion of a proxy object (see figure 5.9) the proxy object type,
the factory and the predecessor object must be determined. Then, the insert proxy
object method is called which requests a proxy object from the factory and reroutes the data stream to the new proxy object. Through the use of UDP the destination of the packets must be changed only and no further connection management is necessary.

Figure 5.9: Insertion of a Proxy Object

**Removal** The removal of a proxy object (see figure 5.10) is similar to the insertion. The data stream has to be rerouted by the predecessor to the successor of the proxy object only. Hence, the predecessor gets a reroute directive. If the packets arrive at the successor the proxy object gets a remove request and it will terminate itself if it gets no further data packets. Therefore, the client sends a data stream specific timeout time with the remove order. Thereby, the proxy object waits the timeout time for a packet and terminates if the time is reached.

Figure 5.10: Removal of a Proxy Object

**Parameter setting** Parameter setting of a proxy object (see figure 5.11) is easy when using CORBA. The QoS module simply sends a CORBA call with the new parameters to the adapter. The only problem is to decide which parameter with which value to set. However, this is a problem of the QoS module.

**5.4 Summary**

Firstly, constraints for the used program and data stream has been given. In the second section the implementation aspects of the RAPP elements was discussed.
leading to the definition of application IDL interfaces. Therefore, some possible adapter types were introduced to explain program adaptation techniques. Finally, the implemented prototype was introduced which realizes the basic RAPP architecture functionality. As demonstration program filtering of an MPEG video stream was chosen which provides a demonstrative representation.
6. Evaluation

The feasibility of the developed RAPP architecture (see section 4.3) was shown by implementing a prototype (see section 5.3). To evaluate this prototype the packet intervals during proxy insertion and removal have been measured since it can be seen as a highly critical part of RAPP. If the performance during proxy insertion and removal is not satisfactory the whole RAPP project has to be redesigned. The usability of filtering video streams is not discussed here but can be seen in (Yeadon 1996, Yeadon, Davies, Friday & Blair 1998).

Firstly, the evaluation environment is described which determines the structure of the following evaluation scenarios. Secondly, proxy insertion and removal are evaluated. Finally, the results are summarized and conclusions are given.

6.1 Evaluation Environment

To describe the evaluation environment the four parts used MPEG movies, computers and network, RAPP configuration and scenarios have to be considered which is detailed below.

Movies As movies rocket.mpg and chicken.mpg (Sun 1998) were chosen. The first is a real-life movie of a starting rocket recorded from a close distance and the second is a computer animation of a cock and a chicken playing with their baby. As we can see in table 6.1 the rocket movie only contains I-Frames and the chicken movie I-, B- and P-Frames. Because of the small number of 51 frames the whole data stream of the movie rocket.mpg fits into the graph in figures 6.2 and 6.5.

The two movies have been chosen because of their different size and their short duration which is necessary for easy testing because proxy insertion is only possible if a sequence header is reached. Furthermore, a movie containing only I-Frames provides easier evaluation because of the small fluctuations in packet size in comparison to a movie with I-, B- and P-Frames.

Computers and network The used computers and their characteristics are listed in table 6.2. As we can see, different architectures and operating systems have been used and the connected network technology was always ethernet. The two computers in Lancaster are connected to different subnetworks and the Internet connection is realized with ATM. The computer moon in Karlsruhe has to use several ethernet subnetworks and finally FDDI to connect to the Internet.
6. Evaluation

Table 6.1: Movie Data

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>rocket.mpg</th>
<th>chicken.mpg</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size (Byte)</td>
<td>104988</td>
<td>404606</td>
</tr>
<tr>
<td>No. of Frames (Byte)</td>
<td>51</td>
<td>169</td>
</tr>
<tr>
<td>Max I-Frame (Byte)</td>
<td>2682</td>
<td>9086</td>
</tr>
<tr>
<td>Avg I-Frame (Byte)</td>
<td>2070</td>
<td>8658</td>
</tr>
<tr>
<td>Min I-Frame (Byte)</td>
<td>1594</td>
<td>6967</td>
</tr>
<tr>
<td>Max B-Frame (Byte)</td>
<td>—</td>
<td>5448</td>
</tr>
<tr>
<td>Avg B-Frame (Byte)</td>
<td>—</td>
<td>1987</td>
</tr>
<tr>
<td>Min B-Frame (Byte)</td>
<td>—</td>
<td>5448</td>
</tr>
<tr>
<td>Max P-Frame (Byte)</td>
<td>—</td>
<td>1911</td>
</tr>
<tr>
<td>Avg P-Frame (Byte)</td>
<td>—</td>
<td>875</td>
</tr>
<tr>
<td>Min P-Frame (Byte)</td>
<td>—</td>
<td>342</td>
</tr>
<tr>
<td>Image Size</td>
<td>160×120</td>
<td>320×256</td>
</tr>
</tbody>
</table>

Table 6.2: Computers for Evaluation

<table>
<thead>
<tr>
<th>Name</th>
<th>Architecture</th>
<th>Operating System</th>
<th>Memory</th>
<th>Network card</th>
<th>IP address</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>egc006000004</td>
<td>Pentium-MMX 200Mhz</td>
<td>Linux 2.0.30</td>
<td>64MB</td>
<td>Ethernet</td>
<td>148.88.16.113</td>
<td>Lancaster</td>
</tr>
<tr>
<td>athens</td>
<td>2×Pentium-Pro 200Mhz</td>
<td>Linux 2.0.29</td>
<td>64 MB</td>
<td>Ethernet</td>
<td>194.80.35.25</td>
<td>Lancaster</td>
</tr>
<tr>
<td>moon</td>
<td>Sun Ultra1 140Mhz</td>
<td>Solaris 2.6</td>
<td>128 MB</td>
<td>Ethernet</td>
<td>129.13.3.122</td>
<td>Karlsruhe</td>
</tr>
</tbody>
</table>
The network performance during evaluation time was tested by using the Internet Control Message Protocol (ICMP) tool ping which sends echo packets to another host that sends them back to the sender. The round trip delay of sent packets can be seen in table [6.3]. To get the values ping sent 20 packets of size 1k. Interesting

<table>
<thead>
<tr>
<th>Computer</th>
<th>egc0060000004</th>
<th>athens</th>
<th>moon</th>
</tr>
</thead>
<tbody>
<tr>
<td>egc0060000004</td>
<td>—</td>
<td>9.9/10.7/12.3 ms</td>
<td>40/47/51 ms</td>
</tr>
<tr>
<td>athens</td>
<td>8.6/9.4/10.5 ms</td>
<td>—</td>
<td>50.6/52.4/58.4 ms</td>
</tr>
<tr>
<td>moon</td>
<td>46/47/49 ms</td>
<td>47/48/52 ms</td>
<td>—</td>
</tr>
</tbody>
</table>

is the fact that the delay between moon and each computer in Lancaster is not the same at all during the evaluation time.

RAPP configuration The RAPP elements have been distributed to the computers as shown in table [6.4]. The whole evaluation was controlled on computer egc0060000004 wherefore the components name service, player and proxy system control were started on it.

<table>
<thead>
<tr>
<th>Computer</th>
<th>egc0060000004</th>
<th>athens</th>
<th>moon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Player, NameService, PSC</td>
<td>Server, Trader</td>
<td>Filter</td>
</tr>
</tbody>
</table>

Scenarios To evaluate the prototype the chosen movies have been sent with a frame rate of 10Fps(FramePerSeconds) and additionally, the chicken.mpg movie with a frame rate of 30Fps to have an example with advanced requirements to the underlying system because of faster sending rate and higher data volume. Therefore, the proxy insertion/removal evaluation in the following sections uses three graphs which represent the three scenarios. As we saw in the RAPP configuration, server and player were in the same LAN wherefore no outside connection was necessary. To have meaningful evaluation results the filter was inserted on moon wherefore the stream had to be rerouted to Karlsruhe. The filter performs no filtering wherefore the required bandwidth does not change. The filter was inserted and removed by user control and an automatic mechanism was not chosen because the insertion/removal request should be randomly selected. It can be expected that the long distance between Lancaster and Karlsruhe (see figure [6.1]) leads to a result which makes evaluation easier. However, if the performance is absolutely unsatisfactory another example should be chosen.

The results of this experiment are detailed in the following sections wherefore proxy insertion and removal are separated.
6.2 Proxy Insertion Evaluation

What can we expect if a data stream is interrupted and rerouted to another host especially if the host is further away than the client host before? Due to the longer distance and processing time in the proxy object it can be expected that

- the error rate increases and
- the packet delay at client side increases.

Therefore, the client could temporarily stop the movie because packets will come later than required.

The executed tests can be shown in the figures 6.2, 6.3 and 6.4.

![Packet Interval during Proxy Insertion](image)

**Performance before Proxy insertion** At the beginning of figure 6.2 we can see the fluctuations of the last proxy insertion and how the client packet receiving interval converges to the server packet sending interval. The fluctuations in figures 6.3 and 6.4 depend on the packet size which highly fluctuate because of the different
6.2. Proxy Insertion Evaluation

Figure 6.3: Proxy insertion, chicken.mpg with rate 10 Fps

Figure 6.4: Proxy insertion, chicken.mpg with rate 30 Fps
size of I-, B- and P-Frame packets as shown in table 6.1. The outlier of packet 8 is no exception because its a group header packet of size 20 byte.

**Performance during Proxy insertion**  During proxy insertion at packet number 27 we can see a high peak in all three graphs which reflects the longer distance over the installed proxy. The peak in the graphs to chicken.mpg does not seem to be so grave but the small packet size of a P-Frame has to be considered.

**Performance after Proxy insertion**  After proxy insertion the client intervals converge back again to the server intervals. By measuring also at the proxy side the side of the disruptions can now be identified wherefore disruptions can be divided into the two transmissions server-proxy or proxy-client. A good example is figure 6.3 around packet number 37.

Another point is the changed error rate which depends on the movie. In rocket.mpg the error rate fluctuates between 0 – 10% and after proxy insertion it increases shortly to 30 – 40%.

In the next section the proxy removal is evaluated.

### 6.3 Proxy Removal Evaluation

What can we expect if a proxy is removed? Due to the shorter distance it can be expected

- the packet receiving interval gets smaller for a short spell and therefore
- packets will outrun their predecessor packets.

If the client does not cache packets received early they unnecessarily get lost and increase the error rate.

The executed tests can be shown in the figures 6.5, 6.6 and 6.7

**Performance before Proxy removal**  The longer distance because of the inserted proxy leads to higher fluctuations in packet intervals but the convergence to the server intervals is identified.

**Performance during Proxy removal**  During proxy removal the first packets are received earlier wherefore the interval decreases which can be clearly seen in figure 6.5 at packet 28. That is the reverse behaviour compared to proxy insertion.

**Performance after Proxy removal**  After proxy removal the client packet intervals converges very fast to the server packet intervals because of the better connection. Despite, in figure 6.7 the fluctuations become smaller but not as in the other two pictures which can be explained with the three times higher data rate.
Figure 6.5: Proxy removal, movie rocket.mpg

Figure 6.6: Proxy removal, chicken.mpg with rate 10 Fps

Figure 6.7: Proxy removal, chicken.mpg with rate 30 Fps
6.4 Summary and Conclusion

In this chapter the test environment was explained and the implemented prototype was evaluated by measuring the proxy insertion and removal process. The proxy insertion and removal process was evaluated wherefore we can summarize the following facts:

- proxy insertion/removal is highly dependent on the affected nodes in the data stream,
- proxy insertion leads to temporarily higher packet intervals and error rates,
- after proxy insertion the packet intervals mostly converges back to the sender packet intervals and the error rate decreases, too,
- proxy removal leads to packets outrunning preceding packets.

The constraints on the used data stream for reroute points like sequence headers in MPEG is critical for the overall performance because if the proxy insertion and activation of its functionality is delayed too long the user gets no advantage of it. Therefore, the used data stream has to consider e.g. resulting in the regularly insertion of sequence headers in an MPEG Stream. Due to the very small size of sequence headers this should be no problem.

The higher error rates during proxy insertion only lead to a short image quality decrease which was visible in chicken.mpg through the quantization effects for about 1 second. The higher packet interval leads to a short stop (< 1s) especially in movie rocket.mpg, but does not effect the video quality.

Overall, proxy insertion and removal is recognizable by the user through a short stop and slightly worse image quality but this takes only a short time and therefore, is reasonable for the user. If the filter is used to reduce the bandwidth requirements the effects should be reduced and therefore, the use of proxies is conceivable.

During the preparation of the evaluation the author has tested several movies for applicability and the use of movies with a bigger image size than chicken.mpg the underlying applications reached their performance limit. This should be considered by the RAPP architecture because not each proxy object has the same performance features and therefore, the limit of a proxy object could be reached.
7. Conclusion

This thesis treated multimedia data streams in mobile environments and their adaptation by decreasing QoS through insertion of filter proxies into the data stream. This decreases the required bandwidth and leads to better transmission which allows ongoing videos with slightly less image quality. Existing proxy systems use static proxy insertion during connection establishment and therefore, lack

- extensibility,
- flexible behaviour,
- flexible placement and
- scalability.

To address this problems and our aims from the introduction section the RAPP architecture was developed which provides flexible proxy insertion and removal into data streams. The location problems has been solved by using traders and for communication the CORBA standard was used which provides location independent communication facilities. The implemented prototype was used to demonstrate the feasibility of integrating filters into a data stream which was quite easy if UDP as transport protocol is used. As we could see in chapter 2 filter insertion influences the QoS of the data stream but the quality converts back to the old quality if no filter functions are used. If filters are used the quality is improved and this shows the feasibility of RAPP for dynamic proxy insertion into multimedia streams.

However, the usability of RAPP highly depends on the support of the required CORBA adapter for each participating application and the constraints on the application and data stream. Especially the support of reroute points inside a data stream gets important for the whole performance because a fast filter insertion is useless if the data stream is not rerouted to the filter and no filtering can be executed.

Therefore, to answer the questions from the introduction section the following conclusions can be made:

- RAPP as adaptation technique for multimedia data streams is feasible,
- CORBA is usable for transmission of signaling messages for RAPP,
- no performance problems could be identified which especially includes proxy insertion.
For further work the implementation of proxy objects for other data streams should be a main task. Furthermore, the evaluation has to be extended with further scenarios, especially the signaling messages have to be evaluated. The prototype has to be extended with a QoS module wherefore *Open Bindings* could be used. RAPP should also consider proxy object performance limitations wherefore an advanced *proxy description* is imaginable. Besides, application adaptation techniques for providing the CORBA adapter have to be further investigated and, if possible, standardized approaches could be provided. As an improvement for widespread use, accounting and security mechanism have to be considered.

At the moment only UDP as transport protocol is provided but hand over with TCP is interesting for further studies. Besides, TCP is reliable wherefore the performance could be better but TCP is not suitable for multimedia applications.

Using audio and video streams with CORBA and getting QoS data is also addressed by work of the OMG [McGrath, Rutt & Ottensmeyer 1997, OMG 1997d, Mungee, Surendran & Schmidt 1999] and therefore, this should be integrated into RAPP once it gets standardized. Nevertheless, they do not utilize a new concept because they also use an environmental protocol and not GIOP with IIOP which is not suitable for multimedia streams.

The mobility aspect was not considered in this thesis because it was assumed that the underlying system, for example Mobile-IP, provides this service. However, the OMG thinks about CORBA and mobility (OMG 1998b) and this could also be used for RAPP.
A. Appendix

/***************************************************************
* Name : proxy.idl
* Task : IDL file for ProxyService package
* * Created : 13.May.1998
* * last changed : 06.October.1998
* * Version : 1.0
* * Author : Michael Ebner, Jochen Seitz
* * Comments :
***************************************************************/

#ifndef UK_AC_LANCS_COMP_PROXYSERVICE_IDL
#define UK_AC_LANCS_COMP_PROXYSERVICE_IDL

module uk {
module ac {
module lancs {
module comp {
module ProxyService {

//========================================================================
// NamedValue
//========================================================================

// for parsing arguments, values, properties,... between methods

typedef string Identifier;
typedef unsigned long Flags;

interface NamedValue {
    readonly attribute Identifier name; // argument name
    readonly attribute any value;     // argument
    readonly attribute Flags arg_modes; // argument mode flags
};

typedef NamedValue Argument;

}}
}
}
}
#endif
interface Arguments {
    typedef sequence<Argument> Args;
    typedef sequence<Identifier> Identifiers;

    // insert argument
    oneway void put(in Argument argument);

    // get argument with identifier
    Argument get(in Identifier name);

    // remove argument
    oneway void remove(in Argument argument);

    // remove argument with identifier "identifier"

    // returns all arguments
    Args arguments();

    // returns all identifiers
    Identifiers getIdentifiers();

    // number of Arguments
    long size();
};

interface ProxyObject;
interface ProxyDescription;

// Daemon for proxies (player, filter and server)
interface Factory {
    typedef sequence<string> ProxyObjectTypes;
    exception ConnectionRefused {};
}
readonly attribute string id;
readonly attribute ProxyObjectTypes types;

ProxyObject installProxy(
    in ProxyDescription proxyDescription,
    inout StreamData streamData, in ProxyObject predecessor);

oneway void deleteFactory();

//========================================================================
// PROXY OBJECTS
//========================================================================

//========================================================================
// Generic Proxy Object
//========================================================================
interface ProxyObject {

    readonly attribute string id;

    // control message with type Arguments as parameter
    oneway void controlMessage(in Arguments arguments);

    // say predecessor: send to receiver now -> call reRouteStream
    // proxy knows it is not further required and can do things
    // like empty catch, ... but not delete proxy
    // delete has to be explicitly called

    // reRouteProxy has to call reRouteStream by his predecessor
    oneway void reRouteProxy(in ProxyObject predecessor,
                              in StreamData streamData);

    boolean reRouteStream(in StreamData streamData);

    // remove proxy object
    oneway void deleteProxy();

};

//========================================================================
// Class-dependent Proxy Objects
//========================================================================

interface FilterProxyObject : ProxyObject {

    boolean filterOn();
    boolean filterOff();
    boolean encreaseFiltering();
    boolean decreaseFiltering();

};

//========================================================================
// Type-dependent Proxy Objects
//========================================================================
interface MpegFilterProxyObject : FilterProxyObject {
    boolean dropBFrames(); // erase all B-Frames in MPEG-Stream
    boolean dropBandPFrames(); // erase all B- and P-Frames
    boolean dropColor(); // change colors to black and white
    boolean requantize(); // requantize resolution
}

// Implementation-dependent Proxy Objects

// no implementation dependent proxy objects defined

// Application Specific Data

interface ServerAdapter :
    ProxyObject {
        attribute long sendRate; // frame per second if rate < 100
    }

interface MpegFactory;

struct PlayerArgs {
    MpegFactory factory;
    string filename;
    long rate;
    long event;
    string filterAction;
    string filterParameter;
    string dither;
};

interface PlayerAdapter : ProxyObject {
    oneway void informNewSendRate(in long sendRate);

    string insertProxyObject(in string source, in string sink,
        in ProxyDescription proxyDescription);

    boolean removeProxyObject (in string proxyID);

    boolean removeDataStream();

    oneway void setSendRate(in long sendRate);
}

interface MpegFactory : Factory {
    ServerAdapter installServer(
        inout StreamData streamData,
        in string filename,
in long rate, in PlayerAdapter player);

PlayerAdapter installPlayer(
   in PlayerArgs playerArgs,
   out string serverID);
);

//========================================================================
// Trader Data
//========================================================================

//@interface ProxyTrader;

// Type Definitions
//@interface ProxyTrader;

typedef sequence< Factory> ProxyFactoryList;
// This type defines an array of handles to proxy factories.
// It is used as a return parameter in the trader’s query
// operation.

typedef sequence<long> IntArray;
// This type defines an array of integer values used to define
// the location preferences for proxy objects.

//@interface ProxyTrader;

interface ProxyDescription{

   attribute IntArray locationSequence;
   // defines the preferred placement for this proxy object.
   attribute long inStreamType;
   // describes the stream type of the incoming stream according to MIME
   attribute long inSubType;
   // describes the stream subtype of the incoming stream
   attribute long inSubTypeEnhancement;
   // describes enhancements to the stream subtype of the incoming stream
   attribute long outStreamType;
   // describes the stream type of the outgoing stream according to MIME
   attribute long outSubType;
   // describes the stream subtype of the outgoing stream
   attribute long outSubTypeEnhancement;
   // describes enhancements to the stream subtype of the outgoing stream
   attribute boolean lossProneCompression;
   // specifies whether this proxy uses lossy compression
   attribute Factory proxyFactory;
   // for simplicity reasons, the exporting proxy factory is included in
   // the proxy description
   string proxyShortDesc();
}
void proxyPrint( in ProxyTrader trader );
// Just a procedure to pretty-print the contents of a
// proxy description

// Interface Definitions for ProxyTrader

interface ProxyTrader{

// The export procedure is used to export proxy offers to the
// proxy trader. If the export was successful, the procedure returns
// an identifier used for manipulating this object, otherwise it
// returns 0.
long export( in Factory proxyFactory,
in long proxyClass,
// Handle to offering proxy factory
// Class of proxy object to be registered:
// 1: Filtering proxy
// 2: Transforming proxy
// 4: Caching proxy
// All combinations are allowed by
// adding the IDs of the different types.
in ProxyDescription proxy );
// Description of proxy functionality;

// The withdraw procedure is used to withdraw an offer from
// the trading service. One has to specify the ID returned as
// the result of the export function. If the withdrawal was
// successful, the procedure returns TRUE otherwise FALSE.
boolean withdraw( in long proxyOfferID );

// To query the trading service, the query procedure is used.
// This procedure returns a list of handles to proxy factories
// offering the desired proxy objects. If this returned list is
// empty, no suitable proxy offer could be found.
ProxyFactoryList query( in long clientID,
// used to identify the client
// (meaningful for security checks to be added...)
in ProxyTrader serverProxyTrader,
// Handle to the server's proxy trader.
// Used if a proxy pair has to be installed.
in ProxyDescription proxy,
// This proxy description contains the
// requirements the client has defined
// for the proxy object to be inserted.
in long preference,
// used to influence the sequence of the
// suitable proxy factories. (Default value 0)
in long howMany );
// limits the number of handles to proxy
// factories returned in the result.

// For debugging reasons, this procedure lists all the offers
// that have been exported to the trading service. If the service
// does not provide any offer, the procedure returns FALSE.
// ===================================================================
// boolean showProxies();

// In order to print something onto the graphical user interface
// of the trader, it offers this procedure.
// ===================================================================
void writeToWindow( in string ausgabe );

#endif
Glossary

Acronyms

A

ACM ................................. Association for Computing Machinery
API ................................. Application Programming Interface
ATM ................................. Asynchronous Transfer Modus

C

CCITT ........................ Consultative Committee for International Telegraph and Telephone (now ITU-T)
CSMA .............................. Carrier Sense Multiple Access

E

ETSI ................................. European Telecommunications Standards Institute

G

GSM ................................. Global System for Mobile telecommunications (formerly Groupe Spécial Mobile)

H

HTTP ................................. Hypertext Transfer Protocol

I

ICMP ................................. Internet Control Message Protocol
IEC ................................. International Electrotechnical Commission
IETF ................................. Internet Engineering Task Force
IFIP ................................. International Federation for Information Processing
IP ......................................................... Internet Protocol
ISO ..................................................... International Organization for Standardization
ITU ..................................................... International Telecommunication Union
ITU-T ................................................. ITU Telecommunications Standardization Sector (formerly CCITT)

L
LAN .................................................. Local Area Network
LBX .................................................... Low Bandwidth X Protocol

M
MAN .................................................... Metropolitan Area Network
MPEG ................................................. Moving Picture Experts Group

R
RAPP .................................................. Reactive and Adaptive Proxy Placement

T
TCP .................................................... Transmission Control Protocol

U
UDP .................................................... User Datagram Protocol

W
WLAN ................................................ Wireless Local Area Network
WWAN ................................................ Wireless Wide Area Network
WWW ................................................ World Wide Web
Terms

jitter The variability in end-to-end delay (latency) experienced in a stream interaction

latency The end-to-end delay for a given information quality of service (QoS) A set of non-functional properties associated with a given service

QoS management Supervision and control to ensure that the desired level of QoS is sustained

static QoS management QoS management functions carried out at binding creation time

admission control A static QoS management function which performs a test to see if a given interaction can be accepted (given its desired level of QoS)

QoS negotiation A static QoS management function to reach agreement on the given level of QoS to be sustained

resource reservation A static QoS management function which pre-allocates resources in an attempt to guarantee a given level of QoS

dynamic QoS management QoS management functions carried out during the lifetime of a binding

QoS monitoring A dynamic QoS management function which checks that a given level of QoS is being sustained

QoS renegotiation A dynamic QoS management function enabling the level of QoS to be changed

QoS policing A dynamic QoS management function to ensure that an application adheres to its QoS contract

QoS maintenance A dynamic QoS management function to ensure that a given object attains the desired level of QoS

throughput The volume of information delivered in unit time for a given interaction
Bibliography

ALO. 1998. WWW-Homepage. ALOHA Networks, inc. PO Box 29472 San Francisco, CA 94129-0472.
   URL: http://www.alohanet.com/


   URL: http://bmrc.berkeley.edu/projects/mpeg/

   URL: http://clintonvideo.broadcast.com/news/clintontestimony/real2.stm

   URL: http://www.w3.org/pub/Conferences/WWW4/Papers/56


URL: http://drogo.cselt.stet.it/ufv/leonardo/paper/isce96.htm


URL: http://www.comp.lancs.ac.uk/computing/research/mpg/most/guide.html


URL: www.prenhall.com

URL: http://www.research.microsoft.com/os/sosp%2d15/fox.txt


URL: http://www.iridium.com


Laukien, M., Dr. U. Seimet, M. Newhook & M. Spruiell. 1998. ORBacus For C++ and Java. 3.0 preview release ed. 44 Manning Rd., Billerica, MA 01821, USA: Object-Oriented Concepts.


URL: http://www.oldenbourg.de


URL: www.prenhall.com


URL: http://www.quoininc.com/quoininc/COMCORBA.pdf

URL: http://wwwipd.ira.uka.de/ tichy/publications/Catalogue.doc

URL: http://www.wiley.com/compbooks/vogel/index.html


URL: http://www.cs.columbia.edu/ baz/ps/hot-os-v.ps

URL: http://www.cs.columbia.edu/ baz/ps/hot-os-vi.ps

URL: http://www.cs.columbia.edu/ baz/ps/mobicom-97.ps.gz
Index

Ad hoc network, 27
Adaptation, 29
Adapter, 65
   Library, 69, 70
   Surrogate, 69, 70
   Transducer, 69
   Wrapper, 68, 70
Application interface, 66
ATM, 13
BOA, 42
Channels, 26
COM, 43
CORBA, 33
   Naming Service, 45
   Trading Service, 45
CORBA domains, 44
CORBA facilities, 44
CORBA services, 44
Design Pattern
   Adapter, 67
   Bridge, 68
   Proxy, 47, 48
DII, 42
DOM, 33
DSI, 42
ESIOP, 43
ETSI, 20
Factory, 71
GIOP, 43
GloMop-Project, 51
GSM, 28
HTTP, 51
I-TCPP, 50
IDL, 35
IIOP, 43
IOR, 43
Iridium, 24
Laptop, 21
Mobile
   communication, 23
   end systems, 21
   telephone, 21
Mobility, 19
   Personal, 20
   Terminal, 19
MPEG, 14
   frames, 17
   video stream, 17
Multimedia, 3
   communication, 8
   end systems, 7
OA, 42
Object model, 33
OMA, 33
OMG, 33
OreOs, 51
Pager, 21
Palmtop, 21
PDA, 21
POA, 42
Prototype, 74
Proxies, 47
Proxy Object
   Creating, 72
Proxy Objects
   Classes, 54
   Classification, 53
   Interfaces, 55
   Location, 56
   PIP, 58
   Types, 54
QoS module, 64
Quality of Service (QoS), 11
RAPP, 52