# A CORBA-based Proxy Architecture for Mobile Multimedia Applications

Jochen Seitz, Nigel Davies, Michael Ebner, Adrian Friday
*Computing Department, Lancaster University,*
*Lancaster LA1 4YR, U. K.*
*Tel. ++44 (0)1524 65201, Fax. ++44 (0)1524 593608*
*email (seitz, nigel, ebner, adrian)@comp.lancs.ac.uk*

Key words: Distributed object-oriented multimedia, CORBA, proxy, filtering, mobile computing

Abstract: In many cases users of mobile computers wish to have the same applications running and to have access to the same information as they would when connected to a fixed network. Such transparency is difficult to realise given the differences in the fundamental characteristics of wired and wireless links. These differences, apparent in variations in parameters such as delay, throughput and error rate can have a significant impact on demanding applications such as those which exploit multimedia communications. To overcome these problems, application requirements have to be adapted to match the capabilities of the wireless communication link. One general method of achieving this is the introduction of one or more proxies between the distributed components. This paper presents a generalised architecture for installing and managing proxies in the network in order to enable applications to continue operation in the face of variations in quality of service. The developed architecture is based on the CORBA standard to improve flexibility and acceptance.

## 1. INTRODUCTION

Today's innovations in the area of processors, displays and battery techniques promote the use of portable, mobile computers. However, progress in developing wireless communication architectures suitable for these mobile computers cannot compete with the introduction of high performance architectures for wired networks. Nevertheless, users want to use their mobile as they would use a computer in a fixed network. Hence, the challenge is to adapt the communication requirements to suit the wireless link.

One approach is the use of proxies acting on behalf of the mobile users (see section 2). These proxies can help to reduce the communication requirements for the wireless link and, therefore, integrate mobile users into distributed applications. However, existing proxy systems tend to be rather inflexible, requiring application/data specific proxies which must be instantiated on pre-determined nodes. This lack of flexibility in proxy use and run-time configuration limits the scalability and potential usefulness of these systems.

This paper defines a CORBA-based middleware platform to manage the insertion, modification and deletion of proxies into a given data stream on behalf of a mobile client. In a first prototype, this architecture, termed the RAPP architecture, has been applied to a video application using MPEG filters as proxy objects.

The paper is organised as follows: Section 2 generally deals with the proxy concept. Section 3 introduces the RAPP architecture. This architecture consists of two main processes: the Proxy Selection Process (PSP) to decide if and what type of proxy has to be inserted or modified (cf. section 3.2) and the Proxy Installation Process (PIP, cf. section 3.3) that, given the information about the type of proxy, determines which specific proxy instance has to be installed at which node. The concept of the RAPP Architecture has been implemented in a

first prototype, which is detailed in section 4. Finally, section 5 concludes with a summarisation of the paper and an outlook to future work.

## 2. THE PROXY CONCEPT

Proxies can be found in many areas of communication technology. The most obvious examples are proxies used as part of security architectures based on firewalls [Lodin 1998] or those which are used to support the World-Wide Web [Luotonen 1994]. In general, a *proxy* is an intermediate system that acts on behalf of a client by receiving data from a data source (e.g., a server), processing the data and then relaying it to the client. This makes it possible for an application-level proxy, which is used in a firewall architecture to determine whether the data meets security standards and then allows or denies access to the client. A World-Wide Web proxy may otherwise be able to filter and to compress data to save bandwidth or to pre-cache web pages to allow faster access to them. The concept of a proxy can be compared to that of intelligent agents [White 1998], although agents usually are equipped with more autonomy than proxies. Related ideas can also be found in [Liscano 1997].

Proxy services can be used in the context of mobile computing to adapt the requirements of a communications stream or application to match the services currently available to the mobile user [Fox 1996b]. In addition, proxies can be installed on the edge of a wired network allowing different protocols to be used in the fixed and wireless domains (see figure 1) [Badrinath 1993].

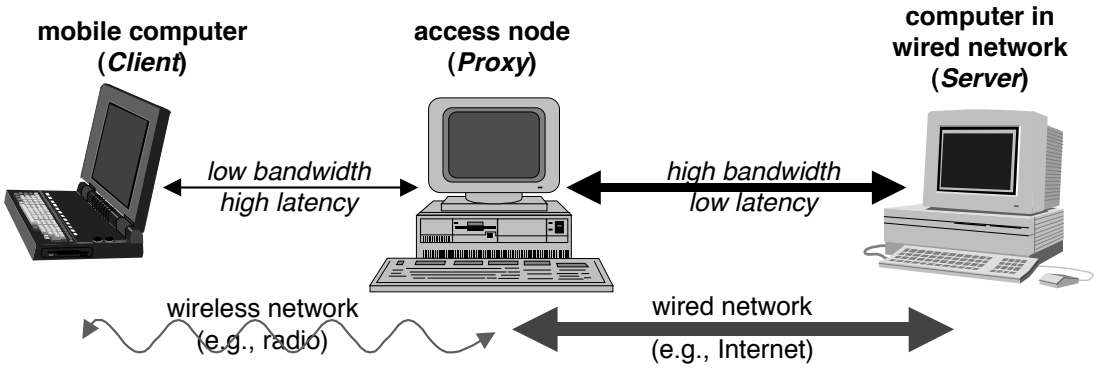


*Figure 1.* Example of a Proxy

Generally, proxies can be divided into two main categories, those which modify protocols and those which modify content.

– **Protocol Proxies**
  Protocol proxies can work either at the *network or transport layer*, modifying the network or transport protocol to enable communications across the wireless link, or at the *middleware or application layer*, modifying specific protocols such as HTTP. Examples of systems which use this class of proxy include *Indirect TCP (I-TCP)* [Bakre 1995], [Fieger 1997], Padmanabhan's work on modifying *HTTP* to reduce packet round trip times [Padmanabhan 1995] or the *Low Bandwidth X Protocol (LBX)* [Fulton 1993].
– **Content Specific Proxies**
  Proxies can also modify the application data to reduce the required bandwidth or to pre-compute data. One example, covering the area of Computer Aided Design or of network management using a pen-based palmtop computer, is described in [Citrin 1997].

Systems which combine both of the above approaches have also been developed:

- In [Brooks 1995] a system based on *HTTP transducers* (proxies) is described which uses a modified version of HTTP and filters data in order to reduce the required bandwidth.
- The same application domain is covered in the GLOMOP project [Fox 1995]. Besides modifying HTTP, the project suggests loss-prone distillation in order to reduce the amount of data for graphics. The follow-on project *Pythia* adapts to the characteristics of the mobile node (e.g., resolution and number of colours of its display) [Fox 1996a].
- One rather general approach is a proxy architecture developed by B. Zenel [Zenel 1995], [Zenel 1997]. It comprises several filters on different levels: transport oriented filters (for ICMP or TCP), application protocol oriented filters (for NFS or SMTP) and filters for application data, e.g. for HTML-pages or MPEG-images.
- Finally, the approach of *service proxies* divides the applications into the mobile and the proxy part. Furthermore, the mobile applications might take into account changes of the mobile node's characteristics (e.g. by inserting a PCMCIA-card) and thus adapting to the new situation on the mobile host [Hokimoto 1996].

Of course, this list of proxy-based approaches is not complete. Nevertheless, a number of important shortcomings in existing approaches can be identified.

*Extensibility*. A proxy is normally dependent on both the application and the type of data the applications exchange. Most approaches have 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. More specifically, the adaptation required to support operation over a wireless link is achieved by inserting an appropriate proxy: fine-grained tuning of the proxy's operation to cope with subsequent variations is not possible.

*Flexible Placement*. The location of the proxy is an important factor in determining the performance of the system. However, optimal placement is often not possible in a heterogeneous network. We return to this issue in detail in section 3.3.

*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 1996a]). More generally, for the system to scale proxies must be distributed throughout the system.

We have attempted to address the above issues in our architecture which is described in the remainder of this paper.

# 3. THE RAPP ARCHITECTURE

This section details the RAPP architecture. RAPP is an abbreviation for **R**eactive **A**daptive **P**roxy **P**lacement and is based on the "Common Object Request Broker Architecture (CORBA)" standard [OMG 1998]. After a general overview, the processes used for proxy selection and installation are described in detail.

## 3.1 General Overview

Figure 2 illustrates the general RAPP architecture. Given a distributed multimedia application consisting of a client, a server and a specific application protocol, the RAPP architecture extends this application using CORBA adapters and implements special processes: the *Proxy Selection Process (PSP)* and the *Proxy Installation Process (PIP)*. Both are described later in this section.

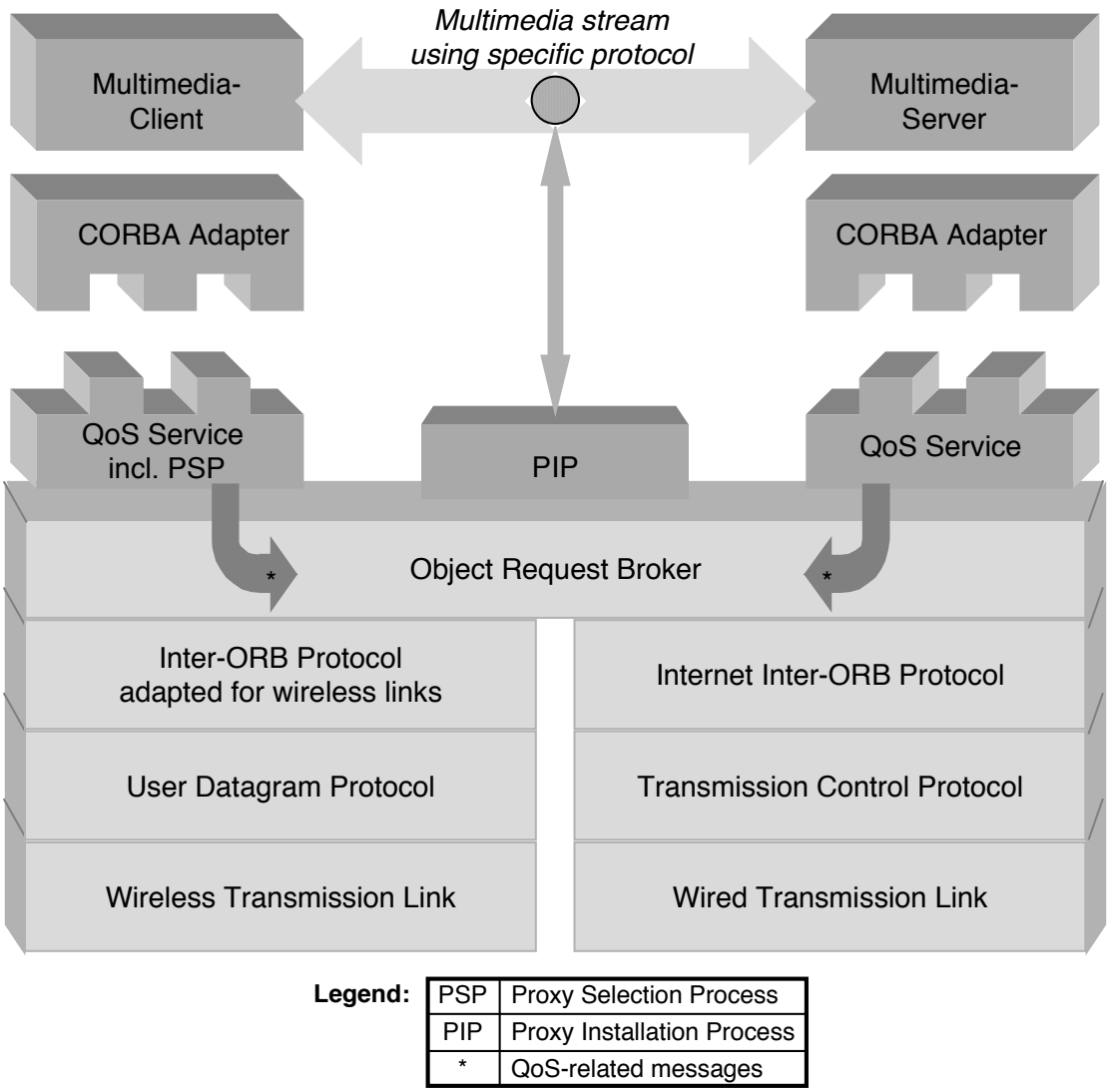


*Figure 2*. The RAPP Architecture

The general aim of the RAPP architecture is to react to decreasing quality of service (QoS) by installing proxy objects into the data stream, which reduce the required bandwidth by implementing stream-specific filters. Furthermore, RAPP allows adaptation to the current QoS by tuning the proxy objects so that despite changing transmission characteristics the communication process will adapt to achieve best results. Finally, increasing QoS leads to modifying or de-installing proxies.

For the wireless link connecting the mobile host, we are proposing an adaptation of the Internet Inter-ORB Protocol IIOP. First concepts have already been worked out [Wolf 1998]. Nevertheless, this work is still in progress.

## 3.2 The Proxy Selection Process

On client side, the *Proxy Selection Process (PSP)* is the active instance of the RAPP architecture. Therefore, the applications provide the PSP with a classification of their data streams. The PSP monitors the quality of service (QoS) for each communication data stream, and, after noticing a QoS decrease chooses one of the current data streams for proxy insertion. This choice is dependent on user preferences described in this section.

### 3.2.1 Stream Classification

As the kind of proxy object to be inserted is dependent on the characteristics of the data stream, a classification of communication data streams has been defined based on the content types and subtypes of the *Multipurpose Internet Mail Extensions MIME* standard [Freed 1996]. For the RAPP architecture, we decided to restrict the number of content types to five: "text", "application", "image", "audio" and "video". The content types "multipart" and "message" are useful in an e-mail environment, but seemed to be not very advantageous for describing data streams. Each content type may be further characterised by a subtype, i.e. each data stream can be classified as a <content type, subtype> pair. For example, an MPEG video data stream can be characterised by <video, mpeg>.

### 3.2.2 QoS Monitoring

In order to decide when a proxy object has to be installed, the PSP monitors the actual QoS for every communication data stream. Therefore, the communication protocols and the distributed application provide quality of service information. Because the CORBA adapter on the server side sends messages containing (application related) QoS characteristics of the stream, the PSP can compare the QoS of the incoming stream with the characteristics of the same stream leaving the server. Whenever the difference is beyond a certain threshold, the PSP requests a proxy instantiation.

### 3.2.3 User Preferences

Although selection and instantiation of a proxy object should be done automatically, the preferences of a user might also influence this process. Therefore, we decided to define some user preferences to control the selection process:

1. The user might express a generic priority for different data streams. For example, if there are several streams and the quality of service decreases, this priority will help to decide which stream should be modified by inserting a proxy object.
2. For every stream type, the user can decide whether they allow loss-prone compression techniques or not.
3. Since the quality the user perceives is depending on his mobile system, he might allow the system's characteristics to be considered during the proxy selection process as well.
4. Finally, the user can decide between three options for proxy selection and placement: totally automatic, application-specific or user-specific.

## 3.3 The Proxy Installation Process

The need for a proxy to be installed and the type of proxy has been determined in the proxy selection process. This process then addresses the proxy installation process requesting proxy installation. The proxy installation process consists of a hierarchy of instances:

- The *proxy objects* implement filters suited for certain communication data streams.
- The *proxy factories* manage a repository of several proxy objects and are able to install, modify or uninstall a proxy object according to the requests of a client.
- The *proxy 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.

Before describing the steps involved in installing a proxy, we consider the issue of proxy placement.

### 3.3.1 Placement of Proxy Instance

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. This approach does not scale very well if you consider an environment including wide area networks such as the Internet and world-wide mobile clients. In this case, the mobile client might be connected to the Internet without being a part of the administrative domain which is hosting their connection. Hence, the local administration will probably not permit proxy instances to be created on the mobile user's behalf. There are some projects (e.g., the Mowgli project [Liljeberg 1996]) that deal with mobile LAN environments and proxies for such environments, but they merely suppose that a proxy instance might be placed on the access node connecting the wireless part of the network to the WAN.

The RAPP project suggests a rather simple but efficient solution for the problem of proxy placement. First of all, let us consider an idealised world. The optimal location for a proxy would be either on the server, or on the client, or partly on both ends, dependent on the kind of proxy. Some examples will underpin this idea:

- If the proxy shall reduce the bandwidth needed for an MPEG video stream, it might decrease the number of colours or the resolution. Optimally, this is done on the sender to prevent data from being sent that will not reach their destination due to the proxy's filtering.
- If the client needed a proxy realising some kind of pre-caching for HTML-pages, the proxy should optimally be located on the client side, because this is the place where all pages have to be gathered anyway.
- Finally, if the required bandwidth between a client and a server should be minimised using a compression technique, both server and client should host one half of the proxy each: The server's proxy must compress the data and the decompressing part of the proxy runs on the client.

Thus, one can conclude two characteristics of proxies: First, a proxy object might either be atomic or consist of two or more components. Second, the optimal location for a proxy is largely dependent on its functionality.

Leaving the ideal world, there are other practical considerations which might affect the placement of a proxy instance. Firstly, a proxy might require specific operating system or hardware support. For example, an image processing proxy may require a dedicated machine with video processing support. Secondly, the security requirements of both the proxy (and associated application) and the potential hosts may affect the proxy's placement. A node which might be perfectly suitable to run a given proxy may not be prepared to do so unless it is able to authenticate and verify the proxy's operations. Similarly, a proxy may not wish to run on a host which can not guarantee not to interfere with the proxy's activities. Finally, the host node's actual load or its performance might also influence the decision. Hence, in the authors' opinion, proxy placement is likely to be suboptimal at best. Placing a proxy somewhere in the network arbitrarily might lead to unforeseen behaviour. However, finding the optimal placement for a given proxy is possible only if you know the structure of the network connecting client and server, its routing behaviour and the actual and future load of the network. Since such information is unavailable in a network such as the Internet we suggest concentrating on finding suitable proxy locations that might lead to little negative implications: the area around the sender, and the area around the receiver.

Thus, we propose a simple framework for proxy placement based on the following five options (see figure 3):

`L1`: The proxy should be located on the server.

`L2`: The proxy should be located within the server's domain (e.g., the same subnet).

L3: The proxy should be located in the client's domain.
L4: The proxy should be located on the client.
L5: The proxy could be located anywhere in the network.

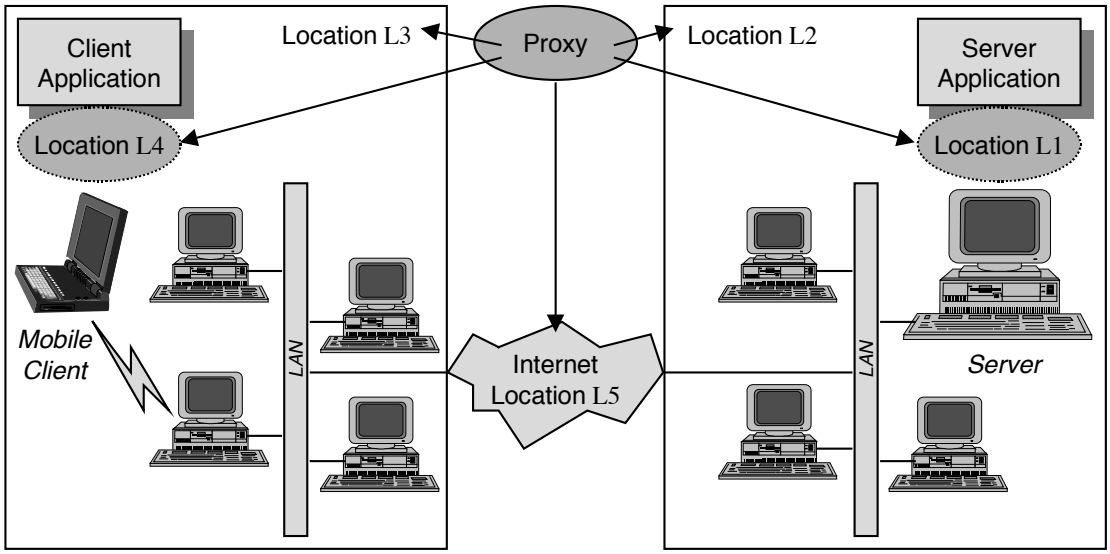


*Figure 3*. Options of Proxy Placement

The preferred location for a proxy is described by a sequence containing location options given above. The first option in the sequence is the best choice for installing the proxy, the other options follow according to their suitability. If an option is not contained in the sequence, this option is not permitted for proxy placement, since it might be senseless.

Considering a proxy instance for an MPEG video transmission, its preferred location could be described by the location sequence {L1, L2, L3, L5}. The best solution, as already pointed out, would be to install the proxy on the server (L1). If this is not possible because of the additional load the proxy would impose on the server or because of security reasons the proxy should be located close to the server, that is in the server's domain (L2). If this is not possible either, the proxy could be instantiated in the client's domain (L3), but within the fixed network, making sure, that only the filtered stream goes over the wireless link. Finally, if these three options cannot be realised, the last option would be to put the proxy somewhere in the network (L5), possibly leading to the video stream being circumstantially re-routed. However, it is not advisable to instantiate the proxy on the mobile client (L4), since this would not help against the limited bandwidth on the wireless link.

### 3.3.2 Determination of Proxy Objects — Proxy Trading Service

Once a mobile client determines that a proxy object should be inserted into the data stream (either automatically or by user intervention), it addresses a special *proxy trading service* supplying the proxy service it desires. All proxy factories have to export the service description for all the proxy objects they are responsible for. This idea of having a trader offering service descriptions and returning handles to objects fulfilling the requirements has also been adopted for the *Telecommunications Information Networking Architecture TINA* for advertising new network services [Hamada 1997].

The proxy trading service processes incoming requests in the following way. Firstly, the service determines which of its local offers could fulfil both the functional and positional requirements in the request. Secondly, the service queries the proxy trading service located in the server's domain since this remote service might offer proxy objects that would also be

suitable. Having received the list from the server's proxy trading service, the client's proxy trading service merges this list with the result of its own local lookup. If both lists are empty, the client's proxy trading service may then ask other proxy trading services for proxy factories offering the required proxy objects, via a federation of proxy trading services [Baker 1997]. Figure 4 illustrates the proxy trading service.

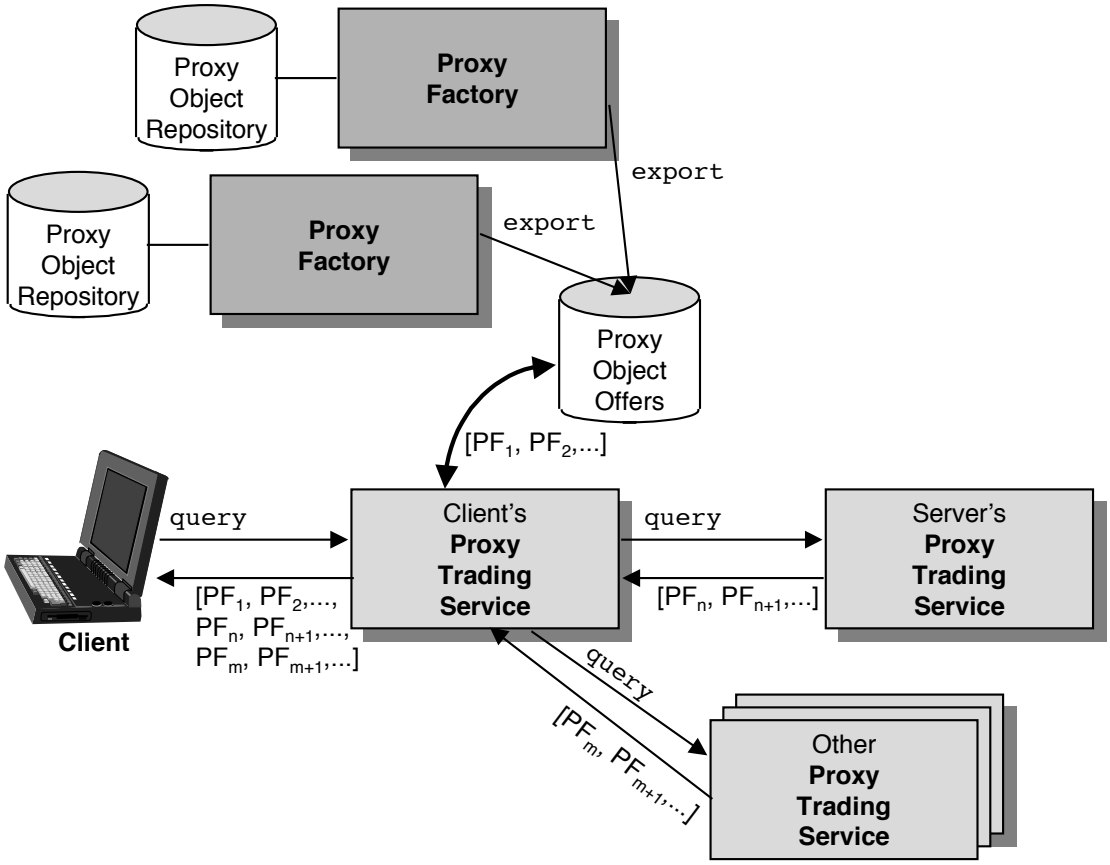


*Figure 4*. The Proxy Trading Service

The enquiry for a certain proxy service will thus lead to a list of proxy factories being returned by the proxy trading service. As pointed out in the last section, the proxy placement is depending on its functionality. Hence, the list might on one hand contain proxy factories close to the client or close to the server, on the other hand it might include pairs of proxy factories if the proxy consists of two halves. 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.

### 3.3.3 Installation of Proxy Object — Proxy Factory

The client then chooses from the returned list and requests the proxy factory to install the proxy. This installation is done according to the location sequence of the proxy and, therefore, is transparent to the client. The installation of a proxy typically requires that the data stream from server to client is re-routed. Therefore, the client opens a new connection to the installed proxy object and tells the proxy to open a connection to the server. Once the connection between proxy and server has been established, the stream is routed from sender over the proxy to the client.

It is possible to install several proxies into one stream. Therefore, the proxy, that prefers a location close to the client, might be told not to open a connection to the server but to the next proxy that already has been installed. In contrast, if the proxy is rather close to the server, it must be installed between the last proxy and the server. Anyway, due to some proxies consisting of two halves, it must be possible to install several proxies at one time.

Once a proxy is installed, it can be controlled by the CORBA control mechanisms conceived for the RAPP architecture. Thus, the proxy object provides some rather general operations to control the proxy functionality. Thus, the client may request modifications of the proxy functionality without having to know the specific implementation. Once the filter is not needed anymore, it can be uninstalled.

### 3.3.4 Security Precaution

Since proxy selection and instantiation is automated, this process could be endangered by third parties trying to achieve a denial of service or even trying to redirect the stream to themselves. Thus, there must be some authentication mechanism for having access to proxy factories and for manipulating a given data stream.

The RAPP architecture provides a simplified authentication procedure as used in a Kerberos environment [Kohl 1993]. As there has been some work in the area for Kerberos-like authentication in a mobile environment, e.g., the *Charon* project [Fox 1996b], this work will be adopted for the RAPP implementation. The major aims of introducing an authentication procedure are:

- Making sure that only the client (or an instance authorised by the client) can insert proxy objects into his data stream, modify or delete them.
- Making sure that the proxy factory itself cannot abuse the client's identity.
- Making sure that an inserted proxy is not influencing other data streams in any way.

## 4. THE PROTOTYPE

To test the feasibility of the RAPP architecture and to gain some performance characteristics, we adapted an existing distributed video-on-demand application supporting MPEG video streams. The developed prototype can be seen as an add-on to the Object Management Group's work on control and management of audio/video streams [OMG 1997, Mungee 1999]. While the OMG standard deals with establishing and managing audio/video streams from a server to a client, the present prototype enhances this functionality by allowing the insertion of proxy objects into the stream to react on variations in transmission quality. This prototype includes a client- and a server-side application and a set of filters to be installed between client and server to reduce the required bandwidth [Yeadon 1996a]. The filter was developed to support multicast in heterogeneous environments leading to a hierarchical filtering architecture. The components of the application are described in the next two sections, before detailing the extended functionality of this application after having been adapted to the RAPP architecture.

## 4.1 Filter Object

For a proxy object, we chose the filter developed by Yeadon for MPEG 1 video streams. It allows several ways of filtering: dropping B- or P-frames, dropping colours, requantisation etc. The filter can be installed and parametrised by a filter daemon receiving a user request. For this, the graphical user interface shown in figure 5 [Yeadon 1996c] has been

implemented. Another graphical user interface can be used to change the parameters once the filter has been installed.

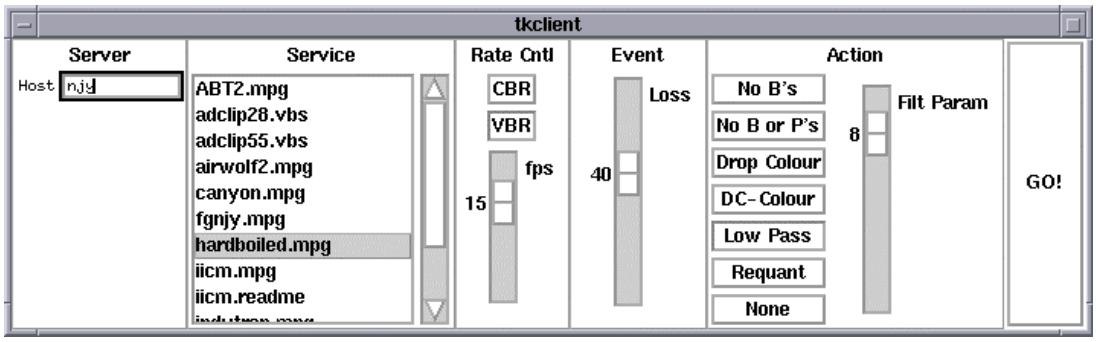


*Figure 5*. Client Application Control Panel

The effect on the visual quality of the video stream produced by such a combination of filters is shown in figure 6 [Yeadon 1996b]. The filter uses frame dropping and colour to black/white mapping to reduce a 1*Mbit/s* MPEG stream to a 9.57*Kbit/s* MPEG stream, enabling transmission of MPEG video images over very low bit rate channels (the quality of the video image is, of course, extremely poor and in a practical system better results could be obtained using alternative encoding techniques such as H.263).

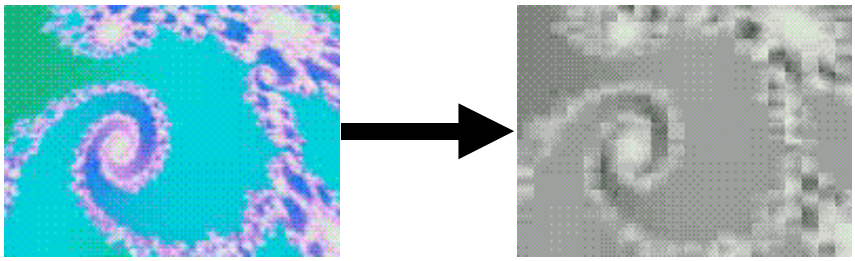

*Figure 6*. Effects of Filtering

The filter can monitor the quality of service and instantiate additional filters or adapt existing ones. However, this filtering technique can only be applied to MPEG video streams. With the RAPP architecture, different kinds of proxy objects can be installed automatically.

## 4.2 Functionality

In order to adapt the described distributed application, all its components (client, server and filter) had to be equipped with a CORBA interface wrapping the functionality of the component. Furthermore, this CORBA interface provides for the filter being exported to the proxy trading service.

The client requests an MPEG video stream from the sender. The sender starts the transmission of the stream and denotes the characteristics of the stream using CORBA calls. Hence, the client object can compare the characteristics of the stream sent by the sender to the characteristics of the incoming stream and decide based on this comparison, whether a proxy object should be installed.

Once this decision is made, the client enquires the proxy trading service for possible proxy objects suitable for an MPEG video stream. The trading service determines the suitable proxy factories and returns a list of them. The client chooses one of the list (until now, simply the first one) and asks for a proxy to be installed. This installation leads to the stream

being re-routed, now going from sender via the installed proxy object containing the MPEG filter to the client. Figure 7 illustrates the video streams before and after the installation of the proxy object.

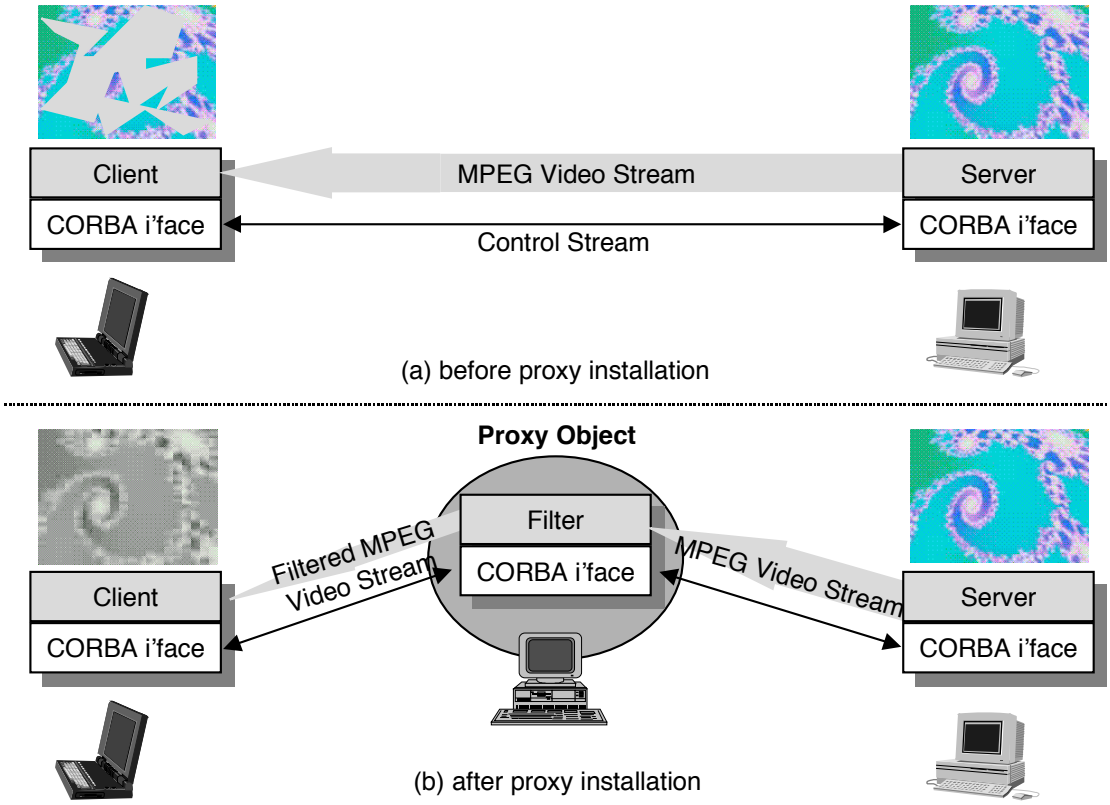


*Figure 7*. Installing a Proxy Object

## 4.3 First Results

For the prototype, we used OMNIBROKER, Version 2.0.4 as CORBA-compliant Object Request Broker [Laukien 1998]. Thus, we implemented the CORBA adapters in the programming language Java and communicated with the original program parts (written in C) using sockets. Hence, there have been only little changes in the original program and the CORBA adapters could be implemented very quickly. Server, Client and Proxy run under Linux and Sun/OS.

In the first prototype, the installation of the proxy object containing the MPEG filter has to be requested by the user. In a local environment, the user does not experience a big latency after having sent out the request to insert the MPEG filter. After the installation, the user can parametrise the filter to change the video quality. The messages used for filter modification are CORBA object calls. Thus, the client need not know the location of the proxy object.

Generally, the first prototype proves that the concept of the RAPP architecture is feasible. Further work, described in the next section, will refine the architecture and allow its quantitative evaluation.

## 5. CONCLUSIONS

This paper has introduced a flexible architecture for creating and managing proxies in a CORBA environment. The architecture is sufficiently general to enable proxies to be developed for new and existing applications and data types. The issue of proxy placement, central to the development of a scalable solution, has been addressed through the definition of a framework for specifying proxy placement. The system is able to dynamically place proxies based on user and application preferences and host and proxy availability. The control of all proxies is carried out using CORBA invocations and new proxies can easily be incorporated given that they support a specified CORBA proxy control interface. Furthermore, these interfaces can be used to fine-tune the behaviour of proxies to provide a further level of adaptability.

To prove the validity of the architecture, we took an existing distributed multimedia application and extended it with CORBA interfaces as required by the architecture. The overhead turned out to be tolerable and the number of lines of code to be implemented rather small compared to the overall implementation of the application.

Our future work will deal with the following items:

- We will add other kinds of proxies to find out how well our modelling technique scales.
- We will analyse the performance and scalability of the described architecture in detail by investigating application scenarios in which parts of the application run on our computers in Lancaster and other parts will be executed in Karlsruhe/Germany.
- The described architecture has been implemented for unicast only. Since this is not sufficient for multimedia applications like video conferencing software or CSCW tools, we will extend our approach to work for multicast applications, too.
- Finally, we will elaborate the security architecture and add new functionality if required.

## ACKNOWLEDGEMENTS

This work has been partly supported by the ESPRC under the auspices of the Reactive Services project.

## REFERENCES

[Badrinath 1993] B.R. Badrinath, A. Bakre et al.: *Handling Mobile Clients: A Case for Indirect Interaction*. Proceedings of the Fourth Workshop on Workstation Operating Systems, Napa, USA, October, 1993, http://www.cs.washington.edu/research/mobicomp/mef/mobile/rutgers.html.

[Baker 1997] S. Baker. *CORBA Distributed Objects Using Orbix*. ACM Press / Addison Wesley, Harlow/England; Reading/Massachusetts, 1997.

[Bakre 1995] A. Bakre and B.R. Badrinath: *I-TCP: Indirect TCP for Mobile Hosts*. Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, Canada, May 30 - June 2, 1995, IEEE Computer Society Press.

[Brooks 1995] C. Brooks, M.S. Mazer et al.*: Application-Specific Proxy Servers as HTTP Stream Transducers*. Proceedings of the Fourth International World Wide Web Conference, Boston, MA, USA, December 11—14, 1995.

[Citrin 1997] W. Citrin, P. Hamill et al.: *Support for mobile Pen-based Applications*. Proceedings of the third annual ACM/IEEE International Conference on Mobile Computing and Networking MobiCom'97, Budapest, Hungary, September 26—30, 1997, ACM Press.

[Fieger 1997] A. Fieger and M. Zitterbart: *Evaluation of Migration Support for Indirect Transport Protocols*. Proceedings of the IEEE Global Telecommunications Conference GLOBECOM'97, Phoenix, Arizona, USA, November 3-6, 1997.

[Fox 1995] A. Fox and E.A. Brewer: *GloMop: Global Mobile Computing by Proxy*. http://www.research.microsoft.com/os/sosp%2D15/fox.txt, 1995.

[Fox 1996a] A. Fox and E.A. Brewer: *Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation*. Proceedings of the Fifth International World Wide Web Conference, Paris, France, May 6—10, 1996.

[Fox 1996b] A. Fox and S.D. Gribble: *Security on the Move: Indirect Authentication Using Kerberos*. Proceedings of the Second ACM International Conference on Mobile Computing and Networking — MobiCom'96, White Plains, New York, USA, 1996.

[Freed 1996] N. Freed and N. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. IETF Network Working Group. Request for Comments 2046, November 1996.

[Fulton 1993] J. Fulton and C. Kantarjiev: *An Update on Low Bandwidth X (LBX)*. The X Resource 5(1), pp. 251—266, January, 1993.

[Hamada 1997] T. Hamada, H. Kamata and S. Hogg: *An Overview of the TINA Management Architecture*. Journal of Network and Systems Management 5(4), December 1997.

[Hokimoto 1996] A. Hokimoto, K. Kurihara and T. Nakajima: *An Approach for Constructive Mobile Applications Using Service Proxies*. Proceedings of the 16th International Conference on Distributed Computing Systems ICDCS, Hong Kong, May 27—30, 1996, IEEE Computer Spciety Press.

[Kohl 1993] J. Kohl and C. Neuman: *The Kerberos Network Authentication Service (V5)*. IETF Network Working Group. Request for Comments 1510, September 1993.

[Laukien 1998] M. Laukien and U. Seimet: *OmniBroker, Version 2.0.4*. Object-Oriented Concepts. Manual, April 16, 1998.

[Liljeberg 1996] M. Liljeberg, H. Helin et al.: *Mowgli WWW Software: Improved Usability of WWW in Mobile WAN Environments*. Proceedings of the IEEE Global Internet 1996 Conference, London, UK, November 1996.

[Liscano 1997] R. Liscano, R. Impey, Q. Yu and S. Abu-Hakima: Integrating Multi-Modal Messages across Heterogeneous Networks, in Proceedings of ENM '97 — First IEEE Enterprise Networking Mini-Conference, IEEE International Conference on Communications ICC'97, Montréal, Québec, Canada, 11 - 12 June 1997, p. 45–53.

[Lodin 1998] S.W. Lodin and C.L. Schuba: *Firewalls fend off invasions from the Net*. IEEE Spectrum 35(2), pp. 26-34, February 1998.

[Luotonen 1994] A. Luotonen and K. Altis: *World-Wide Web Proxies*. First International Conference on the World-Wide Web, CERN, Geneva, Switzerland, May 25-27, 1994.

[Mungee 1999] S. Mungee, N. Surendran and D.C. Schmidt: *The Design and Performance of a CORBA Audio/Video Streaming Service*, Hawaii International Conference on System Sciences (HICSS), Minitrack on Multimedia DBMS and the WWW, Hawaii, January 1999.

[OMG 1997] OMG: *Control and Management of Audio/Video Streams*, OMG RFP Submission, Version 1.0, Friday, May 30, 1997.

[OMG 1998] OMG: *The Common Object Request Broker: Architecture and Specification*. Object Management Group (OMG). Revision 2.2, February 1998.

[Padmanabhan 1995] V.N. Padmanabhan and J.C. Mogul: *Improving HTTP Latency*. Computer Networks and ISDN Systems 28(1/2), pp. 25—35, December 1995.

[White 1998] J. White: *Mobile Agents White Paper*, General Magic, Sunnyvale, CA, USA, 1998, URL http://www.generalmagic.com/technology/techwhitepaper.hmtl.

[Wolf 1998] C. Wolf: *Adaptive Mobile Applications based on a CORBA Infrastructure*. Diploma Thesis, Institute of Telematics, University of Karlsruhe, Germany; Distributed Multimedia Research Group, Lancaster University, UK, 1998.

[Yeadon 1996a] N.J. Yeadon: *Quality of Service Filtering for Multimedia Communication*. PhD Thesis, Computing Department, Lancaster University, Lancaster, UK, 1996.

[Yeadon 1996b] N.J. Yeadon, F. Garcia et al.: *Continuous Media Filters for Heterogeneous Internetworking*. Proceedings of SPIE - Multimedia Computing and Networking (MMCN'96), San Jose, CA, USA, January 1996.

[Yeadon 1996c] N.J. Yeadon, F. Garcia et al.: *Filters: QoS Support Mechanisms for Mulipeer Communications*. IEEE Journal on Selected Areas in Computing 14(7 [Special Issue on Distributed Multimedia Systems and Technology]), pp. 1245-1262, September 1996.

[Zenel 1995] B. Zenel and D. Duchamp: *Intelligent Communications Filtering for Limited Bandwidth Environments*. Proceedings of the 5th Workshop on Hot Topics in Operation Systems HotOS-V, Orcas Island, Washington, USA, May 4—5, 1995.

[Zenel 1997] B. Zenel and D. Duchamp: *A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment*. Proceedings of the third annual ACM/IEEE International Conference on Mobile Computing and Networking MobiCom'97, Budapest, Hungary, September 26—30, 1997, ACM Press.