Improving Video Server Scalability with Virtual Video File System Concept

Michal Ropka, Michal Ślusarczyk, Krzysztof Zieliński
University of Mining and Metallurgy, Distributed Systems Research Group, Krakow, Poland
michs@raster.krakow.pl

Abstract: This paper describes the architecture and design of a Virtual Video File System which is a multimedia database solution. The system provides the functionality of a typical video-on-demand system and is combined with archive storage to obtain larger scalability. Its architecture has been designed with CORBA in mind. The Oracle Video Server and the UniTree Storage System interfaces have been wrapped with CORBA objects. The results is a solution which can be easily expanded or adapted to individual needs.

Keywords: CORBA, multimedia, video on demand, mass storage, streaming.

1. INTRODUCTION

New emerging network services build for e-health, e-learning and e-business applications require online access to audio and video documents. In spite of rapid technological progress, multimedia document storage and streaming still presents a challenging technical problem.

Well-known software developers, like Oracle or Sun, provide their own multimedia solutions, but they are often not powerful enough for today's needs. A video server and a client which simply allows playback and recording of films is not enough for a video-on-demand service or a scientific multimedia database implementation. Such applications require more complex solutions, such as cataloguing functions, protection against

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35616-7_23
W. Cellary et al. (eds.), *Internet Technologies, Applications and Societal Impact*

unauthorized access, multimedia documents indexing, annotation capabilities, etc.

Another problem concerning most existing multimedia systems is insufficient scalability. Video files are significantly larger than most other kinds of data (like text files or even static images). What's more, operations on multimedia content need mass storage devices with short access time, while space on such devices is still rather expensive. This is why multimedia systems are often too costly or too limited in scale to cope with modern application requirements. For instance, a modern U.S. hospital collects an average of 11 Terabytes of multimedia every year.

This paper concerns improving video server scalability with the Virtual Video File System (VVFS) concept. VVFS aims to fill a gap in the currently available multimedia software [1, 3, 10] by combining a low-cost software video server running on a UNIX workstation with massive storage. It is intended to be a system which improves video server functionality and scalability. It combines modern hardware and software technologies to achieve the best effect. The presented concept is rather general in nature, but it has been implemented using the Oracle Video Server [5, 6] and the UniTree Storage System [12].

The paper is structured as follows: In Section 2 the VVFS concept is elaborated. In Section 3 a short comparison of other systems is presented. Next, Section 4 describes the technical aspects of VVFS construction. Preliminary system performance evaluation is presented in Section 5. Typical application scenarios for the constructed system are specified in Section 6. The paper ends with a discussion of future development possibilities and conclusions.

2. VVFS CONCEPTS

From the user's point of view, VVFS is a sophisticated file system, designed for storing video films. The concepts are based on well-known UNIX-like file systems with directories, files, links and access permissions. In addition, it has some specific features intended for multimedia content.

The VVFS logical structure is a hierarchical tree. The basic node is a directory. There is one root directory without a parent directory. Every directory (excluding root) has exactly one parent directory. Every directory contains 0 or more file system units, such as directories, files or links. Every file corresponds to exactly one film. Aside from multimedia content, it contains definable properties, such as short film descriptions, lists of authors, etc. Every file has its owner and a set of access rights.

Links are provided to ease the management of large film databases. There are two types of links: links to files and links to directories. There is no distinction between hard and symbolic links, as in UNIX-like file systems. Links in the VVFS share selected features of hard and symbolic links.

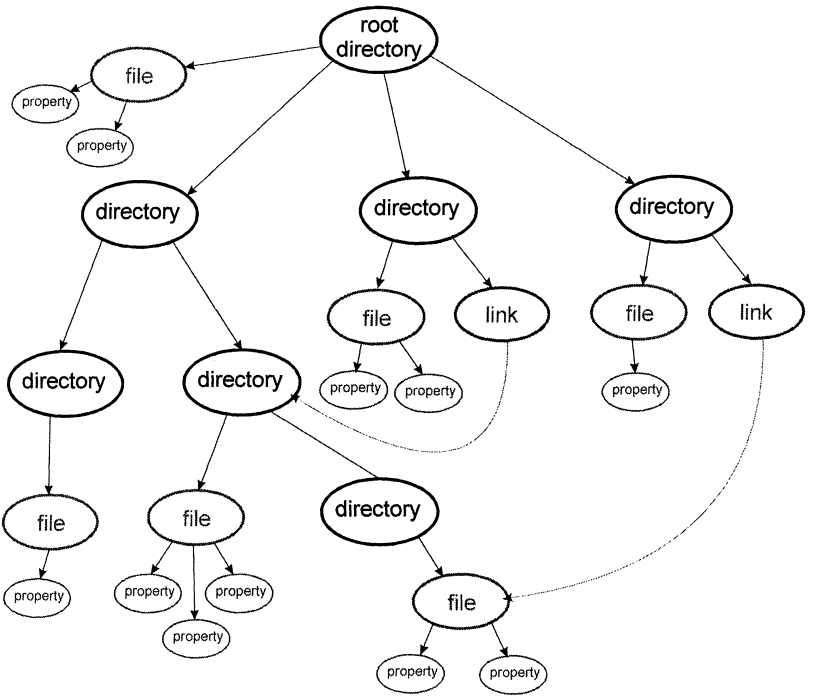

Figure 1. Filesystem logical structure

2.1 System tasks

The system maintains a user database. Every user has an account identified by a login name. Unauthorized access is prevented through the use of passwords. The administrator can add user accounts to the system or remove them. He can also change user passwords. An ordinary user can also change his password.

The most important task of the VVFS is providing functionality for a multimedia database. Our solution is a file system which combines the advantages of UNIX-like systems with additional special features. Thanks to the links mechanism, the same films can be accessed from various

directories, which may be user-based, subject-based and so on. A film is more than just a multimedia stream stored in a file - it also has properties, such as a short description or additional metainformation, helpful in operating a large video library.

Films may be archived on an FTP server when not used. This resolves the problem of maintaining very large multimedia databases, spread over a number of fast but relatively small hard drives and a slow, but capacious FTP server. Such a solution dramatically reduces hardware costs.

Of course, our complex system also provides the basic functionality of a video server, namely playback and recording of films. The de-archiving process may be automatized. The user can request a film to be available at a specified time and the system takes care of handling the appropriate data transfers.

3. COMPARISON WITH OTHER KNOWN SYSTEMS

Besides our system, there are several other systems with similar goals. Most of them are products of university laboratories. In this chapter we would like to compare them with the Virtual Video File System.

The most advanced such system is the **Video-on-Demand** from Berkeley [1]. However, its goals are slightly different from ours. Its authors concentrated on sophisticated algorithms for archive management. The system seems to be a very interesting solution specifically for video on demand, but may not suffice in applications where data protection is significant. Our system provides an access rights mechanism to protect the content from unauthorised access.

The other interesting solution is the **Hierarchical Storage Architecture for Video-on-Demand Services** from Taiwan [10]. In contradiction to the previous system, it is purely theoretical, without an implementation. The main goal of the system is optimization of storage and video server usage.

Our system is now at the first stage of its development and it doesn't contain elaborate archiving and load balancing algorithms. We tried to look at the wider problem and concentrate mainly on open architecture of the system design. As a result, the system represents a framework which can be further enhanced to suit specific, atypical needs. It is intended to be used in a variety of applications. The proposed open architecture is the most important advantage of our system in comparison to other solutions.

4. TECHNICAL ASPECTS OF VVFS CONSTRUCTION

This section explains the technical side of the project. It contains a brief description of the logical and physical architecture, as well as mapping between the logical file system structure and its physical representation.

4.1 Physical architecture

The physical configurations of VVFS may be set up according to a particular hardware and network configuration. All the system components, including external units, such as the video server and the FTP server may work on the same physical machine, although the purpose of such a configuration is questionable. Another extreme is total distribution: each VVFS module, as well as each external unit may work alone on a separate physical server.

VVFS cooperates with external systems which are described below. The particular implementations used in projects are given in parentheses.

Video Server (Oracle Video Server) This is a high-performance video server. It is used by our system for the purpose of playback and recording of films. The native file system for the Oracle Video Server is **Media Data Store (MDS)**, which is optimized for fast transfers, but definitely not for user convenience. It doesn't support directories. All the files are stored in the same namespace. Fortunately, our system conceals the MDS nature from the user while preserving its advantages.

Data Server (Oracle Data Server) Oracle Data Server is a high-performance, advanced relational database engine. It is used by the system to store data.

FTP Storage Server (UniTree) Our system uses a FTP server to archive infrequently used films. Any FTP server can be used, but we provide special support for the UniTree system. UniTree is a hierarchical storage system. It may cooperate with high-capacity storage devices, like magnetic tapes. Its disadvantage is longer access time when compared with other FTP servers.

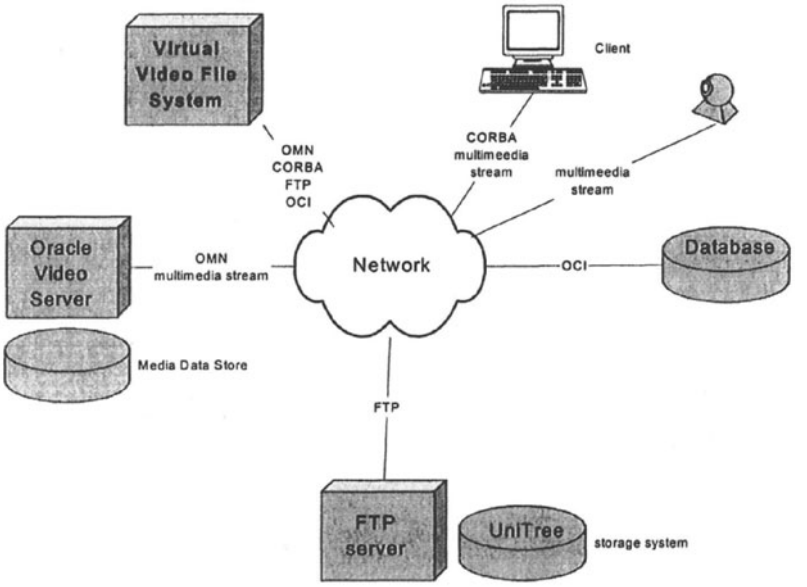

Figure 2. VVFS physical model

4.2 Logical architecture

The system is not implemented as a single program, but rather as a collection of modules. Each module implements a specific logical aspect of the system. This architecture provides for future expansion of the system. Its behaviour can be changed by adding or replacing modules.

The server-side modules are implemented as CORBA objects. Each implements a specific IDL interface.

There is one special server-side module called *system manager* which integrates all the external system functionality. It is intended to be used by client applications to communicate with the rest of the system.

The client application also communicates with the system manager through CORBA. Contrary to other modules, it doesn't implements any IDL interface, so it is the root in the dependency schema. A short description of each module is presented below.

DB manager - All transient data are stored in a database. The database manager is a piece of middleware which connects the database to the rest of system. Modules access the database via a simple database manager API.

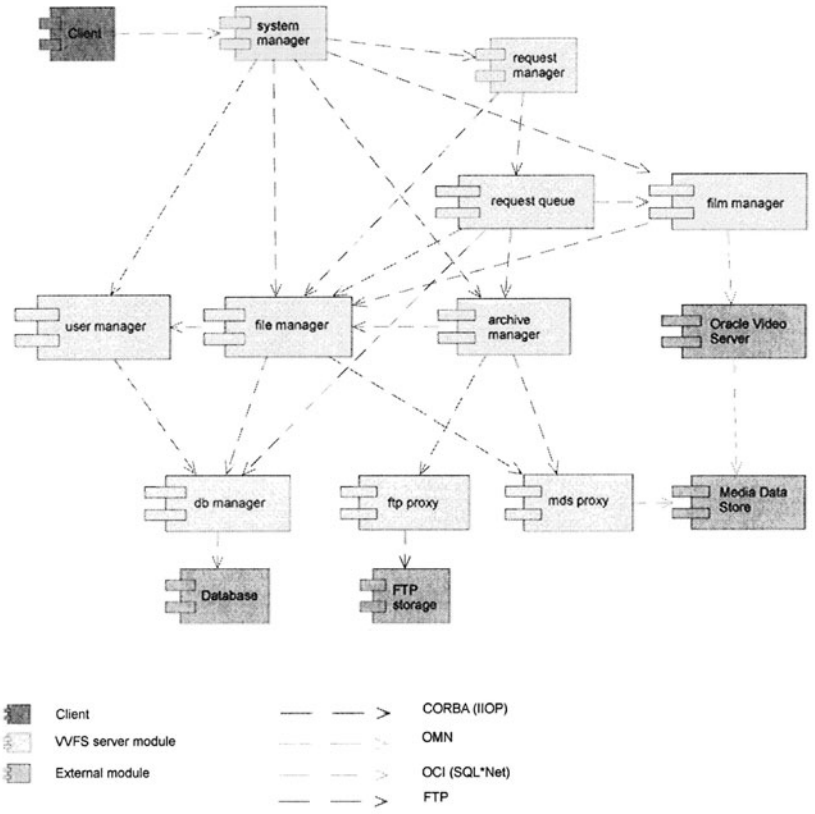

Figure 3. VVFS logical model

MDS proxy - The MDS proxy is a piece of middleware joining the MDS with the rest of system. It communicates with the MDS via OMN. It implements low-level functions for MDS physical files.

FTP proxy - The FTP proxy is a piece of middleware between the FTP server and the rest of the system. This is the only module which implements the FTP protocol. It provides basic FTP operations, but it is also prepared for cooperation with the UniTree system.

User manager - This module is responsible for user management and authorization.

File manager - This module is responsible for logical file system services other than the operations which deal with archiving files.

Archive manager - This module is responsible for archiving files.

Film manager - The film manager module is responsible for playback and recording of films. It provides all the functions needed to create and control a multimedia stream.

Request manager - This module is responsible for serving requests. It contains algorithms, which decide when a download from an archive server should be started and when a local copy should be removed.

Request queue - This is an executive unit for the request manager.

System manager - This module encapsulates the external functionality of the whole system. It is intended to be used by a client application.

Client application - An application, implemented in Java, which provides a GUI for the system.

4.3 Physical mapping of the file system structure

All information concerning the directory tree is stored in the database. File system properties (see Section 2) are also stored there.

Multimedia content may exist on an MDS server, an FTP server or both. The exact locations of individual physical files are stored as file system properties.

Each virtual file contains two physical files: one with the multimedia stream and one with tags used by OVS – although the data structure may handle any number of physical files associated with a virtual file. This may be used in future versions of the system.

5. PRELIMINARY SYSTEM PERFORMANCE EVALUATION

The most significant parameter of system performance is transfer speed between the storage server (FTP) and the local video server file system (MDS). This speed depends on network speed as well as on endpoint hardware performance. The best speed achieved was about 1000kB/s (8Mbps)[1]. Tests were performed over a 100Mbps MAN. We achieved similar results during direct FTP transfers. This bandwidth was shared evenly for concurrent transfers.

When the storage server implements a hierarchical storage system, such as UniTree, infrequently used files can be archived on slow-access media. The access time may be even longer then the transfer time.

[1] Compare with high quality MPEG 1 stream - about 1.5MBbs.

Number of files	Transfer of single file	Summarized transfer
1	910 kB/s	910 kB/s
2	496 kB/s	992 kB/s
3	251 kB/s	1007 kB/s

Table 1. Concurrent transfer speed

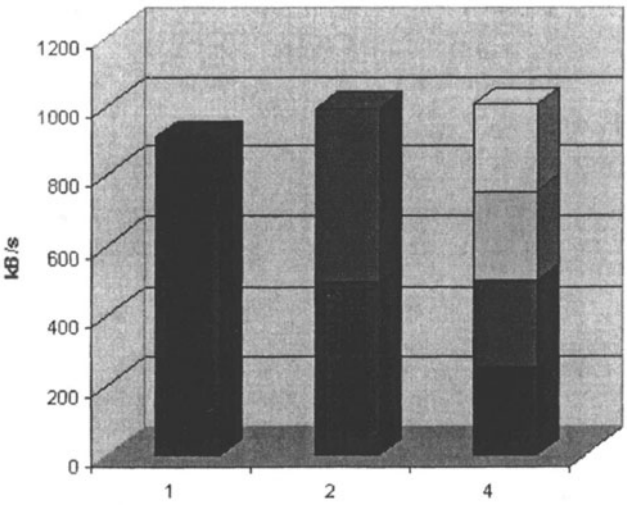

Figure 4. Concurrent transfer speed

The waiting problem is solved by the request mechanism. If a user requests a film early enough, there is a great probability that the file will be available when needed. In this case any transfer is scheduled by the system. Users are not directly involved in the transfer process, so the overall system convenience increases.

At the present stage our system implements a simple decision algorithm, but its sophistication can be increased by upgrading the request manager module. The present algorithm may not be sufficient for systems where films need to be retrieved from archives very often and the MDS capacity is rationally small.

The video streaming performance depends totally on the video server and network used. Our system controls the playback or recording only by sending appropriate commands directly to the video server. The system has passed 5 concurrent streams test correctly. This number is recommended by

Oracle Company for small servers (such as the one our system was tested on). Of course, it can be increased for stronger machines.

Logical operations dealing with file or user management don't require much CPU power, so they are executed almost simultaneously, even when many users are logged in. The system is fully multithreaded. Critical sections are as compact as possible. Convenience of work depends strongly on client-to-server network connection speed.

Exception handling is another issue, and a very important one in large, distributed systems. Our system is quite well protected against common emergencies. Of, course it is impossible to predict every situation and guarantee recoverability in every case, but we tried to do as much as possible in this regard. The system was successfully tested in the following situations:

- loss of communication with one of the server modules,
- loss of communication with the database,
- loss of communication with the video server,
- lack of expected file on MDS or FTP server,
- insufficient free space on MDS or FTP server.

6. APPLICATION SCENARIOS

Before presenting particular applications we would like to give a brief outlook of the client application, which provides a GUI to manipulate the system. It is intended to be used both by ordinary users and administrators. Some functions are available only for administrators.

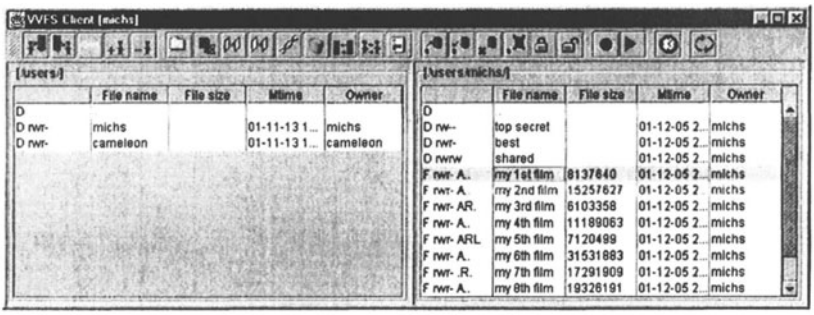

Figure 5. Client application (main window)

The main window of the client application is shown on Figure 5. Two panels allow exploration of the file system in a manner similar to working with popular file commanders. The vertical splitting line can be moved left

or right for user convenience. The current directory for each panel is shown in its top left-hand corner.

Each line in a panel refers to a single file, directory or link. Lines are divided into 5 columns, containing the following pieces of information about objects: mode, name, size (only for files and links to files), modification time and owner.

The top toolbar contains buttons to execute various operations. Most operations are associated with pop-up windows. As an example, we present the property view/edit window.

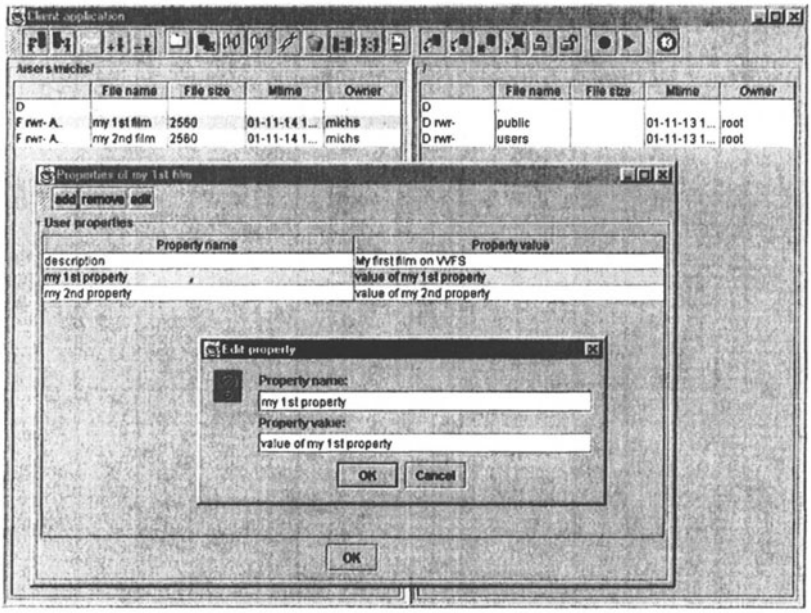

Figure 6. Client application (property window)

The VVFS may by applied in many situations. Below are selected application examples.

6.1 Scientific center

Individual users have their private accounts and directories, where they store their own films - for example video documentation of scientific experiments.

Thanks to the access permission mechanism, their films need not be accessible to others. Instead, those films which are intended to be publicly available may be accessed through links stored in a public directory.

Infrequently used films may be archived and requested to become available when necessary. The requesting mechanism should work very well for planned presentations.

6.2 Video on demand

In this case, many users have read-only access to films. All films are supplied by the administrator. This simple solution can be successfully applied in a hotel. To further simplify matters, it is enough to create one account for the administrator and one for an ordinary user. The latter can then be shared by all users. Our system enables the opening of more then one session per user.

The directory tree may be sorted by genres, subjects, known actors, and many other features. It may also be simultaneously sorted by all of these criteria, thanks to links.

If we want to charge users for accessing the system (as is the case in today's video libraries), user accounts should be separate. In, addition the system may be enhanced by adding a billing manager, responsible for calculating charges.

6.3 Monitoring

The system may be used to keep records of monitored objects, such as department stores, banks, private properties and so on. A special client may record hour-long films sequentially. Such films may be simultaneously recorded by various video cameras. A set of subdirectories, created in the main directory may represent years, which can be further divided into months, then days and hours, while file names may represent individual cameras. Such a structure makes it very easy to find any given film. Of, course, films recorded a long time ago will not be available in the local file system, but may be promptly downloaded from the archive, if necessary. The archive capacity can be great.

7. FUTURE DEVELOPMENT AND CONCLUSIONS

The system is based on modern hardware and software technologies, like CORBA, UniTree, Oracle Database and Oracle Video Server. Most of those are market leaders, but of course all of them have some drawbacks. We tried to take advantage of their features, on the one hand, and to minimize the external effects of their disadvantages on the other.

At all times we remembered that requirements grow with time, and no matter how elaborate a project is at any given point, it will be obsolete in a few years. That is why we have paid special attention to developing an open architecture, which would be easy to extend. The system may be expanded in the following ways:

- modifying existing modules (to change system behaviour),
- creating new modules (to provide new functionality).

Some modules are dependent on external units, such as the video server or the database server. The system may cooperate with off-the-shelf video servers or database engines after applying modifications to selected modules. The rest of system may be left untouched.

The client application may also be altered to provide new functionality after addition of new modules or in situations which call for a special user interface (for example a text-based UI).

Although the system performs advanced tasks, we have managed to make it very user-friendly. The GUI provided by the client application is intuitive and similar to well-known user interfaces for traditional file systems. The file system structure is similar to UNIX-like file systems.

REFERENCES

[1] David W. Brubeck and Lawrence A. Rowe. *Hierarchical Storage Management in A Distributed Video-on-Demand Systems.* IEEE Multimedia. Vol.3. No.3.

[2] Eric Belden Jack Melnick, Phil Locke. *Oracle Call Interface Programmer's Guide, Release 8.1.6.* Oracle Corporation, 1999.

[3] J.E. Bachleshwieler L.A.Rowe, D.A. Berger. *The Berkeley Distributed Video-on-Demand System. Multimedia Computing - Proceedings of the Sixth NEC Research Symposium.* SIAM 1996.

[4] Cathy Baird Lefty Leverenz, Diana Rehfield. *Oracle 8i Concepts Release 2 (8.1.6).* Oracle Corporation, 1999.

[5] Oracle Corporation. *Oracle Video Server TM Developers Guide*, 3.0 edition, 1998.

[6] Oracle. *Oracle R Media Net Developer's Guide*, 3.3 edition, 1997.

[7] Tristan Richardson. *The OMNI Naming Service.* Olivetti & Oracle Research Laboratory, Cambridge, May 1997.

[8] Tristan Richardson. *The OMNI Thread Abstraction.* Olivetti Research Laboratory, Cambridge, March 1997.

[9] Sai-Lai Lo and David Riddoch. *The omniORB2 version 2.8 User's Guide.* AT & T Laboratories, Cambridge, September 1999.

[10] Hong Zhu Lai Yuan Chen Lui, Yin-Dar Lin. *A Hierarchical Network Storage Architecture for Video-on-Demand Services. Proceedings of IEEE 21th Conference on Local Computer Networking.* October 1996.

[11] *The Common Object Request Broker: Architecture and Specification*, 2.2 edition, 1998.

[12] *UniTree User Guide Release 2.0.* http://www.cyf-kr.edu.pl/ack/unitree/UT-ug/usertitl.html.