Exploiting Group Communication for Highly Available Video-On-Demand Services*

(Extended Abstract)

Tal Anker Gregory V. Chockler Idit Keidar Michael Rozman Jonathan Wexler

{anker,grishac,idish,micr,wexler}@cs.huji.ac.il
http://www.cs.huji.ac.il/{~anker,~grishac,~idish,~wexler}
Institute of Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel

Abstract

Video on Demand services are popular today in hotels and luxury cruise boats. Increasing improvement in communication technology, will invite widespread utilization of VoD services in private homes, provided by telecommunication companies and via the Internet. In such an environment, scalability and fault tolerance will be key issues. In this paper we describe a highly available distributed Video on Demand (VoD) *service, which is inherently scalable and fault tolerant. The VoD service is provided by multiple servers, that may reside in different sites. When a server crashes (or disconnects from its clients), it is replaced by another server in a transparent way; the clients are unaware of the change in the service provider.*

The VoD service exploits the Transis group communication system. Transis assists in achieving fault tolerance, and greatly simplifies the overall service design. Furthermore, the fault tolerance is achieved practically for free: it consumes negligible bandwidth, storage space, and cpu time.

1 Introduction

In this paper we describe a highly available distributed *Video on Demand (VoD)* service. The VoD service is provided by multiple servers, that may reside in different sites. When a server crashes (or disconnects from its clients), it is replaced by another server in a transparent way; the clients are unaware of the change in the service provider. The fault tolerance is achieved at a very low overhead.

VoD services require high *Quality of Service (QoS)* communication for video transmission. Nonetheless, a highly available VoD service is concerned with more than just transmitting a stream of video: Supporting fault tolerance as mentioned above requires the servers to consistently share information about clients. Furthermore, the servers and clients need to exchange messages for connection establishment, flow control and *negotiation and re-negotiation* of *Quality of Service (QoS)* [16]. These tasks may greatly benefit from the group communication paradigm [9], described below.

Group communication systems [1] provide reliable communication services within dynamically changing groups, as well as membership services, which inform the members when other members crash or join the group. These properties make group communication systems an attractive building block for a reliable VoD service. Unfortunately, introducing reliability (message recovery and ordering) requires message buffering and flow control mechanisms which may violate the QoS properties provided by the underlying communication media. These properties are crucial for the performance of the video application.

Recently, several emerging projects addressed the challenge of incorporate QoS communication into the framework of group communication. For example, [9] introduces a *Multimedia Multicast Transport Service (MMTS)*, that incorporates multiple QoS options into the group communication framework. Maestro [8] extends the Ensemble [12] group communication system by coordinating several protocol stacks with different QoS guarantees. The Collaborative Computing Transport Layer (CCTL) [17] implements similar concepts, geared towards distributed collaborative multimedia applications.

Such systems are flexible, and modularly support integration of new QoS options, e.g., the cyclic udp QoS [21] that was implemented as a protocol layer in the Horus system [23]. Furthermore, these systems can exploit various underlying communication proto-

*This work was supported by ARPA grant number 030-7310 and by the Israeli Ministry of Science grant number 032-7658.

cols and technologies, e.g., RSVP [25], ST-II [22] and ATM QoS.

The vic [14] video conferencing tool over the MBone[1] is a flexible framework for packet video. This approach uses a *conference bus* for broadcasting the various media in a conference session (e.g., whiteboard media, audio, and video). The conference bus may be coupled with a *coordination tool*. The MMTS can be viewed as integrating both the conference bus and the coordination tool.

The VoD service presented in this paper was implemented using the MMTS concept: it exploits the Transis [10] group communication system, enhanced with QoS communication. The video stream is multicast using QoS communication, and Transis is used for consistent information sharing, connection establishment, and flow control. These tasks take up about one thousandth of the total communication bandwidth used by the VoD service. Thus, we achieve fault tolerance practically for free. Furthermore, the use of Transis greatly simplifies the service design.

Video on Demand services are popular today in hotels and luxury cruise boats. Today, these services provide a limited variety of video material and limited control over the displayed material. Using our concepts, the VoD service can become scalable, and provide a rich repertoire and full control over the viewed film. Furthermore, a high bandwidth communication infrastructure (e.g., ATM backbone networks along with the Internet infrastructure) is being established in many countries around the world, and high bandwidth communication lines will reach millions of homes in the near future. This increasing improvement in communication technology, will invite widespread utilization of VoD services in private homes, provided by telecommunication companies and via the Internet. In such an environment, scalability and fault tolerance will be key issues.

Existing video on demand systems today [18, 19, 16, 24] are usually neither scalable nor tolerant to server faults, and are therefore inappropriate for multi-user failure-prone networks such as neighborhood servers or the Internet. Current research in the area of VoD often focuses on recovery from disk failures [11, 7], but rarely addresses the question of smooth provision of service in the presence of server and communication failures. Researchers and industries building multi-media applications use ad-hoc solutions, that do not correctly deal with the subtleties of concurrency in distributed computing. The concepts presented in this paper will be beneficial in making future VoD services fault tolerant and scalable.

2 The Video on Demand Service

Clients connect to the *Video on Demand (VoD) service* and request a movie to watch from a list of offered films. The service supports the ATM Forum VoD specs [6]: it allows the client to have full VCR like control over the transmitted material, e.g., pause, restart, and arbitrary advance and rewind. The client's user interface is shown in Figure 1.

The VoD service is *replicated* in order to achieve high availability and *fault tolerance*: Each film is held by a subset of the servers. The service is provided by a dynamically changing group of servers. The clients connect to a generic service, and are not aware of the fact that it is provided by multiple servers. This design makes the service inherently *scalable*, and resilient to machine and network failures. When a server crashes or detaches, other servers take over its job and serve its clients in a transparent way. *Load balancing* is introduced in order to allow for maximum availability: e.g., a new server may be brought up to alleviate the load of overloaded servers, and servers may migrate clients "on the fly".

Different clients that exploit the VoD service may have different network and computing capabilities. *Negotiation and re-negotiation* of *Quality of Service (QoS)* allows the client and server to agree on certain QoS parameters, adjusted to actual capabilities of the client. Our video service uses feedback-based flow control for QoS re-negotiation [16], dynamically adjusting the transmission rate according to network and client capabilities.

3 The Environment

The VoD service exploits a group communication system enhanced with QoS communication. We implemented a prototype of the VoD service to demonstrate feasibility of our concepts. The prototype was implemented using UDP/IP over 10 Mbit/s Ethernet for video transmission, and the Transis [10] group communication system. The prototype runs on SUN Sparc machines, running SunOS 4.1 UNIX. The video is stored and transmitted in MPEG [13] format, and is decoded using software. We are currently in the process of replacing the software video decoders by hardware decoders on PCs running Windows 3.11/95/NT. Later, we intend to exploit native ATM for the QoS communication. In Section 3.1 we describe the services of Transis. In Section 3.2 we describe how we plan to run the VoD service over an ATM network.

[1] Information about the MBone can be found in http://www.best.com/ prince/techinfo/mbone.html.

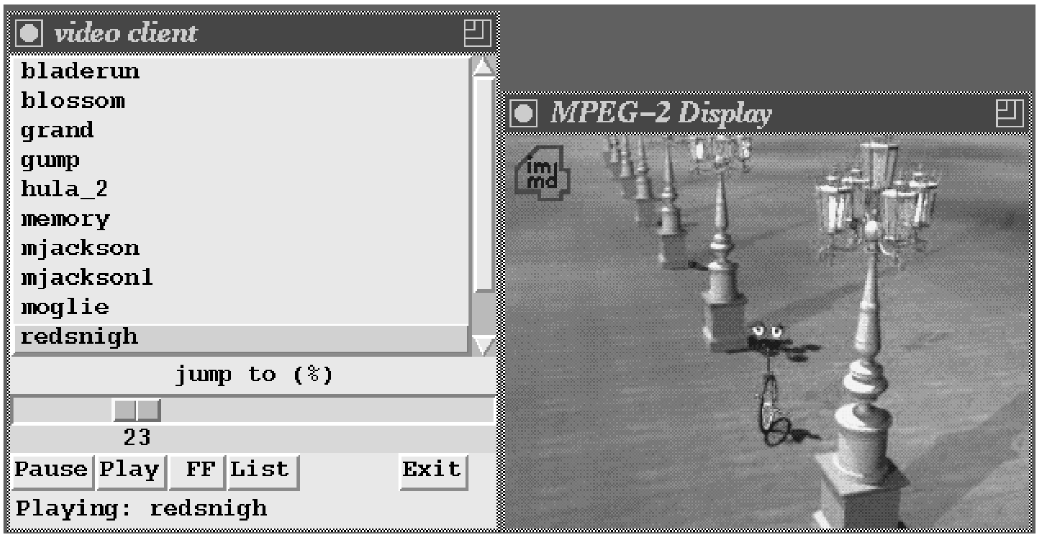


Figure 1: The VoD Client's Interface

3.1 Transis Services

Transis [10] is a group communication system that supports efficient and reliable multicast and membership services. The Transis multicast services offer various types of message ordering. For example, the *Causal* multicast service guarantees that the reply to a message is never delivered before the message itself at any target. The *Total* multicast service extends the *Causal* service in such a way that all messages are delivered in the same order at all their targets.

Transis allows the processes to be easily arranged into multicast groups. A multicast group is identified by the *logical name* assigned to it when the group is created. Each message targeted to the group's logical name is guaranteed to be delivered to all the currently connected and operational group's members. This allows to handle a set of processes as a single logical connection. Furthermore, processes may dynamically join or leave these groups. Transis provides the group members with indication of the current *membership*: the group of currently connected processes. The membership reports are delivered within the flow of regular messages.

3.2 Exploiting ATM Networks

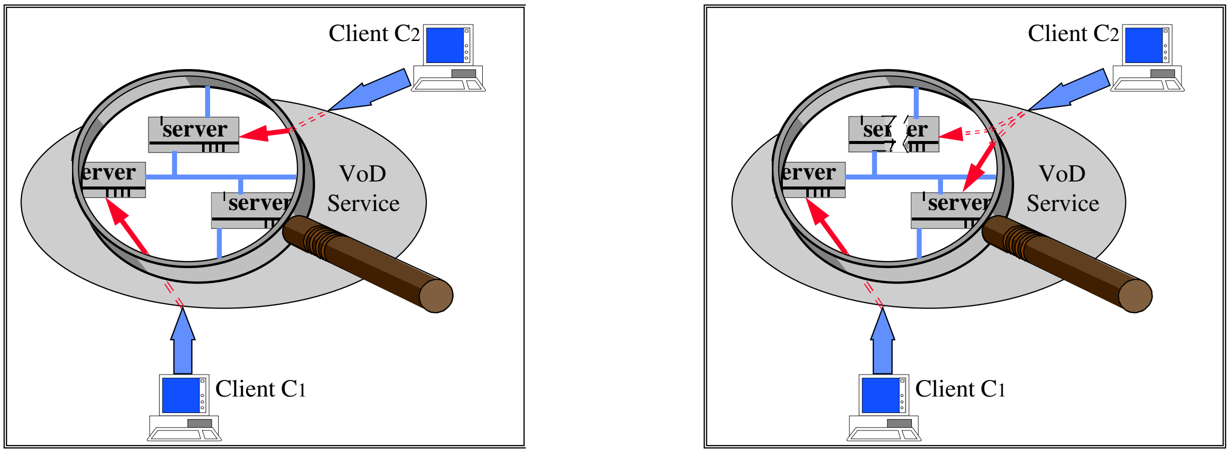
We intend to use our *Multimedia Multicast Transport Service* [9], that incorporates multiple *quality of service (QoS)* options into the group communication framework. The MMTS will open native ATM UNI 3.1 [3] or UNI 4.0 [5] connections for QoS, and will run the Transis system over classical UDP/IP with LAN emulation over ATM (LANE) [4]. We also intend to exploit the CONGRESS group resolution service [2], as a basis for membership and failure detection in the MMTS.

4 Service Design: Exploiting Group Communication

The VoD service exploits the Transis [10] group communication system for fault tolerance, load balancing, connection establishment, negotiation of *Quality of Service (QoS)*, and video control. It exploits QoS communication for video transmission. In this section we describe the service design. The full details of the VoD implementation may be found in [20].

When a client wants to connect to the service, it sends a message to an abstract *service group*. The servers respond by sending a list of available films. The client sends the name of the selected film to the service group. One of the servers that hold this film responds and forms a two-way connection with the client: the server transmits video material, and the client sends control messages for flow control purposes as well as for speed control and for random access within the movie.

Each film is replicated at a subset of the servers. For each movie, the servers that hold the movie are organized in a *movie group*. The members of this group share a database of clients' states. The state includes the movie that the client is watching, the offset in the movie stream and the transmission rate: a total of a few dozens of bytes. Each server multicasts the states of its clients to the rest of the group using the Transis reli-



(a) Before the Server Failure

(b) Recovery from the Server Failure

Figure 2: Transparent VoD Service

able multicast service, approximately every half a second. Thus, the storage space and bandwidth required for this information is negligible w.r.t. the buffer space and bandwidth required for the video transmission[2]. Furthermore, the Transis multicast service is idle while no messages are sent. Hence the total overhead required for fault tolerance is very small.

When a server crashes or detaches, Transis notifies the members of the movie group that remain connected of the change in the group membership. The remaining servers continue to serve the clients of the crashed server. Each client is served by exactly one server, which is chosen according to the load distribution, deduced from the clients' states. The protocol exploits Transis semantics[3] to guarantee that all the remaining servers see the same picture of the load distribution. The client continues communicating with the same abstract connection, and is oblivious to the change, as demonstrated in Figure 2. A similar process occurs when a new server joins, or when the servers decide to migrate clients because the load is poorly distributed.

When the servers migrate a client, a jitter occurs in the transmission of the video stream to this client[4]. In order to guarantee smooth video display at such times, the client maintains a buffer of future frames. The flow control mechanism attempts to keep enough frames in the buffer to account for jitter periods.

The client program consists of three threads. One thread handles the *graphical user interface (GUI)*, and is implemented using TCL/TK. Another thread implements a *communication* module, which receives frames from the VoD server and stores them in a shared buffer. The communication module also sends control messages to the server. The GUI thread forwards control commands to the communication thread.

The third thread is the *MPEG player*, which consumes frames from the shared buffer, streams them into an MPEG decoder, and displays them on the screen. The MPEG decoder is implemented in software, and we are currently porting it to exploit a hardware MPEG decoder. The hardware MPEG decoder generates an analog signal for an external video displayer, which in our case is a video capture hardware for displaying the video in a window. The client design is depicted in Figure 3.

5 Conclusions

In this paper, we presented a scalable fault tolerant Video on Demand (VoD) service. The fault tolerance is achieved practically for free: it consumes negligible bandwidth, storage space, and cpu time. The concepts presented in this paper may be exploited in a highly available VoD service for hotels and for residential neighborhoods. Such services will flourish in the near future, due the increasing improvement in communication technology and infrastructure. Existing Video on Demand systems rarely address the issues of server failures and scalability, which are key issues in multi-user failure-prone networks such as neighborhood servers or the Internet.

Our service exploits the concept of MMTS [9] which incorporates QoS communication within the frame-

[2] The acceptable quality for video playback encoded using MPEG-1 requires at least 1.5Mbit/s.

[3] This is guaranteed by the virtual synchrony [15] property provided by Transis.

[4] The jitter period depends on the level of synchrony among the servers.

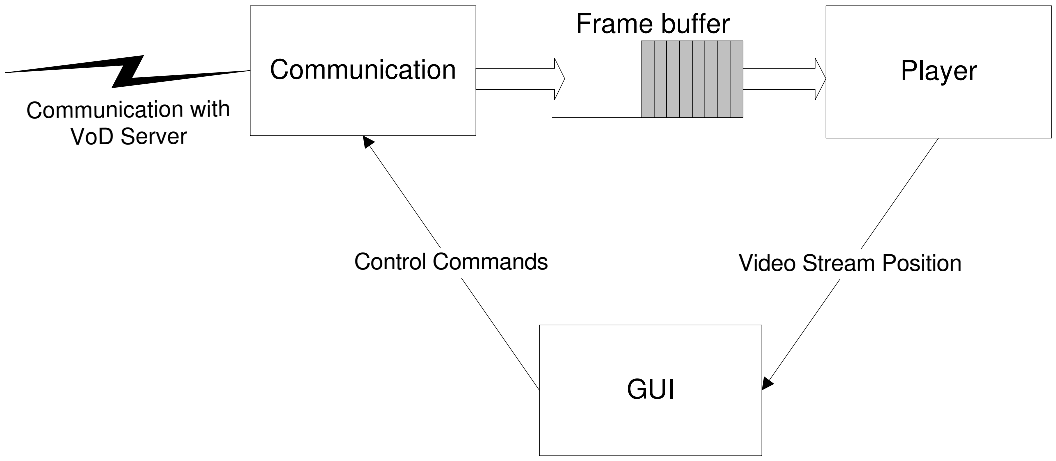


Figure 3: The VoD Client's Multi-Threaded Design

work of group communication. The group communication system provides the following advantages:

1. The **group abstraction** simplifies connection establishment and allows for a transparent migration of clients, while maintaining simple client design. The clients are oblivious to the number and identity of the servers providing the service.

2. The **reliable group multicast semantics** allow the servers to share information which allows them to consistently agree about client migration.

3. The **membership services** detects conditions for client migrations, both for re-distributing the load, and for guaranteeing fault tolerance. The membership changes are reported to all the members in a consistent way.

4. Using **reliable messages**, we guarantee that client control messages will reach the server.

The use of group communication greatly simplifies the service design. The VoD service was implemented using the Transis group communication system. The server was implemented in C++, with around 2000 lines of code, and the client (excluding the display module) was implemented in C, using only 1300 lines of code. Without the Transis services, such an application would have been far more complicated, and the code size would have turned out significantly larger.

Acknowledgments

We are grateful to our advisor Prof. Danny Dolev who made this project possible. Special thanks to Mordechai Keidar for his helpful comments.

References

[1] ACM. *Communications of the ACM 39(4), special issue on Group Communications Systems*, April 1996.

[2] T. Anker, D. Breitgand, D. Dolev, and Z. Levy. Congress: CONnection-oriented Group-address RESolution Service. Tech. Report CS96-23, Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel, December 1996. Available from:
`http://www.cs.huji.ac.il/labs/transis/`.

[3] ATM Forum. *ATM User Network Interface (UNI) Specification Version 3.1.* Prentice Hall, Englewood Cliffs, NJ, June 1995. ISBN 0-13-393828-X.

[4] The ATM Forum. *LAN Emulation Over ATM Specification - Version 1.0*, February 1995.

[5] The ATM Forum Technical Committee. *ATM User-Network Interface (UNI) Signalling Specification Version 4.0, af-sig-0061.000*, July 1996.

[6] The ATM Forum Technical Committee. *Audiovisual Multimedia Services: Video on Demand Specification 1.0, af-saa-0049.000*, January 1996.

[7] S. Berson, L. Golubchik, and R. R. Muntz. Fault Tolerant Design of Multimedia Servers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 364–375, May 1995.

[8] K. Birman, R. Friedman, and M. Hayden. The Maestro Group Manager: A Structuring Tool For Applications With Multiple Quality of Service Requirements. Technical report, Dept. of Computer Science, Cornell University, Ithaca, NY 14850, USA, February 1997.

[9] G. Chockler, N. Huleihel, I. Keidar, and D. Dolev. Multimedia Multicast Transport Service for Groupware. In *TINA Conference on the Convergence of*

Telecommunications and Distributed Computing Technologies, September 1996. Full version available as Technical Report CS96-3, The Hebrew University, Jerusalem, Israel.

[10] D. Dolev and D. Malki. The Transis Approach to High Availability Cluster Communication. *Communications of the ACM*, 39(4), April 1996.

[11] R. Haskin and F. Schmuck. The Tiger Shark File System. In *Proceedings of IEEE Spring COMPCON*, Feb. 1996.

[12] M. Hayden and R. van Renesse. Optimizing Layered Communication Protocols. Technical Report TR96-1613, Dept. of Computer Science, Cornell University, Ithaca, NY 14850, USA, November 1996.

[13] ISO/IEC 13818 and ISO/IEC 11172. *The MPEG Specification*. `http://www.mpeg2.de/`.

[14] S. McCanne and V. Jacobson. Vic: A Flexible Framework for Packet Video. In *Proceedings of ACM Multimedia*, pages 511–522, November 1995.

[15] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended Virtual Synchrony. In *International Conference on Distributed Computing Systems*, number 14th, June 1994.

[16] M. Rautenberg and H. Rzehak. A Control System for an Interactive Video on Demand Server Handling Variable Data Rates. In *Interactive Distributed Multimedia Systems and Services (IDMS)*, pages 265–276, March 1996. LNCS 1045.

[17] I. Rhee, S. Cheung, P. Hutto, and V. Sunderam. Group Communication Support for Distributed Multimedia and CSCW Systems. Technical report, Dept. of Mathematics Computer Science, Emory University, Atlanta, GA 30322, October 1996. To appear in ICDCS'97.

[18] L. A. Rowe, K. D. Patel, B. C. Smith, and K. Liu. MPEG Video in Software: Representation, Transmission, and Playback. In *High Speed Networking and Multimedia Computing, IS&T/SPIE Symp. on Elec. Imaging Sci. & Tech.*, February 1994.

[19] L. A. Rowe and B. C. Smith. A Continous Media Player. In *Int. Workshop on Network and OS Support for Digital Audio and Video*, number 3, pages 334–344, November 1992.

[20] M. Rozman and J. Wexler. A Distributed Fault-Tolerant Video-On-Demand Service. Lab project, High Availability lab, The Hebrew University of Jerusalem, Jerusalem, Israel, December 1996. Available from: `http://www.cs.huji.ac.il/labs/transis/`.

[21] B. C. Smith. *Implementation Techniques for Continous Media Systems and Applications*. PhD thesis, University of California at Berkeley, 1994.

[22] C. Topolcic. *Experimental Internet Stream Protocol: Version 2 (ST-II)*, October 1990. Internet RFC 1190.

[23] W. Vogels and R. van Renesse. *Support for Complex Multi-Media Applications using the Horus system.* Ithaca, NY 14850, USA, December 1994. On-line html document: `http://www.cs.cornell.edu/Info/People/rvr/papers/rt/novsdav.html`.

[24] K. H. Wolf and M. W. K. Froitzheim. Interactive Video and Remote Control via the World Wide Web. In *Interactive Distributed Multimedia Systems and Services (IDMS)*, pages 91–104, March 1996. LNCS 1045.

[25] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. In *IEEE Network*, September 1993. The RSVP Project home page: `http://www.isi.edu/div7/rsvp/rsvp.html`.