32

Providing a Scalable Video-on-Demand System using Future Reservation of Resources and Multicast Communications

Abdelhakim Hafid

Computer Research Institute of Montreal
Telecommunications and Distributed Systems Division
1801, McGill College avenue, #800, Montreal, Canada, H3A 2N4
Phone: (514) 840-1234, Fax: (514) 840-1244, E-mail: ahafid@crim.ca

Abstract

In a video-on-demand (VoD) system the user can select and play movies according to his/her own quality of service (QoS) requirements; upon receipt of the user request, a typical VoD system checks whether there are enough available resources to deliver the requested movie to the user's host. If the response is yes, the movie presentation can start; otherwise, a rejection is sent back to the user; this means that the response is based only on the system's load at the time the request is made and assumes that the service duration (movie length) is infinite. In this paper we propose a scalable VoD (SVoD) system which decouples the starting time of the service from the time the service request is made and requires that the duration of the requested service must be specified. In response to a user request, SVoD determines the QoS with which the movie can be presented at the time the service request is made, and at certain later times carefully chosen. As an example, if the requested QoS cannot be supported at the time the service request is made, SVoD allows to compute the earliest time, when the user can play the movie with the desired QoS; this time can also be determined in a way to use multicast communication to deliver the same movie to several users. The scalability achieved by SVoD is quantified and compared with that of typical VoD; the performance quantification, performed through the use of simulations, shows that SVoD achieves high resource utilization and decreases notably the blocking probability of user requests.

1 INTRODUCTION

A video-on-demand (VoD) system allows users to select and play movies according to his/her own quality of Service (QoS) requirements (Brubeck and Rowe 1996; Buddhikot et al. 1994; Gemmel et al. 1995; Lougher 1993; Rangan 1993).

Building QoS into Distributed Systems A. Campbell & K. Nahrstedt (Eds.)

Typically, when a VoD system receives a user request to play a movie with a certain QoS, it checks whether there are available resources to open one or more streams to deliver the movie. If the response is yes, the movie presentation can start; otherwise, a rejection is sent back to the user. This implies that a second attempt of the user cannot take advantage of information obtained through the first request to change, if possible, the requirements to fit the current system load; a user who tries several times to play the movie during a short period of time, e.g. 5 minutes, without success will be certainly discouraged and will likely look for another provider. Furthermore, a user can be rejected at the time he/she made his/her request while system resources become available just a short time, e.g. 1 minute, after the user made his/her request. This can bring unpleasantness to users and loss of revenues to service providers. This is due to the fact that existing VoD systems provide the user with the service that can be supported at *the time the service request is made*, and assume that the service is requested for *indefinite duration*. We believe that such systems do not fit the needs for future multimedia (MM) service providers and users.

In this paper we present a scalable video-on-demand (SVoD) system which decouples the starting time of the service from the time the request (to play a movie) is made and requires that the duration of the requested movie must be specified. SVoD is based on the ideas behind the negotiation approach with future reservation (NAFUR) which is described in (Hafid et al. 1997). In response to a service request, NAFUR allows to compute the presently QoS available and at certain future times. Certain of these future times may be times which correspond to the starting times of the same service requested by other users for whom the resources are already reserved. The user has the choice to start: (a) immediately with the presently QoS available; or (b) in a later time with the requested QoS and likely with less money to pay since only a part of the required resources should be reserved (the other part is already reserved for other users);

The scalability of SVoD is achieved through the use of future reservations of the system's resources and the use of multicast communications to deliver the same movie to several users.

The paper is organized as follows. Section 2 presents an overview of NAFUR. Section 3 describes SVoD architecture. Section 4 presents the simulation model used to quantify the performance of SVoD and VoD. The simulation results are presented in Section 5. Finally, Section 6 concludes the paper.

2 AN OVERVIEW OF NAFUR

NAFUR is a new QoS negotiation approach that decouples the starting time of the service from the time the service request is made and requires that the duration of the requested service must be specified. NAFUR allows to compute the QoS that can be supported for the time the service request is made, and at certain later times carefully chosen. As an example, if the requested QoS cannot be supported for the time the service request is made, the proposed approach allows to compute the earliest time, when the user can start the service with the desired QoS. NAFUR is designed to be used by any distributed system requiring negotiation with the user (human or not), e.g. video-on-demand and teleconferencing systems.

For sake of simplicity and clarity we model a distributed system as a single module and a QoS manager (Figure 1) which represents the access point to the module. The ability of the QoS manager to process a service request is directly related to the admission criteria which the QoS manager uses to decide whether a new request is accepted or not. This criteria is that the sum of previously assigned resources plus the resources required by the new request should not exceed the resources of the system. In the case of acceptance, the QoS manager reserves the required resources. When the QoS manager receives a request to terminate the service it simply de-allocates the reserved resources. When a renegotiation request is received, the QoS manager may decrease the amount of the reserved resources if the new QoS is less restrictive than the QoS currently provided, otherwise, it checks the admission criteria with the new QoS constrained by the current load of the system. In this section, the terms "system" or "QoS manager" will be used interchangeably.

More specifically, the following operations are provided by the QoS manager to the users (Figure 1):

(1) ServiceInq (in *req*: Request; *resPeriod*: Time; out *pro*: Proposals);

(2) ServiceRes (in *req*: Request; out *s*: Status);

The operation ServiceInq makes an inquiry about the availability of a particular service characterized by a request *req*. This parameter is a tuple of three elements $req = <Q, starttime, length>$, where Q is a set of QoS parameters characterizing the quality of the requested service, *starttime* is the desired starting time for the service, and *length* is the length of the time for which the service is requested; the period for which the service is requested is therefore the time interval $[starttime, starttime + length]$. The result *pro* is a set of proposals where each proposal indicates the QoS available for a period of *length* at some future times. Formally, a proposal is defined as a tuple $<time, QoS>$ where *QoS* represents the QoS that can be supported by the system over the interval $[time, time + length]$.

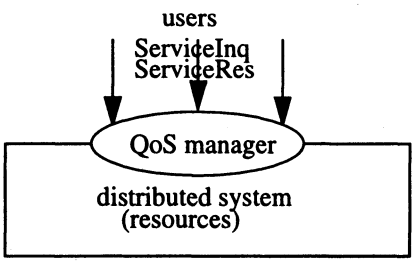

Figure 1. System model

The parameter *resPeriod* (argument of ServiceInq) indicates for how long the service reservations (made to support *pro*) should be kept (on a temporary basis) until a subsequent invocation of the ServiceRes operation will make an effective reservation for a particular proposal. The operation ServiceRes is used to effectively make a service reservation. Typically, it will be called after the execution of a ServiceInq operation, and its *req* parameter will correspond to one of the proposals

contained in pro. The status parameter, s, indicates the success or failure of the operation.

To support the operation ServiceInq, we assume that the QoS manager has enough knowledge about the available QoS, presently and in the future; this information is called available service projection (asp). Formally asp consists of a list of tuples $[(time_1, QoS_1),..., (time_i, QoS_i), ...]$ where QoS_i corresponds to the QoS that can be supported by the component at time $time_i$.

For each tuple, $<t_i , QoS_i>$, of the available service projection (ASP), the QoS manager checks whether QoS_i holds for a period of time, equal or higher than the length, $length$, of the requested service. If the response is yes, then $<t_i , QoS_i>$ may be considered as a potential proposal; otherwise, the QoS manager will consider the tuples which have their time values smaller than $t_i + length$ and higher than t_i, to compute the minimum QoS which might hold for a period equal to $length$, starting from t_i. The QoS manager produces a list of proposals; however, not all the proposals are useful to be presented to the user. A kind of *filtering* is supported by the QoS manager to compute only the representative ("useful") proposals (Hafid et al. 1997).

NAFUR provides suitable mechanisms to encourage the system resource sharing using multicast communication. Upon receipt of a service request, the QoS manager computes the presently QoS available and at certain future times. Certain of these future times may be times which correspond to the starting times of the same service requested by other users for whom the resources are already reserved. When the QoS manager returns the proposals, if the user selects a proposal for which the required resources (or part) are already reserved (for another user), only a part (or none) of the required resources are reserved.

A more detailed description of NAFUR for an arbitrary system topology and the underlying mathematical theory of NAFUR can be found in (Hafid et al. 1997).

3 SYSTEM ARCHITECTURE

Figure 4 presents the architecture of our SVoD; each video server (respectively host machine) is extended with a agent, we call server agent (respectively user agent); these agents implement a basic version of NAFUR as described below. The architecture shown in Figure 4 is essentially independent from the technologies and software in use; this does not mean that the agents have the same implementation code. Rather, an agent offers an interface which provides a certain number of standard operations, but the implementation of these operations depends on the component, e.g. its technology and the software it supports.

SVoD can be easily extended with new video servers without code modification of the existing agents; one has only to implement the agents to be installed in the new servers. It is obviously imperative that an agent communicates with the server where it is hosted; access primitives allow agents to use abstraction as long as the components agree on the basic language of access. However, a server is free to

implement an access primitive in whatever way it sees fit. In this paper we focus on the operation of a single video server.

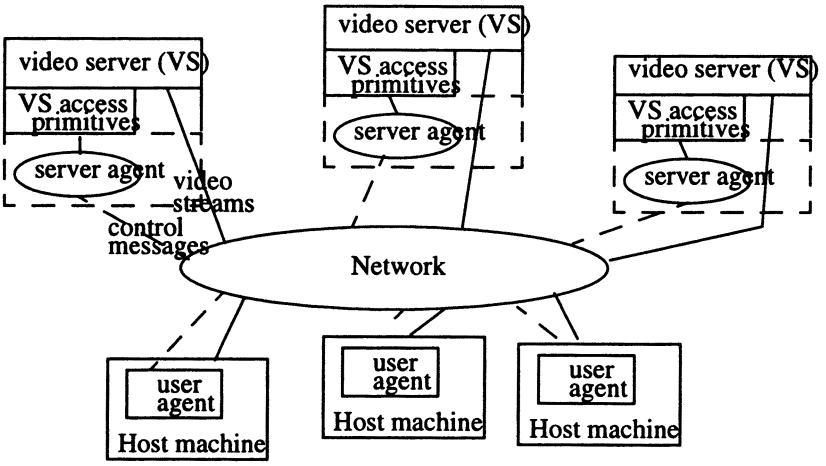

Figure 4. Architecture of SVoD

2.2. User agent

The user agent consists mainly of three components: user interface, QoS manager, and SVoD client controller (Figure 5). The user interface is mainly divided into two parts. The first part offers a means to search and select a movie from a database; this means that a video server is selected to deliver the movie. The second part allows to specify the desired presentation quality and cost constraints; it also allows users to renegotiate the desired QoS during the movie presentation.

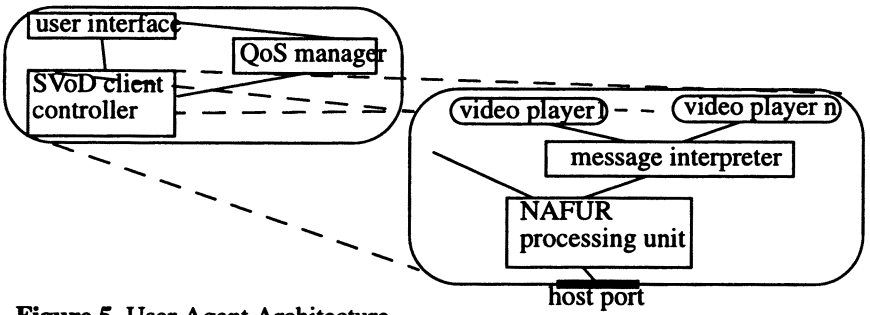

Figure 5. User Agent Architecture

When the user selects a movie and specify his/her requirements, the user interface sends a message to the QoS manager. The latter starts by checking whether the client machine characteristics, such as the screen size and the screen color, support the requested QoS. If the client machine does not support the QoS requested by the user, the QoS manager sends a rejection (with an offer) to the user via the user interface; the user has three choices: abandon the request, accept the offer, or initiate a renegotiation. Otherwise, the QoS manager sends a message (which contains the user requirements) to the SVoD client controller. Then, the latter builds a message, we call simply request, and sends it to the corresponding server agent; it

enters a loop waiting (at a specific host port) for a response and initiates a timer.

If a time-out is reached before the reception of a response, a rejection is sent to the user via the user interface; otherwise, the proposals contained in the received message (which corresponds to the response of the server agent) are presented to the user, via the user interface by "NAFUR processing unit". If the latter accepts one the proposals, then SVoD client controller invokes (immediately or in later time for delayed presentation) the appropriate video player to display the movie; this is done via a "message interpreter" which maps the message received via the host port to a primitive to start the corresponding video player. Otherwise, a rejection message is sent to the server agent.

2.2 Sever agent

A video server in our system can be any kind of video server which has the functionality (1) to provide delayed service provision (future reservation of resources) when there are not enough resources to support the delivery of the requested movie; and/or (2) to serve several users with a single video stream. Ideally, this functionality should be inherent to the server which means that the server integrates this functionality as part of its software; this means that the server implements a subset of the suite of algorithms provided by NAFUR. However, any existing video server (research server, e.g., the continuous media file server described in (Neufeld et al. 1996), or commercial server, e.g. VDOLive On-Demand (VDOnet 1996)) can be used if it installs a server agent (see Figure 4).

At any time, a server agent is waiting at a specific server port for requests from user agents. Each time it receives such a request, it stores information (of interest) related to this request (Figure 6), it updates the information it has, and it forwards the request to the server. Then, the server checks its capacity to deliver the requested movie with the desired QoS and sends the response to the server agent; we assume that the server starts the delivery of the movie only after a confirmation.

information_List

movie identifier	starting time	movie length	quality of service
"Casablanca"	20:00	1,5 hour	TV quality

Figure 6. Information maintained by the server agent

If the server's response is a rejection, the server agent checks (in its information_List) whether the delivery of the requested movie is scheduled for another user; if the response is no, the server agent determines the time when there will be enough resources to support the request (when one or more current or lately scheduled presentations end). In both cases a message is sent to the user agent; the message contains proposals for delayed presentation of the requested movie. Then, the server agent enters a state waiting for a confirmation message from the user agent.

If the server's response is an acceptance, then the server agent checks the possibility (in the future) of using multicast communication to deliver the requested

movie; if the response is no, the server's agent sends an acceptance message to the user agent; otherwise, a message that contains two proposals (actual and delayed presentation using multicast) is sent to the user agent. In both cases, the server agent enters a state waiting for a confirmation

The network in our system should support the multicast facility; otherwise, SVoD will make use only of future reservation of resources facility. Obviously, if there are not enough available network resources, the delivery of the movie cannot be performed; we assume that the network has (at any time) available resources to deliver the movie from the server to the user. However, some protocols can be used to check the availability of network resources; for example, RSVP (Zhang et al. 1993) or the Tenet Protocol suite (Ferrari and Verma 1990) can be used for network resources reservation for immediate movie presentation; for delayed presentations, the network should be capable of reserving its resources in advance.

4 MODEL DESCRIPTION

To evaluate the performance of our system we performed a number of simulation experiments. More specifically, we run three classes of simulations:

(1) Simulations without future resources reservation facility nor multicast communication

(2) Simulations with future resources reservation facility

(3) Simulations with future resources reservation facility and multicast communication.

The service request to play a movie is defined by $<id, Q, starttime, length>$ (see Section 2).

4.1 Simulation parameters

The simulations are parameterized by the following:

- *User request type:* this process is used to model the class of QoS the clients will ask for. We have assumed that we have three classes 1 , 2 , and 3 that correspond to Q_1 , Q_2 , and Q_3 respectively. Each class is characterized by the amount resources to reserve in order to support this class.The users will request the more popular class, 1 , more often. The probability that a user requests the class i is given by p_i. The following service request type pattern is assumed: $p_1 = 0.8$, $p_2 = 0.1$, and $p_3 = 0.1$.

- *Number of users making requests (NU):* the number of users making requests over a given period of time; we consider a population of 400-1900 users.

- *User Request pattern in time:* indicates the distribution of user requests over a day; this distribution presents a peak during the evening when most users likely ask to play a movie. A normal distribution, characterized by its mean ($\lambda = 3.5$) and its variance ($\sigma = 60$), is selected to model the evolution of this parameter.

- *Length of the service requested (LSR):* this process is used to model the generation of the lengths, *length* , associated with the service requests. We assume that the length associated to a service request is uniformly (randomly) distributed between

60 min and 90 min.
- *Maximum delay parameter (MDP):* indicates the maximum difference (between the time the request is made and the starting time of a delayed presentation) which is acceptable by the user; a value of 0 for this parameter means that the user does not accept any delayed presentation of the requested movie.
- *Movie selection pattern:* indicates how users select one of the available movies (e.g. 50 different movies). We assume that most popular movies are the most requested; the following is a default selection pattern: 80% of users selects the five most popular movies ($i = 1, ..., 5$); 15% of users selects the 25 less popular movies ($i = 6, ..., 25$); 5% selects the 20 least popular movies ($i = 26, ..., 50$);

4.2 Performance measures

The main metric we adopted for evaluation and comparison was the *rejection probability*: *rejection probability*= (number of rejections)/(number of service requests); a rejection corresponds to a rejection initiated by the system or a rejection initiated by the user who does not accept a delayed presentation (in this case *MDP* is smaller than the difference between the time of the request and the time of the future proposal returned by NAFUR).

Simulation Hypothesis
We assume that (1) the system is characterized by its maximum capacity R; and (2) Q_1, Q_2, and Q_3 can be supported if 0.0050% of R, 0.0070% of R, and 0.0030% of R is available, respectively.

5 SIMULATION RESULTS

We study the impact of 4 parameters on the performance of our SVoD. These parameters are: the time the requests are made, the number of users in the system (NU), maximum delay parameter (MDP), the amount of the system resources.

All results are compared to a typical VoD system. For figures 8 through 12, curves denoted by U correspond to user requests issued; curves denoted by S correspond to SVoD without multicast; curves denoted by SM correspond to SVoD with multicast; and curves denoted by V correspond to typical VoD.

5.1 System behavior over time

In Figures 8 the X-axis indicates the time in minutes (e.g. if 0 represents 17:00, then the peak of user requests is around 20:00); the Y-axis indicates the number of requests made and accepted in the last 20 minutes. The figure shows the user requests made over time, and the number of requests accepted by the system when using a typical VoD, SVoD without multicast facility, and SVoD with multicast facility. For this experiment we assume that the number of users is 1000, and users do not accept presentations which are delayed more than 1 hour (60 minutes).

Figure 7 shows that the number of accepted user requests is much higher with SVoD than VoD; particularly, this holds during the peak of user request distribution (between 120 and 280).

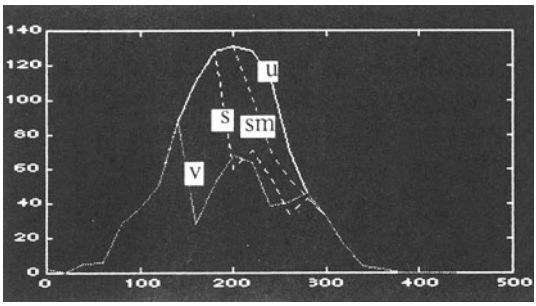

Figure 7. User requests issued and user requests accepted

The number of requests made and the number of streams started in the last 20 minutes are shown in Figure 8. A large number of requests rejected using VoD are scheduled for future presentations using SVoD. SVoD is much better at handling a large number of requests made over relatively short period of time.

It is obvious that SVoD with multicast facility performs much better than SVoD without such a facility; the use of multicast communication is the best way to serve more users in a video-on-demand system than the approach of one-to-one connections between the users and the system. This is due to the sharing of (server and network) resources between a number of users asking for the same movie

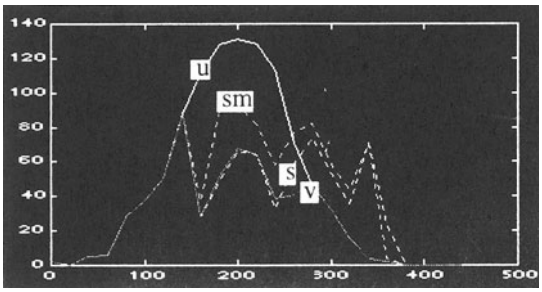

Figure 8. User requests issued and users served

5.2 Maximum delay parameter

In Figure 9, the X-axis indicates the maximum delay parameter; the Y-axis indicates rejection probability multiplied by 100. The figure shows that the blocking probability decreases when the value of the maximum delay parameter (MDP) increases; this holds when using SVoD since VoD is not affected by varying MPD. The impact of MPD is most significant when using SVoD with multicast facility. One may argue that it is not realistic to assume that users will accept delayed presentations with a delay of 30 or 60 minutes; nevertheless, we believe that users will prefer to get a feedback from the system about the status of their request. NAFUR provides the feedback on the time the requested presentation can start and the quality of the presentation. We think that a good number of users will be attracted by

money discounts for delayed presentations. Last and not least, users may book in advance in order to obtain a reservation.

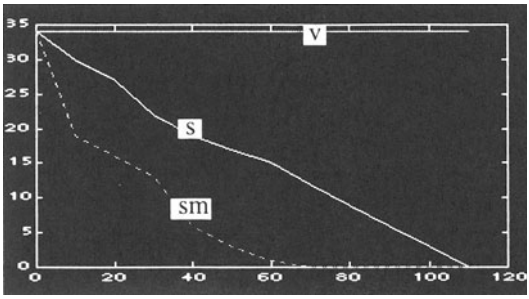

Figure 9. Blocking probability Vs maximum delay parameter

5.3 Number of users

In Figure 10, the X-axis indicates the number of users making requests; the Y-axis indicates rejection probability multiplied by 100. The figure shows the impact of increasing the number of users requests. Blocking probability increases when the number of users increases for VoD and SVoD. The best gain using SVoD is achieved when the number of users is around 800-900; the blocking probability with SVoD is around 0-5 and the blocking probability with VoD is around 30-35%. When the number of users is more than 1000, the gain achieved by SVoD (without multicast facility) slightly decreases to stabilize; while the blocking probability with SVoD (with multicast facility) increases slowly than the blocking probability of others.

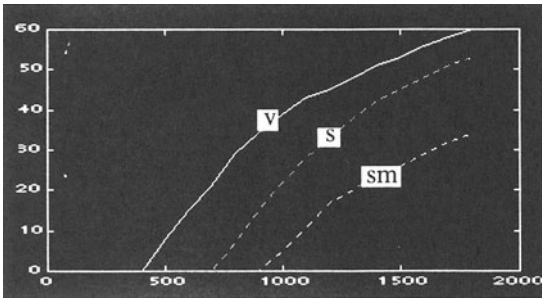

Figure 10. Blocking probability Vs number of users

5.4 Amount of system resources

In Figure 11, the X-axis indicates the capacity of the system in terms of % of R (e.g. R, 1.5*R, 2*R); the Y-axis indicates rejection probability multiplied by 100. The impact of increasing the maximum amount of the system resources is shown in Figure 11. For this experiment the number of user requests is 1500, and the value of the maximum delay parameter is 60 minutes.

The blocking probability of SVoD and VoD decreases when the amount of the system resources increases. However, the blocking probability decreases faster

than the blocking probability of VoD. Thus, in this experiment if we want to have SVoD with 0% blocking probability we have to increase its maximum amount of resources (R) by 0.65*R.

If the designer's of SVoD knows about the average number of potential users with the distribution of their requests over time, he/she can build a system with an average 0 as a blocking probability.

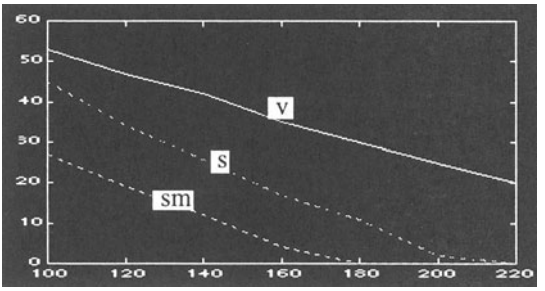

Figure 11. Blocking probability Vs amount of system resources

6 CONCLUSION

Upon receipt of a user request to play a movie, a typical video-on-demand (VoD) system does start the movie presentation if there are enough available resources; otherwise, a reject message is sent to the user. Since no more information is sent back to the user, the latter may try several times to get the movie without success (e.g., the server is loaded at maximum for a certain period of time).

In this paper we have proposed a scalable video-on-demand system (SVoD) which uses a future reservation of resources and multicast communications. When SVoD receives a user request (with some QoS requirements) and has not enough resources to support it, it does compute the QoS that can be supported immediately and the exact time in the future when the user can play the requested movie. SVoD schedules future presentations in a way to make extensive use of multicast communications. That is, if a presentation of a certain movie is scheduled in some time in the future, say T, and one or more users ask to play the same movie at $T_1 \leq T$, SVoD will schedule the presentations for these users (after their confirmation) to start by T; SVoD will use multicast communication to deliver this movie at T and thus will make use of less much resources than using one-to-one connection with the users.

A performance analysis of SVoD shows that we can service more users than typical SVoD systems. This allows us to create a system that can service a large number of users for a reasonable cost. We can state that SVoD is more efficient and flexible than typical VoD systems. Furthermore, the ideas behind SVoD can be easily used to enhance the functionality of existing commercial and research VoD systems.

7 REFERENCES

Brubeck, D.W. and Rowe, L.A. (1996) *Hierarchical Storage Management in a Distributed Video-On-Demand System.* IEEE Multimedia, Vol. 3, No. 3, pp. 37-47.

Buddhikot, M., Parulkar, G., Cox, J. (1994) *Design of a Large Scale Multimedia Storage Server.* Journal of Computer Networks and ISDN Systems, Elsevier (North Holland), December.

Ferrari, D. and Verma, D. (1990) A scheme for Real-Time Channel Establishment in Wide-Area Networks. IEEE JSAC, Volume 8, No3

Gemmel, D. et al. (1995) *Multimedia Storage Servers: A Tutorial.* IEEE Computer, Vol. 28, No. 5

Hafid, A., Bochmann, G.v., Dssouli R. (1997) *A Negotiation Approach with Future Reservation (NAFUR): A Detailed Study.* Computer Networks and ISDN Systems (to appear)

Lougher, P. (1993) *The Design of a Storage Server for Continuous Media*, Ph.D. Thesis, Lancaster University, UK

Neufeld, G., Makaroff, D., Hutchison, N. (1996) *Design of a Variable Bit Rate Continuous Media File Server for an ATM Network.* Proceedings of IS&T/ SPIE'96, San Jose, California

Rangan, V. (1993) *Architecture and Algorithms for Digital Multimedia On-Demand Servers.* Performance 93 and Sigmetrics 93, L.Donatiello and R.Nelson eds.

VDOnet Corp (1996) *VDOLive On-Demand.* Http://www.vdo.net/corporate/ awards.html

Zhang, L. et al. (1993) *RSVP: A New Resource Reservation Protocol.* IEEE Network, 7(5), Sept. 1993.

8 BIOGRAPHY

Dr. Abdelhakim Hafid is a Researcher Staff Member at the Computer Research Institute of Montreal (CRIM), Telecommunications and Distributed Systems Division, working in the area of distributed multimedia applications. He received his Masters and Ph.D. degrees in computer science from University of Montreal on quality of service management for distributed multimedia applications in 1993 and 1996, respectively. From 1993 to 1994 he was visiting scientist at GMD-FOKUS, Systems Engineering and Methods group, Berlin, Germany working in the area of high speed protocols testing. His research results have been published in over 30 publications in international conferences, workshops and journals. His current research interests are in broadband multimedia services and communications.