THE PERFORMANCE OF QUEUING THEORETIC VIDEO ON DEMAND ALGORITHMS

BOURAS C.(1)(2), GAROFALAKIS J.(1)(2), PETRIDIS P.(1), TZIMAS G.(1)(2),

(1)*Department of Computer Engineering and Informatics,*
University of Patras, 26500 Patras, Greece

(2)Computer Technology Institute,
P.O. Box 1122, 26110 Patras,Greece

KEYWORDS

Video On Demand (VOD), Performance of Algorithms, Simulation, Modeling

ABSTRACT

Video On Demand (VOD) Systems comprise an emerging new technology, which is expected to significantly affect everyday life. The implementation of effective commercial VOD systems has to overcome some technical limitations, such as the low bandwidth offered by the existing communication channels. In our work we present some simple queuing theoretic algorithms for the optimal use of a limited number of communication channels keeping a fair policy for the users requests. We compare our algorithms with well-known algorithms, by means of simulation.

1. INTRODUCTION

Based on the advances of the telecommunication technology as well as on the huge progress within electronic industry, in the last few years several software intensive systems have been developed providing real-time or near real-time provision of services. One of these services that is based on state-of-the-art technologies is *Video On Demand* (VOD).

A Video On Demand System provides on demand choice of movies to a set of users. The only difference between a VOD and a VCR (VideoCassette Recorder) is that the user doesn't have VCR equipment or any videotape. The movies are stored in a central video server, which is connected through a high-speed network with local service provision nodes. Every local node is connected through co-axial cables with the user homes providing on demand transmission of TV programme.

Depending on the interaction degree, a VOD system can be categorized as follows [1]:

- *Quasi-Video-on-Demand*: In this case the users are grouped based on their interests. A user can choose between different programmes by changing user teams.
- *Near-Video-on-Demand*: The same programme is retransmitted in fixed time slots thus enabling functions such as forward and reverse play.
- *True-Video-on-Demand*: The user has full control over the presentation of the programme. The T-VOD fully simulates a VCR enabling functions such as forward and reverse play, freeze and random positioning.
- *Adaptive-Video-on-Demand*: In this case the user submits a request for a movie and the decision is made through a routing algorithm.

In figure 1 the physical architecture of a VOD system is depicted [1]:

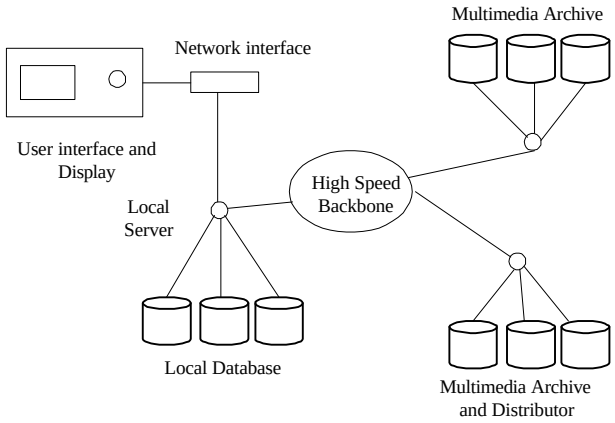


Figure 1: The physical architecture of a general VOD system.

As we can see a local database and a local video server, which is connected by a high bandwidth network with the user's homes, composes a VOD system. The user interacts with the system by using a common computer keyboard. A data file is transmitted to the local nodes where it is cached and forwarded to the users.

The following features characterize the above model:

- It enables a distributed implementation thus providing higher reliability and availability.
- The users can access the information by the local nodes ensuring a lower cost.
- Easier management because of the distributed nature of the system.
- The system is easily expandable.

In this paper we propose four new queuing algorithms that improve the performance of Adaptive Video on Demand (A-VOD) systems. In section 2 we give the current state-of-the-art on VOD algorithms. In section 3 we present our approach for the improvement of the performance of A-VOD systems. In section 4 we describe the simulation process and in section 5 we present our results and comparison with the known algorithms. Finally, in section 6 we present the conclusions of our work.

2. KNOWN ALGORITHMS FOR VIDEO ON DEMAND

In figure 2 we present the model of a VOD system [2]:

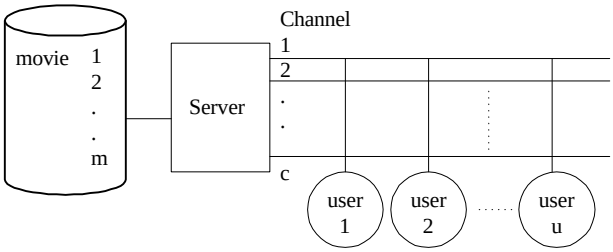


Figure 2: The model of a VOD system.

One of the biggest problems in the implementation of a VOD system is the distribution of the available resources. In the above architecture there are two main problems:

1. The limited number of transmission channels within a co-axial cable that is used for many users.
2. The number of files that the server can transmit concurrently [3].

Let M be the set of m movies, U the set of u users, C the set of c channels, T the duration of a movie (it is the same for all the movies), ô the maximum delay between a request and the beginning of a job and n the maximum delay between a request and the corresponding response (n≤ô≤Ô). In our case we assume that n=ô. We also define m=u/c. The request is characterized by the triple: time of the request, the user and the job and the response is characterized by the time (≤time of the request+n), the user, the channel and the service time (time≤service time≤time of the request+ô). When the service time is negative, the request is rejected.

The decision for the acceptance or rejection of a request is made through a routing algorithm. Here we present two well-known routing algorithms used for VOD systems.

The Harmonic Algorithm [2]

The C available channels are divided in m sets S_1, S_2,…, S_m. In every set S_i, $\frac{C}{i * H_m}$ channels are made available. Where H_m is the harmonic number defined as follows $H_m = \sum_{i=1}^{m} \frac{1}{i}$. Every channel within S_i serves at least i requests concurrently. The *Harmonic Algorithm* decides for the acceptance of the requests as follows: Every job m_j corresponds with a queue Q_j in which the requests for the specific job are stored. At the beginning the queue is empty for every j. When a request is made for a job m_j it is stored in Q_j. The queues aim is to concentrate as many requests as possible in order for them to be served in one transmission. A time start(Q_j) corresponds with every Q_j that is the arrival time of the first request in the queue. At the time start(Q_j)+n the algorithm decides whether it is going to serve the k requests. If there is a free channel in the S_i set with i≤k the transmission of the m_j movie takes place and serves the k requests. In the case that there isn't any free channel with i≤k the algorithm rejects the requests started at start(Q_j) and start(Q_j) initializes based on the earliest request within Q_j.

The SF (Serve a Fraction) Algorithm [4]

We suppose that the notification time and the movie duration are related linearly, that is n=T/l, where l is a constant. The *SF Algorithm* is executed as follows: the time is divided in predefined time slots of n length. The requests are queued in every time slot and in the end of it the decision is made based on the following criterion. In every movie m_j corresponds a weight w_j. The movies with higher weights are served first. As weight we can define the probability of a movie to be selected. In every time slot the SF algorithm provides at a maximum C/l channels for the transmission of the jobs.

3. SIMPLE QUEUING THEORETIC ALGORITHMS FOR VIDEO ON DEMAND

In this section we present four new algorithms based on Queuing Theory for VOD systems. These algorithms try to improve the performance of VOD systems and to establish a new approach in this field.

Every job m_j is related with a queue Q_j in which the requests for the specific job are queued. At the beginning the queue is empty for every j. When a request is made for a job m_j it is stored in Q_j. The queues aim is to concentrate as many requests as possible in order for them to be served in one transmission. A time start(Q_j) corresponds with every Q_j that is the arrival time of the first request in the queue.

The FCFS (first come first serve) algorithm

The FCFS algorithm is based on the following approach: Whenever one of the C channels is free, it transmits the m_j movie that corresponds to queue Q_j for that j that has minimum start(Q_j), that is the queue that has the earliest request is served first. In the case that there isn't any free channel at start(Q_j)+n the algorithm rejects the requests started at start(Q_j) and start(Q_j) initializes based on the earliest request within Q_j.

The MRM (maximum requests for movie) algorithm

In the case of the MRM algorithm we have the following: Whenever one of the C channels is free, it transmits the mj movie that corresponds to queue Qj for that j that we have max|Qj|, that is the queue that has the maximum number of requests is served first. In the case that there isn't any free channel at that moment the Qj's requests made at start(Qj) are rejected and start(Qj) initializes based on the earliest request within Qj.

The FCFSN (first come first serve after N minutes) algorithm

At the time start(Qj)+n the algorithm decides to serve the Qj requests for that j that has minimum start(Qj), that is the queue that has the earliest request is served first. If there is a free channel the movie mj is transmitted and the requests are served. In the case that there isn't any free channel at that moment the Qj's requests made at start(Qj) are rejected and start(Qj) initializes based on the earliest request within Qj.

The MRMN (maximum requests for movie after N minutes) algorithm

At the time start(Qj)+n, for that j that has minimum start(Qj), the algorithm decides to serve the requests in the Qk queue where we have max|Qk|, that is the queue that has the maximum number of requests is served first. If there is a free channel then the movie mk is transmitted. In the case that there isn't any free channel or j≠k, at that moment the Qj's requests made at start(Qj) are rejected and start(Qj) initializes based on the earliest request within Qj.

4 THE SIMULATOR

In what follows we present the simulator [5] implemented for the verification of the results and the comparison between the various algorithms. The model simulated can be seen in the following diagram:

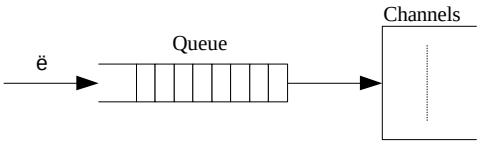


Figure 3: The Simulators model

In order to implement the general model we made the following assumptions:

- *We have a queuing system with C servers with the following functionality*: The time between successive arrivals of requests follows exponential distribution with mean value ë. The model implemented follows the Zipf distribution [6] with parameter è=0.271 where $P(X=i)=\frac{c}{i^{(1-J)}}$. The queue has a Q length and when it is not empty we decide to serve several requests based on a specific algorithm and one of the C channels is reserved.
- *The events characterising our model are the following:*
  1.The arrival request
  2.The departure of served requests
- Our input is a sequence of requests for a set of M jobs that will be served through the C channels. The requests are sorted based on their arrival time and a request is dropped if it hasn't been served for a time period n.
- The number of users is u=T/ë where ë is the mean time between the arrivals and Ô is the movie length.

The input parameters are the following: T: the jobs duration, ë: the time between successive arrival requests, N: the maximum delay between a users request and the systems response, C: the number of channels, M: the number of jobs, Q: the maximum queues length, and m: the number of sets that the C channels will be divided in the case of the harmonic algorithm

The performance metrics that were calculated during the simulation are the following:

ô: The mean delay for the served requests

q: The mean length of the queue

r_s: The total number of served requests

r_l: The total number of dropped requests

r_{ms}: The number of served requests per job

r_{ml}: The number of dropped requests per job

c_s: The number of requests served by every channel

p: The percentage of the channels usability

The parameters used for the execution of the simulation are listed below: Movie_time = 120 min., simulation_end = 10080 min, movie_set = 100 min

L	Number of Users	Load
0.1	1200	Small
0.05	2400	Moderate
0.025	4800	Big

Also, n takes the values 5, 10, 15 min and the channel size (CS) takes values from the following list: 50, 100, …, 1000 having a 50 step. The simulation was executed for 1080 times (180 times for each algorithm).

Some indicative graphs generated by the simulator for the various cases of algorithms are shown below. The following abbreviations are used within the graphs: L: mean interarrival, N: notification time, CS: channel size, PRS: probability request served, MNSRC: mean number of served requests for channel and MPCU: mean percent channel was used.

Figure 1: L = 0.1 - n = 5

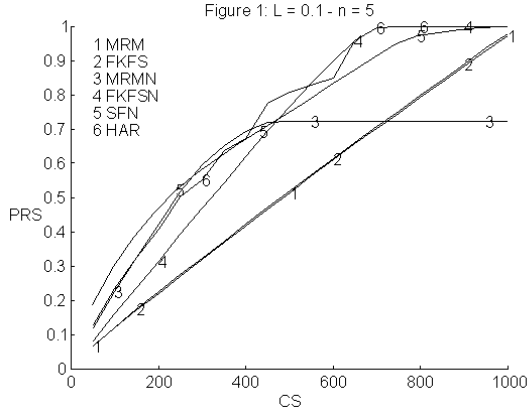


Figure 2: L = 0.1 - n = 10

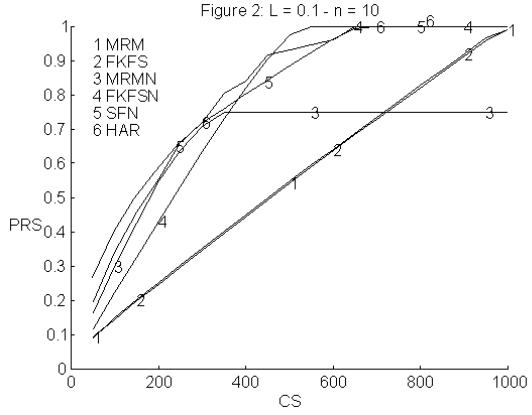


Figure 3: L = 0.1 - n = 15

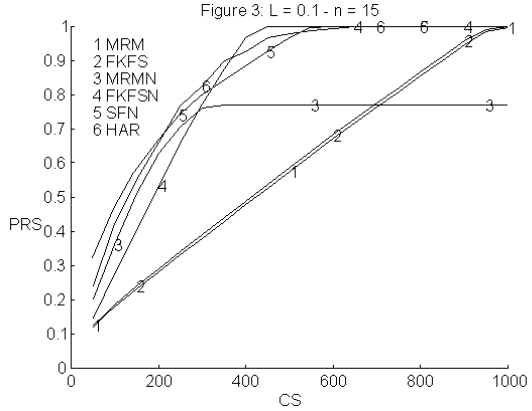


Figure 4: L = 0.05 - n = 5

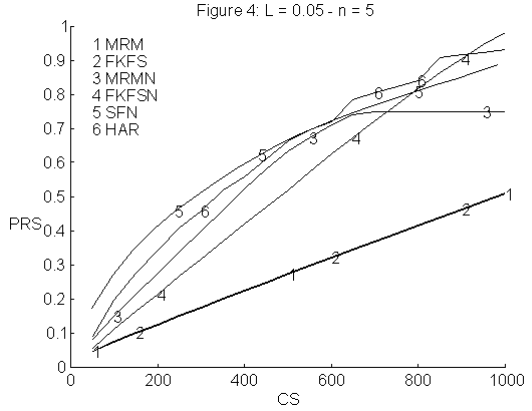


Figure 5: L = 0.05 - n = 10

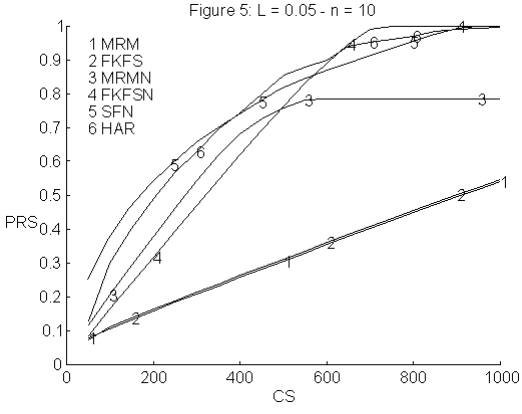


Figure 6: L = 0.05 - n = 15

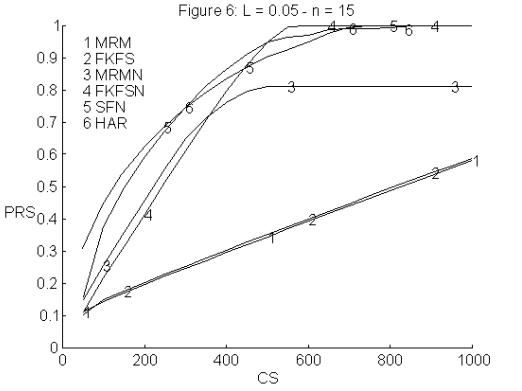


Figure 7: L = 0.025 - n = 5

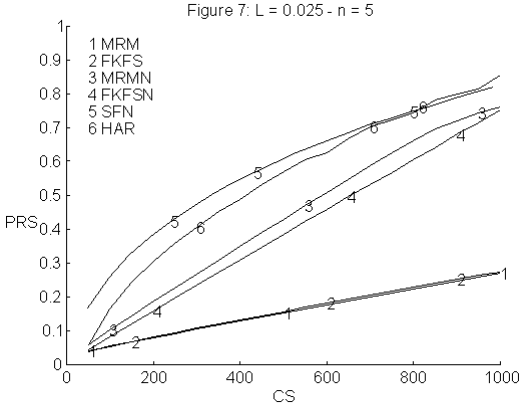


Figure 8: L = 0.025 - n = 10

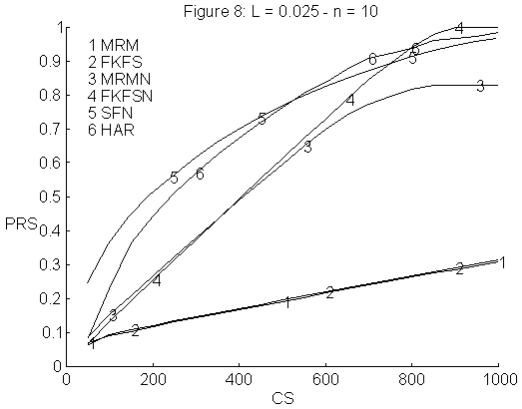


Figure 9: L = 0.025 - n = 15

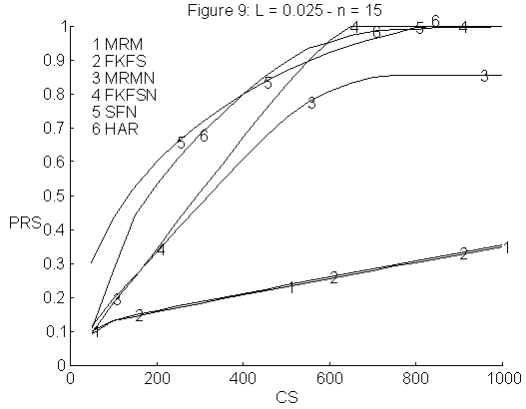


Figure 1: L = 0.1 - n = 5

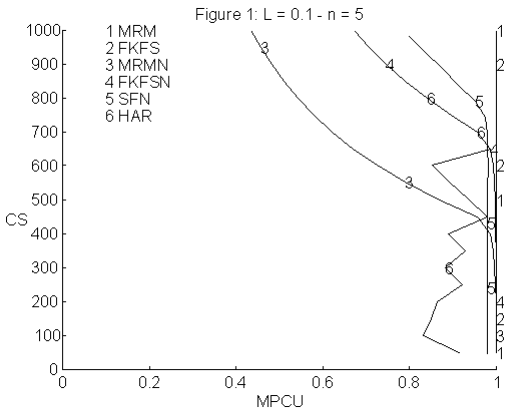


Figure 2: L = 0.1 - n = 10

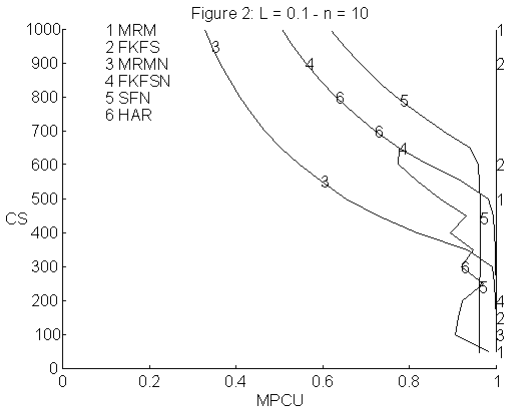


Figure 3: L = 0.1 - n = 15

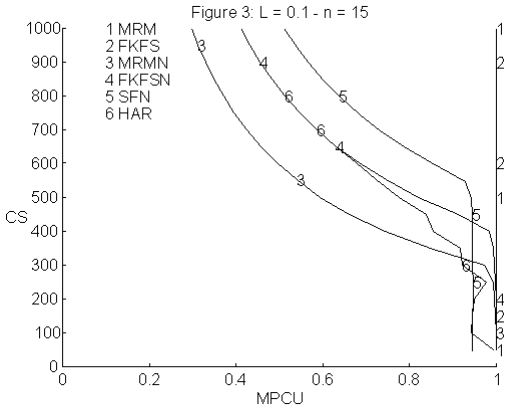


Figure 4: L = 0.05 - n = 5

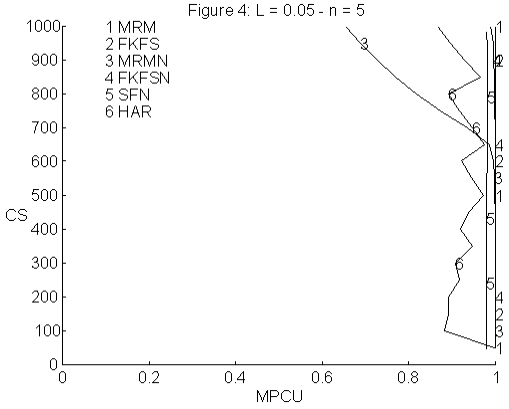


Figure 5: L = 0.05 - n = 10

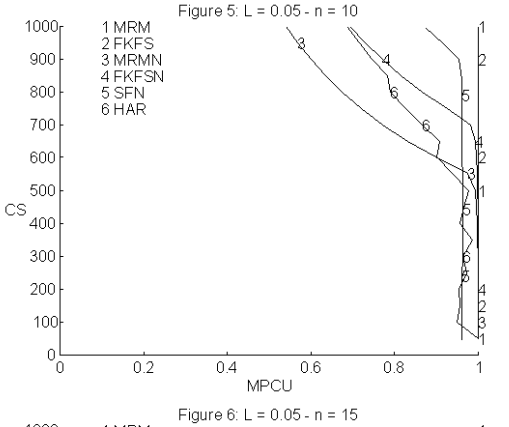


Figure 6: L = 0.05 - n = 15

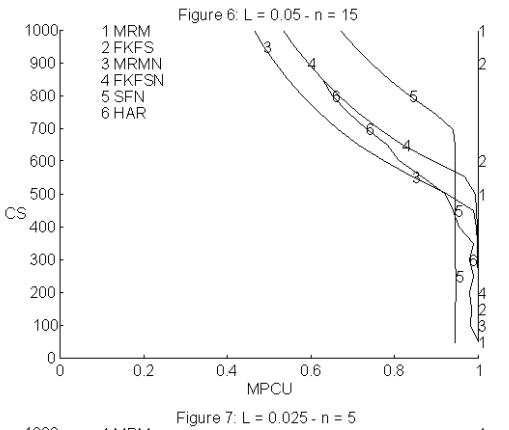


Figure 7: L = 0.025 - n = 5

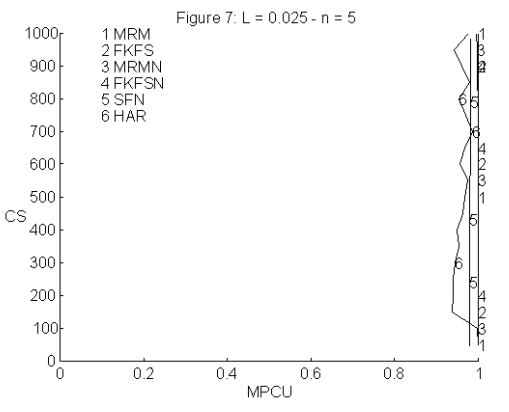


Figure 8: L = 0.025 - n = 10

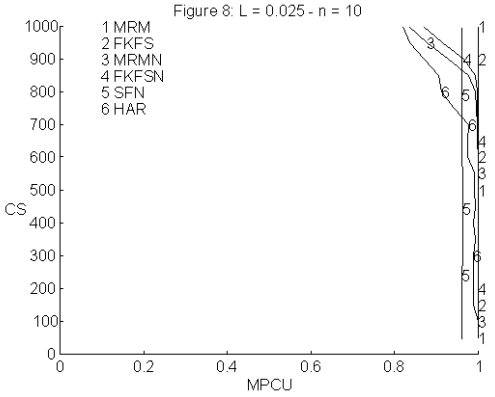


Figure 9: L = 0.025 - n = 15

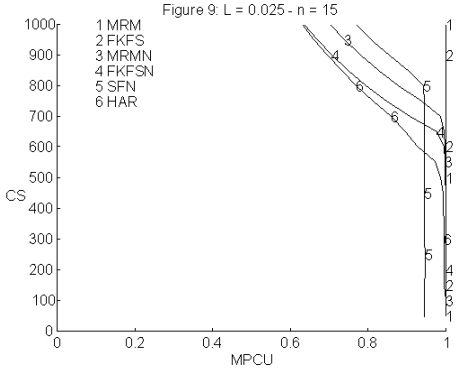


Figure 1: L = 0.1 - n = 5

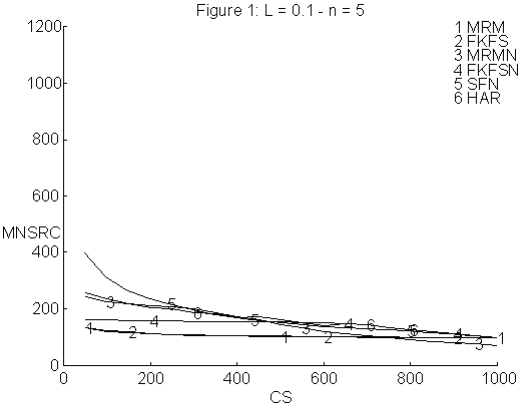
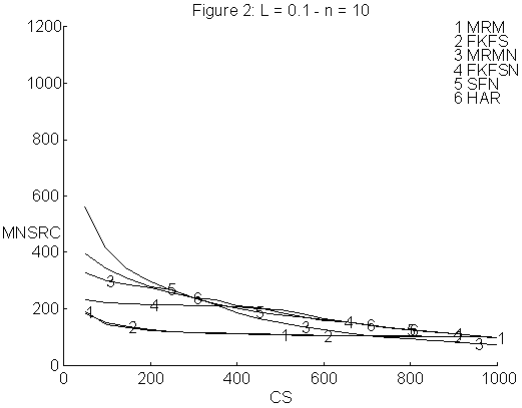


Figure 2: L = 0.1 - n = 10



5. RESULTS AND COMPARISONS

In the next table we can see the results for the six algorithms simulated having as performance metric the probability to serve a request.

In spite of the simple model that we used to simulate the VOD algorithms our final results are very interesting:

1. In the case that we have more channels available than the minimum number of channels needed to achieve serving probability which approaches the maximum value that can be achieved by the specific algorithm, the redundant channels are not used and therefore the mean utilization of the channels is decreased. When we increase the number of available channels then the algorithms SF , Harmonic , FCFSN , FCFS , MRM can always achieve maximum serving probability equal to one. Algorithm MRMN can approach the maximum serving probability that is equal to one, only if we increase the notification time concurrently with the number of available channels.
2. The algorithms FCFSN and FCFS achieve the best behavior by means of serving fairness.
3. For all the cases the increase of n has as a result the increase of serving probability but it also increases the mean serving delay.
4. The MRM and FCFS do not have significant differences due to the fact that they serve when a free channel exists.
5. The MRMN algorithm behaves well only in the case that there are a few channels compared to the load. On the contrary the FCFSN algorithm is a better choice in the case of a large number of channels.
6. The SF and Harmonic algorithms have a better behavior in comparison to the other algorithms but they have increased mean delay serving time. The SF has better channel utilization compared to Harmonic but in the case that n and CS take large values Harmonic is better.
7. The FCFSN algorithm, although less complex than algorithms such as SF and Harmonic, is more fair since it serves the request depending on the waiting time and not on the job popularity.
8. The MRMN behaves well only when there are limited resources, thus making us to try to improve the VOD system performance by serving the most popular jobs request.
9. The SF and Harmonic algorithms are more stable independently of the number of requests. Also, the existence of the notification time improves the performance of an algorithm and is valuable in the case of jobs that have an increased probability [7] to be selected.

Load	*Notification Time*	*Number of Channels*	*Results*
Small	Small	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Small	Small	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Small	Small	Big	1. Harmonic 2. SF 3. FCFSN 4. FCFS 5. MRM 6. MRMN
Small	Moderate	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Small	Moderate	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Small	Moderate	Big	1. Harmonic 2. SF 3. FCFSN 4. FCFS 5. MRM 6. MRMN
Small	Big	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Small	Big	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Small	Big	Big	1. Harmonic 2. SF 3. FCFSN 4. FCFS 5. MRM 6. MRMN
Moderate	Small	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Moderate	Small	Moderate	1. Harmonic 2. SF 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Moderate	Small	Big	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Moderate	Moderate	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Moderate	Moderate	Moderate	1. Harmonic 2. SF 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Moderate	Moderate	Big	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Moderate	Big	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Moderate	Big	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Moderate	Big	Big	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Big	Small	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Big	Small	Moderate	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Big	Small	Big	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Big	Moderate	Small	1. SF 2. Harmonic 3. MRMN 4. FCFSN 5. FCFS 6. MRM
Big	Moderate	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Big	Moderate	Big	1. FCFSN 2. Harmonic 3. SF 4. MRMN 5. FCFS 6. MRM
Big	Big	Small	1. SF 2. Harmonic 3. FCFSN 4.MRMN 5. FCFS 6. Mtm
Big	Big	Moderate	1. Harmonic 2. SF 3. FCFSN 4. MRMN 5. FCFS 6. MRM
Big	Big	Big	1. FCFSN 2. Harmonic 3. SF 4. MRMN 5. FCFS 6. MRM

Table 1: Comparison based on the serving probability

6. CONCLUSIONS

As we have seen all the algorithms have relatively similar performance if the available resources are satisfying compared to the expected incoming request stream. If the number of incoming requests (load) is higher than expected the SF and Harmonic algorithms are more stable, without meaning that the FCFSN or MRMN algorithms could not be used as an alternative.

In the future we plan to extend our work for the case of distributed VOD servers, where the movies are distributed among them. Moreover, it would be interesting to find analytical solutions for the problems we have formulated and extend the results.

7. REFERENCES

[1] A.D. Gelman, H. Kobrinski, L.S. Smoot, S.B. Weinstein, M. Fortier, D. Lemay, A Store and Forward Architecture for Video on Demand Service, *Proc. IEEE ICC*, 1991, 27.3.1-27.3.5.

[2] S. Aggatwal, J.A. Garay, A. Herzberg, Adaptive Video on Demand, *3rd Ann. European Symposium on Algorithms*, 1995, 538-553.

[3] Steven M. McCarthy, Integrating Telco Interoffice Fiber Transport with Coaxial Distribution, *Proc. SPIE-Int. Soc. OPT. Eng.*, 1993, 1786: 23-33.

[4] Bar-Noy, J.A. Garay, A. Herzberg, Shating Video on Demand – Constant Competitive Ratio with Long Notification Time, *Electronic Engineering Times*, 1993.

[5] Avetill M. Low, W. David Kelton, *Simulation Modeling and Analysis*, Mc Graw-Hill Inc., 1991.

[6] A, Dian, D. Sitiram, P. Shahabiddin, Scheduling Policies for an On-Demand Video Server with Batcing, *Electronic Engineering Times*, 1993, 72.

[7] C. Bouras, V. Kapoulas, T. Pantziou, P. Spirakis, Randomized Adaptive Video on Demand, Personal Communicatio, *Short Abstract appeared in Proc 15th Annual ACM Symp. on the Principles of Distributed Computing*, 1996