A Scalable Delivery Framework and a Pricing Model for Streaming Media with Advertisements*

Musab Al-Hadrusi and Nabil J. Sarhan

Department of Electrical & Computer Engineering
Wayne State University
Detroit, MI 48202

ABSTRACT

This paper presents a delivery framework for streaming media with advertisements and an associated pricing model. The delivery model combines the benefits of periodic broadcasting and stream merging. The advertisements' revenues are used to subsidize the price of the media content. The pricing is determined based on the total ads' viewing time. Moreover, this paper presents an efficient ad allocation scheme and three modified scheduling policies that are well suited to the proposed delivery framework. Furthermore, we study the effectiveness of the delivery framework and various scheduling polices through extensive simulation in terms of numerous metrics, including *customer defection probability*, *average number of ads viewed per client*, *price*, *channel utilization*, *arrival rate*, *profit*, and *revenue*.

Keywords: Media streaming, periodic broadcasting, pricing, scheduling, stream merging.

1. INTRODUCTION

The interest in media streaming has grown dramatically over the Internet, cellular networks, and Cable TV. Unfortunately, the distribution of streaming media faces a significant scalability challenge because of the high server and network requirements. Therefore, numerous techniques have been proposed to deal with this challenge, especially in the areas of *resource sharing*) and *request scheduling*. Resource sharing techniques can be classified into *client-pull* and *server-push* techniques, depending on whether the channels are allocated on demand or reserved in advance, respectively. The first category includes *stream merging* techniques,[5,8,11,15,17,18] which reduce the delivery cost by aggregating clients into larger groups that share the same multicast streams. The degrees of resource sharing achieved by client-pull media delivery techniques depend greatly on how the waiting requests are scheduled for service. The second category consists of *Periodic Broadcasting* techniques, which[12–14,16,21] divide each media file into multiple segments and broadcast each segment periodically on dedicated server channels. They are cost-performance effective for highly popular content but lead to channel underutilization when the request arrival rate is not sufficiently high.

The overwhelming majority of prior studies on media delivery focused on regular content without any media advertisements. The use of ads is important for many reasons, including the following. (1) Ads generate revenue, which can be used to pay for the service cost, generate profit, or subsidize the paid content. (2) A streaming solution that supports ads can essentially convert passive startup waiting times for service to active waiting times (i.e., watching ads while waiting for the playback of the desired media content). Today, even for short videos with medium quality, users of online video websites may experience significant delays. The transition, in the near future, to streaming long videos (such as full-length movies) at high quality may lead to even longer delays. (3) Many users like to watch some types of ads, such as movie trailers, to know about other interesting movies to watch. (4) Using ads results in better aggregation of the requests and thus can reduce the delivery costs.

This paper presents a framework for scalable delivery of media content with advertisements including a pricing model. The delivery framework combines the benefits of stream merging and periodic broadcasting. A client starts by joining an ads' broadcast channel for some time and then receives the requested media by stream merging. We discuss two ad delivery options: *Partial-OK* and *Only-Full*. With Partial-OK, a client can join and leave the ads' channel at any time and thus may watch some ads partially. With Only-Full, however, a client joins the ads' channel only at the beginning of an ad and leaves it at the end of an ad. Moreover, we propose a heuristic ad allocation algorithm that arranges ads efficiently to reduce the average request waiting time for starting to receive the ads. The revenues generated from the ads are used

*This work is supported in part by NSF grant CNS-0626861. A preliminary version with partial results to be presented as a poster in ACM Multimedia.

to subsidize the price. Subsidizing the price helps attract more clients, thereby increasing the overall revenue.[3] Pricing depends on the experienced quality-of-service (QoS). In particular, clients with larger ads' viewing time get lower prices. Since the price has to be set before the client makes a selection of the media content to playback, waiting-time prediction[2] can be used to estimate the ads' viewing period based on the requested media and the system's current state. Furthermore, this paper presents three modified scheduling policies for the proposed delivery framework.

We study the effectiveness of various scheduling policies and the two ad delivery options with both Patching and ERMT through extensive simulation. The considered performance metrics include *customer defection* (i.e., turn-away) probability, *average ads' viewing time*, *price*, *system utilization*, *arrival rate*, *profit*, and *revenue*. Generally, customers are more likely to buy a product or a service if it is offered at a lower price. In this paper, we stress that the arrival rate will not only depend on the price but also the customer defection probability, defined as the probability that customers leave the system without being serviced because of waiting time exceeding their tolerance. The practical importance of the impact of the defection probability on the arrival rate is due to the fact that decreasing the price will not always, in the long run, continue to increase the arrival rate without adequate scaling of system capacity. The defection probability is an important Quality-of-Service (QoS) metric because customers who leave due to excessive delay will not likely return in the future. Note that the waiting time is already factored in the defection probability. We experiment with two different envisioned arrival rate functions and their various parameters.

The rest of the paper is organized as follows. Section 2 discusses background information and related work. Section 3 presents the proposed delivery framework, ad allocation scheme, and scheduling modifications. Subsequently, Section 4 discusses the performance evaluation methodology. The main results are discussed and analyzed in Section 5.

2. BACKGROUND INFORMATION

2.1 Resource Sharing

Resource sharing techniques[5–8,10,11,13,18,19] face the scalability challenge of multimedia streaming servers by utilizing the multicast facility. Stream merging techniques combine streams when possible to reduce the delivery cost. They include *Patching*[6,11] and *Earliest Reachable Merge Target* (ERMT).[8,9] Patching expands the multicast tree dynamically to include new requests. A new request joins the latest *regular* (i.e., full) stream for the object and receives the missing portion as a *patch*. Hence, it requires two download channels (each at the video playback rate) and additional client buffer space. When the playback of the patch is completed, the client continues the playback of the remaining portion using the data received from the multicast stream and already buffered locally. To avoid the continuously increasing patch lengths, regular streams are retransmitted when the required patch length for a new request exceeds a pre-specified value called *regular window* (Wr). By contrast, ERMT is a near optimal hierarchical stream merging technique. It also requires two download channels, but it makes each stream sharable and thus leads to a dynamic merge tree. A new client joins the closest reachable stream (*target*) and receives the missing portion by a new stream. After the merger stream finishes and merges into the target, the later can get extended to satisfy the playback requirement of the new client(s), and this extension can affect its own merge target.

Whereas stream merging techniques deliver data in a *client-pull* fashion, periodic broadcasting techniques[10,12,14] employ the *server-push* approach. In particular, they divide each supported video into multiple segments and broadcast them periodically on dedicated channels. Thus, they can service unlimited numbers of customers but can be used only for the most popular videos, and they require customers to wait until the next broadcast times of the first segments. Moreover, server channels may become underutilized when videos are requested infrequently.

2.2 Request Scheduling

A video streaming server maintains a waiting queue for every video and applies a scheduling policy to select an appropriate queue for service whenever it has an available *channel*. A channel is a set of resources (network bandwidth, disk I/O bandwidth, etc.) needed to deliver a multimedia stream. All requests in the selected queue can be serviced using only one channel. The number of channels is referred to as *server capacity*.

The main scheduling policies include *First Come First Serve* (FCFS),[7] *Maximum Queue Length* (MQL),[7] *Maximum Factored Queue Length* (MFQL),[1] and *Minimum Cost First* (MCF).[20] MCF achieves the best overall performance by capturing the significant variation in stream lengths caused by stream merging techniques through selecting the requests requiring the least cost. The length of a stream (in time) is directly proportional to the cost of servicing that stream since

the server allocates a channel for the entire time the stream is active. *MCF-P (RAP)* is the preferred implementation of MCF. It selects the queue with the least cost per request and treats regular streams and transition patches in a preferential manner because they are shared by later patches.

2.3 Using Advertisements

Supporting ads has been discussed in only few studies. In [4], the primary data and ads are delivered using piggybacking on the same channels. Piggybacking adjusts the movie playback rate so that two streams can merge with each other, thereby impacting the playback quality and suffering from technical challenges and complexities to change the movie playback rate. Moreover, ads are inserted randomly multiple times during the playback of the primary media. The study [3] provided a general discussion of pricing based on the techniques in [4]. It discussed the general relationships among arrival rate, price, and ads' ratio (to the total user viewing time).

3. PROPOSED DELIVERY FRAMEWORK AND PRICING MODEL

The proposed delivery framework combines the benefits of stream merging and periodic broadcasting. Periodic broadcasting is used for the ads because they are potentially accessed more frequently than any individual primary media content, especially if each client is required to view a minimum number of ads. The primary media contents, however, are delivered using a scalable stream merging technique and can benefit from a high degree of aggregation because of the use of ads.

The main characteristics of the framework can be summarized as follows. (1) Clients start by joining an ads' broadcast channel for some time and then receive the requested media by stream merging. (2) Ads are combined and broadcast on dedicated server channels. Hence, when beginning listening to an ads' channel, the client views different ads until streaming of the desired media commences. (3) Ads are only viewed prior to watching the actual media content. Uninterrupted viewing of the primary media allows for a more enjoyable playback experience. (4) Scalable stream merging is used for the primary media content. (5) Clients snoop on preceding media streams while viewing the ads so as to reduce the delivery costs for joining the preceding streams. We refer to this feature as *early snooping*. Stream merging techniques require two client download channels at the video playback rate. Thus, while listening to one ads' channel, the client uses the other channel for snooping.

In the pricing model, the price for streaming a media file is determined based on the client's total ads' viewing time. The system should estimate the expected ads' viewing time for each potential request dynamically because the clients need to know the price before purchasing the service. This can be done using waiting-time prediction[2] based on the current state of the system and the requested media file. Hence, the system provides clients with updated menus containing the list of media files and the corresponding expected ads' viewing durations and prices. The revenues generated from the ads are used to subsidize the price. A certain portion can be distributed uniformly among all clients and the rest distributed proportionally to the client total ads' viewing time. Price subsidization attracts more clients and thus increases the overall revenue and profit.

3.1 Ad Allocation and Delivery

Ads are combined and broadcast periodically on dedicated server channels. Hence, when beginning listening to an ads' channel, the client views different ads until streaming of the desired media commences. The number of ads or the total playback duration of unique ads should be large enough to ensure that the same ad is not viewed more than once by the same client when the system is heavily loaded. The ads may or may not be uniform in length. Two options are possible as to whether to allow partial ad-viewing or only full-ad viewing. In the first option, called *Partial-OK*, a client can join and leave the ads' channel at any time and thus may watch some ads partially. With the second option, called *Only-Full*, a client joins the ads' channel only at the beginning of an ad and leaves it at the end of an ad. Thus, with this option, the client may experience a waiting time to begin listening to the ads' channel. The maximum waiting time is equal to the longest ad duration. The Only-Full is more appealing to companies that seek to advertise their products and services but has the disadvantage of introducing a waiting time before viewing the ads. Moreover, even when a channel becomes available during the ad's playback, the streaming of the primary media can begin only after the current ad has finished.

To reduce the waiting time for receiving ads in the Only-Full option, multiple ads' channels can be used with time-shifted versions of the combined ads, as shown in Figure 1. In the figure, a total of four ads are supported and broadcast on three channels. Assuming the ad length is 30 seconds, with these three channels, the maximum time for a new request

to reach the beginning of an ad is $30/3 = 10$ seconds, and the average time is $10/2 = 5$ seconds. With only one channel, the maximum waiting time to reach an ad is 30 seconds. Recall that with the Partial-OK, option only one ads' channel is required because clients view ads immediately without waiting time, except for the initial buffering time to smooth out the delay jitter.

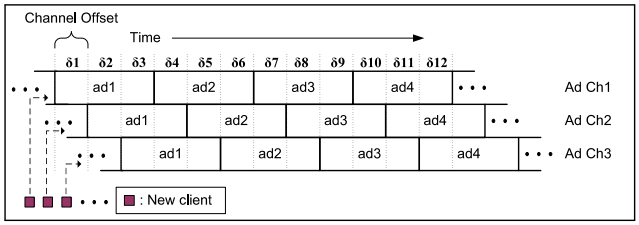


Figure 1: Ads' Broadcast Channels

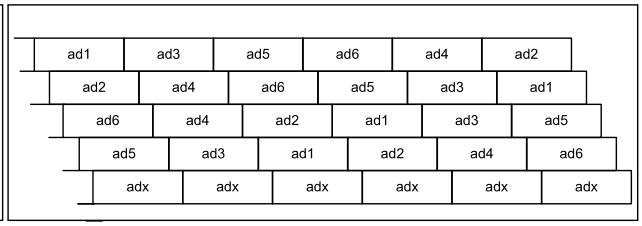


Figure 2: Heuristic Allocation

The ad allocation problem arises in the Only-Full option. It involves two aspects: how to order the ads on each channel and how to determine the offset from one channel to the other, which is simply referred to here as *channel offset*. The channel offset is the time interval between the beginnings of two ads' groups in two successive channels. In Figure 1, the offset = $\delta_1 = \delta_2$. Next, we discuss the allocation problem in the case of uniform ad lengths and variable ad lengths.

3.1.1 Uniform Ad Lengths

The ad allocation problem is simple in the case of uniform ad lengths. It is reduced to only how to determine the offset from one channel to the other. In general, the waiting time is simply the time to reach the beginning of the broadcast of the next closest ad. Thus, the waiting times depends on the spacing (or intervals) between (the beginnings of) successive ads. The maximum waiting time is equal to the length of the largest interval, whereas the average waiting time is the weighted sum of all intervals divided by two. Because of the repeating nature of ads, the average can be found during the time of one ads' group. The weighted sum is required if the intervals are not of equal length. Note that longer intervals should have higher weight because more new requests are likely to be initiated during them. The uniform ad lengths lead to uniform intervals. In Figure 1, the intervals between successive ads are δ_1, δ_1, ..., δ_{12}. These intervals are of the same length and are equal to the maximum waiting time. The average waiting is simply half this maximum waiting time. The first two intervals correspond to the channel offset. The optimal channel offset is basically the offset that leads to uniform intervals between successive ads. It can be shown that the optimal offset can be given by

$$Channel\ Offset = Ad\ Length\ /\ Number\ of\ Channels. \tag{1}$$

3.1.2 Variable Ad Lengths

For variable-length ads, the problem is more challenging. Assuming a fixed order in all channels simplifies the problem but is generally inefficient because it may lead to large waiting times. Thus, a solution that finds the best order of ads in each channel is required. Because the best solution is the one that makes the intervals between successive ads as uniform as possible, the channel offset can be found as follows:

$$Channel\ Offset = Length\ of\ One\ Ads'\ Group\ /\ (Number\ of\ Channels\ \times Number\ of\ Ads). \tag{2}$$

As discussed earlier, the average waiting time is the weighted sum of intervals between successive ads within one ads' group length. Assuming that δ_i is the i^{th} interval, the maximum and average waiting times can determined as follows:

$$Maximum\ Waiting\ Time = \max_{i}^{n} \delta_i, \tag{3}$$

$$Average\ Waiting\ Time = \sum_{i}^{n} (\delta_i/2) \times \frac{\lambda \times \delta_i}{\sum_i^n \lambda \times \delta_i} = \frac{\sum_i^n \delta_i^2}{2 \times \sum_i^n \delta_i}, \tag{4}$$

where λ is the request arrival rate and n is the number of intervals. Note that $\lambda \times \delta_i$ is the number of request arrivals within interval δ_i. Because of the square of δ_i in the equation, reducing the maximum waiting time tends to reduce the average waiting time.

Finding the optimal ad allocations is an NP hard problem. We propose a *heuristic ad allocation algorithm* that achieves a close performance to the optimal solution. The heuristic works as follows for a maximum of five channels. (1) On channel 1, put the longer ads closer to the ends. (2) On channel 2 (if it is not the last channel), reverse the order of channel 1. (3) On channel 3 (if it is not the last channel), put the second half of the the ads of channel 1 first, followed by the ads of the first half of those on channel 1. (4) On channel 4 (if it is not the last channel), reverse first half of ads of channel 1 and then reverse the second half and put them together. (5) For the last channel, compute the best combination leading to the least waiting time given the prior ordering of the other channels. The algorithm can be extended to a larger number of channels, but generally there is no need for larger channels as the decrease in the waiting time would be too small with further channels. Figure 2 clarifies the heuristic allocation. The ads are numbered in decreasing order of length. Figures 3 and 4 show the optimal allocation (found using exhaustive test) and the heuristic allocation, respectively for 4 specific ads and three channels. Note that the average waiting time is only 6.2% longer than the optimal. The average waiting time with the worst allocation (not shown) is approximately 19 seconds, which is about twice that of the optimal.

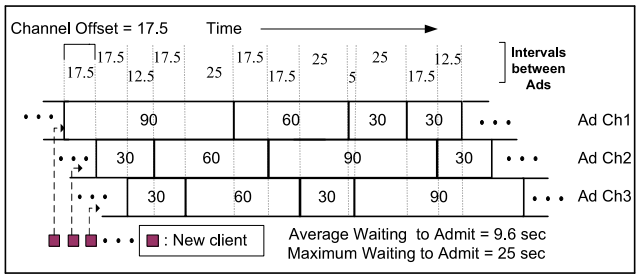


Figure 3: Optimal Ad Allocation

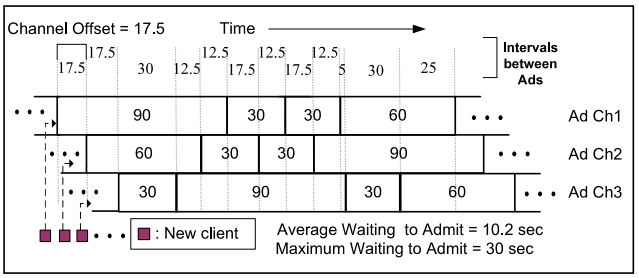


Figure 4: Heuristic Ad Allocation

We have studied the effectiveness of the heuristic algorithm compared with the optimal for different configurations of ad lengths and numbers of ad channels. The results show that the heuristic allocation achieves only 5% longer waiting times on the average. By reordering the channels of the heuristic allocation and finding the best order, the heuristic algorithm produces the optimal allocation in most cases.

3.2 Proposed Scheduling Modifications

Most existing scheduling policies are not well suited for the proposed delivery framework. For example, MCF, MQL, and MFQL attempt to serve requests as soon as possible, thereby reducing significantly the ads' viewing time. Thus, we present two modifications of MCF-P (RAP) to ensure that ads are viewed by a large number of users: *Each N* and *Any N*. *Each N* considers a video for scheduling only if each waiting request for it has viewed at least N ads, whereas *Any N* considers a video for service only if any one of its waiting requests has viewed at least n ads. These modified versions can work with MQL and MFQL but we primarily use them for MCF-P (RAP) because it is the best performer. Moreover, we propose a policy, called *Maximum Ads Time* (MAT), which selects for service the video whose waiting requests have the longest total ads' viewing time.

4. EVALUATION METHODOLOGY AND RESULTS

We have developed a simulator for a media server that supports various video delivery techniques and scheduling policies. We have validated the simulator by reproducing several graphs in previous studies. The simulation stops after a steady state analysis with 95% confidence interval is reached. Next, the workload characteristics and performance metrics are discussed.

4.1 Workload Characteristics

Table 1 summarizes the workload characteristics used. Like most prior studies, we assume that the arrival of the requests to the server follows a Poisson Process with an average arrival rate λ and that the access to videos is highly localized and follows a Zipf-like distribution with skewness parameter $\theta = 0.271$. We characterize the waiting tolerance of customers in terms of the number of ads by a Poisson distribution. To keep the analysis focused, we do not experiment with other models based on prediction results.

Table 1: Summary of Workload Characteristics

Parameter	**Model/Value(s)**
Request Arrival	Poisson Process
Request Arrival Rate	Variable, Default = 40 Requests/min
Server Capacity	400-900
Video Access	Zipf-Like, Skewness Parameter $\theta = 0.271$
Number of Movies	120
Movie Length	120 min
Waiting Tolerance Model	Poisson, min = 3 ads, mean= 5 ads, max = 8 ads
Ad Length	30 sec
Number of Different Ads	8
Number of Ads Channels	Variable, Default = 3

We consider here a commercial *Movie-on-Demand* system with 120 titles, each of which is 120-minute long. Stream merging is done using Patching. We analyze the impacts of both server capacity (i.e., number of server channels) and number of ads' channels. Without loss of generality, we assume here a *cost-plus* model for the price. The price covers the movie royalty fee, delivery fee, and operational cost minus subsidization credit. All revenues from the ads are distributed to the clients proportionally to their total ads' viewing times. In the discussed example, the revenue per ad per user is 10 cents, the movie royalty fee is 70 cents, and the delivery cost per GB is 50 cents. Based on service positioning analysis, the service provider seeks to get 70 cents per movie request to cover their operational cost and attain the sought profit. A fixed fraction of the 70 cents is used as a profit.

4.2 Performance Metrics

The analyzed performance metrics here are *customer defection probability*, *average number of ads viewed per client*, and *price*, *channel utilization*, *arrival rate*, *profit*, and *revenue*.

The overall revenue is challenging to estimate, although it can be given simply as the product of the volume sold and the price. The volume here is the total number of streams delivered to clients, which directly depends on the customer defection probability. The complication happens because the price influences the arrival rate and number of streams delivered. Thus, subsidizing the price can attract more clients and can eventually increase the overall revenue. By increasing the arrival rate, the delivery costs also decrease because of the higher degrees of request aggregation and stream merging. Finding the profit also exhibits similar complications as the revenue.

In this paper, we stress that the arrival rate will not only depend on the price but also the customer defection probability. We experiment with two main functions:

$$\lambda = \frac{c_1(1-d)(1-p/p_{max})}{c_2 + c_3 d^2 + c_4 (p/p_{max})^2} \quad and \quad \lambda = \frac{c_1(1-d)(1-p/p_{max})}{c_2 + c_3 d + c_4 p/p_{max}}, \tag{5}$$

where d is the defection probability, p is the price, p_{max} is the maximum price at which no customer will be interested in the service, and c_1, c_2, c_3, and c_4 are constants. The default values of these parameters in this paper are 60, 0.5, 1, and 1, respectively. Dividing p by p_{max} serves to normalize the price. Figures 5 and 6 depict these two functions, respectively.

5. RESULT PRESENTATION AND ANALYSIS

In this section, we generally assume the Only-Full ad delivery option and a total of three ads' channels, unless otherwise indicated.

5.1 Effectiveness of Existing Scheduling Policies

Figure 7 compares the performance of existing scheduling policies in the proposed delivery framework environment when Patching is used for delivering the primary media content. The relative performance is similar in the case of ERMT and thus not shown to limit the number of figures. The performance of FCFS, MQL, and MCF-P (RAP) are plotted in terms of the defection rate, average number of views ads, and unfairness. MFQL is not shown because it does not perform well

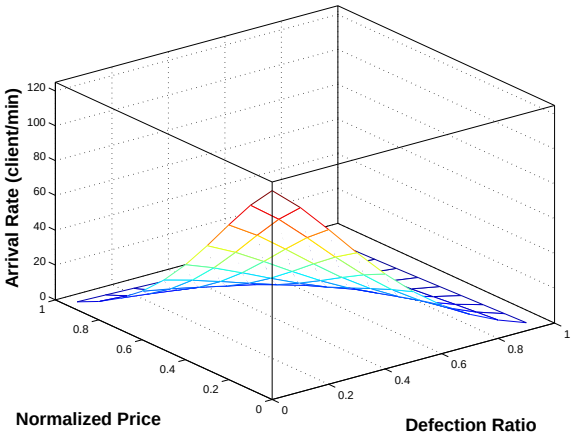


Figure 5: Arrival Rate Function 1 [$c_1 = 60$, $c_2 = 0.5$, $c3 = c_4 = 1$]

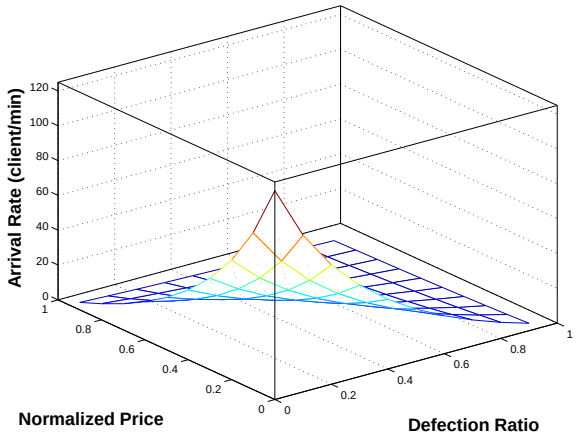
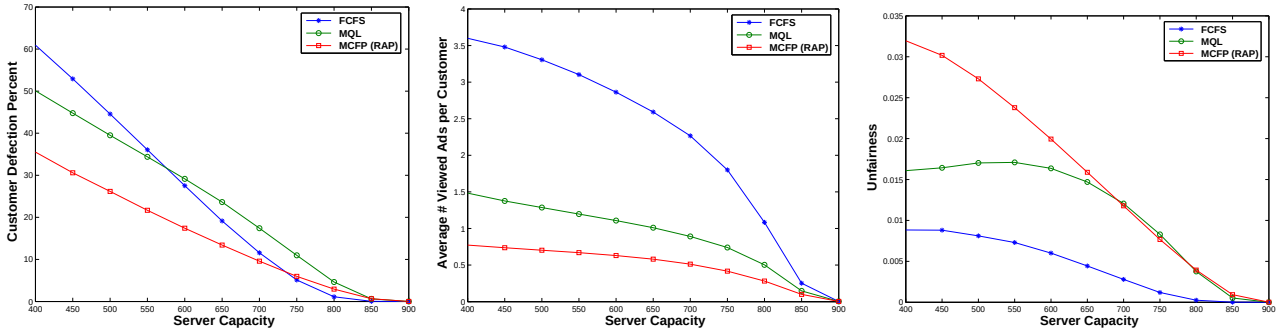


Figure 6: Arrival Rate Function 2 [$c_1 = 60$, $c_2 = 0.5$, $c3 = c_4 = 1$]

in the stream merging environment. The results here demonstrate that MCF-P achieves the best overall performance in terms of the two most important metrics. As expected, FCFS has the best fairness. It also reduces the defection probability better than MCF-P for very high server capacities. Unfortunately, the average viewing time with the overall better policies (MCF-P and MQL) is rather small. With MCF-P, a large number of customers did not view any ad at all, which reduces the price subsidization. These results motivate the new variants of MCF-P: Each N and Any N. These variants can also be used for MQL but we choose only MCF-P because of its higher performance.



(a) Customer Defection Rate (b) Number of Viewed Ads (c) Unfairness

Figure 7: Comparing Effectiveness of Existing Scheduling Policies [Patching]

5.2 Effectiveness of New Scheduling Polices

Let us now discuss the effectiveness of the proposed scheduling policies. Figure 8 illustrates the impact of N in MCF-P (Any N) on the defection probability. MCF-P (Any 3) causes the highest defections because it requires that at least one client views at least three ads. This increases the ads' viewing time and thus the probability of exceeding customer waiting tolerance for the desired content. Note that N has two conflicting impacts on defection rate. Increasing N increases the waiting time for primary content and also increases resource and data sharing. Thus, MCF-P (Any 2) has a more balanced impact and achieves the least defections.

Figure 9 and 10 plot the defection rate versus the average number of viewed ads with all new policies for Patching and ERMT, respectively. The curve for each policy is generated by varying the server capacity. Higher server capacities produce lower defection rates. Note that MCF-P (Each 3) occupies the least interesting area of operation, which has very high defection rates as a result of very high ads' viewing times. MCF-P (Any N) has nice regions of operations, which can be easily controlled by N (as well as server capacity). MAT has a wide range of operation and can be controlled only by changing the server capacity. Next, we primarily focus on MAT, MCF-P (Any 2), and MCF-P (Each 2). The same value of N is chosen for the two MCF-P variants to ensure a realistic comparison.

Figure 11 compares MAT, MCF-P (Each 2), and MCF-P (Any 2) in eight different performance metrics when assuming arrival rate function 1 in all except for Figure 11(i), which assumes function 2. MCF-P (Any 2) remains the best overall performer in terms of the defection rate, profit, and revenue. The relative profit remains essentially the same with the two arrival rate functions. The same is true for the revenue (but it is not shown for function 2 to reduce the number of figures). We have also experimented with different c_3 and c_4 values. MCF-P consistently gives the highest profit and revenue although the relative performance of MAT and MCF-P (Each 2) was not always the same. MCF-P (Any 2) has high unfairness, but this metrics carries less importance. Although MCF-P (Each 2) lowers the price the most (as shown in Figure 11(e)) due to the largest ads' viewing time, it yields lower profit and revenue because the increase in the defection rate is more significant. Interestingly, the average delivery cost with MCF-P (Any 2) and MCF-P (Each 2) remains nearly constant with the server capacity. One would expect the delivery cost to increase with server capacity because of lower resource and data sharing. The arrival rate here, however, also increases with server capacity when these two policies are used as shown in Figure 11(f). Increasing the arrival rates balances the impact of increasing the server capacity on delivery cost. As expected the server channel utilization is the highest with MAT because it does not impose any minimum ad's viewing time requirement. MCF-P (Each 2) and MCF-P (Any 2) have lower utilization because they force some requests to wait even when channels are available. MCF-P (Each 2) imposes more waiting and thus results in lower utilization than MCF-P (Any 2). The subsequent results are all with Patching but the relative behavior is very close to that with ERMT.

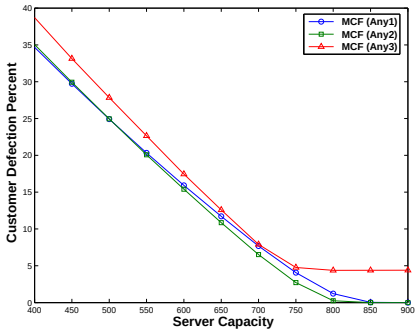


Figure 8: Impact of N on MCF (ANY) [Patching]

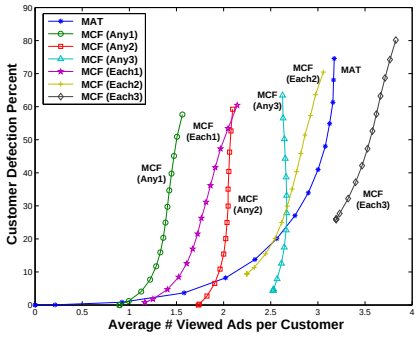


Figure 9: Comparing Effectiveness of New Scheduling Polices [Patching]

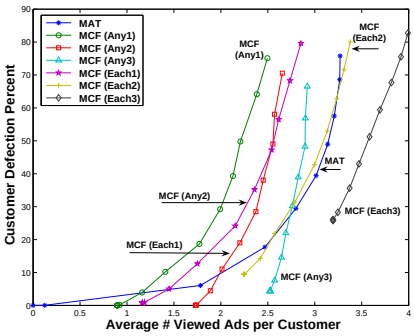


Figure 10: Comparing Effectiveness of New Scheduling Polices [ERMT]

5.3 Effectiveness of Partial-OK vs. Only-Full

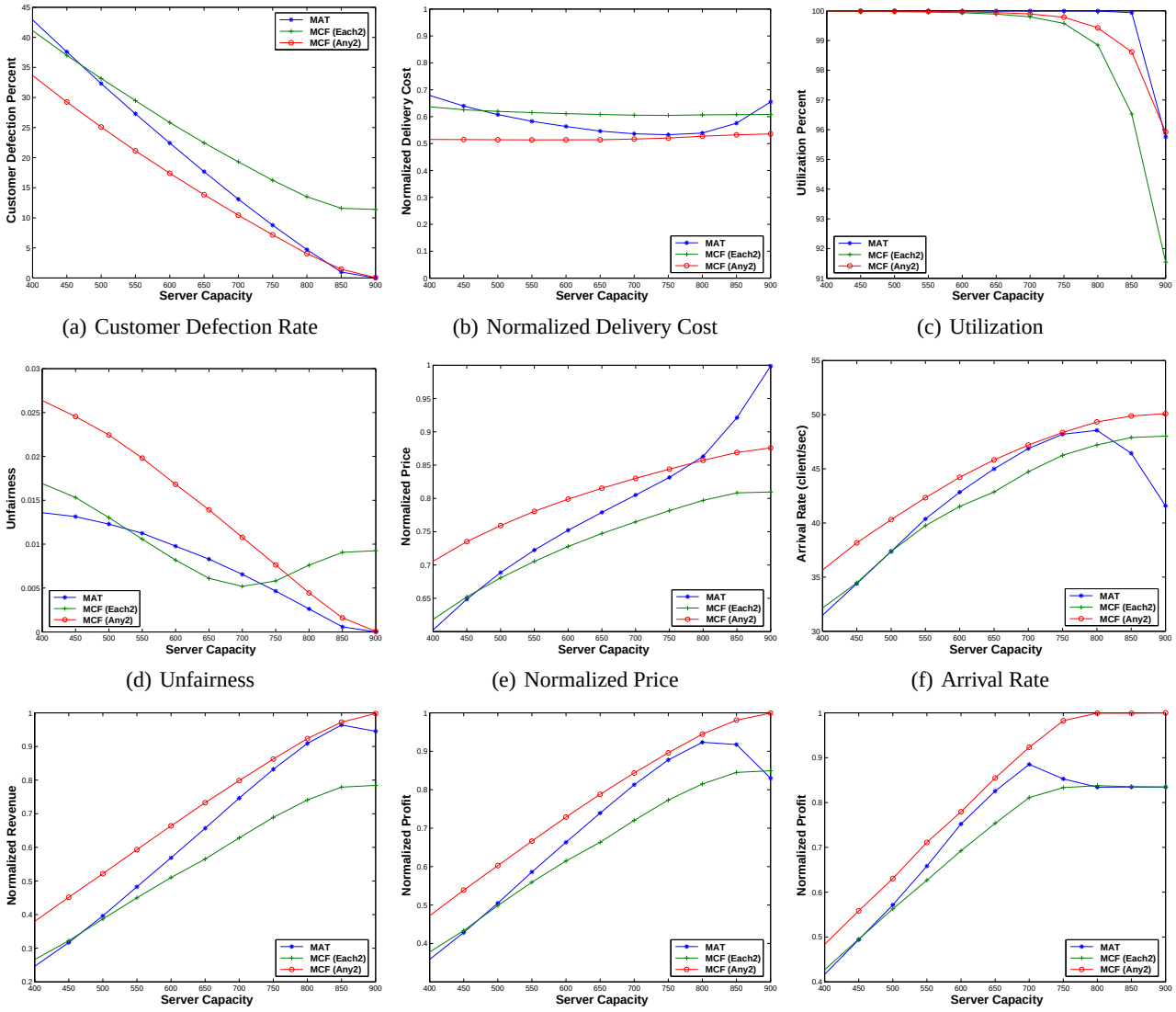
Let us now compare the effectiveness of the two ad delivery options: Partial-OK and Only-Full. Figure 12 confirms that the Partial-OK option leads to shorter ads' viewing times and thereby lower defection rates. It also achieves higher revenue and profit. The utilization is lower for higher server channels than Only-Full, but this is not necessarily a bad thing. The reduced utilization is because with Partial-OK, almost all requests are serviced at high server capacities while not all channels are used completely. Moreover, the Partial-OK option has better fairness for moderate and high server capacities. It is clear that Partial-OK is a better option in terms of performance, but some advertising companies may be reluctant to accept such an option. Moreover, ads' pricing may not be straightforward with this option.

5.4 Effectiveness of Patching vs. ERMT

Figure 13 compares the performance of Patching and ERMT in the proposed delivery framework when MCF-P (Any 2). These results demonstrate that ERMT achieves significantly better than Patching in all performance metrics (except utilization). This is due to the the hierarchical stream merging nature of ERMT. The utilization is lower with ERMT for high server capacities because it services (almost) all requests without using all the channels. The only disadvantage is that ERMT is much more complex to implement. The high performance benefits, however, could justify its application.

5.5 impact of Number of Ads Channels

Finally, let us discuss the impact of the number of ads' channels. Server capacity is fixed at 650. Figure 14 illustrates that MCF-P (Any 2) remains the best performer regardless of the number of ads' channels. As expected, the number of viewed ads and thus the defection rate increases with the number of ads' channel because of the fewer number of channels used



(a) Customer Defection Rate (b) Normalized Delivery Cost (c) Utilization

(d) Unfairness (e) Normalized Price (f) Arrival Rate

(g) Normalized Revenue (h) Normalized Profit (1) (i) Normalized Profit (2)

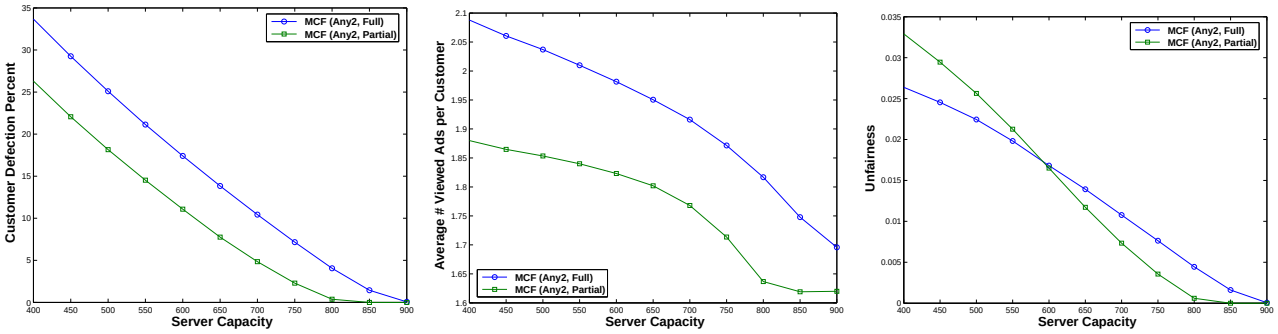
Figure 11: Comparing Effectiveness of MAT, MCF-P (Any 2) and MCF-P (Each 2) [Patching, Arrival Rate Function 1 except for (i) which is for Function 2]

for delivering the desired (primary) media content. The profit also decreases with the number of ads' channels. Increasing the number of ads' channels reduces the initial waiting time for receiving the ads, but a value larger than four in the studied workload is not beneficial because the waiting time decreases only slightly.

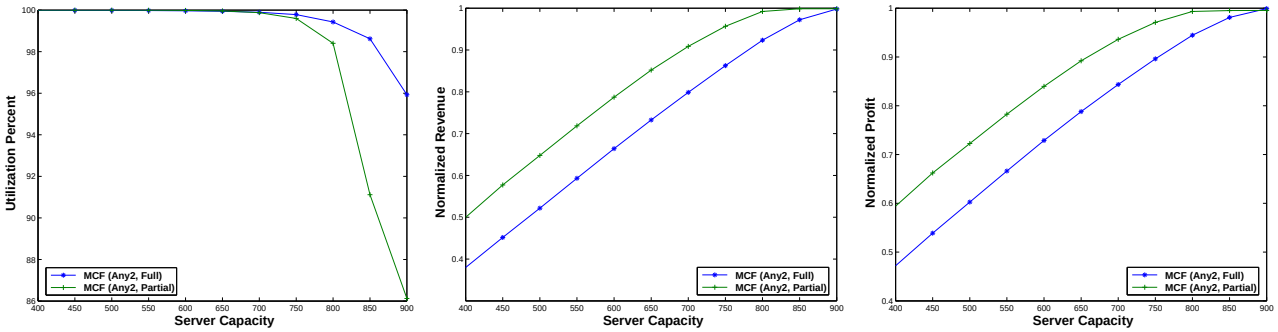
6. CONCLUSIONS

We have presented a delivery framework for streaming media with advertisements and an associated pricing model. The delivery model combines the benefits of periodic broadcasting and stream merging. The advertisements' revenues are used to subsidize the price of the media streaming, and the pricing is determined based on the total ads' viewing time. We have discussed two ad delivery options: *Only-Full* and *Partial-OK*. In addition, we have presented an efficient heuristic ad allocation scheme and three modified scheduling policies for the proposed framework. These policies are *Maximum Ads Time* and two variants of *Minimum Cost First* (MCF): *Any N* and *Each N*.

We have studied the effectiveness of various scheduling policies and the two ad delivery options with both Patching

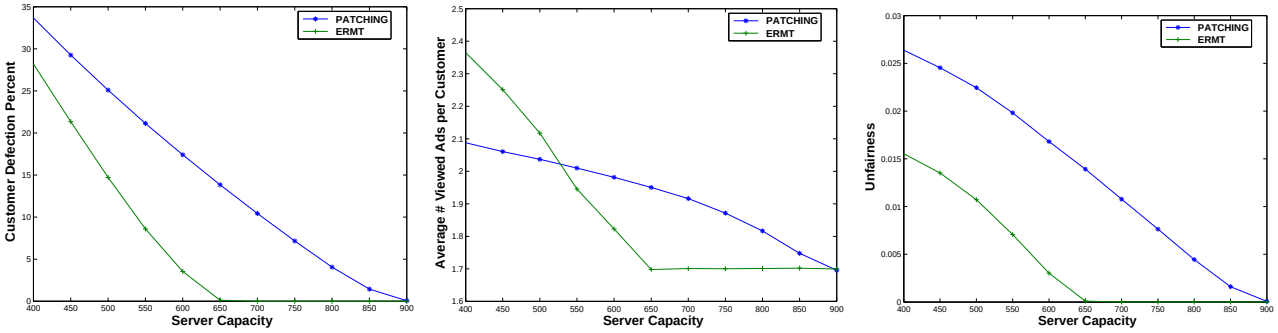


(a) Customer Defection Rate (b) Average Number of Viewed Ads (c) Unfairness

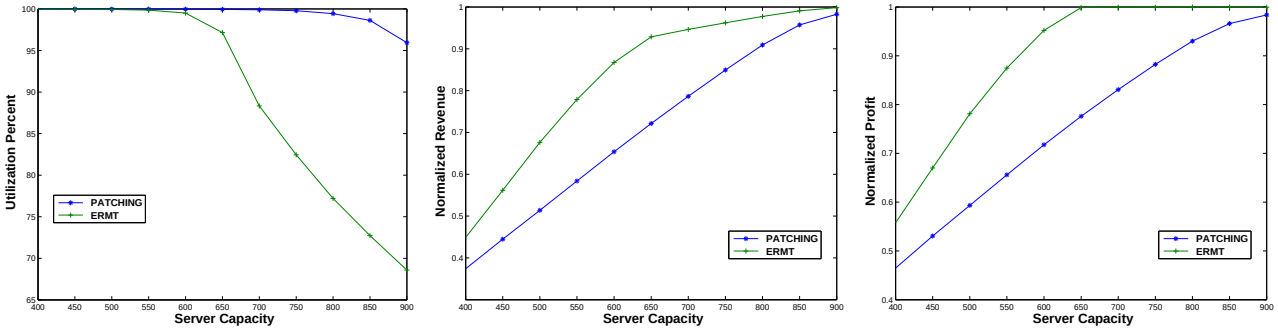


(d) Utilization (e) Normalized Revenue (f) Normalized Profit

Figure 12: Comparing Effectiveness of Only-Full and Partial-OK [Patching, Arrival Rate Function 1]

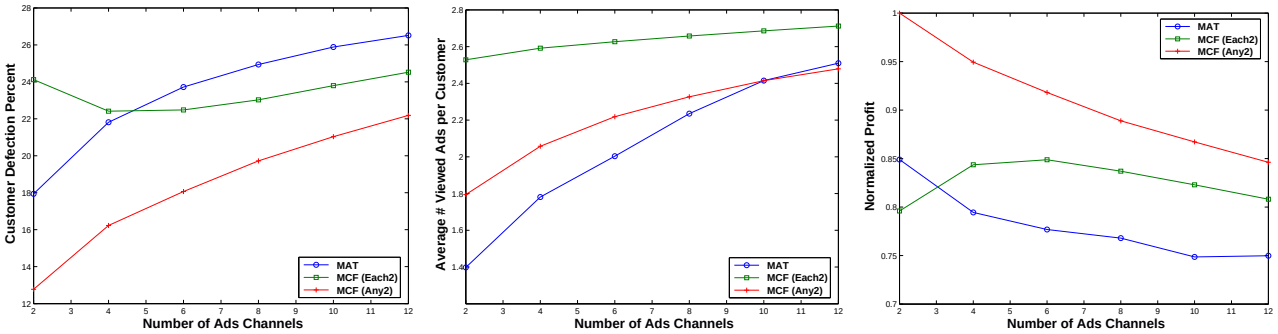


(a) Customer Defection Rate (b) Average Number of Viewed Ads (c) Unfairness



(d) Utilization (e) Normalized Revenue (f) Normalized Profit

Figure 13: Comparing Effectiveness of Patching and ERMT [MCF-P (Any 2)]



(a) Customer Defection Rate (b) Average Number of Viewed Ads (c) Normalized Profit

Figure 14: Impact of Number of Ads' Channels [Patching, Server Capacity = 650]

and ERMT through extensive simulation. The analyzed metrics include *customer defection probability*, *average number of ads viewed per client*, *price*, *channel utilization*, *arrival rate*, *profit*, and *revenue*. We have considered the impacts of price and defection rate on the request arrival rate. The main results can be summarized as follows.

- MCF-P (Any N) achieves the best overall performance. Moreover, it can control the average ads' viewing time by adjusting N. The best value of N in the studied workload is 2.
- The Partial-OK ad delivery options significantly performs better than Only-Full but may not be applicable for all types of ads because of the partial viewing.
- The number of ads' channels should be as small as possible so as not to hurt performance. Moreover, increasing their number reduces the waiting time for receiving the ads only slightly after a certain point.

REFERENCES

1. C. C. Aggarwal, J. L. Wolf, and P. S. Yu. The maximum factor queue length batching scheme for Video-on-Demand systems. *IEEE Trans. on Computers*, 50(2):97–110, Feb. 2001.
2. M. Alsmirat, M. Al-Hadrusi, and N. J. Sarhan. Analysis of waiting-time predictability in scalable media streaming. In *Proc. of ACM Multimedia (To Appear)*, Sept. 2007.
3. P. Basu and T. D. C. Little. Pricing considerations in video-on-demand systems. In *Proc. of ACM Multimedia*, pages 359–361, Nov. 2000.
4. P. Basu, A. Narayanan, W. Ke, T. D. C. Little, and A. Bestavros. Optimal scheduling of secondary content for aggregation in video-on-demand systems. In *Proc. of International Conference on Computer Communications and Networks*, pages 104–109, Oct. 1999.
5. Y. Cai and K. A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proc. of ACM Multimedia*, pages 211–214, Oct. 1999.
6. S. W. Carter and D. D. E. Long. Improving Video-on-Demand server efficiency through stream tapping. In *the International Conference on Computer Communication and Networks (ICCCN)*, pages 200–207, Sept. 1997.
7. A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM Multimedia*, pages 391–398, Oct. 1994.
8. D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for Video-on-Demand servers. In *Proc. of ACM Multimedia*, pages 199–202, Oct. 1999.
9. D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, Sept. 2001.
10. L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, July 1998.
11. K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true Video-on-Demand services. In *Proc. of ACM Multimedia*, pages 191–200, 1998.

12. K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan Video-on-Demand system. In *Proc. of ACM SIGCOMM*, pages 89–100, Sept. 1997.
13. C. Huang, R. Janakiraman, and L. Xu. Loss-resilient on-demand media streaming using priority encoding. In *Proc. of ACM Multimedia*, pages 152–159, Oct. 2004.
14. L. Juhn and L. Tseng. Harmonic broadcasting for Video-on-Demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.
15. H. Ma, G. K. Shin, and W. Wu. Best-effort patching for multicast true VoD service. *Multimedia Tools Appl.*, 26(1):101–122, 2005.
16. J.-F. Pâris, S. W. Carter, and D. D. E. Long. Efficient broadcasting protocols for video on demand. In *Proc. of the Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 127–132, July 1998.
17. B. Qudah and N. J. Sarhan. Towards scalable delivery of video streams to heterogeneous receivers. In *Proc. of ACM Multimedia*, pages 347–356, Oct. 2006.
18. M. Rocha, M. Maia, I. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proc. of ACM Multimedia*, pages 966–975, Nov. 2005.
19. N. J. Sarhan and C. R. Das. Caching and scheduling in NAD-based multimedia servers. *IEEE Trans. on Parallel and Distributed Systems*, 15(10):921–933, Oct. 2004.
20. N. J. Sarhan and B. Qudah. Efficient cost-based scheduling for scalable media streaming. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, January 2007.
21. L. Shi, P. Sessini, A. Mahanti, Z. Li, and D. L. Eager. Scalable streaming for heterogeneous clients. In *Proc. of ACM Multimedia*, pages 337–346, Oct. 2006.