Waiting-Time Prediction and QoS-Based Pricing for Video Streaming with Advertisements

Nabil J. Sarhan Musab S. Al-Hadrusi
Media Research Lab, Dept. of Electrical and Computer Engineering
Wayne State University Detroit, Michigan 48202
Email: {nabil,hadrusi}@wayne.edu

***Abstract*—This paper considers scalable delivery of streaming video content with advertisements. It proposes a highly accurate waiting-time prediction algorithm that estimates the expected number of viewed ads by utilizing detailed information about the system state and the applied scheduling policy. It also proposes a pricing scheme based on the expected waiting times, which include the ads' viewing times. The revenues generated by the ads are used to subsidize the price and are allocated to clients proportionally to their expected waiting times. The envisioned system presents the client with an updated menu of the supported videos (such as full-length featured movies) along with the corresponding ads' viewing times and prices. We analyze the effectiveness of the proposed prediction algorithm and pricing scheme in terms of many metrics, considering numerous design and workload parameters, and capturing purchasing capacity and willingness models.**

I. Introduction

The distribution of streaming media faces a significant scalability challenge because of the high server and network requirements. Service providers have attempted to address this challenge by using *Content Distribution Networks* (CDNs) and *Peer-to-Peer* (P2P) approaches. While the first approach requires maintaining a huge number of geographically distributed servers, the second still relies heavily on central servers [1], [2]. Both these approaches mitigate the scalability problem but do not eliminate it because the fundamental problem is due to unicast delivery [2]. Multicast is highly effective in delivering high-interest and high-usage content and in handling flash crowds. Recently, there has been a growing interest in enabling native multicast to support high-quality on-demand video distribution, IPTV, and other major applications [3], [4], [2]. This paper considers the multicast approach.

Multicast-based 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], [6], [2], which reduce the delivery cost by aggregating clients into larger groups that share the same multicast streams. The second category consists of *Periodic Broadcasting* techniques [5] (and references within), which 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.

Supported in part by U.S. NSF grants CNS-0626861 and CNS-0834537.

The overwhelming majority of prior studies focused on regular content without any media advertisements. The interest in using ads has grown dramatically in the media industry. Using ads generates revenues, converts passive startup waiting times for service to active waiting times (i.e., watching ads while waiting), and results in better aggregation of the requests, thereby reducing the delivery costs. Many clients also like to watch some types of ads, such as movie trailers.

This paper utilizes the recently proposed framework for scalable delivery of media content with advertisements [7]. This 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. The system is mainly targeted for supporting premium content with royalty fees, such as featured movies. The ads' revenues are used to subsidize the price and thus attract more clients. Pricing depends on the total experienced waiting times, which include the ads' viewing times. Clients with longer ads' viewing times get lower prices.

The main challenge with the scalable delivery framework is that the price has to be set before the client makes a selection of the media content to playback. The main contribution of this paper is to address this challenge by proposing a highly accurate *waiting-time prediction* algorithm. Waiting-time prediction in multicast-based delivery is complicated because of the uncertainty of waiting-times due to request aggregation (by stream merging). This algorithm estimates the ads' viewing period based on the requested media and the system's current state and involves three types of predictions: *video selection prediction*, *channel availability prediction*, and *scheduling qualification prediction*. It considers the dynamic and complex natures of stream merging and request scheduling and works with any stream merging technique and scheduling policy. This paper presents two alternative implementations of the prediction algorithm.

Additionally, this paper proposes a pricing scheme based on the expected ads' viewing time rather than the actual ads' viewing time. By using the proposed waiting-time prediction algorithm, the price can be determined based on the expected ads viewing time, and this price is presented to the client. Although this pricing scheme involves some sort of price prediction, it is not risky to do so because of the achieved high accuracy in waiting-time prediction, and more importantly because only the way the ads' revenues are allocated to clients is affected and not the actual cost. The waiting-time prediction

978-0-7695-4217-1 2010
DOI 10.1109/ISM.2010.13

is used primarily to determine the price. The estimated ads' viewing time may or may not be presented to the client along with the price. The clients would appreciate this feedback information, which serves as expected times of service and thus an important *human-centered* metric [6], but in that case, the system should use a mechanism to ensure that the clients do actually watch the ads (such as requiring their input to advance to the next ad, using interactive ads, etc.). In this paper, we assume that such a mechanism is employed and thus the expected waiting times (or ads' viewing times) are presented to the clients along with the associated prices. The pricing scheme also provides incentives to clients who view more ads than expected.

Another contribution of this paper is incorporating client *purchasing capacity* and *willingness* models based on economy theory [8] to study the effectiveness of the proposed waiting-time prediction and pricing schemes. A further contribution of the paper is the extensive analysis of waiting-time predictability, pricing, and effectiveness of various scheduling and stream merging strategies in the scalable delivery framework, considering numerous metrics. These metrics include *waiting-time prediction accuracy*, *percentage of clients receiving waiting times*, *client defection* (i.e., turn-away) probability, *average waiting time*, *price*, *arrival rate*, *profit*, and *revenue*. The paper also analyzes the impacts of various design and workload parameters. Because of the large number of parameters, we conducted extensive investigation to ensure representative results.

The **unique features and contributions** of this paper can be summarized as follows: (1) analyzing waiting-time predictability in multicast-based delivery of media content with advertisements, (2) proposing a pricing scheme based on waiting-time prediction, (3) utilizing purchasing capacity and willingness models in the evaluation, and (4) providing extensive analysis, considering a significantly large number of influences on scheduling decisions. A recent study [6] discussed waiting-time prediction in a much simpler setting, in which only stream merging but no periodic broadcasting is applied. That paper did not consider advertisements or pricing.

The rest of the paper is organized as follows. Section II discusses background information and related work. Section III presents the proposed pricing schemes and waiting-time prediction algorithms. Subsequently, Section IV discusses the performance evaluation methodology. The main results are discussed in Section V.

II. Related Work

Stream Merging and Request Scheduling

Stream merging techniques include *Patching* [9], *Transition Patching* [10], and *Earliest Reachable Merge Target* (ERMT) [11]. Patching expands the multicast tree dynamically to include new requests. A new request joins the latest regular (i.e., full-length) stream for the video and receives the missing portion as a patch. 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. Transition Patching, however, allows streams to merge up to two times. By contrast, ERMT is a near optimal hierarchical stream merging technique and allows streams to merge multiple times. A new client or a newly merged group of clients snoops on the closest stream that it can merge with if no later arrivals preemptively catch them [11].

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 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*. Among the many available scheduling policies, *Minimum Cost First* (MCF) [12], 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 delivery cost, measured as the number of seconds of the video data to be delivered. Due to stream merging, this cost is usually much smaller than the video length. *MCF-P*, the preferred implementation of MCF, selects the queue with the least cost per request.

Using Advertisements

Supporting ads has been discussed in only few studies. In [13], ads are inserted randomly multiple times during the playback of the primary media, and the employed resource sharing technique adjusts the video playback rates. A general discussion of pricing was provided in [14]. In [7], a scalable delivery framework was proposed. Its main characteristics can be summarized as follows. (1) Clients start by joining an ads' broadcast channel for some time and then receive the requested video by stream merging. ("Video" or "requested video", unless otherwise indicated, refers to one of the primary videos and not an ad.) (2) Ads are combined and broadcast on dedicated server channels. Hence, when beginning to listen to an ads' channel, the client views different ads until streaming of the desired media commences. Multiple channels can be used with time-shifted versions of the combined ads, as shown in Figure 1, to reduce the waiting time for reaching the beginning of an ad. With only one ads' channel, the maximum value of this time is equal to the ad length (ad_len). With N_{adCh} number of channels, the value is reduced to ad_len/N_{adCh}. (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) Receiving portions of the requested media starts as soon as the client joins an ads' channel.

To ensure that ads are viewed by a large number of clients, three constraint-based scheduling policies were proposed in [7]: *Any N*, *Each N*, and *Maximum Ads Time* (MAT). Other policies, like MCF, attempt to serve requests as soon as possible, thereby reducing significantly the ads' viewing time. Any N and Each N are modified versions of MCF-P. *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. In contrast, MAT selects for service the video whose waiting requests have the longest total ads' viewing time.

Figure 1 shows an example illustrating the operation of the scalable delivery framework with constraint-based scheduling (Any 2 in particular). The server has three ads' channels on which three specific ads ($Ad1$, $Ad2$, and $Ad3$) are broadcast periodically. It also has four channels, all initially unused, for servicing the requested videos using stream merging. The figure shows the clients' requests made at any point of time and the specific ads that the clients might be viewing. Based on their arrival times, requests a and b are admitted to $AdCh1$ at time $T1$, request c is admitted to $AdCh2$ at time $T2$, and so on. At time $T7$, the clients for requests a and b have already watched two ads, thereby meeting the Any 2 qualification criterion for videos $V1$ and $V2$, respectively. The server has four channels available at that time and thus both videos $V1$ and $V2$ are serviced using two separate channels, leaving two available channels. Request e for $V1$, which was admitted to $AdCh1$ at time $T4$, will be serviced with request a using the same channel.

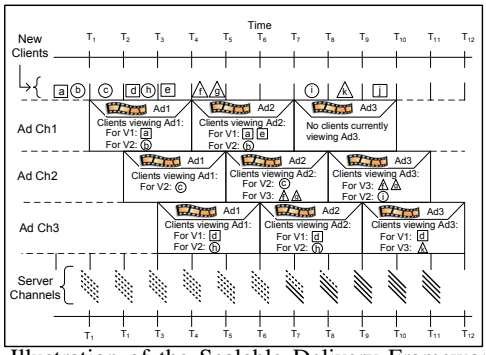


Fig. 1: Illustration of the Scalable Delivery Framework with Constraint-Based Scheduling [Any 2]

III. Proposed Pricing and Waiting-Time Prediction Schemes

This paper proposes a waiting-time prediction algorithm to estimate the expected ads' viewing times in the scalable delivery framework. It also proposes a pricing scheme based on the expected ads' viewing time, royalty fee, and the current delivery cost of the requested video. (When desirable, it is also possible to remove the last from the price function. Note that the delivery cost varies with the current access rate. Videos with larger concurrent requests have lower delivery costs because of the better resource and data sharing achieved by stream merging.) In the pricing model, all ads' revenues are used to subsidize the price, thereby attracting more clients. Additionally, the amount of subsidization the client receives is determined based on the predicted ads' viewing time. The subsidization increases with the expected ads' viewing time (or expected waiting time) because of the lower expected QoS level and the larger contribution to the generated ads' revenues. Although this pricing scheme involves some sort of price prediction, this is justified by the following two reasons. (1) As will be shown later, the waiting-time prediction algorithm is highly accurate. (2) Inaccurate prediction involves no risk because only how the ads' revenue is allocated to clients is affected and not the actual service cost. The total service cost (which includes the delivery cost) is determined dynamically based on the requested video.

In the envisioned system, the user is first presented with a menu showing the list of supported videos with their corresponding expected ads' viewing times and prices. It may be desirable that the system allows the clients to stop the service without charge while receiving ads but before the streaming of the requested media starts. This can help in the unlikely situation when the client views much more ads than expected. To encourage clients who wait longer than expected to use the system again in the future, the system offers them *bonus points*, which can be applied later to increase their subsidization allocation.

The idea of waiting-time prediction for media streaming has recently been proposed in [6]. That study has presented a prediction algorithm that considers the applied scheduling policy and utilizes information about the current server state, including the current waiting queue lengths, the completion times of running streams, and statistics such as the average request arrival rate for each video. It uses the completion times of running streams to know when channels will become available, and thus when waiting requests can be serviced. The algorithm, however, works only for stream merging techniques and cannot be used in the scalable delivery framework. Moreover, [6] did not consider advertisements or pricing.

Waiting-time prediction in the scalable delivery framework is complicated for the following reasons. (1) Both periodic broadcasting and stream merging techniques are used concurrently. (2) Different clients may join different ads' channels and of course have to be dealt with in a differentiated manner. (3) Clients should receive multiple numbers of ads. Partial viewing of ads complicates pricing and may not be an attractive choice to advertisement companies. (4) Constraints on minimum ads' viewing times must be met, such as those of Any N and Each N scheduling. (5) The number of available channels at a future time is somewhat hard to predict because of the interleaving of ads' channels. (6) Multiple ads' channels may become available simultaneously.

This paper presents a highly accurate waiting-time prediction algorithm, called *Assign Closest Ad Completion Time* (ACA), for the scalable delivery framework. The main idea can be summarized as follows. As illustrated in Figure 2, when a new request is made at time T_{Now}, it is mapped to the ads' channel with the closest ad start (or end) time (Ad Ch 2 in this example). The algorithm then examines the future ad start times on that channel in order (T_0, T_1, ...) for possible assignment as the expected time for that request. These ad start times represent possible expected times for serving the requested video. The request is tied to a specific ads' channel because we assume here that partial-ad viewing is not permitted and thus requests can be served only at ad boundaries. At each ad start time, the algorithm estimates the number of available channels and predicts the videos that can be served at that time. The process continues until the expected video to be served is the same as the currently requested video or the *prediction window* (Wp) is exceeded. Wp controls the implementation complexity and introduces a tradeoff between prediction accuracy and the percentage of clients receiving expected times. As Wp increases, the algorithm gives expected times to more incoming requests but

at the expense of increasing the time complexity (a function of W_p × number of videos) and more importantly reducing the the accuracy. In the considered example, $W_p = 3$ ad lengths and thus the algorithm examines the ad start times in the order T_0, T_1, T_2, and T_3, until an expected time is determined for the requested video.

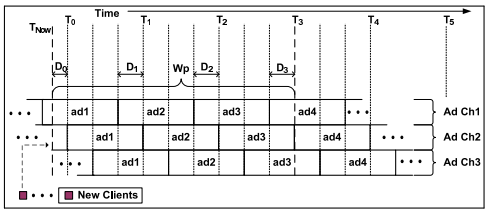


Fig. 2: Illustration of the ACA Algorithm

Figure 3 shows a simplified version of the algorithm, which is performed upon the arrival of request R_i for video v_j if the server is fully loaded. Otherwise, the request can be serviced immediately. The algorithm has two major loops: the outer one (Lines 7 − 38) goes over successive ad start times on ad channel $adChNo$ (corresponding to the new client), and the inner (Lines 21 − 33) goes over all available channels at that time. Note that at any ad start time T, a video may be assigned only once. Thus, when a video is selected, its objective function is reset to -1, specifying that it should not be selected again.

```
1 for (v = 0; v < N_v; v++) // Initialize
2    assigned_time[v] = −1; // Not assigned expected time
3 adChNo = Get # of the ads' channel with the closest start time;
4 T = Get next ad's start time; 4 T_0 = T; examined_times = 0;
5 // Find number of available channels at time T
6 N_c = available_channels + will_be_available(T_Now, T);
7 while (T < T_Now + W_p) { // Loop till prediction window is exceeded
8    for (v = 0; v < N_v; v++){
9       if (isQualified(v, T, adChNo)) {
10         if (assigned_time[v] == −1)
11            expected_qlen = qlen(v, adChNo) + λ[v]×
12            ((T_0 − T_Now) + examined_times × ad_len/N_adCh);
13         else // Video v has been assigned an expected time
14            expected_qlen = λ[v]×
15            (T − assigned_time[v])/N_adCh;
16         objective[v] = find scheduling objective for video v;
17      } // end if (isQualified(v, T, adChNo))
18      else
19         objective[v] = −1; // v is not qualified
20   } // end for (v = 0; v < N_v; v++)
21   while (c = 0; c ≤ N_c; c++){ //for every available channel
22      // Find the expected video to serve at time T
23      expected_video = find video with maximum objective;
24      // -1 objectives are discarded
25      if(expected_video == v_j) {
26         Assign T as the expected time to request R_i;
27         return;
28      }
29      else {
30         assigned_time[expected_video] = T;
31         objective[v] = −1; // Can't be selected again
32      }
33   } // end while (c = 0; c ≤ N_c; c++)
34    T = T + ad_len; //Proceed to the next edge
35    // Find number of available channels at time T
36    N_c = left_over + will_be_available(T − ad_len/N_adCh, T);
37    examined_times++;
38} // end while (T < T_Now + W_p)
```

Fig. 3: Simplified Algorithm for ACA [performed upon arrival of request R_i for Video v_j]

Figure 4 illustrates the operation with an example. Each ads' channel is split here into three logical channels, one for each video.

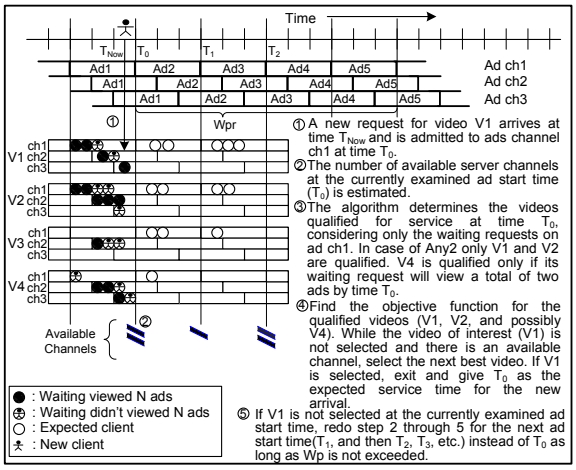


Fig. 4: An Example Illustrating the Operation of ACA

Video Selection Prediction

Let us now discuss how to predict the videos to be served at any particular ad start time T. First, the algorithm (Line 9) determines the videos that will be qualified at that time based on any minimum ads' viewing constraints, such as those imposed by Any N and Each N. The qualification algorithm will be discussed later on. For each video that qualifies, the algorithm (Lines 10 − 15) estimates its waiting queue length at time T, if the scheduling policy requires so (such as MCF-P, Any N, and Each N), based on the video arrival rates, which are to be computed periodically but not frequently. The expected queue length for video v at time T can be found as follows:

$$expected\_qlen[v] = qlen(v, adChNo) + \lambda[v]\times \\ ((T_0 - T_{Now}) + examined\_starts \times ad\_len/N_{adCh}), \quad (1)$$

where $qlen(v, adChNo)$ is the queue length of video v on channel $adChNo$ at the current time (T_{Now}), $\lambda[v]$ is the arrival rate for video v, ad_len is the ad length, N_{adCh} is the number of ads' channels, T_0 is the nearest ad start time, and $examined\_starts$ is the number of examined ad start times so far during the current run of the prediction algorithm. Dividing by N_{adCh} in Equation (1) is because not all arrivals belong to the considered ads' channel. Note that the same video may be serviced again at a later ad start time. Equation (1) assumes that video v has not been identified before (while running the ACA algorithm for the new request) as the expected video to be serviced at an earlier ad start time. Otherwise, the expected arrivals will have to be found during the time interval between the latest assigned time ($assigned\_time[v]$) at which video v is expected to be served and T. In that case, the expected queue length for video v at time T is given by

$$expected\_qlen[v] = \lambda[v] \times (T - assigned\_time[v])/N_{adCh}. \quad (2)$$

Finally, the algorithm (Line 16) finds the objective function for each qualified video based on the scheduling policy and selects (Line 23) the video with the maximum objective function.

Channel Availability Prediction

The number of available server channels (N_c) is computed

as follows. For the closest ad start time (T_0 in Figure 2), the number of available channels is equal to the number of currently available channels plus the number of channels that will be available in the interval (D_0) between T_{Now} and T_0, inclusive. The currently available channels at time T_{Now} may not be used for later ad start times (T_1, T_2, etc.) because they may be consumed by requests on other ads' channels ($AdCh1$ and $AdCh3$). Instead, an average leftover value can be computed dynamically to estimate the number of available server channels that are carried over from one ad start time to the immediately following ad start time on the next ads' channel. Thus, the number of available channels at a later ad start time T is computed (Line 36) as the number of channels that will be available in the immediately preceding ad_len/N_{adCh} duration (D_1, D_2, or D_3) plus the average leftover value.

Scheduling Qualification Prediction

Scheduling policies imposing minimum ads' viewing times (such as Any N and Each N) require the determination whether a video will be qualified for service at a considered ad start time. This is done by the proposed *qualification algorithm*. It determines whether video v on ads' channel $adChNo$ will be qualified for service at time T. For Any N, this algorithm can be explained as follows. If the video has waiting clients on ad channel $adChNo$ (the ads' channel to which the new request Ri is mapped to), then qualify it only if at least one of these clients has viewed at least N ads. If the video, however, does not have any waiting clients, or it has already an expected time during the current run of the prediction algorithm (for request Ri), then the algorithm determines the probability ($QProb$) that it will have future clients and that at least one of them will have viewed at least N ads by the time T. The function $Prob(1, v, t_d)$ returns the probability of at least one arrival for video v during duration t_d. The video will only be qualified if $QProb$ is larger than a certain threshold, called the *qualification threshold* (Q_{Th}). To clarify the concept, let us discuss how $QProb$ for video v at time T can be computed upon the arrival of a new request (for another video) at time T_{Now} for the case when video v has no waiting requests and has not been assigned an expected time during the current run of the ACA algorithm. Referring to Figure 2, $QProb$ in this case is the probability that new expected requests for video v will view at least N ads and can be given by $QProb = P_1 + (maxViewedAds - N) \times P_2$, where P_1 is the probability of at least one arrival for video v during duration D_0 (whose length is $T_0 - T_{Now}$), P_2 is probability of at least one arrival during a duration of length ad_len/N_{adCh} (corresponding to D_1, D_2, etc.), and $maxViewedAds$ is the number of ad intervals between T_{Now} and T.

The main difference in the case of Each N is that the video is qualified only if each waiting client (real or expected) has viewed at least N ads. Ensuring the satisfaction of the Each N constraint for expected clients introduces some complications, which have been addressed in the algorithm. MAT, MCF-P, MQL, and MFQL do not require qualification.

Alternative Prediction Scheme

As discussed earlier, the ACA algorithm predicts the scheduling outcomes during a prediction window (Wp), which is constrained to reduce the implementation complexity and ensure satisfactory prediction accuracy. Hence, in practical situations, ACA may not be able to find an expected time for every incoming client. This issue can be addressed by using a hybrid scheme: if ACA does not return an expected time, the average waiting time for the video is used. This hybrid scheme is called *ACA+APW*. APW stands for *Assign Per-Video Average Waiting Time* (APW).

IV. Evaluation Methodology

The analysis is performed through extensive simulation, considering the following performance metrics: *prediction accuracy*, *percentage of clients receiving expected times of service* (PCRE), *client defection probability*, *average waiting time*, *price*, *arrival rate*, *profit*, and *revenue*. The defection probability is the probability that clients leave without being serviced because of waiting times exceeding their tolerance. The average deviation between the expected and actual times of service is used as a measure of accuracy. The accuracy decreases with the deviation. The reported average waiting time includes the time spent viewing ads and the initial waiting time to receive the ads.

Table I summarizes the workload characteristics used. We assume that the request arrival follows a Poisson Process and that the access to videos follows a Zipf-like distribution with skew parameter $\theta = 0.271$ [6]. (The skew reaches its peak when $\theta = 0$.) The Zipf-like distribution captures varying *video attractiveness*. The Stretched Exponential [15] resulted in almost the same relative behavior.

We characterize the waiting tolerance of clients by two models: *Model A* and *Model B* [6]. In Model A, the waiting tolerance of clients follows an exponential distribution with mean μ_{tol}. In Model B, clients receiving expected times less than μ_{tol} wait for service, and the waiting tolerance of all other clients follows an exponential distribution with mean μ_{tol}.

To characterize the arrival rate, we use the *willingness-based* model [8], which captures client purchasing capacity and willingness behavior. Economic theory suggests that the allocation of wealth is highly skewed and follows Pareto distribution. Similarly, the capacity of a client to spend for a particular service or product can be modeled using that distribution [8]. The Pareto probability density function can be given as follows: $f_p(x) = \alpha \times b \times x^{-(\alpha+1)}$ for $x \geq b$, where b (also called *scale*) represents the minimum value of x, and α represents the shape of the distribution. Most clients have capacities close to b. The distribution is more skewed for larger values of α. Hence, as α increases, fewer clients can pay much more than b. Clients with larger capacities are more likely to spend more. The willingness probability of a client with capacity y to pay for a product or service with price p can be given by

$$Prob(willingness) = \begin{cases} 1 - (\frac{p}{y})^{\delta} & 0 \leq p \leq y, \\ 0 & p > y. \end{cases} \quad (3)$$

where δ is the *elasticity* [8]. As δ increases, more clients are willing to spend. The arrival rate is initially an input and represents the maximum rate the server can attain and properly

handle. Based on our extensive analysis, the relative behavior of various considered schemes is not very sensitive to the values of b, α, and δ. We set their defaults values based on the considered system discussed next and to ensure realistic willingness behavior.

We consider here a commercial *Movie-on-Demand* system with 80 titles, each of which is 120 minutes long. Without loss of generality, we assume here *cost-plus* pricing. The price in the considered system 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 client is 10 cents, the movie royalty fee is 70 cents, and the delivery cost per GB is 50 cents [7]. 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, and a fixed fraction of the 70 cents is used as a profit [7].

TABLE I: Summary of Workload Characteristics

Parameter	Model/Value(s)
Request Arrival	Poisson, Default Rate: 40 Req/min
Server Capacity	200-550, Default: 350
Video Access	Zipf-Like, Skew Parm. $\theta = 0.271$
Movie Related	80 120-min movies
Waiting Tolerance Model	Model A: Poisson, min: 3 ads, mean: 5 ads, max: 8 ads. Model B
Ad-Related Characteristics	Ad Len: 30 sec, # different ads: 8, # ads channels: 3 (based on [7])
Min. Ads Constraint (N)	Default: 2
Prediction Window (W_p)	Default: 4
Qualification Thresh. (Q_{Th})	Default: 0.25
Scale, Shape, Elasticity	Defaults: b : 1.5, α : 1, δ : 7

V. Main Results

We include only the results of the parameters that have significant effects on the relative behavior.

Effectiveness of Waiting-Time Prediction

Let us start by analyzing the performance of the proposed ACA prediction algorithm in terms of average deviation and PCRE. Two straightforward approaches are used to evaluate its effectiveness: *Assign Overall Average Waiting Time* (AOW) and *Assign Per-Video Average Waiting Time* (APW), which work exactly as named. Figure 5 compares the average deviation results under Any 2, Each 2, and MAT, respectively, when stream merging is done using ERMT. Patching and Transition Patching exhibit a similar behavior but achieve slightly lower deviations (as shown in Figures 6 and 7). These results demonstrate that ACA is highly accurate, especially when combined with Any 2. The deviation in that case is within 7 seconds, which is less than 25% of the ad length.

Figures 6 and 7 show the impacts of the qualification threshold (Q_{Th}), prediction window (W_p), and minimum number of ads constraint (N) on the average deviation and PCRE, respectively. The results are shown for different stream merging techniques or server capacities. The effect of dynamic Q_{Th} is not shown because it does not perform well. As expected, both these metrics increase with Q_{Th} and W_p, which suggests that they should be chosen based on a good compromise. In contrast, the average deviation decreases with N (in the Any policy) up to a certain point ($N = 2$ or 3) and then starts to increase. Although PCRE always decreases with N, its value at 2 is close to that at 1. As will be shown later, N should be chosen based on the overall system performance and not only the average deviation.

Figure 8 compares the two alternative implementations of the prediction algorithm (ACA and ACA+APW) in terms of the average deviation. As mentioned earlier, ACA+APW gives expected times to all clients, whereas ACA cannot. The figure shows the average deviation results for three values of the prediction window (which does not affect APW). The figure demonstrates that the hybrid ACA+APW implementation has lower accuracy than ACA but higher than APW. The hybrid implementation is best used with a large value of the prediction window, so as to increase the percentage of clients receiving expected times by ACA (instead of APW).

Analysis of Deviation Distributions

We so far compared various prediction schemes in only the average deviation. We discuss now the distributions of the deviation results, so that we can compare various schemes in the range, standard deviation (σ), and confidence interval (CI). Table II shows the means, standard deviations, and the 90% confidence intervals for various schemes. The results for ACA and ACA+APW are shown for three values of the prediction window. As expected, ACA provides the smallest standard deviation, and the shortest confidence interval, and these values increase with the prediction window. The hybrid scheme performs better than APW in terms of the standard deviation and the confidence interval but is worse than ACA. Note that the means of the distribution can be positive or negative. A negative value indicates a stronger negative deviation component, whereas a positive value indicates a stronger positive deviation component. A negative deviation means that a user waits shorter than expected, while a positive deviation means waiting longer than expected. It is possible to assign different weights for negative and positive deviations, but in this paper we treat them equally.

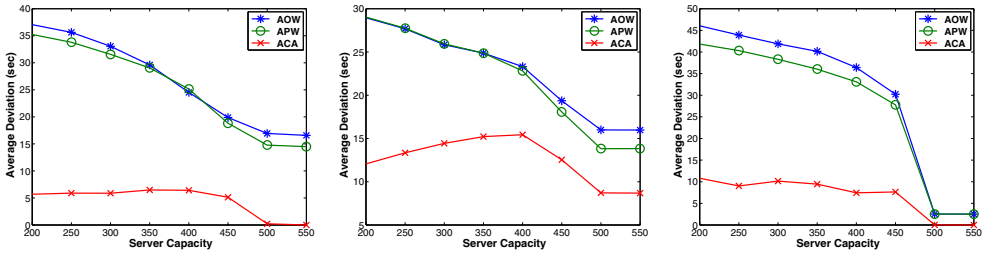
TABLE II: Summary of Deviation Distributions [ERMT, Any 2, Model A, 350 Channels, Units]

Scheme	Mean (Ads)	σ (Ads)	90% CI (Ads)
APW	0.0234	1.2909	[-3.1887,3.8113]
ACA, W_p=1	0.0852	0.4914	[0,1]
ACA, W_p=3	0.1193	0.6259	[0,1]
ACA, W_p=5	-0.0151	0.7601	[-1,1]
ACA+APW, W_p=1	0.3359	1.1843	[-2.0567,2.9433]
ACA+APW, W_p=3	0.5028	1.0609	[-2.243,2.757]
ACA+APW, W_p=5	0.2761	1.1038	[-2.0,3.0]

Overall Analysis of Pricing Scheme and ACA Algorithm

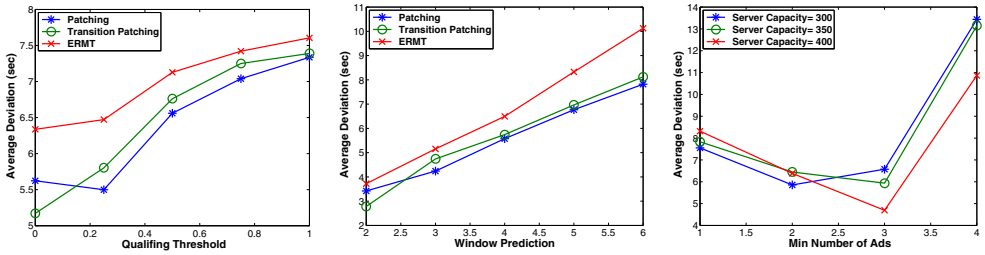
Let us now discuss the overall impact of the pricing scheme, applied in conjunction with the ACA algorithm, under the willingness-based arrival rate model. Figure 9 compares MAT, Each 2, and Any 2 in six different performance metrics. For a change, the results are shown for waiting tolerance Model B. The results with Model A are relatively the same, although they have slightly higher defection rates and slightly shorter waiting times. Any 2 is consistently the best overall performer in terms of the defection rate, profit, and revenue. Although Each 2 lowers the price the most (as shown in Figure 9(d)) due to the largest ads' viewing time, it yields lower profit and revenue because the increase in the defection rate is more significant.

Figure 10 depicts the impact of the minimum number of ads



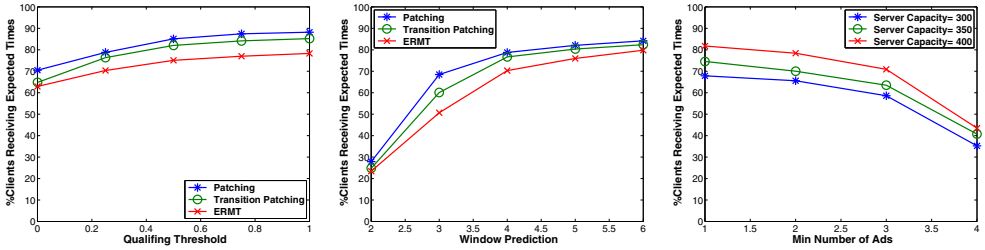
(a) Any 2 (b) Each 2 (c) MAT

Fig. 5: Comparing the Effectiveness of Various Prediction Approaches [ERMT, Model A]



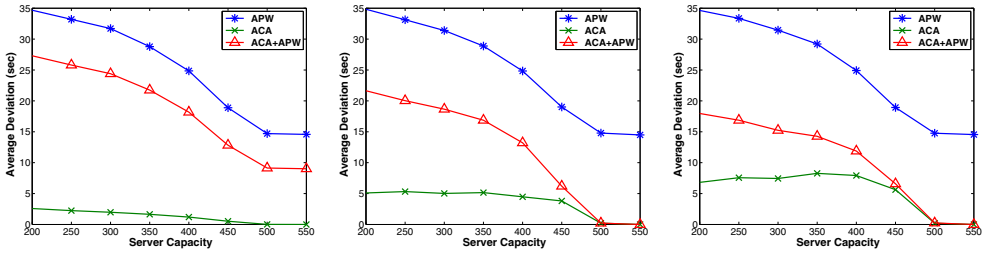
(a) Qualification Threshold (b) Prediction Window (c) Minimum Number of Ads

Fig. 6: Impacts of Design Parameters on Prediction Accuracy [ACA, Any, Model A, when not shown: $N = 2$]



(a) Qualification Threshold (b) Prediction Window (c) Minimum Number of Ads

Fig. 7: Impacts of Design Parameters on PCRE [ACA, Any, Model A, when not shown: $N = 2$]



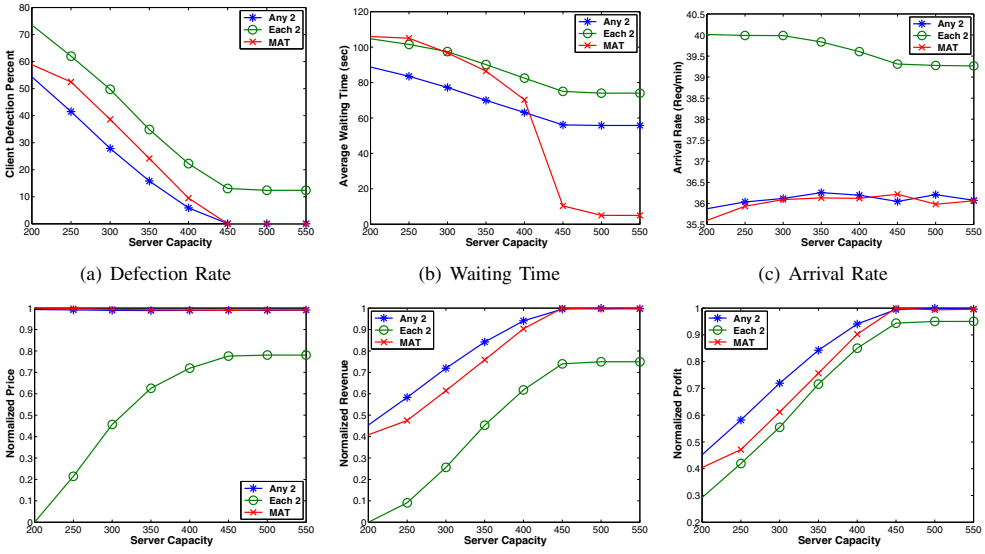
(a) $W_p = 1$ (b) $W_p = 3$ (c) $W_p = 5$

Fig. 8: Comparing the Two Alternative Implementations of the Prediction Algorithm [ERMT, Any 2, Model A]

constraint (N) in the Any scheduling policy on defection rate, average waiting time, and normalized profit. As expected, both the defection probability and the average waiting time increase with N. The defection rate and the average waiting time (which include the ads' viewing time) have conflicting impacts on the profit. The profit increases with the ads' viewing time and decreases with the defection rate (because it affects the arrival rate in the long run). The overall impact on the profit is plotted in Figure 10(c). It is clearly evident that the values 1 and 2 yield the highest profit, with 2 being slightly better.

VI. Conclusions

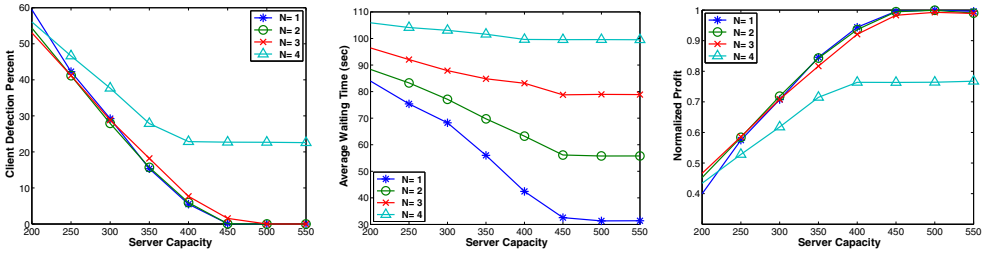
The main results can be summarized as follows. (1) The proposed ACA waiting-time prediction algorithm achieves high accuracy, and it is best when combined with MCF-P (Any N) scheduling policy. The prediction accuracy in this case is within 25% of an ad length. (2) When combined with the prediction algorithm, MCF-P (Any N) also gives the best overall performance among all considered scheduling policies in terms of client defection probability, revenue, and profit. (3) The value of the minimum numbers of ads constraint (N)



(a) Defection Rate (b) Waiting Time (c) Arrival Rate

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

Fig. 9: Comparing the Effectiveness of Scheduling Policies under the Willingness-Based Model [ACA, ERMT, Model B]



(a) Defection Rate (b) Waiting Time (c) Normalized Profit

Fig. 10: Impact of the Minimum Ads Constraint under the Willingness Model [ERMT, Any, Model B, $Q_{Th} = 0.2$, $W_p = 4$]

should be selected carefully. In this study, a value of 2 leads to the best performance. (4) The prediction algorithm can give an expected waiting time to each client by using a hybrid implementation (ACA+APW). In this case, the prediction window should be set to a large value to reduce the negative impact on accuracy. The accuracy remains within half an ad length for realistic server capacities.

REFERENCES

[1] C. Wu, B. Li, and S. Zhao, "Diagnosing network-wide P2P live streaming inefficiencies," in *Proc. of IEEE INFOCOM*, April 2009.

[2] V. Aggarwal, R. Caldebank, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and F. Yu, "The effectiveness of intelligent scheduling for multicast video-on-demand," in *Proc. of ACM Multimedia*, 2009, pp. 421–430.

[3] S. Ratnasamy and A. Ermolinskiy, "Revisiting IP-multicast," in *Proceedings of ACM SIGCOMM*, September 2006.

[4] K. Almeroth, "Multicast help wanted: From where and how much? Keynote Speech, Workshop on Peer-to-Peer Multicasting, Consumer Communications and Networking Conference, January 2007."

[5] L. Shi, P. Sessini, A. Mahanti, Z. Li, and D. L. Eager, "Scalable streaming for heterogeneous clients," in *Proc. of ACM Multimedia*, Oct. 2006, pp. 337–346.

[6] N. J. Sarhan, M. A. Alsmirat, , and M. Al-Hadrusi, "Waiting-time prediction in scalable on-demand video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 6, no. 2, 2010.

[7] M. Al-Hadrusi and N. J. Sarhan, "A scalable delivery framework and a pricing model for streaming media with advertisements," in *Proc. of SPIE/ACM Multimedia Computing and Networking Conference (MMCN)*, Januray 2008.

[8] S. Jagannathan and K. C. Almeroth, "The dynamics of price, revenue, and system utilization," in *Proc. of the IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, 2001, pp. 329–344.

[9] Y. Cai and K. A. Hua, "Sharing multicast videos using patching streams," *Multimedia Tools and Applications journal*, vol. 21, no. 2, pp. 125–146, Nov. 2003.

[10] Y. Cai, W. Tavanapong, and K. A. Hua, "Enhancing patching performance through double patching," in *Proc. of 9th Intl Conf. on Distributed Multimedia Systems*, Sept. 2003, pp. 72–77.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for Video-on-Demand servers," in *Proc. of ACM Multimedia*, Oct. 1999, pp. 199–202.

[12] 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.

[13] 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*, Oct. 1999, pp. 104–109.

[14] P. Basu and T. D. C. Little, "Pricing considerations in Video-on-Demand systems," in *Proc. of ACM Multimedia*, Nov. 2000, pp. 359–361.

[15] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "Does Internet media traffic really follow Zipf-like distribution?" *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 359–360, 2007.