# Performance and Waiting-Time Predictability Analysis of Design Options in Cost-Based Scheduling for Scalable Media Streaming*

Mohammad A. Alsmirat and Nabil J. Sarhan

Department of Electrical and Computer Engineering, Wayne State University
5050 Anthony Wayne Drive, Detroit, MI 48202, USA

**Abstract.** Motivated by the impressive performance of cost-based scheduling for media streaming, we investigate its effectiveness in detail and analyze opportunities for further tunings and enhancements. Guided by this analysis, we propose a highly efficient enhancement technique that optimizes the scheduling decisions to increase the number of requests serviced concurrently and enhance user-perceived quality-of-service. We also analyze the waiting-time predictability achieved by the new technique. The simulation results show that it can achieve significant performance improvements while providing users with highly accurate expected waiting times.

**Key words:** Scheduling, stream merging, video streaming, waiting-time prediction

## 1 Introduction

Media streaming applications have grown dramatically in popularity. This paper considers video streaming of pre-recorded content. Unfortunately, the number of video streams that can be supported concurrently is highly constrained by the stringent requirements of multimedia data, which require high transfer rates and must be presented continuously in time. Stream merging techniques [1–9] address this challenge by aggregating clients into successively larger groups that share the same multicast streams. These techniques include *Patching* [1, 5], *Transition Patching* [2, 6], and *Earliest Reachable Merge Target*(ERMT) [3, 4]. Periodic broadcasting techniques [10, 11] (and references within) also address this challenge but can be used for only popular videos and require the requests to wait until the next broadcast time of the first corresponding segment. This paper considers the stream merging approach.

The degrees of resource sharing achieved by stream merging depend greatly on how the waiting requests are scheduled for service. Despite the many proposed stream merging techniques and the numerous possible variations, there has been only little work on the issue of scheduling in the context of these scalable techniques. We stress that the choice of a scheduling policy can be as important as or even more important than the choice of a stream merging technique, especially when the server is loaded. *Minimum*

* This work is supported in part by NSF grant CNS-0626861.

*Cost First* (MCF) [12] is a cost-based scheduling policy that has recently been proposed for use with stream merging. MCF captures the significant variation in stream lengths caused by stream merging through selecting the requests requiring the least cost. *Predictive Cost-based Scheduling* (PCS) [13] is another cost-based scheduling policy that has been proposed to enhance the server bandwidth utilization. PCS predicts future system state and uses the prediction results to potentially alter the scheduling decisions. It delays servicing requests at the current scheduling time (even when resources are available) if it is expected that shorter stream will be required at the next scheduling time. Cost-based scheduling policies perform significantly better than all other scheduling policies.

Motivated by the excellent performance of cost-based scheduling, we investigate here its effectiveness in great detail and discuss opportunities for further tunings and enhancements. In particular, we seek to answer the following two important questions. First, is it better to consider the stream cost only at the current scheduling time or consider the expected overall cost over a future period of time? Second, should the cost computation consider future stream extensions done by ERMT to satisfy the needs of new requests? Based on our detailed analysis, we propose a highly efficient technique, called *Adaptive Regular Stream Triggering* (ART), and analyze its effectiveness when it is used with other scheduling policies. Moreover, we analyze the waiting-time predictability of MCF when ART is applied. Motivated by the rapidly growing interest in human-centered multimedia, the ability to inform users about how long they need to wait for service has become of great importance [14]. Today, even for short videos with medium quality, users of online video websites may experience significant delays. Providing users with waiting-time feedback enhances their perceived QoS and encourages them to wait, thereby increasing throughput.

We study the effectiveness of different cost-computation alternatives and ART through extensive simulation. The analyzed performance metrics include customer defection (i.e. turn-away) probability, average waiting time, unfairness against unpopular videos, average cost per request, waiting-time prediction accuracy, and percentage of clients receiving expected waiting times. The waiting-time prediction accuracy is determined by the average deviation between the expected and actual waiting times. Moreover, we consider the impacts of customer waiting tolerance, server capacity, request arrival rate, number of videos, and video length. The results demonstrate that the proposed ART technique significantly enhances system throughput and reduces the average waiting time for service. Additionally, it can provide accurate expected waiting time to a larger number of clients. The rest of the paper is organized as follows. Section 2 provides background information. Section 3 analyzes cost-based scheduling, explores alternative ways to compute the cost, and presents the proposed ART technique. Section 4 discusses the performance evaluation methodology and Section 5 presents and analyzes the main results.

## 2 Background Information and Preliminary Analysis

### 2.1 Stream Merging

Stream merging techniques aggregate clients into larger groups that share the same multicast streams. In this subsection, we discuss three main stream merging techniques:

Patching [5], Transition Patching [2], and *Earliest Reachable Merge Target* (ERMT) [3]. With Patching, a new request joins immediately the latest multicast stream for the object and receives the missing portion as a *patch* using a unicast stream. 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. Since patch streams are not sharable with later requests and their cost increases with the temporal skew to the latest multicast stream, it is more cost-effective to start new full multicast stream (also called *regular stream*) after some time. Thus, when the patch stream length exceeds a certain value called *regular window* ($Wr$), a new regular stream is initiated instead. Transition Patching allows some patch streams to be sharable by extending their lengths. It introduces another multicast stream, called *transition patch*. The threshold to start a regular stream is $Wr$ as in Patching, and the threshold to start a transition patch is called the *transition window* ($Wt$). A transition patch is shared by all subsequent patches until the next transition patch. ERMT is a near optimal hierarchical stream merging technique. Basically, 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 [3]. ERMT performs better than other hierarchical stream merging alternatives and close to the optimal solution.

### 2.2 Request Scheduling

A scheduling policy selects an appropriate video 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 waiting requests for the selected video can be serviced using only one channel. The number of channels is referred to as *server capacity*. All scheduling policies are guided by one or more of the following primary objectives:($i$) minimize the overall customer defection (turn-away) probability, ($ii$) minimize the average request waiting time, and ($iii$) minimize unfairness. The defection probability is the probability that a user leaves the system without being serviced because of a waiting time exceeding the user's tolerance. It is the most important metric because it translates directly to the number of customers that can be serviced concurrently and to server throughput. The second and the third objectives are indicators of customer-perceived QoS. It is usually desirable that the servers treat equally the requests for all videos. Unfairness measures the bias of a policy against unpopular videos.

The main scheduling policies include *First Come First Serve* (FCFS) [15], *Maximum Queue Length* (MQL) [15], *Minimum Cost First* (MCF) [12], and *Predictive Cost-Based Scheduling* (PCS) [13]. FCFS selects the video with the oldest waiting request, whereas MQL selects the video with the largest number of waiting requests. MCF policy has been recently proposed to exploit the variations in stream lengths caused by stream merging techniques. *MCF-P* (P for "Per"), the preferred implementation of MCF, selects the video with the least cost per request. The objective function here is $F(i) = \frac{L_i \times R_i}{N_i}$, where $L_i$ is the required stream length for the requests in queue $i$, $R_i$ is the (average) data rate for the requested video, and $N_i$ is the number of waiting requests for video $i$. To reduce the bias against videos with higher data rates, $R_i$ can be removed from the objective function (as done in this paper). MCF-P has two variants: *Regular as Full* (RAF) and *Regular as Patch* (RAP). RAP treats regular and transition streams

as if they where patches, whereas RAF uses their normal costs. MCF-P performs significantly better than all other scheduling policies when stream merging techniques are used. PCS is based on MCF, but it predicts future system state and uses this prediction to possibly alter the scheduling decisions. When a channel becomes available, PCS determines using the MCF-P objective function the video $V_{Now}$ which is to be serviced tentatively at the current scheduling time ($T_{Now}$) and its associated delivery cost. Before actually servicing that video, it predicts the system state at the next scheduling time ($T_{Next}$) and estimates the delivery cost at that time assuming that video $V_{Now}$ is not serviced at time $T_{Now}$. PCS does not service any request at time $T_{Now}$ and thus postpone the service of video $V_{Now}$ if the delivery cost at time $T_{Next}$ is lower than that at time $T_{Now}$. Otherwise, video $V_{Now}$ is serviced immediately. To reduce possible channel underutilization, PCS delays the service of streams only if the number of available server channels is smaller than a certain threshold. PCS has two alternative implementations: *PCS-V* and *PCS-L*, which differ in how to compute the delivery cost or required stream length at the next scheduling time. They perform nearly the same and thus we consider only PCS-V because of its simplicity.

## 3 Alternative Cost Computations

### 3.1 Analysis of Cost-Based Scheduling

We seek to understand the behavior of cost-based scheduling and its interaction with stream merging. Understanding this behavior helps in developing solutions that optimize the overall performance. One of the issues that we explore in this study is determining the duration during which the cost should be computed. In particular, we seek to determine whether the cost should be computed only at the current scheduling time ($T_{sched}$) or over a future duration of time, called *prediction window* ($W_p$). In other words, should the system select the video with the least cost per request at time $T_{sched}$ or the least cost per request during $W_p$. The latter requires prediction of the future system state.

We devise and explore two ways to analyze the effectiveness of computing the cost over a period of time: *Lookahead* and *Combinational*. In Lookahead, the service rate (which is the rate at which a video gets serviced) is computed dynamically for each video that has waiting requests. The total cost for servicing each one of these videos is computed during $W_p$. Lookahead Scheduling selects the video $j$ that minimizes the expected cost per request. Thus, the objective function to minimize is $F(j) = \frac{\sum_{i=1}^{n} C_i}{\sum_{i=1}^{n} N_i}$, where *n* is the number of expected service times for video $j$ during $W_p$, $C_i$ is the cost required to service the requests for video $j$ at service time $i$, and $N_i$ is the number of requests expected to be serviced at service time $i$. The number of requests at future service times is predicted by dynamically computing the arrival rate for each video. As discussed earlier, ERMT may extend streams to satisfy the needs of new requests. MCF-P, however, does not consider later extensions in computing the cost. In analyzing cost-based scheduling, we also need to consider whether it is worthwhile to predict and consider these later extensions. Hence, we consider a variant of Lookahead Scheduling that considers these extensions.

In contrast with Lookahead Scheduling, Combinational scheduling predicts the best sequence in which various videos should be serviced and performs scheduling based on this sequence. Thus, it considers any correlations on the cost among successive video selections. The best sequence is found by generating all possible sequences for the next *n* stream completion times during $W_p$, for only the *n*-best videos according to the MCF-P objective function. The objective function of each sequence is then calculated. Consider the sequence $S_j = \{V_1, V_2, V_3, ..., V_n\}$, where $V_i$ is the video selected to be serviced at the next $i^{th}$ stream completion time. The objective function for this sequence is $F(S_j) = \frac{\sum_{i=1}^{n} C_{V_i}}{\sum_{i=1}^{n} N_{V_i}}$, where $C_{V_i}$ is the cost required to service $V_i$, and $N_{V_i}$ is the number of waiting requests for that video. $C_{V_i}$ is determined based on the used MCF-P variant.Combinational Scheduling chooses the sequence that is expected to lead to the least overall cost. Although many optimizations are possible to reduce the implementation complexity, we focus primarily on whether exploiting the correlations between successive video selections is indeed important in practical situations.

### 3.2 Proposed Adaptive Regular Stream Triggering (ART)

As will be shown later, our extensive analysis reveals a significant interaction between stream merging and scheduling decisions. One of the pertaining issues is how to best handle regular (i.e., full) streams. MCF-P (RAP) considers the cost of a regular stream as a patch and thus treats it in a differentiated manner. The question arises as to whether it is worthwhile, however, to delay regular streams in certain situations. Guided by extensive analysis, we propose a highly efficient technique, called *Adaptive Regular Stream Triggering* (ART). A possible implementation is shown in Figure 1. The basic idea here is to delay regular streams as long as the number of free channels is below a certain threshold, which is to be computed dynamically based on the current workload and system state.

```
V_Now = find the video that will be serviced at T_Now;
if (freeChannels ≥ freeChannelThresh)
    Service the requests for V_Now;
else {
    currStreamLen =
      find the required stream length to serve V_Now at T_Now;
    if (currStreamLen < movieLen) // not a full stream
        Service the requests for V_Now;
    else //full stream
        Postpone the requests for V_Now;
}
```

Fig. 1: Simplified Implementation of ART

Figure 2 shows an algorithm to dynamically find the best value of $freeChannelThresh$. As in [13], the algorithm changes the value of the threshold and observes its impact on

```
// This algorithm executed periodically
currDefectionRate = defectedCustomers/servedCustomers;
if (currDefectionRate < lastDefectionRate) {
    if (last action was decrement
        and freeChannelThresh > 2)
        freeChannelThresh − −;
    else if (last action was increment)
        freeChannelThresh + +;
} else if (currDefectionRate > lastDefectionRate){
    if (last action was increment
        and freeChannelThresh > 2)
        freeChannelThresh − −;
    else if (last action was decrement)
        freeChannelThresh + +;
}
lastDefectionRate = currDefectionRate;
```

Fig. 2: Simplified Algorithm for Computing *freeChannelThresh* Dynamically [13]

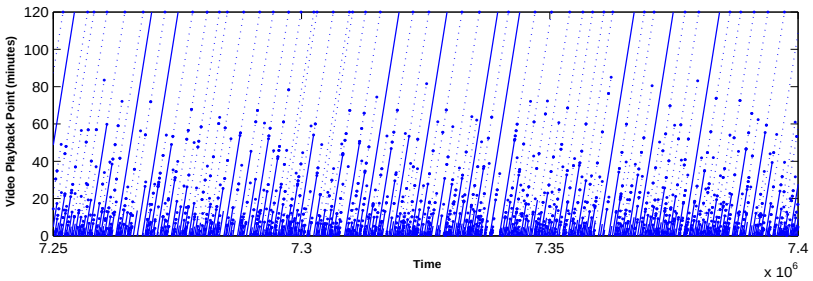
customer defection (i.e., turn-away) probability over a certain time interval. The value of the threshold is then updated based on the trend in defection probability (increase or decrease) and the last action (increase or decrease) performed on the threshold. The algorithm is to be executed periodically but not frequently to ensure stable system behavior.

To further demonstrate the main idea of ART, Figure 3 plots the ERMT merge tree without and with ART, respectively. The solid lines show the initial stream lengths and the dotted lines show later extensions. The circles identify successive extensions. With ART, we can see a gap before the initiation of a regular stream because of the postponement. We can also observe that the number of initial regular streams in the merge tree (called *I Streams* in this paper) is relatively much smaller with ART. For example, there is only one *I Stream* in the merge tree with ART while there are many more *I Streams* in the merge tree without ART.

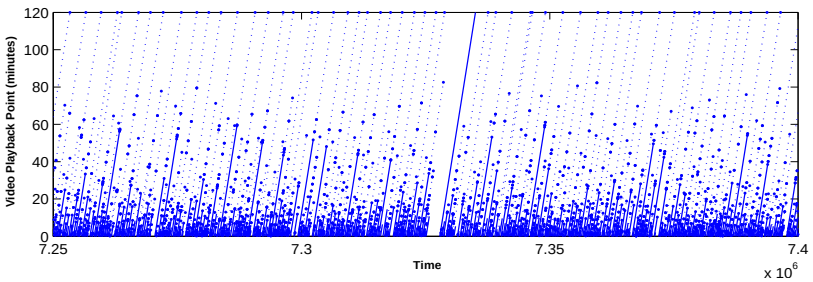
## 4 Evaluation Methodology

We study the effectiveness of the proposed policy through simulation. The simulation stops after a steady state analysis with $95\%$ confidence interval is reached. Table 1 summarizes the workload characteristics used.

We characterize the waiting tolerance of customers by three models: A, B, and C. In *Model A*, the waiting tolerance follows an exponential distribution with mean $\mu_{tol}$. In *Model B*, users with expected waiting times less than $\mu_{tol}$ will wait and the others will have the same waiting tolerance as Model A. We use *Model C* to capture situations in which users either wait or defect immediately depending on the expected waiting times. The user waits if the expected waiting time is less than $\mu_{tol}$ and defects immediately if the waiting time is greater than $2\mu_{tol}$. Otherwise, the defection probability decreases linearly from $1$ to $0$ for the expected waiting times between $\mu_{tol}$ and $2\mu_{tol}$.




(a) MCF-P without ART




(b) MCF-P with ART

Fig. 3: Impact of ART on ERMT Stream Merge Tree [*Video 11, MCF-P, Server Capacity = 450*]

Table 1: Summary of Workload Characteristics

| **Parameter** | **Model/Value(s)** |
|---|---|
| Request Arrival | Poisson Process |
| Request Arrival Rate | Variable, Default is 40 Req./min |
| Server Capacity | 300 to 750 channels |
| Video Access | Zipf-Like with $\theta = 0.271$ |
| Number of Videos | 120 |
| Video Length | 120 min |
| Waiting Tolerance Model | A, B, C, Default is A |
| Waiting Tolerance Mean ($\mu_{tol}$) | Variable, Default is 30 sec |

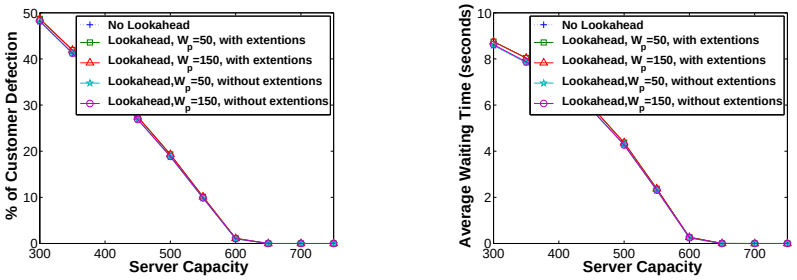
The analyzed performance metrics include customer defection probability, average waiting time, and unfairness against unpopular videos. The waiting-time predictability is analyzed by two metrics: waiting-time prediction accuracy and the percentage of clients receiving expected waiting times. The waiting-time prediction accuracy is determined by the average deviation between the expected and actual waiting times. For waiting-time prediction, we use the algorithm in [14]. Note that this algorithm may not provide an expected waiting time to each client because the prediction may not always be performed accurately.

# 5 Result Presentation and Analysis

## 5.1 Comparing Different Alternatives for Computing the Cost

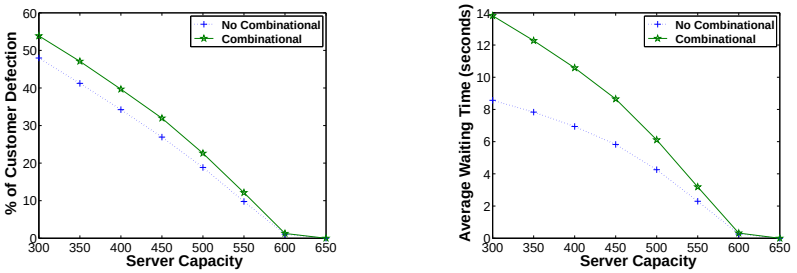
Let us start by studying the effectiveness of Lookahead and Combinational Scheduling. Interestingly, there is no clear benefit for computing the cost over a future period of time. In some cases, as shown in Figures 4 and 5, the performance in term of customers defection and average waiting time may be worse than when computing the cost at the current scheduling time. Although computing the cost over a time interval seems intuitively to be an excellent choice, it interferes negatively with stream merging.

Next, we discuss how the interaction between stream merging and scheduling can be utilized by using the proposed ART technique, which can be used with any scheduling policy. Based on the prior results, we only consider next computing the cost at the current scheduling time.




(a) Defection Rate (b) Average Waiting Time

Fig. 4: Effectiveness of Lookahead Scheduling, with and without Future Extensions [*ERMT*]

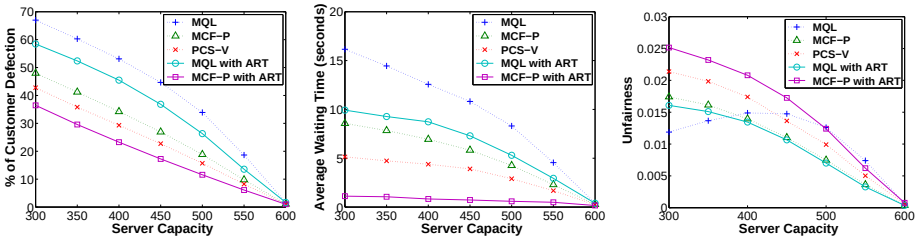



(a) Defection Rate (b) Average Waiting Time

Fig. 5: Effectiveness of Combinational Scheduling [*ERMT, $W_p$ = 4 next stream completion times*]

## 5.2 Effectiveness of ART

Figure 6 shows the effectiveness of the proposed ART technique when ERMT is used. With MCF-P, ART reduces the defection probability and average waiting time by up to $25\%$ and $80\%$, respectively. It also yields significant improvements when used with

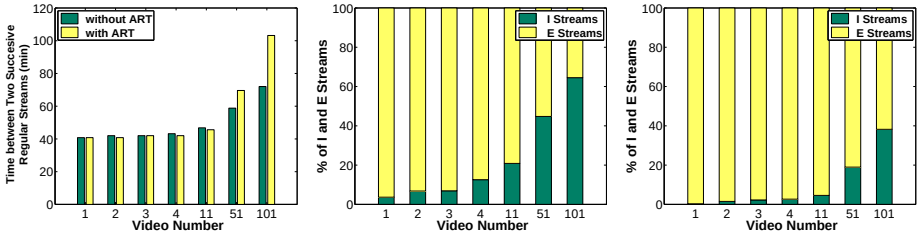



(a) Defection Rate (b) Average Waiting Time (c) Unfairness

Fig. 6: Effectiveness of ART [*ERMT*]

MQL. Moreover, MCF-P when combined with ART performs better than PCS-V in terms of the defection probability and average waiting time. Unfairness, the least important metric, is a little larger with ART because of its nature in favoring videos with shorter streams, but it is still acceptable compared with MQL and PCS-V. These results indicate that MCF-P when combined with ART is the best overall performer.

Figure 7 depicts the impact of ART on regular streams in ERMT. We observe that when ART postpones regular streams, it forces ERMT to make more merges, which in turn, increases system utilization. We can also observe that the number of regular streams does not decrease significantly despite of postponing these streams. In contrast, Figure 7(a) indicates that the average time between two successive regular streams for popular videos is even smaller with ART than that without it. ERMT keeps extending streams, which eventually become regular streams. Figures 7(b) and 7(c) compare the percentage of initial regular streams (I Streams) and extended regular streams (E Streams) without and with ART, respectively. We can see that the percentage of extended regular streams with ART is much higher. This supports the fact that the number of regular streams is not reduced by postponing. In summary, we can say that ART improves ERMT by replacing many *I Streams* with *E Streams*. To further support the fact that more customers are served with only one stream when using ART, Figure 8 demonstrates the impact of ART on the cost per request. We can see that the cost per request with ART is lower for different server capacities.




(a) Avg. Time between Two Regular Streams (b) Initial and Extended Regular Streams without ART (c) Initial and Extended Regular Streams with ART

Fig. 7: Impact of ART on Regular Streams [*ERMT, MCF-P, Server Capacity = 450*]

The results for Transition Patching and Patching have the same behavior as ERMT and thus are not shown for space limitation. Patching results can be found partially in Figures 10 and 11. As with ERMT, ART reduces significantly the customer defection rate and the average waiting time when it is combined with MCF-P (RAP) and MQL. MCF-P (RAP) combined with ART gives almost the same results as PCS-V in terms of customer defection probability, but it reduces the average waiting time significantly. Unfairness with ART is a little larger but still acceptable compared with that of MQL for medium and high server capacities and remains less than that of PCS-V.

Interestingly, ART improves Transition Patching and Patching despite that their best scheduling policy, MCF-P (RAP), depends on a conflicting principle. As discussed earlier, MCF-P (RAP) gives preference to regular streams while ART postpones them in certain situations. As illustrated in Figure 9, the main impact of ART is dynamically optimizing $Wr$, which is larger than that of MCF-P (RAP) and smaller than that of MCF-P (RAF) for popular videos, and even greater than that of MCF-P (RAF) for unpopular videos. With PCS-V, $Wr$ values are very close to that of MCF-P (RAP). The horizontal line in the figure marks the equation-based value of $Wr$ [16]. (Note that the equation does not yield optimum values because it is based on important simplifying assumptions.)

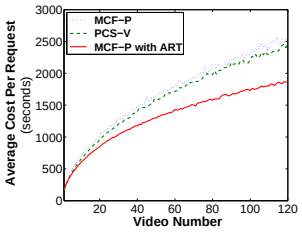
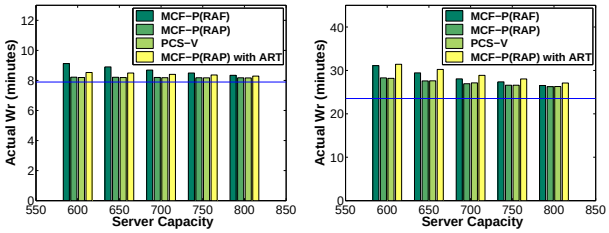



Fig. 8: Comparing the Impact of PCS and ART on Cost per Request [*ERMT, MCF-P, Server Capacity = 300*]

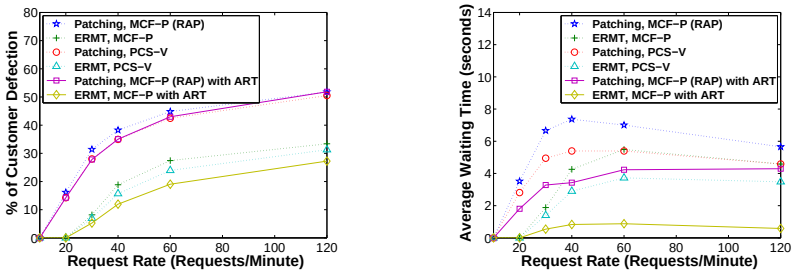



(a) Video 1 (b) Video 20

Fig. 9: Comparing Actual *Wr* in MCF-P (RAF), MCF-P (RAP), PCS-V and MCF-P (RAP) with ART [*Patching*]

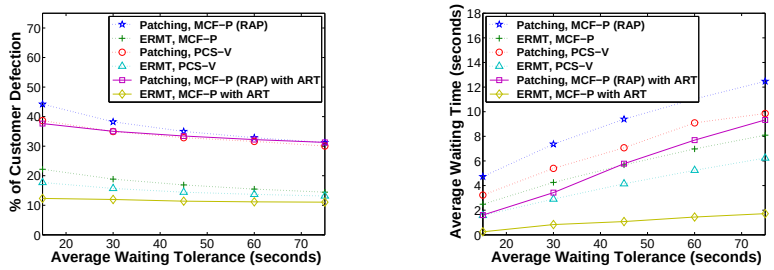
### 5.3 Impact of Workload Parameters on the Effectiveness of ART

Figures 10 and 11 illustrate the impact of the request arrival rate and customer waiting tolerance on the effectiveness of ART. The results for both Patching and ERMT are shown. We have also studied the impacts of the number of videos and video length, but the figures are not shown to save space. The results demonstrate that ART always yields significant performance improvements under all studied workload parameters. In addition, ART always achieves smaller customer defection probability and average waiting time than PCS-V in the case of ERMT. In Patching, the same trend is observed for the average waiting time, but PCS-V and "MCF-P combined with ART" perform nearly the same in terms of customer defection probability, especially when the server is highly loaded.



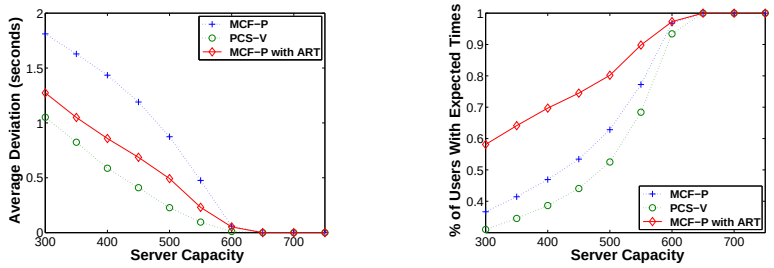

(a) Defection Rate (b) Average Waiting Time

Fig. 10: Impact of Request Arrival Rate [*Server Capacity = 500*]




(a) Defection Rate (b) Average Waiting Time

Fig. 11: Impact of Customer Waiting Tolerance [*Server Capacity = 500*]




(a) Average Deviation (b) Customer With Expected Time

Fig. 12: Waiting-Time Predictability of MCF-P, MCF-P with ART, and PCS-V [*ERMT*, $W_p = 0.5\mu_{tol}$ , *Model B*]

### 5.4 Waiting-Time Predictability with ART

Figure 12 compares the predictability of MCF-P, MCF-P combined with ART, and PCS-V in terms of the average deviation and percentage of clients receiving expected time of service (PCRE) under waiting tolerance Model B. The results with Model C are similar and thus are not shown to save space. The results demonstrate that ART significantly improves the predictability of MCF-P. In particular, ART reduces the average deviation by up to $30\%$ and $75\%$ for models B and C, respectively. It also increases the number of clients receiving expected times by up to $35\%$. Moreover, MCF-P combined with ART gives more customers expected times than PCS-V with a relatively less significant increase in the average deviation.

## 6 Conclusions

We have analyzed cost-based scheduling for scalable video streaming. The results indicate that there is no clear advantage of computing the cost over a future time window, compared with computing the cost only at the next scheduling time. The results also show that the proposed ART technique substantially improves the customer defection probability and the average waiting time. With ART, significantly more clients can receive expected waiting time for service, but at a somewhat lower waiting time accuracy.

## References

1. Hua, K.A., Cai, Y., Sheu, S.: Patching: A multicast technique for true Video-on-Demand services. In: Proc. of ACM Multimedia. (1998) 191–200
2. Cai, Y., Hua, K.A.: An efficient bandwidth-sharing technique for true video on demand systems. In: Proc. of ACM Multimedia. (Oct. 1999) 211–214
3. Eager, D.L., Vernon, M.K., Zahorjan, J.: Optimal and efficient merging schedules for Video-on-Demand servers. In: Proc. of ACM Multimedia. (Oct. 1999) 199–202
4. Eager, D.L., Vernon, M.K., Zahorjan, J.: Bandwidth skimming: A technique for cost-effective Video-on-Demand. In: Proc. of Multimedia Computing and Networking Conf. (MMCN). (Jan. 2000) 206–215
5. Cai, Y., Hua, K.A.: Sharing multicast videos using patching streams. Multimedia Tools and Applications journal **21**(2) (Nov. 2003) 125–146
6. Cai, Y., Tavanapong, W., Hua, K.A.: Enhancing patching performance through double patching. In: Proc. of 9th Intl Conf. on Distributed Multimedia Systems. (Sept. 2003) 72–77
7. Rocha, M., Maia, M., Cunha, I., Almeida, J., Campos, S.: Scalable media streaming to interactive users. In: Proc. of ACM Multimedia. (Nov. 2005) 966–975
8. Ma, H., Shin, G.K., Wu, W.: Best-effort patching for multicast true VoD service. Multimedia Tools Appl. **26**(1) (2005) 101–122
9. Qudah, B., Sarhan, N.J.: Towards scalable delivery of video streams to heterogeneous receivers. In: Proc. of ACM Multimedia. (Oct. 2006) 347–356
10. Huang, C., Janakiraman, R., Xu, L.: Loss-resilient on-demand media streaming using priority encoding. In: Proc. of ACM Multimedia. (Oct. 2004) 152–159
11. Shi, L., Sessini, P., Mahanti, A., Li, Z., Eager, D.L.: Scalable streaming for heterogeneous clients. In: Proc. of ACM Multimedia. (Oct. 2006) 337–346
12. Sarhan, N.J., Qudah, B.: Efficient cost-based scheduling for scalable media streaming. In: Proc. of Multimedia Computing and Networking Conf. (MMCN). (January 2007)
13. Alsmirat, M., Sarhan, N.J.: Predictive cost-based scheduling for scalable media streaming. In: Proc. of IEEE International Conference on Multimedia and Expo. (June 2008)
14. Alsmirat, M., Al-Hadrusi, M., Sarhan, N.J.: Analysis of waiting-time predictability in scalable media streaming. In: Proc. of ACM Multimedia. (Sept. 2007)
15. Dan, A., Sitaram, D., Shahabuddin, P.: Scheduling policies for an on-demand video server with batching. In: Proc. of ACM Multimedia. (Oct. 1994) 391–398
16. Qudah, B., Sarhan, N.J.: Analysis of resource sharing and cache management techniques in scalable video-on-demand. In: Proc. of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). (Sept. 2006) 327–334