Frame-Based Periodic Broadcast and Fundamental Resource Tradeoffs

Subhabrata Sen[1], Lixin Gao[2], and Don Towsley[1]

[1] Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{sen,towsley}@cs.umass.edu

[2] Dept. of Computer Science
Smith College
Northampton, MA 01060
gao@cs.smith.edu

Technical Report 99-78
Department of Computer Science

Abstract

Multimedia streaming applications consume significant amounts of server and network resources due to the high bandwidth requirements and long duration of audio and video clips. For popular videos, periodic broadcast techniques [1–4] exploit the client's buffer space and network reception bandwidth capabilities to reduce server and network transmission bandwidth requirements to serve a new client, while guaranteeing that the playback startup latency does not exceed a prescribed threshold. In this paper, we systematically study the fundamental tradeoffs among the server and network transmission bandwidths, client buffer, client reception bandwidth, and the playback startup delay for periodic broadcast. We adopt a fine-grained fluid transmission model and consider a class of *frame-based* periodic transmission schemes which assign a fixed transmission bandwidth to each frame in the video, and continuously transmit each frame according to this rate. The model accommodates both CBR and VBR streams. Using this framework, we consider both unconstrained and constrained client resources (buffer or bandwidth). For the above scenarios, we develop and explore broadcast schemes that minimize the transmission overhead while guaranteeing a particular playback delay to clients. We also consider the problem of using a *single* transmission scheme to satisfy clients with heterogeneous rate constraints, and develop an appropriate reception scheme to jointly minimize the client playback startup delay and client buffer requirements. Extensive evaluations of these different scenarios suggests that the design space defined by a playback startup delay of a few tens of seconds, client reception bandwidth equal to $2-4$ times the mean video bandwidth, client buffer space sufficient to accommodate $15-40\%$ of the video, and server and network transmission bandwidths of $3-5$ times the mean video bandwidth, is a feasible and very attractive operating region from the server, network, as well as client performance viewpoint. Last, we present a parsimonious extension of the frame-based broadcasting scheme where the server only transmits data that is required by some client. Evaluations suggest that for both CBR as well as VBR videos, the parsimonious scheme can yield significant savings in server and network bandwidth usage, particularly for small playback startup delay guarantees.

I. Introduction

The emergence of the Internet as a pervasive communication medium has fueled a dramatic convergence of voice, video and data on this new digital information infrastructure. A broad range of applications (entertainment and information services, distance learning, corporate telecasts, narrowcasts, etc.) is enabled by the ability to stream continuous media data from servers to clients across a high-speed network. However, due to the high bandwidth requirements ($4-6$ Mbps for full motion MPEG-2) and the long-lived nature (tens of minutes to $1-2$ hours) of digital video, server and network resource constraints (in particular, server I/O bandwidth and network bandwidth) are a major limiting factor in the widespread usage of video streaming over the Internet.

Many Internet applications have asynchronous clients that may request a video stream at different times. Provision of economically viable high-volume video services requires effective techniques that minimize the incremental cost of serving a new client, while also limiting the client playback start-up latency and the likelihood of rejecting requests due to resource constraints.

One approach to reducing the server and network resources is to allow multiple clients to receive all or part of a single transmission [1–6]. Several recently proposed techniques such as periodic broadcast and patching [1–4, 7] exploit the client's buffer space and the existence of sufficient client network bandwidth to listen to multiple simultaneous transmissions, to reduce server and network transmission bandwidth requirements while guaranteeing a bounded playback startup latency. Understanding the tradeoffs involving different resources on the client and server and network side and the tradeoffs between these resources and the playback startup delay is essential in designing a video streaming system with good performance. Achieving this understanding is the goal of this paper.

Prior work in periodic broadcast has focused on developing particular broadcast schemes tailored to specific points in the multidimensional resource space involving server and network transmission bandwidths, client buffer, client reception bandwidth, and playback startup delay. Existing broadcast schemes [1–5] exploit the fact that clients play back a video sequentially and, therefore, video data for a later portion of the video can be received later than the data for an earlier portion. The server divides a video object into multiple segments, and continuously broadcasts them on a collection of transmission channels. To limit playback startup latency, earlier portions of the video are broadcast more frequently than later ones. Clients listen to multiple channels at the same time and store future segments for later playback. *Workahead* reception ensures the continuous playback of the whole video.

In order to explore the space of resource tradeoffs, we adopt a fluid transmission model and consider a class of *frame-based* periodic broadcast schemes, in which the server transmits each frame continuously at a particular bandwidth. This model exploits the fact that different video frames have different playback deadlines, and assumes that a frame can be transmitted in arbitrarily small units. This frame-based fluid transmission model has several key advantages. First, such fine-grained transmission of the video data has the potential to maximize server and network resource savings by increasing the potential for sharing data among asynchronous clients. Intuitively, such a model facilitates developing broadcast schemes with superior performance than non-fluid periodic broadcast schemes using segmentation at a coarser granularity than one frame. Second, a frame is continuously broadcast as a fluid. This permits the offline computation of a *single* reception schedule that can be used by different clients (with identical bandwidth and buffer resources) requesting the same video at different times. This greatly aids in the development of optimal broadcast schemes for resource-constrained situations. Third, the model provides a natural framework for handling both Constant Bit Rate (CBR) and Variable Bit Rate (VBR) streams.

Using the above framework, we explore both constrained and unconstrained client resource (buffer or bandwidth) situations, and investigate how resource tradeoffs impact the performance of such broadcast schemes and the role played by the client playback startup delay in determining the server, network and client resource require-

ments. For each situation we (i) present a broadcast scheme that minimizes the server and network transmission bandwidth requirements, while guaranteeing a prescribed playback startup delay, and then (ii) use that scheme to explore the resource tradeoff space.

We first consider the case where there are no explicit constraints on the client buffer or reception bandwidth. We identify a periodic broadcast scheme, UBUR (unconstrained buffer unconstrained reception bandwidth), which provably minimizes the transmission bandwidth. We develop practical guidelines for determining operating regions of reasonable server, network and client resource requirements, and client startup delays. Evaluations using both CBR and VBR video traces suggest that a playback startup delay of a few tens of seconds to $2-3$ minutes is a very attractive operating region for efficient use of server and network bandwidth, as well as client reception bandwidth. In this region, the transmission (and peak reception bandwidth) usage is about $2-5$ times the mean video bandwidth, depending on the video length, and startup delay. The corresponding worst case client buffer occupancy is a considerable $37-40\%$ of the entire video.

We next investigate the resource tradeoffs space when clients have finite resources. We present the CBUR (constrained buffer unconstrained reception bandwidth) broadcast scheme that minimizes the transmission bandwidth requirements, when the client has insufficient buffer resources. We observe that most of the savings in server and network bandwidth can be accrued with relatively modest amounts of client buffer ($10-20\%$ of the video size), and for startup delays in the few tens of seconds range. We develop UBCR (unconstrained buffer constrained reception bandwidth), a heuristic greedy algorithm to minimize the transmission bandwidth requirements, for the case that client reception bandwidths are limited. We also consider the issue of handling resource heterogeneity, and explore using a *single* server transmission scheme to serve clients with heterogeneous resource constraints. We develop H-UBCR (heterogeneous, unconstrained buffer constrained reception bandwidth), a heuristic lazy client reception schedule to jointly minimize the playback startup delay and client buffer requirements, given any frame-based fluid server transmission schedule, and a particular client bandwidth constraint. Extensive evaluations across different scenarios suggest that the design space defined by a playback startup delay of a few tens of seconds, client reception bandwidth equal to $2-4$ times the mean video bandwidth, and client buffer space to accommodate $15-40\%$ of the video, and server and network transmission bandwidths of $3-5$ times the mean video bandwidth, is very attractive feasible operating region from the server, network, as well as client performance viewpoint.

Finally, pure periodic broadcast schemes, can waste some server and network transmission capacity, when client demand is light. We present a parsimonious extension (P-UBUR) of the frame-based UBUR broadcasting scheme where the server only transmits data that is required by some client. We analytically compute a closed form expression for the transmission overhead of this scheme. Evaluations suggest that this scheme can yield significant savings in transmission bandwidth usage for both CBR as well as VBR videos, particularly for small playback startup delay guarantees. We also observe that for parsimonious transmissions, startup delays have limited effectiveness in reducing transmission overheads if the arrival rate is not high.

The remainder of the paper is organized as follows. Section II introduces the formal model and key concepts used throughout the paper. Sections III-VI in order deal with unconstrained client resources, homogeneous

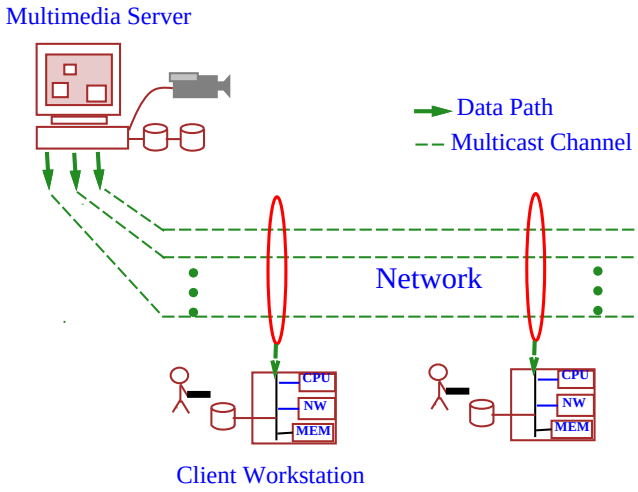


Fig. 1. **Video Broadcast Service:** Video streams originate at a multimedia server, and travel through the network, to multiple asynchronous clients. Periodic broadcast is performed at the server.

client resource constraints, heterogeneous client resource constraints, and parsimonious frame-based broadcast. Section VII describes related work and, finally, Section VIII concludes the paper.

II. Model Overview

Central to any effective periodic delivery scheme is the ability of the client to simultaneously listen to multiple transmission channels and to store frames ahead of their playback times. Periodic broadcast operates well within the buffer space and I/O bandwidth availability of today's end systems. Both per-byte storage cost and access latencies for main memory and disks are decreasing dramatically. In addition, system bus speeds are also increasing. Commodity PCs already offer a 100 MHz system bus and 64–128 MB of main memory, as well as several gigabytes of disk storage. These trends suggest that a significant fraction of client stations have enough high-bandwidth storage space to accommodate several minutes worth of high-quality streaming video. These clients also have sufficient I/O and disk bandwidth to simultaneously listen to multiple transmission channels. For example, the ubiquitous Ultra ATA IDE disk interfaces offer about 33 Mbps. Newer PCs can support transfer rates of 40–100 Mbps with Ultra SCSI or Fiber Channel I/O interfaces.

Fig. 1 depicts an example periodic broadcast system. The reception of different video frames by the client can be achieved in several ways. In one approach, the server transmits video frames on various multicast channels, with clients joining and leaving the groups to receive the appropriate frames. Alternatively, the client can listen to all server transmissions of the video, and use a local filter to decide which frames to keep. This model is particularly appropriate for clients on a shared media, such as an Ethernet or a cable access network. In the general case, when the clients are not on a shared media, proxies inside the network can filter the transmission to avoid sending unnecessary frames to the downstream clients. In this paper, we assume that join and leave latencies are small, or that fluid transmission is performed by a proxy that transmits frames to clients on a shared media. We then focus on the behavior of the tradeoff space involving the server and network transmission bandwidth requirements, client buffer, client reception bandwidth requirements, and client playback startup delay.

	Definition
N	Length of video (in frames)
f_j	size of frame j
r_j	transmission rate for frame j (in bits per frame time)
$\mathbf{r}$	vector of transmission rates $(r_1, r_2, \ldots, r_N)$ for the video
e_j	earliest reception start time for frame j, relative to client arrival time
d	playback startup delay
B	size of client buffer
R	maximum instantaneous client reception bandwidth
T	server network transmission bandwidth

TABLE I
Fluid Model: THIS TABLE SUMMARIZES THE KEY PARAMETERS IN THE MODEL

A. System Model

In this section, we provide a formal model of the system, and introduce notations and key concepts used in the rest of the paper. Without loss of generality, we consider a discrete-time system at the granularity of a frame time (e.g., 33 msec. for a 30-frame/second video). We focus on a single N-frame video.

The fluid server continuously transmits frame j at rate r_j. The corresponding *server transmission schedule* determines the set of rates $\{r_j\}_{i \le j \le N}$. The server (and network) transmission bandwidth overhead is given by $T = \sum_{j=1}^{N} r_j$. In the *fluid* model, we assume that the transmission of a single frame can be spread out like a fluid across multiple time units.

The client receives the video according to a *client reception schedule* which specifies the time interval over which the client retrieves each frame. We define the *playback startup delay* d for a client to be the time interval between when it arrives, and the time it starts playback of the video. In the fluid scheme, the client can start receiving the video as soon as it arrives, but may have to wait for some time to build up a sufficient playback buffer to ensure lossless and starvation-free playback. Without loss of generality, we assume that a client arrives at some time a. Unless otherwise stated, all other times in the context of this client are defined relative to this arrival time. Using this convention, the client starts to playback the video at time d and plays frame j at (relative) time $d + j - 1$. A client can receive portions of the frame in any order. In order to guarantee starvation-free playback, frame j has to be completely received client by time $d + j - 1$. Each frame is stored in the client's workahead buffer of size B. To guarantee lossless playback, frame j should not be received too early, if the buffer is small.

In the frame-based fluid periodic broadcast, a valid reception schedule for any client can be defined by the set of tuples $\{(e_j, r_j)\}_{1 \le j \le N}$, where e_j (the earliest reception start time) is also defined with respect to the client's arrival time. The tuple (e_j, r_j) specifies that the client should begin receiving frame j at rate r_j, starting e_j times units after the client's arrival time. The client continues receiving the frame at this rate until it has received the full frame f_j/r_j time units later. For a particular server transmission scheme, the set of tuples is identical for all clients with identical resource constraints (although different clients may receive different portions of the same

frame in a different order). This permits the offline computation of a *single* reception schedule that can be used by different clients (with identical bandwidth and buffer resources) arriving at different times for the same video. Note that this is possible because the server continuously transmits each frame at a fixed rate. In the paper, we use the phrase *broadcast scheme* to denote a particular fluid server transmission schedule and the corresponding client reception schedule.

The buffer occupancy profile of the client reception schedule is specified by the *buffer occupancy* vector $\mathbf{B} = (B(1), B(2), \ldots, B(N+d))$ where $B(k)$ is the instantaneous buffer occupancy k time units after client i arrives. The reception bandwidth usage profile is specified by the *rate usage* vector $\mathbf{R} = (R(1), R(2), \ldots, R(N+d))$ where $R(k)$ is the instantaneous client reception bandwidth usage k time units after client i arrives. From these we can compute the maximum client buffer usage $B_{max} = \max_i B(i)$, and the reception bandwidth requirement $R_{max} = \max_i R(i)$.

III. Unconstrained Client Resources

We first consider the scenario where both the client buffer size and reception bandwidth is sufficiently large that they do not impose any constraints on the server transmission or client reception schedules. We are interested in the following problem:

For the unconstrained client resource case, construct a set of transmission rates $\{r_j\}_{1 \le j \le N}$ for the video, that minimizes the server and network transmission bandwidth T, while guaranteeing that the playback startup delay does not exceed d for any client.

We next outline and evaluate a candidate broadcast scheme that achieve the above goal.

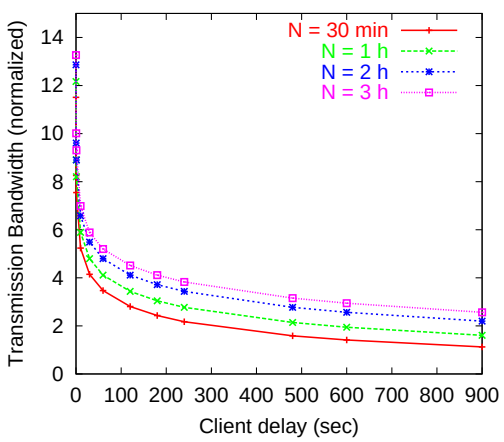
A. Algorithm UBUR

A client plays back frame j, $d + j - 1$ time units after its arrival. Since the client can arrive at any time, frame j must be transmitted in its entirety at least once every $d + j - 1$ frame time. In the continuous fluid transmission scheme, it follows that $r_j = f_j/(d + j - 1)$ is the minimum transmission rate for frame j for guaranteeing a maximum client startup delay d.

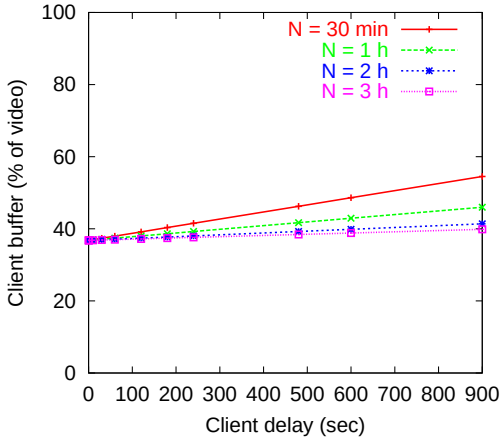
- **Server transmission schedule:**
  Continuously transmit frame j at rate $r_j = f_j/(d + j - 1)$.
- **Client reception schedule:**
  A valid client reception schedule that ensures lossless, starvation-free playback with a delay of d starts to receive frame j at its arrival time (i.e., $e_j = 0$), and continues receiving the frame at its transmitted rate r_j until (relative) time $d + j - 1$, the scheduled playback time of frame j. The server transmission rate r_j is sufficient for the client to receive frame j in its entirety by time $d + j - 1$.

The corresponding server (and network) transmission bandwidth is given by

$$T = \sum_{j=1}^{N} r_j = \sum_{j=1}^{N} f_j/(d + j - 1) \tag{1}$$



(a) T vs d



(b) $B_{max}(\%)$ vs d

Fig. 2. Resource tradeoffs for UBUR scheme for a CBR 6 Mbps video. T is normalized by the mean bandwidth (6 Mbps) of the video.

In the case of CBR, where $f_j = f$ for all j, we have a closed form for the server transmission bandwidth

$$T \approx f \ln \frac{N+d-1}{d-1} \tag{2}$$

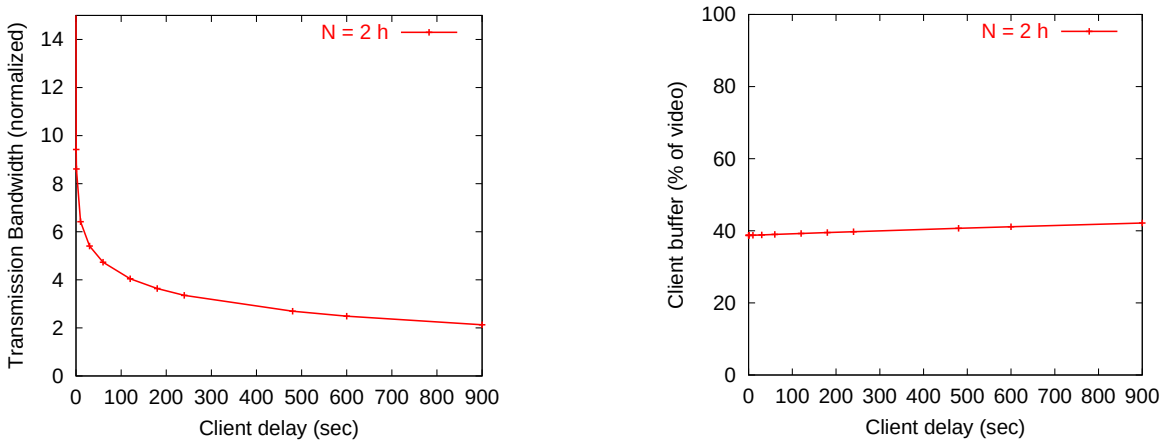
It was shown in [2] and subsequently in [8, 9] that any broadcast policy requires the server bandwidth to be at least $f \ln \frac{N+d}{d}$ to guarantee a maximum client startup delay d. Therefore, this scheme minimizes the server transmission bandwidth.

We refer to the above server transmission and client reception scheme as UBUR (unconstrained buffer unconstrained reception bandwidth) scheme. In the current paper, an important motivation for exploring the optimal UBUR scheme is to use this as a performance baseline for exploring the design space under different client resource constraints, for both VBR and CBR video.

B. Evaluation

In this section, we explore the impact of trading off client startup delay d on the server and network transmission bandwidth as well as client buffering and reception bandwidth requirements.

We first consider a 6 Mbps CBR video. Fig. 2(a) plots the server and network transmission bandwidth requirements (normalized by the mean video bandwidth of 6 Mbps) as a function of the client startup delay d, as given by relation (1). The plots demonstrate that, for a given video length N, T decreases with increasing startup delay d. Across a *range of practical values* of the video length, the most dramatic decrease in transmission bandwidth occurs with d in the range $10 - 200$ sec. For example, when $N = 1$ hour, the normalized value of T reduces from 12 for $d = 33$ msec. to 5.9 for $d = 10$ sec. and to $T = 3$ for $d = 3$ min. The plots indicate that larger delays are of limited utility. Beyond this range of delays upto a few minutes, bandwidth savings are produced only by substantially increasing the startup delay to very large values. The graphs show that most of the savings in server and network transmission bandwidths can be achieved if a client can tolerate a startup delay of a few tens of seconds to a few minutes. In addition, for such delays, the value of T is 3 to 5 times the mean video bandwidth, depending on the video length and the startup delay. This suggests that a startup delay of a few seconds to $2 - 3$ min. is a good operating range from the view point of server and network resource requirements.



(a) T vs d (b) $b(\%)$ vs d

Fig. 3. Resource tradeoffs for UBUR scheme for VBR MPEG-1 *Star Wars*. T is normalized by the mean bandwidth (0.5 Mbps) of the video.

It should be noted that we are considering normalized values of T, and all the observations made in this section hold regardless of the bandwidth requirements of the particular video.

Note that, under the UBUR client reception scheme, the client concurrently receives data from all frames for the first d time units. Therefore, the worst case client reception bandwidth is identical to the server transmission bandwidth. Hence, from the point of view of the client, $20 - 100$ sec. startup delay is also a reasonable operating point.

Fig. 2(b) plots the maximum client buffer occupancy B_{max} for the UBUR scheme. For $d = 33$ msec., the value is about 37% of the entire video, across the range of video lengths considered. For a given video length N, the peak client buffering requirement slowly increases with increasing delay. Note that for startup delays of a few seconds to a few minutes, this additional buffering is marginal. For example, for $N = 1$ hour, $d = 3$ min., the additional buffering is less than one percent of the video. The above trends suggest that the performance benefits of having a startup delay of few seconds to a couple of minutes far outweigh the marginal additional buffer requirement at the client.

Although there are clear common trends, the graphs also clearly underline the impact of the video length on the resource usage. For a given startup delay, T is larger for longer N. In the operation range we identified, T for $N = 3$ hours ranges from 33% higher for $d = 10$ sec. to 69% higher for $d = 3$ min., as compared to T for $N = 30$ min. This serves to caution against generalizing results obtained for a particular video length to different lengths.

Similar trends to the above are noticeable for VBR streams, as shown by the corresponding plots (Fig. 3) for a 2 hour long MPEG-1 *Star Wars* trace with mean (peak) rate of 0.5 Mbps (5.6 Mbps). T is normalized by the mean video bandwidth (0.5 Mbps).

Finally, an interesting point to note here is that the UBUR scheme also smoothes out the bursty transmission and reception bandwidth requirements of VBR video. Variable-bit-rate (VBR) compressed digital video traffic typically exhibits significant burstiness at multiple time scales, owing to the encoding schemes and the content variation between and within video scenes [10–14]. For example, for the *Star Wars* trace, the ratio of the peak to mean video bandwidth is 11 times, making it expensive to provision network resources to support or receive such

transmissions. Transmitting the *Star Wars* video *as is* would require a peak client reception bandwidth equal to its peak rate. In the context of traditional unicast streaming transmission to a client, a technique called *workahead smoothing* [15–17] can reduce the peak and variability of the transmission bandwidths through workahead transmission of frames into the client playback buffer, in advance of their playback times. By transmitting each frame at the lowest possible rate, and requiring clients to receive a frame over a period of time before its scheduled playback instant, the UBUR scheme effectively performs a type of workahead bandwidth smoothing. For *Star Wars*, using UBUR, the peak client reception bandwidth (and server and network transmission bandwidth) reduces by a factor of two to 4.7 for a 1 min. startup delay from the unsmoothed peak of 11. Finally note that, the transmission schedules in our frame-based fluid transmission model are completely smooth, i.e., the server and network transmission bandwidth T is the same across all time, since each frame is continuously transmitted at a constant bit rate.

IV. Homogeneous Client Constraints

In a practical setting, clients may have constraints on either the buffer space, the reception bandwidth, or both. For example, the client network access bandwidth restricts the maximum amount of data that the client can instantaneously receive. We first consider a homogeneous client population, where all the requesting clients for a particular video possess identical resource constraints. In Section V we will consider the heterogeneous client scenario.

A. Constrained client buffer

We assume that the client is constrained to receive each frame j at its transmitted rate r_j continuously from the time it begins receiving the frame until the entire frame has been received. We are interested in the following problem:

Given a finite client buffer size B, construct a feasible set of transmission rates $\{r_j\}_{1\le j\le N}$ that minimizes $T=\sum_{j=1}^{N} r_j$, while guaranteeing a playback startup delay no greater than d for any client.

We present below a greedy algorithm CBUR (constrained buffer unconstrained client reception bandwidth) to solve the problem. The algorithm (Fig. 4) goes through the frames of the video, in order of increasing frame number. Given the aggregate buffer occupancy as a function of time for frames before frame j and the client buffer constraint B, Step 1 allocates the smallest *feasible* reception rate to each frame j, such that the client can receive the entire frame by its playback time, without overflowing the client buffer. The output of this step is a rate r_j and time e_j when the client can start receiving frame j. Note that a *feasible* rate can always be found, as all frames before j will have been played back and, therefore, removed from the buffer by time $(j-1)+d-1$. The rate assignment $r_j=f_j$ is always feasible. Step 2 updates the client buffer occupancy and reception bandwidth usage profiles to reflect the reception of frame j in time interval $(e_j, j+d-1)$ (times are relative to the client's arrival time) at rate r_j.

A straightforward implementation for Step 1 would check, for each u $(0\le u<j+d-1)$ in increasing order, whether selecting $e_j=u$ (the corresponding rate allocation is $r_j=\frac{f_j}{j+d-1-u}$) is *feasible*, i.e., does not result

CBUR (B, d)
Step 0. Initialize the buffer and bandwidth vector.
for $k = 1, \ldots, N + d$
  $B(k) = 0$
  $R(k) = 0$
for $j = 1, \ldots, N$ (determine rate r_j for frame j)
  Step 1. Find $e_j = \min\{k | k \leq j + d - 1\}$ such that receiving frame j
    at a *continuous, fixed* rate $r_j = \frac{f_j}{j+d-1-e_j}$ in time interval $(e_j, j + d - 1)$
    does not overflow the client buffer as follows
      $k = j + d - 1$ and $r_j = f_j/(j + d - 1)$ and $e_j = 0$
      while ($e_j \leq k$)
        if ($f_j - r_j(j + d - 1 - k) \leq B - B(k)$)
          $k = k - 1$
        else
          $r_j = (f_j - B + B(k))/(j + d - 1 - k)$
          $e_j = j + d - 1 - f_j/r_j$
      if ($e_j > k$)
        $e_j = k + 1$
        $r_j = f_j/(j + d - 1 - e_j)$
  Step 2. Update buffer occupancy $\mathbf{B}$ and reception bandwidth $\mathbf{R}$ usage profiles
  for time interval $(e_j, j + d - 1)$ to reflect allocation of r_j
    $\forall k, \quad e_j \leq k \leq j + d - 1, \quad B(k) = B(k) + (k - e_j)r_j$
    $\forall k, \quad e_j \leq k \leq j + d - 1, \quad R(k) = R(k) + r_j$

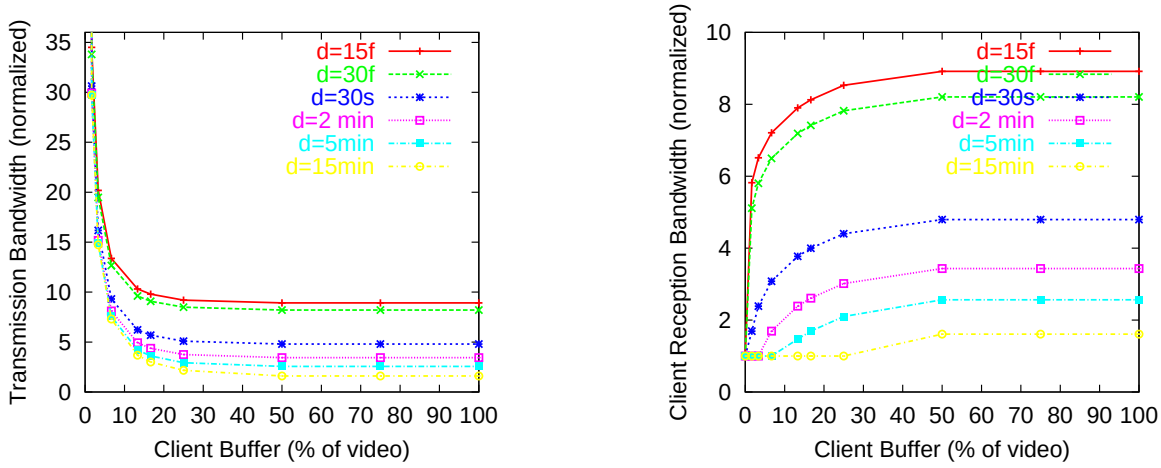
Fig. 4. **CBUR Algorithm:** Pseudocode for determining the server transmission and client reception schedules given that the client buffer size is B and the startup delay is d.

in buffer overflow at any time in $(u, j + d - 1)$. This takes $O(N^2)$ time ($O(N)$ for each u). However, we can compute Step 1 in linear time $O(N)$ (linear in the number of time units between e_j and $j + d - 1$). We make use of the following property

Claim: *Consider $e_j = u, u \in \{y, z\}$, $(0 \leq y < z < j + d - 1)$. Let $\delta_u(k)$ be the buffer occupancy at time k $(\min(y, z) \leq k \leq j + d - 1)$ for frame j. Let $r_u = \frac{f_j}{j+d-1-u}$ be the corresponding minimum constant transmission (and reception) rate for frame j. Then*

1. *$\delta_u(k) = (k - u)r_u$ is a linearly increasing function in time interval $(u, j + d - 1)$.*
2. *$r_y < r_z$*
3. *$\delta_y(k) \geq \delta_z(k)$*

The $O(N)$ algorithm starts with $e_j = 0$ and $r_j = \frac{f_j}{j+d-1}$. This r_j would be the lowest *feasible rate* for this frame for the unconstrained client buffer case. Moving backwards from time $j + d - 1$, the algorithm checks at each time k, whether using the chosen rate would violate the client buffer constraint at that time. If so, the rate r_j is increased just enough to avoid the overflow at time k. The new earliest reception start time $e_j = j + d - 1 - \frac{f_j}{r_j}$. The process is then repeated from time $k - 1$ backwards. Note that as r_j is increased, e_j is shifted further to the



(a) T vs B (b) R_{max} vs B

Fig. 5. **Resource tradeoffs for CBUR scheme**: T and R_{max} as a function of the client buffer size B, for a range of startup delays d. T and R_{max} are normalized by the average bandwidth of the video (length $N = 1$ hour). B is expressed as a percentage of the total size of the video.

right. If the new e_j is shifted to the right of position k, then there is no need to satisfy the buffer constraint at position k. Therefore, position $k+1$ should be the starting time of receiving frame j.

Step 2 takes at most $O(N)$ time, and therefore the computation complexity of CBUR is $O(N^2)$.

The output of the above CBUR algorithm is

- A set of reception rates $\{r_j\}_{1\le j\le N}$ for the video. In our scheme, r_j is also the transmission rate of frame j, i.e., server will continuously transmits each frame j at rate r_j.
- The set of tuples $\{(e_j, r_j)\}_{1\le j\le N}$ defines a valid reception schedule for any client. Any client i receiving frame j at rate r_j starting at time e_j up to time $j+d-1$ would not overflow or underflow its buffer.
- The reception bandwidth profile for the time interval $(0, N+d-1)$. From this, we can compute the maximum client reception bandwidth requirement $R_{max} = \max_i R(i)$, under this scheme.

A.1 Optimality Property

The greedy rate allocation algorithm CBUR minimizes the total transmission bandwidth given any client buffer space.

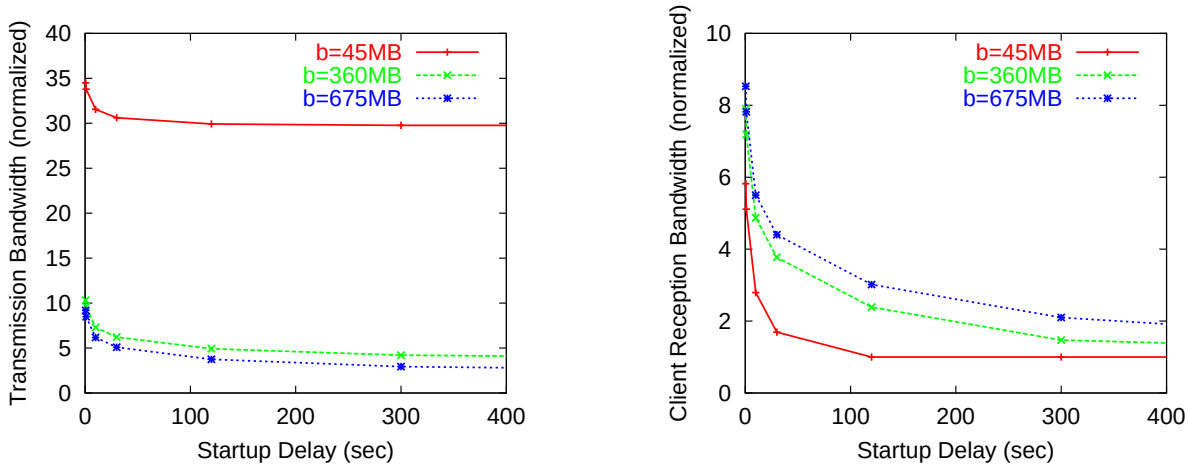
Theorem 1: Given a client buffer size B, the greedy rate allocation algorithm CBUR is optimal in the sense that no other algorithm can further reduce the total server (and network) transmission bandwidth.

The proof of the theorem is presented in Appendix A.

A.2 Performance

We use the optimal CBUR algorithm to study the multidimensional resource tradeoffs in the homogeneous buffer constrained case. Note that the algorithm applies equally well to CBR and VBR video. We focus on a 1 hour long 6 Mbps CBR video for this evaluation.

Fig. 5(a)-(b) respectively plot T and R_{max} as a function of the client buffer size B, for a range of startup delays d, for the CBUR scheme. Note that the client buffer requires sufficient space to hold at least one frame in the video. For our example, $B \ge 25$ KB. The extreme left points in Fig. 5 correspond to $B = 25$ KB.



(a) T vs d (b) R_{max} vs d

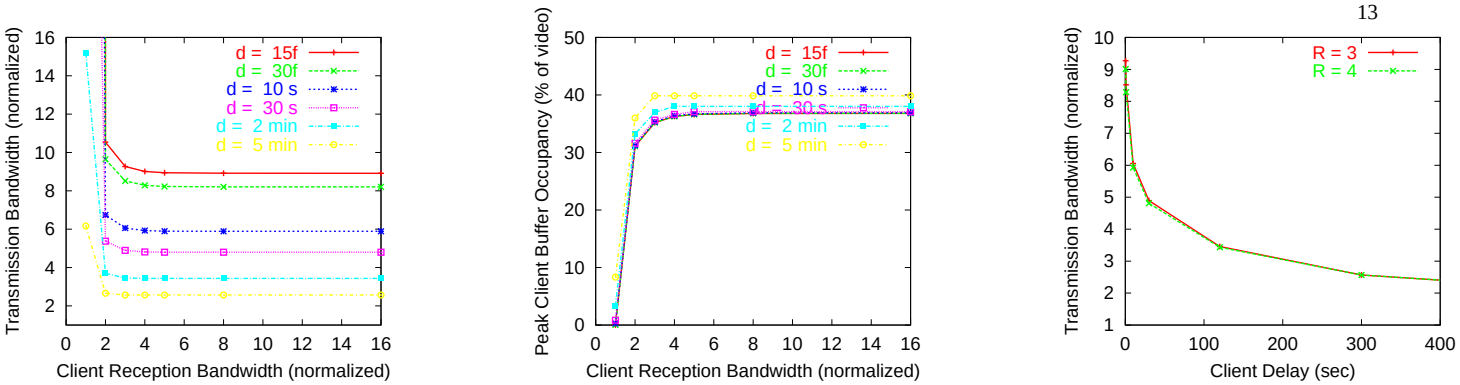
Fig. 6. **Resource tradeoffs for CBUR scheme**: T and R_{max} as a function of the client startup delay d (sec), for a range of client buffer sizes B. T and R_{max} are normalized by the average bandwidth of the video (length $N = 1$ hour).

When $B = 25$ KB, the minimum server and network transmission bandwidth T under the CBUR scheme is N (number of frames in the video) times the bandwidth of the video stream. The corresponding peak client reception bandwidth R_{max} is equal to the video bandwidth. The reason for this is that, for such a small client buffer size, the client has sufficient space to hold one frame at a time, and therefore it can only receive a frame, after the previous frame has been played out, leading to the low R_{max}. As such, under the fluid scheme, each frame has to be completely transmitted at least once every frame time, resulting in the very high server transmission bandwidth.

Fig 5(a) shows that the server transmission bandwidth T appears to be a decreasing convex function of the client buffer size. We observe that there is little further decrease in T once B approaches the worst case client buffer occupancy (for the same startup delay) for the optimal UBUR scheme, and that T has the same value as under the UBUR scheme in this flat region. The plots indicate that most of the savings in server bandwidth can be accrued with relatively modest amounts of client buffer. For a 30 sec. client delay, T is reduced from 108000 times the video bandwidth to 6.2 times the video bandwidth by providing the client with a buffer that can store 13.3% of the video. This is within 30% of the transmission bandwidth T that can be achieved under UBUR when the client buffer imposes no constraint. The trends are similar across the range of client delays we examine.

Fig 5(b) shows that the peak client reception bandwidth R_{max} appears to be an increasing concave function of client buffer space, and that it becomes essentially constant once the the client buffer size approaches the worst case client buffer occupancy (for the same startup delay) for the optimal UBUR scheme. In this region, the curves flatten out, and R_{max} has the same value as under the UBUR scheme with unconstrained client buffer.

Across the range of client buffer sizes, as can expected, the transmission and client reception bandwidths decrease as the startup delay increases. Fig 6(a)-(b) show that the steepest decline in server transmission bandwidth as well as in R_{max} occurs to the left of the plots, when the startup delay is in the range of a few seconds to $100 - 200$ sec. This again suggests that several tens of seconds startup delay is a reasonable operating region for the server, network and client.



(a) T vs R (b) B_{max} vs R (c) T vs d

Fig. 7. **Resource tradeoffs for UBCR scheme**: (a) and (b) plot T and B_{max} as a function of the client reception bandwidth constraint R, for a range of startup delays d. (c) plots T as a function of delay d for $R = 3$ and $R = 4$. T and R are normalized by the mean bandwidth (6 Mbps) of the video (length $N = 1$ hour). B_{max} is expressed as a percentage of the total size of the video.

B. Constrained client reception bandwidth

We next consider the dual problem to the buffer constrained scheduling problem: the *rate constrained* scheduling problem, where the client reception bandwidth is constrained by a given rate R. We are interested in the following problem:

Given a client reception bandwidth R constraint for any client, construct a feasible set of transmission rates $\{r_j\}_{1 \le j \le N}$ that minimizes $T = \sum_{j=1}^{N} r_j$, while guaranteeing that the playback startup delay does not exceed d for any client.

To study the resource tradeoffs in the rate-constrained case, we develop a greedy heuristic solution to the above problem. The technique is similar to the CBUR algorithm (Fig. 4). Our UBCR (unconstrained buffer constrained rate) algorithm also sequences through the frames of the video, in order of increasing frame number. The main difference is in Step 1. Given the aggregate instantaneous reception bandwidth usage as a function of time for frames before j and the instantaneous reception bandwidth constraint R, the UBCR algorithm allocates the smallest *feasible* rate to each frame j, such that the client can still receive the entire frame by its playback time, without violating the rate constraint. Note that this is a greedy scheme which allows us to explore the resource space when the client bandwidth is constrained, but may not result minimize T. The UBCR algorithm can be considered to be a performance upper bound on the achievable T when client bandwidths are constrained.

B.1 Performance

Fig 7(a)-(b) plot T and B_{max} under UBCR as functions of the client reception bandwidth constraint R, for a range of startup delays d. These plots indicate that the server transmission bandwidth can be extremely high if R is equal to the mean video bandwidth, particularly for small startup delays. For example, for $d = 1$ sec., T is 2076 times the video bandwidth when $R = 1$. Dramatic reductions in T are observed as R increases to 2. For the same $d = 1$ sec., T drops to 9.65 times the video bandwidth, a reduction of more than a factor of 215. We observe that T is a decreasing convex function of R and that the minimum value of T is nearly achieved once

R approaches the maximum client reception bandwidth required by the UBUR algorithm (Section III) for the same startup delay. Across the range of delays, most of the reduction in T can be obtained using $2-3$ times the video bandwidth. For example, for $d = 30$ sec, and $R = 3$, the server transmission bandwidth is $T = 4.89$ which is within 2% of the optimal T under unconstrained client buffer and reception bandwidth conditions. This is important from a practical viewpoint as it suggests that clients do not require a lot of additional reception bandwidth capacity for the periodic broadcast to be server and network bandwidth efficient. However, under the UBCR algorithm, this transmission bandwidth efficiency comes at the expense of a relatively large client buffer requirement, as shown in Fig 7(b). The curves show that the peak client buffer occupancy increases steeply from $R = 1$ to $R = 2$, and then gradually flattens out. In the flat region the peak buffer occupancy has the same value as the peak occupancy under the UBUR scheme. Also, once R exceeds 2, there is little difference in the client buffer size requirements. Across the range of values of R, as can be expected, the transmission bandwidth decreases as the startup delay increases. Fig 6(c) considers the impact of startup delay on T, for relatively small values of R (3 and 4). The plots again suggest that a few tens of seconds startup delay is most useful for reducing server and network bandwidth requirements.

V. Heterogeneous Client Constraints

Existing work in periodic broadcasting has focused almost exclusively on developing various server transmission algorithms for the homogeneous client case. In practice however, different clients accessing the same video may have very different buffering and reception bandwidth capabilities. An open question here is how to handle such client heterogeneity while maximizing client satisfaction. In this section, we consider the problem of using a *single* server transmission schedule to satisfy clients with heterogeneous resource constraints. We will first consider the case where clients have heterogeneous bandwidth constraints.

A. Constrained client bandwidth

We first consider the case where some client's reception bandwidth capacity is R. Due to the rate constraint, the buffer at the client must be *sufficiently large* to ensure continuous video playback at the client. Furthermore, it may be necessary for the client to start receiving the video *sufficiently early*. Hence *the rate constraint imposes both a minimum buffer requirement and startup delay at the client*. In this context, a client reception schedule is *feasible* if the reception rate of the schedule never exceeds the rate constraint at any time and the amount of data needed for client playback is always satisfied at any time.

We are interested in the following problem:

Given a set of video frame transmission rates $\{r_j\}$ *for* $1 \leq j \leq N$

1. *what is the minimum client start-up delay so that a feasible reception schedule exists?*
2. *among all feasible schedules, what is the smallest client buffer necessary for feasible reception, and what is a corresponding feasible schedule ?*

In order to conserve the client buffer, we propose a heuristic lazy algorithm H-UBCR (Heterogeneous unconstrained buffer constrained bandwidth) to construct a *feasible* reception schedule which receives data as late as possible, while obeying rate constraint R, and the individual transmission rate constraints of each frame, but

H-UBCR $(R, \{r_j\}_{1 \le j \le N})$
Step 1. for $j = N, \ldots, 1$ (determine the starting reception time for frame j)
Step 1(a). Find $o_j = \max\{k | k \le j + d - 1 - \frac{f_j}{r_j}\}$ such that receiving frame j
at a *continuous, fixed* rate r_j in time interval $(o_j, o_j + \frac{f_j}{r_j})$
does not violate the client reception bandwidth constraint R as follows
$ft = j + d - 1$ (ft is the time by which frame j has been completely received)
and $o_j = ft - \frac{f_j}{r_j}$ and $k = ft$
while $(o_j < k)$
if $(R(k) + r_j > R)$
$ft = k - 1$ and $o_j = ft - \frac{f_j}{r_j}$
$k = k - 1$
Step 1(b). Update buffer occupancy (**B**) and reception bandwidth usage (**R**) profiles
for time interval (o_j, ft) to reflect allocation of r_j
$\forall k, \quad o_j \le k \le ft, \quad B(k) = B(k) + (k - o_j) r_j$ and $\forall k, ft < k \le j + d - 1, \quad B(k) = B(ft)$
$\forall k, \quad o_j \le k \le ft, \quad R(k) = R(k) + r_j$
Step 2. Compute d, b and starting reception time of each frame relative to the client arrival time
$d = -\min_{1 \le k \le N} o_k$
$b = \max_{-d \le k \le N} B(k)$
$\forall j, \quad e_j = o_j + d$

Fig. 8. **H-UBCR Algorithm:** Pseudocode for computing the reception schedule for the client whose reception bandwidth is R).

still ensures that each frame is received by its playback time. We refer to this as the *lazy schedule*. For ease of exposition, we assume that client playback starts at time 0, and that the client starts the reception at time $-d$ $(d \ge 0)$. The schedule is computed from the last frame backwards. For frame j, we compute the latest time o_j (defined relative to the client's playback start time 0) such that the client can receive frame j at a rate equal to r_j, the transmission rate of that frame, starting at time o_j until the entire frame has been received. In addition, the client has to receive the entire frame before time $j + d - 1$, the playback time of frame j. We compute the rate for each frame starting with frame N backwards (Fig. 8). The output of the algorithm is a set of time offsets $\{e_j = o_j + d\}_{1 \le j \le N}$ specifying the earliest reception start time for each frame, relative to the client's arrival time. The set of tuples $\{(e_j, r_j)\}_{1 \le j \le N}$ defines a valid reception schedule for any client with rate constraint R. The client should begin receiving frame j at a constant rate r_j, e_j time units after its arrival. Finally, we note that the complexity of the above algorithm is $O(N^2)$.

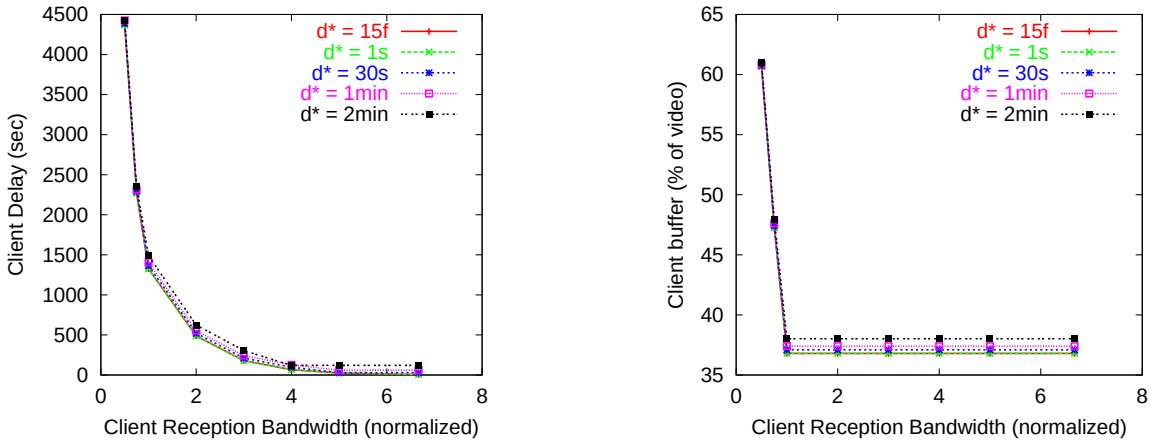
Now define

$$d(R, \mathbf{r}) = \{-\min_{1 \le j \le N} o_j\} \tag{3}$$

$$\text{and} \qquad b(R, \mathbf{r}) = \max\{B(k) | -d \le k \le N\} \tag{4}$$

It follows from the above definitions that b[1] is the minimum buffer requirement and d is the minimum start-up delay with respect to the given rate constraint R and transmission vector $\mathbf{r}$, for the above schedule.

[1]We will denote b for $b(R, \mathbf{r})$ and d for $d(R, \mathbf{r})$ whenever there is no danger of confusion.



(a) d vs R (b) b vs R

Fig. 9. Minimum client startup delay d and buffer requirement b as a function of client reception bandwidth R. The server transmission schedule assumes unconstrained client buffer and bandwidth resources, and a range of target client startup delays d^*. R is normalized by the playback rate of the video. The length of video is 1 hour. b is expressed as a percentage of the total size of the video.

We can use H-UBCR to answer both questions posed at the beginning of Section V-A. In addition, the above algorithm has an important implication for the following heterogeneous buffer constrained client problem.

Given a set of video frame transmission rates $\{r_j\}_{1 \le j \le N}$ *and a client buffer size* B

1. *Is the client buffer* feasible*, i.e., does there exist a feasible client reception schedule ?*
2. *If* B *is feasible, what is the smallest playback startup delay necessary for feasible reception?*

For this problem, we can compute the lazy schedule assuming the reception rate constraint $R = \infty$, and thence b and d. Then, for the lazy schedule, the client buffer allocation B is feasible only if $B \geq b(R, \mathbf{r})$, and d is the required minimum startup delay for any feasible buffer allocation.

A.1 Performance

To explore the impact of heterogenous client resources on performance, we consider the following scenario. The server computes its transmission schedule assuming unconstrained client buffer and bandwidth resources (using the UBUR scheme from Section III), and aims to provide a certain startup delay guarantee to such clients. We refer to this target delay as d^* in the following discussion. When clients with different bandwidth constraints arrive, the client reception schedule is computed using the H-UBCR algorithm outlined above. Fig. 9 plots the corresponding d and b as a function of the client bandwidth constraint R for the particular server transmission schedule.

Fig. 9(a) shows that the client playback startup delay d can be very high if R is small. As R increases, d decreases, initially very rapidly, then more gradually. The curve finally flattens out at d^*, the delay that would be experienced by clients with unconstrained reception bandwidth. Dramatic reductions in startup delay d are obtained with $R = 3-5$ times the video bandwidth, across a range of values of d^*. For example, for $d^* = 1$ min., the client startup delay is a very high $d = 23$ min. for $R = 1$. This reduces to much more acceptable $d = 2$ min. for $R = 4$, a reduction of almost a factor of 12. Also, the graphs indicate that for a given R, a larger d^* results in a larger value of client delay d.

Fig. 9(b) shows that b can be very high if $R < 1$. Once $R = 1$, the worst case buffer usage reduces dramatically to the corresponding value for UBUR (Section III), and remains at this value for further increases in R. For $d^* = 30$ sec., $b = 61\%$ of the video size when $R = 0.5$ times the mean video bandwidth. This reduces to 37% of the video size for $R \geq 1$.

VI. Parsimonious Transmission Schemes

Periodic broadcast is a server-push technique in which the server multicasts frames even if no client needs some frame for continuous playback. Such schemes seems particularly well suited to serving hot videos with high client arrival rates. For such situations, an attractive feature of our frame-based periodic broadcast is that the server can determine the server transmission schedule as well as a single reception schedule for all clients, offline and independent of the client arrival process. The server does not need to track individual clients, making the online interaction of the server and client simpler. However, such schemes can be inefficient if client demand is not high. There may be opportunities to reduce bandwidth usage if the server factors the actual data requirements of arriving clients into the actual transmission. We refer to such schemes as *parsimonious* broadcast schemes. An interesting related work is [3] which dynamically allocates unused transmission channel capacity from one skyscraper video broadcast [1] to transmit initial segments of another broadcast. We, on the other hand, are interested in exploring the potential of any parsimonious scheme in reducing the server and network transmission bandwidth requirements, and the role that playback startup delay plays in this. We consider the unconstrained client resources case, and develop and evaluate performance lower bound on parsimonious transmission schemes, with this goal.

A. P-UBUR Parsimonious Transmission Algorithm

The UBUR periodic broadcast scheme(Section III) guarantees that as long as frame j is transmitted once every $d + j - 1$ time units, an arriving client is able to receive a copy of each frame and to experience starvation-free playback with a startup delay not exceeding d time units. However, by operating oblivious to the request arrival process, the scheme wastes server and network transmission bandwidth as some of the transmissions may not be required by any client. We now outline a parsimonious scheme for the UBUR case, which can reduce the average server and network transmission bandwidth further by transmitting a frame as late as possible, and only if it is required by a client:

Algorithm P-UBUR

- **Server transmission schedule:**
  When a client arrives at time a, for any $1 \leq j \leq N$, the server schedules a new complete transmission of frame j for time $a + j + d - 1$ only if a copy is not being multicast in time interval $(a, a + j + d - 1)$.
- **Client reception schedule:**
  When a client arrives at time a, for any $1 \leq j \leq N$, the client receives the last transmission of frame j in time interval $(a, a + j + d - 1)$ and saves the frame in its buffer. The client starts the playback of frame j at time $a + j + d - 1$.

A.1 Average Transmission Bandwidth Requirement

To evaluate the parsimonious P-UBUR scheme, we derive a closed-form expression for the transmission bandwidth requirements as a function of the video frame sizes, video length (N), the target client startup delay d, and the request arrival distribution. The performance metric we consider is $\overline{T}$, the average amount of server and network transmission bandwidth using the P-UBUR policy. Since the transmission schedule of a frame is independent of other frames, we can consider each frame of the video separately. Let $\overline{T_j}$ denote the average amount of server and network transmission bandwidth used for transmitting frame j of the video. We have the total amount of server and network transmission bandwidth for the video:

$$\overline{T} = \sum_{1 \le j \le N} \overline{T_j}$$

We now derive $\overline{T_j}$ for frame j. Let $\{t_{i,j}\}_{i=0}^{\infty}$ denote the times at which the server schedules a transmission of frame j. These are renewal points in the sense that the server transmission for frame j after time $t_{i,j}$ does not depend on its transmission schedule before time $t_{i,j}$. We consider the process $N_{i,j}$ where $N_{i,j}$ is the number of clients that arrive in the i-th renewal epoch $[t_{i-1,j}, t_{i,j})$. We drop the subscript i since it is a renewal process. We have $E[N_j] = 1 + \lambda(d + j - 1)$ and $\overline{T_j} = \lambda \frac{f_j}{E[N_j]} = \frac{f_j}{d+j-1+1/\lambda}$.

It follows that

$$\overline{T} = \sum_{1 \le j \le N} \frac{f_j}{d + j - 1 + 1/\lambda} \tag{5}$$

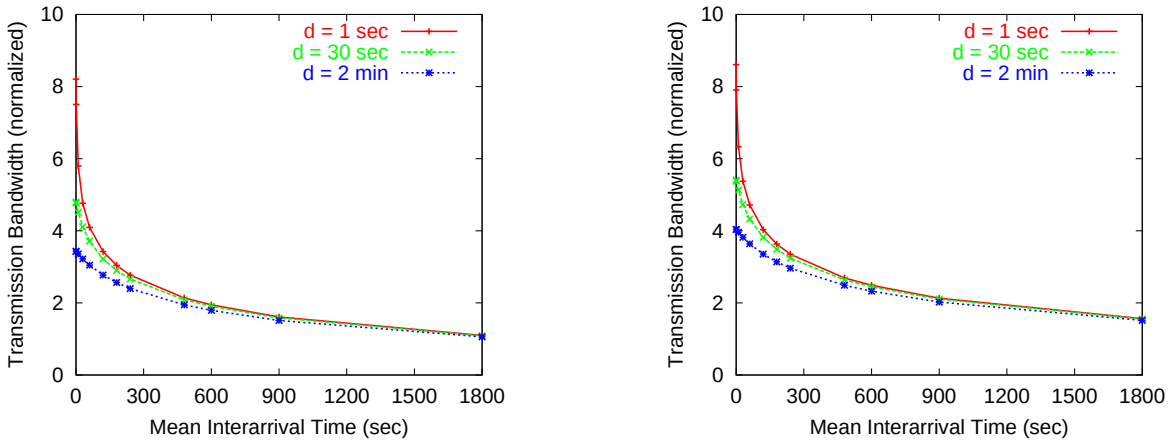
In the case of CBR video, where $f_j = f$ for all j,

$$\overline{T} \approx f \ln \frac{N + d - 1 + 1/\lambda}{d - 1 + 1/\lambda} \tag{6}$$

Note that the probability distribution of client arrivals only influences the expression for $\overline{T}$ through the mean λ. The analysis therefore applies to a wide range of client arrival processes. We note that [18] presents a similar bound on the transmission bandwidth for the particular case of providing immediate service and CBR videos. This work does not consider the impact of playback startup delay, which is a key motivation for the current work.

A.2 Performance Comparison

To investigate the potential bandwidth savings of the parsimonious P-UBUR scheme, we assume that clients arrive according to a Poisson process with rate λ. Fig. 10 plots $\overline{T}$ as a function of the client interarrival time, for the P-UBUR scheme, for a one hour long CBR stream, and two hour long VBR MPEG-1 *Star Wars*, for different values of the client startup delay d. The graphs indicate that there can be significant savings in server and network bandwidth usage, over using UBUR periodic broadcast, particularly if the target client startup delay is small, even for relatively frequent client arrivals. When the mean interarrival time $1/\lambda = 0$, the bandwidth usage under P-UBUR corresponds to the usage for the UBUR periodic broadcast scheme of Section III. For example, for *Star Wars*, corresponding to a 1 sec. startup delay, and interarrival time of 1 min., P-UBUR requires $\overline{T} = 4.7$ which



(a) CBR (b) VBR *Star Wars*

Fig. 10. **Impact of client arrival process on $\overline{T}$:** (a) and (b) plot $\overline{T}$ for the P-UBUR scheme for 1 hour CBR stream, and 2 hour VBR MPEG-1 *Star Wars*, respectively, as a function of the mean client interarrival time ($1/\lambda$).

is nearly a factor of 2 lower than the corresponding value ($\overline{T} = 8.6$) under UBUR periodic broadcast. In this regime, the higher bandwidth requirements for UBUR are most likely due to frequent transmissions of initial frames in the video, some of which are not required by any client. The plots also show that if the client startup delay is large, on the order of a few minutes or more, or if the arrival rate is very high, then the savings under P-UBUR are much more modest. This suggests that the simpler UBUR periodic broadcast scheme (which does not need to track individual clients) may be more appealing in this regime of large startup delays or popular videos.

Finally, both graphs in Fig. 10 show that once $1/\lambda$ exceeds a few minutes, the performances for different startup delays tend to become more similar. To highlight this interplay between startup delay and interarrival time for P-UBUR, Fig. 11 plots T for VBR MPEG-1 *Star Wars* as a function of the delay guarantee d for a range of interarrival times. For a given $1/\lambda$, T is a decreasing convex function of d. The plots indicate that for low interarrival times (high arrival rates), there are significant savings in transmission bandwidth, if d is of the order of several tens of seconds to a few minutes. However, the benefits of increasing d reduces as the interarrival time increases. For $1/\lambda = 30$ sec., T decreases from 5.38 for $d = 1$ sec. to 3.24 for $d = 4$ min. For $1/\lambda = 10$ min., T decreases from 2.48 to 2.19 over the same delay range. This suggests that for parsimonious transmissions, startup delays have limited effectiveness in reducing transmission overheads if the arrival rate is not high.

VII. Related Work

Existing research in periodic broadcast has focused almost exclusively on developing segmentation and transmission schemes for CBR video. However VBR encoding offers some key advantages. For instance, for the same average bandwidth, a variable-bit-rate encoding offers higher quality and more opportunities for statistical multiplexing gain than would be possible for a constant-bit-rate encoding [19, 20]. Recently, [21] considered the problem of using periodic broadcast for transmitting multiple VBR videos over a fixed bandwidth connection from the server. Using nonuniform segmentation schemes developed for CBR video, they study the impact of multiplexing the transmissions of the resultant segments, using smoothing, server buffering and client prefetching. Another recent work [22] proposes a specialized nonuniform segmentation scheme based on the segment-

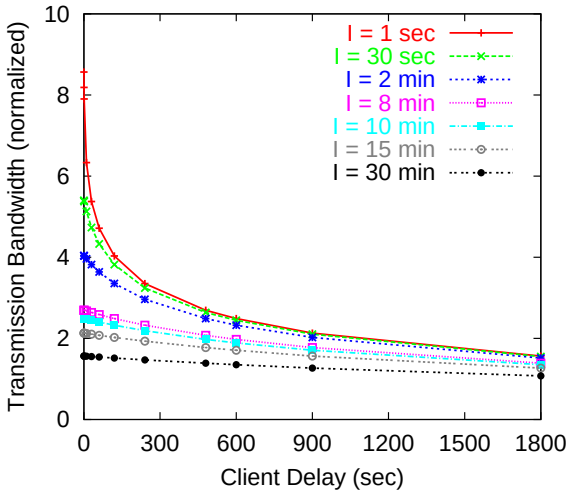


Fig. 11. $\overline{T}$ for the P-UBUR scheme for 2 hour VBR MPEG-1 *Star Wars*, as a function of playback startup delay guarantee d, for a range of values of $I = 1/\lambda$, the mean client interarrival time.

level bandwidth profile of a particular VBR video. The scheme requires any segment beyond the first to be completely received before the previous segment has been completely played out. In contrast, the frame-based uniform segmentation, and the transmission scheduling approach of our fluid periodic broadcast model offers a uniform framework for handling both CBR and VBR video. By utilizing the client playback delay tolerance to spread out the transmission of each frame over multiple time units, our broadcast scheme inherently ends up also effectively smoothing out the bursty transmission bandwidth requirements of VBR video.

Finally, patching or stream tapping [2, 7, 23, 24] is an alternative technique to periodic broadcast. In patching, the server streams the entire video sequentially to for the very first client. Client-side *workahead buffering* is used to allow a new client to receive (part of) its future playback data requirement by listening in to an *existing* ongoing transmission of the same video, with the server transmitting afresh only the remaining required frames. As a result, fewer server and network resources are required to satisfy the clients. Similar to periodic broadcast schemes, patching exploits the client buffer space to store future frames from other video transmissions. Unlike periodic broadcasting, the server transmits video data only on-demand, when new clients arrive, and does not involve any playback startup delay.

VIII. Conclusions

This paper explored several fundamental tradeoffs involving the server and network transmission bandwidths, client buffer, client reception bandwidth, and the playback startup delay for the periodic broadcast of streaming CBR and VBR video. Using a fine-grained *frame-based* fluid transmission model, we explored both constrained and unconstrained client resource (buffer or bandwidth) situations. For each situation we presented a broadcast scheme that minimizes the server and network transmission bandwidth requirements, while guaranteeing a prescribed playback startup delay, and then used that scheme to explore the resource tradeoff space. When client resources are not constrained, we identify a broadcast scheme (UBUR) which minimizes the transmission bandwidth requirements, and developed practical guidelines for determining operating regions of reasonable server, network and client resource requirements, and client startup delays. For the constrained client buffer case, we

presented the CBUR scheme that optimally minimizes the transmission bandwidth requirements. In order to explore the constrained client reception bandwidth case, we developed a heuristic greedy algorithm (UBCR) to minimize the transmission bandwidth requirements.

We considered using a *single* server transmission scheme to serve clients with heterogeneous resource constraints. We developed a heuristic lazy client reception schedule (H-UBCR) to jointly minimize the playback startup delay and client buffer requirements, given any frame-based fluid server transmission schedule, and a particular client bandwidth constraint. Extensive evaluations across all the different scenarios suggest that the design space defined by a playback startup delay of a few tens of seconds, client reception bandwidth equal to $2-4$ times the mean video bandwidth, and client buffer space to accommodate $15-40\%$ of the video, and server and network transmission bandwidths of $3-5$ times the mean video bandwidth, is very attractive feasible operating region from the server, network, as well as client performance viewpoint.

Last, we presented a parsimonious extension (P-UBUR) of the frame-based UBUR broadcasting scheme where the server only transmits data that is required by some client. Evaluations suggest that, for both CBR as well as VBR videos, the parsimonious scheme can yield significant savings in server and network bandwidth usage, particularly for small playback startup delay guarantees. We also observe that for parsimonious transmissions, startup delays have limited effectiveness in reducing transmission overheads if the arrival rate is not high.

We have presented a (mostly) analytical and algorithmic treatment of the problem. The next step is to use the knowledge gained from this exercise for developing practical broadcast schemes with superior performance. Another promising research area is exploring actual implementation issues, including designing efficient protocols for implementing periodic broadcast functionality, by using underlying network support.

REFERENCES

[1] K. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM*, September 1997.

[2] L. Gao, D. Towsley, and J. Kurose, "Efficient schemes for broadcasting popular videos," in *Proc. Inter. Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.

[3] D. Eager and M. Vernon, "Dynamic skyscraper broadcasts for video-on-demand," in *Proc. 4^{th} Inter. Workshop on Multimedia Information Systems*, September 1998.

[4] D. Eager, M. Ferris, and M. Vernon, "Optimized regional caching for on-demand data delivery," in *Proc. Multimedia Computing and Networking (MMCN '99)*, January 1999.

[5] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[6] L. Golubchik, J. Lui, and R. Muntz, "Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers," *ACM Multimedia Systems Journal*, vol. 4, no. 3, 1996.

[7] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," Tech. Rep. 99-22, Department of Computer Science, University of Massachusetts Amherst, 1999.

[8] J.-F. Paris, S. Carter, and D. Long, "A low bandwidth broadcasting protocol for video on demand," in *Proc. 7^{th} Inter. Conference on Computer Communications and Networks*, October 1998.

[9] Y. Birk and R. Mundri, "Tailored transmissions for efficient Near-Video-On-Demand service," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, 1999.

[10] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM*, September 1994.

[11] A. R. Reibman and A. W. Berger, "Traffic descriptors for VBR video teleconferencing over ATM networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 329–339, June 1995.

[12] S. Gringeri, K. Shuaib, R. Egorov, A. Lewis, B. Khasnabish, and B. Basch, "Traffic shaping, bandwidth allocation, and quality assessment for MPEG video distribution over broadband networks," in *IEEE Network Magazine*, pp. 94–107, November/December 1998.

[13] M. Krunz and S. K. Tripathi, "On the characteristics of VBR MPEG streams," in *Proc. ACM SIGMETRICS*, pp. 192–202, June 1997.
[14] M. Krunz and S. K. Tripathi, "Bandwidth allocation strategies for transporting variable-bit-rate video traffic," in *IEEE Communication Magazine*, pp. 40–46, January 1999.
[15] W. Feng, F. Jahanian, and S. Sechrest, "An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video," *Springer-Verlag Multimedia Systems Journal*, vol. 5, pp. 297–309, September 1997.
[16] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. Networking*, vol. 6, pp. 397–410, August 1998.
[17] J. M. McManus and K. W. Ross, "Video-on-demand over ATM: Constant-rate transmission and transport," *IEEE J. Selected Areas in Communications*, vol. 14, pp. 1087–1098, August 1996.
[18] D. Eager, M. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," in *Proc. 5^{th} Inter. Workshop on Multimedia Information Systems*, October 1999.
[19] I. Dalgic and F. A. Tobagi, "Performance evaluation of ATM networks carrying constant and variable bit-rate video traffic," *IEEE J. Selected Areas in Communications*, vol. 15, August 1997.
[20] T. V. Lakshman, A. Ortega, and A. R. Reibman, "Variable bit-rate (VBR) video: Tradeoffs and potentials," *Proceedings of the IEEE*, vol. 86, May 1998.
[21] D. Saparilla, K. W. Ross, and M. Reisslein, "Periodic Broadcasting with VBR-Encoded Video," in *Proc. IEEE INFOCOM*, March 1999.
[22] J.-F. Paris, "A broadcasting protocol for compressed video," in *Proceedings of the EUROMEDIA '99 Conference*, 1999.
[23] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. International Conference on Computer Communications and Networks*, 1997.
[24] K. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM Multimedia*, September 1998.

APPENDIX

I. PROOF FOR THEOREM 1

Proof of Theorem 1: The proof of the theorem is follows from proving the following claim.

Claim: *Given any allocation that satisfies the client buffer constraint, we can convert the allocation to the greedy allocation without increasing the total rate allocation.*

Proof: Suppose frame j is the first frame that is not allocated greedily. We can reduce frame j's rate so that its rate is allocated in smallest possible given the rate allocation for frames before j and client resources constraints.

If such transformation of the allocation still satisfies the client resource constraints counting all frames, we are done in making frame j's allocation greedy.

If not, we increase the rate for later frames as follows. Suppose at the playback position x that client storage constraint is not satisfied. There must be a set F of one or more frames, whose playback times are subsequent to frame$j$'s playback time, and whose reception is scheduled before position x.

Note that if we start to receive frame k at time s_k (all times are relative to the playback starting time) at rate r_k, then at time y, frame k occupies buffer space of $s_k(y)$, where

$$s_k(y) = \begin{cases} 0, & \text{if} \quad y < s_k \quad \text{or} \quad y > k, \\ (y - s_k)r_k = f_k - (k + d - 1 - y)r_k & \text{otherwise} \end{cases} \tag{7}$$

Therefore, we increase rate for frames (in the order of frame number) in F so that each frame starts no latter than x and total rate increase is no more than the amount of decrease on frame j's rate.

Now the total buffer required for position x is less than what is required in the original schedule. This is because the storage requirement difference between the this new schedule and the original schedule at position y is $(j - y)(r_j - r'_j) + \sum_{k \in F}(k - y)(r_k - r'_k) \leq 0$, since $r'_j - r_j + \sum_{k \in F}(r'_k - r_k) \leq 0$.

Note that r_i denotes the original rate for frame i and r'_i denotes the new rate for frame i. We know that if starting reception times of the frames in F is at x or after x, then client storage constraint is satisfied. Therefore,

eventually, we have a rate allocation that ensures the client storage storage space is satisfied. Such a transformation can continue for positions after x. Finally, we have an allocation in which the first j frames have the same rate allocation as under the greedy scheme.

We can incrementally transform the frames in the order of frame number. Eventually, all frames' allocation is greedy.

■