RC 20734 (2/20/97)
Computer Science

IBM Research Report

Efficient Retrieval of Composite Multimedia Objects in the *JINSIL* Distributed System

Junehwa Song, Asit Dan, Dinkar Sitaram

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Efficient Retrieval of Composite Multimedia Objects in the *JINSIL* Distributed System

Junehwa Song, Asit Dan and Dinkar Sitaram
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{junesong, asit, sitaram }@watson.ibm.com

The paper will appear in the Proc. of the ACM SIGMETRICS Conference, June 1997.

Abstract

In a distributed environment, presentation of structured, composite multimedia information poses new challenges in dealing with variable bandwidth (BW) requirement and synchronization of media data objects. The detailed knowledge of BW requirement obtained by analyzing the document structure can be used to create a prefetch schedule that results in efficient utilization of system resources. A distributed environment consists of various system components that are either dedicated to a client or shared across multiple clients. Shared system components could benefit from *Fine Granularity Advanced Reservation (FGAR)* of resources based on true BW requirement. Prefetching by utilizing advance knowledge of BW requirement can further improve resource utilization. In this paper, we describe the JINSIL retrieval system that takes into account the available bandwidth and buffer resources and the nature of sharing in each componenton the delivery path. It reshapes BW requirement, creates prefetch schedule for efficient resource utilization in each component, and reserves necessary BW and buffer. We also consider good choices for placement of prefetch buffers across various system components.

1 Introduction

The rapid evolution of multimedia technologies has made feasible new ways of creating and presenting complex multimedia documents. Such documents consist of media objects of various types and granularities that are organized into meaningful chunks; e.g., slide presentation or story consisting of multiple (and even simultaneous) images, text data, as well as video and audio clips [10, 20, 15, 22]. Two examples of such composite presentations and resulting variations in the data consumption rates are shown in Figures 1 and 2. The first example is a report of a swimming competition where in addition to the global view, small video windows containing close ups of the leaders are shown towards the end of the report. The second example mingles video, images and narration of the interesting sights around Washington DC (details are provided later).

Efficient presentation of such structured, composite multimedia information (i.e., retrieval and synchronous playback) gives rise to new challenges. In a distributed multimedia environment, all or some pieces of a composite presentation object may reside in one or multiple remote systems away from the client presentation system (See, Figure 3). To avoid jitter in a presentation, appropriate resources need to be reserved on various data paths from the respective sources to the client system [2, 8, 6, 11, 12, 3]. For composite documents, the instantaneous total data consumption rate that needs

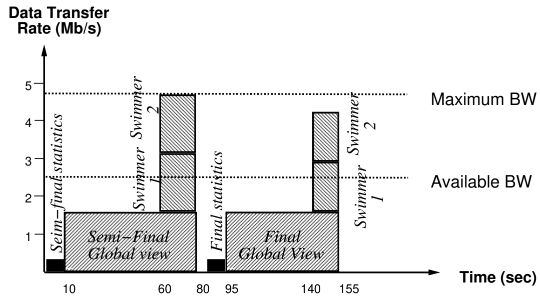


Figure 1: Olympic swimming competition

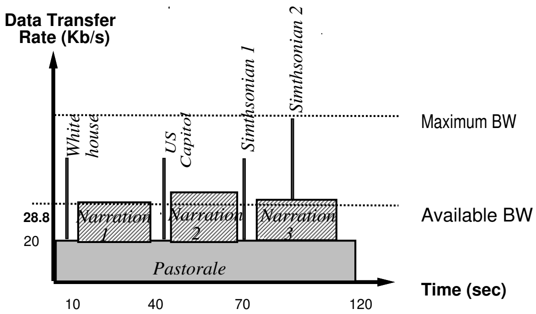


Figure 2: Tour of Washington D.C.

to be supported will vary over time depending on the structure of the presentation [16, 17]. Tables 1 and 2 show the data consumption rate for the above two presentation examples. The variable data rate makes the task of efficient allocation of resources in shared components more complex.

One simple resource allocation policy for applications with variable bandwidth requirement is to reserve a constant BW (equals to the maximum instantaneous bandwidth required by the presentation) for the entire presentation duration. However, there are several disadvantages to this simple approach. First, in many commercial environments (e.g., cable or phone connection to home) the bandwidth is limited at the final stage of the network. This is referred to as the "Last Mile Problem". Hence, it may be impossible or difficult

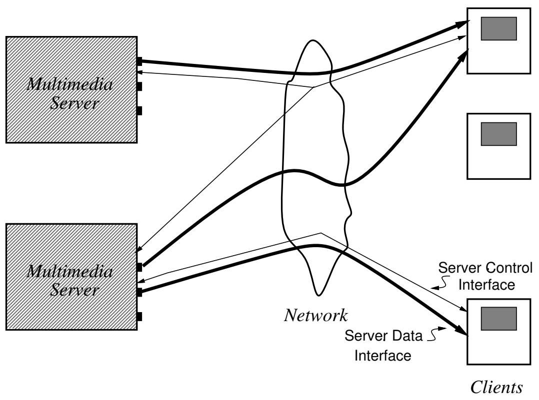


Figure 3: A distributed multimedia environment

to support presentations of complex media documents that require multiple streams of video, audio, image or text data (even for a short duration). Higher up on the data delivery paths (e.g., in the server) the bandwidth may be shared across multiple presentations. Allocating the peak bandwidth reduces the number of presentations that can be admitted.

In this paper, we describe the scheduling and retrieval policies of the *JINSIL* retrieval system that allocates appropriate resources in every component on the delivery paths, and creates object delivery schedules for each stage. The *resource allocation* and creation of a *prefetch schedule* in JINSIL addresses various issues, including obtaining variable BW requirement of a document by analyzing its structure, determining delivery paths based on location of objects, dedicated vs shared resources on these paths, and available buffer and BW resources in each system component. Note that BW variability in a presentation can come from two sources: variations in compression ratio of a stream [19, 9] and composition of different multimedia objects [16, 17, 18]. The JINSIL system deals with the second case, and can use the solutions proposed in [19, 9] for dealing with changing compression ratios.

The main focus of this paper are the scheduling and resource allocation issues addressed by JINSIL in an environment with shared components. First, at each delivery stage, for a given allocation of BW and buffer to a presentation, it creates a prefetch schedule for the atomic objects in the document. Second, if the system component supports the notion of *Fine Granularity Advanced Reservation (FGAR)*, the scheduler matches the resource allocation (e.g., BW and buffer) to the variability in available resources and the exact prefetch schedule, i.e., the traffic is shaped to match the *time-dependent* availability of BW and buffer. The resource allocation policies in JINSIL improve the efficiency of retrieval in a number of ways. The FGAR policy together with the reduction of the peak bandwidth requirement increases the feasibility of presentation, and hence, the number of streams that can be admitted by a system component. Additionally, as shown later, the time-dependent variation in bandwidth requirement may lead to periodic exhaustion of server bandwidth and hence, inefficient server utilization.

The remainder of the paper is organized as follows. Section 1.1 delineates the contributions of this paper and describes it relationships to earlier works. Section 2 describes typical workloads, and a brief overview of the JINSIL system. Section 3 introduces the general problem of BW and buffer satisfiability for a given presentation with variable BW requirement and describes the scheduling and retrieval policy used by the JINSIL system. This policy is referred to as the *Generalized Bandwidth and Buffer Constrained Scheduling (GBBCS)* policy. A performance study of these algorithms using simulation is presented in Section 4. Finally, summary and conclusions of the paper are presented in Section 5.

1.1 Contributions and Relationship to Earlier Work

The JINSIL retrieval system supports presentations of composite multimedia documents with variable BW requirement in both dedicated and shared environments. In an environment with shared components, the resources of a component (e.g., BW and buffer space) are utilized by all streams delivered through that component. Hence, the resources available to a single stream are not fixed. Additional complexities are introduced in dealing with such environments, especially in taking into account variable requirements of all streams and partitioning the available resources amongst the streams.

A number of papers have considered prefetching in a dedicated environment where a constant amount of BW and/or buffer is available per stream. Prefetching mechanisms in a dedicated environment to smooth the burstiness introduced by data compression in a single long data stream are studied in [19, 9].[1] Both papers address optimal delivery schedules consisting of piecewise constant rate such that neither the prefetching of data overflows the client buffer, nor does the chosen delivery rate cause buffer underflow. The optimality condition is defined either as the minimum number of changes in the delivery rate [9] or as the least variability in delivery rates [19].

Bandwidth and buffer satisfiability issues for a composite multimedia document in a dedicated environment are studied in [16, 17]. We extend this work in several ways. First, we consider end-to-end scenarios where some of the system components on the data path may be shared, and hence, efficient utilization of shared resources demands an object delivery schedule with minimum peak bandwidth requirement. We also consider system components that support the FGAR policy for BW and buffer. Structural analysis and time shifting (i.e., prefetching or delaying) have been used in the DEMON project [18] for presentation of composite documents in a networked environment. This has made better use of a fixed BW allocation per request by smoothing the bit rate requirements of a document. However, it has not exploited the FGAR policy for resource reservation.

There is a lot of prior work on storage retrieval to avoid jitter in data delivery [11, 12, 3]. Continuous retrieval of composite document using flash memory is considered in [20]. In [4], authors study the prefetching and delaying schemes to avoid contention in data retrieval from disk space. A characterization and workload of hypermedia applications is proposed in [13]. The workload is used

[1] In [19] performance of the proposed policy is studied in a shared network environment. However, the proposed algorithm did not explicitly take into account the available BW in the shared component (i.e., network) in reshaping the traffic.

to study (via simulation) the performance of the Continuous Media File System (CMFS).

2 Distributed Multimedia System Environments

In this section, we first describe the details of two specific workloads referred to in the earlier section. We then provide an outline of the JINSIL retrieval system.

2.1 Composite Multimedia Documents

A composite multimedia document consists of a mixture of media types (e.g., text, audio, video, image) which are presented according to some pre-specified temporal relationships. The variable BW requirements of a document can be captured in a *bandwidth profile* (described in detail later). In the following, we first describe two typical composite multimedia documents, and generate bandwidth profiles (to be used in the paper) by analyzing their structures.

object id	start time	duration	data rate
image 1	0	10	30 Kb
narration 1	3	6	20 Kb/sec
video 1	10	70	1.6 Mb/sec
video 2	60	20	1.5 Mb/sec
video 3	60	20	1.6 Mb/sec
image 2	85	10	28 Kb
narration 2	87	6	24 Kb/sec
video 4	95	60	1.5 Mb/sec
video 5	140	15	1.4 Mb/sec
video 6	140	15	1.4 Mb/sec

Table 1: Data rates in presentation of an Olympic competition

Olympic Swimming Competition This example shows the highlights of Olympic swimming competition. It is composed of 10 atomic media objects as is shown in the Table 1. Image 1 presents the statistics of the semifinal and the competitors in the game. Along with the Image 1, a narration (Narration 1) describes the statistics. After the image and the narration, Video 1 is played to show the global view of the competition. Near the end of the Video 1, two small screens are opened in the middle of the monitor, to show close up views of the two best competitors (Video 2 and Video 3). The playback of Video 2 and Video 3 overlap with the playback of Video 1. At the end of the game, the last frame of the Video 2 which shows the winner of the competition fills up the whole monitor and stays till the playback of the final competition. Final competition is similarly presented by Image 2, Narration 2, Video 4, 5 and 6, etc. Figure 1 shows the variability in the data consumption rate for this presentation.

D.C. Tour The second example provides a guided tour of Washington D.C. It shows three popular places in D.C. i.e., the White House, U.S. Capitol, and Smithsonian Mall. The playback starts with a background music, Beethoven's Pastorale, which is played throughout the presentation. The first image shows the White House and also gives a narrative description. After some moments, the second image (of the Capitol building) along with its corresponding narration is provided. Soon after, the third and fourth images (of the Smithsonian Mall) are presented one after another along with a narration for both. Figure 2 shows the variability in data consumption rates during this presentation.

object id	start time	duration	data rate
White house	5	30	30 Kb
US Capitol	40	30	28 Kb
Smithsonian 1	75	20	29 Kb
Smithsonian 2	95	15	30 Kb
Narration 1	10	22	15 Kb/sec
Narration 2	43	20	25 Kb/sec
Narration 3	78	24	16 Kb/sec
Pastorale	0	120	20 Kb/sec

Table 2: Data rates in tour of Washington D.C. presentation

Bandwidth profiles We represent the time dependent bandwidth requirement of a composite multimedia document by a bandwidth profile. A bandwidth profile is a list C ,

$$C = <(t_1^c, C_1), ..., (t_L^c, C_L), (t_{L+1}^c, 0)>,$$

where t_l^c represents a time at which the consumption rate changes (and is referred to as a *breakpoint*). The data consumption rate between two time points t_l^c and t_{l+1}^c is represented by C_l. Each client system generates the initial bandwidth profile for a composite document by considering the start times, play durations, and data rates of all its atomic objects. In the BW profile, the breakpoints are generated by taking the union of the start times and end times of all atomic objects. During each interval (t_i^c, t_{i+1}^c), the total data consumption rate of the presentation, C_i, is the sum of playback rates of all atomic objects during this interval. Note that for atomic objects of static types (e.g., images or text), all of the data need to be transfered before they can be displayed. For ease of analysis, these are simulated also as continuous type. Let an image $Image_i$ of size L Kb be played at t_i for d seconds. We consider it as a continuous data type with start time $t_i - \delta$, data rate L/δ, and duration δ. Using this notation, the BW profile for the Olympic Swimming Competition example can be written as,[2]

$$\begin{array}{llll} < & (0,\ 0.015\,\mathrm{Mb/s}), & (2,\ 0\,\mathrm{Mb/s}), & (5,\ 0.02\,\mathrm{Mb/s}), \\ & (11,\ 0\,\mathrm{Mb/s}), & (12,\ 1.6\,\mathrm{Mb/s}), & (62,\ 4.7\,\mathrm{Mb/s}), \\ & (82,\ 0\,\mathrm{Mb/s}), & (84,\ 0.014\,\mathrm{Mb/s}), & (87,\ 0\,\mathrm{Mb/s}), \\ & (89,\ 0.024\,\mathrm{Mb/s}), & (95,\ 0\,\mathrm{Mb/s}), & (97,\ 1.5\,\mathrm{Mb/s}), \\ & (142,\ 4.3\,\mathrm{Mb/s}), & (157,\ 0\,\mathrm{Mb/s}) & > \end{array}$$

We will use this bandwidth profile as the workload for our performance study in Section 4.

2.2 JINSIL System

[2] In this profile, δ are taken as 2 seconds for each static object. Also note that this presentation requires an initial delay of 2 seconds to accommodate initial data fetching before the presentation starts.

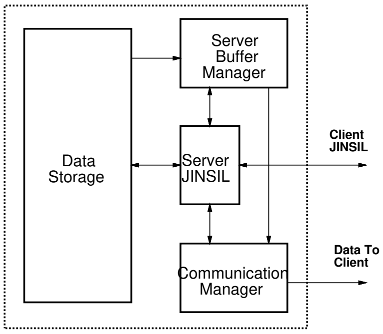


Figure 4: A server system

The JINSIL retrieval system is essentially a system for creating an efficient *object delivery schedule (ODS)* based on the presentation structure and for end-to-end allocation of appropriate resources (e.g., in client and server systems).[3] The ODS describes the starting point and delivery rate of each atomic object in a presentation.[4] Figures 4 and 5 show the interactions of the JINSIL module (i.e., scheduler and prefetcher) with client and the server systems. The authoring system stores the structure of a presentation in a metafile (details can be found in [15, 22]). Upon receiving an user presentation request, the *composite media player* invokes JINSIL for generating an ODS, and for retrieving the specified multimedia objects.

The JINSIL scheduler first generates a bandwidth profile for the requested presentation based on the document structure, and uses this profile to test for feasibility of presentation with the available client buffer and bandwidth. If feasible, the scheduler reserves the needed resources and constructs an ODS to be used for retrieval. The ODS may include prefetching of objects or delaying of the entire presentation. The presentation delay is the minimum delay required for a jitter free presentation (i.e., for continuous delivery). The atomic objects may reside at remote sites and hence, resources need to be reserved in each component on the data path. The JINSIL layer in each stage (e.g., client and server) will pass an ODS to the next stage as a delivery request. This schedule is used in turn by the scheduler at that stage to generate a new bandwidth profile. (The interactions and the issues in multi-stage delivery are discussed in Section 4.6.) Note that in a shared component (e.g., server), the scheduler will trade-off BW and buffer to maximize the throughput of the component. Details of the algorithms for testing for feasibility and resource allocation will be described shortly.

Once JINSIL is successful in resource allocation and creation of an ODS, the composite media player starts playing back the presentation. Note that the presentation may be delayed by JINSIL in constructing a feasible ODS. The composite media player starts accessing the virtual media file system (VMFS). The data is inserted into the filesystem buffer by the local or remote JINSIL prefetcher, depending upon the delivery mode (i.e., push or pull).

[3] Currently, network protocols can not use such information without provisions for an FGAR policy.

[4] The details of the ODS data can be found in [21]. The JINSIL uses ODS data structures for resource allocation, data retrieval as well as for communication between the system components. However, for simplicity of presentation in this paper, we will use a bandwidth profile to represent an ODS.

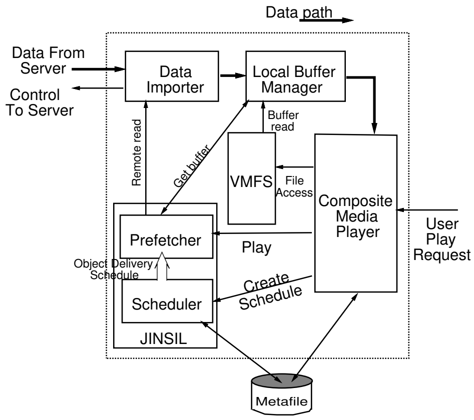


Figure 5: A client system

3 Resource Allocation Scheme

In this section, we describe the two main features of the resource allocation policy in JINSIL : *fine granularity advanced reservation (FGAR)* and *generalized bandwidth and buffer constrained scheduling (GBBCS).*

Fine Granularity Advanced Reservation (FGAR)

To maximize resource utilization in a shared component, it is necessary to take into account variability in resource availability and resource requirements. Since this may require reserving additional (or lesser) resources at a future time (instead of reserving the maximum currently), it is referred to as the *fine granularity advanced reservation (FGAR)* policy. The BW profile defined earlier describes BW requirements of a presentation. The BW and buffer space availability are represented by the lists λ and B, respectively, where $\lambda = <(t_1^\lambda, \lambda_1), \ldots, (t_M^\lambda, \lambda_M), (t_{M+1}^\lambda, 0)>$, and $B = <(t_1^b, B_1), \ldots, (t_{N-1}^b, B_{N-1}), (t_N^b, 0)>$.

Generalized Bandwidth and Buffer Constrained Scheduling (GBBCS)

In a system component where a fixed amount of bandwidth and buffer space are dedicated to a single client, the scheduler can test the feasibility of a requested presentation by using the algorithm proposed in [17]. However, the availability of resources will be time-dependent in system components shared by multiple clients. In such environments, the scheduling policy needs to reflect the time-dependent availability of the shared resources. The proposed GBBCS policy takes into account both the BW profile and time-dependent availability of resources in creating a delivery schedule for the atomic objects in a presentation.

Figure 6 shows the broad steps in the GBBCS policy used by by the client and the server systems. Note that once a presentation delay is introduced (say, in client) the subsequent stages (i.e.,

```
GBBCS(C, λ, B) {
    FeasibilityWithMinimumPrefetching(C, λ, B) ;
    if not schedulable,
        min_delay = ComputeMinDelay (C, B, λ) ;
        C_delay = DelayReq (C, delay);
    else C_delay = C;
    C_reshape = RsvMinPeakBW (C_delay, B, λ));
    return(min_delay, C_reshape, λ, B);
}
```

Figure 6: Generalized Bandwidth Buffer Constrained Scheduling

```
FeasibilityWithMinimumPrefetching(C', λ', B') {
  B°_P = 0;
  For p = P − 1, ..., 1
     { if (C'_p + B°_{p+1}/(t_{p+1} − t_p)) ≤ λ'_p,
          {B°_p = 0 ; λ°_p = C'_p + B°_{p+1}/(t_{p+1} − t_p) ;}
       else { λ°_p = λ'_p ;
          B°_p = B°_{p+1} + (t_{p+1} − t_p)(C'_p − λ'_p) ;
          if B°_p > B'_p, return (p,B°_p) ; } /* overflow at t_p */
     }
     return(0,B°_1);  /* if B°_1 == 0, request is feasible,
  }                       else initial prefetching is required */
```

Figure 7: Algorithm for Testing Feasibility

server) can not easily introduce further delay in the presentation. (This issue is discussed further in Section 4.6.) The parameters of the GBBCS policy are the BW profile (C), and the earlier defined availability lists of BW (λ) and buffer (B) of the invoking system component. The GBBCS policy first tests feasibility of presentation for the given C, λ and B. If the presentation is not feasible, it then checks for feasibility with a presentation delay. The *ComputeMinDelay* step is invoked to compute the minimum time by which the request has to be delayed for resource availability. Depending on the buffer availability, the presentation delay is used either to prefetch sufficient amount of data for a jitter free presentation, or to wait until enough BW and buffers are available. The GBBCS policy then modifies the BW profile as C_{delay} by shifting the profile C by the required amount of delay. Once the presentation is feasible, the step *RsvMinPeakBW* is invoked to generate a reshaped BW profile, $C_{reshape}$, for the current presentation request. The reshaped BW profile, $C_{reshape}$, uses prefetching whenever possible, to minimize the peak BW required by the request C_{delay}. The above procedure *RsvMinPeakBW* also reserves the required bandwidth and buffer, and updates availability lists λ and B. The details of these steps are given in the subsections 3.1 to 3.4.

Note that in a dedicated system component where λ and B are constant values, the smoothing algorithm proposed in [19] can also be used to create $C_{reshape}$. The algorithm in [19] has an additional objective of minimizing the variance in the reshaped BW profile.

The GBBCS policy has two advantages. First, as shown later in Section 4, variable bandwidth requirement may lead to periodic exhaustions in server bandwidth. This also results in wastage of server resources, since a request cannot be scheduled until enough resources are available, and the available resources remain idle. By reducing the variability in BW requirements, the GBBCS policy reduces this wastage in server resources. More importantly, the GBBCS policy is run in all (shared or dedicated) components on the end-to-end path. This may result in successive smoothing of the BW profile. Note that even when enough resources are available in a dedicated system component (say, in the client system), the reshaping of BW profile by JINSIL reduces fluctuations seen by the succeeding stages.

In remainder of this section describes in more detail the steps in GBBCS. We first set up a fundamental equation which describes the relationships among the time dependent resource availability and request. This relationship forms the basis for all the steps used in the JINSIL scheduler. We next describe an algorithm for testing the schedulability of a request that is used by the steps for computing minimum delay, and minimum peak bandwidth reservation. Further details of the steps can be found in [21].

3.1 Relationship amongst Consumption Rate, Reserved BW and Buffer

Let T^C, T^λ and T^B be the set of breakpoints in a BW profile, available BW and available buffer, respectively. Also, let $T^C = \{t^c_l, l = 1..L\}, T^\lambda = \{t^\lambda_m, m = 1..M\}$, and $T^B = \{t^b_n, n = 1..N\}$. To obtain the relationships among C, B, and λ in a feasible allocation, let us consider the combined set of breakpoints, T, i.e., $T = T^C \cup T^\lambda \cup T^B$. Let P be the cardinality of the set T, i.e., the total number of breakpoints in T. We redefine the data consumption rate, available BW and buffer space with respect to the combined set of breakpoints, T, by C', λ' and B', respectively, where $C' =< (t_1, C'_1), ..., (t_P, C'_P), (t_{P+1}, 0) >$, $\lambda' =< (t_1, \lambda'_1), ..., (t_P, \lambda'_P), (t_{P+1}, 0) >$, $B' =< (t_1, B'_1), ..., (t_{P-1}, B'_{P-1}), (t_P, 0) >$.

For multimedia presentations with continuous data consumption, the data should be delivered without causing any buffer underflow or overflow. Let λ°_p be the data delivery rate between the breakpoints t_p and t_{p+1}. Also, let B°_p and B°_{p+1} be the amount of prefetched data at breakpoints t_p and t_{p+1}, respectively. The relationship among C'_p, λ°_p and B°_p and B°_{p+1} can be characterized by

$$B^\circ_{p+1} = B^\circ_p - (t_{p+1} - t_p) * (C'_p - \lambda^\circ_p), \qquad (1)$$

where $(0 \le \lambda^\circ_p \le \lambda'_p)$ and $(0 \le B^\circ_p \le B'_p)$. The second term in the above equation represents the net data reduction (or accumulation) from the buffer during this interval. Any allocation of BW and buffer (λ° and B°) satisfying the above equation is referred to as a feasible allocation.

3.2 Testing for Schedulability of a Request

Given BW and buffer avaialability (λ' and B'), the algorithm in Figure 7 tests for feasibility of the retrieval request with BW

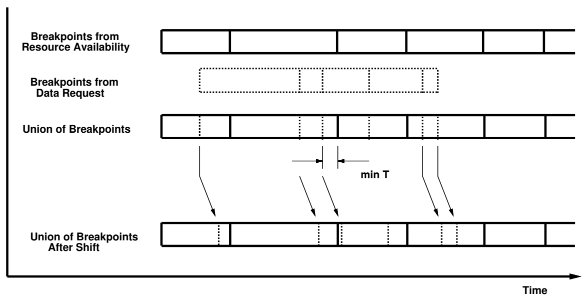


Figure 8: A time shift of a request and changes in breakpoints

profile (C'). If this is feasible, the algorithm also computes the minimum amount of data that needs to be prefetched at each break point. Otherwise, it returns the latest breakpoint at which the feasibility condition (Equation 1) is violated. (This breakpoint (t_p) and the prefetching amount (B°_p) would be used by the minimum delay computation step.) The algorithm steps through all break points starting from the last break point (t_P). In each iteration, it computes the minimum amount of data that needs to be prefetched at (t_p) to satisfy the data consumption rate during the interval $< t_p, t_{p+1} >$. If the required prefetching amount is greater than the available buffer size, the presentation is not feasible under the currently available bandwidth and buffer space. The procedure terminates at this point and returns the index of this breakpoint (i.e., at which buffer overflow occurs). The normal termination of the algorithm implies a feasible schedule has been found. If the initial prefetching amount (B°_1) is non-zero, the presentation has to be delayed to prefetch this data.

3.3 Computation of Minimum Delay

Computation of delay under dedicated resources: In a component with a fixed amount of available bandwidth and buffer space, it is straightforward to compute minimum delay before a presentation can be started (min_delay). Assuming λ_{const} as the constant value of available BW, min_delay can be computed as $min\_delay = B^\circ_1/\lambda$. Note however, delaying a presentation can not remove an buffer overflow condition in such dedicated environments, since available buffer and BW remain constant.

Computation of delay under the FGAR policy: In a shared component with fine granularity advanced reservation, the available resources are time dependent, Hence, the computation of min_delay is more complex. Let $B^\circ_1(t)$ be the initial prefetching amount for a data request with start time t. Similarly, $B^\circ_1(t + \Delta t)$ represents the initial prefetching amount after a time shift in the same data request by Δt. Let $dB^\circ_1(t)/dt$ represents the rate of decrease in the required prefetch amount. Then $dB^\circ_1(t)/dt$ can be estimated as $(B^\circ_1(t+\delta) - B^\circ_1(t))/\delta$ where δ is a small number. The required prefetch amounts, $B^\circ_1(t)$ and $B^\circ_1(t + \delta)$ are computed using the procedure *FeasibilityWithMinimumPrefetching*. Now, $B^\circ_1(t + \Delta t)$ can be estimated as,

$$B^\circ_1(t + \Delta t) = B^\circ_1(t) + \frac{dB^\circ_1(t)}{dt} * \Delta t. \qquad (2)$$

The required minimum delay is now computed by equating the required prefetch amount to the amount that can be prefetched with available BW, i.e.,

$$B^\circ_1(t + \Delta t) = \Delta t * \lambda'_1. \qquad (3)$$

Therefore, from Equations 2 and 3,

$$\Delta t = \frac{B^\circ_1(t)}{\lambda'_1 - \frac{dB^\circ_1(t)}{dt}}. \qquad (4)$$

Note that a time shift in C by Δt may result in a change in the relative orders of breakpoints (illustrated in Figure 8). The stripes in the bars (solid or dotted) represent the breakpoints in resource availability ($T^B \cup T^\lambda$), consumption rate (T^C), and union of these breakpoints before and after shifting of C'. Let $minT$ be the maximum amount of shift which can keep the current relative order of breakpoints, i.e.,

$$minT = min\{t_i - t^c_j \mid t_i \in T^\lambda \cup T^B, t^c_j \in T^C, t_i > t^c_j\}.$$

If $\Delta t \leq minT$ (Δt is obtained from 4), shifting in C by Δt does not affect the relative orders of breakpoints. Hence, Δt could be the required minimum delay. To check the satisfiability of the request C after a time shift of Δt the algorithm in Figure 7 is reexecuted. If the shifted request is feasible, the step *ComputeMinDelay* ends with Δt as the minimum delay. Otherwise, the above steps (i.e., computation of minimum delay) are repeated resulting in a further shift in the request.

If $\Delta t > minT$, i.e., shifting the request by Δt may change the relative order of break points, $\frac{dB^\circ_1(t)}{dt}$ no longer holds in the range $< t, t + \Delta t >$. Here, the request is first shifted by $minT + \delta$, where δ is a small number, and the steps for *ComputeMinDelay* are repeated. The shift $minT + \delta$ will result in changing of relative orders of breakpoints. Finally note that $\frac{dB^\circ_1(t)}{dt}$ in Equation 2 may not always be defined, since $B^\circ_1(t + \delta)$ may be undefined.

As mentioned earlier, the minimum delay is also required when a buffer overflow occurs at a breakpoint t_i during a feasibility test, i.e., $B^\circ_i > B'_i$. Here, a delay is introduced until the overflow is removed, i.e.,

$$B^\circ_i(t + \Delta t) = B'_i(t). \qquad (5)$$

Therefore, from Equation 5 and an equation similar to Equation 2,

$$\Delta t = \frac{B^\circ_i(t) - B'_i(t)}{\frac{dB^\circ_i(t)}{dt}} \qquad (6)$$

As before the steps may need to be repeated if $\Delta t > minT$.

3.4 Minimum Peak Bandwidth Reservation

Once a request succeeds in the feasibility test (after a possible delay), appropriate resources are reserved on behalf of this request. Equation 1 defines the set of feasible allocations. For maximal smoothing of the BW profile, JINSIL selects an allocation where the peak value in the reserved BW, i.e, $max_{1 \leq p \leq P}\{\lambda^\circ_p\}$, is the minimum among all feasible allocations (illustrated in Figure 9). We call this allocation a minimum peak BW reservation.

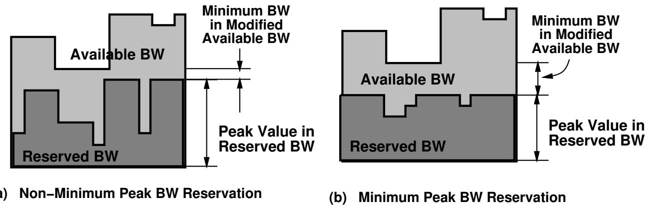


Figure 9: Illustrations of reservations with and without minimum peak BW

RsvMinPeakBW (C', λ', B') {
  $peakBW = \lambda_{lb}$;
  $INV = n$;
  for (;;) {
    $\lambda'_{new} = MIN(peakBW, \lambda')$;
    $p =$ *FeasibilityWithMinimumPrefetching* $(C', \lambda'_{new}, B', INV)$;
    if $(p == 0) \& (B_1^\circ == 0)$ **break** ;
    else { $INV =$ *ComputeINV* $(C', \lambda'_{new}, B', p, INV)$;
      $peakBW = peakBW +$
        MinimumIncrease$(C', \lambda'_{new}, B', p, INV)$; }
  }
  UpdateAvailability $(\lambda, B, \lambda^\circ, B^\circ)$;
  return $(\lambda, B, \lambda^\circ, B^\circ)$;
}

Figure 10: Reservations with minimum peak BW

The algorithm for the minimum peak bandwidth reservation is shown in Figure 10[5] Initially, the algorithm computes a lower bound on the minimum peak bandwidth, λ_{lb}, (detailed later) and sets initial value of peak bandwidth (peakBW) to this lower bound. The algorithm then iteratively computes the actual required minimum peak bandwidth as follows. First, it tests the feasibility of the data consumption request using *FeasibilityWithMinimumPrefetching*.[6] During this feasibility test, the available BW is changed to λ'_{new}, in which the the available BW at each breakpoint is set to the minimum of current peakBW and the actual available BW. If the feasibility test is successful, the current peakBW is taken as the required minimum peak BW. Alternatively, the failure implies a buffer overflow at a breakpoint, called the *Overflow Point* (OP). Hence, the algorithm computes a minimum increase in required peak BW, $\Delta\lambda$, that will eliminate this overflow. It subsequently increases the peak BW by $\Delta\lambda$. (Due to space constraint, the computation $\Delta\lambda$ is omitted from this paper.) The algorithm then starts the next iteration, and ends when the feasibility test is successful. During each iteration, the computation avoids consideration of breakpoints that are not affected by this increase in peak BW. The *Invariant Breakpoint* or INV is defined as the break point at which the required amount of data prefetching is not reduced by this change in peak BW. Hence, the computation starts from INV. Note that the OP and INV points

[5] This is a simplified version of the algorithm for constraints in space and ease of exposition. See [21] for further details (e,g., procedure *MinimumIncrease*).

[6] This procedure *FeasibilityWithMinimumPrefetching* is similar to the procedure in Figure 7, except the feasibility computation starts from a specified breakpoint INV.

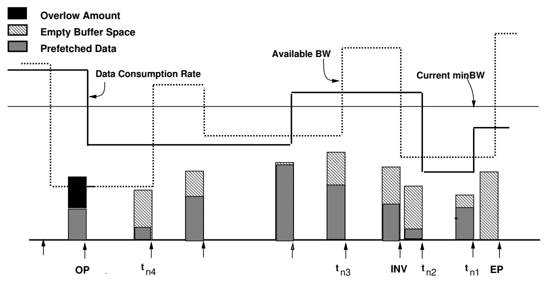


Figure 11: Illustrations of OP, EP and INV

progress in successive iterations in *RsvMinPeakBW*, and hence the algorithm always terminates.

Initial value of minimum peak bandwidth: To compute the initial value of minimum peak bandwidth, we consider the required minimum bandwidth in all intervals, $< t_p, t_{p+1} >$ assuming the maximum prefetching at t_p, i.e.,

$$\lambda_{lb} = \max_{0 \le p < n} \{((t_{p+1} - t_p) * C'_p - B'_p)/(t_{p+1} - t_p)\}.$$

Identification of INV: The procedure *RsvMinPeakBW* keeps track of the last breakpoint (referred to as *Buffer Empty Point* or EP) where the required prefetch amount is zero (see illustrations in Figure 11). EP itself is an Invariant Breakpoint, since increase in peak BW does not change the prefetch amount at EP. The procedure *ComputeINV* determines INV as follows. Starting from EP, the procedure moves toward OP to find the first breakpoint where the available bandwidth is larger than the current peak BW (t_{n3} in Figure(11)). This is the first point at which the allocation of the bandwidth can be increased beyond the current peak BW and therefore the required prefetching amount would be affected. The INV is the point where the prefetched amount remains unchanged.

Complexity of GBBCS: The algorithm *FeasibilityWithMinimumPrefetching* tests the schedulability by scanning each breakpoint once. Therefore, it takes $O(P)$ computation steps where P is the number of breakpoints to consider. The algorithm *ComputeMinDelay* uses *FeasibilityWithMinimumPrefetching* to compute $\frac{dB_i^\circ(t)}{dt}$ as well as to test satisfiability. Therefore, *ComputeMinDelay* takes $O(P * M)$ where M is the number of shifts until the minimum delay is found. Finally, the algorithm *RsvMinPeakBW* uses *FeasibilityWithMinimumPrefetching* in successive iterations. Since in each iteration the OP point progresses at least by one break point, the worst case complexity of *RsvMinPeakBW* is $O(P * P)$. In each iteration, the *RsvMinPeakBW* algorithm also uses *MinimumIncrease* to compute the minimum increase in peak BW, that has the worst case complexity of $O(P)$.

4 Performance Analysis

The performance of the GBBCS policy is evaluated in a two-stage client server environment consisting of a set of clients and a server. The total number of clients are varied in the experiments from 5 to 60. The clients are connected to the server via a dedicated network, and the server has a total BW capacity of 50 Mb/sec. We simulate various configurations with and without prefetch buffer space in the clients and server. The server prefetch buffer is shared by all clients connected to the server while the prefetch buffer in a client is dedicated to a single user. We consider client prefetch buffer sizes up to 12MB and server buffer sizes up to 250 MB.[7]

The BW profile used in the simulation is derived from the Olympic swimming competition example (see Section 2). The clients are assumed to be watching different multimedia documents, however, with the same BW profile. We also consider workloads resulting from presentations with different BW profiles.[8] The system is modeled as a closed system where each client generates a new request immediately upon service completion of its previous request. Note that if a request cannot be satisfied, it is delayed until a feasible delivery schedule is found. The system performance is measured by the server throughput and the average delay introduced by the server scheduler, where the throughput of the system is estimated as the number of request completions per second in the steady-state. For starting the system in a random state, the initial requests from each client is randomly spaced with exponential distribution. Empirically, it is observed that the system became stable after about 15 minutes. Hence, the performance measures are taken starting from 20 minutes after the startup. Observations were made for approximately 1 hour.

4.1 Effects of Fine Granularity Advanced Reservation (FGAR)

We first test the efficiency of the FGAR policy without any prefetch buffer in the system. The delay and throughput under the FGAR policy are compared to those of a policy with peak BW reservation in Figure 12. Figure 12(a) shows the average request delay as a function of the total number of clients in the system. The dotted curve represents the system with the FGAR policy while the solid curve is for the peak BW reservation policy. In both cases, the delay is zero for a smaller number of clients and beyond a certain point (15 for the FGAR policy and 10 for the peak BW reservation policy), increases linearly with the number of clients. The FGAR policy clearly outperforms the peak bandwidth reservation policy since it introduces a smaller average delay for the same number of clients. For 15 clients in the system, all requests can be served without any delay under FGAR policy whereas an average delay of approximately 80 seconds is required under peak BW reservation policy. The difference gets larger as the system becomes further overloaded with an increasing number of clients. Figure 12(b) shows the corresponding throughput as a function of the number of clients. The achievable throughput under the FGAR policy is approximately 50% higher than that of the peak BW reservation policy. In both cases, the throughput rises linearly until the system is saturated. However, in the case of FGAR policy, the throughput drops slightly after the saturation point. This is explained below. Once the system is overloaded, a presentation delay is introduced in all subsequent requests until enough resources start becoming available due to completion of ongoing presentations. This will result in periodic fluctuations in available BW, and hence, unpredictable throughput (e.g., a lower throughput in this example).

Figures 13(a) and 13(b) show the variability in available server bandwidth and presentation delay as a function of time for a system with 20 clients. Both the available server bandwidth and hence, presentation delay fluctuates periodically. There are time intervals where no client requests can be started, leading to wastage of server bandwidth. As shown later, pre-fetching of data can smooth out the bandwidth fluctuations and reduce this wastage of server bandwidth. The presentation delay also fluctuates periodically, following the periodic fluctuations in available BW (see, Figure 13(b)). The phenomenon of periodic exhaustion of server bandwidth leading to cyclic variations in throughput and hence, inefficient server utilization has also been observed in [1]. However, the system studied in [1] consisted of client requests for a single video object requiring fixed BW, and the server was subjected to a time-varying load by changing the arrival rate of client requests.

4.2 Effect of Reshaping BW Profile by Clients

In the presence of client prefetch buffers, it is possible to prefetch portions of the document objects to smooth out the BW requirement of a presentation. The relationship between the available buffer size and required peak bandwidth for the chosen workload is shown in Figure 14(a). As the available client prefetch buffer increases, the required peak BW (in server and network) is reduced. Hence, prefetching into client buffers can be used to compensate for limited network bandwidth.

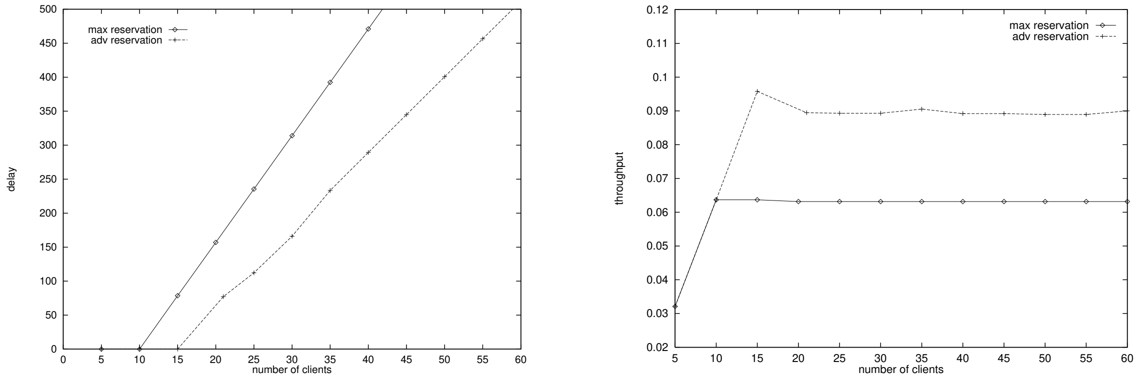
Figure 14(b) shows the impact of client reshaping on the server throughput as a function of the total number of clients. Three cases with varying amounts of prefetch buffer per client (no buffer, 4MB, and 11 MB) are shown. The required peak bandwidth in these cases are 4.7 Mb/s, 3.0 Mb/s and 1.5 Mb/s, respectively. The server is assumed have no prefetch buffer in these cases. For a larger number of clients, the throughput increases with better reshaping of BW profile, i.e., reduction in peak BW requirement. (The highest BW utilizations for these cases are 55%, 72% and 99%, respectively.) Also, in all cases, the throughput increases with the number of clients and then saturates. However, for a smaller number of clients, one interesting phenomenon is observed by comparing the relative order of throughput for these three cases. When the system is lightly loaded, the server utilization is the same for all three cases. However, a larger presentation delay and hence, a larger total service time for achieving larger reduction in peak BW requirement, dominates the relative order in server throughputs for these cases. The situation rectifies with a larger number of clients when the total service time becomes less relevant and reduction in peak BW actually improves server resource utilization.

4.3 Effect of Server Reshaping

We next study the use of server buffer to prefetch data (from disks) and reshape BW profile. The experiments are conducted by varying

[7] Note that apart from the prefetch buffers used for traffic reshaping, the system also maintains I/O and network buffers for continuous presentation to accommodate any jitter in the data retrieval or transmission [5, 19].

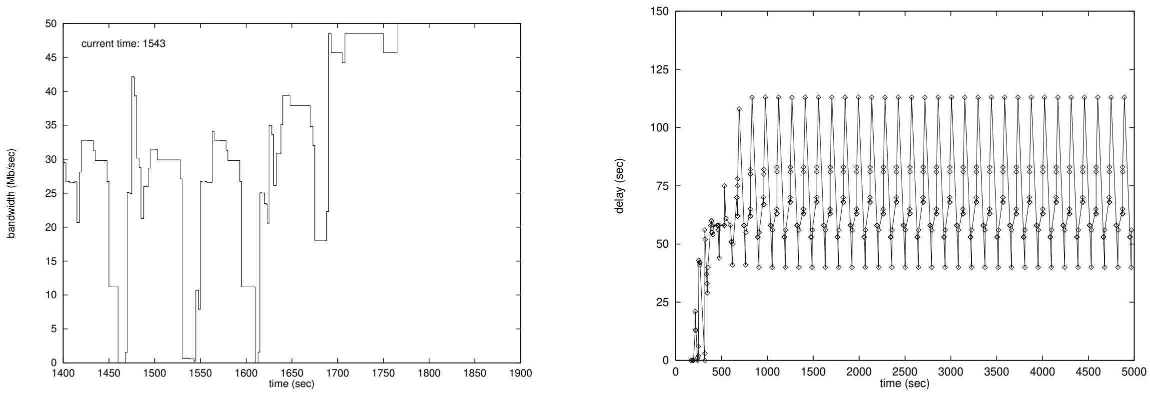
[8] Issues related to multiple requests for the same document or different documents sharing common objects are beyond the scope of this paper. Various caching schemes introduced in the literature could be used to exploit such sharing [7, 14]. Furthermore, the current framework (i.e., prefetching) can be extended to integrate caching policies.



(a) Average presentation delay

(b) System throughput

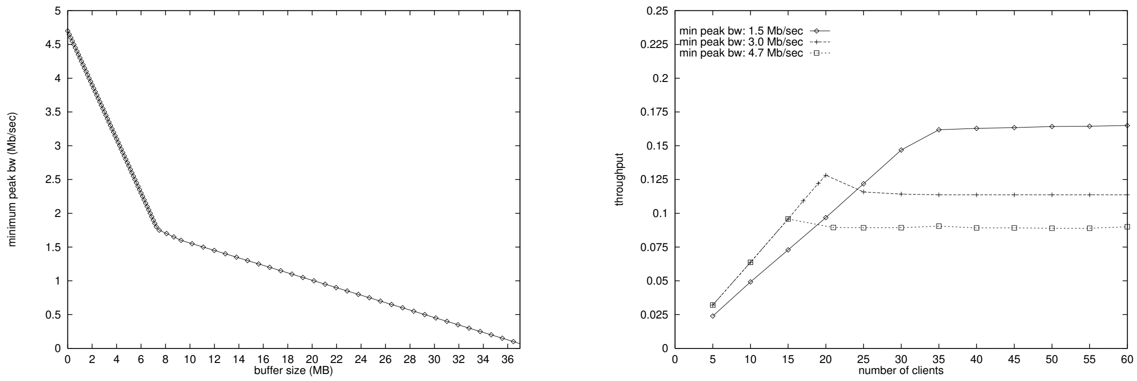
Figure 12: System performance under fine granularity advanced reservation



(a) Available BW

(b) Presentation delay

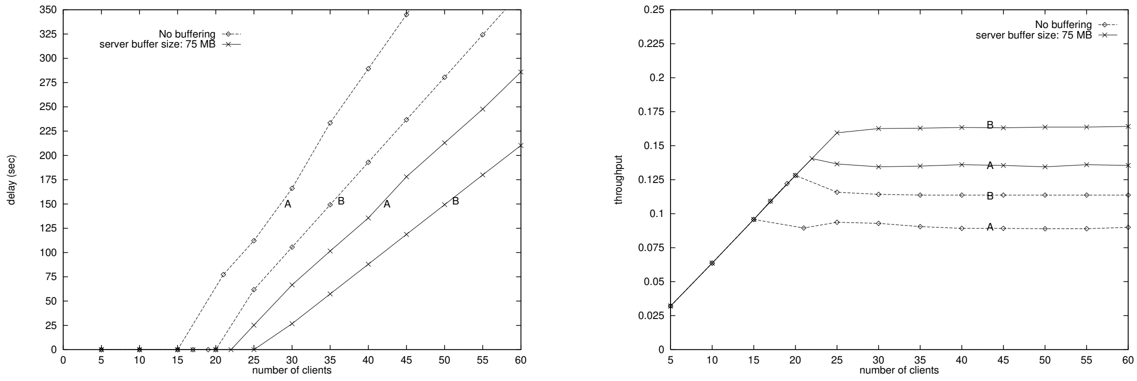
Figure 13: Variability in available server bandwidth and presentation delay (20 Clients)



(a) Client buffer size and corresponding minimum peak bandwidth

(b) Throughput

Figure 14: Effect of client reshaping to advanced reservation



(a) Delay (b) Throughput

Figure 15: Effect of server reshaping

both the sizes of client and server prefetch buffers. We consider two server configurations with and without a server buffer of size 75 MB. For each server configuration, two different client configurations are considered (with and without 4MB client buffer). Figure 15 shows the required average presentation delay and throughput for the above four configurations. The dotted lines are for the cases with no server buffer, while the solid lines represent the case with 75 MB server buffer. The curves marked with A represent the cases without client buffer (i.e., reshaping), while those marked with B represent the cases with 4 MB client buffer. Comparisons of the solid lines with the corresponding dotted lines show that server prefetching reduces the average delay significantly in both client configurations (i.e., with and without client reshaping). In both cases, the server buffer increases the achievable maximum throughput by at least 50%.

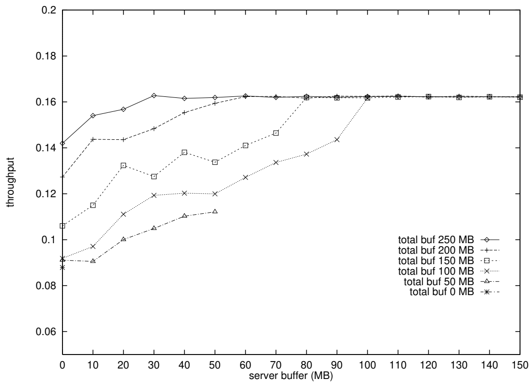


Figure 16: Partitioning of prefetch buffers between clients and servers

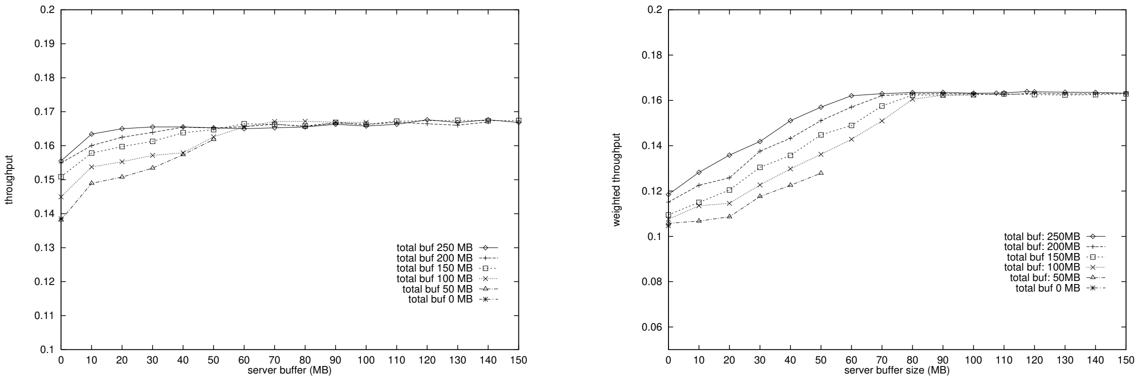
4.4 Placements of Prefetch Buffers

The experiments so far demonstrated the desirability of the FGAR policy combined with prefetching both on servers and clients. The efficacy of prefetching depends on the size of the prefetch buffer. Therefore, an important issue is how a given prefetch buffer should be distributed between the client and server systems. In the following experiments, the total buffer size in the system is kept fixed while the configurations are changed by varying sizes of the client and server buffers. We start with a configuration where all buffer space is evenly distributed among 40 clients. We then reduce the buffer size in each client in decrements of 0.25 MB and increase the server buffers size by 10 MB so as to keep the total amount fixed.

Figure 16 shows the effect of varying the distribution of buffer between the client and the server. The figure plots the average server throughput as a function of the amount of buffer in the server. Several curves, with total buffer size varying from 0 MB to 250 MB are shown. In each case, the throughput rises as the amount of server buffer is increased, indicating that the maximum throughput is achieved when all the buffer is in the server. This is because increases in server buffer can be shared among all the streams in the system, whereas increase in client buffer benefits only one client. Hence, from the point of view of maximizing server throughput, it may be desirable to allocate all the buffer to the server. However, a certain amount of client buffer is necessary not only to avoid jitter in transmission, but also to make feasible presentation of some composite documents in the face of limited network BW (e.g., "last mile problem"). One interesting point to note from the above figure is that the maximum server throughput can not be achieved without a buffer size of 100MB for this workload. This is also the configuration that maximizes the throughput while minimizing the total buffer requirement.

4.5 Mixed Workloads

In our earlier experiments, we have considered only a single document (and hence, BW profile) accessed by all clients. We next consider mixed workloads where clients choose randomly from a set of BW profiles. To make these profiles realistic, we have created 5 new profiles from the original profile. Profile 2 represents the same document with high resolution audio and video (e.g., MPEG-2). This is created by scaling up the profile by 3. Profile 3 is a low data rate document and includes only audio, image and text data. This is created by scaling down the profile by 6. Profiles



(a) Two profiles (MIXED2)

(b) Six profiles (MIXED6)

Figure 17: Throughput under various partitioning of buffer space: mixed workloads

4, 5 and 6 are created from profiles 1, 2 and 3 by shortening the presentation durations by half. In MIXED6 workload, the clients choose with equal probability one of the above 6 profiles upon a new request. Another workload, called MIXED2 was created by mixing two profiles where the clients choose with equal probability the profile 1 and its reverse profile, i.e., the BW requirement for playing the presentation backwards(!).

The experiments for studying proper placement of prefetch buffer reported in Section 4.4 is based on workload composed of single profile (i.e., profile 1). We now repeat these experiments for the MIXED2 and MIXED6 workloads. Figure 17 shows the results of these experiments. As before, the server throughput is measured by the average number of service completions per second. While in MIXED2 workload both the profiles retrieve equal amount of data, the different profiles in MIXED6 workload retrieve different amount of data. Hence, to normalize the throughput across all workloads, a service completion of each request is weighted by the amount of data retrieved relative to that of profile 1. The results shown in Figure 17 also confirm the observations made in earlier experiments in Figure 16. In both MIXED2 and MIXED6, the throughputs are increased (up to the maximum achievable value) while the server buffer space is increased. In both cases, maximum throughput is achieved for the curves with total buffer size of 100 MB or more. However, this maximum throughput is achieved for a server buffer size of 60 MB for the MIXED2 workload[9]

4.6 Resource Allocation in Multihop Environment

The JINSIL layer in each stage will pass an ODS to the next stage as a delivery request which is in turn used by the scheduler at that stage to generate a new ODS. However, the resource allocation by applying GBBCS remains a challenging problem in a system where the data path passes through multiple stages (see Figure 18). This is due to the fact that once a presentation delay, Δ, is introduced in a stage, S_i, other than the client (i.e., in some intermediate stage or in the data server) the satisfiability of the delayed presentation needs to be rechecked in all earlier stages. If the resources dedicated to the requesting client are fixed at all times in all the earlier stages, such a delay, Δ, does not alter the satisfiability of the requested presentation in all earlier stages nor does it changes the BW profile presented to stage S_i. Therefore, the effect is a simple time shifting of the ODS in all earlier stages. However, if resources are shared in any of the previous stages the schedule creation process needs to be started from the first affected stage (i.e., one with shared resources). Note that due to variability in available resources in various stages and hence, their interactions via traffic reshaping, the amount of delay by which the presentation needs to shifted is not easy to estimate.

[9] We conjecture that the server throughput is higher for the MIXED2 workload since the profile 2 somewhat complements profile 1 and thus reduces the variability in available BW.

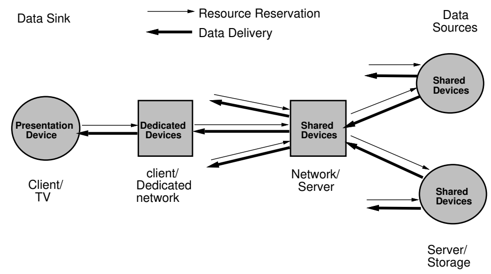


Figure 18: Illustrations of end-to-end (multi-hop) data delivery path

5 Conclusions

A composite multimedia document may consist of multiple video and audio clips, images and other data objects. The synchronous presentation of such structured, composite multimedia information poses serious challenges in a distributed environment where all or various pieces of a composite presentation document may reside in one or multiple remote systems away from the client presentation system. Appropriate resources need to be allocated in various data paths from the respective sources to the client system. Even if all the data are stored in a single system the instantaneous data consumption rate will vary over time depending on the structure of the presentation. Presentation of complex media documents may require multiple streams of video, audio, image or text data for a short duration. If sufficient peak bandwidth is not available, it may still be possible to playback such documents by prefetching part or some of the media objects. Higher up on the data delivery paths the bandwidth may be shared across multiple presentation. Reserving the peak bandwidth reduces the number of presentations that can be admitted. The variability of bandwidth requirements also leads to periodic exhaustion of server resources, leading to low server throughput.

In this paper, we describe the *JINSIL* retrieval system that addresses the above issues for presentation in a shared environment. It allocates appropriate resources in every component on the delivery paths, and creates object delivery schedules for each stage. The *resource allocation* and creation of *object delivery schedule* are based upon the document structure, location of objects and resulting delivery paths, dedicated vs shared resources on these paths, and available buffer and BW in each system component. For an environment where the system component supports the FGAR policy for buffer and BW, the JINSIL system matches the allocated resources (e.g., BW and buffer) to the exact object delivery schedule, i.e., it shapes the traffic to match the *time-dependent* availability of BW and buffer. It then creates an object delivery schedule between any two stages on the data delivery path, for a given allocation of BW and buffer on behalf of this presentation. Prefetching will reshape the traffic so as to minimize the peak bandwidth requirement, increasing the number of streams that can be served. Traffic reshaping also reduces the variability in bandwidth requirements. This alleviates the problem of periodic exhaustion of server resources and increases server utilization.

References

[1] K. Almeroth, A. Dan, D. Sitaram, and W. Tetzlaff, "Long-term channel allocation strategies for video applications", *IBM Research Report, RC 20249*, 1995. (to appear in INFOCOM 1997)

[2] David P. Anderson, "Metascheduling of continuous media", *ACM Transactions on Computer Systems*, 11(3):226–252, 1993.

[3] S. Berson, L. Golubchik, and R. R. Muntz, "Fault tolerant design of multimedia servers", In *ACM SIGMOD*, 1995.

[4] S. Chaudhuri, C. Shahabi, and S. Ghandeharizadeh, " Avoiding retrieval contention for composite multimedia objects", In *Proceedings of VLDB conference*, 1995.

[5] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, " Buffering and caching in large scale video servers", In *Proceedings of IEEE CompCon*, pages 217–224, 1995.

[6] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, " Channel allocation under batching and vcr control in video-on-demand systems", *Journal of Parallel and Distributed Computing*, 30(2):168–179, November 1995.

[7] A. Dan and D. Sitaram, " Multimedia caching strategies for heterogeneous application and server environments", *Multimedia Tools and Applications*, 4(3), 1997 (Special issue on *The State of the Art in Multimedia*).

[8] A. Dan, D. Sitaram, and P. Shahabuddin, " Scheduling policies for an on-demand video server with batching", In *Proceedings of ACM Multimedia*, October 1994.

[9] W. Feng, F. Jahanian, and S. Sechrest, " An optimal bandwidth allocation for the delievery of compressed prerecorded video", Technical Report CSE-TR-260-95, University of Michigan, August 1995.

[10] E.A. Fox, " Advances in interactive digital multimedia systems", *IEEE Computer - Special Issue on Multimedia Information Systems*, 24(11):9–19, October 1991.

[11] J. Gemmel and S. Christodoulakis, " Principles of delay sensitive multimedia data storage and retrieval", *ACM Trans. on Information Systems*, 10(1):51–90, 1992.

[12] J. Gemmel, H. Vin, D. Kandlur, V. Rangan, and L. Rowe, " Multimedia storage servers: A tutorial", *IEEE Computer*, pages 40–49, May 1995.

[13] C. Gopal and J. F. Buford, " Delivering hypermedia sessions from a continuous media server", In S. M. Chung, editor, *Multimedia Information Storage and Management*. Kluwer Academic Publishers, 1996.

[14] M. Kamath, K. Ramamritham, and D. Towsley, " Continuous media sharing in multimedia database systems", In *4th International Conference on Database Systems for Advanced Applications (DASFAA'95)*, April 1995.

[15] M. Kim and J. Song, " Multimedia documents with elastic time", In *ACM Multimedia Conference'95*, 1995.

[16] T. D. C. Little and A. Ghafoor, " Multimedia synchronization protocols for broadband integrated services", *IEEE JSAC*, 9(9):1368–1382, December 1991.

[17] T. D. C. Little and A. Ghafoor, " Scheduling bandwidth-constrained multimedia traffic", *Computer Communications*, 15(5):381–387, July 1992.

[18] J. Rosenberg, G. Cruz, and T. Judd, " Presenting multimedia documents over a digital network", *Computer Communications*, 15(6), July 1992.

[19] J. Salehi, J. F. Kurose Z. L. Zhang, and D. Towsley, " Supporting stored video: Reducing rate variablity and end-to-end resource reservation through optimal smoothing", In *ACM Sigmetrics*, 1996.

[20] C. Shahabi and S. Ghandeharizadeh, " Continuous display of presentations sharing clips", *Multimedia Systems Journal*, pages 76–90, 1995.

[21] J. Song, A. Dan, and D. Sitaram, " Jinsil: A system for presentation of composite multimedia objects in a distributed environment", IBM Research Report, 1997.

[22] J. Song, M. Kim, G. Ramalingam, R. Miller, and B. Yi, " Interactive authoring of multimedia documents", In *IEEE Symposium on Visual Languages*, 1996.