Selecting among Replicated Batching Video-on-Demand Servers *

Meng Guo, Mostafa H. Ammar, Ellen W. Zegura
Networking and Telecommunication Group
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332
{mguo, ammar, ewz}@cc.gatech.edu

ABSTRACT

A Video-on-Demand (VoD) service offers a large selection of videos from which customers can choose. Designers of VoD systems strive to achieve low *access latency* for customers. One approach that has been investigated by several researchers allows the server to *batch* clients requesting the same video and to serve clients in the same batch with one multicast video stream. This approach has the advantage that it can save server resources as well as server access and network bandwidth, thus allowing the server to handle a large number of customers without sacrificing access latency. VoD *server replication* is another approach that can allow a VoD service to handle a large number of clients, albeit at the additional cost of providing more servers. While replication is an effective way to increase the service capacity, it needs to be coupled with appropriate selection techniques in order to make efficient use of the increased capacity. In this paper, we investigate the design of server selection techniques for a system of replicated batching VoD servers. We design and evaluate a range of selection algorithms as they would be applied to three batching approaches: Batching with Persistent Channel Allocation, Patching, and Hierarchical Multicast Stream Merging (HMSM). We demonstrate that server replication combined with appropriate server selection scheme can indeed be used to increase the capacity of the service leading to improved performance.

1. INTRODUCTION

A Video-on-Demand (VoD) service offers a large selection of videos from which customers can choose. The service is *on-demand* because customers expect to receive their videos *soon* after they issue their requests to the server. Designers of VoD systems, therefore, strive to achieve low *access latency* for customers. One of the main challenges in the design of a VoD system is how to allow the system to handle a large number of customers, maintain a low cost of operation and at the same time provide for acceptable access latency.

*This work is supported by NSF grant ANI-9973115 and the Georgia Tech Broadband Institute.

NOSSDAV'02, May 12-14, 2002, Miami, Florida, USA.

One approach that has been investigated allows the server to *batch* clients requesting the same video and to serve clients in the same batch with one multicast video stream [1, 2, 6, 17]. This approach has the advantage that it can save server resources as well as server access and network bandwidth, thus allowing the server to handle a large number of customers. VoD data delivery schemes can be broadly classified into two categories: *periodic broadcast* and *scheduled multicast* [8]. Periodic broadcast is also referred to as data centered approach. In this approach, the server channel is dedicated to video objects and is broadcasting repeatedly [8, 1, 22, 19]. Scheduled multicast, also called user centered approach, dedicates channels to individual users. When the server channel is free, the server selects a batch of clients to multicast according to some scheduling policy [5, 6, 10, 15, 16]. The server selection problem is more interesting for the scheduled multicast approach, since the server behavior is based on client demand. In this paper, we focus on scheduled multicast schemes only.

VoD *server replication* is another approach that allows a VoD service to handle a large number of clients, albeit at the additional cost of providing more servers. In such a system, servers maintaining the same videos are placed in multiple locations in the network. The main question when server replication is employed is that of server selection, i.e., how to direct clients to the *best* server. There has been a significant amount of research recently regarding server selection issues [18, 4, 11, 21, 23], with a focus on unicast services in which a client is served by a single point-to-point flow from a server. The only prior investigation of multicast server selection that we are aware of is the work reported in [12, 13, 14]. That work, however, considers servers that are continuously transmitting multicast data (as would be the case in a live video or audio broadcast). In such a system, the only important performance metric is overall network bandwidth consumption, and there is no consideration of client latency or server access bandwidth.

In this paper we study the design of a VoD service in which a set of batching and multicast servers are replicated. Our objective is to investigate the benefits of replication for such a service. Our work focuses on the design of server selection techniques for such a replicated service and the effect of the server selection approach on the performance of a replicated batching-server VoD system. Because the VoD servers use multicast, the issues inherent in the design of a server selection algorithm differ in important ways from unicast server selection work. For example, when deciding to allocate a new client to a multicast server, it is important to take into account other clients that were previously assigned to the same server and will ultimately be batched with the new client. Our work differs significantly from previous multicast server selection problem as well. User access latency is the most important performance metrics in VoD service, this in turn is determined by the server disk bandwidth, batching status and batch scheduling policy of the server. To achieve desirable user access latency, the state of VoD servers (defined by the batches it is currently maintaining) needs to be considered in making server selection decisions.

One of the main motivations for replicating a server is to provide a client with the requested video program in shortest time possible. At the same time, it is desirable to serve client's requests by relatively closer server, so as to reduce the total network bandwidth required, and possibly to improve the quality of the received video. We also want to distribute the VoD traffic across the network to prevent *hot spot* from generation. Our work considers selection algorithms as they would be applied to three batching approaches: Batching with Persistent Channel Allocation [1], Patching [20], and Hierarchical Multicast Stream Merging (HMSM) [10]. In this work, we explore a range of selection techniques that incorporate, to various degrees, knowledge of the batching and allocation schemes of the servers.

This paper is organized as follows. We provide some background on the various batching schemes we consider in Section 2. In Section 3, we describe server selection techniques. In Section 4 and Section 5, we set up the simulation environment, and show some sample results of our performance evaluation experiments. We discuss some implementation issues in Section 6. Finally, we conclude this paper in Section 7.

2. BATCHING VOD SERVERS

A VoD server receives requests from clients and schedules the transmissions of videos over a set of channels. For a *batching* VoD server, a video transmitted on a particular channel is multicast to the clients in the particular batch. There is typically limit on how many videos can be transmitted simultaneously from a single video server, for example, due to limited server disk access bandwidth.

Figure 1 illustrates a typical batching-VoD server structure. When the server receives a client request, it will schedule the client request immediately if there is a free channel available. Otherwise, the client request will be put into a video batch, indexed by the video stream ID. Whenever a channel becomes available, the server will select a batch and allocate the channel to this batch for transmission. Video transmitted over a particular channel is multicast to the clients in the particular batch.

Generally speaking, a batching VoD server operates in two phases: *batch scheduling* and *channel allocation*. Batch scheduling decides which batch should be selected when there is a free channel available. Channel allocation deals with whether the channel is used to transmit the whole video program or only a portion of it.

For the batch scheduling phase, many scheduling policies have been proposed [7, 6, 1]. The *first come first serve (FCFS)* approach schedules the batch whose first client comes

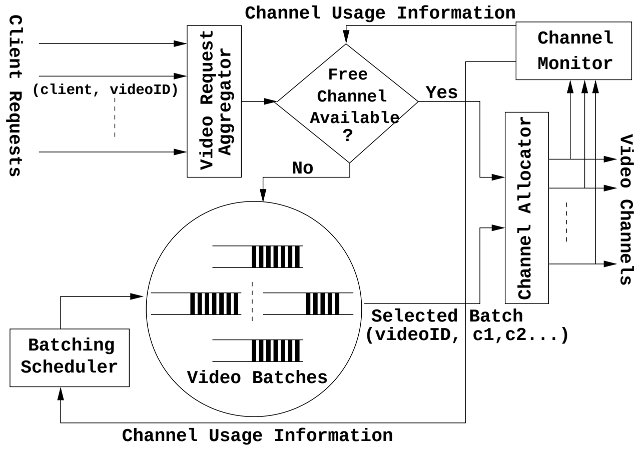


Figure 1: VoD Server Structure

the earliest, with the aim of achieving some level of fairness. The *maximum queue length first (MQL)* approach schedules the batch with the largest batch size, with the aim of maximizing throughput. The *maximum factored queue length (MFQ)* approach [1] selects the batch with the maximum value of $q_i/\sqrt{f}_i$, where q_i is the queue length (number of clients in the batch) for video i, and f_i is the relative frequency of the arrivals of video i. MFQ approach yields excellent empirical results in terms of latency, throughput and fairness. In our work, we assume that all the VoD servers use MFQ as the batch scheduling policy.

The second phase is channel allocation. In this paper, we study three channel allocation policies:

- The *persistent approach* [6, 1]: Once the channel is allocated to a batch of clients, it will continue to be used for multicasting the entire video to the batch.
- The *video patching approach* [20, 10, 5, 17]: This channel allocation technique is designed to provide for better server channel utilization and thus lower user-perceived access latency than the persistent allocation scheme. In video patching, video channels are classified in two categories: regular channels and patching channels. Regular channels transmit the entire video stream, while patching channels only transmit a portion of a video as needed. When the server allocates a free channel to a batch of requests (according to some batch scheduling policy), the server first scans all the on-going channels. If there is no regular channel distributing the desired video, or the starting time of the latest regular channel which is transmitting the desired video is too early for the client to patch, this channel is allocated as a regular channel. Otherwise, the channel is allocated as a patching channel. In this case, the clients first receive data from both the regular channel and the patching channel. The data from the patching channel is used to make up for the data the clients are missing from the regular (already on-going) channel. While receiving from the patching channel, the data being received from the regular channel is buffered by the clients for playout when needed. When the missing portion is completely received, the patching channel is released, and the clients will be merged into the regular channel.

- The *hierarchical multicast stream merging (HMSM) approach* [10]: This approach attempts to capture the advantages of dynamic skyscraper [8], as well as the strengths of stream patching [20]. In HMSM, clients that request the same file repeatedly merge into larger and larger groups, leading to a hierarchical merging structure. In this way, the required server bandwidth can be further reduced by repeatedly merging channels. In ordinary video patching, only regular channels can be merged into, while in HMSM, there is no discrimination between regular and patching channels. All the previous channels that are within a "patchable" range can be merged into. When there are multiple previous channels within the patchable range, we use closest target (CT) scheme to decide which earlier channel to merge into. This scheme simply chooses the closest earlier stream still in the system as the next merge target [9].

The first channel allocation policy requires client access bandwidth to be equal or larger than the video streaming rate. The latter two channel allocation approaches require client bandwidth at least two times of the video streaming rate, and an extra client side buffer.

3. SERVER SELECTION ALGORITHMS

We are concerned in this paper with a scenario in which a set of replicated batching VoD servers are distributed across the network. We assume that all the servers carry the same set of videos and employ the same batch scheduling and channel allocation policies. A large set of clients make requests for video and we are concerned with the question of how to best direct a client's request to one of the replicated servers.

In Section 5, we examine the performance of different server selection algorithms, while in Section 6 we discuss the issues and overheads involved in gathering the information.

3.1 Algorithm Description

Two types of information are potentially useful in designing a server selection algorithm: information about *topology* (e.g., the number of hops from a client to a VoD server) and information about the *servers* (e.g., size of the MFQ queue for the requested video). We propose six server selection algorithms that use increasingly more information to make a decision.

• The first option, the *Closest-server-first* algorithm, selects the server which is closest to the client using the network hop count measure. This provides a baseline illustrating the performance in the absence of any information about the servers.

• The second option adds a minimal amount of information about the servers. Specifically, the *Optimized closest-server-first* algorithm selects the closest server among those with free channels. If no servers have free channels, the closest server is selected.

We next turn to methods that are aware that the VoD servers are using MFQ scheduling. In each of these methods, the client always prefers a server with free channels. However, if there is no such server, these methods aim to select a server that will schedule the client request earliest.

• The *Register-all* algorithm will put the client request into the corresponding queue at all of the video servers. When the client request is satisfied at any one server, the request is withdrawn from the other server queues. This method loads all of the servers with the request and requires the ability to withdraw a request from a server. On the other hand, it is quite simple from the client viewpoint.

• As an alternative, the *Maximum-MFQ-rank-first* algorithm computes the destination queue *rank* at each server and sends the client request to the queue with the best MFQ rank. That is, best MFQ rank is used as a heuristic to select a server that will reduce the latency to serve the client request. This method uses more information than Register-all, but avoids sending the request to all servers and then withdrawing.

• Our *Minimum Expected Cost (MEC)* algorithm is more sophisticated. It makes use of the number of free channels per server, the hop count to each server, and the MFQ value at each server. Shorter access latency and less bandwidth consumption lead to a smaller expected cost. The *expected cost* measure is a weighted sum of three terms, and the client selects the server with smallest expected cost. The first term is proportional to the inverse of the number of channels; the second term is proportional to the MFQ value; the third term is proportional to the hop count. Specifically, the expected cost [1] at server i is

$$\frac{W_1}{a+c_i} + \frac{W_2}{m_{i,j}+1} + W_3 \times d_i$$

In this formula, i is the server number, j is the videoID. $m_{i,j}$ means the MFQ value of video j at server i, c_i is the number of free channels at server i, a is the adjustment parameter which is the load balancing threshold controller, and d_i is the hop count from the client to server i. The W_k's are the weight associated with the various terms. These weights are assigned so that $W_1 >> W_2 > W_3$.

How this expected cost value determines the server selection decision can be explained under two cases: some servers have free channel, no server has free channels available. For the first case where some servers have free channels, under above weight allocation, these servers have lower cost value than those servers with no free channels. Furthermore, when the number of free channels at each server is small, the difference of c_i at server i dorminates the whole cost value. Secondly, when all the servers are busy, the first term is same for all servers. Under such a situation, we assign $W_2 > W_3$, which means a high MFQ value is preferred over a low hop count, since user access latency has higher priority than bandwidth consumption.

• Our last algorithm, *Merging-Aware Minimum Expected Cost (MAMEC)* is appropriate when the servers are using patching, either of the basic style or HMSM. Specifically, it adds one additional term to the server cost described above. This term reflects the preference for a server that can start a patching channel soon. To evaluate whether a patching channel can be used, one must examine the difference in the expected time for this batch to be served, as compared to the most recently scheduled regular channel. If the expected difference is too large, the request will not be served with a patching channel. If the expected difference is not too large, then servers with smaller expected difference are preferred since they will be able to merge sooner. We define a function

[1]One can envision other possible composite cost functions. This particular measure gives good performance in simulations.

$e(i, j)$ to represent this term.

$$e(i,j) = \begin{cases} r_{i,j} - s_{i,l} & r_{i,j} - s_{i,l} + avg_j < B \\ \frac{N}{2} & r_{i,j} - s_{i,l} + avg_j > B \\ N & r_{i,j} - s_{i,l} > B \end{cases}$$

Here, $r_{i,j}$ denotes the time when the client requests video j at server i, $s_{i,l}$ means the starting time of the latest regular channel l which is broadcasting video j, avg_j is the average latency when requesting video j, B is the client side buffer size in terms of video playback time, and N is some large number.

We then use the following formula to describe the merging aware expected cost measure. We allocate the weights such that $W_1 >> W_4 >> W_2 > W_3$.

$$\frac{W_1}{a+c_i} + \frac{W_2}{m_{i,j}+1} + W_3 \times d_i + W_4 \times e(i,j)$$

The client selects the server whose merging aware expected cost value is smallest. The MAMEC algorithm works similar to the MEC algorithm described above, but with a new term that indicates the channel merging probability. Note that the new term is given weight W_4 much larger than W_2 and much smaller than W_1 above, so that this term will not influence the client's preference for a server with free channels. When all the servers are busy, the client will select the server whose batch is most likely be merged into other channels.

3.2 Dual server reception

We also explore the option of *dual server reception* in which a client is allowed to receive video stream from channels of different VoD servers. Dual server reception only makes sense under video patching and HMSM schemes. Dual server reception allows a much wider channel patching range, and can achieve more efficient channel merging.

On the other hand, batch scheduling and channel allocation as well as server selection under this scenario is more complex. Whenever the server is ready to schedule a batch on a free channel, it has to contact other servers to decide whether this channel should be a patching channel or a regular channel. If it decides the newly allocated channel is a patching channel, it has to inform other servers to start sending videos to the clients in this channel.

Server selection now actually involves two separate steps. For the first step, the client selects the server using the algorithms described above. In the second step, there might be more than one server whose channels are within the patchable range, there is another server selection process required to decide which server the channel should be merged into. Currently, we only explore an algorithm which aims to reduce the client side buffer size, which means we always patch to the server whose patchable channel's starting time is closest to this newly allocated channel.

4. SIMULATION SETUP

In this section, we set up the simulation environment and describe the performance metrics for the proposed server selection algorithms.

4.1 Simulation Environment

We use the GT-ITM transit-stub model to generate a network topology [3] composed of about 1400 routers. The average degree of the graph is 2.5, and core routers have a much higher degree than edge routers. VoD servers and clients are associated with the edge routers.

In our current simulation, all the VoD servers have the same configuration. Each video server has $M = 100$ video programs, and has a capacity of transmitting $C = 1000$ video streams simultaneously. Other than the constraint on the video server, we assume infinite link bandwidth. We further assume that each video program has a play time of 100 minutes. MFQ is used in all the servers, and an alternative technique is used to calculate the factored queue length. We use $\sqrt{q_j \times \Delta t_j}$ as the factored queue length, where for video j, Δt_j is the time interval since the last time this video is scheduled.

Each client's request can be represented by a three tuple $(requesttime, clientID, videoID)$. The client request arrival conforms to a Poisson process with an average rate of λ (20 $\sim$ 110 per minute). The client which sends out request is randomly distributed all over the network. The video selection probability conforms to a Zipf distribution. For λ requests per minute, $0.7 \times \lambda$ requests are for the 7 hot movies; $0.225 \times \lambda$ requests are for 18 normal videos; $0.075 \times \lambda$ requests are for the 75 unpopular movies. In our experiment, the total simulation time is one day.

4.2 Performance Metrics

We are interested in three performance metrics: user perceived latency, network bandwidth consumption and channel merge rate. We describe these three performance metrics in detail.

User perceived latency L is measured by the total amount of latency experienced by the clients over the total number of client requests. We define R to be the set of all client requests; L_r represents the access latency for request r.

$$L = \frac{\sum_{\forall r \in R} L_r}{|R|}$$

Network bandwidth consumption B is measured by the total amount of bandwidth used by all the multicast channels throughout the simulation time. Each video channel serves client requests through a multicast tree t; all the multicast trees used forms set T. $|t|$ denotes the number of edges in tree t. Thus, the network bandwidth consumption can be represented as:

$$B = \sum_{\forall t \in T} |t|$$

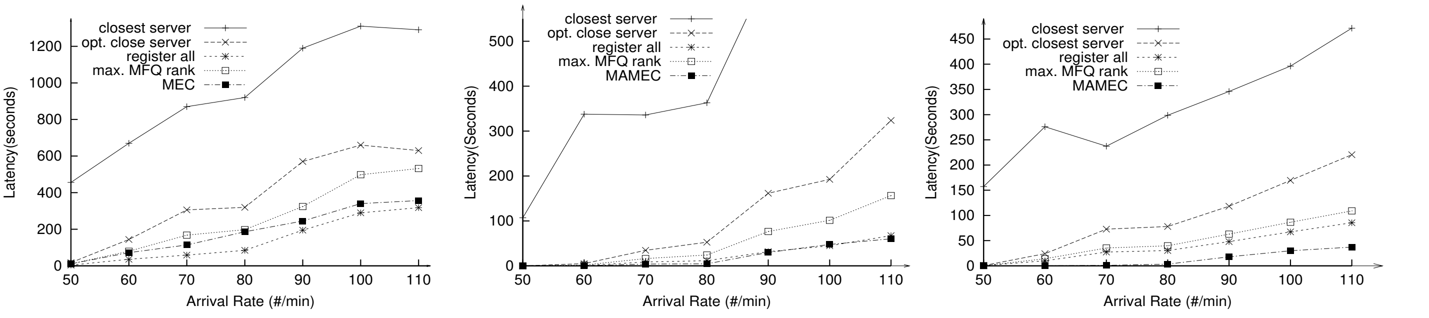
We also define channel merge rate as the number of clients that are merged over the total number of clients.

$$r = \frac{\sum_{\forall p \in P} |p|}{\sum_{\forall a \in A} |a|}$$

In this formula, P denotes the set of patching channels, and A denotes the set of all the scheduled channels. $|p|$ and $|a|$ means the number of clients in the channel. Channel merge rate represents the effectiveness of batching in the system.

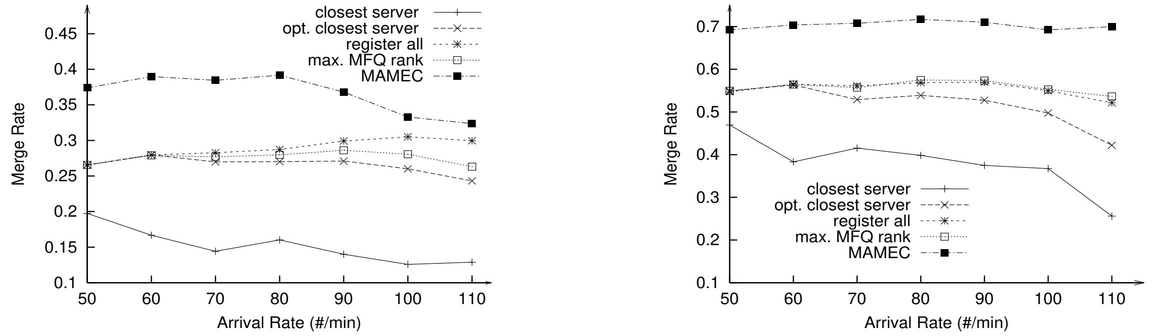
5. PERFORMANCE EVALUATION

We conduct simulation experiments on three schemes: persistent channel allocation, video patching, and HMSM. First we compare the user-perceived latency under different server selection algorithms. We then study the influence of the number of servers on the overall performance. Finally, we consider the effect of dual server reception on performance.



(a)Persistent Allocation (b) Video Patching (c) HMSM

Figure 2: Client Perceived Latency



(a) channel merge rate in video patching (b)channel merge rate in HMSM

Figure 3: Channel Merge Rate Comparison

5.1 User Perceived Latency

Figure 2 shows user perceived latency under the three channel allocation schemes. The x axis represents the number of requests for a certain video every minute, the y axis represents the access latency. *Closest server* selection approach performs worst under all three schemes, since it does not consider any batch scheduling information. Optimized closest server selection significantly reduces user perceived latency, simply by considering the number of free channels at each video server first. The *Register all* approach has very short access latency, the reason is that each client always select the server which schedules the client request earliest. The MEC algorithm latency is very close to the Register all approach, which shows our expected cost computation formula is fairly reasonable.

Under video patching and HMSM schemes, the client access latency is much smaller than the persistent channel allocation scheme, since these two schemes make better use of multicast channels by intelligently merging the client batches during the video transmission process. We also note that MAMEC algorithm performs best among all server selection approaches.

Given the server capability, the user perceived latency is primarily determined by two factors: the server load and batching efficiency. We explain the performance difference by analyzing how the client requests are directed to the different servers under the persistent allocation scheme, and by analyzing the channel merging behavior under video patching and HMSM. Figure 4 shows the server load distribution under the persistent channel allocation scheme (patching and HMSM has very similar server load distribution).

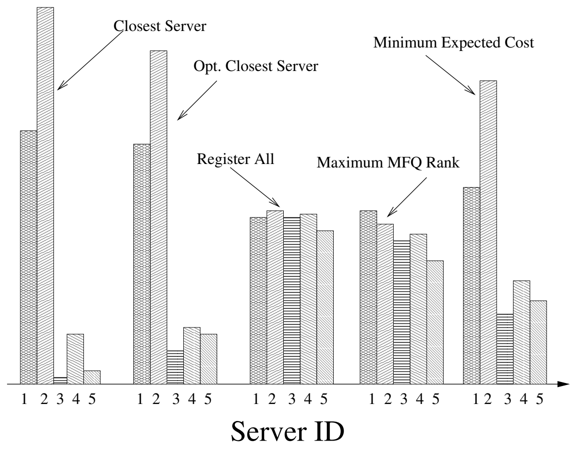
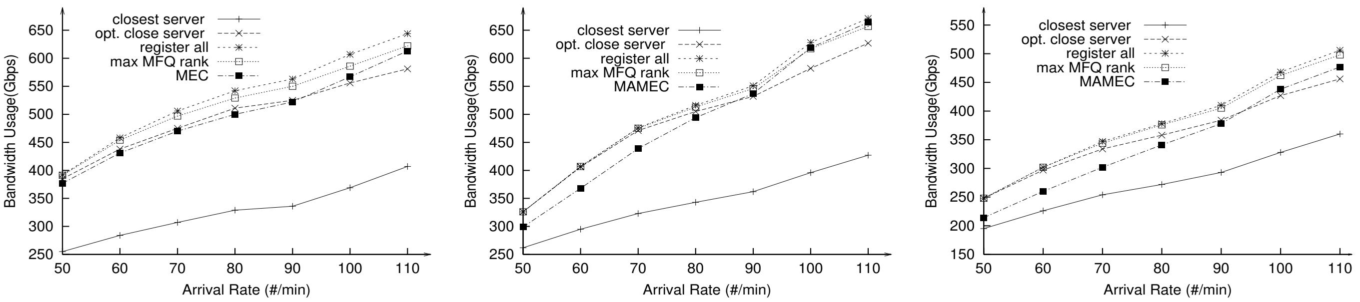


Figure 4: Server Load Distribution

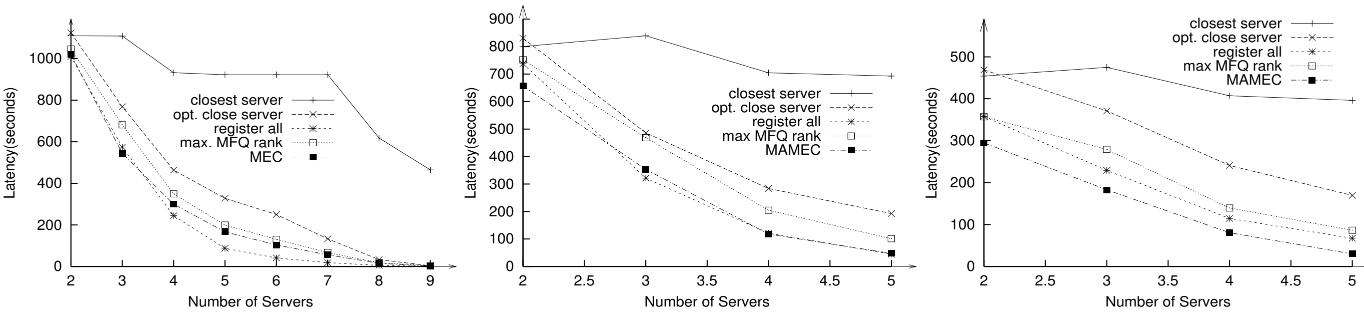
The Closest server selection algorithm always directs client requests to the closest server, causing servers 1 and 2 to be overloaded while servers 3 and 5 are barely used. Maximum MFQ rank algorithm and Register All algorithm does not consider the topology information at all, and therefore, the client requests are evenly distributed on the different servers. The MEC algorithm considers both server state and topology information; it achieves better load balancing than the Closest server selection algorithm, but favors the well-located (closer) servers.

Figure 3 shows the channel merge rate under different server selection approaches. Under video patching and HMSM schemes, the user perceived latency is tightly correlated with



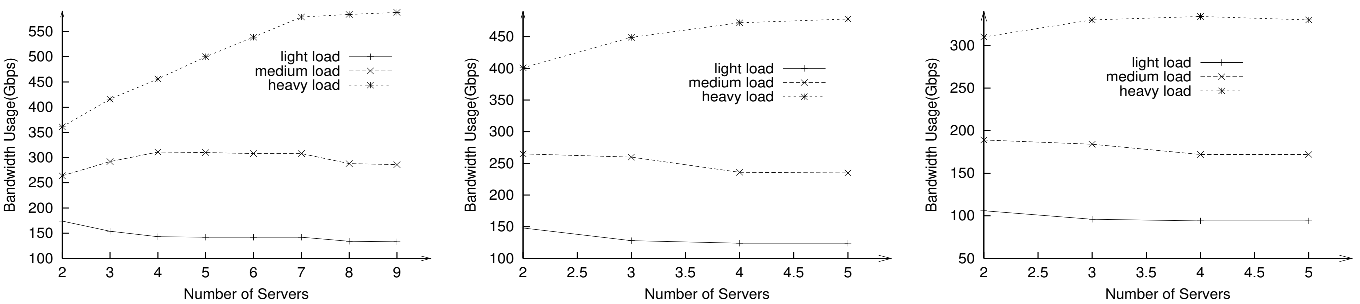
(a)Persistent Allocation (b) Video Patching (c) HMSM

Figure 5: Network Bandwidth Consumption



(a)Persistent Allocation (b) Video Patching (c) HMSM

Figure 6: User Access Latency Under Different Number of Servers



(a)Persistent Allocation (b) Video Patching (c) HMSM

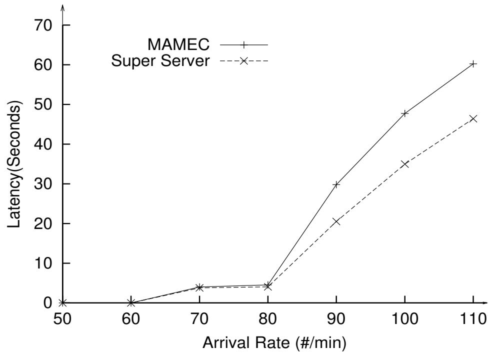
Figure 7: Network Bandwidth Consumption Under Different Number of Servers

channel merge rate. Higher channel merge rate will release more video channels at earlier stage, and thus start scheduling new request sooner. The MAMEC algorithm achieves the best user-perceived latency by efficiently merge the client requests.

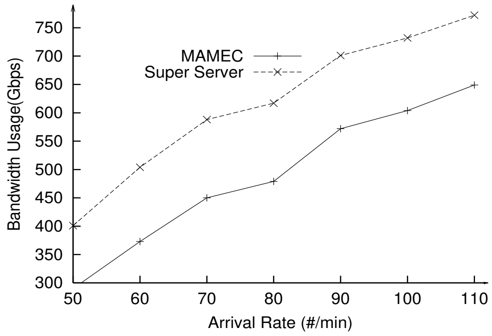
5.2 Network Bandwidth Consumption

Figure 5 shows the network bandwidth consumption under different schemes. The Closest server selection algorithm consumes least bandwidth, since the client always selects the nearest server, and these clients tend to share more common links. This algorithm also sacrifices access latency, thus creating larger batches, causing more shared links among the same channel. Other algorithms tend to consume more network bandwidth resources. These algorithms put higher priority on server batching state information than the client-server distance. The clients selecting the same server does not show high geographically clustering property. Since the client requests are more evenly distributed among all the VoD servers, this reduces the chance to form larger video batches on a single server, this also makes the number of shared links in the multicast tree decrease. There is a trade off between the network bandwidth consumption, and user perceived latency. Our *expected cost* approach achieves fairly good results in both categories.

From figure 5, we observe that the network bandwidth consumption increases as the client request rate gets higher. But network bandwidth consumption does not increase in the same speed of client request rate. Two reasons contribute to such property. First, as the client request rate increases, the user access delay also increases, in consequence, the average batch size for each video channel gets larger.

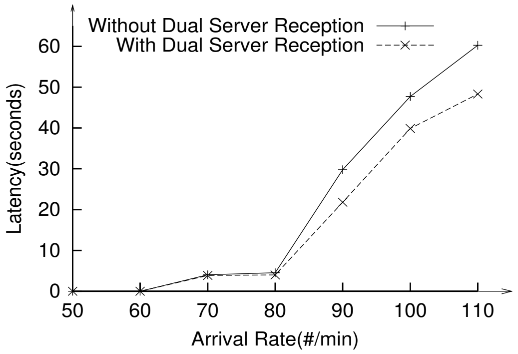


(a) Access Latency Comparison

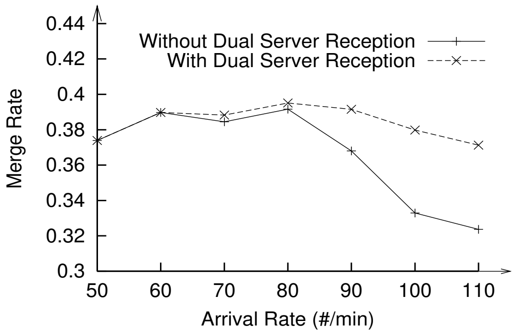


(b)Bandwidth Consumption Comparison

Figure 8: Super Server Vs. Replicated Server



(a) Influence on Access Latency



(b)Influence on Channel Merge Rate

Figure 9: Influence of Dual Server Reception

Thus, a multicast tree tend to have more shared links than the smaller multicast trees. Second, as the client request rate increases, clients close to each other have higher probability to be served by the same server, this further increase the degree of link sharing within the multicast tree.

We also notice that video patching and HMSM schemes consume less bandwidth resource than persistent channel allocation scheme. The reason is under video patching and HMSM, multicast groups can be merged during the video transmission process, creating larger multicast tree with more shared links.

5.3 Influence of Number of Servers

We study the performance as the number of servers varies. Figure 6 shows the influence of the number of servers on the user perceived latency. In these experiments, the client request rate is 100 requests per minute.

As the number of servers increases, the latency decreases rapidly. After the number of servers is above a certain value, adding more servers does not improve the performance significantly. The performance of the Closest server selection algorithm is not as sensitive to the number of servers as other server selection algorithms. The effect of increasing the number of servers on the access latency is not surprising, because adding servers increases service capacity. It is, nevertheless, important to confirm that the service replication and selection maintains the property that performance improves as more capacity is added. It is also worthwhile in determining when the preformance reaches a level of diminishing return.

To evaluate the influence of the number of servers on the bandwidth resource usage, we study the bandwidth usage behavior under three traffic loads: light (10 requests/min), medium (40 requests/min), and heavy (80 requests/min). Figure 7 shows the network bandwidth consumption when we use different number of servers. We show the result of MEC/MAMEC algorithms only. For the persistent allocation scheme, when the traffic load is light, as the number of servers increase, the total resource decreases. When the traffic load is medium, the network bandwidth consumption first increases, and then decreases. This is because as the number of servers increase, while the client can find a closer server, the load is distributed among the servers reducing the multicast efficiency; the influence of this efficiency becomes more apparent when the traffic load is heavy, the network bandwidth consumption keep increasing with the number of servers. For patching and HMSM schemes, efficient channel merging increases the advantage of multicast transmission, even under heavy traffic load. Hence the network resource consumption does not increase too much.

5.4 Comparison with a "Super" VoD Server

It should be clear that one can increase capacity without replication by simply adding power to a single server. In this section, we assume there is a "super" VoD server, which has the capacity of the sum of all the replicated servers. We run MEC/MAMEC algorithms on the replicated server environment, and compare the result with the super server environment. A super server provides a lower bound of on server selection algorithms in terms of access latency.

Figure 8 shows the performance of one super server versus replicated servers under the video patching scheme. Persistent allocation and HMSM have similar results.The super server provides modestly better access latency, When this server schedules a new request, it has more on-going regular channels broadcasting the same video program, thus the newly scheduled channel has a better chance to merge into other channels. On the other hand, the super server consumes significantly more bandwidth than the replicated server environment. Since all the clients receive the video streams from the same server, the average distance from the client to the server is larger than the replicated server case.

Based on the simulation result, we argue that increasing VoD service capacity through replication is preferable for the following reasons: First, replication allows shorter paths, thus improving overall bandwidth consumption. Second, the service capacity of a single server cannot be increased without bound as one ultimately would reach the limit of the server's access link bandwidth. Further, replicated VoD servers are more robust, flexible, and cost-effective.

5.5 Influence of Dual Server Reception

In the previous results, the client request is served by one VoD server. In some cases, however, a "patchable" channel maybe on another replicated server. In this case, if the client's request can be served by two servers, the channel merge rate can be improved, and the client access latency will be reduced. Dual server reception can only be used under video patching and HMSM schemes. In this section, we study the influence of dual server reception.

Figure 9 compares the simulation result with/without dual server reception under the video patching scheme. The HMSM case has very similar results. Figure 9 (a) compares user access latency using the MAMEC algorithm. When the client request rate is low, there is not too much difference, since the video servers are not fully loaded yet. As the client request rate increases, allowing dual server reception can achieve a higher channel merge rate by merging into video channels of other video servers. This in turn reduces the user access latency. Dual server reception can also reduce the requirement for client buffer size by merging into the closest channel of all video servers. We will study the influence of different merging selection algorithm on overall performance in future research.

6. IMPLEMENTATION ISSUES

We envision a system architecture similar to the one proposed in [23], and shown in Figure 10. Replicated VoD servers and clients are distributed over the network. There is a central directory server which stores each video server's state information, including channel status and batching status. *Proxy servers* placed in appropriate locations in the network are used to direct clients to "best" servers.

In such an architecture, clients first submit requests to the proxy servers (in a manner similar to DNS), proxy servers aggregate the client requests and forward the requests to the directory server. Directory server receives request from the proxy server, and returns server state information. The proxy server then makes a server selection decision and return the identity of the server(s) that this client should contact for service. The proxy servers are also responsible for video filtering. If two servers send the same video streams to the client, the proxy server should be able to drop one of the video stream and send a cancel signal back to that server.

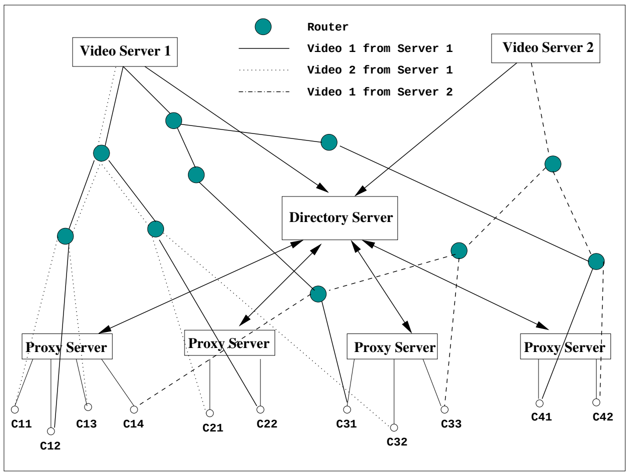


Figure 10: Replicated VoD Service Architecture

The proxy servers need to maintain the information needed to make server allocation decisions. The type of information needed varies for the various selection techniques we described. In the most general algorithms the state of each server's video channel as well as the state of its client batches need to be known at the proxy servers. Generally, there are two approaches to achieve this purpose: VoD server push and directory server pull. For server push, whenever the server state information is changed, it will push the new value to the directory server. This approach can guarantee that the information maintained at the directory server is always up to date when it is requested from a client. This approach may create useless traffic if the state information is updated more frequently than needed. For the pull approach, if the server state is changed, an invalidation message will be sent to the directory server. When the client request comes, the directory server will pull the state information from video servers if the server state is invalidated. This approach may cause extra delay since the client has to wait for the directory server to fetch the updated state information from the video server. A combined push/pull approach is a reasonable solution for server state information update.

In practice, it may not be feasible to keep complete and up-to-date information for all the servers at all the proxy servers. One way to reduce the overhead of maintaining this information at the proxy servers is to not insist on full accuracy and consistency. In our future research, we plan to consider the effect of using slightly out-of-date and possibly inaccurate information on the performance of the selection techniques. An important observation from our simulations is that while selection algorithms that take into account all information about servers perform best, other algorithms which use only limited information provide relatively good performance as well. This seems to indicate that selecting according to slightly out-of-date information may also perform reasonably well.

7. CONCLUDING REMARKS

In this paper, we consider the design of selection techniques for a set of replicated batching VoD servers. Our work explores the use of a range of selection techniques that use a varying amount of knowledge about the servers and their location. We show that, with the exception of the very naive Closest Server selection technique, server replication can indeed be used as a way to increase the capacity of the service leading to improved performance. As discussed in Section 4, using replication is a better approach to increase service capacity when compared to simply providing a more powerful single server. We have demonstrated here that one can indeed design server selection algorithms to make full use of the increased capacity provided by replication.

In our future research we plan to consider several issues relating to replication of batching VoD servers, include:

- Evaluation of the effect of out of date information on the performance of the selection techniques. We will design a protocol which exchange server state information as described above, and observe the overall performance.
- The use of partially replicated VoD servers where not all the videos are available at all the servers. The video request pattern is likely to follow the Zipf distribution law, where most of the video requests are focused on a small subset of hot videos. For unpopular videos, replication may cause performance degradation, with longer user perceived latency due to low MFQ values, and more bandwidth usage due to very small multicast groups. It is more reasonable to replicate video programs according to how popular they are.
- The use of heterogeneous replicated servers in which different servers may use different batching techniques. In the real world, it is highly possible that different VoD servers adopt different scheduling approaches. How to fit our server selection algorithms to such an environment is important for implementation considerations.

8. REFERENCES

[1] C.C. Aggarwal, J.L.Wolf, and P.S. Yu. On optimal batching policies for video-on-demand storage servers. In *Proceedings of the IEEE International Conference on Multimedia Systems*, June 1996.

[2] K. Almeroth and M.Ammar. On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service. In *Journal on Selected Areas of Communication*, August 1996.

[3] K. Calvert, M. Doar, and E. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, June 1997.

[4] R. L. Carter and M. E. Crovella. Server selection using dynamic path characterization in wide-area networks. In *Proceedings of IEEE INFOCOM*, 1997.

[5] S.W. Carter and D.D.E. Long. Improving Video-on-Demand Server Efficiency Through Stream Tapping. In *Proc. of 6th Int'l Conf. on Computer Communications and Networks*, September 1997.

[6] A.Dan, D.Sitaram, and P.Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proceedings of 2nd ACM Multimedia Conference*, Oct. 1994.

[7] H.D. Dykeman, M.H. Ammar, J.W. Wong. Scheduling algorithms for Videotex systems under Broadcast Delivery. In *Proceedings of ICC'86.*

[8] D. Eager and M. Vernon Dynamic skyscraper broadcasts for video-on-demand. In *Proc. 4th International Workshop on Multimedia Information Systems*, September 1998.

[9] D. Eager, M.Vernon, and J. Zahorjan, Optimal and Efficient Merging Schedules for Video-on-Demand Servers, In *Proc. 7th ACM Intl. Multimedia Conf. (ACM MULTIMEDIA 99)*, 1999.

[10] D.L. Eager, M.K. Vernon, and J. Zahorjan. Minimizing Bandwidth Requirements for On-Demand Data Delivery, In *IEEE Trans. on Knowledge and Data Engineering* Sept./Oct. 2001.

[11] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar. A novel server selection technique for improving the response time of a replicated service. In *Proceedings of IEEE INFOCOM*, 1998.

[12] Z. Fei, M. Ammar, E. Zegura. Optimal Allocation of Clients to Replicated Multicast Servers. In *Proceedings of the 1999 IEEE International Conference on Network Protocols*, November 1999.

[13] Z. Fei, M. Ammar, E. Zegura. Efficient Server Replication and Client Re-Direction for Multicast Services. In *Proceedings of SPIE/ITCOM Conference on Scalability and Traffic Control in IP Networks*, August 2001.

[14] Z. Fei, M. Ammar, E. Zegura. Multicast Server Selection: Problems, Complexity and Solutions. To appear in *IEEE Journal on Selected Areas in Communication.*

[15] L. Gao, D. Towsley. Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast In *Proceedings of IEEE Multimedia Computing Systems*, June 1999.

[16] L. Gao, D. Towsley. Threshold-Based Multicast for Continuous Media Delivery In *IEEE Transactions on Multimedia*, Dec. 2001

[17] L. Golubchik, J.C.S. Lui, and R. Muntz. Reducing I/O demand in Video-On-Demand storage servers. In *Proc. 1995 ACM Sigmetrics Join Int'l Conf. on Measurement and Modeling of Computer Systems*, May 1995.

[18] J. Guyton and M. Schwartz. Locating nearby copies of replicated Internet servers. In *Proceedings of ACM SIGCOMM*, 1995.

[19] K.A. Hua, S. Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, 1997.

[20] K.A. Hua, Y. Cai, S. Sheu. Patching: a multicast technique for true video-on-demand services . In *Proceedings of the sixth ACM international conference on Multimedia*, 1998.

[21] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service location protocol. *RFC 2165*, June 1997.

[22] S.Viswanathan , T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. In *Multimedia Systems*, August, 1996.

[23] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: A server selection architecture and use in a replicated web service. *IEEE/ACM Transactions on Networking*, August 2000.