Optimised Batch Patching with Classes of Service

Paul P. White
Visitor
Dept. Computer Science, UCL
Gower St.,London WC1E 6BT, UK
Tel: +44 7679 2000
p.white@cs.ucl.ac.uk

Jon Crowcroft
Professor
Dept. Computer Science, UCL
Gower St.,London WC1E 6BT, UK
Tel: +44 7679 7296
j.crowcroft@cs.ucl.ac.uk

ABSTRACT

In this paper we present a new technique called Optimised Batch Patching with Classes of Service(OBP with CoS) which can be used to leverage the benefits of multicast within the context of 'near' video-on-demand systems in IP networks. OBP with CoS builds on an earlier scheme, known as Optimised Patching, but is different in two respects. Firstly, in OBP with CoS, the server artificially delays requests in order to increase the probability of accumulating duplicates, thereby allowing greater exploitation of multicast compared to Optimised Patching albeit at the expense of higher latency until commencement of service. Secondly, the client is able to request a latency class of service which reflects how long the client is prepared to wait until playout of the video. The server scheduling algorithm attempts to minimise network bandwidth consumption within the constraints imposed by the class of each request. We present analysis and simulations of our scheme in order to validate its effectiveness.

Keywords

Multicast, video-on-demand, multimedia.

1. INTRODUCTION

Today IP networks are being used for many applications and services beyond conventional data transfer. One example is Video-On-Demand(VOD). VOD describes a system whereby a user is able to request any video from a library stored on one or more servers. In response to the request, the server will deliver the video as an isochronous stream for playout by the user. With 'true' VOD the delay between the user submitting a request and commencement of playout of the requested video should be small, typically a few seconds. An alternative is what could be referred to as 'near' VOD whereby the delay to commencement of playout is significant, typically a few minutes, but still acceptable to the user under certain conditions. For example, in the case of movies-on-demand, where the playout duration of a movie is say 90 minutes, a delay of several minutes might be acceptable to the user provided there is an appropriate cost reduction compared to 'true' VOD.

A major landmark in the recent evolution of IP networks is the introduction of multicast routing and forwarding techniques for group communication. When the same information needs to be sent to multiple recipients, a single multicast transmission using a protocol such as Protocol Independent Multicast(PIM)[4] will consume less network bandwidth and impose less of a load on the sender compared to a separate unicast transmission to each recipient.

Although many schemes have been proposed that attempt to exploit the benefits of multicast for VOD most of them are based on 'periodic broadcast' principles[1][3][5][7] whereby each video is split into portions which are continuously multicast over a number of multicast channels and as such are only bandwidth efficient provided request arrival rate is high.

A multicast scheme that is bandwidth-efficient over a wider range of request arrival rates is that of Patching[6] which facilitates 'true' VOD as well as 'near' VOD.

In the case of near VOD, each request incurs a delay at the server prior to commencement of service. In [6] this delay is only ever caused by bandwidth limitations on the server output link although, as we will explain later, an alternative is for delay to be artificially imposed by the server scheduling algorithm in order to increase the likelihood of obtaining duplicate requests thereby allowing further exploitation of multicast.

By the time service of a delayed request is about to commence the server may have accumulated additional requests for the same video. Each such group of identical requests is known as a batch. Following the service point of each request batch as determined by the server scheduling algorithm, the batch must then be served over 1 or 2 channels, that is multicast groups, either a so-called regular channel alone, or the combination of a regular channel and a so-called patching channel. A regular channel delivers a full video from start to finish while a patching channel delivers only the missing part of the video from the start until the point at which the clients of a batch join the regular channel. In [6] the multicast groups were realised at the application layer in order to minimise congestion on the internal server bus. However the technique would also work using network layer multicast which would result in network bandwidth savings in a multicast-capable IP network.

In [6], the preferred method of patching, known as grace patching, works as follows. At the service point of a batch, the server identifies the newest regular channel that is serving the requested video and equates the channel's age to a corresponding buffer space, e.g. to store 10 minutes of Mpeg video requires approximately 100 Mbytes of buffer space. If the corresponding buffer space exceeds the buffer capabilities of the clients in the batch then patching cannot be used for the batch in which case a new regular channel must be opened instead and its identity conveyed to the clients in the batch. Each client then joins the regular channel in order to receive the entire video.

Assuming the buffer space of the clients is sufficient then patching can be used as follows. The server opens a new channel known as a 'patching' channel which will deliver only the early part of the requested video that the clients will miss upon joining the video's newest regular channel. The server then conveys the identity of both patching channel and newest regular channel to each client in the batch. Each of these clients then joins both channels and buffers the regular channel while playing out the patching channel. When the patching channel has been

exhausted each client then switches to playout of the buffered regular channel.

For 'true' VOD the delay of each request at the server prior to commencement of service must be negligible which means there is little or no time for the server to accumulate duplicates before servicing a given request. In this case each batch would contain just a single request and any patches could be delivered via unicast.

The 'batch and multicast' approach to delivery of a patch as used by near VOD results in a lower incremental bandwidth consumption per request compared to true VOD albeit at the expense of higher latency until start of transmission as perceived by the user.

[6] compared the performance of a VOD multicast system using patching with one that always started a new transmission of the full video for each batch of requests, and found significant performance improvements in the patching case. For example, [6] found that compared to a no-patching approach, patching was able to support 'true' VOD at 4 times the request rate for a typical server configuration.

Apart from the number of requests in each batch, with a patching scheme there is an additional factor that determines bandwidth efficiency, namely whether a batch is served via a new regular multicast or the combination of a patch and an existing regular multicast. The scheme presented in [6] bases this decision entirely on the buffering capabilities of the clients. In other words, to serve a batch [6] always uses a patch provided each client in the batch has sufficient buffering capabilities. This strategy is not optimal with regard to bandwidth consumption since the amount of traffic contained in a patch increases as the age of the latest regular multicast increases. Once the regular multicast reaches a certain age it becomes more efficient to start a new regular multicast rather than to continue patching to the existing latest regular multicast.

[2] presents a proof of this for true video-on-demand patching and refers to the scheme as Optimised Patching. The time interval following commencement of a regular multicast during which it is more efficient to patch to that regular multicast than begin a new regular multicast is known as the Patching Window. The Patching Window therefore defines the minimum time interval between successive regular multicasts of the same video.

In [2], service of any pending batch commenced as soon as a free channel became available on the server output link. Such a scheme is biased towards minimising latency to commencement of service which is obviously desirable from a user's point of view.

An alternative approach would be to minimise bandwidth consumption on the server output link which in turn would minimise network load and as such would be desirable from a network provider's point of view. Network bandwidth consumption can be minimised if the server delays commencement of service of a batch beyond the point at which a free channel first becomes available. In other words the server artificially delays service of each request in order to increase the number of requests contained in each batch and as a result the number of requests satisfied by any resultant patch. In what we refer to as a 'simple batching scheme' the server aggregates duplicate requests over a fixed batching interval before commencing service of each batch of duplicates at the end of the interval.

With a simple batching scheme the batching interval is equal to what we refer to as the maximum holding time(MHT) of a request which is the amount of time that a request may be artificially held for by the server before commencing service. The higher the MHT of each request, the greater the probability that a request matches on a duplicate before it is serviced. Since accumulated duplicate requests can be served via a single multicast patch rather than a number of unicast patches, increasing the MHT will increase the bandwidth saving. It is therefore reasonable to expect the network provider to charge more for providing a lower MHT. Consequently there is a tradeoff between latency and cost.

A simple batching scheme applies the same MHT to each request for a given file and as a result is unable to achieve an optimal latency/cost tradeoff for each client. For some clients the MHT or associated cost might be more than they are willing to tolerate.

A more efficient approach would support heterogeneity among the MHTs of different requests for the same video. The MHT of a given request could then be set by the client according to their desired latency/cost tradeoff.

In this paper we present a scheme known as Optimised Batch Patching with Classes of Service(OBP with CoS) which combines the concepts of Optimised Patching, simple batching and client-control of the latency/cost tradeoff.

In order to present our work in the most logical fashion we build it up from its core components and assess it at each stage. We begin in section 2 by describing the basic technique of Batch Patching in more detail. Then in section 3 we derive equations that can be used by a Batch Patching server to calculate the optimal patch window size for a given request arrival rate, video run time and batching interval. We call the resultant scheme Optimised Batch Patching(OBP) and validate our analysis using simulation.

In section 4 we extend OBP so that each client can specify a latency class which reflects a MHT in accordance with their desired latency/cost tradeoff. We call the resultant scheme OBP with Classes of Service(OBP with CoS) and present simulation results in order to illustrate its behaviour and performance.

We also discuss how the analysis results of classless OBP could be used to estimate the optimal window size for the more general case of OBP with CoS .

2. BATCH PATCHING

With Batch Patching the server divides time into epochs of duration b seconds. During each epoch the server accumulates requests and stores them in batches of duplicates. At each epoch boundary the server commences service of each batch. Any transmissions will be done at the same rate in terms of frames/s as the actual playout rate of the video itself. We now analyse such a scheme for a server hosting a single video file and assuming that each client has an identical amount of buffer space equal to B bytes. We also assume that the server always allows kb seconds

(k<1) for clients to join any multicast channel that they are instructed to join by the server. The timing diagram for the scheme is shown in Figure 2-1 and will now be explained in detail.

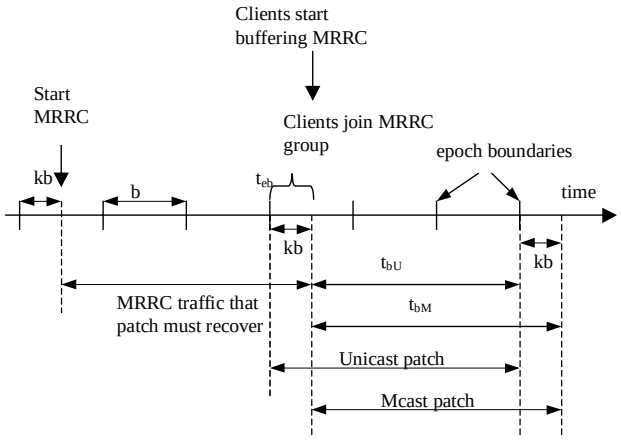


Figure 2-1: Timing Diagram for Classless Batch Patching

We let t_{eb} denote the time at an epoch boundary where a non-empty batch exists. Here a decision must be made as to the manner in which service of the batch should occur, namely either a new regular multicast or a patch to an existing regular multicast.

Should the server choose to start a new regular multicast, it will inform each client in the batch of the multicast address of the new regular channel. The server will then wait kb seconds before commencing transmission of the full video over the new regular channel.

Should the server choose to deliver a patch then it will inform each client in the batch of the multicast address of the Most Recent Regular Channel(MRRC) and use a patch to recover the early part of the file that has already been transmitted over the MRRC. If the number of clients in the batch exceeds 1 then the patch will be delivered by multicast in which case the clients will also be informed of the multicast address of the patching channel. In the case of a batch containing a single client the patch will simply be delivered via unicast.

Although the server begins informing clients of the multicast address of the MRRC at time=teb, the time could be as late as teb+kb before each client has joined the channel and begun receiving traffic on it. This means that the patch is responsible for recovering all packets that were transmitted over the MRRC before teb+kb. Assuming that the patch is transmitted at the same[1] rate as the regular channel then the duration, dpatch of the patch will be given by

$$\text{dpatch} = (t_{eb} + \text{kb}) - (\text{start time of MRRC}) \quad \textbf{(1)}$$

[1] Same with regard to frames/s. The rate(frames/s) of a regular channel is the same as the playout rate of the video itself. Transmitting the patch at this same rate will require the least amount of client buffering.

We assume that the server commences any unicast patch at time=t_{eb} but delays commencement of any multicast patch until time=t_{eb}+kb. Transmission of the patch will then cease at t_{eb}+dpatch in the case of a unicast patch and t_{eb}+kb+dpatch in the case of a multicast patch.

Although a client will join the MRRC and begin receiving packets over it at any time between t_{eb} and t_{eb}+kb, any packets received over the MRRC before t_{eb}+kb are discardable since such packets will be recovered by the patch anyway. Once the patch finishes, playout of the buffered MRRC channel will begin at the client and lag network transmission of the MRRC by dpatch seconds in the case of a multicast patch and dpatch-kb in the case of a unicast patch. The time-transformed buffer requirements at the client in the two cases are then as follows

$$t_{bM} = \text{dpatch} \quad \textbf{(2)}$$

$$t_{bU} = \text{dpatch} - \text{kb} \quad \textbf{(3)}$$

where t_{bM} and t_{bU} are the time transformed buffer requirements for the multicast patch and unicast patch cases respectively. The actual buffer requirements, B_M and B_U bytes, for a multicast and unicast patch respectively will then be as follows:

$$B_M = dpatch * rate(t_{bM}) \quad \textbf{(4)}$$

$$B_U = (dpatch - kb) * rate(t_{bU}) \quad \textbf{(5)}$$

where rate(t_b) bytes/s is the maximum of the mean rate of the video over any interval equal to t_b. t_b can be any value {b, 2b, 3b…} in the case of a multicast patch and any value {(1-k)b, (2-k)b, (3-k)b…} in the case of a unicast patch. The larger the value of t_b the smaller the value of rate(t_b). However it is unrealistic to assume that the value of rate would be made available to the server for every value of t_b. It is more likely that only the worst case values of rate, denoted $rate_1$ and $rate_{(1-k)}$ for t_b = 1 and (1-k) epochs respectively would be made available to the server. These values could be measured for each video and stored at the server. Rearranging equations (4) and (5) and inserting the worst case values of rate gives the following

$$dpatch_M < \frac{B}{rate_1} \quad \textbf{(6)}$$

$$dpatch_U < \frac{B}{rate_{(1-k)}} + kb \quad \textbf{(7)}$$

where dpatch_M and dpatch_U are the maximum duration multicast and unicast patches that a client buffer of size B can accommodate. If a potential patch violates equation (6) in the case of multicast or equation (7) in the case of unicast then the server is precluded from sending it and must instead begin a new regular multicast in the next epoch.

3. OPTIMISED BATCH PATCHING(OBP)

Equations (6) and (7) impose an upper limit on the duration of a patch in accordance with the client buffer capabilities. If the value of B is large, then allowing a patch duration as large as that determined by equations (6) and (7) will not yield the maximum bandwidth efficiency. This was discussed briefly in section 1. Instead, higher bandwidth efficiency will be achieved by limiting the patch duration to some lower value known as the Patching Window size which we denote as W. We call such an approach Optimised Batch Patching(OBP). With OBP, the duration, dpatch of any patch, whether unicast or multicast, must satisfy the condition of equation (8) as well as (6) and (7).

$$dpatch \le W \tag{8}$$

We will now derive an equation that relates the bandwidth consumption of Optimised Batch Patching with the parameters b, W, request arrival rate, λ, and mean video rate, r. The derived equation will allow us to calculate the optimal value of W for specific values of b, λ and r. In the following analysis we consider a single video file and assume that clients have ample buffer space for all values of dpatch. In other words we assume that equations (6) and (7) are never violated.

We refer to an RM-epoch as one in which a regular multicast was started and a non-RM-epoch as one in which a regular multicast did not begin. In order to calculate the mean transmission rate on the server output link we need to determine both the mean interval between RM epochs and the mean of the aggregate number of bytes contained in all patches commencing between two adjacent RM epochs chosen at random.

Following an RM epoch, the next RM epoch will be triggered by the next occurrence of a non-empty batch whose associated value of dpatch exceeds W and so violates equation (8). Now the value of dpatch for a patch commencing in non-RM epoch number *i* is given by the following

$$dpatch = bi \tag{9}$$

where non-RM epoch number 1 is the first epoch following an RM epoch. From equation (9) it can be seen that provided $i \le W/b$ then equation (8) will not be violated. Hence the first W/b epochs following an RM epoch will definitely be non-RM epochs. Following the first *W/b* non-RM epochs the next non-empty batch to occur will definitely violate equation (8) and hence cause the next epoch thereafter to be a RM epoch.

These observations yield the following equation for the mean number, n of epochs between two adjacent RM epochs

$$n = \text{int}\left(\frac{W}{b}\right) + \left(\sum_{i=1}^{\infty} i\,P(0)^i\left(1-P(0)\right)\right) \tag{10}$$

where P(i) is the probability of the number of requests in a batch being equal to i. Assuming a Poisson arrival process, P(0), which we denote simply as P for ease of representation is given by

$$P = P(0) = e^{-\lambda} \tag{11}$$

where λ is the mean number of requests per epoch. Returning now to (10) let us examine the special case where W is an integer multiple of b. Equation (10) then becomes

$$n = \frac{W}{b} + \sum_{i=1}^{\infty} i\,P^i\left(1-P\right) \tag{12}$$

Expanding the second term on the right hand side of equation (12) we have

$$\sum_{i=1}^{\infty} i\,P^i(1-P) = P + P^2 + P^3 + P^4 + P^5 + \ldots \tag{13}$$

Substituting for the geometric series on the right hand side of equation (13) we have

$$\sum_{i=1}^{\infty} i\,P^i(1-P) = \frac{P}{(1-P)} \tag{14}$$

Substituting equation (14) into (12) we have

$$n = \frac{W}{b} + \frac{P}{(1-P)} = \frac{W(1-P)+bP}{b(1-P)} \tag{15}$$

Now only the first W/b epochs following an RM epoch are capable of generating a patch. For each of these epochs the probability of generation of a patch will be given by 1-P. Hence the mean, μ of the aggregate duration of all patches commencing between two adjacent RM epochs chosen at random is given by

$$\mu = b\sum_{i=1}^{\frac{W}{b}} i(1-P) \tag{16}$$

Solving the summation in equation (16) gives the following

$$\mu = b(1-P)\left(\frac{W}{b}+1\right)\left(\frac{W}{2b}\right) = \frac{(1-P)W^2}{2b} + \frac{(1-P)W}{2} \tag{17}$$

The number of bytes, α corresponding to μ is given by the following

$$\alpha = \frac{(1-P)rW^2}{2b} + \frac{(1-P)rW}{2} \tag{18}$$

where r is the average rate in bytes/s of a single transmission of the video. To obtain an estimate of the mean transmission rate on the server output link we need to consider an RM epoch and the subsequent run of non-RM epochs before the next RM epoch. The mean time period, t of such a sequence is given by

$$t = nb + b = b(1+n) \tag{19}$$

And the number of bytes, β in all transmissions commencing in this period is given by

$$\beta = \alpha + Tr$$

(20)

where T is the total run time of the video. The average transmission rate, R on the server output link can now be written as follows

$$R = \frac{\alpha + Tr}{b(1+n)} \qquad (b < W < T)$$

(21)

Substituting equations (15) and (18) into equation (21) and simplifying gives

$$R = \frac{(1-P)rW^2 + (1-P)brW + 2rbT}{2bW + \frac{2b^2}{(1-P)}}$$

(22)

For W>0, equation (22) has a single minimum which can be found by differentiating R and setting the result equal to 0. This yields the following equation

$$W = \frac{-b + \sqrt{Pb^2 + 2(1-P)bT}}{b(1-P)}$$

(23)

The result in equation (23) gives the positive value of W that yields the minimum of equation (22) which is continuous in time. In deriving (22) we assumed that W was an integer multiple of b. Hence for consistency the true solution, that is the value of W that is an integer multiple of b and which gives the minimum in R is given by equation (24)

$$W = b\,\mathrm{int}\left(\frac{-b + \sqrt{Pb^2 + 2(1-P)bT}}{b(1-P)} + \frac{1}{2}\right)$$

(24)

Equation (24) is an important result that can be exploited by the server to optimise bandwidth usage of classless patching. In addition it can be used to test buffering capabilities of clients and price them accordingly if limitations in their buffering capabilities prevent achievement of optimal bandwidth usage.

Figure 3-1 plots equation (22) for a video of duration 90 minutes with b set to 1 minute at different values of λ, the mean number of requests per batching interval.

Figure 3-2 plots curves for the Optimised Patching scheme of [2] which produces true VOD. The curves in Figure 3-2 were obtained from the following equation[2] which was derived in [2] and gives the normalised transmission rate for Optimised Batching as a function of video run time, T, patching window size, W and arrival rate, λ.

$$\frac{R}{r} = \frac{2T\lambda + W(W+1)}{2W\lambda + 2}$$

(25)

For the λ=1 case the corresponding simulation results are also shown. The close match between simulation and analysis for the classless OBP case provides a strong degree of confidence in both our analysis and simulation model for the classless OBP case. The same simulation model will later be used to analyse OBP with CoS.

In Figure 3-1 the minima of the curves for λ equal to ∞, 2, 1, 0.5 and 0.25 are located at W equal to 12, 13, 15, 19 and 24 minutes respectively. Using this information the server could continuously optimise its value of W to reflect the current arrival rate, λ. Alternatively, in order to simplify implementation of the server the value of W could be set to a fixed value that was close to optimal across a wide range of arrival rates. For example in Figure 3-1 with W equal to 17 minutes the value of R/r on each curve is no more than 5% greater than its minimum value for that curve.

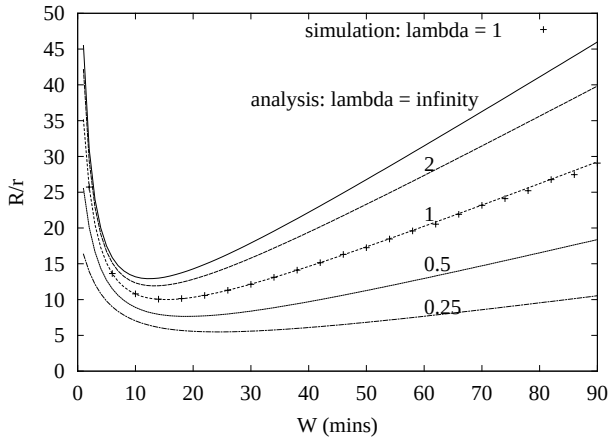
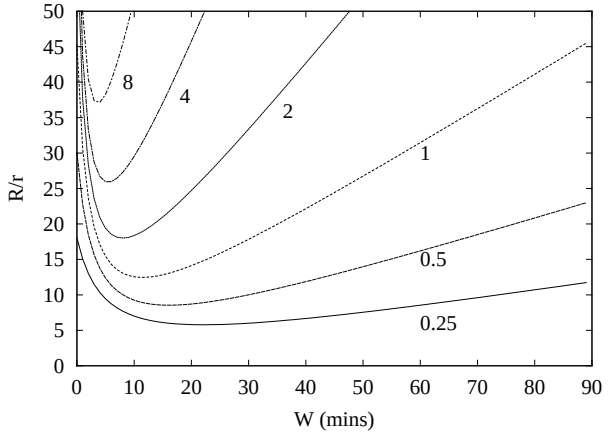


Figure 3-1: Normalised Transmission Rate vs W at Different Values of λ(requests/min) for Optimised Batch Patching of a 90 Minute Video with b = 1 minute



[2] In equation (25) the units of W and T are seconds while those of λ are requests/second. These parameters were scaled to minutes and requests/min respectively in order to produce the plots in Figure 3-2.

Figure 3-2: Normalised Transmission rate vs W(mins) at Different Values of λ(requests/min) for Optimised Patching of a 90 Minute Video with b = 1 minute

Comparision of Figure 3-1 and Figure 3-2 clearly illustrates the bandwidth savings of Optimised Batch Patching compared to Optimised Patching, particularly for larger values of λ. In the Batch Patching plots of Figure 3-1, with optimal setting of W the value of R/r can be kept below 13 no matter how high the arrival rate. By contrast in the Patching plots of Figure 3-2, the value of R/r rises rapidly with λ and at a value of λ=8 requests/epoch optimal setting of W gives a high value of R/r of 37.

4. OPTIMISED BATCH PATCHING WITH CLASSES OF SERVICE(OBP WITH COS)

In the previous section we showed that for a given arrival rate, video run time, and epoch period b, a specific value of W exists that yields a minimum in the bandwidth consumption on the server output link in the case of classless OBP. We now extend the basic principles of OBP to allow a latency class to be associated with each request. We call this extended scheme OBP with CoS(Classes of Service). Here the latency class indicates the maximum number of epochs that the server is allowed to hold the request for before making a final service decision.

Table 4-1: Request Parameters for OBP with CoS

Request Parameters
VideoFileName
B (buffer size)
Class
MaxPrice
Type

Table 4-1 shows that each client request for a specific video file also indicates the client's current receiver buffer setting together with desired latency class. The class determines the cost of receiving a specific video. In addition if the value of B is insufficient to permit optimised batch patching then a cost penalty may be incurred as discussed in more detail in section 4.1.4. The MaxPrice parameter indicates the maximum price the user is willing to pay for fulfillment of the request.

4.1.1 OBP with CoS Request Type

Each OBP with CoS request includes a type parameter which indicates whether the request is tentative or final. Only final requests with a MaxPrice value less than or equal to the actual cost are submitted to the service scheduler at the server. Any final request where this is not the case and any tentative request will trigger a response containing a buffer/class price matrix such as that shown in Table 4-2.

Class	Buffer Space			
	100M	400M	700M	1G
1	£4.00	£3.40	£2.70	£2.00
2	£3.20	£2.70	£2.10	£1.60
3	£2.60	£2.10	£1.70	£1.30
4	£2.00	£1.70	£1.35	£1.00

Table 4-2: Example buffer/class matrix returned in response to a tentative request.

If returned in response to a tentative request the buffer/class matrix indicates what the cost of fulfilling the tentative request would be should it be resent with type set to final. The buffer/class matrix may also contain additional pricing information indicating how the price would change if the request was resent with class and/or buffer size set to different values. The buffer/class matrix could be embedded inside a response containing client-side interactive functionality, e.g. a java applet or a page of javascript code. The initial settings contained in the tentative request could be default values that were configured at the client. Upon receiving the response the user would have the option of altering the request settings at the client side and viewing the corresponding cost before dispatching the request as type 'final' for processing by the server scheduling algorithm. The MaxPrice parameter in the final request would reflect the cost indicated to the user for the final settings.

4.1.2 OBP with CoS Service Scheduling Algorithm

Table 4-3 shows the instance variables of the batch object which stores information about a batch of requests for the same file. B is the minimum buffer size of all requests in the batch and class is the minimum class of all requests in the batch.

Table 4-3: Instance variables of the batch object

Batch Object Variables
VideoFileName
B (buffer size)
Count
Class

Figure 4-1 shows the service scheduling algorithm that is applied at each epoch boundary. First each new request that arrived in the previous epoch is added to the corresponding batch and the batch parameters updated. Then all mature batches, that is those of class=1, are scheduled for service. For all remaining batches the class of each request is decremented and the class of the batch object decremented to reflect this.. Figure 4-1 assumes that the server is dimensioned in terms of link bandwidth and multicast addresses such that sufficient resources exist to commence each multicast transmission dictated by the server scheduling mechanism and patching technique.

```
For each new request, r {
        batch = getBatch(r.videoFileName);
        If (r.B < batch.B) {
                batch.B=r.B;
                batch.count++;
        }
        if (r.class<batch.class) {
                batch.class=r.class;
        }
}

For each batch {
        if (batch.class>1)
           batch.class --;
        else {
           calculate dpatch;
           calculate D;
           Breq←BufferReq(batch)
           if ( dpatch>D ) | ( Breq>batch.B )
              start new MRRC in kb seconds
           else
              patch(batch);
        }
}

function Patch(batch) {
        if (batch.count>1)
                Start Multicast of patch in kb seconds
        else
                Start unicast patch immediately
}

function BufferReq(batch) {
        if (batch.count==1)
             Breq=B_M
        else
             Breq=B_U
        return Breq
}
```

Figure 4-1: Service Scheduling Algorithm for OBP with CoS

4.1.3 Simulation of OBP with CoS

We use the notation *(share1:share2:share3:share4)* to denote the expected proportion of requests taken up by each request class according to their probabilities. For example (1:1:1:1) indicates that a given request could be any class between 1 and 4 inclusive with an equal probability. (1:0:0:0) indicates that all requests are of class 1.

In Figure 4-2 we plot our simulation results for a 90 minute video using OBP with CoS and different CoS ratios(λ=1). As can be seen Figure 4-2 the higher the proportion of lower latency class requests the higher the bandwidth consumption. Comparing (1:1:1:1) to (1:0:0:0) is equivalent to comparing OBP with CoS to classless OBP in an environment where each class of request is equiprobable. The curve of Optimised Batching is equivalent to all requests being of latency class 0 (no artificially imposed server-side latency) and consequently has the highest bandwidth consumption of all the curves. Hence the plots clearly demonstrate the bandwidth savings attainable using OBP with CoS compared to Optimised Patching. The bandwidth savings of OBP with CoS compared to Batch Patching will become even higher as λ increases.

For OBP with CoS the server could set the value of W for a video to the optimal value for classless OBP as given by equation (24). In the case of Figure 3-1 this setting of W is 15 and results in a value of R/r on each curve within 6% of the minimum value for that curve.

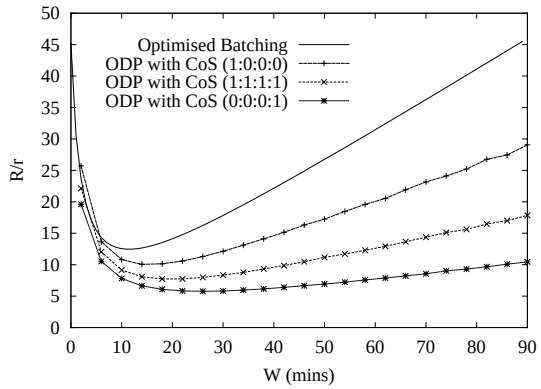


Figure 4-2: Simulation Results of Normalised Transmission Rate vs W for a 90 Minute Vdeo using OBP with CoS and different CoS ratios(λ=1, b=1 minute)

4.1.4 Handling and Costing of Requests

Specification of a lower latency class increases both the QoS experienced by the user and the network bandwidth consumed. Consequently it is reasonable to expect the end-user to be charged more for a lower latency class than a higher latency one in order to compensate the network provider for the associated increase in bandwidth consumption. In addition if the value of B in a client's request is insufficient to permit optimised batch patching then a cost penalty may be added by the server. For this purpose the server would check for fulfillment of equation (6) with dpatchM set to the projected value of dpatch for a final service decision made at the latest[3] possible epoch boundary for the class of the request.

In the case of VOD one could imagine at least two likely pricing structures. Firstly pricing could be done in accordance with the 4-tuple (popularity, running time, latency class, buffer cost penalty). Less popular videos would incur a higher cost since the incremental bandwidth consumption per request is higher for a multicast tree with fewer receivers. A second pricing structure one could imagine would simply be according to the 2-tuple (latency class, buffer cost penalty). This is workable in the case of a video server delivering feature films which are likely to be of similar length. The server could monitor the popularity of each video and remove from its archive those videos that were not sufficiently popular and consequently had poor earnings/overhead ratios. The overhead of a video comprises storage overhead at the server as well as bandwidth consumption in the network.

[3] A final service decision on a request of class n must be made by the nth subsequent epoch boundary at the latest.

5. SUMMARY

In this paper we have presented our scheme called Optimised batch Patching with Classes of Service(OBP with CoS) which provides a very efficient delivery mechanism for 'near' VOD. OBP with CoS extends the Optimised Patching scheme presented in [2] in two ways. Firstly, in OBP with CoS, the server artificially delays requests in order to increase the probability of accumulating duplicates, thereby allowing greater exploitation of multicast compared to Optimised Patching albeit at the expense of higher latency until commencement of service. Secondly, the client is able to request a latency class of service in accordance with their desired latency/cost tradeoff.

In the case of classless OBP we derived equations that can be used by the server to calculate optimal maximum patch duration for a given request arrival rate, video run time and epoch period. Moreover we validated this analysis using simulation. In addition we discussed how the analysis results could be used to estimate the optimal patching window size for the more general case of OBP with CoS. Following this we simulated OBP with CoS in order to illustrate the attainable bandwidth savings.

We expect the demand for QoS-enabled on-demand multicast services in the Internet to increase in the future for the following reasons

1) Collaboration between content and network providers

There is a trend within the communications industry towards increased collaboration between content and network providers as exemplified by the recent Time Warner/AOL merger. For content providers, networks provide a means to increase availability of their content while for network providers, content represents added value to their network. In order to satisfy a high customer demand for content within the constraints of available network bandwidth, on-demand multicast techniques will be needed together with a suitable QoS framework to ensure that end-users QoS expectations are met.

2) Increased bandwidth to the end-user

Many services suitable for on-demand multicast services such as VOD require a bandwidth in excess of that currently available to the typical home user in most countries, including the UK. However this will almost certainly change over the next decade as adoption of new technologies such as ADSL will make high bandwidth domestic access to IP networks commonplace.

3) Popularisation of the Set-top box

Digital TV transmission and acceptance of the set-top box in homes will remove the distinction between TV and data distribution and create a domestic environment that caters for the interactive needs of end-users.

6. ACKNOWLEDGEMENTS

The authors would like to thank British Telecom Labs, Ipswich, England for supporting this work, especially Alan O'Neill.

7. REFERENCES

[1] C.C. Aggarwal, J.L. Wolf and P.S. Yu. "A Permutation-Based Pyramid Broadcasting Scheme for Metropolitan VOD Systems". *Proc. of the IEEE International Conference on Multimedia Systems '96*, Hiroshima, Japan, June 1996.).

[2] Y. Cai, K. Hua and K. Vu. Optimizing Patching Performance, Proc. ACM/SPIE Multimedia Computing and Networking, January 1999, pp. 204-215

[3] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. *Proc. of ACM Multimedia*, pages 15-23, San Francsisco, California, October 1994

[4] Estrin et al. Protocol Independent Multicast(PIM-SM), RFC2362, June 1998

[5] K.A. Hua and S.Sheu. Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan VOD Systems. *Proc. of the ACM SIGCOMM'97*, Cannes, France, September 1997.

[6] Kien A. Hua, Y. Cai and S. Sheu, Patching: A Multicast Technique for True On-Demand Services, ACM Multimedia'98 Proc., September 1998, pp. 191-200

[7] S. Viswanathan and T. Imielinski. Metropolitan Area Video-On-Demand Service using Pyramid Broadcasting. *Multimedia Systems*, 4(4):179-208, August 1996.