Real-Time Traffic Transmissions Over the Internet

Marco Furini and Donald F. Towsley, Fellow, IEEE

Abstract—Multimedia applications require the transmission of real-time streams over a network. These streams often exhibit variable bandwidth requirements, and require high bandwidths and guarantees from the network. This creates problems when such streams are delivered over the Internet. To solve these problems, recently, a small set of differentiated services has been introduced. Among these, Premium Service is suitable for transmitting real-time stored stream (full knowledge of the stream characteristics). It uses a bandwidth allocation mechanism (BAM) based on the stream peak rate. Due to the variable bandwidth requirement, the peak rate BAM can waste large amount of bandwidth. In this paper we propose a new BAM that uses less bandwidth than the peak rate BAM, while providing the same service. Our BAM does not affect the real-time stream quality of service (QoS) and does not require any modification to the Premium Service Architecture. We also introduce several frame dropping mechanisms that further reduce bandwidth consumption subject to a QoS constraint when coupled with the above BAM. The proposed BAM and the dropping mechanisms are evaluated using Motion JPEG and MPEG videos and are shown to be effective in reducing bandwidth requirements. Further, since VCR operations are very useful in video streaming, we propose a mechanism that introduces these operations in our BAM. Through simulations we show the effectiveness of this mechanism.

Index Terms—Bandwidth allocation mechanism, frame dropping mechanisms, multimedia traffic management.

I. INTRODUCTION

PAY PER VIEW movies, distance learning, and digital libraries are examples of multimedia applications that require the transmission of real-time streams over a network. Such streams (such as video) can exhibit significant bit rate variability [8], depending on the encoding system used, and can require high network bandwidth. Moreover these streams require performance guarantees from the network, such as guaranteed bandwidth and loss rate, and this poses significant problems when such streams are delivered over the Internet. In fact, the real-time network applications, that currently run over the Internet, achieve a quality of service (QoS) that is far from what is desired.

In order to provide required performance guarantees in the Internet, there has been considerable activity recently on defining and introducing a small set of differentiated services that aim to improve the service of certain classes of applications. One such proposal, the Premium Service Architecture, introduced by Nichols et al. [10] is well suited for supporting stored video transmission (i.e., a video whose characteristics are known a priori). In such architecture, the required performance guarantees are provided by allocating a fixed bandwidth channel equal to the video peak rate. Thus, the video stream can be transmitted without any problems, as the needed bandwidth is always present.

However, through experimental results we show that, if the peak rate BAM is used to transmit a VBR stream (such as video), then a large amount of bandwidth can be wasted. This poses a problem as bandwidth is a precious resource.

One possible approach to reduce bandwidth requirements, is to reduce the video VBR. However, even using smoothing techniques [4], [11], the variability is still present and, hence, the BAM can still waste a large amount of bandwidth. Bit rate variability can also be reduced by modifying the video QoS, as suggested by Zhang et al. [18], but in this case the client must settle for a lower QoS. Another approach is to allocate the bandwidth in a dynamic way, instead of through a fixed bandwidth channel. For instance, [4], [9] suggest using renegotiation mechanisms to avoid bandwidth wastage. While being effective, this technique may, at some point in time, require additional bandwidth while transmitting a video. This raises a potential problem, as the required additional bandwidth may not be available, leading this technique to not provide the needed bandwidth guarantees. Advanced bandwidth reservation could be useful, as suggested by Sikora and Teitelbaum [13], but at the moment, this is not available.

From these considerations, a BAM should not modify the video QoS and should not ask for additional bandwidth while transmitting a video stream. For these reasons a peak rate BAM is used in [10].

The first contribution of this paper is the introduction of a new BAM that provides the same service of the peak rate BAM, while using less bandwidth. Briefly, our BAM starts allocating the peak bandwidth, as the peak rate BAM, but then progressively reduces this allocation as the peak rate of the remaining stream decreases. This is possible because we consider video with known characteristics. In this way, our BAM does not ask for additional bandwidth while transmitting a video stream and does not affect the video QoS. We show that when coupled with optimal smoothing [11], our BAM substantially reduces the bandwidth consumption even though the bandwidth requirements of the resulting smoothed stream does not decrease over time.

Although the proposed BAM can substantially reduce bandwidth consumption, there may still be a further need to reduce bandwidth consumption. Our second contribution consists of several frame dropping mechanisms that further reduce bandwidth consumption subject to a QoS constraint when coupled...
with the above BAM. These mechanisms provide the flexibility for the client to negotiate a tradeoff between bandwidth consumption and QoS degradation with the server (and network). Using these mechanisms, we show through simulation that it is possible to substantially reduce the needed bandwidth while dropping few frames; depending on the movie, we can save up to 43% of the bandwidth while dropping only 1% of the frames.

Further, since VCR functions, such as Pause, Fast Forward and Rewind are usually deployed in video transmissions, we show how to implement such operations in our BAM. The main problem in handling these VCR functions arises from the decreasing bandwidth allocation given by the BAM. During VCR operations the requested bandwidth could be greater than the one actually allocated and, hence, techniques like [1]–[3] cannot be used. For this reason, we introduce a new mechanism that implements VCR functions by storing a part of the video at the client side and by skipping frames at the server side. Simulation shows that, to support VCR operations, there is no need to store a large quantity of video at the client side, and that a few megabytes of storage are sufficient. Even though this quantity is little, it is possible to further reduce it, by slightly reducing the quantity of saved bandwidth. For instance, decreasing the saved bandwidth by only 1%, almost 100 MB of stored video can be avoided.

The paper is organized as follow. In Section II we introduce and evaluate our BAM. In Section III we investigate possible benefits in delivering slightly imperfect QoS and in Section IV we present, and evaluate, some heuristic algorithms for dropping frames. In Section V we propose a VCR mechanism that is evaluated in Section VI. Conclusions are presented in Section VII.

II. AN EFFICIENT BANDWIDTH ALLOCATION MECHANISM

In this section we introduce a new BAM that provides the same service as the peak rate BAM, while using less bandwidth. The peak rate BAM is used inside the Premium Service Architecture [10] in order to support real-time stored stream transmission. This BAM allocates a fixed bandwidth channel equal to the stream peak rate and hence the allocated bandwidth is sufficient to transmit the stream. However, if we consider a VBR video transmission, this allocation can waste large amount of bandwidth which may not be acceptable when someone pays for it, as is likely to happen in the coming years.

To avoid bandwidth wastage, our proposed BAM uses dynamic bandwidth allocation. However, even though dynamic allocation is used, our BAM never asks for additional bandwidth. This substantially differs from other dynamic BAMS. For instance, the renegotiation mechanisms, described in [6], [9], may make requests for additional bandwidth. They cannot guarantee that these requests will be satisfied by the network.

Conversely, our BAM provides bandwidth guarantees as well as the peak rate BAM, while using less bandwidth. This is achieved by allocating the peak bandwidth to the premium channel, but progressively reducing this allocation as the peak rate of the remaining stream decreases. This is possible because we are considering streams with known characteristics. Consequently, there is no need to ask for additional bandwidth while transmitting a stream, and hence our BAM provides the same guarantees as the peak rate BAM, while using less bandwidth.

To better describe our BAM, in the following we consider a sender (that provides the service) and a receiver (that desires the service). The receiver requests a video, composed of \( N \) frames, from the sender. Without loss of generality, we assume a discrete time-model where one time unit corresponds to the time between successive frames. For a 24 fps full motion video, the duration of a frame is 1/24th of a second. We denote by \( a(i) \) the amount of data sent at time \( i \), \( i = 1, \cdots, N \). We introduce the following bandwidth function, which will be used by the BAM:

\[
\text{band}(i) = \max \{a(j) | j \geq i\}, \quad i = 1, \cdots, N.
\]  

(1)

If the bandwidth is allocated using this nonincreasing function, there is no need to ask for additional bandwidth. Conversely, it is possible to reduce the allocated bandwidth when it is no longer needed: at time \( j \) [just before sending the quantity \( a(j) \)], a request to de-allocate the bandwidth is sent if \( \text{band}(j) < \text{band}(j-1) \) and the new allocated bandwidth will be \( \text{band}(j) \), instead of \( \text{band}(j-1) \). The overhead introduced by the de-allocation messages is very small compared to the video transmission. Experiments show that at most 20 de-allocation messages are sufficient for a 28 min video.

In Fig. 1, we show the behavior of the \( \text{band} \) function for a pure VBR video [Fig. 1(a)] and for a smoothed version of the same movie [Fig. 1(b)]. We note that our BAM allocates less bandwidth than the peak BAM because it decreases the bandwidth when future frames don’t need it.

The bandwidth utilization, \( U \), achieved using our bandwidth allocation mechanism is

\[
U = \frac{\sum_{i=1}^{N} a(i)}{\sum_{i=1}^{N} \text{band}(i)}
\]

and is greater than what is obtained using the classic BAM because \( \text{band}(i) \leq \text{Peak \ rate} \), \( i = 1, \cdots, N \).

In this section we present experimental results obtained from analyzing several Motion JPEG (MJPEG) and MPEG encoded videos.

A. Motion JPEG

The MJPEG algorithm compresses each video frame independently using the JPEG still-picture standard. We use four different MJPEG videos [4], Big, Sleepless in Seattle (SIS), Crocodile Dundee (CD), and ET, each consisting of 40,000 frames (28 min). Each video is smoothed [11] considering the client buffer of 1 MB [17] and zero startup delay. These experiments illustrate the benefits of our BAM by showing the reduction in the amount of required bandwidth that it introduces.

In Fig. 2 we show the bandwidth allocation curve \( \text{band} \), the bandwidth requirements for two of the four MJPEG videos and the peak rate allocation. First we note, that even when the traffic is smoothed, it is still quite variable and that this variability results in over bandwidth allocation by a peak rate BAM. Our BAM allocates less bandwidth while still providing the same QoS because \( \text{band} \) better fits the traffic shape than a peak rate
allocation. We also note that the benefits are strongly correlated to the traffic peak rate position. In fact, the benefits are less for *ET* and *Crocodile Dundee* because the peak rate occurs late in the video (refer to [7] for a detailed description). However, considering the worst situation (peak rate at the end of the trace), our BAM behaves exactly as the peak rate BAM. In all of the other situations, our BAM outperforms the peak rate BAM.

Fig. 4 quantifies the benefits of the proposed BAM over a peak rate allocation. We observe that the bandwidth reduction ranges from 5% (*ET*) to 25% (*Big*). This reduction results in better bandwidth utilization. In one case, *Big*, the increase exceeds 20%.

**B. MPEG Traces**

In this section we present results obtained from analyzing MPEG traces. MPEG is an inter-frame dependency encoding mechanism that yields a smaller average frame size than the MJPEG encoding. We use four different videos: *MTV*, *The Silence of the Lambs* (SL), *Jurassic Park* (JP) and *Starwars* (SW).

All traces are 28 min long (except *Starwars*, which is 121 min long), contain 12 frame groups of picture (GOPs) and are 24 fps. We consider a client with 1 MB of buffer available for storing video [17]. We also considered a startup delay of 24 frames (1 s). We use this value in order to obtain considerable smoothing benefits while not incurring too long a startup delay at the client [11].

Fig. 3 shows the allocation using the peak rate BAM and the proposed BAM. We observe that the peak rate BAM wastes a considerable amount of bandwidth while the proposed BAM allocates less bandwidth without compromising the QoS of the video delivered. Due to the lack of space, results obtained from analyzing *Jurassic Park* and *Starwars* are not presented here; we refer the reader to [7] for further details. As we already stated, the benefits of our BAM are correlated to the traffic peak rate position, but the worst situation of our BAM corresponds to the normal situation of the peak rate BAM.

Fig. 4 quantifies the benefits of our proposed BAM over a peak rate allocation. The benefits introduced are remarkable since the reduction in bandwidth requests ranges from 2% (*Starwars*) to more than 30% (*MTV*). *Starwars* yields little benefit because the peak rate of the smoothed traffic occurs very close to the end of the video (see [7] for a detailed description).

**C. Conclusion**

In this section we presented experimental results obtained from analyzing several video (MJPEG and MPEG) traces.

We quantified the allocated bandwidth and we showed that our BAM can substantially reduce the required bandwidth for both types of videos without affecting the QoS. If in the coming years, the cost to the customer is proportional to the used bandwidth, our mechanism will result in a lower cost for transmitting the same video with the same QoS.

**III. DELIVERY OF IMPERFECT QoS**

In previous sections we presented a BAM for transmitting perfect QoS real-time streams, such as video, and illustrated its benefits over a peak rate BAM. Although the bandwidth improvement can be substantial, there may be times when a client may require further bandwidth reduction while being willing to tolerate some QoS degradation. Hence, in this section we present several frame dropping mechanisms, which, when coupled with our BAM, provides the client and the server with the capability to tradeoff bandwidth utilization with the QoS. Note that, imperfect QoS, does not imply unpredictable QoS. Although less than perfect, the QoS is made known to the client. In fact, sender and receiver must agree on this QoS: they should establish a contract in which the receiver agrees to receive a defined nonperfect QoS video and the sender agrees to provide a service with that particular QoS.
Imperfect QoS means that some video frames are dropped. If the video is also smoothed, as recommended, the smoothed traffic presents a slightly imperfect QoS video. The client receives a continuous, although imperfect, stream and continues to play it without halting after dealing with the imperfections through loss concealment [15]. No change is required to the BAM presented in the previous section as it doesn’t matter if the data, used by the band function, represents perfect or non-perfect QoS.

In [7] we describe an algorithm which minimizes the peak rate of a smoothed MJPEG video when the client has a constraint on the number of frames that can be dropped. This algorithm relies on the MINFD algorithm [18] that optimally minimizes the number of frames that have to be dropped if the system has both network bandwidth and client buffer constraints.

Unfortunately, we are concerned with a different problem: given a specified QoS (e.g., the number of frames to drop), we are interested in reducing the average allocated bandwidth and not in minimizing the peak rate of the smoothed video. For this reason MINFD cannot be used. Instead it is necessary to develop other algorithms (presented in the next section) that can reduce the amount of bandwidth needed for transmitting a video, given a number of frames to drop.

IV. FRAME DISCARD ALGORITHMS

In this section, we present several heuristic algorithms that further try to reduce the allocated bandwidth by dropping video frames prior to smoothing it. As stated earlier, imperfect QoS is acceptable in many scenarios. For example, the client could agree to receive imperfect QoS video, provided that the cost was less (for instance in distance learning the video QoS might not be very important). We propose algorithms that discard a certain number of frames corresponding to the QoS established by the server and the client (let us denote this number by \( k \)) in order to reduce the bandwidth needed for transmitting the video. We develop separate sets of algorithms for MJPEG and MPEG. These algorithms are proposed because the optimal selection of the \( k \) frames is too time consuming (\( k \) frames have to be selected over the entire number of video frames). We compare the performance of these algorithms to a baseline random discard algorithm. Our experiments show that the proposed discard algorithms provide slightly better performance than a random discard algorithm.

A. Motion JPEG

**Discard Largest Frames (DLF):** DLF discards the largest \( k \) frames of the video. This algorithm may discard consecutive frames, which may lead to a poor video playout quality at the receiver side.

**Discard Largest Frame with Distance \( \lambda \) (DLF (\( \lambda \))):** A variation of the previous algorithm. The algorithm uses \( \lambda \) as a parameter that indicates the minimum distance between discarded frames. Unfortunately, there is no way to suggest the optimal value of the \( \lambda \) parameter, as it is affected by the characteristics of the considered video. Hence, the sender should test for different values of \( \lambda \), and then can select the best one.

Our experiments compare random frame discard and our heuristics to the mechanism, described in [7], which uses MINFD to minimize the bandwidth peak rate. We use three algorithms: DLF, DLF(2) and DLF(5), in order to compute the amount of the reduced bandwidth reached by these algorithms. Fig. 5 illustrates the bandwidth reduction over a peak rate BAM as a function of the number of dropped frames. We observe that our algorithms result in lower bandwidth consumption than MINFD. This is because the mechanism that uses MINFD minimizes the bandwidth peak rate, whereas our algorithms are concerned with reducing the average allocated bandwidth. Note that, as expected, our heuristics perform better (even though not substantially) than a random frame discard policy (RAND). This is because, while discarding the same number of frames, our heuristics discard larger frames.

In all of the experiments, we observe that discarding a small percentage of frames can greatly reduce the allocated bandwidth. In Fig. 5(a), discarding 2% of the frames results in a 16% reduction in the allocated bandwidth over a peak rate allocation. All four discard policies perform similarly with DLF slightly better than DLF(2) which is slightly better than DLF(5), which is quite similar to RAND. This is because DLF can discard consecutive frames (in the worst case all of the discarded frames could be consecutive frames and this could result in poor playback quality), whereas DLF(2) and DLF(5) cannot. In Fig. 5(b) the four algorithms produce almost the same results, a reduction in the allocated bandwidth from 13% (perfect QoS) to 22% (5% of dropped frames). Readers can refer to [7] for results from analyzing ET and Big.

So far we focused on the number of dropped frames. Unfortunately there may not be much correlation between dropped frames and perceptual quality of playout [14]. One approach to accounting for the perceptual quality of playout is to use a cost function to measure the perceived video quality. Many ways exist to define a cost function, but its definition goes beyond the scope of this paper. For this reason, we focus on a cost function designed for MJPEG videos and introduced in [18] which attempts to penalize algorithms that drop neighboring frames. This cost function takes two aspects into consideration: the length of a sequence of consecutive discarded frames and the distance between two adjacent but nonconsecutive discarded frames. It assigns a cost \( c_j \) to a discarded frame \( j \) depending on whether it belongs to a sequence of consecutive discarded frames or not. If frame \( j \) belongs to a sequence of consecutive discarded frames, the cost is \( I_j \), if the frame \( j \) is the \( I_j \)th consecutively discarded frame in the sequence. Otherwise the cost is given by \( 1 + 1/\sqrt{d_j} \), where \( d_j \) represents the distance from
the previous discarded frame. More details about this cost function can be found in [18]. We present the cost achieved by the heuristic algorithm when applied to Sleepless in Seattle in Fig. 6: DLF performs completely worse that the others, while DLF(2), DLF(5), RAND and MINFD achieve very similar results.

Based on this cost function, DLF is not worth using because its cost is too high and because its performance is almost like that of DLF(2), DLF(5) and RAND. These three algorithms have very similar values, but since DLF(2) results in a greater bandwidth reduction, it is preferred to DLF(5) and to RAND. However, we point out that a different cost function could lead to different conclusions.

B. MPEG

In this section we present some heuristics specifically designed for MPEG videos, as the MPEG encoding differs from the MJPEG encoding. In MPEG, the frames don’t have the same importance as some frames depend on other frames. We use MPEG videos organized in GOP with a size of 12 frames. MPEG can use three types of frames: I, P and B. The GOP is composed of: IBIB2IB3IB4PB5PB6PB7IB8. To decode a B frame, both the previous and future I or P frames are needed. To decode a P frame the previous P or I frame is needed. Thus, the discard of an I frame results in the discard of 14 frames (the entire GOP plus the two B frames of the previous GOP that depend on the I frame), the discard of a P1 frame results in the discard of 11 frames, the discard of a P3 frame results in the discard of 8 frames, and the discard of a P5 frame results in the discard of 5 frames. Only a B frame discard results in no additional frame discard. Based on these dependencies we propose the following algorithms:

Drop I Frame (DIF): DIF drops the largest GOPs (plus the two B frames preceding the I frame) of the video. If L is the maximum number of GOPs that can be discarded, the algorithm discards the L largest GOPs of the video. Hence, if k is the maximum number of frames that can be dropped, L = k/14.

DIF (λ): a variation of the previous algorithm. The algorithm uses λ as a parameter that indicates the minimum distance (in GOP) between discarded GOPs.

Discard First P Frame (DFPF): DFPF discards the largest group of frames that depend on the P1 frame (i.e., 11 frames). Hence, L = k/11.

Discard Second P Frame (DSPF): DSPF discards the largest group of frames that depend on the P2 frame. Hence, L = k/8.

Discard Third P Frame (DTPF): DTPF discards the largest group of frames that depend on the P3 frame. Hence, L = k/5.

Discard B Frame (DBF): DBF discards only the B frames of the video. The algorithm orders the B frames and discards the largest L frames. (L = k).

In our experiments we use DIF, DIF(2), DIF(5), DFPF, DSPF, DTPF and DBF in order to compute the amount of reduced bandwidth achieved by these algorithms. Random discard is not presented, as it produces less benefit than the proposed heuristics (as was the case for MJPEG videos). Fig. 7 illustrates the bandwidth reduction over a peak rate allocation as a function of dropped frames. We observe that discarding a small percentage of frames can greatly reduce the allocated bandwidth.

In Fig. 7(a) DIF and DFPF achieve great results since a drop of 5% of the video results in a bandwidth reduction of 58%. Note that a drop of 1% of the video allows a bandwidth saving of up to 42%. In Fig. 7(b) a drop of 1% of the video results in a bandwidth reduction of 16%. DIF and DFPF result in a bandwidth saving of 24% when 5% of the video is dropped. With Starwars a drop of 1% of the video allows a bandwidth saving of 7% and a drop of 5% of the video results in a bandwidth reduction of 22% (using DIF). With MTV, once again, our heuristics allow a bandwidth saving of 48% when 5% of the video is dropped and a drop of 1% of the video results in a bandwidth reduction of 43%. For a detailed description readers can refer to [7].

It is difficult to directly relate bandwidth reduction (the sender’s goal) to the perceived video quality (the client’s goal). One can introduce a cost function for MPEG videos, such as the one introduced for MJPEG. However, the choice of discard algorithm depends on the cost function. For this reason, it is difficult to recommend a particular heuristic. After selecting a cost function, the sender can select a particular discard algorithm. The important thing is that, based on the experimental results obtained, it is clear that heuristics produces benefits.

V. INTERACTIVE OPERATIONS

In this section we provide our BAM with interactive functions, such as pause, fast forward (FF), and rewind (REW). These operations are very useful in applications such as distance learning and remote digital libraries.

Several works have considered the problem of adding VCR functions to streaming video applications. Sicar et al. [2] require the server to transmit the video at higher rate. Shenoy and Harrick [12] propose alternative video encoding with a coarser granularity and a lower resolution. Chen et al. [1] propose to cache all previously displayed frames on a local disk. All these mechanisms require additional resources or employ different encoding mechanisms. Unfortunately, in common Internet environments, additional resources, such as additional bandwidth, bigger buffers or large storage devices at the client, may not be available. Furthermore, our BAM has been designed to decrease the allocated bandwidth while transmitting the video.
In such a scenario, the implementation of VCR functions is not trivial, because the allocated bandwidth may not be sufficient to allow the user to rewatch a part of the video. In fact, the play out of a previous part of the movie could require more bandwidth than the one currently allocated.

In this section we propose a mechanism that can be considered a hybrid between frame skipping mechanisms [3] and storing mechanisms [1]. Our approach introduces VCR functions without requiring additional bandwidth and without the need to completely store the video at the client, as in [1]. As we show, there is a tradeoff between the amount of video that is necessary to store at the client side, in order to provide VCR functions, and the quantity of saved bandwidth. We investigate this tradeoff through simulation, where we use pure VBR video traces.

A. Pause

Pause can be performed by simply informing the sender to temporarily stop the video transmission, in order to avoid client buffer overflow. Needless to say, that the sender has to hold the allocated bandwidth, otherwise, when the client wants to resume video play out, it may be impossible to resume video transmission.

B. Fast Forward

FF operation is performed by skipping frames at the sender. This means that the sender, while transmitting the same number of frames per second, delivers a stream that corresponds to a larger sequence of the movie. This allows the client to play out a fast movie. This technique is easy to implement since the sender, after receiving a FF request, starts transmitting every nth frames, where n represents the speed factor requested by the client. Note that, to perform this operation, no additional bandwidth is requested and that the video QoS depends only on the requested speed factor.

In Fig. 8, we show an example. Fig. 8(a) represents a normal video transmission. Fig. 8(b) represents the same video as if it was transmitted in FF(2) mode (n = 2). Skipping frame technique is trivial to implement when video is encoded with intraframe technique (as Motion JPEG). If video is encoded using an inter-frame technique (as MPEG), the sender can use one of the algorithms proposed in Section IV-B in order to drop frames (the choice depends on the requested speed factor).

C. Rewind

REW operation is more difficult to handle than the FF function. In fact, bandwidth is allocated using a nonincreasing function (band). This poses a problem when video is played out in reverse order or when a portion of the video is played out again. In fact, the allocated bandwidth may be not sufficient to perform these operations.

For instance, suppose that a REW function is requested at time j. The allocated bandwidth at time j is band(j). The REW operation causes the sender to transmit an amount of data a(h) during a unit of time where h < j. Since a(h) can exceed that band(j), as h < j, the amount of data a(h − band(j)) (if any) cannot be sent by the sender. To solve this problem, we require the client to store this portion of the video. Since this quantity depends on the time the client requests the REW operation and since this time is unpredictable, we consider the worst situation, such as when the function band assumes its minimum value $b_{\text{min}}$, where $b_{\text{min}} = \min(band(i), i = 1, \ldots, N)$. This means that the client, during the normal video play out, needs to store $s(i)$ data, where $s(i) = \max(0, a(i) - b_{\text{min}})$, for each $i = 1, \ldots, N$.

In conclusion, if a REW operation is requested at time j, then the sender has to transmit $\min(b_{\text{min}}, a(i))$ data of $a(i)$, that will be integrated with the quantity $s(i)$ already present at the client, for $i = 1, \ldots, j-1$. For instance, in Fig. 9 we show the portion of the video that has to be stored at the client in order to provide VCR operations.

Once again, depending on the speed factor requested by the user, the sender sends one frame out of n (n = 1 represents normal speed) for MJPEG videos and can choose one of the discarding algorithms proposed in Section IV-B, for MPEG encoded video.

Note that $b_{\text{min}}$ is critical since it affects both the amount of data that has to be stored at the client and the amount of data that the sender retransmit. In Section VI we investigate these dependencies.

D. BAM Modification

The handling of the previous VCR functions could raise a play out problem. For instance, consider Fig. 10. At time $t_0$ the video is transmitted in FF mode up to time $t_1$ (only some frames are transmitted in FF mode). After $t_1$, the video is played out at normal speed. Now, suppose that at time $t_2$ ($t_2 \geq t_1$) a REW operation is requested and that the client wants to rewatch the video from time $t_0$.

As we know, the bandwidth is not sufficient to transmit the video ($\text{band}(t_2) \leq \text{band}(t_0)$) and, for this reason, the client should use the stored $s(i)$ portion. Unfortunately, between $t_0$ and $t_1$, some frames haven’t been transmitted (skipping frames policy), and hence the client couldn’t have stored the needed portion of these frames. For these reasons, the video cannot...
be correctly played out. To avoid this problem the bandwidth cannot be decreased during FF transmission, and the allocated bandwidth can be decreased only if the previous part of the video has been normally played out. This means that bandwidth can be decreased at time $j$ only if it was allowed at time $i$, for each $i = 1, \cdots, j - 1$. This can raise a utilization problem. Suppose that, just at the beginning of the video, the client operates in FF mode for few frames and then it normally plays out the video: the bandwidth would never be decreased. To avoid this problem, we suggest that the sender set a play out threshold time unit for FF operation. In this case, the sender asks the client if it is still interested in the previous part of the video that it quickly watched. If not, the bandwidth is immediately decreased.

This is reasonable in distance learning, where a student can quickly browse until he/she reaches the part he/she is interested in. Regarding movies play out, it is unlikely that a FF operation is performed at the beginning, unless the user is not interested in this part (previews, casting list, etc.) or he/she is looking for a particular scene of the movie. In both cases, it is unlikely that the user will watch the FF part later. Hence, we believe that it is reasonable to introduce the threshold mechanism.

VI. EXPERIMENTAL RESULTS

As stated, there is a dependency between the saved bandwidth and the amount of data that has to be stored at the client. In this section, we investigate this trade-off, by modifying the final part of the bandwidth function, increasing its minimum up to the video peak rate (no saved bandwidth with respect to the BAM proposed in [10]). Note that this modification does not affect the nonincreasing behavior.

Through simulation, we show that a small variation in $b_{\text{min}}$ (i.e., small change in the saved bandwidth) allows the client to substantially reduce the amount of video that must be buffered. Using pure VBR videos, we show that there is no need to store gigabytes of video at the client, but only a few MB are sufficient to provide VCR functions.

A. Motion JPEG

In Fig. 11(a) (Big) we show that to save the 9% of the bandwidth and to provide VCR operations, the client is required to store almost 150 MB of video. If we decrease the saved bandwidth to 6% then the client has to store only a few megabytes of the video. In Fig. 11(b) Crocodile Dundee we show that a decrease of 2% of the saved bandwidth allows the client to avoid storing more than 20 MB of video.

B. MPEG Video

Results obtained from analyzing MTV show that, to save an amount of bandwidth equal to 11%, and to provide VCR functions, the client has to store 70 MB of the video. Decreasing the saved bandwidth by only 1%, allows the client to reduce the storage to less than 10 MB of video. From analyzing The Silence of the Lambs a decrease of only 1% of the saved bandwidth allows the client to avoid storing more than 13 MB of the video.

VII. CONCLUSION

In this paper, we presented a new BAM that can be used in the premium service architecture [10] to handle real-time VBR streams over the Internet. Our BAM increases the bandwidth utilization for a stream transmission under the premium service architecture and it is easily implemented in [10], as it does not require any architectural modification. We showed, through several experiments, that its use can greatly reduce the allocated bandwidth for transmitting the same traffic with the same QoS and the same guarantee. We showed further possible benefits by sending slightly imperfect QoS video. We developed some heuristics that can be used to drop frames in order to minimize the bandwidth consumption. All of the experiments presented in this paper show that the use of our algorithms can possibly lead to a great bandwidth reduction while dropping very few frames. We also provided VCR capabilities to our BAM, without requiring additional bandwidth and without the need of completely storing the video at the client. We showed that, to provide VCR functions, only few MB of video at the client side, are needed to be stored. This amount depends on the quantity of saved bandwidth. Through several simulations we investigated the dependency between saved bandwidth and quantity of stored video at the client side. We showed that a little reduction of the saved bandwidth allows decreasing the stored video (at the client) of several MB.

Since bandwidth is a precious resource, we think that our BAM may be very useful both for the server and the client. The client could be happy to pay less for the same service or for a slightly imperfect service and the server could be happy because reducing the bandwidth needed for one service could mean making bandwidth available for other services.

Our study has assumed knowledge of the bandwidth characteristics of the video stream (as well as the Premium Service). This is reasonable in the case of stored video. We are investigating the problem of handling video streams with unknown characteristics.
REFERENCES


Donald F. Towsley (M’78–SM’93–F’95) received the B.A. degree in physics and the Ph.D. degree in computer science from the University of Texas, Austin, in 1971 and 1975, respectively. From 1976 to 1985, he was a member of the faculty, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, where he is currently a Distinguished Professor in the Department of Computer Science. He has held Visiting Positions at the IBM T. J. Watson Research Center, Yorktown Heights, NY (1982–1983); Laboratoire MASi, Paris, France (1989–1990); INRIA, Sophia-Antipolis, France (1996); and AT&T Labs—Research, Florham Park, NJ (1997). His research interests include networks, multimedia systems, and performance evaluation. He currently serves on the editorial board of Performance Evaluation.

Dr. Towsley has previously served on several editorial boards, including the IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE/ACM TRANSACTIONS ON NETWORKING. He was a Program Co-Chair of the joint ACM SIGMETRICS and PERFORMANCE ’92 conference. He is a member of ACM, ORSA and the IFIP Working Groups 6.3 and 7.3. He has received the 1998 IEEE Communications Society William Bennett Paper Award and several best conference paper awards from ACM SIGMETRICS. He is a Fellow of the ACM.

Marco Furini received the degree and the Ph.D. degree in computer science from the University of Bologna, Italy, in 1995 and 2001, respectively. From August 1998 to May 1999, he visited the Department of Computer Science, University of Massachusetts, Amherst. His scientific interests include analysis and design of protocols and mechanisms for supporting real-time network applications.