# A Packet Selection Algorithm for Adaptive Transmission of Smoothed Video Over a Wireless Channel [*]

Zhimei Jiang and Leonard Kleinrock [†], Fellow, IEEE

**Abstract**

*This paper discusses techniques for transmitting smoothed video more efficiently over a wireless network. The fluctuation of wireless channel conditions can add a significant amount of delay to video packets and cause them to miss their play-out time. If a video is smoothed, it is possible to selectively deliver packets delayed at the base-station to reduce the impact of the missing packets to the video quality. In this paper, we present a simple yet effective packet selection algorithm which accomplishes this goal. The algorithm determines whether to transmit a packet based on channel conditions as well as how likely higher priority layers in the rest of the video may be delivered on time. We introduce the concept of "quality index" to measure the performance of the algorithm. Results from trace-driven simulations show that the performance of the algorithm is close to optimum under a wide variety of channel conditions when the threshold is set between 0.7 and 0.9.*

## 1 Introduction

Video is expected to be one of the most important applications over the Internet in the future. However, efficient transmission of video, especially variable bit rate video, has always been among the most challenging problems to network designers, largely because of the video's high bandwidth demand, the QoS requirements, and the significant rate variability [15]. Video smoothing has been introduced as an effective way to reduce the variability of the bandwidth requirement for delivering video, which can potentially simplify other operations such as resource allocation and also improve network utilization [20].

The idea of video smoothing is to preload part of the video to a smoothing buffer at the client before play-out, so that after play-out has started, the rest of the video may be transmitted in a less bursty fashion without compromising the quality of the video [7, 16]. For a given video, the video smoothing algorithm generates a *transmission schedule*, which is the rates at which the video will be delivered during play-out, based on buffer size, available bandwidth, and allowed play-out delay. A valid transmission schedule must guarantee that, given the bandwidth it requires, the smoothing buffer will not overflow or underflow during the entire play-out. Depending on user requirements, the smoothing algorithm may also need to optimize certain characteristics of the transmission schedule, such as peak rate, number of rate changes, etc [9].

When a video needs to be delivered over a wireless network, due to the fluctuation of channel conditions, it becomes even more important to perform smoothing. The fluctuation of wireless channel conditions may

[*]This work was supported by the Advanced Research Projects Agency, ARPA/CSTO, under Contract DABT-63-94-C-0080 "Transparent Virtual Mobile Environment".

[†]Zhimei Jiang is with At&T Research Labs. Leonard Kleinrock is with the Computer Science Department, University of California at Los Angeles. (email: jiang@research.att.com, lk@cs.ucla.edu)

cause the throughput of the wireless link to drop below the arrival rate of the video stream from the wired network. Undelivered packets are queued at the base-station waiting to be sent out. However, long delays at the base-station can cause the packets to miss their play-out time and result in degradation of video quality. If the video is smoothed, the scheduled arrival time of a packet is normally earlier than its play-out time. Therefore, more delay at the base-station can be tolerated, and the packet drop rate may be reduced. More interestingly, assume video is layer encoded and each layer is assigned a priority according to how important it is to the quality of the video. Then for a fixed packet drop rate, the video quality may be further improved by reducing the number of high priority packets being dropped. In this paper, we study a packet selection algorithm which accomplishes this goal. The algorithm intelligently selects packets to deliver so that a higher video quality may be obtained, in the case that not all the packets can be transmitted on time. We introduce the concept of *quality index* to evaluate the algorithm performance and to study possible improvements of the algorithm through trace driven simulations. The results show that by selecting a threshold appropriately, in most cases, the improved algorithm can perform within 1-3% of the optimum result determined by the channel conditions and is significantly better than without the packet selection algorithm.

The rest of the paper is organized as follows. In section 2, we briefly summarize related research on adaptive video transmission. Section 3 describes the setup of the system we will be studying. Starting from section 4, we will be focusing on the packet selection algorithm. The algorithm will be presented and its performance will be investigated in detail.

## 2 Related work

When a video is delivered over a network, it typically shares the link with data packets and other audio/video streams. As a result, the available bandwidth can sometimes be lower than the required value thus causing long packet delay and even packet loss. To handle this type of situations and to reduce the impact of packet loss on video quality, various approaches have been proposed to adaptively transmit or play out the video based on network conditions. Specifically,

- At the player's end, the play-out time may be adjusted within a small range based on information about delay and the amount of data in the buffer, to prevent potential play-out discontinuity [17, 19].
- At the network node, adaptive forward error control algorithm may be implemented to reduce packet loss rate [5, 6]. In addition, video encoded with one standard may be transcoded to another standard that requires less bandwidth [1].

- At the video source, the output rates may be adjusted based on feedback information about the network [5, 6]. In particular, for real time video, parameters in the coding algorithm may be adjusted to generate video at different rates [3, 4, 13]. If the video is pre-encoded, bit-streams may be scaled by cutting high frequency components, by requantization, or by re-encoding reconstructed pictures [8, 11, 18].

Some of these schemes mentioned above may output video that is slightly smoothed. However, unlike our packet selection algorithm, none of them treat smoothed video explicitly. The packet selection algorithm proposed in this paper selectively delivers packets to achieve a better overall video quality. In making the selection, it takes into consideration both channel conditions and the characteristics of the smoothed video.

## 3 System Setup

Figure 1 depicts the system we will be studying. Basically, the smoothed video stream is delivered to the remote user through a network path which is comprised of a wired portion and a wireless portion. The wired network is assumed to have sufficient bandwidth, thus does not introduce any delay jitter to the packets. The wireless channel delivers packets as fast as the channel condition allows. Delayed packets are stored temporarily at the base-station buffer before being forwarded over the wireless link. The packet selection algorithm runs at the base-station to determine which of the delayed packets to deliver. Packets which have missed their play-out time are removed from the buffer without being transmitted.

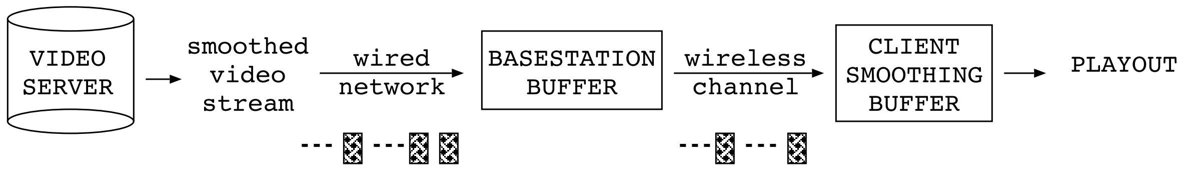


Figure 1: *Structure of a video smoothing and transmission system.*

We assume time is divided into time slots of constant length. Once the play-out starts, exactly one video frame is played at the beginning of every time slot. Each time slot is further divided into $R$ mini-slots of equal length, with one mini-packet being delivered in one mini-slot. The mini-slot corresponds to the transmission unit of the wireless link. For instance, in EDGE (Enhanced Data rate for GSM Evolution), a user occupying one channel receives one (mini-)packet every 20 msec [22]. If a packet from the wired link is larger than one mini-packet, it is split into several mini-packets which are transmitted one by one separately. A mini-packet received by the client may be corrupted due to channel interference, etc. The packet selection algorithm may decide to retransmit or simply drop the packet as described in the next section. For simplicity, we assume the feedback to base-station is instantaneous.

## 4 Packet selection algorithm

In the existing video transmission systems, packets are always delivered in the same order as they are played*, and are dropped only if buffers within the network overflow. As a result, when packet dropping does occur, it usually affects a number of consecutive frames, and what a user sees is a video with alternating good and bad periods. Clearly, this may not be the best from the user's point of view, as users typically prefer a stable black and white video over a color one with part of or entire frames missing from time to time, even though the two may contain the same amount of data. This preference on the visual effect allows us to improve the perceived video quality without increasing the amount of data actually transmitted. In particular, we may predict how the rest of the video will be delivered and try to deliver more high priority packets by discarding lower priority ones in advance. And we would like to do it in such a way that the variation of video quality may be kept as low as possible, and at the same time the quality itself may be kept as high as possible. The packet selection algorithm presented below achieves this goal by choosing an appropriate threshold value.

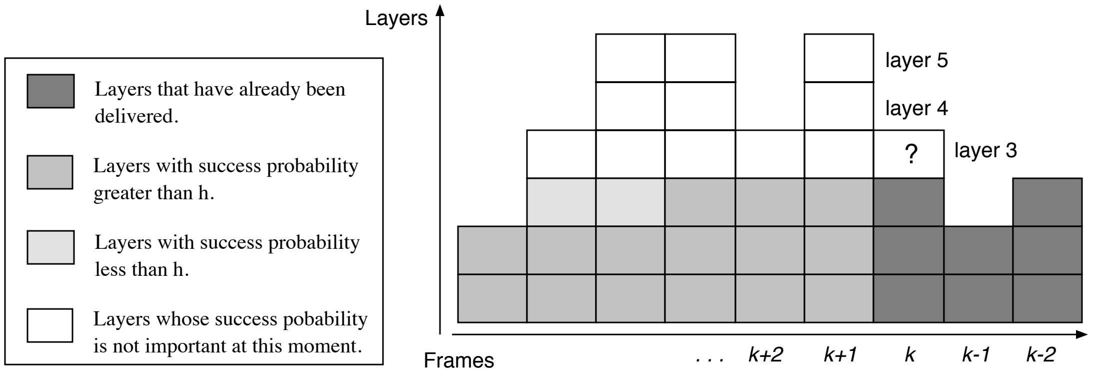
Assume frames are coded with a layered encoding algorithm and lower layers are more important thus having higher priority. In addition, assume packets are sent from the server in the increasing order of their frame number and layer number. Denote layer $l$ of frame $f$ as $L_{f,l}$. The core of our packet selection algorithm may be stated as follows:

> Assume the packet currently at the head of the base-station buffer belongs to layer $i$ of frame $k$. We transmit this packet from $L_{k,i}$ in the next mini-slot if and only if at the *end* of the *next* mini-slot, for any higher priority layer in the rest of the video, the probability that it will be delivered before its play-out time is greater than a given threshold $h$. Otherwise, we skip the rest of frame $k$ and start sending layer 0 of frame $k+1$.

The algorithm is executed at the end of every mini-slot to determine what to transmit in the next mini-slot. To illustrate how the algorithm works, let's consider the video stream shown in figure 2, where each column represents one frame while each row represents one layer. The shade of a rectangle indicates the status of the layer in the corresponding frame. A frame does not have to include all layers and layers in different frames need not be of the same size ( although they are drawn the same in the figure ). At the moment

*This may not be exactly the case in some coding schemes. For instance, in MPEG, a B frame may need the I or P frame behind it in order to be decoded. Therefore the order in which compressed pictures are found in the video stream is different from the order in which they are played. However, the discussions in this paper can still be applied with some minor changes.

shown in figure 2, up to layer 2 of frame $k$ has been transmitted, and the system needs to decide whether to continue to send a packet from layer 3 of frame $k$ in the next mini-slot, or proceed to frame $k+1$. According to our algorithm, a packet from $L_{k,3}$ is delivered if and only if higher priority layers in the rest of the video, which in this case is layer 0 to 2 of all the frames with index greater than $k$, may still be transmitted before their play-out time with probability greater than $h$ at the end of next mini-slot.



Layers that have already been delivered.
Layers with success probability greater than h.
Layers with success probability less than h.
Layers whose success pobability is not important at this moment.
Layers
?
layer 5
layer 4
layer 3
layer 2
layer 1
layer 0
Frames
. . . k+2
k+1
k
k-1
k-2

Figure 2: *Example of how the packet selection algorithm works.*

Before discussing the packet selection algorithm in more detail, we would like to first introduce a term called *success probability*. Specifically, the success probability of layer $l$ of frame $f$, which is denoted by $SP_{f,l}$, is defined as the probability that layer $l$ of frame $f$ may be transmitted before its play-out time given that the system will also deliver as much as possible of up to layer $l$ of all the frames between $f$ and the current frame. Based on our definition of success probability, the packet selection algorithm may be restated as

> Assume the packet currently at the head of the base-station buffer belongs to $L_{k,i}$. Transmit this packet in the next mini-slot if and only if $SP_{f,l} > h$ for all $k+1 \leq f \leq N$ and $0 \leq l \leq i-1$ at the *end* of *next* mini-slot.

where $N$ is the total number of frames. Since $SP_{f,l1} < SP_{f,l2}$ for any $l_1 > l_2$, it is sufficient to just consider whether $SP_{f,i-1} > h$ for all $k+1 \leq f \leq N$. Thus the core of our packet selection algorithm may be further simplified to

> Assume the packet currently at the head of the base-station buffer belongs to $L_{k,i}$. Transmit this packet in the next mini-slot if and only if $\min\{SP_{f,i-1} | k+1 \leq f \leq N\} > h$ at the *end* of *next* mini-slot.

This evaluation takes place at the end of each mini-slot to decide which packet to transmit in the next mini-slot. Again, assuming the next packet in the video stream sequence belongs to $L_{k,i}$, the complete packet selection algorithm works in the following way as shown in figure 3. Basically, if frame $k$ will be played before the end of next mini-slot, skip the rest of $k$ and start sending layer 0 of the next frame which may still make its play-out time. Otherwise, as described above, depending on whether $\min\{SP_{f,i-1} | k+1 \le f \le N\} > h$ holds, either a packet in $L_{k,i}$ or one from $L_{k+1,0}$, is transmitted in the next mini-slot. If $i = 0$, the packet is always transmitted, unless it has already missed its play-out time. Returning to the example shown in figure 2, since at least layer 2 of two frames after $k$ have success probability less than $h$, the packet from $L_{k,3}$ will not be delivered. Instead, the system will start sending packets from layer 0 of frame $k+1$.

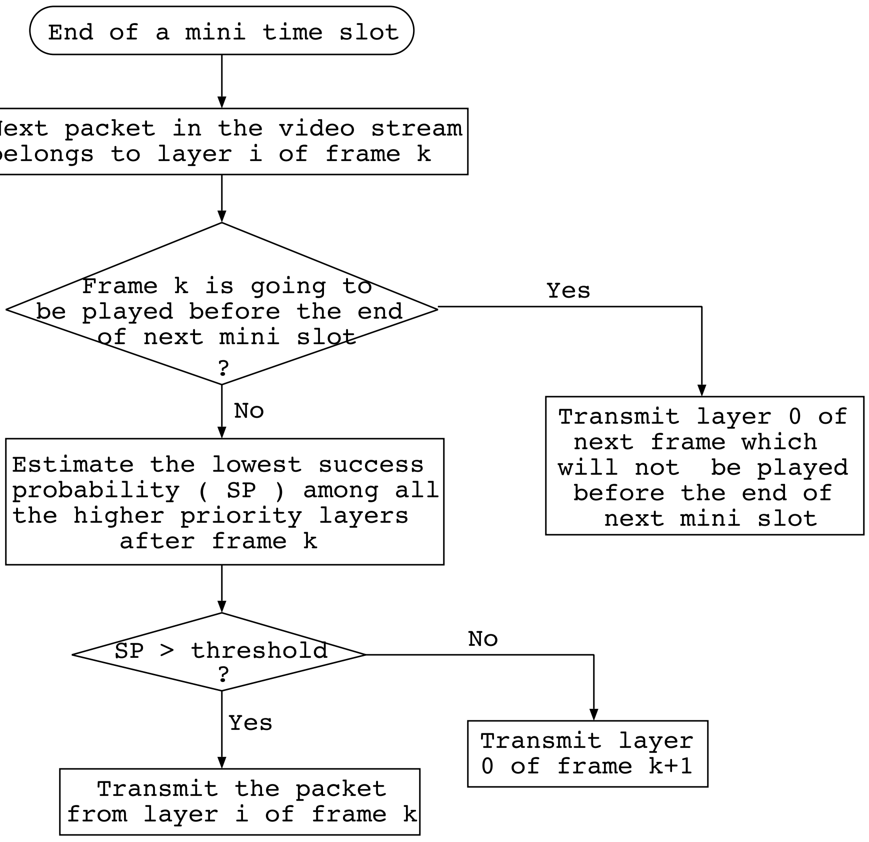

End of a mini time slot
Next packet in the video stream belongs to layer i of frame k
Frame k is going to be played before the end of next mini slot ?
Yes
No
Transmit layer 0 of next frame which will not be played before the end of next mini slot
Estimate the lowest success probability ( SP ) among all the higher priority layers after frame k
SP > threshold ?
No
Yes
Transmit layer 0 of frame k+1
Transmit the packet from layer i of frame k

Figure 3: *Flow chart of the packet selection algorithm.*

The procedure shown in figure 3 is performed at the end of every mini-slot. Packets that are skipped by the algorithm are removed from the base-station buffer immediately. If the channel is in an undesirable state, a mini-packet may not be received by the client correctly. In this case, it will remain at the head of the buffer and will be retransmitted if the conditions in the packet selection algorithm are still satisfied for the next mini-slot. Otherwise, the packet is dropped and the system starts sending a new frame.

Two key parameters which determine the performance of the packet selection algorithm are the success probability and the threshold $h$. In the next two sections, we discuss how to compute success probability and how to select the threshold value to achieve better performance.

## 5 Success probability

The packet selection algorithm requires the value of success probability $SP_{f,l}$, for all $k+1 \leq f \leq N$, in order to decide whether to transmit a packet from $L_{k,l+1}$. In reality, it is very difficult to compute the exact value of $SP_{f,l}$ due to the following reason. According to the definition of success probability, $SP_{f,l}$ is computed based on the assumption that the system will try to send up to layer $l$ of all the frames between $k$ and $f$. However, two things might actually happen to an intermediate frame, say $j$ $(k < j < f)$. One is that up to layer $l$ of the frame is indeed transmitted before its play-out time, and the system starts sending frame $j+1$ immediately after $L_{j,l}$ is finished. The other is that frame $j$ is played before up to layer $l$ of the frame can be transmitted completely, in which case the system discards the rest of $j$ and starts sending frame $j+1$ in the next mini-slot. Therefore, to compute $SP_{f,l}$, we must consider two cases for every intermediate frame $j$, corresponding to whether the previous frame, $j-1$, is finished before its play-out time. As the number of frames increases, the complexity of this computation increases exponentially. For a long video stream, it is therefore virtually impossible to compute the exact value of $SP_{f,l}$ for every remaining frame at the end of every mini-slot. We must find a way to estimate success probabilities.

### 5.1 Estimating the success probability

The discussion above shows that the difficulty of finding a success probability lies in that its value depends on the exact sequence of the channel states, which is unknown in advance. However, if we have knowledge of the probability distribution of channel states, the value of $SP_{f,l}$ may be estimated by predicting the channel conditions in the future. Specifically, the following steps may be taken to obtain the final estimation of the minimum success probability in the algorithm.

**Step 1.** Assume the system needs to find the success probability of layer $l$ of frame $f$, $SP_{f,l}$, to decide whether to transmit a packet from $L_{k,l+1}$. Let $A_{f,l}$ be the amount of data from frame $k+1$ to $f$ up to layer $l$, that is, $A_{f,l} = \sum_{i=k+1}^{f} \sum_{j=0}^{l} s_{i,j}$, where $s_{i,j}$ is the size of layer $j$ of frame $i$. Let $M_f$ be the number of mini-slots left before the play-out time of frame $f$ from the end of next mini-slot. Denote by $P_{A_{f,l},M_f}$ the probability that *at least* $A_{f,l}$ mini-packets are transmitted in $M_f$ mini-slots. This $P_{A_{f,l},M_f}$ may be used to estimate the value of $SP_{f,l}$ at the end of next mini-slot.

**Step 2.** For a long video, it may still be hard to compute $P_{A_{f,l},M_f}$ since both $A_{f,l}$ and $M_f$ can be very large. Assume one frame is played in every time slot which is comprised of $R$ mini-slots. Let $x_{f,l}$, be the average number of packets that need to be transmitted in $R$ mini-slots in order to send at least $A_{f,l}$ packets in $M_f$ mini-slots, that is $x_{f,l} = \lceil \frac{A_{f,l}R}{M_f} \rceil$. To further simplify the computation of success probability, we may again estimate $P_{A_{f,l},M_f}$, hence $SP_{f,l}$, with $P_{x_{f,l},R}$. When both $M_f$ and $R$ are reasonably large, say greater than 10, $P_{A_{f,l},M_f}$ is very close to $P_{x_{f,l},R}$.

**Step 3.** What the packet selection algorithm needs is actually the minimum value among all the success probabilities for layer $l$ of frames after the current one. Since for any frame $f$, $SP_{f,l}$ may be estimated by $P_{x_{f,l},R}$, instead of computing all the success probabilities and then taking the minimum, we may find the maximum average number of packets need to be transmitted in one time slot, say $X_l$, i.e. $X_l = \max\{x_{f,l} | k+1 \leq f \leq N\}$, then $P_{X_l,R}$ is the estimation of the minimum success probability we are looking for.

To illustrate how to compute the minimum success probability, let us consider the simple VBR video example shown in figure 4. Assume the video has four layers; the number of packets included in each layer is labeled in the figure. There are only five frames left and the system needs to decide whether to transmit a packet from $L_{k-1,3}$, i.e. layer 3 of frame $k-1$, or start sending frame $k$. Assume the play-out time of the last frame is $T$; the current time is $T-65$, and $R=15$. If a packet from $L_{k-1,3}$ is indeed delivered in the next mini-slot, then at the *end* of next mini-slot, the value of $x_{f,2}$ for the remaining four frames are

$$\begin{aligned}
x_{k,2} &= \lceil 12 * 15/19 \rceil = \lceil 9.47 \rceil = 10 \\
x_{k+1,2} &= \lceil (6+12) * 15/34 \rceil = \lceil 7.94 \rceil = 8 \\
x_{k+2,2} &= \lceil (15+6+12) * 15/49 \rceil = \lceil 10.10 \rceil = 11 \\
x_{k+3,2} &= \lceil (9+15+6+12) * 15/64 \rceil = \lceil 9.84 \rceil = 10
\end{aligned}$$

The maximum of the four is $x_{k+2,2}$, thus $X_2 = x_{k+2,2} = 11$. We can then compare $P_{11,15}$ with the threshold to decide whether to transmit the packet from $L_{k-1,3}$. If the current time is $T-79$, $x_{k,2}$ to $x_{k+3,2}$ can be computed similarly. But this time $x_{k+2,2} = \lceil (15+6+12) * 15/63 \rceil = \lceil 7.85 \rceil = 8$ is less than $x_{k+3,2} = \lceil (9+15+6+12) * 15/78 \rceil = \lceil 8.08 \rceil = 9$. So $X_2 = x_{k+3,2} = 9$.

Clearly, the value of $x_{f,l}$ changes after each mini-slot. Recomputing $x_{f,l}$ from scratch at the end of each mini-slot can be very time-consuming. There are various ways to reduce the amount of computations by locating large frames which generally result in larger $x_{f,l}$, by taking advantage of the $x_{f,l}$ computed for the previous mini-slot, and by precomputing certain parameters to avoid going through the entire video stream as described in [12]. We will not go into more details in this paper due to space limitations.

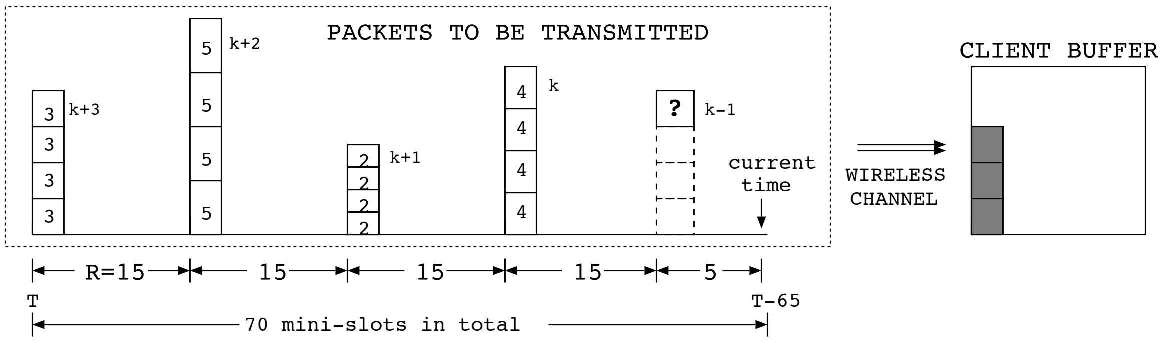


Figure 4: *Example of computing minimum success probability of a VBR video.*

## 5.2 Computing $P_{X_l,R}$

To complete the computation of success probability, the only thing remaining is to calculate $P_{X_l,R}$, the minimum success probability described above (Step 3). $P_{X_l,R}$ may generally be computed based on a given channel model or through actual measurement during the transmission. For the simulations to be described in the next section, we assume the channel may be characterized by the two state Markov chain shown in figure 5. Specifically, we assume in each mini-slot the wireless channel is in one of two possible states: GOOD (G) state and BAD (B) state. In state G, exactly one mini-packet can be transmitted in one mini-slot, while in state B, nothing gets transmitted. The channel may switch between these two states with certain probabilities as shown in the figure. For instance, $P_{GG}$ is the probability that the next mini-slot will also be in state G given that the current mini-slot is in state G. The other three probabilities $P_{GB}$, $P_{BG}$, and $P_{BB}$, are defined similarly. Based on the results in [12], we have

$$P_{X_l,R} = \sum_{i=X_l}^{R} \frac{D_{G,R}^{(i)}(0)}{i!} \tag{1}$$

for this channel model, where

$$D_{G,R}(z) = \left(\frac{1}{2} + \frac{2\overline{p} + zp - q}{2\sqrt{F}}\right)\left(\frac{E+\sqrt{F}}{2}\right)^R + \left(\frac{1}{2} - \frac{2\overline{p} + zp - q}{2\sqrt{F}}\right)\left(\frac{E-\sqrt{F}}{2}\right)^R$$

with $E = zp + q$ and $F = (zp - q)^2 + 4z\overline{p}\,\overline{q}$ both being functions of $z$. $p$ and $q$ are transition probabilities shown in figure 5. In a memoryless system, where $p = 1 - q$, equation (1) can be simplified to

$$P_{X_l,R} = \sum_{i=X_l}^{R} \binom{R}{i} p^i (1-p)^{R-i} \tag{2}$$

Having described the packet selection algorithm as well as how to compute the success probability, the next step is to evaluate the performance of the algorithm under various channel conditions.

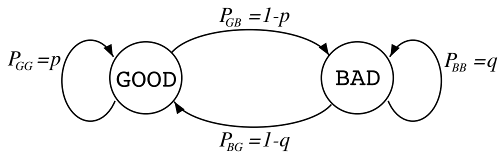


Figure 5: *Model of the two state Markovian channel.*

## 6 Performance study

The goal of our packet selection algorithm is to achieve better video quality over wireless links. Simulations were conducted to investigate the performance as well as possible improvements of the algorithm. In this section, we discuss some of the simulation results.

### 6.1 Performance measurement and simulation setup

To measure the performance of the algorithm, we introduce the concept of *quality index*. Assume every packet in the video stream is assigned a weight, generally the higher the packet's priority the higher its weight. We calculate the *quality* of a video stream played at the client as the summation of the weight over all the received packets. Similarly, after each simulation, given the actual sequence of channel states during the entire play-out, we can also compute the best quality that may be obtained for the same video during the same period. Finally, the *quality index*, $QI$, of the played video is given by the ratio of its quality to the computed best achievable quality[†]. Clearly, the higher the quality index, the better the algorithm performs. And the highest quality index an algorithm can achieve is 1. If a fixed number of packets are transmitted, the more high priority packets are included, the better the received video quality and the higher the quality index. Note that quality index is not to be interpreted as an indication of video quality, instead it measures how well the algorithm performs on the video for a given channel condition. In our simulations, we assume the weight of a layer $i$ packet is $N - i$, where $N$ is the total number of layers. For instance, for a video with three layers, layer 0 is assigned weight 3, layer 1 packet has weight 2, and layer 2 packet has weight 1.

The simulations were conducted using a 5000 frame long trace from the Terminator II video [21]. Frame sizes are different from frame to frame, and we assume each frame is divided equally into three layers. A smoothed transmission schedule with minimum peak rate is used by the server to schedule output packets. The resulting peak rate is 14k bits per frame. Data are delivered at this peak rate whenever they are not

[†]Our definition of *quality* is different from conventional measurements for video quality, such as PSNR. However, since our interest is in the performance of the packet selection algorithm, only the *difference* between the quality of the received video and the best one may be obtained with the same sequence of channel states is important. The result should be very similar had we chose to use PSNR.

going to cause overflow, otherwise a lower rate which will make the smoothing buffer just full is employed. A bandwidth of 14k bits per time slot is assumed to be reserved on the path from the server to the user. And the system load is calculated to be 90.6% if the wireless portion of the path is at G state during the entire play-out. Depending on the value of $p_{GG}$ and $p_{BB}$ used in a simulation, the actual load, which we only refer to as the load of the wireless link, is larger and can even be greater than one. In fact, situations where the load is close to or greater than one, are what we are most interested, since the packet selection algorithm will be most needed in these cases.

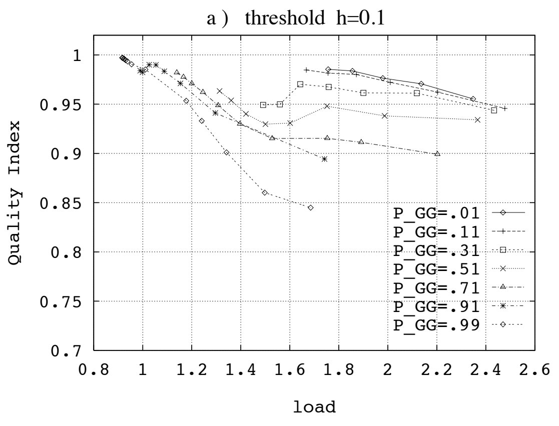
We assume the first 25 frames are preloaded to the client buffer before play-out. Once play-out starts, one frame is played at the beginning of every time slot, which is divided into 28 mini-slots. One mini-packet is sent over the wireless channel every mini-slot and one regular packet is sent from the video server every 4 mini-slots, where the size of a mini-packet is 500 bits and that of a regular packet is 2000 bits or lower depending on the actual transmission rate. As mentioned previously, the wireless channel is modeled as the general two state Markov chains shown in figure 5. We adjust the transition probabilities, $p_{GG}$ and $p_{BB}$, to different channel conditions and system loads. Eq. (1) and (2) are used to estimate the value of success probability with $R = 28$.

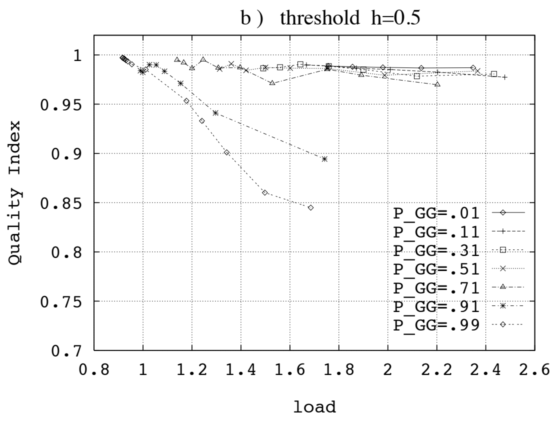
## 6.2 Performance vs. threshold

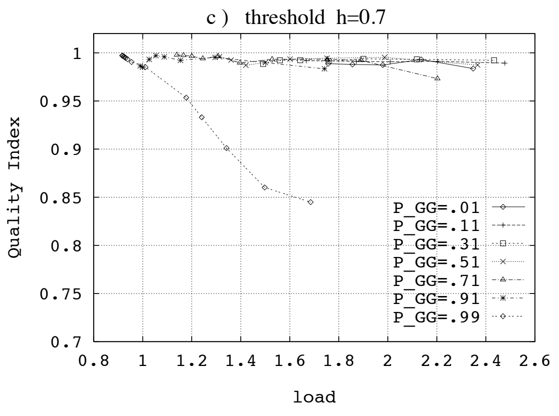
The performance of the proposed packet selection algorithm relies heavily upon the threshold used in the algorithm. In this section, we examine the relation between the two under various network conditions. Specifically, we vary the value of $p_{GG}$ from 0.01 to 0.91 in steps of 0.1 and we also consider the case with $p_{GG} = 0.99$. For a fixed $p_{GG}$, the value of $p_{BB}$ is adjusted to obtain different network loads.

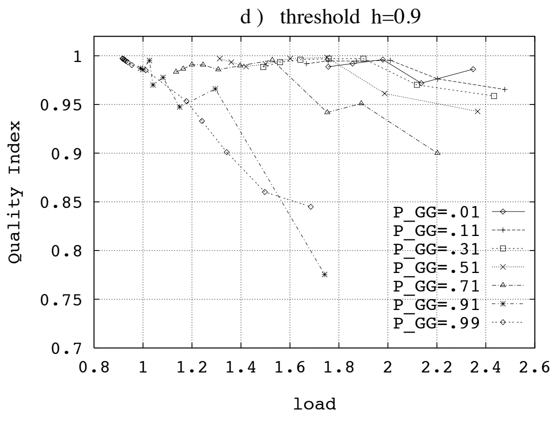
For each pair of $p_{GG}$ and $p_{BB}$, we investigate the algorithm performance by measuring the output video's quality index using five different thresholds 0.1, 0.5, 0.7, 0.9, and 0.99. Figures 6(a)-(e) show the results with the quality index plotted as a function of system load for different $p_{GG}$ and thresholds. Each graph corresponds to a fixed threshold, and for each curve in a graph, $p_{GG}$ is fixed.

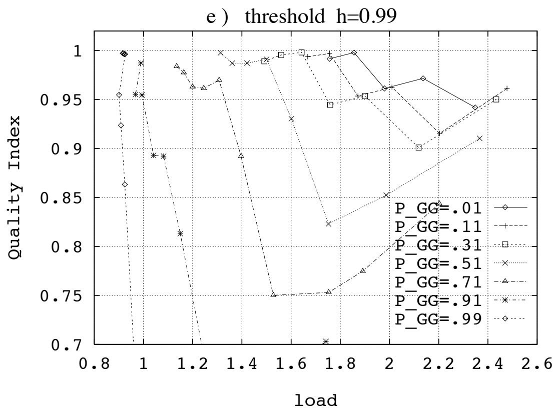
Comparing the five graphs in figure 6, we find that the quality index which indicates the algorithm performance, first increases as the threshold increases, then as the threshold approaches one, it starts to decrease. This is exactly the kind of result we expected. The reason is, with a small threshold value, the algorithm tends to transmit more packets for each frame before advancing to the next one. This means by the time it is ready to transmit later frames, it may not have enough time for even their high priority components. Therefore, some high priority packets may not be delivered because the system spent too much time on the earlier low priority packets. The performance of the algorithm is certainly not at its best











Trace file : Terminater II, 5000 frames.

Frame rate: 1 frame / time slot

1 time slot = 28 mini-slots.

Peak transmission rate after smoothing:

14000bits/frame.

Figure 6: *Quality index of the received Terminator II video under various channel conditions using different thresholds ( from the original algorithm ).*

in this case. As threshold value increases, less low priority packets are transmitted which leaves more room for higher priority packets behind them, thus the performance improves and the quality index increases. However, if the threshold continues to increase, the system approaches the other extreme state. It becomes more and more likely that only the highest priority layer will be sent, even if some less important packets could have been delivered without affecting the outcome of the future frames. As a result, the performance degrades with the increasing threshold value. In figure 6, the performance tops at around $h = 0.7$. The exact threshold value at which the algorithm performs the best depends upon channel conditions and may change for different video streams.

## 6.3 Improving the algorithm performance

With the packet selection algorithm we have been discussing so far, it is possible that the algorithm decides to skip the rest of a frame and start sending layer 0 of the next frame, but no packet of this new frame has yet arrived at the base-station. While waiting for these packets, the bandwidth, which is otherwise wasted, could have been used to deliver packets skipped by the algorithm. Evidently, the problem becomes more and more likely to occur as threshold increases, and it is also responsible to the rapid degradation of performance seen in figure 6 as threshold approaches one.

One way of improving the efficiency of bandwidth usage in this situation is to go back and transmit packets that were previously skipped by the algorithm. We call this "backing up" the transmission. Here, it is assumed that packets skipped by the algorithm are kept in the base-station buffer until after their play-out time. As soon as the awaited high priority packet arrives, the original algorithm is resumed and the new packet is delivered. Transmission may be backed up whenever similar situations occur. Figures 7(a)-(e) show the simulation results with the backup procedure incorporated, using the same set-ups as those in figure 6. The backup procedure always chooses, among all the queued packets, the one with highest priority and lowest index to deliver.

This new set of graphs shows that, at small threshold values, the backup scheme does not change the performance much. While for higher thresholds, the results are improved significantly. For some combinations of $P_{GG}$ and $P_{BB}$, the quality index obtained with $h = 0.9$ exceeds that when $h = 0.7$. Moreover, with this improved version, the performance also becomes more stable. In particular, it is within 1-3% of the optimum result over a relative large range, from $h = 0.7$ to $h = 0.9$. This makes it easier to select an appropriate threshold value for different video streams.

Comparing 7(a) with the other four graphs in figure 7, we find that very small threshold performs consistently bad. Therefore, systems which do not implement packet selection algorithm, which is equiv-

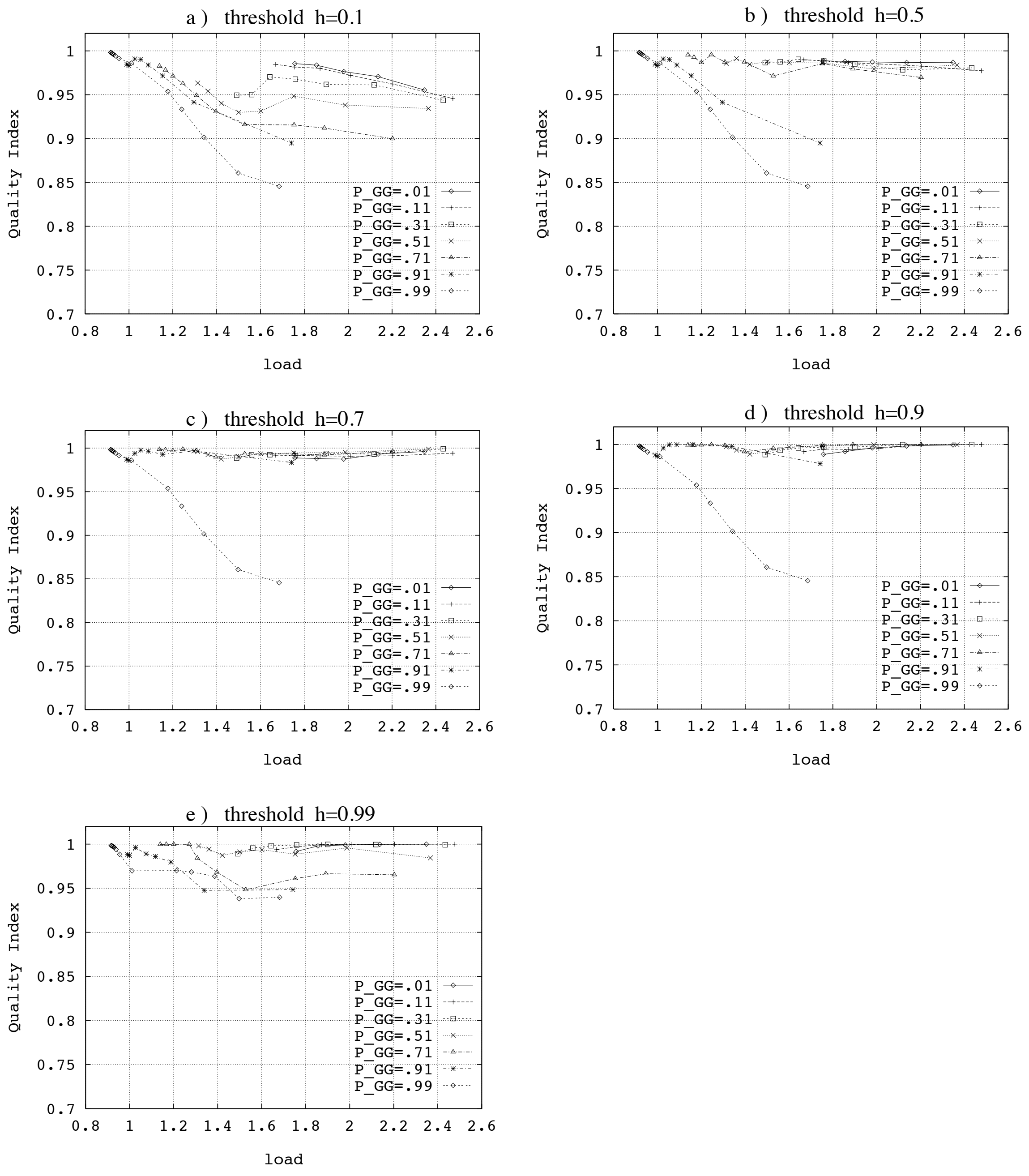


Figure 7: *Quality index of the received Terminator II video under various channel conditions using different thresholds ( from the modified algorithm with the backup procedure incorporated ).*

alent to the case of $h = 0$, do not perform well. The actual amount of improvement from implementing the packet select algorithm depends upon the channel conditions and the threshold value, and is generally higher for higher $P_{GG}$ and higher network load.

Another thing we noticed in figure 6 is that, at $h = 0.9$ and $h = 0.99$, as system load increases, the quality index drops sharply at some load values, and then goes back up as the load continues to increase. Further investigations show that this type of phenomenon occurs more often when there are fewer number of layers and for constant bit rate video [12]. In those cases, whether to include a layer or not makes a big difference on the success probability, thus causing an entire layer to be dropped as soon as $P_{BB}$ exceeds certain value. We also find that the problem diminishes when the backup procedure is adopted. Compared with figure 6, by incorporating the backup procedure into the packet selection algorithm, the sudden drops in performance almost completely disappear as shown in figure 7.

### 6.4 Performance, threshold, vs. channel characteristics

The wireless channel used in our simulation study can be fully characterized by the value of $P_{GG}$ and $P_{BB}$. In the last few subsections, we compared the performance of the algorithm with and without backup under different system loads. In this subsection, we break it down to see how the performance is affected by $P_{GG}$ and $P_{BB}$, as well as how to select the threshold to obtain better result.

With the original algorithm, figure 6 shows that, under the same load, performance is generally better for small $P_{GG}$, that is, when both good periods and bad periods are short; and worse for larger $P_{GG}$, i.e., when the system usually stays in the good state for a long time, but once in the bad state, it also takes a long time for it to switch back. However, with the backup procedure incorporated, the difference between various combinations of $P_{GG}$ and $P_{BB}$ becomes almost negligible, especially for $h = 0.7$ and $h = 0.9$ (figure 7). In other words, with a threshold between 0.7 and 0.9, the algorithm performs close to optimum under most of the channel conditions. The only exception is when $P_{GG}$ is very high, say above 0.99. Figure 7 shows that, at $P_{GG} = 0.99$, the packet selection algorithm almost brings no improvement to the performance unless the threshold is also set to be very high. In actual implementation, we may safely set the threshold between 0.7 and 0.9 to obtain near optimum performance in most cases, except for $P_{GG}$ close to 1, in which case the threshold also needs to be set to be very close to 1.

## 7 Conclusion

In this paper, we discussed how to reduce the impact of packet loss on video quality. Specifically, we studied a packet selection algorithm that selectively drops low priority packets to improve the overall quality of received video. The algorithm computes the probability of higher priority layers being delivered on time and transmits a packet only if this probability is greater than a given threshold. Simulation results show that by selecting the threshold appropriately, the original algorithm can achieve within 5% of the optimum result under most channel conditions. Moreover, by employing the backup procedure, which uses the otherwise idle network to deliver skipped packets, the performance may be further improved to within 1-3% of the optimum, and also becomes more stable within a larger range of threshold values and under different network conditions. This makes it easier to choose thresholds for different video.

Our packet selection algorithm decides which packets to transmit based on predictions about how layers in the future frames may be delivered over wireless links. Similar algorithm can be applied to the Internet, where bandwidth availability also changes with time. The algorithm may be implemented at either the video server or some intermediate network switches. One additional problem we must consider in the Internet environment is that the delay of the feedback will become more significant and must be taken into account by the algorithm for predicting future network conditions.

## References

[1] S. Acharya and B. Smith, "Compressed domain transcoding of MPEG," *Proceedings. IEEE International Conference on Multimedia Computing and Systems*, Austin, TX, June 1998, pp. 295-304.

[2] R. Aravind, M. R. Civanlar, and A. R. Reibmain, "Packet loss resilience of MPEG-2 scalable video coding algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.6, no.5, p.426-35, Oct. 1996.

[3] E. Amir, S. McCanne, and R. Katz, "Receiver-driven bandwidth adaptation for light-weight sessions," *Proceedings ACM Multimedia 97*, Seattle, WA, USA, Nov. 1997, pp. 415-26.

[4] B. Belzer, J. Liao, and J. D. Villasenor, "Adaptive video coding for mobile wireless networks," *Proceedings of 1st International Conference on Image Processing*, Austin, TX, USA, Nov. 1994, pp. 972-6.

[5] J.-C. Bolot, T. Turletti, "A Rate Control Mechanism for Packet Video in the Internet," *Proceedings IEEE INFOCOM '94*, Toronto, Canada, 12-16 Jun. 1994, pp. 1216-23.

[6] J.-C. Bolot and T. Turletti, "Adaptive Error Control For Packet Video in the Internet," *Proceedings of International Conference on Image Processing*, Lausanne, Switzerland, Sept. 1996, pp. 25-8.

[7] N. G. Duffield, K. K. Ramakrishnan, A. R. Reibman, "SAVE: an algorithm for smoothed adaptive video over explicit rate networks," *Proceedings of IEEE INFOCOM '98*, San Francisco, CA, USA, March 1998, pp. 1093-102.

[8] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, USA, April 1995, pp. 95-106.

[9] W. Feng, J. Rexford , "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," *IEEE INFOCOM '97 Proceedings,* Kobe, Japan, 1997, pp. 58-67.

[10] C.-Y. Hsu and A. Ortega, "Rate control for robust video transmission over wireless channels," *Proceedings of the SPIE*, San Jose, CA, USA, Feb. 1997, vol.3024, pp. 1200-11.

[11] S. Jacobs and A. Eleftheriadis, "Real-time dynamic rate shaping and control for Internet video applications," *1997 IEEE First Workshop on Multimedia Signal Processing*, Princeton, NJ, USA, June 1997, pp. 558-63.

[12] Z. Jiang, "Video Smoothing and Transmission of Smoothed Video Over Wireless Channels," *UCLA Ph.D Thesis*, 1998.

[13] H. Kanakia, P. P. Mishra, and A. R. Reibman, "An adaptive congestion control scheme for real time packet video transport," *IEEE/ACM Transactions on Networking*, Dec. 1995, vol.3, pp. 671-82.

[14] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Lecall, *MPEG Video Compression Standard*, Chapman & Hall, New York, NY, 1997.

[15] M. Nomura, T. Fujii, N. Ohta, "Basic characteristics of variable rate video coding in ATM environment," *IEEE Journal on Selected Areas in Communications*, June 1989, vol.7, pp.752-60.

[16] T. Ott, T. V. Lakshman, and A. Tabatabai, "A scheme for smoothing delay-sensitive traffic offered to ATM networks," *IEEE INFOCOM '92 Proceedings,* Florence, Italy, 4-8 May, 1992, pp. 776-785.

[17] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-area Networks," *Proceedings IEEE INFOCOM '94*, Toronto, Canada, 12-16 June 1994, pp. 680-8.

[18] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, April 1996, vol.6, :191-9.

[19] M. C. Yuang, P. L. Tien, and S. T. Liang, “Intelligent video smoother for multimedia communications,” *IEEE Journal on Selected Areas in Communications*, Feb. 1997, vol.15, pp. 136-46.

[20] Z. L. Zhang, J. Kurose, J. D. Salehi and D. Towsley, “Smoothing, statistical multiplexing and call admission control for stored video,” *IEEE Journal of Selected Areas in Communications*, Aug. 1997.

[21] Trace file of “Terminator II,” from ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/. The frame sizes are in BITS. Pattern: IBBPBBPBBPBB, 25 frames/second.

[22] ETSI. SMG2 EDGE 006/99, ”EDGE: concept proposal for enhanced GPRS”.