

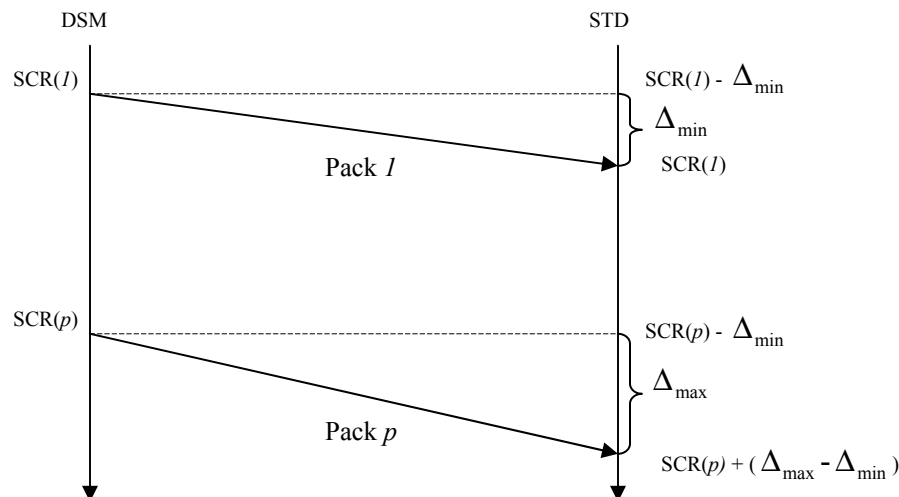
1) In transmission of MPEG-1 streams on a network with maximum and minimum delays of  $\Delta_{\max}$  and  $0$ , extra buffering is needed to avoid both starvation and overflow. Derive the buffering needed, clearly with help of figures.

To solve the problem of starvation and buffer overflow, we need a buffer of size  $2 \cdot (\Delta_{\max})$ .

For illustration, let suppose the maximum and minimum delays in transmission of MPEG-1 streams on a network be  $\Delta_{\max}$  and  $\Delta_{\min}$  respectively.

Buffer Underflow:

The worst case for buffer underflow (starvation) is when the first pack is transmitted with a minimum delay ( $\Delta_{\min}$ ) while the later pack  $p$  is transmitted with a maximum delay ( $\Delta_{\max}$ ). The following figure shows this scenario:

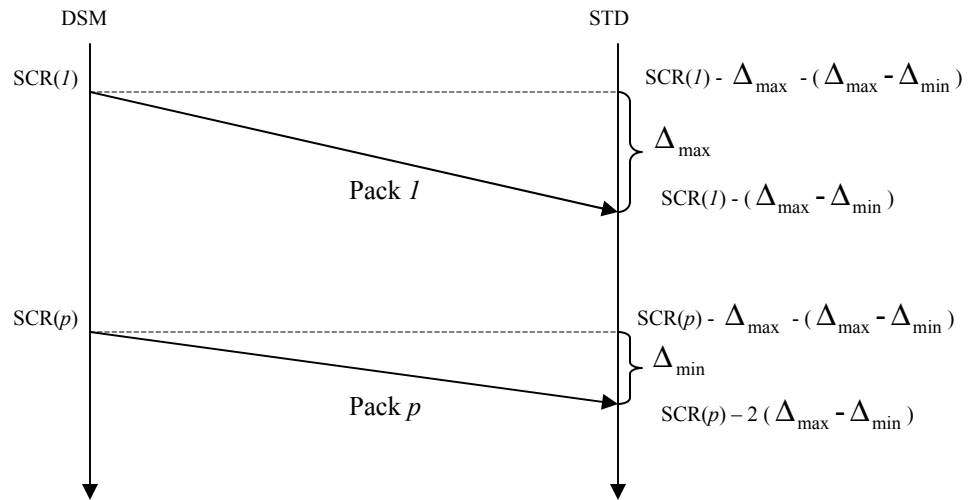


Suppose that the first pack is transmitted by the digital storage medium (DSM) when the DSM's clock show  $SCR(1)$  and the later pack  $p$  is transmitted by the DSM when the DSM's clock show  $SCR(p)$ . After a minimum delay, when the first pack arrives at the system target decoder (STD), the STD's clock is set to  $SCR(1)$ . The clock of the STD is thus slower than that of the DSM for  $\Delta_{\min}$ . Subsequently, when the pack  $p$  is transmitted from the DSM, the STD's clock would show  $SCR(p) - \Delta_{\min}$ . Because the pack  $p$  suffers from a maximum delay, it arrives at the STD when the STD's clock shows  $SCR(p) + (\Delta_{\max} - \Delta_{\min})$ . In order to prevent the "starvation" problem, the STD can start buffering

incoming packs and delay setting its clock to  $SCR(I)$  after waiting for  $(\Delta_{max} - \Delta_{min})$  after it receives the first pack.

### Buffer Overflow:

The worst case for buffer overflow happens when the first pack is transmitted with a maximum delay and the latter pack  $p$  is transmitted with a minimum delay. The following figure shows this scenario:



Assume that the first pack is transmitted by the DSM when the DSM's clock show  $SCR(I)$  and the later pack  $p$  is transmitted by the DSM when the DSM's clock show  $SCR(p)$ . After a maximum delay, when the first pack arrives at the system target decoder (STD), the STD's clock is set to  $SCR(I) - (\Delta_{max} - \Delta_{min})$  (remind that we delay setting the clock to solve the "starvation" problem). The clock of the STD is thus slower than that of the DSM for  $(2\Delta_{max} - \Delta_{min})$ . Subsequently, when the pack  $p$  is transmitted from the DSM, the STD's clock would show  $SCR(p) - (2\Delta_{max} - \Delta_{min})$ . Because the pack  $p$  suffers from a minimum delay, it arrives at the STD when the STD's clock shows  $SCR(p) - 2(\Delta_{max} - \Delta_{min})$ . Therefore, to prevent the "buffer overflow" problem, the STD needs a buffer whose size is at least  $(\Delta_{max} - \Delta_{min})$ .

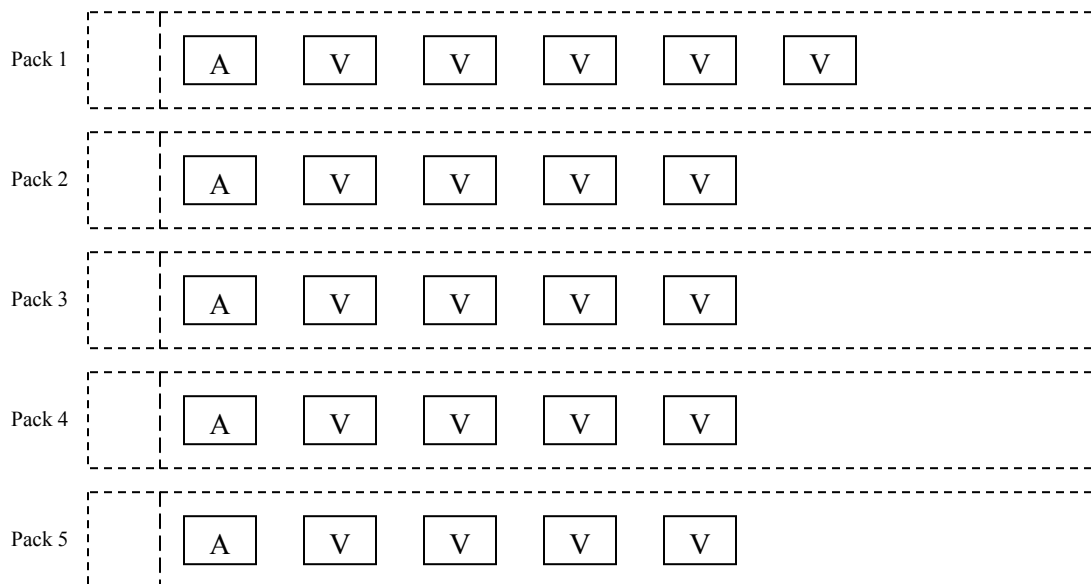
In this problem, the maximum and minimum delay for transmission in network is  $\Delta_{max}$  and 0 respectively. In conclusion, we require a buffer of size  $2 * (\Delta_{max})$ .

2) Describe the structure of packets and packs for a mpeg-1 stream. Clearly mention the various time stamps that are present and their purpose. Assume one packet of audio and corresponding number of video packets in one pack. Audio bandwidth is A and Video Bandwidth is V. Ignore bandwidth consumed by headers. Describe the sequence of first 26 packets when the ratio of V (video bandwidth) to A (audio bandwidth) is 4.2. You need to show the packet sequence (in terms of video or audio packet) with the PACK boundaries. How many bits are allocated to each of the timestamps and why?

In MPEG-1 stream, a pack consists of a pack header and the interleaving of multiple compressed streams. A system clock reference (SCR) is inserted into each pack header for the purpose of maintaining the continuity of playback. Each SCR value in a pack is the value of encoder's clock at the time the last byte of the SCR is constructed at the encoder. The SCR is used to ensure that media units are available at the decoder prior to their respective decoding time and that the consumption rate of packs at the STD matches the rate of transmission by the DSM. In addition, the first pack header may contain additional system information, such as rate bound, media lock flags, and needed media buffer size.

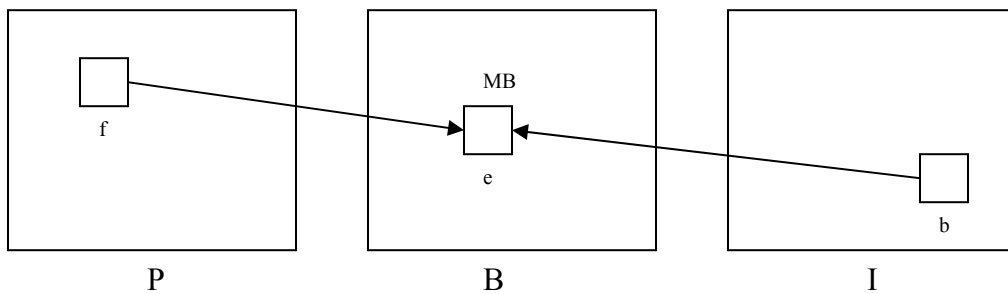
A packet is composed of a packet header followed by packet data. The packet header begins with a 32-bit code specifying what type of media the packet. The packer header may contain a decoding timestamp (DTS) and a presentation timestamp (PTS). The decoder will use the DTS to determine when to decode the frame and use the PTS to determine when to display the frame. The values of DTS and PTS for a packet refer to the first frame in that packet. Moreover, packet headers contain STD buffer size and scale fields, the product of which defines the maximum input size needed at the STD for the packet. All timestamps (SCR, DST and PST) are represented by 33-bit value. The reason for this is that the 33-bits clock is enough to encode a video whose length is 24 hours.

Because the ratio of video bandwidth to audio bandwidth is 4.2, we can write the sequence of the first 26 packets as shown below:



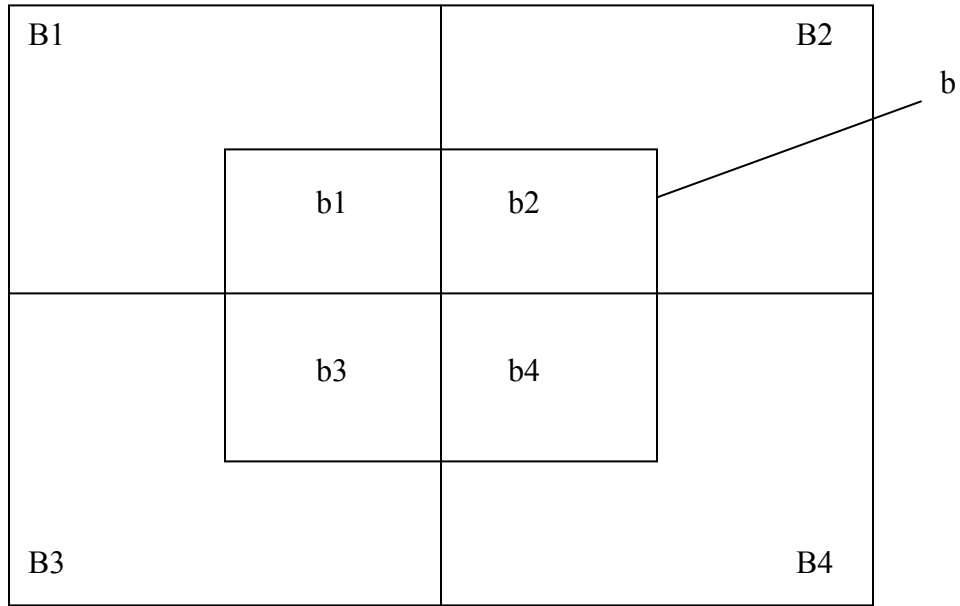
In the above figure,  $\boxed{A}$  represents an audio packet and  $\boxed{V}$  indicates a video packet. As can be seen, the first 26 packets (or 5 packs) comprise 5 audio packets and 21 video packets. If this sequence of 5 packs repeats, we can achieve 4.2:1 the ratio of V to A.

3) In the process of MPEG editing, you have to convert a B frame to a P frame. The B frame has two reference frames, a preceding P frame and a succeeding I frame. The first task in this is to remove the dependency between macroblocks within the B frame and 16x16 regions of the I frame (i.e., get rid of the backward motion vectors). Describe this process with figures for one macroblock of B using the following steps: first show a 16x16 reference region within the I frame from which a backward motion vector might exist for the macroblock being considered in the B frame. Then show why you need to find the DCT of this 16x16 reference region. Lastly, describe the sequence of operations in order to compute the DCT of that 16x16 region. We don't need the exact set of equations, but only the general sequence of operations at a high conceptual level to show that you understand how this is accomplished.



Recall that a motion vector for MB is stored as error (or difference) values  $e$  from the values of the predicted block  $f$  and  $b$ . For bidirectional predicted frame, the DCT values of the macroblock is  $(f+b)/2 + e$ . To convert B frame into P frame, we need to re-encode  $e$  as  $e' = ((b-f)/2 + e)$  so that the microblock can be reconstructed by adding  $f$  to  $e'$  while it gives the same value as the original prediction. Because DCT is close under addition operation, we can convert B frame to P frame using the above equation if we can find the DCT values of  $f$  and  $b$ . However,  $f$  and  $b$  might not lie on the 16\*16 macroblock boundary.

Consider the figure below. Assume that  $b$  lies among 4 macroblocks: namely B1, B2, B3 and B4. We already know the DTC values of B1, B2, B3 and B4 and we need to compute the DCT values of  $b$ . We claim that we can calculate the DCT values of  $b_1, b_2, b_3$  and  $b_4$  from the DCT values of B1, B2, B3 and B4 respectively. This can be done by first masking B1, B2, B3 and B4 and keeping only portions that are portions of  $b$ . Then multiply the macroblock by a matrix that has only 0's or 1's values. Next, we shift the portions that are parts of  $b$  into their correct positions in the matrix.



For example, we shall show we can find the DCT value of b1 from B1. Consider the figure below:

y	y	y	y
y	y	y	y
y	Y	b11	B12
y	Y	b13	B14

B1

The region composed of b11, b12, b13 and b14 is the portion that we want to maintain while the region where y's lie in is the portion that we don't want. We can do masking as follows:

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}
 \times
 \begin{array}{|c|c|c|c|} \hline y & y & y & y \\ \hline y & y & y & y \\ \hline y & y & b11 & b12 \\ \hline y & y & b13 & b14 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline y & y & b11 & b12 \\ \hline y & y & b13 & b14 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline y & y & b11 & b12 \\ \hline y & y & b13 & b14 \\ \hline \end{array}
 \times
 \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & b11 & b12 \\ \hline 0 & 0 & b13 & b14 \\ \hline \end{array}$$

Then, we shift the region that is a portion of b1 into correction position by multiplying the resulting matrix as follows:

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & b_{11} & b_{12} \\ \hline 0 & 0 & b_{13} & b_{14} \\ \hline \end{array} \times \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline b_{11} & b_{12} & 0 & 0 \\ \hline b_{13} & b_{14} & 0 & 0 \\ \hline \end{array}$$
  

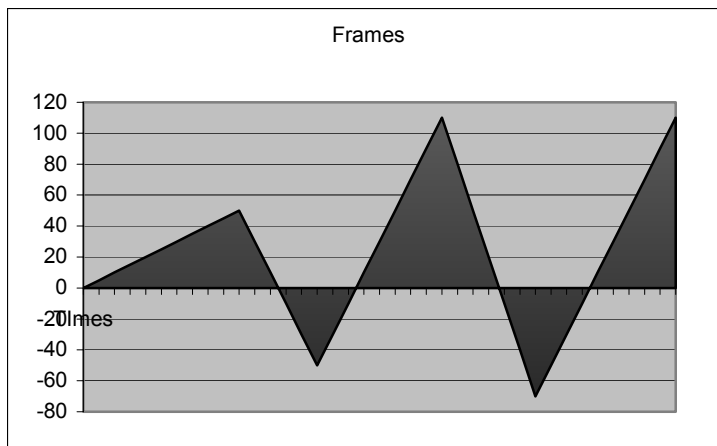
$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline b_{11} & b_{12} & 0 & 0 \\ \hline b_{13} & b_{14} & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b_{11} & b_{12} & 0 & 0 \\ \hline b_{13} & b_{14} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Lastly, we combine each portion by adding their DCT blocks to get the DCT values of the entire region.

4) What is the need for buffering in multimedia storage and retrieval from disk? How do calculate it for an arbitrary placement of media blocks on disk? Clearly show using figures and graphs.

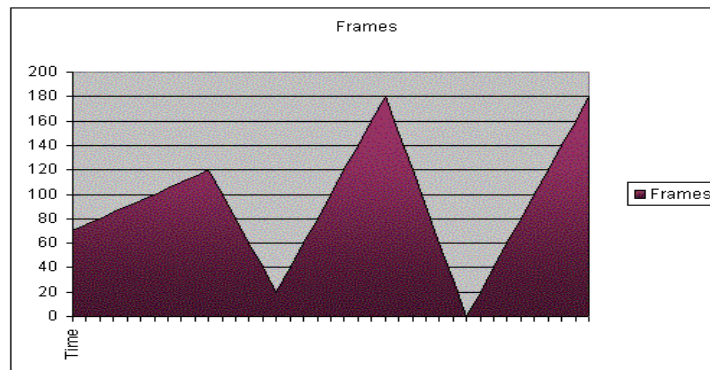
In arbitrary placement of media blocks on disk, the blocks are placed on the disk without any structure. Consecutive blocks might be placed far apart, yielding an arbitrary seeking time for a reading head. Therefore, the server can't stream a media continuously. One possible way to solve this problem is to pre-fetch the adequate amount of media content and store it on the buffer before playing the media. To determine how much read-ahead media content is needed, we shall illustrate it using the figures and graphs below:

Suppose we can characterize the nature of a given multimedia stream as the following graph. In the graph, the horizontal axis represents the time and the vertical axis indicates the difference between the total number of frames produced by the server and the total number of frames consumed by the user.

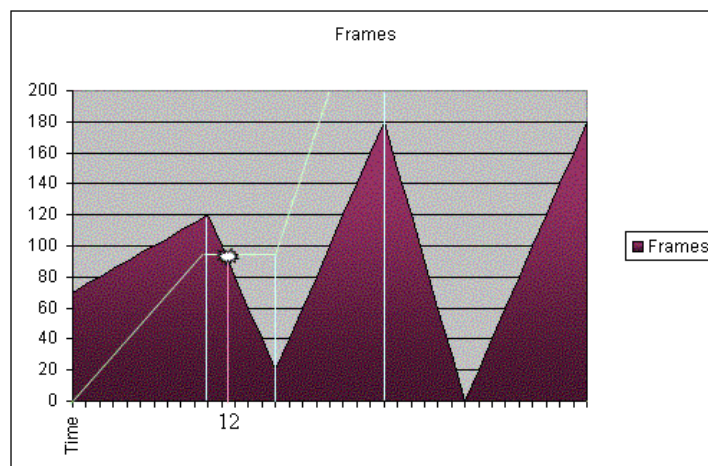


As can be seen from the above figure, at some points the total number of frames produced by the server minus the total number of frames consumed by the user drops below zero. This indicates that at those times the display device has no media content to display, yielding media discontinuity.

Consider the secure figure where we lift the graph so that during the playing period the total number of frames produced by the server minus the total number of frames consumed by the user never drops below zero.



This above graph is what we try to achieve because the display device always has enough amount of media content to play. The way we can lift the graph is to buffer enough media content before starting playing the media. In order to determine how much media content we need to buffer, we draw a new line on the previous figure, showing the number of frames produced by the server without being subtracted by the total number of frames consumed by the user. The intersection point between this line and the graph in the above figure will give the amount of needed buffering. After stopping the perfecting, the total number of frames produced by the server minus the total number of frames consumed by the user will follow the graph in the above figure and never drop below zero.

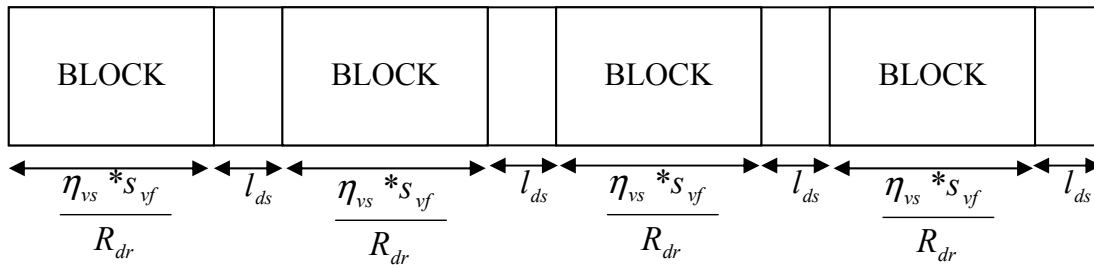


5) Derive the formula for calculating the length of media blocks and separation between them in constrained placement of blocks on disk.

We first define the following notations, which we will use for this question.

Symbol	Explanation	Unit
$l_{ds}$	Scattering parameter	sec
$\eta_{vs}$	Granularity of video storage	frames/block
$s_{vf}$	Size of a video frame	bits/frames
$R_{dr}$	Rate of data transfer from disk	bits/sec
$R_{vd}$	Rate of video display	bits/sec
$R_{vr}$	Video recording rate	frames/sec

In constrained placement, blocks on disk need not to be placed immediately after one that precedes it. However, the space between two consecutive blocks is bound by some value so that the server can read and playback them media continuously.



For the sake of simplicity, assume that reading and displaying operations can be done in parallel.

In order to play a media without discontinuity, the media access time must be lower than the media playback time. Hence, we have:

$$\text{media access time} \leq \text{media playback time}$$

$$\text{block seeking time} + \text{block access time} \leq \text{block playback time}$$

$$\text{block seeking time} + (\text{size of block})/(\text{block play rate}) \leq (\text{size of block})/(\text{block playback rate})$$

$$l_{ds} + \frac{\eta_{vs} * s_{vf}}{R_{dr}} \leq \frac{\eta_{vs}}{R_{vr}}$$

6) What is the maximum number of users that can be supported by a disk with transfer bandwidth of 20MBytes/s and maximum seek time of 10ms? Assume multimedia files are stored in constrained manner on the disk, with block size = 2 frames = 40KBytes, separation between blocks of 1ms. Assume playback rate of 30 frames/sec.

Let  $\beta$  be the average scattering,  $\gamma$  be the play back time and n be the number of user. We have the following formula for admission control:

$$n_{\max} = \left\lceil \frac{\gamma}{\beta} - 1 \right\rceil$$

, where  $\gamma$  = playback time = (second/1 block) = (second/ 2 frames) = 2/30 = 66.67 ms  
and  $\beta$  = average scattering = 1ms + 40KB/20MBps = 3 ms. Therefore, we have

$$n_{\max} = \left\lceil \frac{66.67}{3} - 1 \right\rceil = 22$$