Chapter 9 - Stored Streaming Applications

As mentioned in previous chapters, there is a fundamental difference between delivering interactive live video conference streams and delivering stored video across networks. With live video conferencing, the main goal was to establish smooth playback while keeping latency to less than 250 milliseconds. For stored video streams, the main goal is to have the video at the receiver any time before it is needed for display.

The "just-in-time" policy and the stored nature of video have a number of implications on supporting its delivery to a user. First, just-in-time delivery means that the sender can send the data arbitrarily far in advance as long as it knows about the request. This means that scheduling can be done ahead of time, even if on only a limited basis. Second, a buffer at the receiver can be used to smooth the resource requirements out over time, particularly for the initial playback delay. Third, stored video, unless CBR encoded, will typically have greater bandwidth variation requirements over time. Because the scenes of the clip are commonly put together (e.g., in a movie), there will be greater variation in content resulting in greater variability in the compressed data. Finally, the data is already captured and stored. For streaming across reservation-based networks, analyzing the data ahead of time can be done and can simplify delivery because no compression needs to be done. For streaming across best-effort networks, while one can analyze the data ahead of time to make plans, the video may need to be altered to fit within the available network resources. This latter step can be increase the load on a server because it will need to take a compressed stream and generate another stream out of it to fit within the available network resources.

In the remainder of this chapter we will examine techniques for delivering stored video data across both reservation-based and best-effort networks.

9.1 Reservation-based Stored Video Streaming

Reservation-based stored video streaming is, perhaps, one of the easiest forms of video delivery to manage because both the network bandwidth is well known (i.e., reserved) and the video is stored so that its network characteristics are also well known.

9.1.1 A Naïve Streaming Model

For our video streaming model, we assume that the video is encoded at 30 frames per second and that we will be reserving network bandwidth on in (bytes/frame). Multiplying this number by 30 will give us the actual bandwidth requirement. A simple model to send the stored video stream across the networks is to simply send one frame at a time. After a small amount of delay added at the receiver to manage the small network delay jitter, the frame can be played back. Thus, if we have the following:

```
Frames      3    5    6    3    2    3    7    6    7

Network      3    5    6    3    2    3    7    6    7

Playback      3    5    6    3    2    3    7    6    7
```

Here, the individual video frames are 3, 5, 6, etc. bytes. The network bandwidth used is then exactly the same amount required per frame. Upon receiving the data, the frame is played back.

Obviously, reserving network bandwidth on a per frame basis does not make sense and is typically allocated as a constant amount of bandwidth. Thus, if we reserve network bandwidth at the maximum rate (7 bytes per frame), it would look like this:

```
Frames      3    5    6    3    2    3    7    6    7

Network     7    7    7    7    7    7    7    7    7
Sent        3    5    6    3    2    3    7    6    7

Playback     3    5    6    3    2    3    7    6    7
```

Here, the network bandwidth is reserved at 7 bytes / frame. The sender, however, only sends the frame that needs to be sent. Obviously, the disadvantage of this approach is that the network bandwidth is over allocated (i.e., there is more bandwidth reserved than used). The advantage, however, is that buffering on the receiver is minimized. Given the fact that buffering can be relatively cheap to add and taking advantage of the just-in-time delivery semantics, sending data ahead at the maximum reserved rate is wise to do. Thus, the server could do the following:

```
Frames      3    5    6    3    2    3    7    6    7

Network     7    7    7    7    7    7    stop transmission
Sent        7    7    7    7    7    7    0    0    0

Playback     3    5    6    3    2    3    7    6    7
```

Allocating network resources at the maximum frame size's rate would seem to be overkill. As a example, we have added the buffer size to the previous example below:

```
Frames      3    5    6    3    2    3    7    6    7

Network     7    7    7    7    7    7    stop transmission
Sent        7    7    7    7    7    7    0    0    0

Playback     3    5    6    3    2    3    7    6    7
Buffer        4    6    7   11   16   20   13   7    0
```

As shown in this example, the buffer continues to grow until the transmission is stopped (at frame 7). Therefore, figuring out a smaller rate which still provides continuous playback can be useful in (i) reducing the overall bandwidth requirement for the stream and (ii) reduce the amount of buffer needed on the client.

9.1.2 The Minimum Streaming Bandwidth

One question that can arise is what is the minimum amount of bandwidth necessary in order to (i) provide continuous playback and (ii) minimize the latency in startup delay. For the previous example, we can potentially reduce the rate. For example, 6, 5, and 4 are shown below:

```
Frames      3    5    6    3    2    3    7    6    7

Network     6    6    6    6    6    6    6    6    6
Sent        6    6    6    6    6    6    6    0    0

Playback     3    5    6    3    2    3    7    6    7
Buffer        3    4    4    7   11   14   13   7    0
```

For 5:

Frames	3	5	6	3	2	3	7	6	7
Network	5	5	5	5	5	5	5	5	2
Sent	5	5	5	5	5	5	5	5	2
Playback	3	5	6	3	2	3	7	6	7
Buffer	2	2	1	3	6	8	6	5	0

For 4:

Frames	3	5	6	3	2	3	7	6	7
Network	4	4	4	4	4	4	4	4	4
Sent	4	4	4	4	4	4	4	4	4
Playback	3	5	6	3	2	3	7	6	7
Buffer	1	0	**-2**	**-1**	1	2	**-3**	**-2**	**-5**

Note: negatives indicate buffer underflow!

These examples give us a bit of insight into how to figure out the minimum rate. At rate 6, the buffer continues to grow as in the bandwidth = 7 case. In 4, the buffer runs out on the 3rd frame. The main observation is that assuming a constant bandwidth requirement, the minimum rate needed to get to any point, *i*, in the movie is the sum of all the frames to *i* divided by *i* (i.e., the constant rate):

$$\text{Equation: } \text{sum}(0\text{-}i) / I \qquad (1)$$

This, however, does not ensure that buffer underflow does not occur before frame *i.* For example, consider the first 6 frames of the video sequence. Here, the total number of bytes needed to be delivered by frame 6 is (3 + 5 + 6 + 3 + 2 + 3 = 22) 22 bytes. This corresponds to an average rate of 22/6 = 3.67. Thus, if we use a rate of 4, 24 bytes are delivered by frame 6, allowing it to be played back properly. *However*, as shown in the example buffer underflow occurs at frame 3. This is because for frame 3, (1) requires ((3 + 5 + 6)/3) bytes/frame (or 4.67 bytes/frame).

In order to ensure continuous playback then, what we are really interested in is the maximum of all the average rates required from (1) or:

$$\text{Equation:} \max_{(0<=i<=N)} \text{sum}(0\text{-}i) / I \qquad (2)$$

By calculating (2), we can ensure minimal start up delay as well as guarantee the minimum required bandwidth.

9.1.3 The Minimum Streaming Bandwidth with Prefetching

While the equation (2) above can find the minimum amount of bandwidth required while also minimizing start-up delay, one problem we run into is if the initial set of frames is quite large. For example, consider the following sequence:

Frames	9	3	5	6	3	2	3	7	6

Using (2) above, we end up requiring 9 bytes /frame in order to deliver the first frame of video across the network. One potential way to mitigate this problem is to start the video

slightly delayed. For example, suppose 2 additional frames of delay are inserted before the beginning of the move, then the large first frame can be spread out and its bandwidth requirement reduced. Using our previous notation:

Frames	9	3	5	6	3	2	3	7	6		
Network	4	4	4	4	4	4	4	4	4	4	4
Sent	4	4	4	4	4	4	4	4	4	4	4
Playback			9	3	5	6	3	2	3	7	6
Buffer	4	8	3	4	3	1	2	4	5	2	0

Even with a bandwidth=4 allocation, we are able to play this stream back without interruption with just a little start up delay.

With the minimum bandwidth streaming equations, we can simply modify them to allow us to include prefetching time:

EQUATION

Other calculations are also possible. For example, given a fixed rate bandwidth allocation, one can determine how much prefetch is needed in order to achieve continuous playback. To do so, we first calculate the maximum amount of buffer underflow that would occur using no start up delay and the fixed rate. Once determined, we then need to simply buffer enough to make up this difference:

9.1.4 Managing Reservations for Stored Video Streaming

So far, we determined how to calculate the minimum amount of bandwidth necessary to deliver a stored video stream without and with prefetching delay. We have also showed how to calculate how much initial playback delay is necessary in order to achieve continuous playback using a fixed bandwidth reservation rate. All of the rates are fixed for the duration of the video stream.

Another observation to make is that in the calculation of equation (2) and (4), there is a single point that causes the bandwidth to be determined (i.e., the *j* in which the max occurred). After this point, however, it can be shown that the bandwidth requirement from that point to the end of the movie must be less than or equal to the bandwidth from the beginning of the movie to *j*.

9.2

Chapter 10 - Video Distribution Systems