Multimedia Systems

In earlier chapters, we generally concerned ourselves with how operating systems handle conventional data, such as text files, programs, binaries, word-processing documents, and spreadsheets. However, operating systems may have to handle other kinds of data as well. A recent trend in technology is the incorporation of **multimedia data** into computer systems. Multimedia data consist of continuous-media (audio and video) data as well as conventional files. Continuous-media data differ from conventional data in that continuous-media data—such as frames of video—must be delivered (streamed) according to certain time restrictions (for example, 30 frames per second). In this chapter, we explore the demands of continuous-media data. We also discuss in more detail how such data differ from conventional data and how these differences affect the design of operating systems that support the requirements of multimedia systems.

20.1 Internet protocols such as IP and TCP do not typically reserve resources during connection setup time at the intermediate routers. Consequently, during congestion, when the buffers in routers fill up, some of the packets might be delayed or lost, thereby violating any quality of service requirements. Congestion losses could be avoided if the Internet uses circuit switching and reservation of buffer resources to ensure that packets are not lost.

20.2 Unicasting is the situation where a server sends data to only one client. If the content is required by multiple clients and if unicast was the only mechanism available, then the server would have to send multiple unicast streams to reach the different clients. Broadcasting allows a server to deliver the content to all of the clients irrespective of whether they wish to receive the content or not. This technique could result in unnecessary work for those clients that do not need the content but still get the broadcast data. Multicasting is a reasonable compromise where the server can send data to some subset of the clients and requires support from the network router to intelligently duplicate the streams at those points in the network where the destination clients are attached to different sub networks.

20.3 The bit rate required to support uncompressed video is 352*240*30*3/(1024*1024) MB/s = 7.2509 MB/s. The required compression ratio is therefore 7.2509*8/1.5 = 38 : 1.

20.4 The bandwidth required for uncompressed data is $30 \times 640 \times 480 \times 3$ bytes per second assuming 640×480 frames at a rate of 30 frames per second. This works out to about 26 MB/s. If the file is compressed by a ratio of 200 : 1, then the bandwidth requirement drops to 135 KB/s.

20.5 The sizes of the images, video, and audio before compression are 33.33 MB, 2.75 MB, and 0.8 MB respectively.

20.6 Both delay and jitter are important issues for live real-time streaming. The user is unlikely to tolerate a large delay or significant jitter. Delay is not an important issue for on-demand real-time streaming as the stream does not contain live clips. These applications can also tolerate jitter by buffering a certain amount of data before beginning the playback. In other words, jitter can be overcome by increasing delay, and since delay is not an important consideration for this application, the increase in delay is not very critical.

20.7 Progressive download is the situation where a media file is downloaded on demand and stored on the local disk. The user is able to play the media file as it is being downloaded without having to wait for the entire file to be accessed. Real-time streaming differs from progressive downloads in that the media file is simply streamed to the client and not stored on the client disk. A limited amount of buffering might be used to tolerate variances in streaming bandwidth, but otherwise the media file is played and discarded without requiring storage.

20.8 Batch 1 (R1, R4, R5), batch 2 (R6, R9), and batch 3 (R2, R3, R7, R8, R10). Within batch 1, requests are scheduled as: (R5, R1, R4). Within batch 2, requests are scheduled as: (R9, R6). Within batch 3, requests are scheduled as: (R2, R8, R7, R3, R10).

20.9 The batches are as follows: Batch 1: (R1), Batch 2: (R4, R5, R6, R9), Batch 3: (R10), Batch 4: (R2, R3, R7, R8).

20.10 Cineblitz differentiates clients into two classes: those that require real-time service and those do not. The resources are allocated such that a fraction of it is reserved for realtime clients and the rest are allocated to non-realtime clients. Furthermore, when a client requires realtime service enters the system, it is admitted into the system only if there are sufficient resources to service the new client. In particular, when a client makes a request, the system estimates the service time for the request and the request is admitted only if the sum of the estimated service times for all admitted requests does not exceed the duration of service cycle T.

20.11 If P_1 were assigned a higher priority than P_2, then the following scheduling events happen under rate-monotonic scheduling. P_1 is scheduled at $t = 0$, P_2 is scheduled at $t = 25$, P_1 is scheduled at $t = 50$, and P_2 is scheduled at $t = 75$. P_2 is not scheduled early enough to meet its deadline. If P_1 were assigned a lower priority than P_2, then the

following scheduling events happen under rate-monotonic scheduling. P_2 is scheduled at $t = 0$, P_1 is scheduled at $t = 30$, which is not scheduled early enough to meet its deadline. The earliest-deadline schedule performs the following scheduling events: P_1 is scheduled at $t = 0$, P_2 is scheduled at $t = 25$, P_1 is scheduled at $t = 55$, and so on.