

- b. For each time slot indicate which packets appear on the output after the token(s) have been removed from the queue. Thus, for the  $t = 0$  time slot in the example above, packets 1 and 2 appear on the output link from the leaky buffer during slot 0.
- P21. Repeat P20 but assume that  $r = 2$ . Assume again that the bucket is initially full.
- P22. Consider P21 and suppose now that  $r = 3$ , and that  $b = 2$  as before. Will your answer to the question above change?
- P23. Consider the leaky-bucket policer that polices the average rate and burst size of a packet flow. We now want to police the peak rate,  $p$ , as well. Show how the output of this leaky-bucket policer can be fed into a second leaky bucket policer so that the two leaky buckets in series police the average rate, peak rate, and burst size. Be sure to give the bucket size and token generation rate for the second policer.
- P24. A packet flow is said to conform to a leaky-bucket specification  $(r,b)$  with burst size  $b$  and average rate  $r$  if the number of packets that arrive to the leaky bucket is less than  $rt + b$  packets in every interval of time of length  $t$  for all  $t$ . Will a packet flow that conforms to a leaky-bucket specification  $(r,b)$  ever have to wait at a leaky bucket policer with parameters  $r$  and  $b$ ? Justify your answer.
- P25. Show that as long as  $r_1 < R w_1 / (\sum w_j)$ , then  $d_{\max}$  is indeed the maximum delay that any packet in flow 1 will ever experience in the WFQ queue.



## Programming Assignment

In this lab, you will implement a streaming video server and client. The client will use the real-time streaming protocol (RTSP) to control the actions of the server. The server will use the real-time protocol (RTP) to packetize the video for transport over UDP. You will be given Python code that partially implements RTSP and RTP at the client and server. Your job will be to complete both the client and server code. When you are finished, you will have created a client-server application that does the following:

- The client sends SETUP, PLAY, PAUSE, and TEARDOWN RTSP commands, and the server responds to the commands.
- When the server is in the playing state, it periodically grabs a stored JPEG frame, packetizes the frame with RTP, and sends the RTP packet into a UDP socket.
- The client receives the RTP packets, removes the JPEG frames, decompresses the frames, and renders the frames on the client's monitor.

The code you will be given implements the RTSP protocol in the server and the RTP depacketization in the client. The code also takes care of displaying the transmitted video. You will need to implement RTSP in the client and RTP server. This programming assignment will significantly enhance the student's understanding of RTP, RTSP, and streaming video. It is highly recommended. The assignment also suggests a number of optional exercises, including implementing the RTSP DESCRIBE command at both client and server. You can find full details of the assignment, as well as an overview of the RTSP protocol, at the Web site <http://www.awl.com/kurose-ross>.