Multimedia Networking

Annalise Tarhan

January 14, 2021

1 Consider the figure below.

1.1 Suppose that the client begins playout as soon as the first block arrives at t_1. How many blocks of video will have arrived at the client in time for their playout? Explain.

Packet	Arrival Time
0	t_0
1	t_1+
2	t_2+
3	t_2+
4	t_3+
5	t_4+
6	t_8+

Only the first, fourth, fifth, and sixth packets arrive in time for playout, since each must arrive at a time interval number less than their packet number in the chart above.

1.2 Suppose now that the client begins playout at $t_1 + \Delta$. How many blocks of video will have arrived at the client in time for their playout? Explain.

All of the first six packets arrive in time, since each must now arrive at a time interval number less than or equal to their packet number plus one in the chart above. Only the last packet misses its playout time.

1.3 In the same scenario as the previous part of problem, what is the largest number of blocks that is ever stored in the client buffer awaiting playout? Explain.

Two. Just before t_3 there are two packets awaiting playout, the third and fourth. After the fifth packet arrives between t_3 and t_4, there are again two packets waiting. At all other times, there is only one or no packets waiting.

1.4 What is the smallest playout delay at the client such that every video block has arrived in time for its playout? Explain.

The constraint is the most delayed packet, which happens to be the last one. It arrives just after t_9, so it cannot be played until t_{10}. Counting back, the first packet cannot be played until t_4.

2 Recall the simple model for HTTP streaming in figure 9.3. Recall that B denotes the size of the client's application buffer and Q denotes the number of bits that must be buffered before the client application begins playout. Also r denotes the video consumption rate. Assume that the server sends bits at a constant rate x whenever the client buffer is not full.

2.1 Suppose that $x < r$. As discussed in the text, in this case playout will alternate between periods of continuous playout and periods of freezing. Determine the length of each as a function of Q, r, and x.

Playout begins when there are Q bits in the buffer and the buffer is drained at a rate of $r - x$. $\quad Q - (r - x)t = 0 \quad \rightarrow \quad t = \frac{Q}{r-x}$
When the buffer is empty, it will refill at rate x until it contains Q bits.
$0 + xt = Q \quad \rightarrow \quad t = \frac{Q}{x}$

The video plays for $\frac{Q}{r-x}$ time units and freezes for $\frac{Q}{x}$.

2.2 Now suppose that $x > r$. At what time $t = t_f$ does the client application buffer become full?

As we saw, it takes $t_1 = \frac{Q}{x}$ for the buffer to become full enough to start playback. From there, it fills more slowly, at a rate of $x - r$, until it fills up all B bits of the buffer. The time it takes to fill from Q to B is t_2, where $(x - r)t_2 = B - Q$. $t_2 = \frac{B-Q}{x-r}$. The total time it takes for the buffer to fill is $t_1 + t_2 = \frac{Q}{x} + \frac{B-Q}{x-r}$.

3 Recall the simple model for HTTP streaming shown in figure 9.3. Suppose the buffer size is infinite but the server sends bits at variable rate $x(t)$. Specifically, suppose $x(t)$ has the following saw-tooth shape. The rate is initially zero at time $t = 0$ and linearly climbs to H at time $t = T$. It then repeats this pattern again and again.

3.1 What is the server's average send rate?

$\frac{H}{2}$

3.2 Suppose that $Q = 0$, so that the client starts playback as soon as it receives a video frame. What will happen?

Unless r is extremely low, the slow rate of arrival will cause playback to freeze after every frame until rate increases sufficiently. After the rate drops back down to 0, it will return to freezing after every frame.

3.3 Now suppose $Q > 0$ and $HT/2 \geq Q$. Determine as a function of Q, H, and T the time at which playout first begins.

Since the rate at which bits are sent is linear, $x(t) = ct$ where c is a constant. (As long as $t \leq T$, which is all we are concerned with in this part of the problem.) At $t = T$, $ct = H$, so $c = \frac{H}{T}$. Then $x(t) = \frac{Ht}{T}$. We want to find t where the cumulative number of bits arrived equals Q, so find the integral of $x(t)$, $\int \frac{Ht}{T} = \frac{H}{T} \int t = \frac{H}{T} \frac{t^2}{2}$.

Set the integral equal to Q and solve for t:
$Q = \frac{H}{T} \frac{t^2}{2} \rightarrow t^2 = \frac{2QT}{H} \rightarrow t = \sqrt{\frac{2QT}{H}}$.

3.4 Suppose $H > 2r$ and $Q = \frac{HT}{2}$. Prove there will be no freezing after the initial playout delay.

The client will start playback when the buffer contains $\frac{HT}{2}$ bits, which occurs at the end of the first sawtooth pattern. It is a given that $H > 2r$, or $r < \frac{H}{2}$. The number of bits played out during the second sawtooth pattern is rT, and by the given, $rT < \frac{HT}{2}$. So, the bits in the buffer at the end of the first sawtooth still

won't be exhausted by the end of the second, and it will have been replenished by another $\frac{HT}{2}$ bits. Since there will be no freezing during the second sawtooth and the third sawtooth begins with an even larger buffer, there will be no further freezing as long as the same pattern continues.

3.5 Suppose $H > 2r$. Find the smallest value of Q such that there will be no freezing after the initial playback delay.

Since $H > 2r$, the rate of playback is less than the average rate bits are being sent, so in the long run there will be no freezing. What we are concerned with is making sure there are enough bits buffered during the first sawtooth so that there is no further freezing. To start, the formula for the cumulative number of bits sent is $\frac{H}{T}\int_0^t t = \frac{H}{T}\frac{t^2}{2} = \frac{Ht^2}{2T}$. To account for the number of bits consumed during playout, this becomes $\frac{Ht^2}{2T} - rt + Q$, which of course only applies after Q bits have been buffered. To find the time when the number of bits hits a minimum, take the derivative and set it equal to zero.

$\frac{Ht^2}{2T} - rt + Q\ dt = 0$
$\frac{H}{2T}(t^2\ dt) - (rt\ dt) = 0$
$\frac{Ht}{T} - r = 0$
$t = \frac{rT}{H}$

This means that the number of bits at time t, the minimum, is $\frac{H(\frac{rT}{H})^2}{2T} - r(\frac{rT}{H}) + Q$. Since we want to minimize the buffer, set equal to zero and solve for Q.

$\frac{H(\frac{rT}{H})^2}{2T} - r(\frac{rT}{H}) + Q = 0$
$\frac{\frac{r^2T^2}{H}}{2T} - \frac{r^2T}{H} + Q = 0$
$Q = \frac{r^2T}{H} - \frac{r^2T}{2H}$
$Q = \frac{r^2T}{2H}$

3.6 Now suppose that the buffer size B is finite. Suppose $H > 2r$. As a function of Q, B, T, and H, determine time $t = t_f$ when the client application buffer first becomes full.

To solve this problem, extend the formula for the cumulative number of bits in the buffer past the first sawtooth. t is replaced by $t\ mod\ T$ in the first two addends since the rate the buffer is filled relates to where it is in a sawtooth, not which sawtooth it is in.

$\frac{H(t\ mod\ T)^2}{2T} + \frac{HT*\lfloor T/t \rfloor}{2} - rt + Q$

To find the time when the buffer is full, set the cumulative bits equal to B and solve for t_f.

$\frac{H(t_f\ mod\ T)^2}{2T} + \frac{HT*\lfloor T/t_f \rfloor}{2} - rt_f + Q = B$

This should probably be simplified further, but frankly, I've spent far too much

time on this problem already. (Also, I don't know how to work with the mods/floor.) Watch this: "The remainder of the problem is left as an excercise to the reader."

4 Recall the simple model for HTTP streaming shown in figure 9.3. Suppose the client application buffer is infinite, the server sends at the constant rate x, and the video consumption rate is r with $r < x$. Also suppose playback begins immediately. Suppose that the user terminates the video early at time $t = E$. At the time of termination, the server stops sending bits (if it hasn't already sent all the bits in the video.)

4.1 Suppose the video is infinitely long. How many bits are wasted (that is, sent but not viewed)?

(This fudges the time between bits being sent and being received.)
Bits sent at time t: xt
Bits used at time t: rt
Bits wasted: $xE - rE$.

4.2 Suppose the video is T seconds long with $T > E$. How many bits are wasted (that is, sent but not viewed)?

$min[xE - rE,\ T - rE]$

5 Consider a DASH system for which there are N video versions (at N different rates and qualities) and N different audio versions (at N different rates and qualities). Suppose that we want to allow the player to choose at any time any of the N video versions and any of the N audio versions.

5.1 If we create files so that the audio is mixed in with the video, so the server sends only one media stream at a given time, how many files will the server need to store?

N^2

5.2 If the server sends the audio and video streams separately and has the client synchronize the streams, how many files will the server need to store?

$2N$

6 In the VoIP example in section 9.3, let h be the total number of header bytes added to each chunk, including UDP and IP headers.

6.1 Assuming an IP datagram is emitted every 20 msecs, find the transmission rate in bits per second for the datagrams generated by one side of this application.

In the example, the sender generates bytes at a rate of 8,000 bytes per second, or 64,000 bits per second. Sending 50 datagrams per second means sending $50 * 8h$ header bits per second, for a total of $64,000 + 40h$ bits per second.

6.2 What is a typical value of h when RTP is used?

RTP headers are normally 12 bytes and RTP typically runs on top of UDP, which contributes another 8 bytes. IP headers are 20 bytes, so $h = 40$.

7 Consider the procedure described in section 9.3 for estimating average delay d_i. Suppose that $u = 0.1$. Let $r_1 - t_1$ be the most recent sample delay, let $r_2 - t_2$ be the next most recent sample delay, and so on.

7.1 For a given audio application suppose four packets have arrived at the receiver with sample delays $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$, and $r_1 - t_1$. Express the estimate of delay d in terms of the four samples.

The formula for estimating delay is $d_i = (1-u)d_{i-1} + u(r_i - t_i)$. Use $d_1 = r_4 - t_4$ as the observed delay for the oldest sample.

$d_1 = r_4 - t_4$
$d_2 = (1 - 0.1)d_1 + 0.1(r_3 - t_3) = 0.9(r_4 - t_4) + 0.1(r_3 - t_3)$
$d_3 = 0.9(0.9(r_4 - t_4) + 0.1(r_3 - t_3)) + 0.1(r_2 - t_2)$
$d_4 = 0.9(0.9(0.9(r_4 - t_4) + 0.1(r_3 - t_3)) + 0.1(r_2 - t_2)) + 0.1(r_1 - t_1)$
$d = 0.9(0.81(r_4 - t_4) + 0.09(r_3 - t_3) + 0.1(r_2 - t_2)) + 0.1(r_1 - t_1)$
$d = 0.729(r_4 - t_4) + 0.081(r_3 - t_3) + 0.09(r_2 - t_2) + 0.1(r_1 - t_1)$

7.2 Generalize your formula for n sample delays.

$d = 0.1 \sum_{i=1}^{n} (0.9)^{i-1}(r_i - t_i)$

7.3 For the formula in the last part, let n approach infinity and give the resulting formula. Comment on why this averaging procedure is called an exponential moving average.

$\lim_{n\to\infty} 0.1 \sum_{i=1}^{n} (0.9)^{i-1}(r_i - t_i) = 0.1 \sum_{i=1}^{\infty} (0.9)^{i-1}(r_i - t_i)$
The weight given to older samples decreases exponentially as new samples are collected.

8 Repeat the first two parts of the previous problem for the estimate of average delay deviation.

8.1 For a given audio application suppose four packets have arrived at the receiver with sample delays $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$, and $r_1 - t_1$. Express the estimate of deviation v in terms of the four samples.

The formula for estimating the average deviation is $v_i = (1-u)v_{i-1} + u|r_i - t_i - d_i|$
Use $v_1 = 0$ as the deviation for the oldest sample.
$v_2 = 0.9(0) + 0.1|r_3 - t_3 - (0.9(r_4 - t_4) + 0.1(r_3 - t_3))|$
$v_3 = 0.9(0.1|r_3 - t_3 - (0.9(r_4 - t_4) + 0.1(r_3 - t_3))|) + 0.1|r_2 - t_2 - 0.9(0.9(r_4 - t_4) + 0.1(r_3 - t_3)) + 0.1(r_2 - t_2)|$
$v_4 = 0.9(0.9(0.1|r_3 - t_3 - (0.9(r_4 - t_4) + 0.1(r_3 - t_3))|) + 0.1|r_2 - t_2 - 0.9(0.9(r_4 - t_4) + 0.1(r_3 - t_3)) + 0.1(r_2 - t_2)|) + 0.1|r_4 - t_4 - 0.9(0.9(0.9(r_4 - t_4) + 0.1(r_3 - t_3)) + 0.1(r_2 - t_2)) + 0.1(r_1 - t_1)|$

8.2 Generalize your formula for n sample delays.

$v = 0.1 \sum_{i=1}^{n} (0.9)^{i-1} |r_i - t_i - d_i|$

9 For the VoIP example in section 9.3, we introduced an online procedure for estimating delay. In this problem we will examin an alternative procedure. Let t_i be the timestamp of the ith packet received; let r_i be the time at which the ith packet is received. Let d_n be our estimate of average delay after receiving the nth packet. After the first packet is received, we set the delay estimate equal to $d_1 = r_1 - t_1$.

9.1 Suppose that we would like $d_n = (r_1 - t_1 + r_2 - t_2 + ... + r_n - t_n)/n$ for all n. Give a recursive formula for d_n in terms of d_{n-1}, r_n, and t_n.

$d_n = d_{n-1} * \frac{n-1}{n} + \frac{r_n - t_n}{n}$
$d_n = \frac{d_{n-1}(n-1) + r_n - t_n}{n}$

9.2 Describe why for Internet telephony, the delay estimate described in section 9.3 is more appropriate than the delay estimate outlined in the previous part of this problem.

The observed delays of recently received packets is a much better indication of how long the delays of soon to arrive packets will be. This estimate gives equal weights to all observed delays, even very old ones. An exponential moving average is more appropriate.

10 Compare the procedure described in section 9.3 for estimating average delay with the procedure in section 3.5 for estimating round-trip time. What do the procedures have in common? How are they different?

The procedure for estimating TCP round-trip times is an exponential moving average, just like the procedure for estimating average delay. The only difference is what they are measuring.

11 Consider the figure below. A sender begins sending packetized audio periodically at t=1. The first packet arrives at the receiver at t=8.

11.1 What are the delays (from sender to receiver, ignoring any playout delays) of packets 2 through 8?

7, 9, 8, 7, 9, 8, 8

11.2 If audio playout begins as soon as the first packet arrives at the receiver at t=8, which of the first eight packets sent will not arrive in time for playout?

3, 4, 6, 7, 8

11.3 If audio playout begins at t=9, which of the first eight packets will not arrive in time for playout?

3, 6

11.4 What is the minimum playout delay at the receiver that results in all of the first eight packets arriving in time for their playout?

t=10

12 Consider again the figure in the previous problem.

12.1 Compute the estimated delay for packets 2 through 8 using the formula for d_i from section 9.3.2. Use a value of $u = 0.1$.

$d_i = (1-u)d_{i-1} + u(r_i - t_i)$
$d_2 = 0.9 * 7 + 0.1(7) = 7$
$d_3 = 0.9 * 7 + 0.1(9) = 7.2$
$d_4 = 0.9 * 7.2 + 0.1(8) = 7.28$
$d_5 = 0.9 * 7.28 + 0.1(7) = 7.252$
$d_6 = 0.9 * 7.252 + 0.1(9) = 7.4268$
$d_7 = 0.9 * 7.4268 + 0.1(8) = 7.48412$
$d_8 = 0.9 * 7.48412 + 0.1(8) = 7.535708$

12.2 Compute the estimated deviation of the delay from the estimated average for packets 2 through 8, using the formula for v_i from section 9.3.2. Use a value of $u = 0.1$.

$v_i = (1-u)v_{i-1} + u|r_i - t_i - d_i|$
$v_2 = 0.9 * 0 + 0.1|7 - 7| = 0$
$v_3 = 0.9 * 0 + 0.1 * 1.8 = 0.18$
$v_4 = 0.9 * 0.18 + 0.1 * 0.72 = 0.234$
$v_5 = 0.9 * 0.234 + 0.1 * 0.252 = 0.2358$
$v_6 = 0.9 * 0.2358 + 0.1 * 1.5732 = 0.36954$
$v_7 = 0.9 * 0.36954 + 0.1 * 0.51588 = 0.384174$
$v_8 = 0.9 * 0.384174 + 0.1 * 0.464292 = 0.3921858$

13 Recall the two FEC schemes for VoIP described in section 9.3. Suppose the first scheme generates a redundant chunk for every four original chunks. Suppose the second scheme uses a low-bit rate encoding whose transmission rate is 25% of the transmission rate of the nominal stream.

13.1 How much additional bandwidth does each scheme require? How much playback delay does each scheme add?

In the first scheme, if the four original chunks required n units of bandwidth, sending the redundant chunk as well requires $1.25n$ units of bandwidth, a 25% increase. It increases the playout delay by an amount of time equal to the time it takes for the four additional packets in a group to arrive.

In the second scheme, the additional stream also increases the bandwidth requirement by 25%. The playout delay, however, is only increased by one packet.

13.2 How do the two schemes perform if the first packet is lost in every group of five packets? Which scheme will have better audio quality?

In the first scheme, the stream, although slightly delayed, will not suffer any loss of quality. In the second scheme, the audio will play sooner after it was sent, but one of the packets in each group will suffer a 75% reduction in quality.

13.3 How do the two schemes perform if the first packet is lost in every group of two packets? Which scheme will have better audio quality?

In the first scheme, the audio will be almost unplayable, with half of it missing. The redundant packets will be useless. In the second scheme, it will be completely intelligible, but the overall quality will be reduced by 40%.

14 Skype

14.1 Consider an audio conference call in Skype with $N > 2$ participants. Suppose each participant generates a constant stream of rate r bps. How many bits per second will the call initiator need to send? How many bits per second will the other $N-1$ participants need to send? What is the total send rate, aggregated over all participants?

Each of the N participants sends their r bits per second to the call initiator, who aggregates all of their data, as well as their own, into one stream of Nr bits per second. The initiator sends this stream to each of the $N-1$ other participants.
Initiator: $(N-1) * Nr$
All other participants: $(N-1)r$
Total: $(N-1)r + (N-1) * Nr = Nr - r + N^2r - Nr = r(N^2-1)$

14.2 Repeat the previous part of this problem for a Skype video conference call using a central server.

Each of the N participants sends their r bits per second to a central server, which delivers all of the separate streams (because video can't be aggregated) to each participant. For all participants, the total send rate is Nr bits per second.

14.3 Repeat the previous part of this problem but for when each peer sends a copy of its video stream to each of the $N-1$ other peers.

Each of the N participants sends their r bits per second to all of the other $N-1$ participants, for a total of $Nr(N-1)$.

15 UDP Sockets

15.1 Suppose we send two IP datagrams, each carrying a different UDP segment. The first datagram has source IP address A1, destination address B, source port P1, and destination port T. The second datagram has source IP address A2, destination IP address B, source port P2, and destination port T. Suppose that A1 is different from A2 and that P1 is different from P2. Assuming that both datagrams reach their final destination, will the two UDP datagrams be received by the same socket? Why or why not?

Unlike TCP, UDP is connectionless, so sockets are identified only by the local IP address and port number, not the address and port number of the host it is communicating with. For this reason, both datagrams will be received by the same socket, since the destination IP address and destination port numbers are the same.

15.2 Suppose Alice, Bob, and Claire want to have an audio conference call using SIP and RTP. For Alice to send and receive RTP packets to and from Bob and Claire, is only one UDP socket sufficient (in addition to the socket needed for the SIP messages)? If yes, then how does Alice's SIP client distinguish between the RTP packets received from Bob and Claire?

Yes, only one UDP socket is needed. The source is identified in the synchronization source identifier field in the RTP header.

16 True or False

16.1 If stored video is streamed directly from a Web server to a media player, then the application is using TCP as the underlying transport protocol.

True. If it was using UDP, there would be an intermediary extracting the RTP packet from the UDP segment and the media chunk from the RTP packet. Also, there would be an additional control connection using a protocol like RTSP.

16.2 When using RTP, it is possible for a sender to change encoding in the middle of a session.

True. The sender would inform the receiver using the payload type field in the RTP header.

16.3 All applications that use RTP must use port 87.

False. RTP uses ports in the unprivileged range from 1024 to 65535.

16.4 If an RTP session has a separate audio and video stream from each sender, then the audio and video streams use the same SSRC.

False. If the streams are separate, they will be assigned different SSRCs and it will be the receiver's responsibility to synchronize them..

16.5 In differentiated services, while per-hop behavior defines differences in performance among classes, it does not mandate any particular mechanism for achieving these performances.

True. Per-hop behavior defines only the externally observable behavior, not the implementation.

16.6 Suppose Alice wants to establish an SIP session with Bob. In her INVITE message she includes the line: m=audio 48753 RTP/AVP 3. Alice has therefore indicated in this message that she wishes to send GSM audio.

False. This indicates that she wants to receive GSM audio.

16.7 Referring to the preceding statement, Alice has indicated in her INVITE message that she will send audio to port 48753.

False. She will receive audio on port 48753.

16.8 SIP messages are typically sent between SIP entities using a default SIP port number.

True. Well-known port number 5060 is for SIP.

16.9 In order to maintain registration, SIP clients must periodically send REGISTER messages.

True. Register messages indicate that the most recently sent IP address is still valid.

16.10 SIP mandates that all SIP clients support G.711 audio encoding.

False. SIP supports several audio encodings.

17 Consider the figure below, which shows a leaky bucket policer being fed by a stream of packets.

17.1 For each time slot, identify the packets that are in the queue and the number of tokens in the bucket, immediately after the arrivals have been processed but before any of the packets have passed through the queue and removed a token.

Time	Queue	Tokens
t=0	1,2,3	2
t=1	3,4	1
t=2	4,5	1
t=3	5,6	1
t=4	6	1
t=5	-	1
t=6	7,8	2
t=7	9,10	1
t=8	10	1

17.2 For each time slot, indicate which packets appear on the output after the token(s) have been removed from the queue.

Time	Tokens
t=0	1,2
t=1	3
t=2	4
t=3	5
t=4	6
t=5	-

t=6	7,8
t=7	9
t=8	10

18 Repeat the previous problem but assume that r=2. Assue again that the bucket is initially full.

Time	Queue	Tokens
t=0	1,2,3	2
t=1	3,4	2
t=2	5	2
t=3	6	2
t=4	-	2
t=5	-	2
t=6	7,8	2
t=7	9,10	2
t=8	-	2

Time	Tokens
t=0	1,2
t=1	3,4
t=2	5
t=3	6
t=4	-
t=5	-
t=6	7,8
t=7	9,10
t=8	-

19 Consider the previous problems and suppose now that r=3 and that b=2 as before. Will your answers change?

No. Since the bucket can only hold two tokens, it will not be able to accept the additional one anyway. (Note: At first, I thought this was contradicted by the following problem. There the size of the bucket is smaller than the rate, meaning more tokens would be deposited at each interval than the bucket could hold. I think the difference is that this problem is discrete, tokens are added once per interval, and the next one is continuous, so one token is deposited at a time, each 1/r intervals.)

20 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.

The maximum rate of flow using a leaky bucket policer is $rt + b$. In order to satisfy $rt + b \leq p$ when $t = 1$, $r + b \leq p$. In order to satisfy it when $t = 2$, $2r + b \leq 2p$. The easiest way to satisfy these equations would be to set $b = 0$ and control the rate directly. But, tokens can only be added if the bucket has available capacity. So, the next best option would be to set $b = 1$ and $r = p - 1$. (This only makes sense if tokens are added one at a time, once every 1/r intervals, and not all at once.)

This satisfies both equations:
$r + b \leq p \rightarrow (p - 1) + 1 \leq p \rightarrow p \leq p$
$2r + b \leq 2p \rightarrow 2(p - 1) + 1 \leq 2p \rightarrow 2p - 1 \leq 2p$
In general,
$rt + b \leq pt \quad \rightarrow \quad (p - 1)t + 1 \leq pt \quad \rightarrow \quad pt - t + 1 \leq pt \quad \rightarrow \quad 1 \leq t$

21 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 $rt + b$ 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?

No, it won't. Consider a flow that begins with a full bucket. On the first interval, $r + b$ packets arrive and exhaust the bucket. For the next n intervals,

r packets arrive, exhausting the bucket each time but never forced to wait. On interval $n+1$, no more than r packets can arrive, since that would violate the specification:
$t = 0$: $r + b$
$(t = 1, t = n)$: nr
$t = n + 1$: $r + 1$
Total: $n+2$ intervals allows $r(n+2)+b$ tokens. Adding the extra one on interval $n+1$ would make the total $r + b + nr + r + 1 = r(n + 2) + b + 1$, violating the specification. If at any point during the flow one fewer packet arrived, it would only make room for one extra packet somewhere else, but never exceeding $r + b$ packets on a single interval.

22 Show that as long as $r_1 < R\frac{w_1}{\sum w_j}$ then d_{max} is indeed the maximum delay that any packet in flow 1 will ever experience in the WFQ queue.

The worst case scenario for a packet in flow 1 is to arrive at the end of a burst in traffic. If there are b_1 packets in front of it, which are removed from the queue at a rate of $R\frac{w_1}{\sum w_j}$, the unfortunate packet will need to wait $\frac{b_1}{R\frac{w_1}{\sum w_j}} = d_{max}$ units of time before it is sent. If the burst lasts longer than one time interval, there will be an additional r_1 packets added to the queue in each unit of time. However, packets are removed at a rate of $R\frac{w_1}{\sum w_j} > r_1$, so there will be no accumulation as packets are removed faster than they can be added, and the packet's wait time will actually be shorter the further away it is from the beginning of the burst.