SOLUTIONS MANUAL

Second Edition

Data Networks

DIMITRI BERTSEKAS
Massachusetts Institute of Technology

ROBERT GALLAGER
Massachusetts Institute of Technology

PRENTICE HALL, Upper Saddle River, New Jersey 07458

Printed in the United States of America

12

ISBN 0-13-200924-2

Prentice-Hall International (UK) Limited,London
Prentice-Hall of Australia Pty. Limited, Sydney
Prentice-Hall Canada Inc., Toronto
Prentice-Hall Hispanoamericana, S.A., Mexico
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Pearson Education Asia Pte. Ltd., Singapore
Editora Prentice-Hall do Brasil, Ltda., Rio de Janeiro

Solutions Manual -- Data Networks, 2/E by Dimitri Bertsekas and Robert Gallager

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Several of our students have contributed to this solutions manual. We are particularly thankful to Rajesh Pankaj, Jane Simmons, John Spinelli, and Manos Varvarigos.

CHAPTER 1 SOLUTIONS

1.1

There are 250,000 pixels per square inch, and multiplying by the number of square inches and the number of bits per pixel gives 5.61×10^8 bits.

1.2

a) There are 16×10^9 bits going into the network per hour. Thus there are 48×10^9 bits per hour traveling through the network, or 13.33 million bits per second. This requires 209 links of 64 kbit/sec. each.

b) Since a telephone conversation requires two people, and 10% of the people are busy on the average, we have 50,000 simultaneous calls on the average, which requires 150,000 links on the average. Both the answer in a) and b) must be multiplied by some factor to provide enough links to avoid congestion (and to provide local access loops to each telephone), but the point of the problem is to illustrate how little data, both in absolute and comparative terms, is required for ordinary data transactions by people.

1.3

There are two possible interpretations of the problem. In the first, packets can be arbitrarily delayed or lost and can also get out of order in the network. In this interpretation, if a packet from A to B is sent at time τ and not received by some later time t, there is no way to tell whether that packet will ever arrive later. Thus if any data packet or protocol packet from A to B is lost, node B can never terminate with the assurance that it will never receive another packet.

In the second interpretation, packets can be arbitrarily delayed or lost, but cannot get out of order. Assume that each node is initially in a communication state, exchanging data packets. Then each node, perhaps at different times, goes into a state or set of states in which it sends protocol packets in an attempt to terminate. Assume that a node can enter the final termination state only on the receipt of one of these protocol packets (since timing information cannot help, since there is no side information, and since any data packet could be followed by another data packet). As in the three army problem, assume any particular ordering in which the two nodes receive protocol packets. The first node to receive a protocol packet cannot go to the final termination state since it has no assurance that any protocol packet will ever be received by the other node, and thus no assurance that the other node will ever terminate. The next protocol packet to be received then finds neither node in the final termination state. Thus again the receiving node cannot terminate without the possibility that the other node will receive no more protocol packets and thus never terminate. The same situation occurs on each received protocol packet, and thus it is impossible to guarantee that both nodes can eventually terminate. This is essentially the same argument as used for the three army problem.

CHAPTER 2 SOLUTIONS

2.1

Let x(t) be the output for the single pulse shown in Fig. 2.3(a) and let y(t) be the output for the sequence of pulses in Fig. 2.3(b). The input for 2.3(b) is the sum of six input pulses of the type in 2.3(a); the first such pulse is identical to that of 2.3(a), the second is delayed by T time units, the third is inverted and delayed by 2T time units, etc. From the time invariance property, the response to the second pulse above is x(t-T) (i.e. x(t) delayed by T); from the time invariance and linearity, the response to the third pulse is -x(t-2T). Using linearity to add the responses to the six pulses, the overall output is

$$y(t) = x(t) + x(t-T) - x(t-2T) + x(t-3T) - x(t-4T) - x(t-5T)$$

To put the result in more explicit form, note that

$$x(t) = \begin{cases} 0 & ; \quad t<0 \\ 1-e^{-2t/T} & ; \quad 0 \le t < T \\ (e^2-1)e^{-2t/T} & ; \quad t \ge T \end{cases}$$

Thus the response from the ith pulse ($1 \le i \le 6$) is zero up to time (i-1)T. For $t < 0$, then, $y(t) = 0$; from $0 \le t < T$

$$y(t) = x(t) = 1 - e^{-2t/T} \quad ; \quad 0 \le t < T$$

From $T \le t < 2T$,

$$\begin{aligned} y(t) &= x(t) + x(t-T) \\ &= (e2 -1)e^{-2t/T} + [1 - e^{-2(t-T)/T}] \\ &= 1 - e^{-2t/T} \end{aligned}$$

Similarly, for $2T \le t < 3T$,

$$\begin{aligned} y(t) &= x(t) + x(t-T) - x(t-2T) \\ &= (e^2-1)e^{-2t/T} + (e^2-1)e^{-2(t-T)/T} - [1 - e^{-2(t-2T)/T}] \\ &= -1 + (2e^4-1)e^{-2t/T} \quad ; \quad 2T \le t < 3T \end{aligned}$$

A similar analysis for each subsequent interval leads to

$$\begin{aligned} y(t) &= 1 - (2e^6 - 2e^4 + 1)e^{-2t/T} && ; \quad 3T \le t < 4T \\ &= -1 + (2e^8 - 2e^6 + 2e^4 - 1)e^{-2t/T} && ; \quad 4T \le t < 6T \\ &= -(e^{12} - 2e^8 + 2e^6 - 2e^4 + 1)e^{-2t/T} && ; \quad t \ge 6T \end{aligned}$$

The solution is continuous over t with slope discontinuities at 0, 2T, 3T, 4T, and 6T; the value of y(t) at these points is y(0) = 0; y(2T) = .982; y(3T) = -.732; y(4T) = .766; y(6T) = -.968. Another approach to the problem that gets the solution with less work is to use x(t) to first find the response to a unit step and then view y(t) as the response to a sum of displaced unit steps.

2.2

From the convolution equation, Eq. (2.1), the output r(t) is

$$r(t) = \int_{-\infty}^{\infty} s(\tau)h(t-\tau)d\tau = \int_{\tau=0}^{T} h(t-\tau)d\tau$$

Note that $h(t-\tau) = \alpha e^{-\alpha(t-\tau)}$ for $t-\tau \geq 0$, (i.e. for $\tau \leq t$), and $h(t-\tau) = 0$ for $\tau > t$. Thus for $t < 0$, $h(t-\tau) = 0$ throughout the integration interval above. For $0 \leq t < T$, we then have

$$r(t) = \int_{\tau=0}^{t} \alpha e^{-\alpha(t-\tau)}d\tau + \int_{\tau=t}^{T} 0\, d\tau = 1 - e^{-\alpha t} \quad ; 0 \leq t < T$$

For $t \geq T$, $h(t-\tau) = \alpha e^{-\alpha(t-\tau)}$ over the entire integration interval and

$$r(t) = \int_{\tau=0}^{T} \alpha e^{-\alpha(t-\tau)}d\tau = e^{-\alpha(t-T)} - e^{-\alpha t} \quad ; t \geq T$$

Thus the response increases towards 1 for $0 \leq t \leq T$ with the exponential decay factor α, and then, for $t \geq T$, decays toward 0.

2.3

From Eq. (2.1),

$$r(t) = \int_{-\infty}^{\infty} e^{j2\pi f\tau}h(t-\tau)d\tau$$

Using $\tau' = t-\tau$ as the variable of integration for any given t,

$$r(t) = \int_{-\infty}^{\infty} e^{j2\pi f(t-\tau')}h(\tau')d\tau'$$

$$= e^{j2\pi ft}\int_{-\infty}^{\infty} e^{-j2\pi f\tau'}h(\tau')d\tau'$$

$$= e^{j2\pi ft}H(f)$$

where H(f) is as given in Eq. (2.3).

2.4

$$h(t) = \int_{-\infty}^{+\infty} H(f)e^{j2\pi ft}df$$

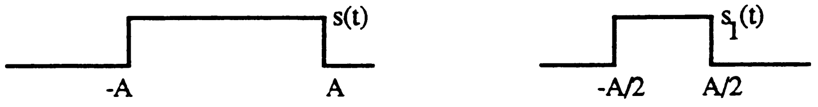
Since H(f) is 1 from $-f_0$ to f_0 and 0 elsewhere, we can integrate $\exp(j2\pi ft)$ from $-f_0$ to f_0, obtaining

$$h(t) = \frac{1}{j2\pi t}[\exp(j2\pi f_0 t) - \exp(-i2\pi f_0 t)] = \frac{\sin(2\pi f_0 t)}{\pi t}$$

Note that this impulse response is unrealizable in the sense that the response starts before the impulse (and, even worse, starts an infinite time before the impulse). None the less, such ideal filters are useful abstractions in practice.

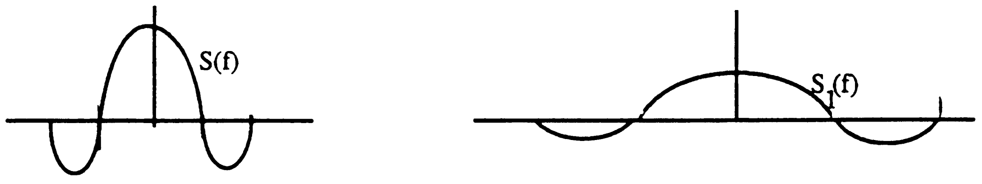
2.5

The function $s_1(t)$ is compressed by a factor of β on the time axis as shown below



$$S_1(f) = \int_{-\infty}^{\infty} s_1(t)e^{-j2\pi ft}dt = \int_{-\infty}^{\infty} s(\beta t)e^{-j2\pi ft}dt$$

$$= \int_{-\infty}^{\infty} s(\tau)e^{-j2\pi f\tau/\beta}\frac{d\tau}{\beta} = \frac{1}{\beta}S\left(\frac{f}{\beta}\right)$$



Thus $S_1(f)$ is attenuated by a factor of β in amplitude and expanded by a factor of β on the frequency scale; compressing a function in time expands it in frequency and vice versa.

2.6

a) We use the fact that $\cos(x) = [\exp(jx) + \exp(-jx)]/2$. Thus the Fourier transform of $s(t)\cos(2\pi f_0 t)$ is

$$\int_{-\infty}^{\infty} s(t)\frac{\exp(j2\pi f_0 t) + \exp(-j2\pi f_0 t)}{2}\exp(-j2\pi ft)\,dt$$

$$= \frac{s(t)}{2}\exp[-j2\pi(f-f_0)t]\,dt + \frac{s(t)}{2}\exp[-j2\pi(f+f_0)t]\,dt$$

$$= \frac{S(f-f_0)}{2} + \frac{S(f+f_0)}{2}$$

b) Here we use the identity $\cos^2(x) = [1+\cos(2x)]/2$. Thus the Fourier transform of $s(t)\cos^2(2pf_0t)$ is the Fourier transform of $s(t)/2$ plus the Fourier transform of $s(t)\cos[2p(2f_0)t]/2$. Using the result in part a, this is $S(f)/2 + S(f-2f_0)/4 + S(f+2f_0)/4$.

2.7

a) E{frame time on 9600 bps link} = 1000 bits / 9600 bps = 0.104 sec.

E{frame time on 50,000bps link} = 0.02 sec.

b) E{time for 10^6 frames on 9600 bps link} = $1.04 \cdot 10^5$ sec.

E{time for 10^6 frames on 50,000 bps link} = $2 \cdot 10^4$ sec.

Since the frame lengths are statistically independent, the variance of the total number of bits in 10^6 frames is 10^6 times the variance for one frame. Thus the standard deviation of the total number of bits in 10^6 frames is 10^3 times the standard deviation of the bits in one frame or $5 \cdot 10^5$ bits. The standard deviation of the transmission time is then

S.D.{time for 10^6 frames on 9600 bps link} = $5 \cdot 10^5 / 9600 = 52$ sec.

S.D.{time for 10^6 frames on 50,000 bps link} = $5 \cdot 10^5 / 50{,}000 = 10$ sec.

c) The point of all the above calculations is to see that, for a large number of frames, the expected time to transmit the frames is very much larger than the standard deviation of the transmission time; that is, the time per frame, averaged over a very long sequence of frames, is close to the expected frame time with high probability. One's intuition would then suggest that the number of frames per unit time, averaged over a very long time period, is close to the reciprocal of the expected frame time with high probability. This intuition is correct and follows either from renewal theory or from direct analysis. Thus the reciprocal of the expected frame time is the rate of frame transmissions in the usual sense of the word "rate".

2.8

Let x_{ij} be the bit in row i, column j. Then the ith horizontal parity check is

$$h_i = \Sigma_j x_{ij}$$

where the summation is summation modulo 2. Summing both sides of this equation (modulo 2) over the rows i, we have

$$\Sigma_i h_i = \Sigma_{i,j} x_{ij}$$

This shows that the modulo 2 sum of all the horizontal parity checks is the same as the modulo 2 sum of all the data bits. The corresponding argument on columns shows that the modulo 2 sum of the vertical parity checks is the same.

2.9

a) Any pattern of the form

```
--- 1 1 0 ---
--- 0 1 1 ---
--- 1 0 1 ---
```

will fail to be detected by horizontal and vertical parity checks. More formally, for any three rows i_1, i_2, and i_3, and any three columns j_1, j_2, and j_3, a pattern of six errors in positions $(i_1\, j_1)$, $(i_1\, j_2)$, $(i_2\, j_2)$, $(i_2\, j_3)$, $(i_3\, j_1)$, and $(i_3\, j_3)$ will fail to be detected.

b) The four errors must be confined to two rows, two errors in each, and to two columns, two errors in each; that is, geometrically, they must occur at the vertices of a rectangle within the array. Assuming that the data part of the array is J by K, then the array including the parity check bits is J+1 by K+1. There are (J+1)J/2 different possible pairs of rows (counting the row of vertical parity checks), and (K+1)K/2 possible pairs of columns (counting the column of horizontal checks). Thus there are (J+1)(K+1)JK/4 undetectable patterns of four errors.

2.10

Let $x = (x_1, x_2, \dots x_N)$ and $x' = (x'_1, x'_2, \dots x'_N)$ be any two distinct code words in a parity check code. Here N = K+L is the length of the code words (K data bits plus L check bits). Let $y = (y_1, \dots y_N)$ be any given binary string of length N. Let D(x,y) be the distance between x and y (i.e. the number of positions i for which $x_i \neq y_i$). Similarly let D(x',y) and D(x,x') be the distances between x' and y and between x and x'. We now show that

$$D(x,x') \leq D(x,y) + D(x',y)$$

To see this, visualize changing D(x,y) bits in x to obtain y, and then changing D(x',y) bits in y to obtain x'. If no bit has been changed twice in going from x to y and then to x', then it was necessary to change D(x,y) + D(x',y) bits to change x to x' and the above inequality is satisfied with equality. If some bits have been changed twice (i.e. $x_i = x'_i \neq y_i$ for some i) then strict inequality holds above.

By definition of the minimum distance d of a code, $D(x,x') \geq d$. Thus, using the above inequality, if $D(x,y) < d/2$, then $D(x',y) > d/2$. Now suppose that code word x is sent and fewer than d/2 errors occur. Then the received string y satisfies $D(x,y) < d/2$ and for every other code word x', $D(x',y) > d/2$. Thus a decoder that maps y into the closest code word must select x, showing that no decoding error can be made if fewer than d/2 channel errors occur. Note that this argument applies to any binary code rather than just parity check codes.

2.11

The first code word given, 1001011 has only the first data bit equal to 1 and has the first, third, and fourth parity checks equal to 1. Thus those parity checks must check on the first data bit. Similarly, from the second code word, we see that the first, second, and fourth parity checks must check on the second bit. From the third code word, the first, second, and third parity check each check on the third data bit. Thus

$c_1 = s_1 + s_2 + s_3$
$c_2 = s_2 + s_3$
$c_3 = s_1 + s_3$
$c_4 = s_1 + s_2$

The set of all code words is given by

0000000	0011110
1001011	1010101
0101101	0110011
1100110	1111000

The minimum distance of the code is 4, as can be seen by comparing all pairs of code words. An easier way to find the minimum distance of a parity check code is to observe that if x and x' are each code words, then x + x' (using modulo 2 componentwise addition) is also a code word. On the other hand, x + x' has a 1 in a position if and only if x and x' differ in that position. Thus the distance between x and x' is the number of ones in x + x'. It follows that the minimum distance of a parity check code is the minimum, over all non-zero code words, of the number of ones in each code word.

2.12

$$\begin{array}{r|l} & D^3 \\ \hline D^4+D^2+D+1 & D^7+D^5+D^4 \\ & D^7+D^5+D^4+D^3 \\ \cline{2-2} & \qquad\qquad D^3 \quad = \text{Remainder} \end{array}$$

2.13

Let $z(D) = D^j + z_{j-1}D^{j-1} + ... + D^i$ and assume $i<j$. Multiplying G(D) times Z(D) then yields

$$g(D)z(D) = D^{L+j} + (z_{j-1} + g_{L-1})D^{L+j-1} + (z_{j-2} + g_{L-1}z_{j-1} + g_{L-2})D^{L+j-2} + ... \\ + (g_1 + z_{i+1})D^{i+1} + D^i$$

Clearly the coefficient of D^{L+j} and the coefficient of D^i are each 1, yielding the desired two non-zero terms. The above case $i<j$ arises whenever z(D) has more than one non-zero term. For the case in which z(D) has only one non-zero term, i.e. $z(D) = Dj$ for some j, we have

$$g(D)z(D) = D^{L+j} + g_{L-1}D^{L+j-1} + ... + D^j$$

which again has at least two non-zero terms.

2.14

Suppose g(D) contains (1+D) as a factor; thus g(D) = (1+D)h(D) for some polynomial h(D). Substituting 1 for D and evaluating with modulo 2 arithmetic, we get g(1) = 0 because of the term (1+D) = (1+1) = 0. Let e(D) be the polynomial for some arbitrary undetectable error sequence. Then e(D) = g(D)z(D) for some z(D), and hence e(1) = g(1)z(1) = 0. Now $e(D) = \Sigma_i e_iD^i$, so $e(1) = \Sigma_i e_i$. Thus e(1) = 0 implies that an even number of elements e_i

are 1; i.e. that $e(D)$ corresponds to an even number of errors. Thus all undetectable error sequences contain an even number of errors; any error sequence with an odd number of errors is detected.

2.15

a) Let D^{i+L}, divided by $g(D)$, have the quotient $z^{(i)}(D)$ and remainder $c^{(i)}(D)$ so that

$$D^{i+L} = g(D)z^{(i)}(D) + c^{(i)}(D)$$

Multiplying by s_i and summing over i,

$$s(D)D^L = \Sigma_i\, s_i z^{(i)}(D) + \Sigma_i\, s_i c^{(i)}(D)$$

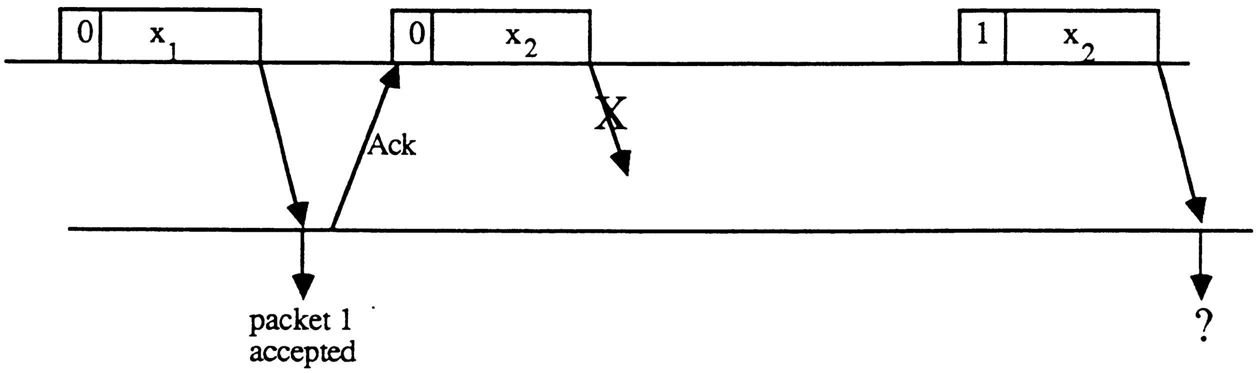
Since $\Sigma_i\, s_i c^{(i)}(D)$ has degree less than L, this must be the remainder (and $\Sigma_i\, s_i z^{(i)}(D)$ the quotient) on dividing $s(D)D^L$ by $g(D)$. Thus $c(D) = \Sigma_i\, s_i c^{(i)}(D)$.

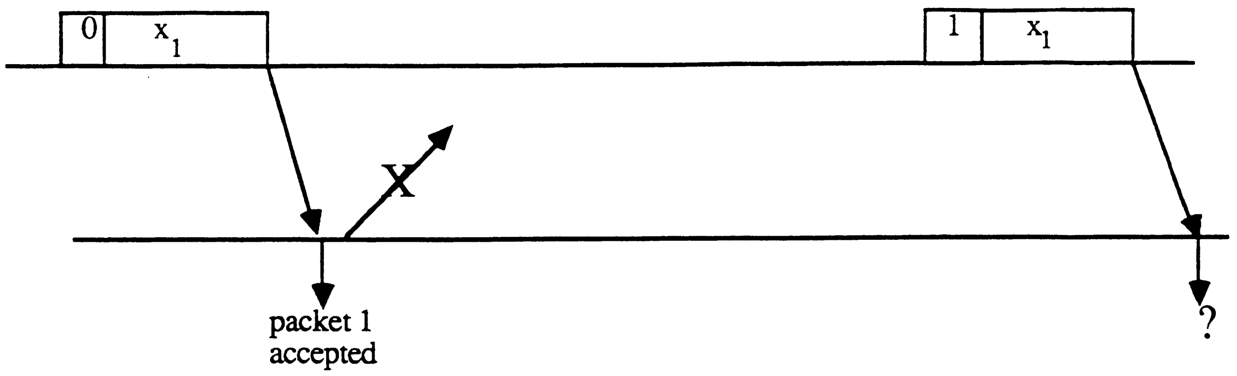
b) Two polynomials are equal if and only if their coefficients are equal, so the above polynomial equality implies

$$c_j = \Sigma_i\, s_i c_j^{(i)}$$

2.16

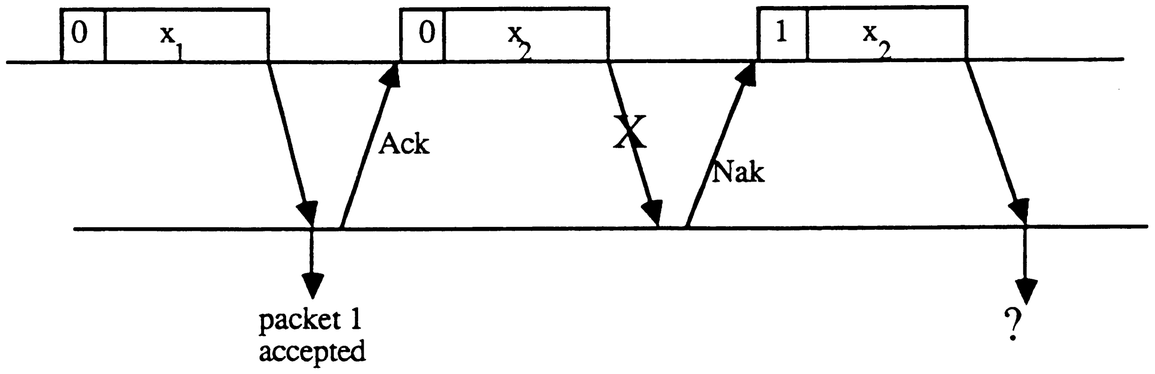
a) Consider the two scenarios below and note that these scenarios are indistinguishable to the receiver.

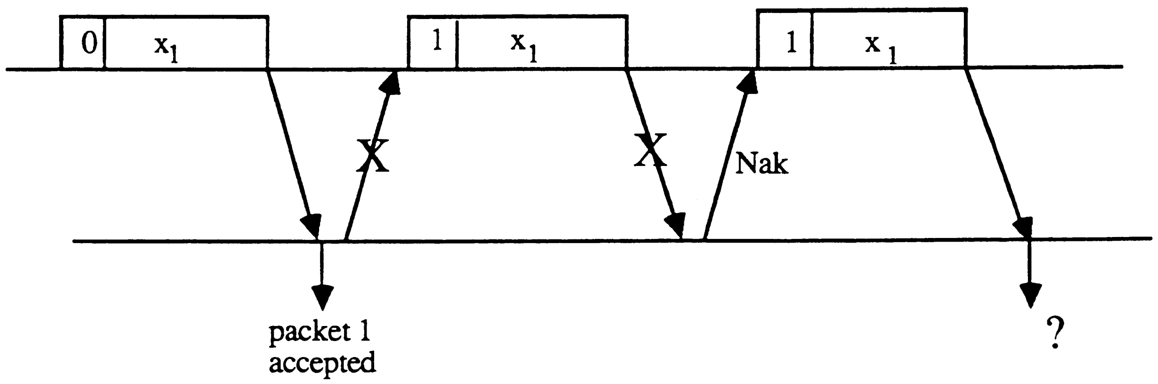




If the receiver releases the packet as x_2 in the questioned reception, then an error occurs on scenario 2. If the receiver returns an ack but doesn't release a packet (i.e. the appropriate action for scenario 2), then under scenario 1, the transmitter erroneously goes on to packet 3. Finally, if the receiver returns a nak, the problem is only postponed since the transmitter would then transmit $(2,x_2)$ in scenario 1 and $(2,x_1)$ in scenario 2. As explained on page 66, packets x_1 and x_2 might be identical bit strings, so the receiver can not resolve its ambiguity by the bit values of the packets.

b) The scenarios below demonstrate incorrect operation for the modified conditions.





2.17

a) $T = T_t + T_f + 2T_d$

b) $q = (1-p_t)(1-p_f)$

A packet is transmitted once with probability q, twice with probability (1-q)q, three times with probability $(1-q)^2q$, etc. Thus the expected number of transmissions of a given packet is

$$E\{\text{transmissions per packet}\} = \sum_{i=1}^{\infty} iq(1-q)^{i-1} = \frac{1}{q}$$

To verify the above summation, note that for any x, $0 \le x < 1$,

$$\sum_{i=1}^{\infty} ix^{i-1} = \sum_{i=1}^{\infty} \frac{dx^i}{dx} = \frac{d}{dx}\sum_{i=1}^{\infty} x^i = \frac{d}{dx}\left(\frac{x}{1-x}\right) = \frac{1}{(1-x)^2}$$

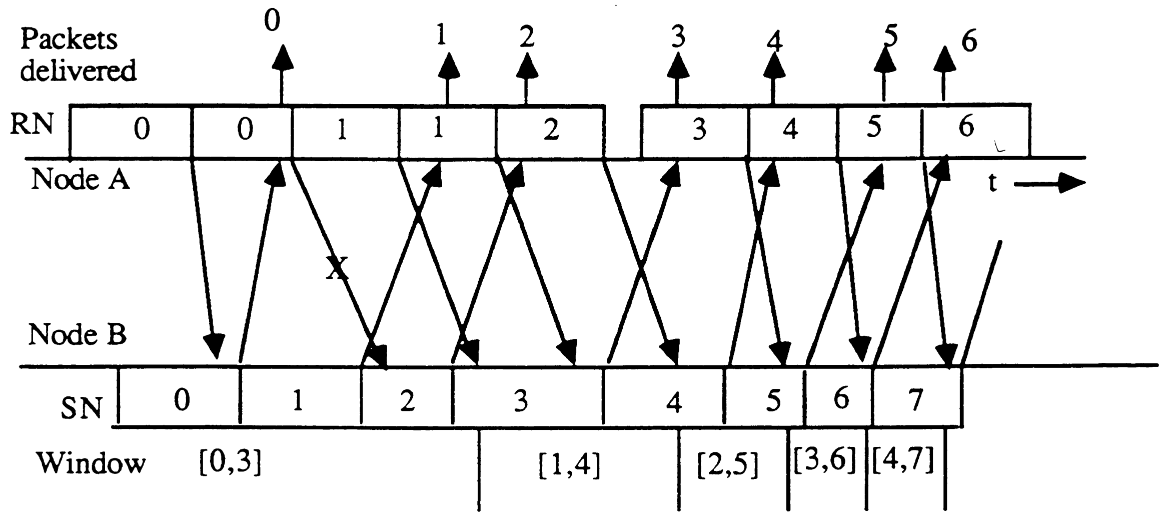
Using x for (1-q) above gives the desired result.

c) $E\{\text{time per packet}\} = (T_t + T_f + 2T_d)/q$

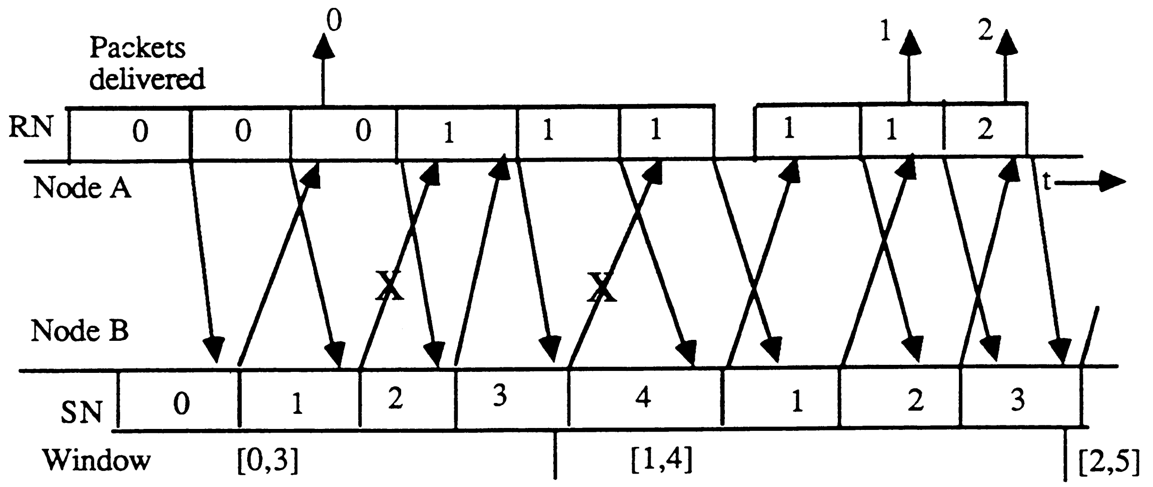
$= (1.3)/0.998 = 1.303$

Note that p_t and p_f have very little effect on E[time per packet] in stop and wait systems unless they are unusually large.

2.18

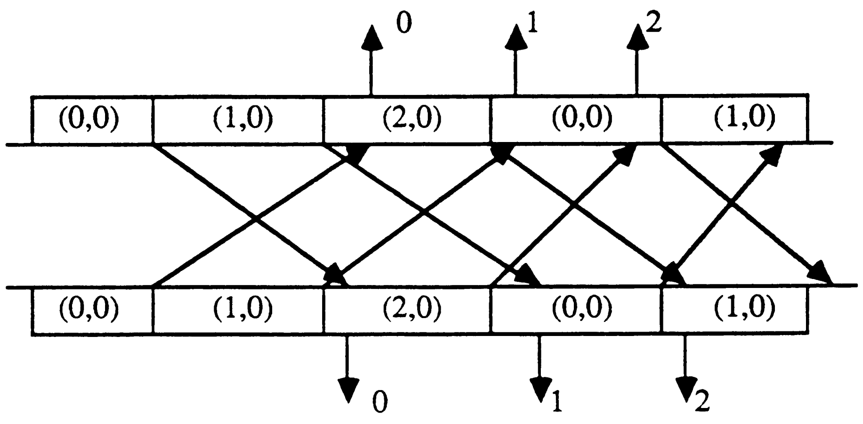


Assume that the transmitter always sends the next packet in order until reaching the end of the window, and then goes back to the beginning of the window.



2.19

The simplest such deadlock occurs if there is sufficient propagation delay in the system that each side can send n-1 frames (containing packets numbered 0 to n-2) before finishing receipt of the first frame from the other side. In this case, the nth frame from each side will carry the packet numbered n-1 without acking any packets from the other side. Thus each side will go back to packet 0, but in the absence of errors, each side will be looking for packet n by time the repeat of packet 0 occurs. Each side will then cycle from 0 to n-1, and neither side will ever receive any acks. The diagram below illustrates this for n=3. The first number in each frame position is SN and the second is RN.



2.20

The simplest example is for node A to send packets 0 through n-1 in the first n frames. In case of delayed acknowledgements (i.e. no return packets in the interim), node A goes back and retransmits packet 0. If the other node has received all the packets, it is waiting for packet n, and if the modulus m equals n, this repeat of packet 0 is interpreted as packet n.

The right hand side of Eq. (2.24) is satisfied with equality if $SN = SN_{min}(t_1)+n-1$. This occurs if node A sends packets 0 through n-1 in the first n frames with no return packets from node B. The last such frame has $SN = n-1$, whereas SN_{min} at that time (say t_1) is 0.

Continuing this scenario, we find an example where the right hand side of Eq. (2.25) is satisfied with equality. If all the frames above are correctly received, then after the last frame, RN becomes equal to n. If another frame is sent from A (now call this time t_1) and if SN_{min} is still 0, then when it is received at B (say at t_2), we have $RN(t_2) = SN_{min}(t_1)+n$.

2.21

Let $RN(\tau)$ be the value of RN at node B at an arbitrary time τ; SN_{min} is the smallest packet number not yet acknowledged at A at time t (which is regarded as fixed) and $SN_{max}-1$ is the largest packet number sent from A at time t. Since $RN(\tau)$ is non decreasing in τ, it is sufficient to show that $RN(t+T_m+T_d) \le SN_{max}$ and to show that $RN(t-T_m-T_d) \ge SN_{min}$.

For the first inequality, note that the packet numbered SN_{max} (by definition of SN_{max}) has not entered the DLC unit at node A by time t, and thus can not have started transmission by time t. Since there is a delay of at least T_m+T_d from the time a packet transmission starts until the completion of its reception, packet SN_{max} can not have been received by time $t+T_m+T_d$. Because of the correctness of the protocol, $RN(t+T_m+T_d)$ can be no greater than the number of a packet not yet received, i.e. SN_{max}.

For the second inequality, note that for the transmitter to have a given value of SN_{min} at time t, that value must have been transmitted earlier as the request number in a frame coming back from node B. The latest time that such a frame could have been formed is $t-T_m-T_d$, so by this time RN must have been at least SN_{min}.

2.22

a) If the transmitter never has to go back or wait in the absence of errors, then it can send a continuous stream of new packets in the absence of errors. In order for such a continuous stream to be sent, each packet must be acknowledged (i.e. SN_{min} must advance beyond the packet's number) before the next n-1 frames complete transmission. Thus these n-1 frame transmission times are in a race with the time, first, for the given packet to propagate over the channel and, second, for the acknowledgement to wait for the feedback frame in progress, then wait to be transmitted in the next feedback frame and propagated back to the original transmitter. In order for the feedback to always win the race, the minimum time for the n-1 frames to be transmitted must be greater than the maximum time for the feedback, i.e.,

$$(n-1)T_{min} > 2T_d + 2T_{max}$$

$$T_{max} < [(n-1)/2]T_{min} - T_d$$

b) If an isolated error occurs in the feedback direction, the feedback could be held up for one additional frame, leading to

$$(n-1)T_{min} > 2T_d + 3T_{max}$$

$$T_{max} < [(n-1)/3]T_{min} - (2/3)T_d$$

2.23

After a given packet is transmitted from node A, the second subsequent frame transmission termination from B carries the acknowledgement (recall that the frame transmission in progress from B when A finishes its transmission cannot carry the ack for that transmission; recall also that propagation and processing delays are negligible. Thus q is the probability of n-1 frame terminations from A before the second frame termination from B. This can be rephrased as the probability that out of the next n frame terminations from either node, either n-1 or n come from node A. Since successive frame terminations are equally likely and independently from A or B, this probability is

$$q = \sum_{i=n-1}^{n} \frac{n!}{i!(n-i)!} 2^{-n} = (n+1)2^{-n}$$

2.24

If an isolated error in the feedback direction occurs, then the ack for a given packet is held up by one frame in the feedback direction (i.e., the number RN in the feedback frame following the feedback frame in error reacknowledges the old packet as well as any new packet that might have been received in the interim). Thus q is now the probability of n-1 frame terminations from A before 3 frame terminations from B (one for the frame in progress, one for the frame in error, and one for the frame actually carrying the ack; see the solution to problem 2.23). This is the probability that n-1 or more of the next n+1 frame terminations come from A; since each termination is from A or B independently and with equal probability,

$$q = \sum_{i=n-1}^{n} \left(\frac{(n+1)!}{i!(n+1-i)!}\right) 2^{-n-1} = [n+2+(n+1)n/2]2^{-n-1}$$

2.25

As in the solution to problem 2.23, q is the probability of n-1 frame terminations coming from node A before two frame terminations come from node B. Frame terminations from A (and similarly from B) can be regarded as alternate points in a Poisson point process from A (or from B). There are two cases to consider. In the first, the initial frame is received from A after an even numbered point in the Poisson process at B, and in the second, the initial frame is received after an odd numbered point. In the first case, q is the probability that 2n-2 Poisson events from A occur before 4 Poisson events occur from B. This is the probability, in a combined Poisson point process of Poisson events from A and B, that 2n-2 or more Poisson events come from A out of the next 2n+1 events in the combined process. In the second case, q is the probability that 2n-2 Poisson events from A occur before 3 events occur from B. Since these cases are equally likely,

$$q = \frac{1}{2}\sum_{i=2n-2}^{2n+1} \left(\frac{(2n+1)!}{i!(2n+1-i)!}\right) 2^{-2n-1} + \frac{1}{2}\sum_{i=2n-2}^{2n} \left(\frac{(2n)!}{i!(2n-i)!}\right) 2^{-2n}$$

2.26

We view the system from the receiver and ask for the expected number of frames, γ, arriving at the receiver starting immediately after a frame containing a packet that is accepted and running until the next frame containing a packet that is accepted. By the assumptions of the problem, if the packet in a frame is accepted, then the next frame must contain the next packet in order (if not, the transmitter must have gone back to some earlier packet, which is impossible since that earlier packet was accepted earlier and by assumption was acked in time to avoid the go back).

Since the next frame after a packet acceptance must contain the awaited packet, that packet is accepted with probability 1-p. With probability p, on the other hand, that next frame contains an error. In this case, some number of frames, say j, follow this next frame before the awaited packet is again contained in a frame. This new frame might again contain an error, but the expected number of frames until the awaited packet is accepted, starting with this new frame, is again γ. Thus, given an error in the frame after a packet acceptance, and given j further frames before the awaited packet is repeated, the expected number of frames from one acceptance to the next is $1+j+\gamma$.

Note that j is the number of frames that the transmitter sends, after the above frame in error, up to and including the frame in transmission when feedback arrives concerning the frame in error. Thus the expected value of j is β. Combining the events of error and no error on the next frame after a packet acceptance, we have

$$\gamma = (1-p) + p(1+\beta+\gamma) = 1 + p(\beta+\gamma)$$

Solving for γ and for $\nu = 1/\gamma$,

$$\gamma = (1+\beta p)/(1-p) \qquad \nu = (1-p)/(1+\beta p)$$

2.27

Note that the sending DLC could save only one packet if it waited for acknowledgements rather than continuing to transmit. Similarly the sending DLC could save an arbitrarily large number of packets by taking packets from the network layer at a rate faster than they can be transmitted. Thus what is desired is to show that at most $\beta+1$ packets need be stored without ever forcing the transmitter to wait. Thus we assume that a new packet is admitted from the network layer only when there are no previously transmitted packets that are known to have been unsuccessful on the last transmission (i.e. the system repeats nak'ed packets before accepting and transmitting new packets; the system accepts and transmits new packets while waiting for feedback information on old packets).

When the system is first initiated, one packet is admitted to the sending DLC from the network layer. We use this as the basis of an inductive argument on successive times at which a new frame is generated. By the inductive hypothesis, at most $\beta+1$ packets were stored at the end of the previous frame generation instant. At the time of generating the new frame, there are at most β outstanding frames (including the one just being completed) for which feedback has not been received. A new packet will be accepted from the network layer only if all packets stored are also in frames for which no feedback has yet been received. Thus if a new packet is accepted, the total number saved is increased to at most $\beta+1$, and if no new packet is accepted, the total number saved remains (by the inductive hypothesis) no more than $\beta+1$.

2.28

Under the given assumptions, the ARPANET ARQ works like ideal select repeat. That is, frames from the 8 channels can be sent in round robin order and the feedback for a channel is always available by time the channel is to be reused. Thus a packet is repeated if and only if the previous transmission was unsuccessful. Since all channels are constantly busy and only the frames in error lead to retransmission, the efficiency is $1-p$.

2.29

a) When packet z is transmitted, the transmitter rule ensures that $z \le SN_{min}+n-1$. At that time, $SN_{min} \le RN$ since a packet cannot be acked before being received. Thus, at transmit time

$$z \le RN + n - 1$$

Since RN is nondecreasing, this is also satisfied at receive time. To derive the lower bound on z, note that the transmitter rule specifies $z \ge SN_{min}$. Since $SN_{min}+n$ has never been sent before the current transmission of z, the first come first serve order on the link ensures that it is not received before z. Thus y_{top} at receive time is less than $SN_{min}+n$ at transmit time, so

$$z \ge SN_{min} > y_{top} - n$$

$$z \ge y_{top} - n + 1$$

b) We must ensure that m is large enough to always satisfy

$$z + m > y_{top} + k$$

We know from a) that $z > y_{top}-n$, and adding $n+k$ to both sides of this equation, we know that $z+n+k > y_{top}+k$. Thus, choosing $m = n+k$ (or, more generally, $m \ge n+k$) always satisfies the above equation. If m is chosen any smaller (say $m = n+k-1$), then when $z = y_{top}-n+1$ (which can happen after a goback), $z+m$ will equal $y_{top}+k$, causing erroneous operation.

c) From b), $m \ge n+k > n$. From Eq. (2.47), $z \le RN+n-1 < RN+m$; thus $z-m < RN$.

d) From b) and c), $m \ge n+k$ assures correct operation at the receiver. Since $m > n$, correct operation at the transmitter is assured as in goback n.

e) Initially $y_{top} = RN-1$, so for $k=1$, the receiver can initially accept only RN. On each accepted packet, RN and y_{top} are each incremented by 1, so at all times only RN can be accepted. Thus $k=1$ is ordinary goback n ARQ. For $k=n$, all received z must satisfy $z \le y_{top} +k$, and we have ordinary selective repeat ARQ.

2.30

a) The sequence below shows the stuffed bits underlined for easy readability:

$$0\,1\,1\,0\,1\,1\,1\,1\,1\,\underline{0}\,0\,0\,1\,1\,1\,1\,1\,\underline{0}\,1\,0\,1\,0\,1\,1\,1\,1\,1\,\underline{0}\,1\,1\,1\,1\,1\,\underline{0}\,0\,1\,1\,1\,1\,0\,1\,0$$

b) Here the flags are shown underlined and the removed (destuffed) bits as x's:

$\underline{0\,1\,1\,1\,1\,1\,1\,0}\,1\,1\,1\,1\,1\,x\,1\,1\;\;0\,0\,1\,1\,1\,1\,1\,x\,0\,1\,1\,1\,1\,1\,x\,1\,1\,1\,1\,1\,x\,1\,1$

$0\,0\,\underline{0\,1\,1\,1\,1\,1\,1\,0}\,1\,0\,1\,1\,1\,1\,1\,x$

2.31

The modified destuffing rule starts at the beginning of the string and destuffs bit by bit. A zero is removed from the string if the previous six bits in the already destuffed portion of the string have the value 01^5. For the given example, the destuffed string, with flags shown underlined and removed bits shown as x's, is as follows:

$0\,1\,1\,0\,1\,1\,1\,1\,1\,x\,1\,1\,1\,1\,1\,1\,0\,1\,1\,1\,1\,1\,x\,1\,\underline{0\,1\,1\,1\,1\,1\,1\,0}$

2.32

The hint shows that the data string $01^501x_1x_2...$ must have a zero stuffed after 01^5, thus appearing as $01^500x_1x_2....$ This stuffed pattern will be indistinguishable from the original string $01^500x_1x_2...$ unless stuffing is also used after 01^5 in the string $01^500x_1x_2....$ Thus stuffing must be used in this case. The general argument is then by induction. Assume that stuffing is necessary after 01^5 on all strings of the form $01^50^kx_1x_2....$ Then such a stuffed sequence is $01^50^{k+1}x_1x_2....$ It follows as before that stuffing is then necessary after 01^5 in the sequence $01^50^{k+1}x_1x_2....$ Thus stuffing is always necessary after 01^5.

2.33

The stuffed string is shown below with the stuffed bits underlined and a flag added at the end.

$1\,1\,0\,1\,1\,0\,1\,0\,\underline{0}\,0\,1\,0\,\underline{0}\,1\,0\,\underline{0}\,1\,1\,1\,0\,1\,0\,\underline{0}\,1\,0\,1$

The destuffing rule is to decode (destuff) the string bit by bit starting at the beginning. A given 0 bit is then deleted from the string if the preceding three decoded bits are 010. The flag is detected when a 1 is preceded by the three decoded bits 010 and the most recently decoded bit was not deleted. The above is a general rule for detecting any type of flag sequence, rather than just 0101; for this special case, it is sufficient to look for the substring 0101 in the received string; the reason for the simplification is that if an insertion occurs within the flag, it has to occur by simply a repetition of the first flag bit.

2.34

Let γ be $\log_2 E\{K\} - j$. Since j is the integer part of $\log_2 E\{K\}$, we see that γ must lie between 0 and 1. Expressing $A = E\{K\}2^{-j} + j + 1$ in terms of γ and $E\{K\}$, we get

$$A = 2^{\gamma} + \log_2 E\{K\} - \gamma + 1$$

$$A - \log_2 E\{K\} = 2^{\gamma} - \gamma + 1$$

This function of γ is easily seen to be convex (i.e., it has a positive second derivative). It has the value 2 at $\gamma = 0$ and at $\gamma = 1$ and is less than 2 for $0 < \gamma < 1$. This establishes that