3.60 **60.** Let $g(x)=x^3+x+1$. Consider the information sequence 1001.

Solutions follow questions:

a. Find the codeword corresponding to the preceding information sequence.

Using polynomial arithmetic we obtain:

```
              1010
1011    | 1001000
        | 1011
          01000
           1011
           00110
```

Codeword = 1001110

b. Suppose that the codeword has a transmission error in the first bit. What does the receiver obtain when it does its error checking?

```
              0001
1011    | 0001110
        |    1011
              101
```

CRC calculated by Rx = 101 → error

Chapter 5:

11. Consider the Stop-and-Wait protocol as described in the chapter. Suppose that the protocol is modified so that each time a frame is found in error at either the sender or receiver, the last transmitted frame is immediately resent.

Solutions follow questions:

a. Show that the protocol still operates correctly.

The protocol will operate correctly because the only difference is that frames are retransmitted sooner than otherwise. The detection of errors in an arriving frame at the receiver will cause an ACK to be sent sooner, possibly causing the transmitter to retransmit sooner. The detection of errors in an arriving frame at the transmitter will cause an immediate retransmission of the current information frame.

b. Does the state transition diagram need to be modified to describe the new operation?

The state transition diagram remains the same.

c. What is the main effect of introducing the immediate-retransmission feature?

The main effect is a speedup in the error recovery process.

12. In Stop-and-Wait ARQ why should the receiver always send an acknowledgment message each time it receives a frame with the wrong sequence number?

Solution:

The sender cannot send the next frame until it has received the ACK for the last frame so, if the receiver gets a frame with the wrong sequence it has to be a retransmission of the previous frame received. This means that the ACK was lost so the receiver has to ACK again to indicate the sender that it has received the frame.

15. A 1 Mbyte file is to be transmitted over a 1 Mbps communication line that has a bit error rate of $p = 10^{-6}$.

Solutions follow questions:

The file length $n = 8 \times 10^6$ bits, the transmission rate $R = 1$ Mbps, and $p = 10^{-6}$.

a. What is the probability that the entire file is transmitted without errors? Note for n large and p very small, $(1-p)^n \approx e^{-np}$.

P[no error in the entire file] $= (1-p)^n \approx e^{-np}$, for $n >> 1$, $p << 1$

$= e^{-8} = 3.35 \times 10^{-4}$

We conclude that it is extremely unlikely that the file will arrive error free.

b. The file is broken up into N equal-sized blocks that are transmitted separately. What is the probability that all the blocks arrive correctly without error? Does dividing the file into blocks help?

A subblock of length n/N is received without error with probability:

P[no error in subblock] $= (1-p)^{n/N}$

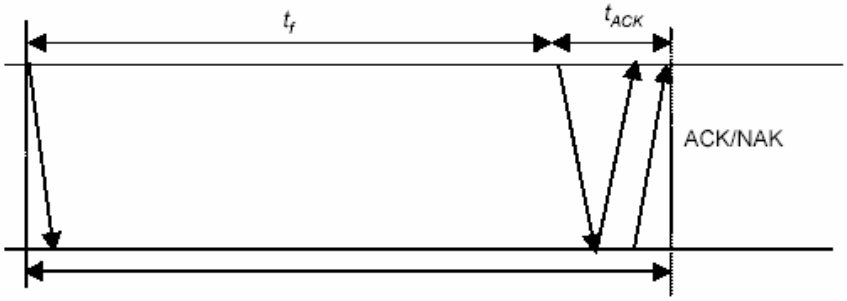
A block has no errors if all subblocks have no errors, so

P[no error in block] = P[no errors in subblock]^N =$((1-p)^{n/N})^N = (1-p)^n$

So simply dividing the blocks does not help.

c. Suppose the propagation delay is negligible, explain how Stop-and-Wait ARQ can help deliver the file in error-free form. On the average how long does it take to deliver the file if the ARQ transmits the entire file each time?

Refer to the following figure for the discussion.



We assume the following:

- t_0 = basic time to send a frame and receive the ACK/NAK ≈ $t_{timeout}$
- t_{total} = total transmission time until success
- n_f = number of bits/frame
- n_a = number of bits per ACK
- n_t = number of transmissions
- P_f = probability of frame transmission error

$t_0 = t_f + t_{ACK} = n_f/R + n_a/R$ ($t_{prop} \approx 0$).

$P[n_t = i\,] = P$[one success after $i-1$ failure] = $(1-P_f)\,P_f^{\,i-1}$

$t_{total} \mid i$ transmissions = $i.t_0$

$$E[t_{total}] = \sum_{i=1}^{\infty} i t_0 P[n_t = i] = t_0(1-P_f)\sum_{i=1}^{\infty} i P_f^{\,i-1} = t_0(1-P_f)/(1-P_f)^2 = t_0/(1-P_f)$$

Here, $n_f = n >> n_a$ thus $t_0 \approx t_f = n/R$; and $P_f = 1 - P$[no error] = $1 - e^{-np}$

E[total] = $n/R\,(1-P_f) = n/[Re^{-np}]$ = 8 / (3.35 x 10^{-4}) = 23,847 seconds = 6.62 hours!

The file gets through, but only after many retransmissions.

d. Now consider breaking up the file into N blocks. (Neglect the overhead for the header and CRC bits.) On the average how long does it take to deliver the file if the ARQ transmits the blocks one at a time? Evaluate your answer for N = 80, 800, and 8000.

For 1 block $P_f = 1 - P_b = 1 - (1-p)^{n/N}$ and $n_f = n/N$

if $t_{prop} \approx 0$ and $n_a << n/N$: $t_0^{\,b} = n_f/R = n/NR$

$T_b = E[t_{total}^{\ b}] = t_0^{\ b} / (1 - P_f) = n(1-p)^{-n/N} / NR$ average time to transmit one block

$T = E[t_{total}] = N\ T_b = n(1-p)^{-n/N} / R = 8\ (1-p)^{-n/N} = 8\ e^{np/N}$ if $n/N >> 1$, $p << 1$

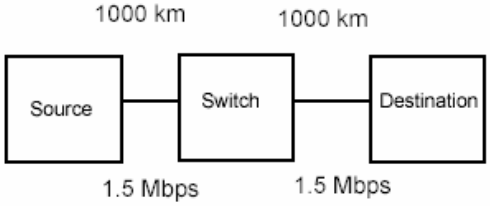
- $N = 80 \Rightarrow T \approx 8\ e^{0.1} = 8.84$ sec
- $N = 800 \Rightarrow T \approx 8\ e^{0.01} = 8.08$ sec
- $N = 8000 \Rightarrow T \approx 8\ e^{0.001} = 8.008$ sec

Each subblock has a higher probability of arriving without errors, and so requires fewer retransmissions to deliver error free. The overall delay is reduced dramatically.

e. Explain qualitatively what happens to the answer in part (d) when the overhead is taken into account.

As N increases, the effect of overhead becomes more significant because the headers constitute a bigger fraction of each subblock.

18. A 64-kilobyte message is to be transmitted from the source to the destination. The network limits packets to a maximum size of two kilobytes, and each packet has a 32-byte header. The transmission lines in the network have a bit error rate of 10^{-6}, and Stop-and-Wait ARQ is used in each transmission line. How long does it take on the average to get the message from the source to the destination? Assume that the signal propagates at a speed of 2×10^5 km/second.



Solution:

Message Size	65536	bytes
Max Packet Size	2048	bytes
Packet Header	32	bytes
Available for info	2016	bytes
# of packets needed	32.51	packets
Total	33	packets

bit error rate	1E-06		
bits/packet	16384		
Probability of error in packet	0.016251		1 – (1– bit_error_rate) ^ (bits/packet)
Propagation speed	2E+05	Km/s	
Distance	1000	Km	
Bandwidth	1.5	Mb/s	

We assume that the ACK error, the ACK time, and processing time are negligible.

T_{prop} = distance / propagation speed = 0.0050 s
T_f = packet size / bandwidth = 0.0109 s
T_0 = T_{prop} + T_f = 0.0159 s
P_f = probability of error in packet = 0.016251

$E[T_{total}]$= T_0/ (1 - P_f) = 0.0162

There is pipelining effect that occurs as follows: After the first packet arrives at switch 1, two transmissions take place in parallel. The first packet undergoes stop-and-wait on the second link while the second packet undergoes stop-and-wait in the first link. The packet arriving at the switch cannot begin transmission on the next link until the previous packet has been delivered, so there is an interaction between the transmission times of the two packets. We will neglect this effect. The time to send every packet over two links is then the initial packet transmission time + 33 additional packet timess, and so the average time is $E[T_{total}]$ * 34 = 0.522 seconds.

25. Consider the Go-Back-N ARQ protocol.

Solutions follow questions:

a. What can go wrong if the ACK timer is not used?

When no traffic arrives at a receiver during bidirectional Go-Back-N ARQ, and the receiver has to send an ACK, it usually sends the ACK after the ACK timer expires. If the ACK timer is not used, there are only two options remaining:

1) The ACK must be sent immediately (that is, use piggybacking only if frame already available)

Although this will function correctly, it is an inefficient use of bandwidth in the general case.

2) The ACK must only be sent if it can be piggybacked

This is problematic if traffic arrives sporadically. The sender will wait a long time until a piggyback opportunity arises.

b. Show how the frame timers can be maintained as an ordered list where the time-out instant of each frame is stated relative to the time-out value of the previous frame.

Assume that the timer counts down from $t_{timeout}$. In order to have a separate timer for each frame, we need not implement N timers. Only the oldest frame can timeout. The system can save, for each frame, an arrival offset time that is related to the frame that preceded it and place these offsets in an ordered list based on the frame sequence numbers. If an ACK for the oldest frame arrives, the system simply increments the timer by the offset of the following frame in the list. If an ACK for any other frame arrives, the timer is incremented by the sum of all of the offsets in the list that are up to this newly acknowledged frame.

c. What changes if each frame is acknowledged individually instead of by using a cumulative acknowledgment (R_{next} acknowledges all frames up to R_{next} –1)?

If each frame needs to be acknowledged individually, then the number of ACK messages will increase and the rate at which the transmission window can be increased will be reduced.