# SOLUTIONS TO PRACTICE PROBLEMS

## DATA AND COMPUTER COMMUNICATIONS

TENTH EDITION

WILLIAM STALLINGS

# TABLE OF CONTENTS

# CHAPTER 2 PROTOCOL ARCHITECTURE

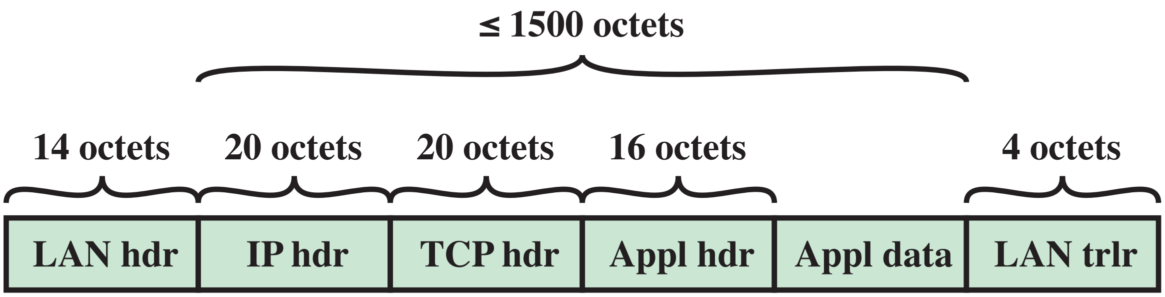
**2.1**

| Service access point addressing | Internet addressing | Physical addressing |
|---|---|---|
| The transport layer header includes a type of address called a service access point address or port address, which makes a data delivery from a specific process on one computer to a specific process on another computer. | If a packet traverses a network boundary, we need another addressing to differentiate the source and destination systems. The network layer adds a header, which indicates the logical address of the sender and receiver. | If the frames are to be distributed to different systems on the network, the data link layer adds the header, which defines the source machine's address and the destination machine's address. |

**2.2 a.**

```
+--------------+
| Application  |
+--------------+
| TCP          |
+--------------+
| IP           |
+--------------+
| Ethernet     |
+--------------+
```

**b.**



**c.** The LAN data field includes 56 bytes of IP, TCP, and Appl headers leaving 1444 bytes for Appl data. A 4096-byte message segments into 3 frames with 1444 bytes payload, 1444 bytes payload, and 1208 bytes payload. Thus, we have
3 × (14 + 20 + 20 + 16 + 4) bytes of overhead to transport 4096 bytes, or 222 / (4096 + 222) = 5.1% overhead.

# Chapter 3 Data Transmission

**3.1**

| Function | $A$ | $f$ | $\omega$ | $T$ | $\phi$ | $t_0$ |
|---|---|---|---|---|---|---|
| 5 sin 6($t$ − 1) | 5 | 0.955 | 6 | 1.047 | −0.167 | 1 |
| 5 sin (120$t$ + 0.75) | 5 | 19.099 | 120 | 0.052 | 0.75 | 0.006 |
| 6 sin (2π/5)($t$ − 3) | 6 | 0.2 | 1.257 | 5 | −3.77 | −3 |
| 6 cos (10π$t$ − 0.1) | 6 | 5 | 31.416 | 0.2 | 1.471 | 0.047 |

**3.2** The maximum data rate is, as given by Nyquist, $C = 2B \log_2 M$, where $B$ is the bandwidth and $M$ is the number of signaling levels. Therefore:

$$C = (2)\ (10{,}000)\ (\log_2 16) = 80{,}000 \text{ bps}$$

**3.3** $C = B\log_2\left(1+SNR\right) = 300\times\log_2\left(1+10^{30/10}\right) = 300\log_2 1001 \approx 2990 \text{ bps}$

**3.4** m(t) = 5 cos 1000пt cos 4000пt = 2.5[cos 3000пt + cos 5000пt]
= 2.5[cos 2п1500t + cos 2п2500t]

So the signal has components at f = 1500 Hz and f = 2500 Hz, for a bandwidth of 1000 Hz

**3.5** **a.** U = 3 / (3 + 1 + 1 + 2) = 43%

**b.** For any word, 4 possible bits could be in error (3 data, 1 parity), so since bit errors are independent,

$$P_c = (1 - P_b)^4 = (1 - 0.05)^4 = 0.8155.$$

Please note that only 4 possible bits can be wrong. If the start or stop bits were wrong, then NO bits can be received because these are framing bits: The receiver would not see the character unless the start and stop bits are correct.

**c.** Since bit errors are independent events, then $(P_c)^{10}$ = 0.1285

# CHAPTER 4 TRANSMISSION MEDIA

**4.1** Given $V_2$ = 40 μV and $V_1$ = 20 db, then

$$G_{\text{dB}} = 20\log\frac{V_2}{V_1} = 20\log\frac{40\times10^{-6}}{20\times10^{-6}} = 20\log 2 = 20(0.3) = 6\text{ dB}$$

**4.2** The reference antenna must supply an output 6 dB higher than 700 W.

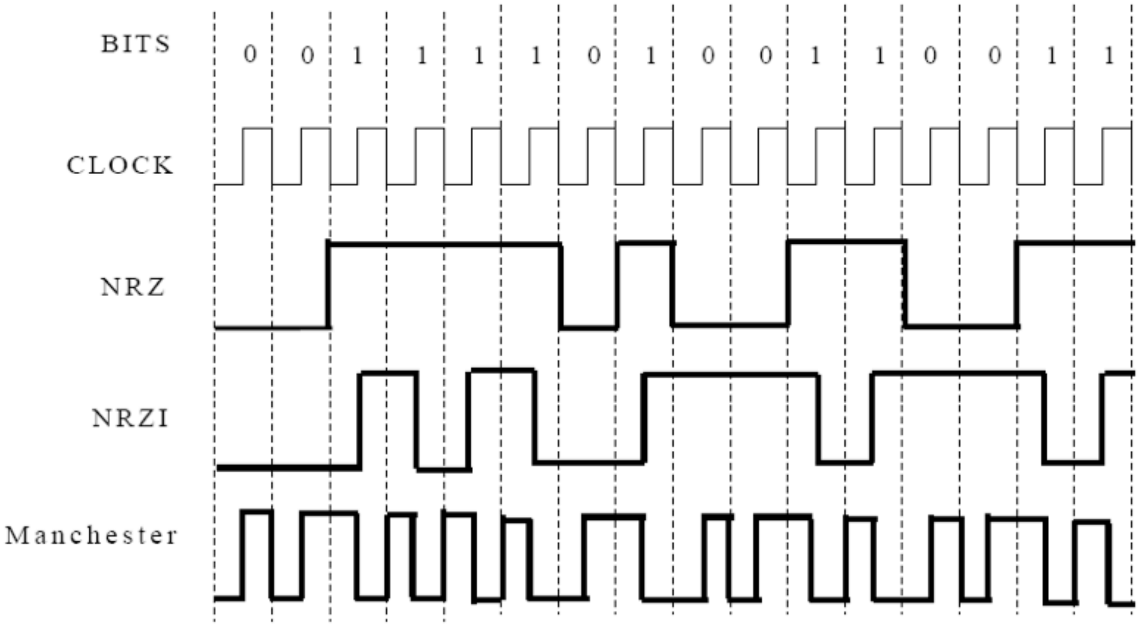
$$\begin{aligned} G_{\text{dB}} &= 10\log\frac{P_2}{P_1} \\ 6 &= 10\log\frac{P_2}{700} \\ 0.6 &= \log\frac{P_2}{700} \\ 10^{0.6} &= \frac{P_2}{700} \\ 4 &= \frac{P_2}{700} \\ P_2 &= 2800\text{ W} \end{aligned}$$

**4.3**

$$\begin{aligned} d &= 3.57\left(\sqrt{Kh_1} + \sqrt{Kh_2}\right) \\ 100 &= 3.57\left(\sqrt{Kh_1} + \sqrt{1.33\times250}\right) \\ \frac{100}{3.57} - 18.23 &= \sqrt{Kh_1} \\ 1.33h_1 &= (9.78)^2 \\ h_1 &= 71.9\text{ m} \end{aligned}$$

# CHAPTER 5 SIGNAL ENCODING TECHNIQUES

**5.1 a.**



**b.** Manchester encoding: (D) Non-of the above.

Manchester encoding will provide a signal change on every bit of data transferred, but has the downside of making the data-rate only have the baud rate (i.e., rate at which the signal can change on the wire).

**c.** NRZI: (A) Non-return to zero

It inverted signals a 1 by making a transition, but signals a 0 by staying at the same signal. Thus, it has problems with long sequences of zeros

**d.** NRZI: (A) Non-return to zero

It inverted signals a 1 by making a transition, but signals a 0 by staying at the same signal. Thus, it has problems with long sequences of zeros.

**5.2** bit-period/duration = 1/5000 = 0.2 [ms]

1.2/0.2 = 6 bits are transmitted totally

Bit sequence: 1 1 0 1 0 1

**5.3** **a.** The sampling rate must be twice the highest frequency, so $f_s \geq 2B =$ 6400 Hz

**b.** To achieve Rb = 36 kbps, we must have $nf_s \geq 36000$, so n ≥ 5.6 and we set n = 6

**c.** $L = 2^6 = 64$

**5.4** Let n be the number of bits per sample. Then $n = \lceil \log_2 L \rceil$. Thus,

$$\tau = \frac{1}{nf_s}$$

**5.5** **a.** We have $SNR_{dB} = 6.02n + 1.76$ dB ≥ 40, which yields n ≥ 6.35.

Thus, we set n = 7 and $L = 2^7 = 128$.

**b.** $SNR_{dB} = 6.02n + 1.76$ dB $= 6.02 \times 7 + 1.76 = 43.9$ dB

# CHAPTER 6 ERROR DETECTION

**6.1** The resultant accuracy is 2 minutes in one day or 2/(60 × 24). The allowable error is 0.4. Therefore the maximum number of bits is

$$\frac{0.4}{2/(60\times 24)} = 288 \text{ bits}$$

**6.2** 3

**6.3** The code words are 0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111. It can detect 1-bit errors. It can correct no errors.

**6.4** With $m$ = 32 there are $2^m$ possible m-bit messages. For each of those messages, there must be a unique $n$-bit codeword. It must also be possible that any one of the $n$ bits of a valid codeword could be flipped and yet still be identifiable as originally having been the valid $n$-bit codeword. Therefore, we must reserve $n$+1 of the $n$-bit patterns for each $m$-bit message – one for the valid codeword, and $n$ at Hamming distance 1 from the valid codeword.

If $n = m + r$, then we must be sure that $r$ (and thus $n$) is large enough for there to be $n$+1 codewords for each $m$-bit message. So, if there are $2^m$ possible messages, we need to be able to form at least $2^m(n+1)$ codewords. Therefore:

$$2^m(n+1) \leq 2^n$$

$$2^m(m+\mathrm{r}+1) \leq 2^{m+r}$$

$$m+r+1 \leq 2^r$$

Given $m$ = 32, we find that $r \leq 2^r - 33$. The smallest integer for which this equation holds is 6.

**6.5 a.** The total number of information bits per block is $m \times m = m^2$. The total number of bits per block is $m^2 + m + m = m^2 + 2m$. Therefore, the data rate is $\frac{m^2}{m^2+2m} = \frac{1}{1+(2/m)}$.

**b.** The data rate for the single parity bit code is $\frac{m^2}{m^2+1} = \frac{1}{1+(1/m^2)} > \frac{1}{1+(2/m)}$. So the single parity bit code has less redundancy.

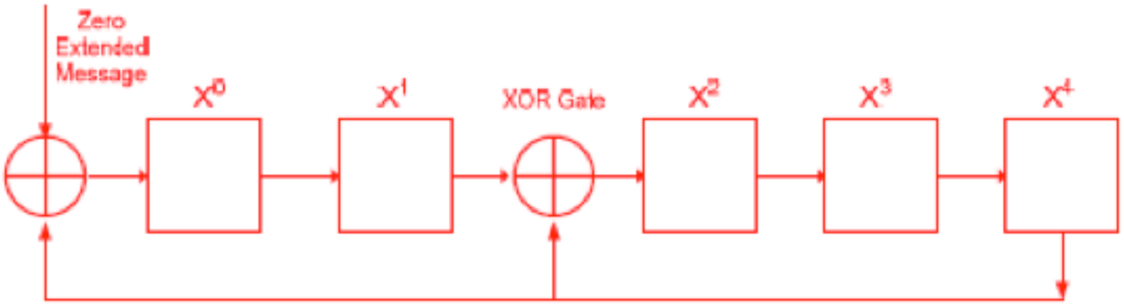
**6.6**

```
      1101
     -------
101 / 111001
      101
      ---
       100
       101
       ---
         101
         101
         ---
           0   So, since remainder is zero, F is without error.
```

**6.7 a.**

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

**b.**



**6.8 a.** There are 23 valid code words:

| Data block | Parity bit | Code word |
|---|---|---|
| 000 | 0 | 0000 |
| 001 | 1 | 0011 |
| 010 | 1 | 0101 |
| 011 | 0 | 0110 |
| 100 | 1 | 1001 |
| 101 | 0 | 1010 |
| 110 | 0 | 1100 |
| 111 | 1 | 1111 |

**b.** The code can detect all single-bit and all triple-bit errors.

**c.** The probability P of an undetected bit error is equal to the probability that two or four errors occur anywhere in a code word.

$$\begin{aligned} P &= \begin{pmatrix} 4 \\ 2 \end{pmatrix} p^2 (1-p)^2 + \begin{pmatrix} 4 \\ 4 \end{pmatrix} p^4 \\ &= 6p^2(1-p)^2 + p^4 \\ &= 6(0.01)^2(0.99)^2 + (0.01)^4 \approx 5.88 \times 10^{-4} \end{aligned}$$

# CHAPTER 7 DATA LINK CONTROL PROTOCOLS

**7.1 a.** Propagation delay = $(10 \times 10^{3}$ m$)/(2.5 \times 10^{8}$ m/s$) = 4 \times 10^{-5}$ seconds

**b.** Transmission delay = 1000 bits/$10^{8}$ bps = $10^{-5}$ seconds

**c.** Total delay = 4 $10^{-5}$ seconds + $10^{-5}$ seconds = $5 \times 10^{-5}$ seconds

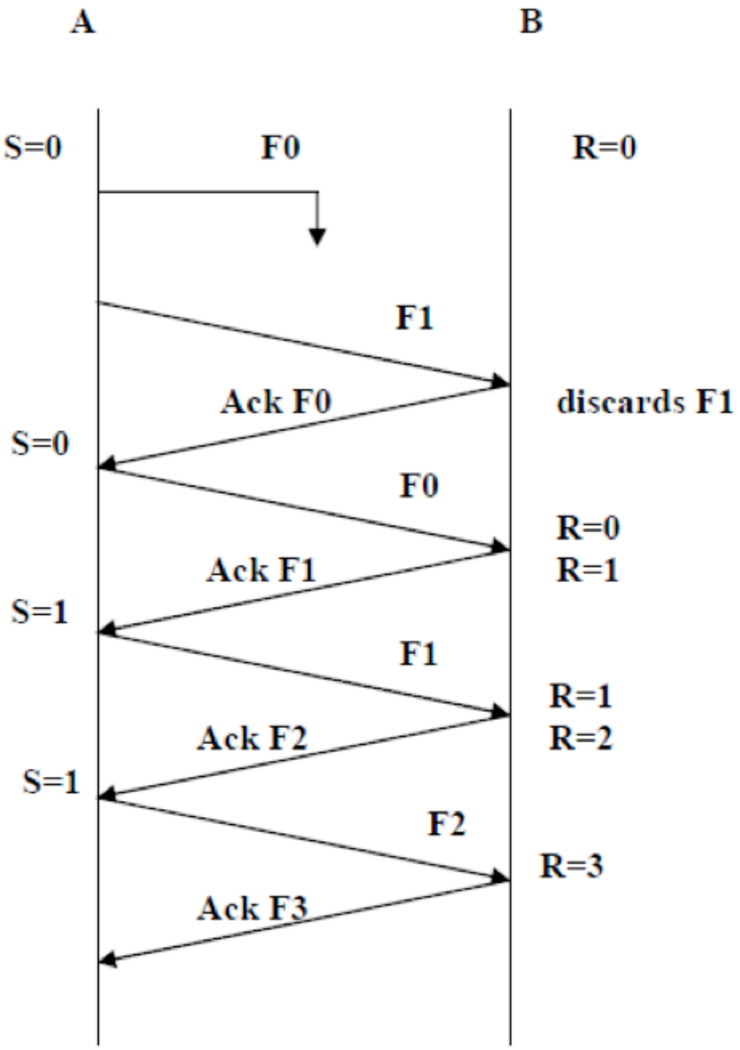
**7.2 a.** end-to-end Delay = (m/s + L/R) seconds

**b.** The bit is just leaving Host A

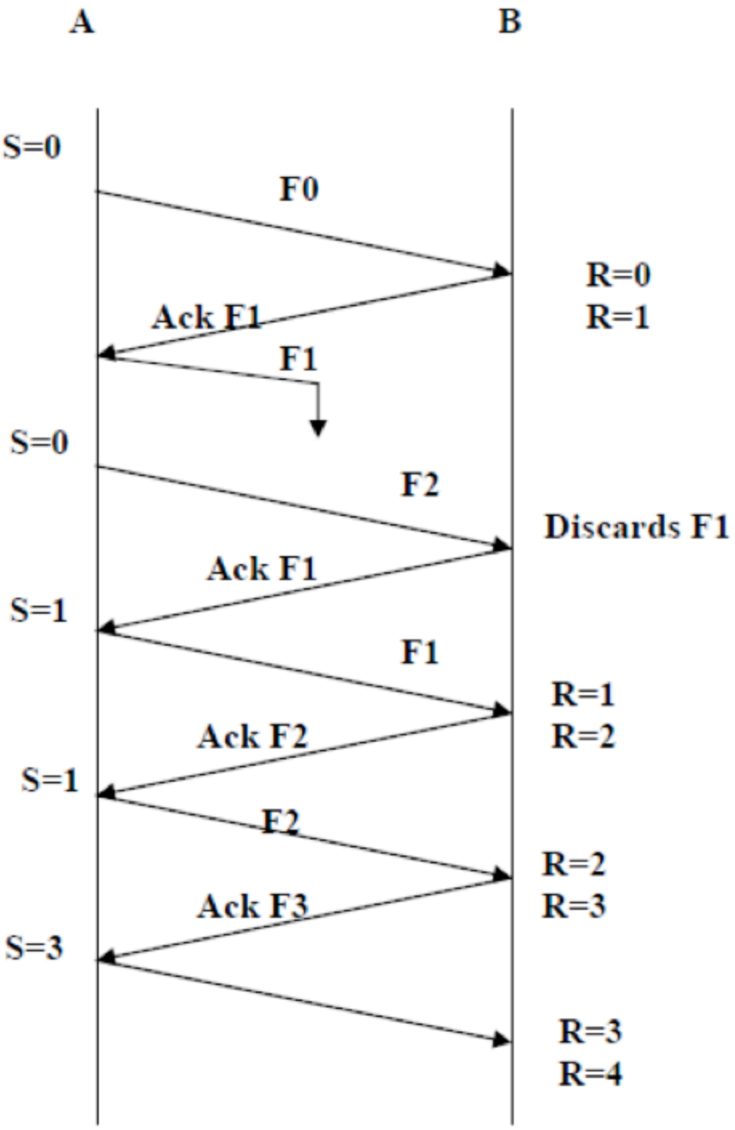
**c.** The first bit is in the link and has not reached Host B

**d.** Want m=L/R × S= 893 km

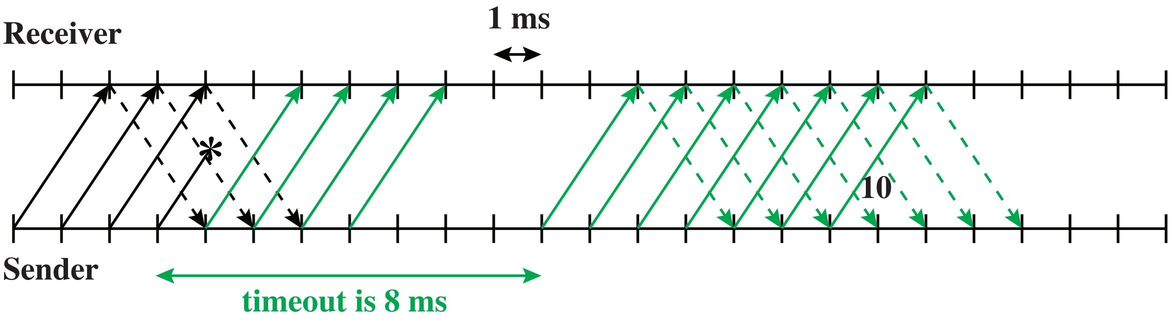
**7.3 a.**


A
B
S=0
F0
R=0
F1
Ack F0
discards F1
S=0
F0
R=0
R=1
Ack F1
S=1
F1
R=1
R=2
Ack F2
S=1
F2
R=3
Ack F3

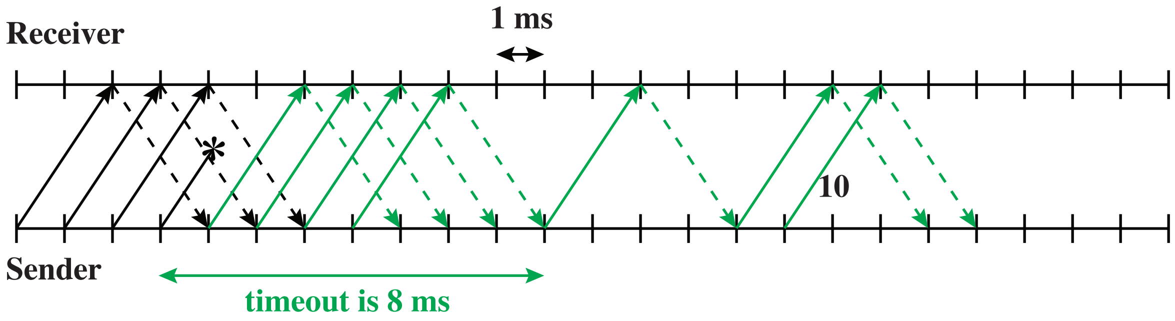
**b.**


A
B
S=0
F0
R=0
R=1
Ack F1
F1
S=0
F2
Discards F1
Ack F1
S=1
F1
R=1
R=2
Ack F2
S=1
F2
R=2
R=3
Ack F3
S=3
R=3
R=4

**7.4 a.**

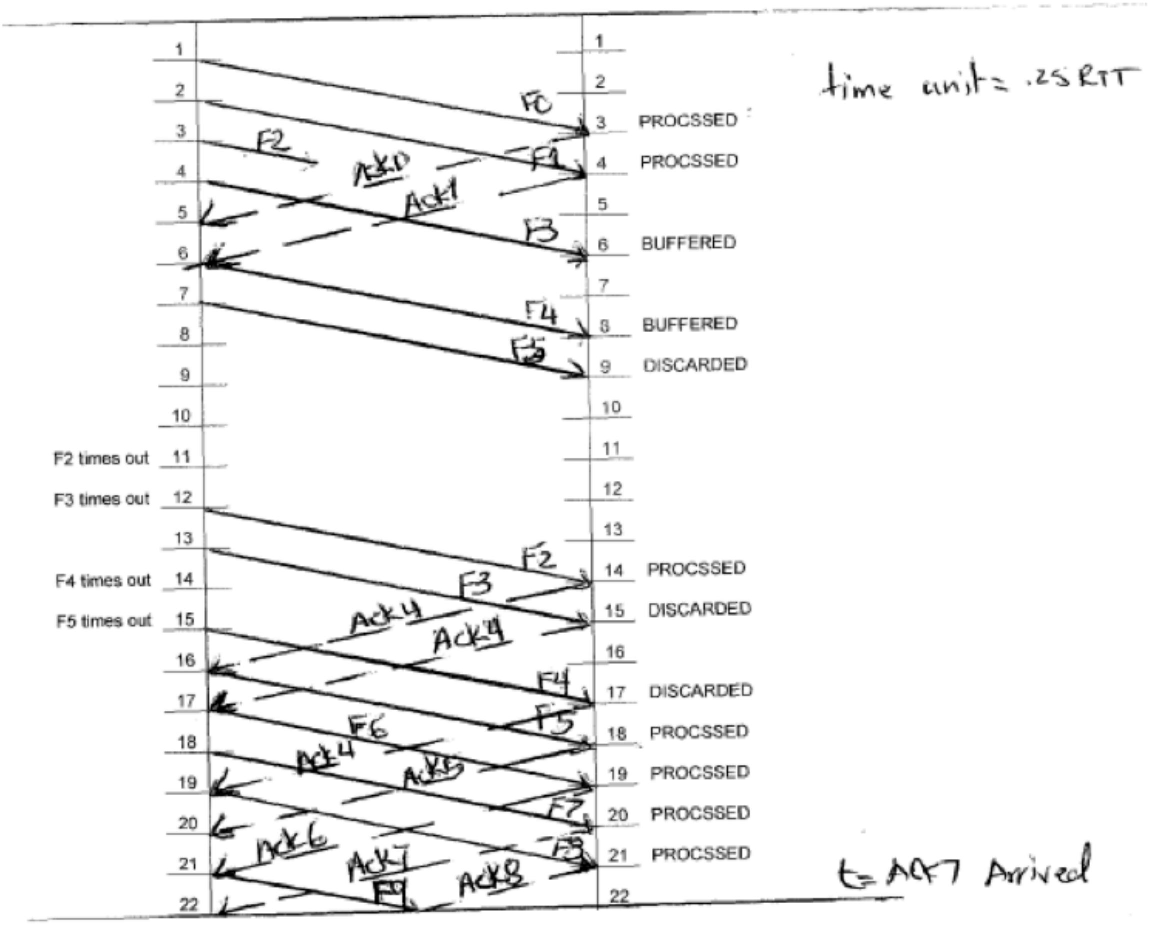

Receiver
1 ms
*
10
Sender
timeout is 8 ms

**b.**



**c.** After 8 ms of time out, the sender will resend the 4th packet. It takes 4 more ms to receive the ACK for the 4th packet. The window is never full if there are some not-yet-send packets in the window until receiving the ACK. The minimum window size should be 13 so that the window is never full.

## 7.5



# CHAPTER 8 MULTIPLEXING

**8.1** Synchronous TDM is interleaved at the bit level unless the sources are asynchronous, so the answer is: 0010. The following grid makes this clear:

10**0**11100**0**11000**1**11110**0**000

12**3**45612**3**45612**3**45612**3**456

**8.2** It is easier to solve the problem if we break the times down in this problem, such as the time to transmit a packet and a total round time (e.g., the time it takes from the start of George's first packet to the start of George's second packet).

The total number of packets (also represents the total number of TDM rounds) needed to complete the 1000-kB file transfer is:

packets = data size/packet size = $1 \times 10^6/1000 = 1000$.

A single packet transmission takes:
$T_{\text{packet}} = 8 \times 1000$ bits/$2 \times 10^6$ bits/sec = 0.004 seconds.

There is an idle time of 6 ms after each transmission (commonly referred to as a guard time), so the total time for a node to transmit is:

$T_{\text{node}} = T_{\text{packet}} + T_{\text{guard}} = 0.004$ s + 0.006 s = 0.01 seconds.

A total round time therefore takes:
$T_{\text{round}}$ = nodes × Tnode = $5 \times 0.01 = 0.05$ seconds.

For George to transmit the whole file, it takes 999 rounds and then only $T_{packet}$ additional time, since George transmits at the start of every round. This covers all 1000 packet transmissions. Therefore,

$T_{George} = 999 \times T_{round} + T_{packet} = 999 \times 0.05$ s + 0.004 s = 49.954 seconds.

**8.3 a.** $m_1(t)$ has a sampling rate of 7.2 kHz. $m_2(t)$, $m_3(t)$, and $m_4(t)$ each has a sampling rate of 2.4 kHz. The SCAN module rotates at a rate of 2400 rotations per second and samples $m_1(t)$ three times per rotation and samples the other signals once per rotation.

**b.** $m_1(t)$ has 7200 samples per second. Each of the other three signals has 2400 samples for second, for a total of 14,400 samples per second.

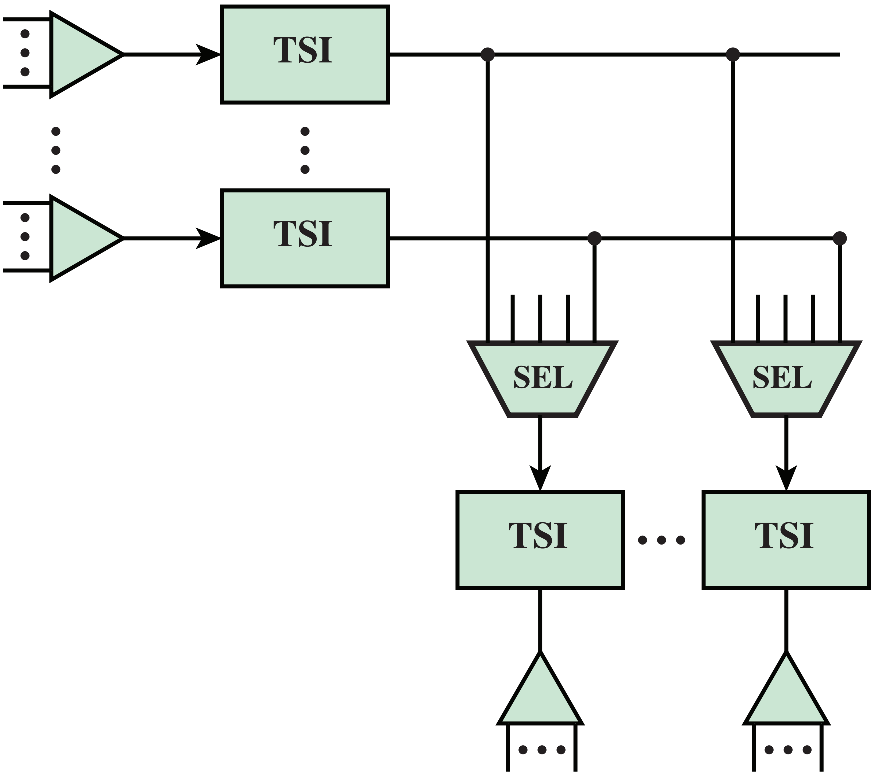
**c.** $L = 1024 = 2^{10} = 2^n$

Thus the output bit rate is 10 × 14,400 = 144 kbps

**d.** The minimum channel bandwidth is half the data rate, or 72 kHz

# CHAPTER 9 WAN TECHNOLOGY AND PROTOCOLS

**9.1**



**9.2** At time $t_0$ the sending host begins to transmit. At time $t_1$ = L/R1, the sending host completes transmission and the entire packet is received at the packet switch (no propagation delay). Because the switch has the entire packet at time $t_1$, it can begin to transmit the packet to the receiving host at time $t_1$. At time $t_2$ = $t_1$ + L/R2, the switch completes transmission and the entire packet is received at the receiving host (again, no propagation delay). Thus, the end-to-end delay is

$$L/R1 + L/R2 = L \times (R1 + R2)/(R1 \times R2).$$

**9.3 a.** 1M/100k = 10 users, since in circuit switching the resources are pre-allocated, we must allocate for the maximum rate.

**b.** In packet switching, the statistical multiplexing allows the instantaneous traffic (not the long term average) to potentially exceed the maximum capacity then more than 10 users can be admitted since, statistically, they will not always be on at the same time. Let's assume that N is the maximum number of users, then we can use the binomial distribution to calculate the probability of a certain number of users (say M) out of N will be active at any point in time. To avoid exceeding the maximum capacity M should not exceed 10 users. So we use the binomial distribution with parameters N (unknown), x > 10, p=0.1 and q=0.9. To get N, some trial and error is needed. Perhaps we can write a program that calculates the probability of x > 10 for N=10 onward and stops when that probability exceeds 0.0004. Any other tool (binomial calculator, excel, etc.) can be used for this task.

The maximum number of users (or sources) that can satisfy the probability of 0.0004 is 35. Any number over 35 will yield a probability of over 0.0004.

**9.4 a.** The time to transmit one packet onto a link is (L + h)/R. The time to deliver the first of the M packets to the destination is Q (L + h)/R. Every (L + h)/R seconds a new packet from the M – 1remaining packets arrives at the destination. Thus the total latency is

$$t_s + (Q + M - 1) (L + h)/R$$

**b.** (Q + M – 1) (L + 2h)/R

**c.** Because there are no store-and-forward delays at the links, the total delay is

$$t_s + (h + ML)/R$$

**9.5 a.** Circuit bandwidth = min (bandwidth of links) = 0.4 MB/s.

Time = 100 ms + (size of message)/circuit data rate.

**b.** Time = (size of message)/( data rate of A → B) + (size of message)/ (data rate of B → C) + (size of message)/(data rate of C → D)

**c.** #packets = 1024000/(8 × 128) = 1000 packets.

Total packet length = 128 + 22 = 150 bytes.

Time = (time to transfer 999 packets from A → B) + (time to transfer the last packet from A → B, B → C and C → D)

# CHAPTER 10 CELLULAR WIRELESS NETWORKS

**10.1** **a.** 40/0.2 = 200 users

**b.** There are 1260/30 = 42 cells. Each cell can support 200/7 users. Total number of users that can be supported is 42 × 200/7 = 1200 users

**10.2** **a.** The number of users supported per sector is 20/(0.2 + 0.2) = 50 users. The number of subscribers supported per cell is 50 × 360/120 = 150 users

**b.** 20 MHz × 5 bps/Hz × 3 × 40 = 12 Gbps (6 Gbps for uplink and 6 Gbps for downlink)

# CHAPTER 11 LOCAL AREA NETWORKS

**11.1** **a.** Whenever a new frame is received, bridge "learns" location (=the interface from which the frame enters) of sender and records sender/location pair in bridge table.

**b.** Here is the algorithm

```
When bridge receives a frame:
index bridge table using MAC dest address
if entry found for destination
then{
if dest on segment (interface) from which frame arrived
then drop the frame
else forward the frame on interface indicated
}
else (i.e., no entry found) flood by forwarding on all
interfaces except the one on which the frame arrived
```

**11.2**

| Frame | Source node | Destination node | Bridge forwards frame to interfaces # |
|---|---|---|---|
| 1 | A | D | 2,3 |
| 2 | H | D | 1,2 |
| 3 | C | H | 3 |
| 4 | G | H | Drop Frame |
| 5 | E | D | 1,3 |
| 6 | B | E | 2 |

**11.3** **a.** Total delay for 1 KB frame = $\{4 \times (8 \times 1024) / (10) + 4 \times 10\} = 3276.8 + 40 = 3316.8$ μsec = 3.317 msec

**b.** Effective Data Rate = $(8 \times 1024) / 3316.8 = 2.47$ Mbps

**c.** Delay of acknowledgement = 40 + [(4 × 20 × 8)/10] = 104 μsec

Total Delay = 3316.8 + 104 = 3412.8 μsec

Effective Data Rate = (8 × 1024)/3412.8 = 2.4 Mbps

**11.4** The bridges with the lowest bridge numbers will be used to prevent loops. Thus, the following bridges form the spanning tree: 10, 22, 32, 34, 40, and 56.

# CHAPTER 12 ETHERNET

**12.1** **a.** Recall that a repeater is a pass-through device that introduces almost no processing delay. So in this case, when A sends a bit, it arrives at C 10 μs later. The time to send whole packet is then:

$$[((1500 + 14 + 8) \times 8 \text{ bits}) / 10 \text{ Mbps}] + 10 \text{ μs} = 1227.6 \text{ μs}$$

The time to send the acknowledgment is:

$$[((10 + 14 + 8) \times 8 \text{ bits}) / 10 \text{ Mbps}] + 10 \text{ μs} = 35.6 \text{ μs}$$

Therefore, the effective data rate is:

$$(1500 \times 8 \text{ bits}) / (1227.6 + 35.6) \text{ μs} \approx 9.5 \text{ Mbps}$$

**b.** A bridge will buffer a packet before sending it out. So the total transfer time of a packet from A to C is: transmission time at A + latency A to B + transmission time at B +latency B to C. To send the forward packet, it takes:

$$[(2 \times (1500 + 14 + 8) \times 8 \text{ bits}) / 10\text{Mbps}] + 10\text{μs} = 2445.2\text{μs}$$

To send the ack, it takes:

$$[(2 \times (10 + 14 + 8) \times 8 \text{ bits}) / 10\text{Mbps}] + 10\text{μs} = 61.2\text{μs}$$

The effective data rate is:

$$(1500\text{bytes} \times 8 \text{ bits}) / (2445.2 + 61.2)\ \mu\text{s} \approx 4.8 \text{ Mbps}$$

**c.** A cut-through switch acts nearly like a repeater in that it does not buffer packets. However, it needs to receive the full header before deciding which port to output on. Therefore, it adds a processing delay of
(22 bytes × 8 bits)/10Mbps = 17.6 μs in each direction. The effective data rate is then:

$$(1500 \text{ bytes} \times 8 \text{ bits}) / (1227.6 + 35.6\mu\text{s} + (17.6 \times 2\ \mu\text{s})) \approx 9.2 \text{ Mbps}$$

**12.2** In order for A to detect the collision, it should not be able to finish its transmission before B's signal reaches A.

$$500 \text{ bits} / (100 \times 10^{6} \text{ bps}) = 5 \times 10^{-6} \text{ sec}$$

This is the duration of A's transmission. In the worst-case scenario, B may start just before A's first bit arrives. And even in the worst scenario, B's first bit should arrive to A before A's transmission ends. So the propagation time should be less than $2.5 \times 10^{-6}$ seconds. Hence,

$$2.5 \times 10^{-6} \text{ seconds} \times 2 \times 10^{8} \text{ m/sec} = 500 \text{ m}$$

**12.3** Let d = distance between hosts X and Y.

Let R = $2 \times 10^{8}$ m/s be the propagation speed of the network

Suppose host X starts transmitting a frame of size F = 400 bits.

Then, before hearing the first bit of the frame, host Y starts transmitting a frame. Since Y transmits BEFORE hearing X this can be at most d/R time after X starts transmitting.

X hears the beginning of Y's transmission d/R time after that. Therefore, the MAXIMUM amount of time that can pass from the time that X starts transmitting until X hears the first bit from Y is 2d/R.

Therefore we must have F/10Mbps ≥ 2d/R or FR/20Mbps ≥ d.

Plugging in all of the values we find that 4 km = $4 \times 10^3$ m ≥ d

**12.4** Manchester encoding follows the rules shown below:

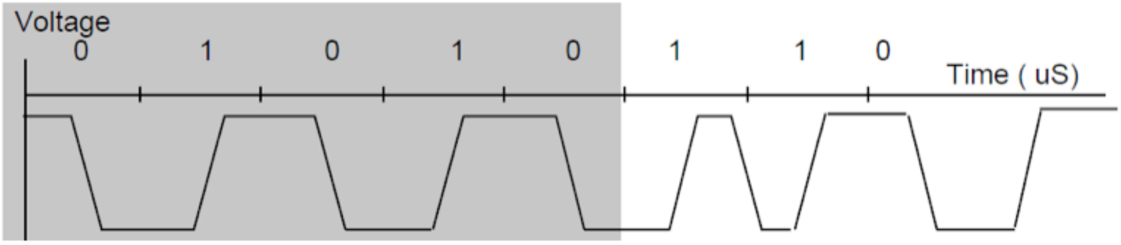
Original Data Value Sent

Logic 0 0 to 1 (upward transition at bit centre)

Logic 1 1 to 0 (downward transition at bit centre)

Preamble consists of 101010 sequence followed by the SFD, with a pattern 11

There are therefore 5 bits of preamble prior to the SFD  shown in the figure.



**12.5** **a.** Round trip propagation delay is $2 \times 10$ μs, so there are 2 slots in 40 μs.

Probability that station A will transmit data within 40 μs is to transmit in either first slot or in second slot.

P(A transmit in first slot) + P(A does not transmit in first slot, A transmit in second slot) = $0.6 + 0.4 \times 0.6 = 0.84$

**b.** P ( Both send after 2 slots) = P(( A send in 1st slot; B do not send in 1st slot, B send in 2nd slot) OR ( B send in 1st slot; A do not send in 1st slot, A send in 2nd slot)) = P(( A send in 1st slot; B do not send in 1st slot, B send in 2nd slot)) + P( B send in 1st slot; A do not send in 1st slot, A send in 2nd slot) = 0.6*0.4*0.6 + 0.6*0.4*0.6 = 0.288

**12.6** Minimum frame size for CSMA/CD is $2 \times T_{pr}$.

$T_{pr} = (4.5 \times 10^{-9}) \times (10 \times 10^{3}) = 4.5 \times 10^{-5}$ sec.

Thus, $(1.0 \times 10^{6}) \times (9.0 \times 10^{-5})$ = 11.25 bytes. CSMA/CD would be a very reasonable protocol for a network of this span and speed since the minimum frame size is not excessive (i.e., larger than 64 bytes)

**12.7** **a.** $2^3$

**b.** P(A will send in the next slot) = 1/23 = 1/8.

P(B will send in the next slot) = 1/25 = 1/32.

P(A & B will collide in the next slot) = P ( A & B will send in the next slot)

= P(A will send in the next slot) * P(B will send in the next slot)

= 1/8 × 1/32

If they collide in the next time (1st) slot, A has 4 collisions and B has 6 collisions. Then both A and B will transmit again in the subsequent time slot (2nd).

P(A will send again) = 1/24 = 1/16.

P(B will send again) = 1/26 = 1/64.

P(A & B will collide in the next slot)

= P(A will send in the next slot) * P(B will send in the next slot)

= 1/16 × 1/64

So the total probability is:

P = 1/8 × 1/32 × 1/16 × 1/64

**c.** 8 × 0.5 = 4.0 -> A will send in slot 4

32 × 0.6 = 19.2 -> B will send in slot 20

# CHAPTER 13 WIRELESS LANs

**13.1** In wireless networks, it is impossible to detect collisions during transmissions because the receiver radio has to be turned off for the duration.

**13.2** The transmitting node sends a RTS and the receiving node sends a CTS before transmission, indicating the period that the channel will be busy. Nodes that can hear the receiver will hear the CTS and stay silent for that period of time.

# CHAPTER 14 THE INTERNET PROTOCOL

**14.1**

| Frame sent on link from | Frame Source (MAC) Address | Frame Destination (MAC) Address | IP Datagram Source Address | IP Datagram Destination Address |
|---|---|---|---|---|
| R4 to R1 | 1A-1A-1A-1A-1A-03 | 1A-1A-1A-1A-1A-06 | 111.111.111.111 | 111.111.117.112 |
| R1 to R2 | 1A-1A-1A-1A-1A-10 | 1A-1A-1A-1A-1A-11 | 111.111.111.111 | 111.111.117.112 |
| R2 to R3 | 1A-1A-1A-1A-1A-12 | 1A-1A-1A-1A-1A-13 | 111.111.111.111 | 111.111.117.112 |

**14.2 a., b.** The following table answers parts (a) and (b)

| **Fragment** | **Size (header + data)** | **flag** | **Offset** |
|---|---|---|---|
| 1 | 1000 (20+980) | 1 | 0 |
| 2 | 1000 (20+980) | 1 | 980 |
| 3 | 1000 (20+980) | 1 | 1960 |
| 4 | 1000 (20+980) | 1 | 2940 |
| 5 | 80(20+60) | 0 | 3920 |

**c.** Since fragments are only reassembled at the destination (H) R2 only has the responsibility of forwarding the 5 fragments it receives from R1 to H. Since all 5 fragments have size less than 1500 (the MTU of L2) R2 simply forwards the 5 fragments it receives, as is, without further fragmenting them.

**d.** The receiving host can tell (i) which fragment is the first (offset =0), which fragment is the last (flag = 0) and, given two fragments B and C, whether C comes immediately after B (if offset B + length B = offset C).

The receiving host then takes all fragments it receives with the same ID and checks to see if they can be put together from first to last without any gaps. If they can, that's the reconstructed datagram. If not, it behaves as if the full datagram was lost.

**14.3** The reasons an IP router may not forward packet may be divided into two types:

(i) Intended behavior; routers may intentionally discard some types of packet. Examples include:

Packet with TTL=0

Packet with IP header checksum error

Packet with an illegal option or control field

Packet for which there is no currently known destination (or to an illegal destination)

Packets that match a filter/firewall control list

Packets sent to the router itself

(ii) Fault; that is, unintended discard, following a fault or overload. Examples include:

Discard due to processing overload

Discard due to corruption while being stored (queued) within a router

Discard because there is no memory available to store the packet

Software error

Hardware or software reset

**14.4** 1000 nodes need 10 bits => 32 – 10 =22 bit prefixes needed

128.20.1110 00 00. 0000 0000/22 = 128.20.224.0/22

128.20.1110 01 00. 0000 0000/22 = 128.20.228.0/22

500 nodes need 9 bits => 32 –9 =23 bit prefixes needed

128.20.1110100 0. 0000 0000/23 = 128.20.232.0/23

128.20.1110101 0. 0000 0000/23 = 128.20.234.0/23

250 nodes need 8 bits => 32 –8 =24 bit prefixes needed

128.20.11101100. 0000 0000/24 = 128.20.236.0/24

128.20.11101101. 0000 0000/24 = 128.20.237.0/24

128.20.11101110. 0000 0000/24 = 128.20.238.0/24

Four more customer networks of size 50 each can be supported (because remaining space = 256 addresses, and minimum granularity = 64 nodes)

**14.5** Traceroute uses ICMP echo messages. These are addressed to the target IP address. The sender manipulates the TTL (hop count) value at the IP layer to force each hop in turn to return an error message. It starts with a TTL of 1, each router along the path decrements the TTL, and discards if zero, returning an ICMP message (which also indicated the router IP address of the router that discarded the message). The switches pass each IP packet the unmodified.

The router J receives the packet with TTL = 1 (it is on the path to B; It decrements the packet TTL. This reduces to zero, the router generates an error message and returns this to the sender (A)

Client A ----ICMP echo src= W, dst=Z, TTL=1------> Router J

Client A <---ICMP error src= Y, dst=W, TTL=64---- Router J

The client receives the ICMP error message and notes that J is one hop away on the path to B.

It then probes by sending the same packet with a TTL of 2.

The router J receives the packet (it is on the path to B); It decrements the packet TTL

This is greater than zero, the router forwards this along the path to the destination.

The router K receives the packet (it is on the path to B); It decrements the packet TTL

This reduces to zero; the router generates an error message and returns this to the sender (A)

It then probes by sending the same packet with a TTL of 3.

The routers J and K receive the packet (on the path to B); The packet is then forwarded to B

This responds with an echo message and returns this to the sender (A)
Receipt of the response indicates that this has reached the final destination.

**14.6** **a.** The data link address of the router's interface connected to network where host A is attached.

**b.** The IP address of host B.

**c.** Routers R1 and R3 will change the Time to Live field and header checksum fields.

**d.** Yes, an ARP query is needed. The answer will be issued by host B and it will be the data link address of host B's network interface

**14.7** **a.** 248 = 11111000
$2^5$ subnets

**b.** $2^{11}$ – 2 hosts

**c.** New sub-netmask = 255.255.[11110000].0 = 255.255.240.0

**14.8** **a.** There are 6 networks. This includes the fact that each link between a pair of routers is a network.

**b.** Possible addresses are:

220.23.16.0/27
220.23.16.32/27
220.23.16.64/27
220.23.16.96/27
220.23.16.128/27
220.23.16.160/27

**c.** See the figure for Problem 18.6.

**14.9** **a.** Yes. Host A can directly access host C access at the link layer, as the learning bridge is transparent to all the hosts.

**b.** Yes. The bridge will not forward the traffic form network segment S1 to S2 in this case, and the two network segments are partitioned into two independent collision domains.

**c.** No. Host A needs to use the same network segment (S1) upon which Host B and the WWW server is communicating on.

**d.** 8 entries

**e.** Yes. Host A sends a BROADCST frame out and the bridge will forward all the broadcast frames that it receives to the other segment.

**14.10** Hubs and switches are in the same broadcast domain

Routers separate broadcast domains

B is in the left broadcast domain

The frame is therefore received by A, B, C

The right broadcast domain does not receive the frame because the router does not forward it.

The frame is NOT received by D, E

# CHAPTER 15 TRANSPORT PROTOCOLS

**15.1** Frame Size = 1500 B (Ether MAC headers are not included)

Packet has the following headers:

IP header (20 B)

UDP header (8 B)

Total header in each packet = 28 B

Total UDP payload data is therefore 1500 – 28 = 1472 B. (i.e. 1472 × 8 bits)

Throughput = Total bits sent per second = 1472 × 8 × 50 = 58880 bps

**15.2** The Ethernet frame has the following structure:

Preamble (8B)

Link: MAC Header (14 B)

dst= B-mac address

src = Router mac address

type = 0x800 (IP)

Network: IP Header (20 B)

src = A IP address <--- IP header generate red by sender (A)

dst = B IP address

Transport: TCP Header (20 B)

DATA (100 B)

Link CRC-32 (4B)

**15.3** n + 1 (one for each connection and one for the listening socket)

**15.4** This scheme is not good since it cannot distinguish between the ACKs for the last byte or FIN.

**15.5** In this problem, there is no danger in overflowing the receiver since the receiver's receive buffer can hold the entire file. Also, because there is no loss and acknowledgements are returned before timers expire, TCP congestion control does not throttle the sender. However, the process in host A will not continuously pass data to the socket because the send buffer will quickly fill up. Once the send buffer becomes full, the process will pass data at an average rate or R < S.

**15.6** **a.** Application will get R/9 Kbps.
This is because after X opens one connection there will be 9 connections and TCP tries to allocate the rate fairly among all the connections.

**b.** It has to open 8 connections. At that point there will be 16 connections open. Since, as mentioned above, TCP tries to allocate the rate fairly among all the connections, X will receive 16/8=1/2 of the full rate.

**c.** In general it would not be able to maintain rate of R/2 because UDP flows do not have congestion control, and they will take as much of the bandwidth as they can

**15.7** We need to determine the size of the various headers and trailers:
Preamble (8 B) + MAC header (14 B) + IP header (20 B) + UDP header (8 B) and UDP payload (690 B) and CRC-32 (4 B) = 744 Bytes

# CHAPTER 16 ADVANCED DATA COMMUNICATIONS TOPICS

**16.1** **a.** Modulation index = (audio amplitude)/(carrier amplitude) = 15/60 = 0.25

Percent modulation = 25%

**b.** The audio signal is 1500 Hz; the carrier is 100,000 Hz = 100 kHz.

**c.** The frequency spectrum of the modulated signal consists of :

$f_c$, ($f_c + f_a$), and ($f_c - f_a$), which are respectively:

100,000 Hz, 101,500 Hz, and 98,500 Hz

**16.2** $s_a(t) = 20 \sin 2\pi(3000t) = 20 \sin 6000\,\pi t$

$s_c(t) = 50 \sin 2\pi(75 \times 10^6\ t) = 50 \sin 150 \times 10^6\ \pi t$

**16.3**

$$a = \frac{\text{Propagation Time}}{\text{Transmission Time}} = \frac{120 \times 10^{-6}}{\left(400 \times 8 \big/ 10 \times 10^6\right)} = \frac{120}{320} = 0.375$$

# CHAPTER 17 WIRELESS TRANSMISSION

**17.1** **a.** As = +1 × (-1, +1, -1, -1, +1, +1) = (-1, +1, -1, -1, +1, +1)

Bs = -1 × (+1, +1, -1, +1, -1, +1) = (-1, -1, +1, -1, +1, -1)

S = As + Bs + N = (-2, 0, 0, -2, +2, 0) + (-1, 0, +1, 0, -1, +1) = (-3, 0, +1, -2, +1, +1)

(-3, 0, +1, -2, +1, +1) is received by a receiver

**b.** Ar = (-3, 0, +1, -2, +1, +1) × (-1, +1, -1, -1, +1, +1) = 3 + 0 - 1 + 2 + 1 + 1 = 6 ) 1

Br = (-3, 0, +1, -2, +1, +1) × (+1, +1, -1, +1, -1, +1) = -3 - 1 - 2 - 1 + 1 = -6 ) 0

The data sent by A can be recognized as 1. The data sent by B can be recognized as 0

**17.2** The signal has 8 time periods in 1μs. So each chip should have 2 time periods of the signal. Since there are 4 chips per bit, we would have phase shifts every 2 time periods of the signal; at the 1 μs point, however, when we move from a 1 bit to a 0 bit, there will not be any phase shift since the chip change and bit change cancel each other.

# CHAPTER 19 ROUTING

**19.1** **a.** The following table shows the routing table for A at each step until convergence.

| S.no | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 |
| 2 | 0 | 1 | 1 | 2 | ∞ | 2 | ∞ | 2 | 1 |
| 3 | 0 | 1 | 1 | 2 | 3 | 2 | 3 | 2 | 1 |

**b.** A advertises a distance of infinity to B, but C and I advertise a distance of 2 to B. Depending on the ordering of the messages, this might happen: I upon knowing that B can be reached from C with distance 2, concludes that it can reach B with a distance of 3(via C) and advertises this to A. A concludes that it can reach B with a distance of B with a distance of 4 (via I), and advertises this to C. C concludes that it can reach B with a distance of 5 (via A). This continues with forever if the distance is unbounded. This is called count to infinity problem.

**19.2** Cost to destination via B

| | | | | |
|---|---|---|---|---|
| **D** | | | A | C |
| **E** | | A | 2 | ∞ |
| **S** | | C | ∞ | 3 |
| **T** | | D | 4 | ∞ |

**19.3** **a.** R6-R5-R2

**b.** The link costs are as follows

T1: 100/1.544 = 65

Ethernet: 100/10 = 10

Token ring: 100/16 = 6

Fast Ethernet: 100/100 = 1

Possible routes with their costs are:

R6-R5-R2 = 75

R6-R4-R1-R2 = 22

R6-R8-R3-R5-R2 = 13

R6-R8-R7-R4-R1-R2 = 33

So the least-cost route is R6-R8-R3-R5-R2

# CHAPTER 20 CONGESTION

**20.1** **a.** Congestion control is not needed, since if the sum of the input rates to any queue is less that the transmission rate, queues will never form.

**b.** Flow control is still needed in this scenario, since a sender can overwhelm a receiver with data. Flow control is a speed-matching issue between sender and receiver.

**c.** Either would be OK, since the network is not congested. There would be a slight preference for packet switching since an arriving packet does not have to wait for transmission if the link is free.

**20.2** 1 + 2 + 4 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 = 99 for flow A

1 + 2 + 4 + 8 + 9 + 10 = 34 for flow B

**20.3** One round-trip for connection establishment; then one round-trip to send request and get first segment (m bytes). At this point your window size is two. The number of round-trips before you get all the S bytes works out to $1 + \log_2\lceil(S/m)\rceil$ where $\lceil x \rceil$ is the smallest integer ≥ x.

**20.4** **a.** $2 \times d$

**b.** $T_{HTMLst} = 2RTT + 3 \times (MSS/b)$

Now, get one file:

$T_{Total} = 2 \times T_{HTMLst} = 4RTT + 6 \times (MSS/b)$

**c.** $T_{HTML\infty} = 3RTT + 3 \times (MSS/b)$

$T_{Total} = 2 \times T_{HTML\infty} = 6RTT + 6 \times (MSS/b)$

**20.5** $T_{HTML\infty}$ = 4RTT + 6 × (MSS/b)

# CHAPTER 21 INTERNETWORK OPERATION

**21.1** Indirect TCP isolates link better and retransmits lost packets quickly. A snooping TCP agent can also initiate retransmissions, just over the wireless link, but cannot maintain high throughput over the wired portion of the connection. If there is high variability in the quality of the wireless link, indirect TCP will outperform snooping TCP

# CHAPTER 24 ELECTRONIC MAIL, DNS, AND HTTP

**24.1** **a.** rutgers.edu

root

yale.edu

**b.** cs.rutgers.edu

**c.** rutgers.edu

root

yale.edu

**d.** yale.edu

root

rutgers.edu

cs.rutgers.edu

**e.** paul.cs.rutgers.edu → cs.rutgers.edu

cs.rutgers.edu → rutgers.edu

cs.rutgers.edu → root

cs.rutgers.edu → yale.edu

**24.2** Application layer protocols ⇒ DNS and HTTP

Transport layer protocols ⇒ UDP for DNS and TCP for HTTP

**24.3** The total amount of time to get the IP address is

$$\text{RTT1} + \text{RTT2} + \Lambda + \text{RTTn}$$

Where Λ denotes the total processing time (apart from round trip time) in Domain name resolution process. Once the IP address is known, RTT0 elapses to set up the TCP connection and another RTT0 elapses to request and receive the small object. The total response time is

$$\text{RTT0} + \text{RTT1} + \text{RTT2} + \Lambda + \text{RTTn}$$

# CHAPTER 26 COMPUTER AND NETWORK SECURITY THREATS

**26.1** All of these activities could create the right conditions to threaten the network.

**a.** The regular daily courier is familiar to employees, so they may not notice anything is wrong should that person walk into the server room.

**b.** Even with good severance packages and benefits, employees who lost their jobs due to downsizing may be disgruntled.

**c.** An employee's traveling to another location may not create a threat, but if the employee has a laptop computer that contains private information or the Web browser has saved passwords, then if the laptop is stolen, a hacker has gained valuable information.

**d.** If the sprinkler system went off, it could damage the company's servers and other computing equipment.

**26.2** Some of the ways by which hackers' compromise computers without code breaking are as follows.

- Key Catcher (hw or sw)
- Via email that has an executable file for an attachment.
- A boot CD that has its own Operating System

**26.3** A Null session problem is commonly a problem that exists on many Systems especially Microsoft based systems where the system allows a person or other system to connect to it without use of username and/or password such as Shares.

**26.4** There are many ways to achieve this.

By alerting administrators via email/pager/phone

By changing firewall configurations to

- increase logging of suspect sessions
- block certain sensitive areas inside or
- block offending areas outside
- throttle offending or suspect traffic
- bring down the Internet connection

There are customer filters that can be configured for signatures that an IDS system looks for and there are standard "out of the box" attack signatures that are known attacks. If the IDS is not configured properly it may send what are known as "false positives" or alerts to an over abundance of traffic, therefore overwhelming people with alerts. While such alerts are active responses, they (as stated above) may become overwhelming.

**26.5** **a.** Adversary

- Sees the messages: N, R, X
- Computes Hash(R)
- Computes X XOR Hash(R) = Hash(P)

Later on:

- Adversary Requests Login, submits N.
- Machine generates random number R'.
- Adversary computes Hash(R').
- Adversary computes y = Hash(P) XOR Hash(R').
- Adversary submits y, and logs in as user.

**b.** To strengthen, simply require the protocol to compute

Hash(R XOR P) instead of Hash(R) XOR Hash(P)

**26.6** The major problem is Buffer overrun. If the input string contains a newline character, then this will write past the end of the input buffer. In the worst case, the size of the string might double. For instance, if the caller allocates a buffer on the stack that is just large enough to hold the string, and passes it to escape(), then a stack-smashing attack would be possible.

Another problem is that memcpy() invokes undefined behavior when invoked on overlapping memory regions.

# CHAPTER 27 COMPUTER AND NETWORK SECURITY TECHNIQUES

**27.1** There is no definite answers to that but one can form an opinion by first considering what IPsec is and what it does. IPsec refers to a set of standards developed by the Internet Engineering Task Force (IETF). IPsec solves two problems that have plagued the IP protocol suite for years: host-to-host authentication (which will let hosts know that they're talking to the hosts they think they are) and encryption (which will prevent attackers from being able to watch the traffic going between machines).

Neither of these problems is what firewalls were created to solve. Although firewalls can help to mitigate some of the risks present on an Internet without authentication or encryption, there are really two classes of problems here: integrity and privacy of the information flowing between hosts and the limits placed on what kinds of connectivity is allowed between different networks. IPsec addresses the former class and firewalls the latter. Note however from Chapter 19 that IPsec does provide a limited type of firewall capability in that it allows the user to specify traffic processing rules for a variety of classes of traffic. This is a firewall type of service, but IPsec only provides a limited flexibility in this area.

**27.2** IPsec would reject the packets and would not pass them to TCP. In SSL, such packets could cause the session to break.