SOLUTIONS MANUAL

HIGH-SPEED NETWORKS AND INTERNETS

PERFORMANCE AND QUALITY OF SERVICE
SECOND EDITION

WILLIAM STALLINGS

TABLE OF CONTENTS

Chapter 2: Protocols and Architecture 1
Chapter 3: TCP and IP 3
Chapter 4: Frame Relay 8
Chapter 5: Asynchronous Transfer Mode 11
Chapter 6: High-Speed LANs 16
Chapter 7: Overview of Probability and Stochastic Processes 18
Chapter 8: Queuing Analysis 22
Chapter 9: Self-Similar Traffic 27
Chapter 10: Congestion Control in Data Networks and Internets 28
Chapter 11: Link Level Flow and Error Control 30
Chapter 12: TCP Traffic Control 34
Chapter 13: Traffic and Congestion Control in ATM Networks 37
Chapter 14: Overview of Graph Theory and Least-Cost Paths 40
Chapter 15: Interior Routing Protocols 45
Chapter 16: Exterior Routing Protocols and Multicast 47
Chapter 17: Integrated and Differentiated Services 49
Chapter 18: RSVP and MPLS 52
Chapter 19: Overview of Information Theory 54
Chapter 20: Lossless Compression 56
Chapter 21: Lossy Compression 61

NOTICE

This manual contains solutions to all of the review questions and homework problems in *High-Speed Networks and Internets*. If you spot an error in a solution or in the wording of a problem, I would greatly appreciate it if you would forward the information via email to me at ws@shore.net. An errata sheet for this manual, if needed, is available at ftp://shell.shore.net/members/w/s/ws/S/

W.S.

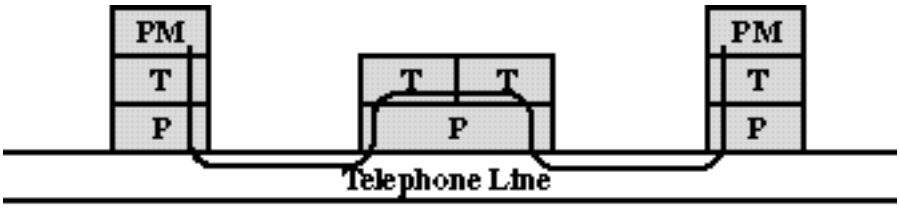
CHAPTER 2
PROTOCOLS AND ARCHITECTURE

2.1 The guest effectively places the order with the cook. The host communicates this order to the clerk, who places the order with the cook. The phone system provides the physical means for the order to be transported from host to clerk. The cook gives the pizza to the clerk with the order form (acting as a "header" to the pizza). The clerk boxes the pizza with the delivery address, and the delivery van encloses all of the orders to be delivered. The road provides the physical path for delivery.

2.2 **a.**

The PMs speak as if they are speaking directly to each other. For example, when the French PM speaks, he addresses his remarks directly to the Chinese PM. However, the message is actually passed through two translators via the phone system. The French PM's translator translates his remarks into English and telephones these to the Chinese PM's translator, who translates these remarks into Chinese.

b.



An intermediate node serves to translate the message before passing it on.

2.3 Perhaps the major disadvantage is the processing and data overhead. There is processing overhead because as many as seven modules (OSI model) are invoked to move data from the application through the communications software. There is data overhead because of the appending of multiple headers to the data. Another possible disadvantage is that there must be at least one protocol standard per layer. With so many layers, it takes a long time to develop and promulgate the standards.

2.4 No. There is no way to be assured that the last message gets through, except by acknowledging it. Thus, either the acknowledgment process continues forever, or one army has to send the last message and then act with uncertainty.

2.5 A case could be made either way. **First**, look at the functions performed at the network layer to deal with the communications network (hiding the details from the upper layers). The network layer is responsible for routing data through the network, but with a broadcast network, routing is not needed. Other functions, such as sequencing, flow control, error control between end systems, can be accomplished at layer 2, because the link layer will be a protocol directly between

the two end systems, with no intervening switches. So it would seem that a network layer is not needed. **Second**, consider the network layer from the point of view of the upper layer using it. The upper layer sees itself attached to an access point into a network supporting communication with multiple devices. The layer for assuring that data sent across a network is delivered to one of a number of other end systems is the network layer. This argues for inclusion of a network layer.

In fact, the OSI layer 2 is split into two sublayers. The lower sublayer is concerned with medium access control (MAC), assuring that only one end system at a time transmits; the MAC sublayer is also responsible for addressing other end systems across the LAN. The upper sublayer is called Logical Link Control (LLC). LLC performs traditional link control functions. With the MAC/LLC combination, no network layer is needed (but an internet layer may be needed).

2.6 The internet protocol can be defined as a separate layer. The functions performed by IP are clearly distinct from those performed at a network layer and those performed at a transport layer, so this would make good sense.

The session and transport layer both are involved in providing an end-to-end service to the OSI user, and could easily be combined. This has been done in TCP/IP, which provides a direct application interface to TCP.

2.7 **a.** No. This would violate the principle of separation of layers. To layer (N – 1), the N-level PDU is simply data. The (N – 1) entity does not know about the internal format of the N-level PDU. It breaks that PDU into fragments and reassembles them in the proper order.

b. Each N-level PDU must retain its own header, for the same reason given in (a).

CHAPTER 3
TCP AND IP

3.1 A single control channel implies a single control entity that can manage all the resources associated with connections to a particular remote station. This may allow more powerful resource control mechanisms. On the other hand, this strategy requires a substantial number of permanent connections, which may lead to buffers or state table overhead.

3.2 UDP provides the source and destination port addresses and a checksum that covers the data field. These functions would not normally be performed by protocols above the transport layer. Thus UDP provides a useful, though limited, service.

3.3 The IP entity in the source may need the ID and don't-fragment parameters. If the IP source entity needs to fragment, these two parameters are essential. Ideally, the IP source entity should not need to bother looking at the TTL parameter, since it should have been set to some positive value by the source IP user. It can be examined as a reality check. Intermediate systems clearly need to examine the TTL parameter and will need to examine the ID and don't-fragment parameters if fragmentation is desired. The destination IP entity needs to examine the ID parameter if reassembly is to be done and, also the TTL parameter if that is used to place a time limit on reassembly. The destination IP entity should not need to look at the don't-fragment parameter.

3.4 If intermediate reassembly is not allowed, the datagram must eventually be segmented to the smallest allowable size along the route. Once the datagram has passed through the network that imposes the smallest-size restriction, the segments may be unnecessarily small for later networks, degrading performance.

On the other hand, intermediate reassembly requires compute and buffer resources at the intermediate routers. Furthermore, all segments of a given original datagram would have to pass through the same intermediate node for reassembly, which would prohibit dynamic routing.

3.5 The header is a minimum of 20 octets.

3.6 Possible reasons for strict source routing: **(1)** to test some characteristics of a particular path, such as transit delay or whether or not the path even exists; **(2)** the source wishes to avoid certain unprotected networks for security reasons; **(3)** the source does not trust that the routers are routing properly.

Possible reasons for loose source routing: **(1)** Allows the source to control some aspects of the route, similar to choosing a long-distance carrier in the telephone network; **(2)** it may be that not all of the routers recognize all addresses and that for a particular remote destination, the datagram needs to be routed through a "smart" router.

3.7 A buffer is set aside big enough to hold the arriving datagram, which may arrive in fragments. The difficulty is to know when the entire datagram has arrived. Note that we cannot simply count the number of octets (bytes) received so far because duplicate fragments may arrive.

a. Each hole is described by a descriptor consisting of hole.first, the number of the first octet in the hole, relative to the beginning of the buffer, and hole.last, the number of the last octet. Initially, there is a single hole descriptor with hole.first = 1 and hole.last = some maximum value. Each arriving fragment is characterized by fragment.first and fragment.last, which can be calculated from the Length and Offset fields.

1. Select the next hole descriptor from the hole descriptor list. If there are no more entries, go to step eight.
2. If fragment.first is greater than hole.last, go to step one.
3. If fragment.last is less than hole.first, go to step one.
   (If either step two or step three is true, then the newly arrived fragment does not overlap with the hole in any way, so we need pay no further attention to this hole. We return to the beginning of the algorithm where we select the next hole for examination.)
4. Delete the current entry from the hole descriptor list.
   (Since neither step two nor step three was true, the newly arrived fragment does interact with this hole in some way. Therefore, the current descriptor will no longer be valid. We will destroy it, and in the next two steps we will determine whether or not it is necessary to create any new hole descriptors.)
5. If fragment.first is greater than hole.first, then create a new hole descriptor "new-hole" with new-hole.first equal to hole.first, and new-hole.last equal to fragment.first minus one.
   (If the test in step five is true, then the first part of the original hole is not filled by this fragment. We create a new descriptor for this smaller hole.)
6. If fragment.last is less than hole.last and fragment.more fragments is true, then create a new hole descriptor "new-hole", with new-hole.first equal to fragment.last plus one and new-hole.last equal to hole.last.
   (This test is the mirror of step five with one additional feature. Initially, we did not know how long the resembled datagram would be, and therefore we created a hole reaching from zero to (effectively) infinity. Eventually, we will receive the last fragment of the datagram. At this point, that hole descriptor which reaches from the last octet of the buffer to infinity can be discarded. The fragment which contains the last fragment indicates this fact by a flag in the internet header called "more fragments". The test of this bit creating in this statement prevents us from creating a descriptor for the unneeded hole which describes the space from the end of the datagram to infinity.)
7. Go to step one.
8. If the hole descriptor list is now empty, the datagram is now complete. Pass it on to the higher level protocol processor for further handling. Otherwise, return.

b. The hole descriptor list could be managed as a separate list. A simpler technique is to put each hole descriptor in the first octets of the hole itself. The descriptor must contain hole.last plus pointer to the predecessor and successor holes.

3.8 The original datagram includes a 20-octet header and a data field of 4460 octets. The Ethernet frame can take a payload of 1500 octets, so each frame can carry an IP datagram with a 20-octet header and 1480 data octets. Note the 1480 is divisible by 8, so we can use the maximum size frame for each fragment except the last. To fit 4460 data octets into frames that carry 1480 data octets we need:

3 datagrams × 1480 octets = 4440 octets, plus
1 datagram that carries 20 data octets (plus 20 IP header octets)
The relevant fields in each IP fragment:

Total Length = 1500 More Flag = 1 Offset = 0	Total Length = 1500 More Flag = 1 Offset = 185	Total Length = 1500 More Flag = 1 Offset = 370	Total Length = 40 More Flag = 0 Offset = 555

3.9 For computing the IP checksum, the header is treated as a sequence of 16-bit words. and the 16-bit checksum field is set to all zeros. The checksum is then formed by performing the one's complement addition of all words in the header, and then taking the one's complement of the result. We can express this as follows:

A = W(1) +' W(2) +' . . . +' W(M)
C = A

where

+' = one's complement addition
C = 16-bit checksum
A = one's complement summation made with the checksum value set to 0
A = one's complement of A
M = length of block in 16-bit words
W(i) = value of ith word

Suppose that the value in word k is changed by Z = new_value – old_value. Let A" be the value of A after the change and C" be the value of C after the change. Then

A" = A +' Z
C" = (A +' Z) = C +' Z

3.10 In general, those parameters that influence the progress of the fragments through the internet must be included with each fragment. RFC 791 lists the following options that must be included in each fragment: Security (each fragment must be treated with the same security policy); Loose Source Routing (each fragment must follow the same loose route); Strict Source Routing (each fragment must follow the same strict route).

RFC 791 lists the following options that should be included only in the first fragment: Record Route (unnecessary, and too much overhead to do in all fragments); Internet Timestamp (unnecessary, and too much overhead to do in all fragments).

3.11 Data plus transport header plus internet header equals 1820 bits. This data is delivered in a sequence of packets, each of which contains 24 bits of network header and up to 776 bits of higher-layer headers and/or data. Three network packets are needed. Total bits delivered = 1820 + 3 × 24 = 1892 bits.

3.12
- **Version:** Same field appears in IPv6 header; value changes from 4 to 6.
- **IHL:** Eliminated; not needed since IPv6 header is fixed length.
- **Type of Service:** This field is eliminated. Three bits of this field define an 8-level precedence value; this is replaced with the priority field, which defines an 8-

level precedence value for non-congestion-control traffic. The remaining bits of the type-of-service field deal with reliability, delay, and throughput; equivalent functionality may be supplied with the flow label filed.

- **Total Length**: Replaced by Payload Length field.
- **Identification:** Same field, with more bits, appears in Fragment header.
- **Flags:** The More bit appears in the Fragment header. In IPv6, only source fragmenting is allowed; therefore, the Don't Fragment bit is eliminated.
- **Fragment Offset:** Same field appears in Fragment header
- **Time to Live:** Replaced by Hop Limit field in IPv6 header, with in practice the same interpretation.
- **Protocol:** Replaced by Next Header field in IPv6 header, which either specifies the next IPv6 extension header, or specifies the protocol that uses IPv6.
- **Header Checksum**: This field is eliminated. It was felt that the value of this checksum was outweighed by the performance penalty.
- **Source Address**: The 32-bit IPv4 source address field is replaced by the 128-bit source address in the IPv6 header.
- **Destination Address**: The 32-bit IPv4 destination address field is replaced by the 128-bit source address in the IPv6 header.

3.13 The recommended order, with comments:

1. **IPv6 header:** This is the only mandatory header, and will indicate if one or more additional headers follow.
2. **Hop-by-Hop Options header:** After the IPv6 header is processed, a router will need to examine this header immediately afterwards to determine if there are any options to be exercised on this hop.
3. **Destination Options header:** For options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header. This header must precede the routing header, because it contains instructions to the node that receives this datagram and that will subsequently forward this datagram using the Routing header information.
4. **Routing header:** This header must precede the Fragment header, which is processed at the final destination, whereas the routing header must be processed by an intermediate node.
5. **Fragment header:** This header must precede the Authentication header, because authentication is performed based on a calculation on the original datagram; therefore the datagram must be reassembled prior to authentication.
6. **Authentication header:** The relative order of this header and the Encapsulating Security Payload header depends on the security configuration. Generally, authentication is performed before encryption so that the authentication information can be conveniently stored with the message at the destination for later reference. It is more convenient to do this if the authentication information applies to the unencrypted message; otherwise the message would have to be re-encrypted to verify the authentication information.
7. **Encapsulating Security Payload header:** see preceding comment.
8. **Destination Options header:** For options to be processed only by the final destination of the packet. These options are only to be processed after the datagram has been reassembled (if necessary) and authenticated (if necessary). If the Encapsulating Security Payload header is used, then this header is encrypted and can only be processed after processing that header.

There is no advantage to placing this header before the Encapsulating Security Payload header; one might as well encrypt as much of the datagram as possible for security.

CHAPTER 4
FRAME RELAY

4.1 The argument ignores the overhead of the initial circuit setup and the circuit teardown.

4.2 a. Circuit Switching

$T = C_1 + C_2$ where

C_1 = Call Setup Time

C_2 = Message Delivery Time

$C_1 = S = 0.2$

C_2 = Propagation Delay + Transmission Time

$= N \times D + L/B$

$= 4 \times 0.001 + 3200/9600 = 0.337$

$T = 0.2 + 0.337 = 0.537$ sec

Datagram Packet Switching

$T = D_1 + D_2 + D_3 + D_4$ where

D_1 = Time to Transmit and Deliver all packets through first hop

D_2 = Time to Deliver last packet across second hop

D_3 = Time to Deliver last packet across third hop

D_4 = Time to Deliver last packet across forth hop

There are $P - H = 1024 - 16 = 1008$ data bits per packet. A message of 3200 bits require four packets (3200 bits / 1008 bits / packet = 3.17 packets which we round up to 4 packets).

$D_1 = 4 \times t + p$ where

t = transmission time for one packet

p = propagation delay for one hop

$D_1 = 4 \times (P/B) + D$

$= 4 \times (1024/9600) + 0.001$

$= 0.428$

$D_2 = D_3 = D_4 = t + p$

$= (P/B) + D$

$= (1024/9600) + 0.001 = 0.108$

$T = 0.428 + 0.108 + 0.108 + 0.108$

$= 0.752$ sec

Virtual Circuit Packet Switching

$T = V_1 + V_2$ where

V_1 = Call Setup Time

V_2 = Datagram Packet Switching Time

$T = S + 0.752 = 0.2 + 0.752 = 0.952$ sec

b. Circuit Switching vs. Diagram Packet Switching

T_c = End-to-End Delay, Circuit Switching

$T_c = S + N \times D + L/B$

T_d = End-to-End Delay, Datagram Packet Switching

N_p = Number of packets = $\left\lceil \frac{L}{P-H} \right\rceil$

$T_d = D_1 + (N-1)D_2$

D_1 = Time to Transmit and Deliver all packets through first hop

D_2 = Time to Deliver last packet through a hop

$D_1 = N_p(P/B) + D$

$D_2 = P/B + D$

$T = (N_p + N - 1)(P/B) + N \times D$

$T = T_d$

$S + L/B = (N_p + N - 1)(P/B)$

Circuit Switching vs. Virtual Circuit Packet Switching

T_V = End-to-End Delay, Virtual Circuit Packet Switching

$T_V = S + T_d$

$T_C = T_V$

$L/B = (N_p + N - 1)P \times B$

Datagram vs. Virtual Circuit Packet Switching

$T_d = T_V - S$

4.3 From Problem 4.2, we have

$$T_d = (N_p + N - 1)(P/B) + N \times D$$

For maximum efficiency, we assume that $N_p = L/(P - H)$ is an integer. Also, it is assumed that D = 0. Thus

$$T_d = (L/(P - H) + N - 1)(P/B)$$

To minimize as a function of P, take the derivative:

$0 = dT_d/(dP)$

$0 = (1/B)(L/(P - H) + N - 1) - (P/B)L/(P - H)^2$

$0 = L(P - H) + (N - 1)(P - H)^2 - LP$

$0 = -LH + (N - 1)(P - H)^2$

$(P - H)^2 = LH/(N - 1)$

$$P = H + \sqrt{\frac{LH}{N-1}}$$

4.4 The number of hops is one less than the number of nodes visited.

a. The fixed number of hops is 2.

b. The furthest distance from a station is half-way around the loop. On average, a station will send data half this distance. For an N-node network, the average number of hops is (N / 4) – 1.

c. 1.

4.5 The mean node-node path is twice the mean node-root path. Number the levels of the tree with the root as 1 and the deepest level as N. The path from the root to level N requires N – 1 hops and 0.5 of the nodes are at this level. The path from the root to level N – 1 has 0.25 of the nodes and a length of N – 2 hops. Hence the mean path length, L, is given by

$$L = 0.5 \times (N - 1) + 0.25 \times (N - 2) + 0.125 \times (N - 3) + \ldots$$

$$L = \sum_{i=1}^{\infty} N(0.5)^i - \sum_{i=1}^{\infty} i(0.5)^i = N - 2$$

Thus the mean node-node path is 2N – 4

4.6 Yes. Errors are caught at the link level, but this only catches transmission errors. If a packet-switching node fails or corrupts a packet, the packet will not be delivered correctly. A higher-layer end-to-end protocol, such as TCP, must provide end-to-end reliability, if desired.

4.7 Reset collision is like clear collision. Since both sides know that the other wants to reset the circuit, they just reset their variables.

4.8 On each end, a virtual circuit number is chosen from the pool of locally available numbers and has only local significance. Otherwise, there would have to be global management of numbers.

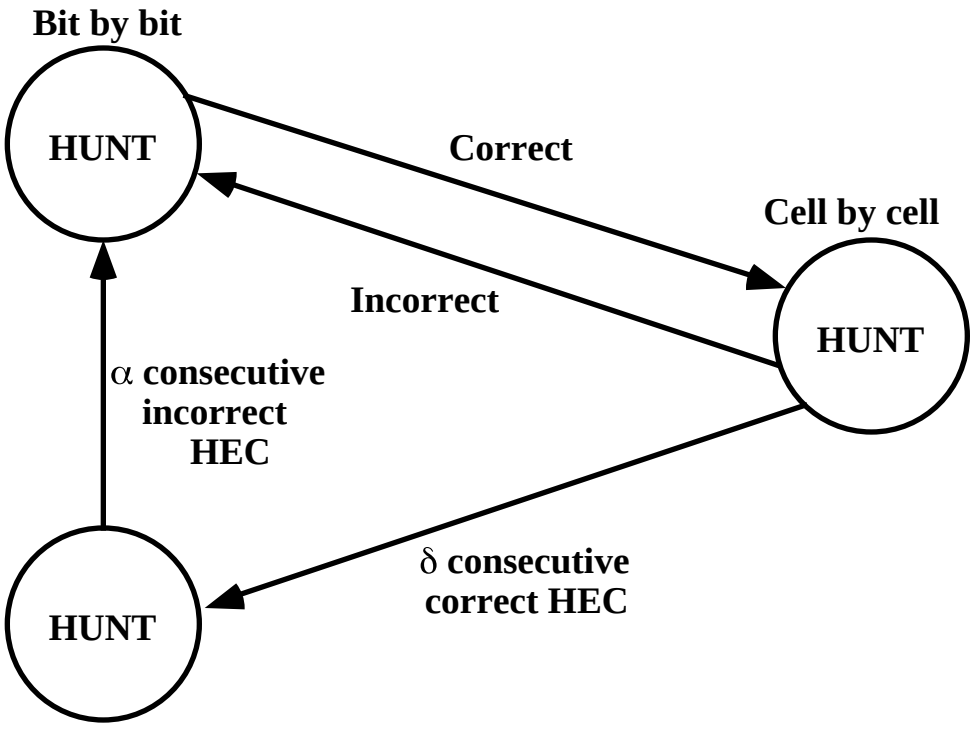
CHAPTER 5
ASYNCHRONOUS TRANSFER MODE

5.1

Controlling → controlled		Controlled → controlling	
0 0 0 0	NO_HALT, NULL	0 0 0 0	Terminal is uncontrolled. Cell is assigned or on an uncontrolled ATM connection.
1 0 0 0	HALT, NULL_A, NULL_B	0 0 0 1	Terminal is controlled. Cell is unassigned or on an uncontrolled ATM connection.
0 1 0 0	NO_HALT, SET_A, NULL_B	0 1 0 1	Terminal is controlled. Cell on a controlled ATM connection Group A.
1 1 0 0	HALT, SET_A, NULL_B	0 0 1 1	Terminal is controlled. Cell on a controlled ATM connection Group B.
0 0 1 0	NO_HALT, NULL_A, SET_B		
1 0 1 0	HALT, NULL_A, SET_B		
0 1 1 0	NO_HALT, SET_A, SET_B		
1 1 1 0	HALT, SET_A, SET_B		

All other values are ignored.

5.2



The procedure is as follows:

1. In the HUNT state, a cell delineation algorithm is performed bit by bit to determine if the HEC coding law is observed (i.e., match between received HEC and calculated HEC). Once a match is achieved, it is assumed that one header has been found, and the method enters the PRESYNCH state.
2. In the PRESYNCH state, a cell structure is now assumed. The cell delineation algorithm is performed cell by cell until the encoding law has been confirmed δ times consecutively.
3. In the SYNCH state, the HEC is used for error detection and correction (see Figure 17.17). Cell delineation is assumed to be lost if the HEC coding law is recognized as incorrect α times consecutively.

The values of α and δ are design parameters. Greater values of δ result in longer delays in establishing synchronization but in greater robustness against false delineation. Greater values of α result in longer delays in recognizing a misalignment but in greater robustness against false misalignment.

5.3 **a.** We reason as follows. A total of X octets are to be transmitted. This will require a total of $\left\lceil \frac{X}{L} \right\rceil$ cells. Each cell consists of (L + H) octets, where L is the number of data field octets and H is the number of header octets. Thus

$$N = \frac{X}{\left\lceil \frac{X}{L} \right\rceil (L + H)}$$

The efficiency is optimal for all values of X which are integer multiples of the cell information size. In the optimal case, the efficiency becomes

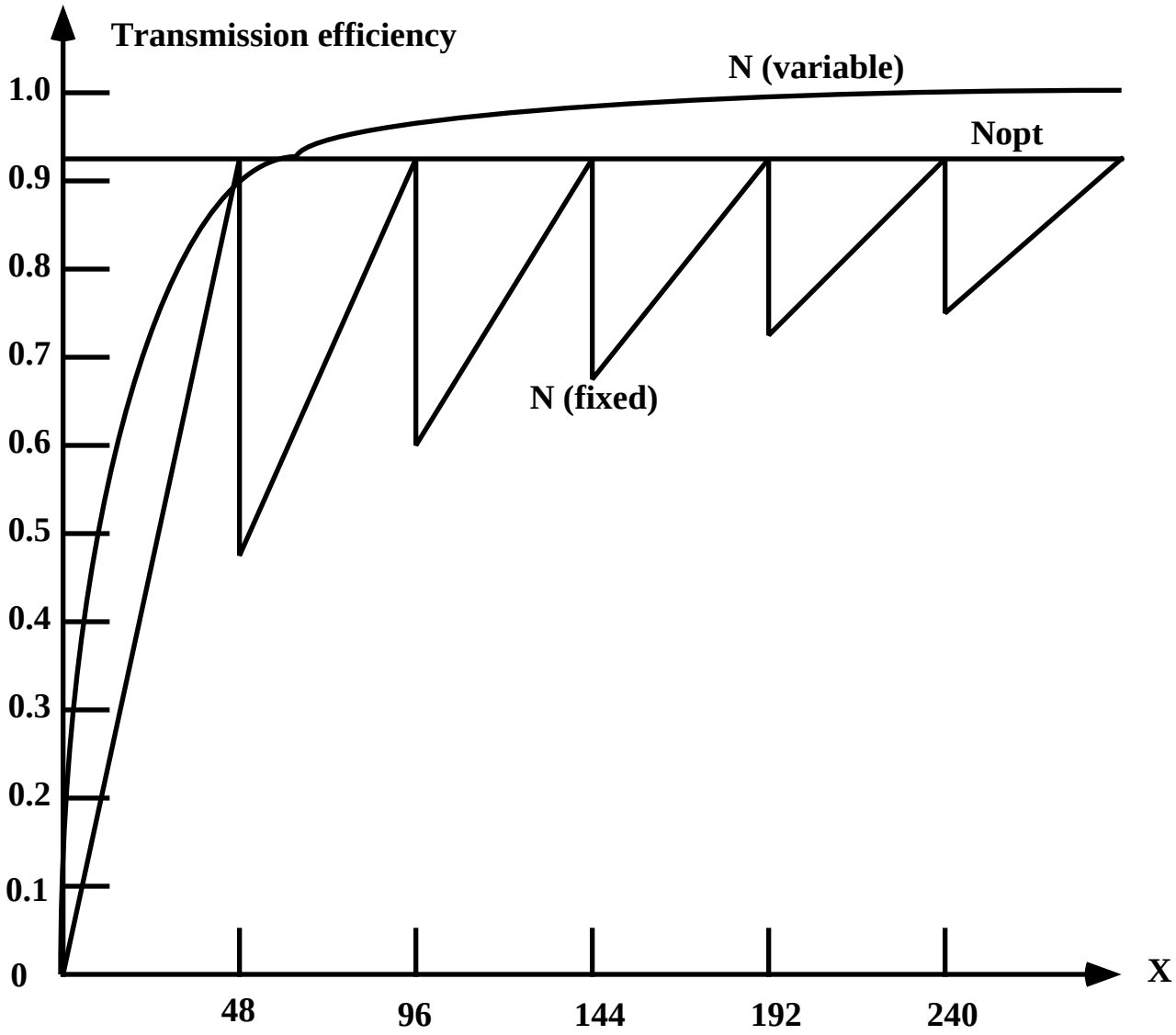
$$N_{\text{opt}} = \frac{X}{\frac{X}{L} + H} = \frac{L}{L + H}$$

For the case of ATM, with L = 48 and H = 5, we have $N_{\text{opt}} = 0.91$

b. Assume that the entire X octets to be transmitted can fit into a single variable-length cell. Then

$$N = \frac{X}{X + H + H_v}$$

c.

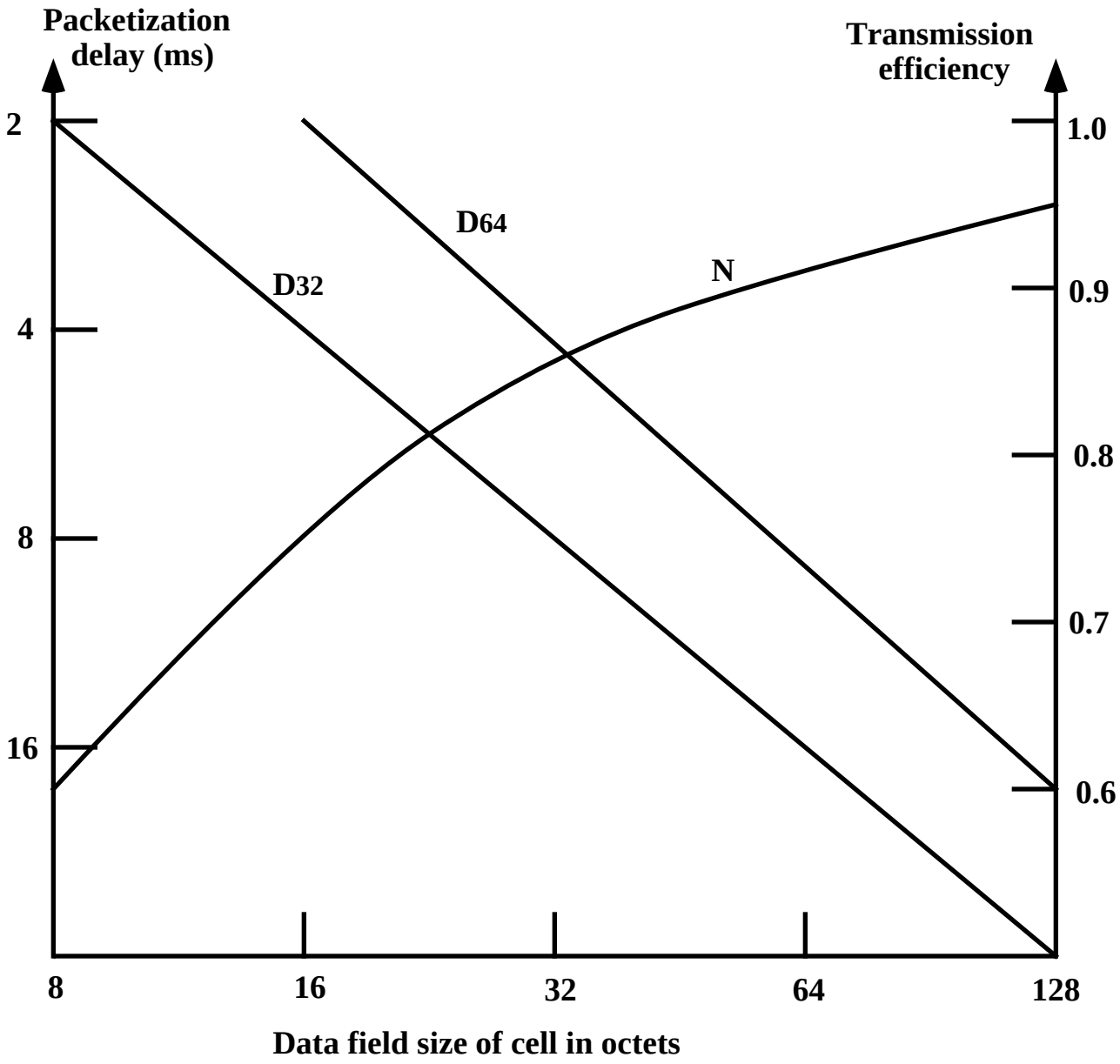


N for fixed-sized cells has a sawtooth shape. For long messages, the optimal achievable efficiency is approached. It is only for very short cells that efficiency is rather low. For variable-length cells, efficiency can be quite high, approaching 100% for large X. However, it does not provide significant gains over fixed-length cells for most values of X.

5.4 **a.** As we have already seen in Problem 5.3: $N = \frac{L}{L+H}$

b. $D = \frac{8 \times L}{R}$

c.



A data field of 48 octets, which is what is used in ATM, seems to provide a reasonably good tradeoff between the requirements of low delay and high efficiency.

5.5 a. The transmission time for one cell through one switch is $t = (53 \times 8) / (43 \times 10^6) = 9.86\mu s$.

b. The maximum time from when a typical video cell arrives at the first switch (and possibly waits) until it is finished being transmitted by the 5th and last one is $2 \times 5 \times 9.86\mu s = 98.6\mu s$.

c. The average time from the input of the first switch to clearing the fifth is $(5 + 0.6 \times 5 \times 0.5) \times 9.86\mu s = 64.09\mu s$.

d. The transmission time is always incurred so the jitter is due only to the waiting for switches to clear. In the first case the maximum jitter is $49.3\mu s$. In the second case the average jitter is $64.09 - 49.3 = 14.79\mu s$.

5.6 **a.** The reception of a valid BOM is required to enter reassembly mode, so any subsequent COM and EOM cells are rejected. Thus, the loss of a BOM results in the complete loss of the PDU.

b. Incorrect SN progression between SAR-PDUs reveals the loss of a COM, except for the cases covered by (c) and (d) below. The result is that at least the first 40 octets of the AAL-SDU is correctly received, namely the BOM that originally began the reassembly, and any subsequent COMs up to the point where cell loss occurred. Data from the BOM through the last SAR-PDU received with a correct SN can be passed up to the AAL user as a partial CPCS-PDU.

c. The SN wraps around so that there is no incorrect SN progression, but the loss of data is detected by the CPCS-PDU being undersized. In this case, only the BOM may be legitimately retrieved.

d. The same answer as for (c).

5.7 **a.** If the BOM of the second block arrives before the EOM of the first block, then the partially reassembled first block must be released. The entire partially reassembled CPCS-PDU received to that point is considered valid and passed to the AAL user along with an error indication. This mechanism works when just the EOM is lost or when a cell burst knocks out some COMs followed by the EOM.

b. This might be detected by the SN mechanism. If not, the loss will be detected when the Btag and Etag fields fail to match. The Length indication may fail to pick up this error if the cell bust loses as many cells as are added by concatenation the two CPCS-PDU fragments. In this case only the first BOM may be legitimately retrieved.

5.8 **a.** Single bit errors are not picked up until the CPCS-PDU CRC is calculated, and they result in the discarding of the entire CPCS-PDU.

b. Loss of a cell with SDU=0 is detected by an incorrect CRC. If the CRC fails to catch the error, the Length field mismatch ensures that the CPCS-PDU is discarded.

c. Loss of a cell with SDU=1 is detectable in three ways. First, the SAR-PDUs of the following CPCS-PDU may be appended to the first, resulting in a CRC error or Length mismatch being flagged when the second CPCS-PDU trailer arrives. Second, the AAL may enforce a length limit which, if exceeded while appending the second CPCS-PDU, can flag an error and cause the assembled data to be discarded. Third, a timer may be attached to the CPCS-PDU reassembly; if it expires before the CPCS-PDU is completely received, the assembled data is discarded.

CHAPTER 6
HIGH-SPEED LANS

6.1 **a.** Assume a mean distance between stations of 0.375 km. This is an approximation based on the following observation. For a station on one end, the average distance to any other station is 0.5 km. For a station in the center, the average distance is 0.25 km. With this assumption, the time to send equals transmission time plus propagation time.

$$T = \frac{10^3 \text{ bits}}{10^7 \text{ bps}} + \frac{375 \text{ m}}{200 \times 10^6 \text{ m/sec}} = 102\mu \text{ sec}$$

b.

$$T_{\text{interfere}} = \frac{375 \text{ m}}{200 \times 10^6 \text{ m/sec}} = 1.875\mu \text{ sec}$$

$$T_{\text{interfere(bit-times)}} = 10^7 \times 1.875 \times 10^{-6} = 18.75 \text{ bit} - \text{times}$$

6.2 **a.** Again, assume a mean distance between stations of 0.375 km.

$$T = \frac{10^3 \text{ bits}}{10^8 \text{ bps}} + \frac{375 \text{ m}}{200 \times 10^6 \text{ m/sec}} = 12\mu \text{ sec}$$

b.

$$T_{\text{interfere}} = \frac{375 \text{ m}}{200 \times 10^6 \text{ m/sec}} = 1.875\mu \text{ sec}$$

$$T_{\text{interfere(bit-times)}} = 10^8 \times 1.875 \times 10^{-6} = 187.5 \text{ bit} - \text{times}$$

6.3 The fraction of slots wasted due to multiple transmission attempts is equal to the probability that there will be 2 or more transmission attempts in a slot.

Pr[2 or more attempts] = 1 – Pr[no attempts] – Pr[exactly 1 attempt]

$= 1 - (1 - p)^N - Np(1 - p)^{N-1}$

6.4 Slot time = Propagation Time + Transmission Time

= 10^3m / (2 × 10^8 m / sec) + 10^2 bits / 10^7 bps

= 15×10^{-6} sec

Slot rate = 1 / Slot time = 6.67×10^4

Data rate = 100 bits / slot × Slot rate = 6.67×10^6 bps

Data rate per station for N stations = $(6.67 \times 10^6) / N$

6.5 Define

PF_i = Pr[fail on attempt i]

P_i = Pr[fail on first (i – 1) attempts, succeed on i^{th}]

Then, the mean number of retransmission attempts before one station successfully retransmits is given by:

$$E\left[\text{retransmissions}\right] = \sum_{i=1}^{\infty} iP_i$$

For two stations:

$PF_i = 1/2^K$ where K = MIN[i, 10]

$$P_i = \left(1 - PF_i\right) \times \prod_{j=1}^{i-1} PF_j$$

$PF_1 = 0.5$; $P_1 = 0.5$

$PF_2 = 0.25$; $P_2 = 0.375$

$PF_3 = 0.125$; $P_3 = 0.109$

$PF_4 = 0.0625$; $P_4 = 0.015$

The remaining terms are negligible

E[retransmissions] = 1.637

CHAPTER 7
OVERVIEW OF PROBABILITY AND STOCHASTIC PROCESSES

7.1 You should always switch to the other box. The box you initially chose had a 1/3 probability of containing the banknote. The other two, *taken together*, have a 2/3 probability. But once I open the one that is empty, the other *now* has a 2/3 probability all by itself. Therefore by switching you raise your chances from 1/3 to 2/3.

Don't feel bad if you got this wrong. Because there are only two boxes to choose from at the end, there is a strong intuition to feel that the odds are 50-50. Marilyn Vos Savant published the correct solution in *Parade Magazine*, and subsequently (December 2, 1990) published letters from several distinguished mathematicians blasting her (they were wrong, not her). A cognitive scientist at MIT presented this problem to a number of Nobel physicists and they systematically gave the wrong answer (Piattelli-Palmarini, M. "Probability: Neither Rational nor Capricious." *Bostonia*, March 1991).

7.2 The same *Bostonia* article referenced above reports on an experiment with this problem. Most subjects gave the answer 87%. The vast majority, including many physicians, gave a number above 50%. We need Bayes' Theorem to get the correct answer:

$$\Pr[\text{Disease}/\text{positive}] = \frac{\Pr[\text{positive}/\text{Disease}]\Pr[\text{Disease}]}{\Pr[\text{positive}/\text{Disease}]\Pr[\text{Disease}] + \Pr[\text{positive}/\text{Well}]\Pr[\text{Well}]}$$

$$= \frac{(0.87)(0.01)}{(0.87)(0.01) + (0.13)(0.99)} = 0.063$$

Many physicians who guessed wrong lamented, "If you are right, there is no point in making clinical tests!"

7.3 Let WB equal the event {witness reports Blue cab}. Then:

$$\Pr[\text{Blue}/\text{WB}] = \frac{\Pr[\text{WB}/\text{Blue}]\Pr[\text{Blue}]}{\Pr[\text{WB}/\text{Blue}]\Pr[\text{Blue}] + \Pr[\text{WB}/\text{Green}]\Pr[\text{Green}]}$$

$$= \frac{(0.8)(0.15)}{(0.8)(0.15) + (0.2)(0.85)} = 0.41$$

This example, or something similar, is referred to as "the juror's fallacy."

7.4 **a.** If $K > 365$, then it is impossible for all values to be different. So, we assume $K \le 365$. Now, consider the number of different ways, N, that we can have K values with no duplicates. We may choose any of the 365 values for the first item, any of the remaining 364 numbers for the second item, and so on. Hence, the number of different ways is:

$$N = 365 \times 364 \times \ldots \times (365 - K + 1) = \frac{365!}{(365 - K)!}$$

If we remove the restriction that there are no duplicates, then each item can be any of 365 values, and the total number of possibilities is 365^K. So the probability of no duplicates is simply the fraction of sets of values that have no duplicates out of all possible sets of values:

$$Q(K) = \frac{365!/(365 - K)!}{365^K} = \frac{365!}{(365 - K)! \times 365^K}$$

b. $P(K) = 1 - Q(K) = 1 - \dfrac{365!}{(365 - K)! \times 365^K}$

Many people would guess that to have a probability greater than 0.5 that there is at least one duplicate, the number of people in the group would have to be about 100. In fact, the number is 23, with P(23) = 0.5073. For $K = 100$, the probability of at least one duplicate is 0.9999997.

7.5 **a.** The sample space consists of the 36 equally probable ordered pairs of integers from 1 to 6. The distribution of X is given in the following table.

x_i	1	2	3	4	5	6
p_i	1/36	3/36	5/36	7/36	9/36	11/36

For example, there are 3 pairs whose maximum is 2: (1, 2), (2, 2), (2, 1).

b. $\mu = E[X] = 1\left(\frac{1}{36}\right) + 2\left(\frac{3}{36}\right) + 3\left(\frac{5}{36}\right) + 4\left(\frac{7}{36}\right) + 5\left(\frac{9}{36}\right) + 6\left(\frac{11}{36}\right) = \frac{161}{36} \approx 4.5$

$E[X^2] = 1\left(\frac{1}{36}\right) + 4\left(\frac{3}{36}\right) + 9\left(\frac{5}{36}\right) + 16\left(\frac{7}{36}\right) + 25\left(\frac{9}{36}\right) + 36\left(\frac{11}{36}\right) = \frac{791}{36} \approx 22.0$

$\mathrm{Var}[X] = E[X^2] - \mu^2 = 1.75 \qquad \sigma_X = \sqrt{1.75} = 1.3$

7.6 **a.**

x_i	-1	2	3	-4	5	-6
p_i	1/6	1/6	1/6	1/6	1/6	1/6

b. E[X] = –1/6. On average the player loses, so the game is unfair

7.7 **a.**

x_i	-E	E	2E	3E
p_i	125 / 216	75 / 216	15 / 216	1 / 216

b. E[X] = – 17E / 216 = -0.8E. Here's another game to avoid.

7.8 **a.** $E[X^2] = Var[X] + \mu^2 = 2504$
b. Var[2X + 3] = 4Var[X] + Var[3] = 16; standard deviation = 4
c. $Var[-X] = (-1)^2 Var[X] = 4$; standard deviation = 2

7.9 The probability density function f(r) must be a constant in the interval (900, 1100) and its integral must equal 1; therefore f(r) = 1 / 200 in the interval (900, 1100). Then

$$\Pr[950 \le r \le 1050] = \frac{1}{200}\int_{950}^{1050} dr = 0.5$$

7.10 Let Z = X + Y

$$\begin{aligned} Var[Z] &= E[(Z - \mu_Z)^2] = E[(X + Y - \mu_X - \mu_Y)^2] \\ &= E[(X - \mu_X)^2] + E[(Y - \mu_Y)^2] + 2E[(X - \mu_X)(Y - \mu_Y)] \\ &= Var[X] + Var[Y] + 2r(X, Y)\sigma_X\sigma_Y \end{aligned}$$

The greater the correlation coefficient, the greater the variance of the sum. A similar derivation shows that the greater the correlation coefficient, the smaller the variance of the difference.

7.11 E[X] = Pr[X = 1]; E[Y] = Pr[Y = 1]; E[XY] = Pr[X = 1, Y = 1]

If X and Y are uncorrelated, then r(X, Y) = Cov[X, Y] = 0. By the definition of covariance, we therefore have E[XY] = E[X]E[Y]. Therefore

Pr[X = 1, Y = 1] = Pr[X = 1]Pr[Y = 1]

By similar reasoning, you can show that the other three joint probabilities are also products of the corresponding marginal (single variable) probabilities. This proves that P(x, y) = P(x)P(y) for all (x, y), which is the definition of independence.

7.12 **a.** No. X and Y are dependent because the value of X determines the value of Y.
b. E[XY] = (–1)(1)(0.25) + (0)(0)(0.5) + (1)(1)(0.25) = 0
E[X] = (–1)(0.25) + (0)(0.5) + (1)(0.25) = 0
E[Y] = (1)(0.25) + (0)(0.5) + (1)(0.25) = 0.5
Cov(X, Y) = E[XY] – E[X]E[Y] = 0
c. X and Y are uncorrelated because Cov(X, Y) = 0

7.13 $\mu(t) = E[g(t)] = g(t)$; $Var[g(t)] = 0$; $R(t_1, t_2) = g(t_1)g(t_2)$

7.14 E[Z] = μ(5) = 3; E[W] = μ(8) = 3
$E[Z^2] = R(5, 5) = 13$; $E[W^2] = R(8, 8) = 13$

$E[ZW] = R(5, 8) = 9 + 4\exp(-0.6) = 11.195$
$Var[Z] = E[Z^2] - (E[Z])^2 = 4$; $Var[W] = 4$
$Cov(Z, W) = E[ZW] - E[Z]E[W] = 2.195$

7.15 We have $E[Z_i] = 0$ $\quad$ $Var[Z_i] = E[Z_i^2] = 1$ $\quad$ $E[Z_i Z_j] = 0$ for $i \neq j$

The autocovariance: $C(t_m, t_{m+n}) = R(t_m, t_{m+n}) - E[Z_m]E[Z_{m+n}] = E[Z_m Z_{m+n}]$

$$E[Z_m Z_{m+n}] = E\left[\left(\sum_{i=0}^{K} \alpha_i Z_{m-i}\right)\left(\sum_{j=0}^{K} \alpha_j Z_{m+n-j}\right)\right]$$

Because $E[Z_i Z_j] = 0$, only the terms with $(m - i) = (m + n - j)$ are nonzero. Using $E[Z_i^2] = 1$, we have

$$C[t_m t_{m+n}] = \sum_{i=0}^{K} \alpha_i \alpha_{n+i} \quad \text{with the convention that } \alpha_j = 0 \text{ for } j > K$$

The covariance does not depend on m and so the sequence is stationary.

7.16 $E[A] = E[B] = 0$; $E[A^2] = E[B^2] = 1$; $E[AB] = 0$; $E[X_n] = 0$

$$\begin{aligned} C(t_m, t_{m+n}) &= R(t_m, t_{m+n}) - E[X_m]E[X_{m+n}] = E[X_m X_{m+n}] \\ &= E[(A\cos(m\lambda) + B\sin(m\lambda))(A\cos((m+n)\lambda) + B\sin((m+n)\lambda))] \\ &= \cos(m\lambda)\cos((m+n)\lambda) + \sin(m\lambda)\sin((m+n)\lambda) = \cos(n\lambda) \end{aligned}$$

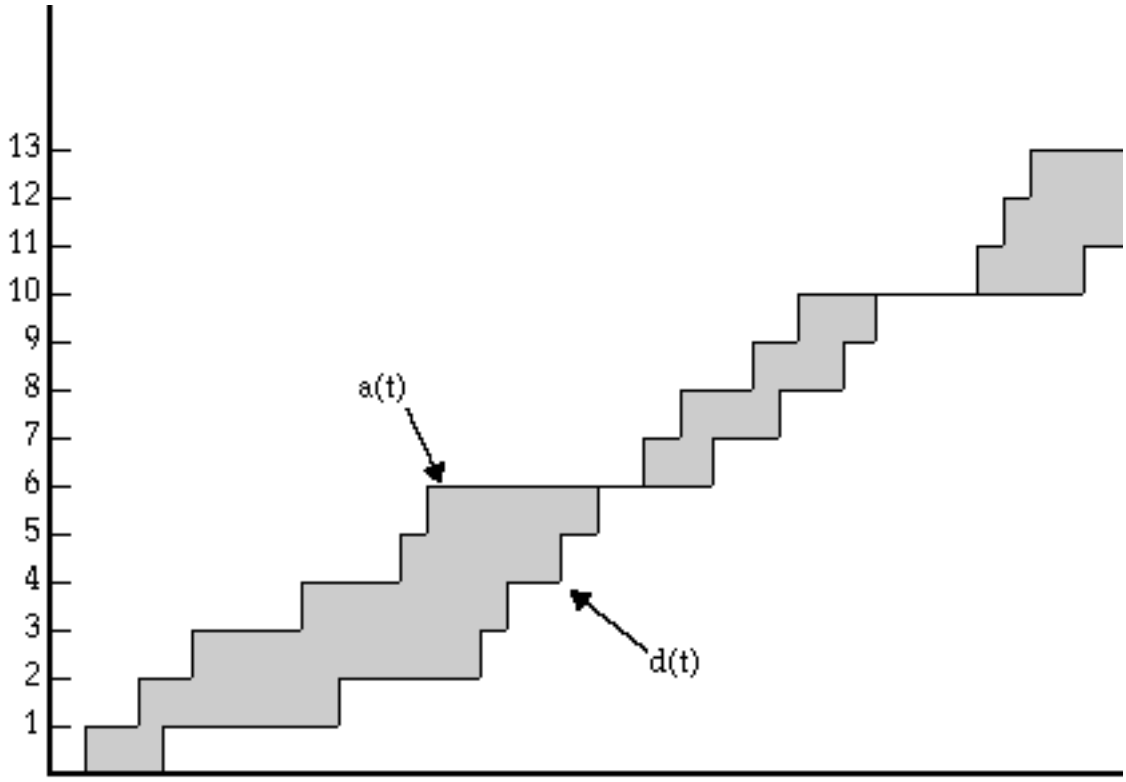
The final step uses the identities

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta; \quad \cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$

CHAPTER 8
QUEUING ANALYSIS

8.1 Consider the experience of a single item arriving. When the item arrives, it will find on average *r* items already in the queue or being served. When the item completes service and leaves the system, it will leave behind on average the same number of items in the queue or being served, namely *r*. This is in accord with saying that the average number in the system is *r*. Further, the average time that the item was in the system is T_r. Since items arrive at a rate of λ, we can reason that in the time T_r, a total of λT_r items must have arrived. Thus $r = \lambda T_r$.

8.2 a.



The height of the shaded portion equals n(t).

b. Select a time period that begins and ends with the system empty. Without loss of generality, set the beginning of the period as t = 0 and the end as t = τ. Then a(0) = d(0) = 0 and a(τ) = d(τ). The shaded area can be computed in two different ways, which must yield the same result:

$$\int_0^{\tau} \left[a(t) - d(t)\right] dt = \sum_{k=1}^{a(\tau)} t_k$$

If we view the shaded area as a sequence (from left to right) of vertical rectangles, we get the integral on the left. We can also view the shaded area as a sequence (from bottom to top) of horizontal rectangles with unit height and

width t_k, where t_k is the time that the kth item remains in the system; then we get the summation on the right. Therefore:

$$\int_0^{\tau} [n(t)]dt = a(\tau)T_r$$

$$\tau r = \lambda \tau T_r$$

$$r = \lambda T_r$$

8.3 $T_q = q / \lambda = 8 / 18 = 0.44$ hours

8.4 No. $12.356 \neq 25.6 \times 7.34$

8.5 If there are N servers, each server expects $\lambda T / N$ customers during time T. The expected time for that server to service all these customers is $\lambda T T_s / N$. Dividing by the length of the period T, we are left with a utilization of $\lambda T_s / N$.

8.6 $\rho = \lambda T_s = 2 \times 0.25 = 0.5$
Average number in system = $r = \rho / (1 - \rho) = 1$
Average number in service = $r - w = \rho = 0.5$

8.7 We need to use the solution of the quadratic equation, which says that for $ax^2 + bx + c = 0$, we have $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

$w = \rho^2 / (1 - \rho) = 4$

$\rho^2 + 4\rho - 4 = 0$

$\rho = -2 + \sqrt{8} = 0.828$

8.8 $\rho = \lambda T_s = 0.2 \times 2 = 0.4$

$T_w = \rho T_s / (1 - \rho) = 0.8 / 0.6 = 4 / 3$ minutes = 80 seconds

$T_r = T_w + T_s = 200$ seconds

$$m_{T_w}(90) = \frac{T_w}{\rho} \ln\left(\frac{100\rho}{100 - 90}\right) = \frac{1.333}{0.4} \ln(4) = 4.62 \text{ minutes}$$

8.9 T_s = (1000 octets × 8 bits / octet) / 9600 bps = 0.833 secs. $\rho = 0.7$

Constant-length message: $T_w = \rho T_s / (2 - 2\rho) = 0.972$ secs.

Exponentially-distributed: $T_w = \rho T_s / (1 - \rho) = 1.944$ secs.

8.10 $T_s = (1)(0.7) + (3)(0.2) + 10(0.1) = 2.3$ ms

Define S as the random variable that represents service time. Then

$$\sigma_{T_s}^2 = \text{Var}[S] = E\left[(S - T_s)^2\right] = (1-2.3)^2(0.7) + (3-2.3)^2(0.2) + (10-2.3)^2(0.1) = 7.21 \text{ms}$$

Using the equations in Figure 8.6a:

$A = 0.5\,(1 + (7.21/5.29)) = 1.18$

a. $\rho = \lambda T_s = 0.33 \times 2.3 = 0.767$

$T_r = T_s + (\rho T_s A)/(1-\rho) = 2.3 + (0.767 \times 2.3 \times 1.18)/(0.233) = 11.23$ ms

$r = \rho + (\rho^2 A)/(1-\rho) = 3.73$ messages

b. $\rho = \lambda T_s = 0.25 \times 2.3 = 0.575$

$T_r = 2.3 + (0.575 \times 2.3 \times 1.18)/(0.425) = 5.97$ ms

$r = 0.575 + (0.575 \times 0.575 \times 1.18)/(0.425) = 1.49$ messages

c. $\rho = \lambda T_s = 0.2 \times 2.3 = 0.46$

$T_r = 2.3 + (0.46 \times 2.3 \times 1.18)/(0.54) = 4.61$ ms

$r = 0.46 + (0.46 \times 0.46 \times 1.18)/(0.54) = 0.92$ messages

8.11 $\lambda = 0.05$ msg/sec; $T_s = (14{,}400 \times 8)/9600 = 12$ sec; $\rho = 0.6$

a. $T_w = \rho T_s/(1-\rho) = 18$ sec

b. $w = \rho^2/(1-p) = 0.9$

8.12 **a.** mean batch service time = $T_{sb} = MT_s$; batch variance = $\sigma_{T_{sb}}^2 = M\sigma_{T_s}^2$

To get the waiting time, T_{wb}, use the equation for T_w from Table 8.6a:

$$T_w = \frac{\rho T_s A}{1-\rho} = \frac{\lambda\left(T_s^2 + \sigma_{T_s}^2\right)}{2(1-\rho)}; \quad \text{then,}$$

$$T_{wb} = \frac{\frac{\lambda}{M}\left(M^2 T_s^2 + M\sigma_{T_s}^2\right)}{2(1-\rho)} = \frac{\lambda\left(MT_s^2 + \sigma_{T_s}^2\right)}{2(1-\rho)}, \quad \text{where } \rho = \frac{\lambda}{M}(MT_s) = \lambda T_s$$

b. $T_1 = \frac{1}{M} \times 0 + \frac{1}{M} \times T_s + \frac{1}{M} \times 2T_s + \ldots + \frac{1}{M} \times (M-1)T_s = \frac{M-1}{2} T_s$

$$T_w = T_{wb} + T_1 = \frac{\lambda\left(MT_s^2 + \sigma_{T_s}^2\right)}{2(1-\rho)} + \frac{M-1}{2} T_s$$

c. The equation for T_w in (b) reduces to the equation for T_w in (a) for M = 1. For batch sizes of M > 1, the waiting time is greater than the waiting time for Poisson input. This is due largely to the second term, caused by waiting within the batch.

8.13 **a.** $\rho = \lambda T_s = 0.2 \times 4 = 0.8$; $r = 2.4$; $\sigma_r = 2.4$

b. $T_r = 12$; $\sigma_{T_q} = 9.24$

8.14 $T_s = T_{s1} = T_{s2} = 1$ ms

a. If we assume that the two types are independent, then the combined arrival rate is Poisson.

$\lambda = 800$; $\rho = 0.8$; $T_r = 5$ ms

b. $\lambda_1 = 200$; $\lambda_2 = 600$; $\rho_1 = 0.2$ $\rho_2 = 0.6$; $T_{r1} = 2$ ms; $T_{r2} = 6$ ms

c. $\lambda_1 = 400$; $\lambda_2 = 400$; $\rho_1 = 0.4$ $\rho_2 = 0.4$; $T_{r1} = 2.33$ ms; $T_{r2} = 7.65$ ms

d. $\lambda_1 = 600$; $\lambda_2 = 200$; $\rho_1 = 0.6$ $\rho_2 = 0.2$; $T_{r1} = 3$ ms; $T_{r2} = 11$ ms

8.15 T_s = (100 octets × 8) / (9600 bps) = 0.0833 sec.

a. If the load is evenly distributed among the links, then the load for each link is 48 / 5 = 9.6 packets per second.

$\rho = 9.6 \times 0.0833 = 0.8$; $T_r = 0.0833 / (1 - 0.8) = 0.42$ sec

b. $\rho = \lambda T_s / N = (48 \times 0.833) / 5 = 0.8$

$$T_r = T_s + \frac{C \times T_s}{N \times (1-\rho)} = 0.13 \text{ sec}$$

8.16 **a.** The first character in a M-character packet must wait an average of M – 1 character interarrival times; the second character must wait M – 2 interarrival times, and so on to the last character, which does not wait at all. The expected interarrival time is $1/\lambda$. The average waiting time is therefore $W_i = (M-1)/2\lambda$.

b. The arrival rate is $K\lambda / M$, and the service time is (M + H) / C, giving a utilization of $\rho = K\lambda(M + H) / MC$. Using the M / D / 1 formula,

$$W_o = \frac{\rho T_s}{2-2\rho} = \frac{K\lambda(M+H)^2 \Big/ \left(C^2 M\right)}{2 - \left(2K\lambda(M+H)/CM\right)}$$

c.

$$T = W_i + W_o + T_s$$

$$= \frac{M-1}{2\lambda} + \frac{K\lambda(M+H)^2}{2MC^2 - 2K\lambda C(M+H)}$$

The plot is U-shaped.

8.17 a. System throughput: $\Lambda = \lambda + P\Lambda$; therefore $\Lambda = \lambda / (1 - P)$
Server utilization: $\rho = \Lambda T_s = \lambda T_s / (1 - P)$

$$T_q = \frac{T_s}{1-\rho} = \frac{(1-P)T_s}{1-P-\lambda T_s}$$

b. A single customer may cycle around multiple times before leaving the system. The mean number of passes J is given by:

$$J = 1(1-P) + 2(1-P)P + 3(1-P)P^2 + \ldots = (1-P)\sum_{i=1}^{\infty} iP^{i-1} = \frac{1}{1-P}$$

Because the number of passes is independent of the queuing time, the total time in the system is given by the product of the two means, JT_r.

CHAPTER 9
SELF-SIMILAR TRAFFIC

9.1 $L_0 = 1$; $L_1 = 2/3$; $L_2 = (2/3)(2/3) = (2/3)^2$; $L_N = (2/3)^N$

9.2 Base 3 uses the digits 0, 1, 2. A fraction in base 3 is represented by $X = 0.A_1A_2A_3\ldots$, where A_i is 0, 1, or 2. The value represented is:

$$X = \frac{A_1}{3} + \frac{A_2}{3^2} + \frac{A_3}{3^3} + \cdots$$

If we imagine the interval [0, 1] is divided into three equal pieces, then the first digit A_1 tells us whether X is in the left, middle, or right piece. For example, all numbers with $A_1 = 0$ are in the left piece (value less than 1/3). The second digit A_2 tells us whether X is in the left, middle, or right third of the given piece specified by A_1. In the Cantor set, we begin by deleting the middle third of [0, 1]; this removes all points, and only those points, whose first digit is 1. The points left over (the only ones with a remaining chance of being in the Cantor set) must have a 0 or 2 in the first digit. Similarly, points whose second digit is 1 are deleted at the next stage in the construction. By repeating this argument, we see that the Cantor set consists of all points whose base-3 expansion contains no 1's.

9.3 Multiplying each member of this sequence by a factor of 2 produces

$$\ldots 1/4, 1/2, 1, 2, 4, 8, 16 \ldots$$

which is the same sequence. The sequence is self-similar with a scaling factor of 2.

9.4 For the Pareto distribution $\Pr[X \le x] = F(x) = 1 - (k/x)^\alpha$

$\Pr[Y \le y] = \Pr[\ln(X/k) \le y] = \Pr[X \le ke^y] = 1 - (k/(ke^y))^\alpha = 1 - e^{-\alpha y}$

which is the exponential distribution.

9.5 **a.** H=1: $E[\mathbf{x}(at)] = aE[\mathbf{x}(t)]$; $Var[\mathbf{x}(at)] = a^2Var[\mathbf{x}(t)]$
H=0.5: $E[\mathbf{x}(at)] = a^{0.5}E[\mathbf{x}(t)]$; $Var[\mathbf{x}(at)] = aVar[\mathbf{x}(at)]$
For H = 1, the mean scales linearly with the scaling factor and the variance scales as the square of the scaling factor. Intuitively, one might argue that this is a "strong" manifestation of self-similarity.

b. In general, $E[a\mathbf{x}(t)] = aE[\mathbf{x}(t)]$; $Var[a\mathbf{x}(t)] = a^2Var[\mathbf{x}(t)]$. Therefore
$E[\mathbf{x}(at)] = a^{H-1}E[a\mathbf{x}(t)]$; $Var[\mathbf{x}(at)] = a^{2H-2}Var[a\mathbf{x}(t)]$

c. H=1: $E[\mathbf{x}(at)] = E[a\mathbf{x}(t)]$; $Var[\mathbf{x}(at)] = Var[a\mathbf{x}(t)]$
H=0.5: $E[\mathbf{x}(at)] = E[a\mathbf{x}(t)]/a^{0.5}$; $Var[\mathbf{x}(at)] = Var[\mathbf{x}(at)]/a$
Again, there is a strong manifestation of self-similarity for H = 1.

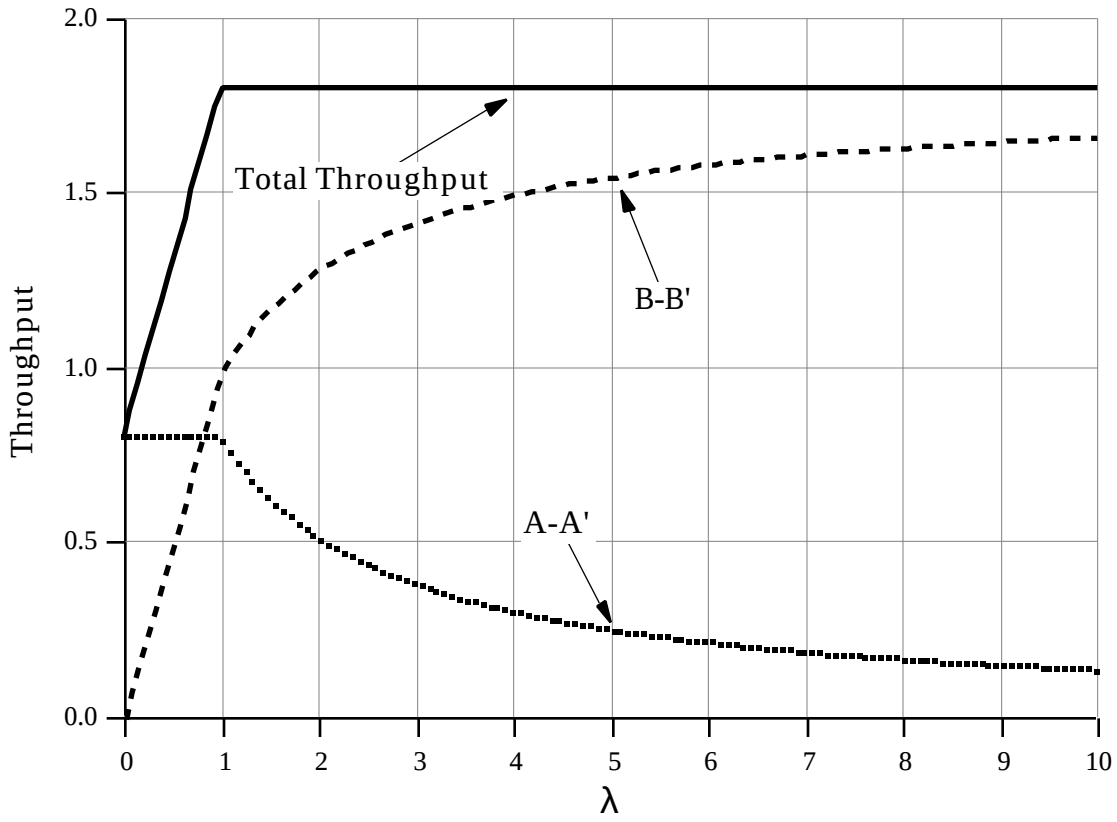
CHAPTER 10
CONGESTION CONTROL IN DATA NETWORKS AND INTERNETS

10.1 **1.** It does not guarantee that a particular node will not be swamped with frames.
2. There is no good way of distributing permits where they are most needed.
3. If a permit is accidentally destroyed, the capacity of the network is inadvertently reduced.

10.2 Throughput is as follows:

	$0 \le \lambda \le 1$	$1 \le \lambda \le 10$	$10 \le \lambda$
A-A' traffic	0.8	$(1.8 \times 0.8)/(0.8 + \lambda)$	$1.44/10.8 = 0.13$
B-B' traffic	λ	$1.8\lambda/(0.8 + \lambda)$	$18/10.8 = 1.67$
Total throughput	$0.8 + \lambda$	1.8	1.8

The plot:



For $1 \le \lambda \le 10$, the fraction of throughput that is A-A' traffic is $0.8/(0.8 + \lambda)$.

10.3 This problem is analyzed in [NAGL87]. The queue length for each outgoing link from a router will grow continually; eventually, the transit time through the queue exceeds the TTL of the incoming packets. A simplistic analysis would say that, under these circumstances, the incoming packets are discarded and not forwarded and no datagrams get through. But we have to take into account the fact that the

router should be able to discard a packet faster than it transmits it, and if the router is discarding a lot of packets, queue transit time declines. At this point, the argument gets a bit involved, but [NAGL87] shows that, at best, the router will transmit some datagrams with TTL = 1. These will have to be discarded by the next router that is visited.

10.4 $k = 2 + 2 \times \frac{T_{td} + R_u}{8 \times L_d} = 2 + 2a$

where the variable a is the same one defined in Chapter 11. In essence, the upper part of the fraction is the length of the link in bits, and the lower part of the fraction is the length of a frame in bits. So the fraction tells you how many frames can be laid out on the link at one time. Multiplying by 2 gives you the round-trip length of the link. You want your sliding window to accommodate that number of frames so that you can continue to send frames until an acknowledgment is received. Adding 1 to that total takes care of rounding up to the next whole number of frames. Adding 2 instead of 1 is just an additional margin of safety.

10.5 The average queue size over the previous cycle and the current cycle is calculated. This value is the threshold. By averaging over two cycles instead of just monitoring current queue length, the system avoids reacting to temporary surges that would not necessarily produce congestion. The average queue length may be computed by determining the area (product of queue size and time interval) over the two cycles and dividing by the time of the two cycles

CHAPTER 11
LINK-LEVEL FLOW AND ERROR CONTROL

11.1 **a.** Because only one frame can be sent at a time, and transmission must stop until an acknowledgment is received, there is little effect in increasing the size of the message if the frame size remains the same. All that this would affect is connect and disconnect time.

b. Increasing the number of frames would decrease frame size (number of bits / frame). This would lower line efficiency, because the propagation time is unchanged but more acknowledgments would be needed.

c. For a given message size, increasing the frame size decreases the number of frames. This is the reverse of (b).

11.2 Let L be the number of bits in a frame. Then, using Equation 11.4:

$$a = \frac{\text{Propagation Delay}}{\text{Transmission Time}} = \frac{20 \times 10^{-3}}{L/\left(4 \times 10^{3}\right)} = \frac{80}{L}$$

Using Equation 11.2:

$$S = \frac{1}{1 + 2a} = \frac{1}{1 + (160/L)} \geq 0.5$$

$$L \geq 160$$

Therefore, an efficiency of at least 50% requires a frame size of at least 160 bits.

11.3 $a = \dfrac{\text{Propagation Delay}}{L/R} = \dfrac{270 \times 10^{-3}}{10^{3}/10^{6}} = 270$

a. S = 1 / (1 + 2a) = 1 / 541 = 0.002

b. Using Equation 11.5: S = W / (1 + 2a) = 7 / 541 = 0.013

c. S = 127 / 541 = 0.23

d. S = 255 / 541 = 0.47

11.4 A → B: Propagation time = 4000 × 5 μsec = 20 msec

Transmission time per frame = $\dfrac{1000}{100 \times 10^{3}} = 10$ msec

B → C: Propagation time = 1000 × 5 μsec = 5 msec

Transmission time per frame = x = 1000 / R

R = data rate between B and C (unknown)

A can transmit three frames to B and then must wait for the acknowledgment of the first frame before transmitting additional frames. The first frame takes 10 msec to transmit; the last bit of the first frame arrives at B 20 msec after it was transmitted, and therefore 30 msec after the frame transmission began. It will take

an additional 20 msec for B's acknowledgment to return to A. Thus, A can transmit 3 frames in 50 msec.

B can transmit one frame to C at a time. It takes 5 + x msec for the frame to be received at C and an additional 5 msec for C's acknowledgment to return to A. Thus, B can transmit one frame every 10 + x msec, or 3 frames every 30 + 3x msec. Thus:

30 + 3x = 50

x = 6.66 msec

R = 1000 / x = 150 kbps

11.5 Round trip propagation delay of the link = 2 × L × t

Time to transmit a frame = B / R

To reach 100% utilization, the transmitter should be able to transmit frames continuously during a round trip propagation time. Thus, the total number of frames transmitted without an ACK is:

$$N = \left\lceil \frac{2 \times L \times t}{B/R} + 1 \right\rceil$$ where $\lceil X \rceil$ is the smallest integer greater than or equal to X

This number can be accommodated by an M-bit sequence number with:

$$M = \lceil \log_2 (N) \rceil$$

11.6 In fact, REJ is not needed at all, since the sender will time out if it fails to receive an ACK. The REJ improves efficiency by informing the sender of a bad frame as early as possible.

11.7 Assume a 2-bit sequence number:

1. Station *A* sends frames 0, 1, 2 to station B.
2. Station *B* receives all three frames and cumulatively acknowledges with RR 3.
3. Because of a noise burst, the RR 3 is lost.
4. *A* times out and retransmits frame 0.
5. *B* has already advanced its receive window to accept frames 3, 0, 1, 2. Thus it assumes that frame 3 has been lost and that this is a new frame 0, which it accepts.

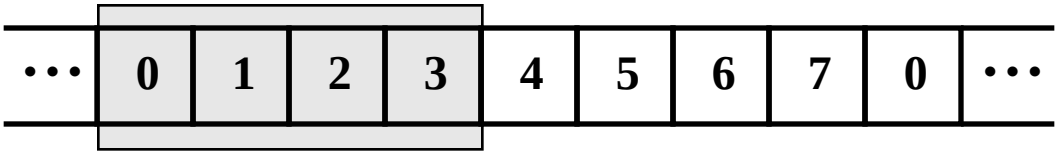
11.8 Use the following formulas:

a	0.1	1.	10	100
S&W	(1 – P) / 1.2	(1 – P) / 3	(1 – P) / 21	(1 – P) / 201
GBN (7)	(1–P) / (1+0.2P)	(1–P) / (1+2P)	7(1–P) / 21(1+6P)	7(1 – P) / 201(1+6P)
GBN (127)	(1–P) / (1+0.2P)	(1–P) / (1+2P)	(1 – P) / (1+20P)	127(1–P) / 201(1+126P)
SREJ (7)	1 – P	1 – P	7(1 – P) / 21	7(1 – P) / 201
SREJ (127)	1 – P	1 – P	1 – P	127(1 – P) / 201

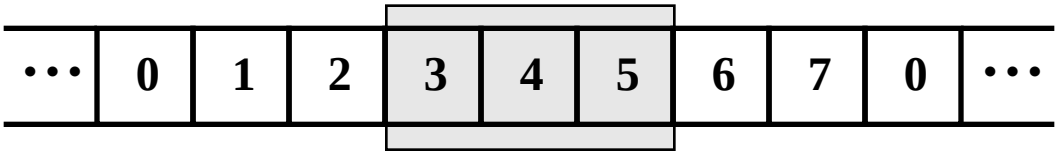
For a given value of a, the utilization values change very little as a function of P over a reasonable range (say 10^{-3} to 10^{-12}). We have the following approximate values for $P = 10^{-6}$:

a	0.1	1.0	10	100
Stop-and-wait	0.83	0.33	0.05	0.005
GBN (7)	1.0	1.0	0.33	0.035
GBN (127)	1.0	1.0	1.0	0.63
SREJ (7)	1.0	1.0	0.33	0.035
SREJ (127)	1.0	1.0	1.0	0.63

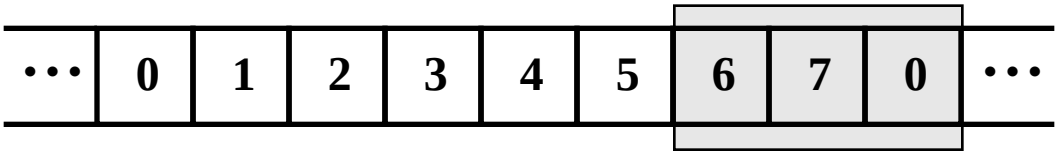
11.9a.



b.



c.



11.10 A lost SREJ frame can cause problems. The sender never knows that the frame was not received, unless the receiver times out and retransmits the SREJ.

11.11 From the standard: "A SREJ frame shall not be transmitted if an earlier REJ exception condition has not been cleared (To do so would request retransmission of a data frame that would be retransmitted by the REJ operation.)" In other words, since the REJ requires the station receiving the REJ to retransmit the rejected frame and all subsequent frames, it is redundant to perform a SREJ on a frame that is already scheduled for retransmission.

Also from the standard: "Likewise, a REJ frame shall not be transmitted if one or more earlier SREJ exception conditions have not been cleared." The REJ frame indicates the acceptance of all frames prior to the frame rejected by the REJ frame. This would contradict the intent of the SREJ frame or frames.

11.12 Let t_1 = time to transmit a single frame

$$t_1 = \frac{1024 \text{ bits}}{10^6 \text{ bps}} = 1.024 \text{ msec}$$

The transmitting station can send 7 frames without an acknowledgment. From the beginning of the transmission of the first frame, the time to receive the acknowledgment of that frame is:

$$t_2 = 270 + t_1 + 270 = 541.024 \text{ msec}$$

During the time t_2, 7 frames are sent.

$$\text{Data per frame} = 1024 - 48 = 976$$

$$\text{Throughput} = \frac{7 \times 976 \text{ bits}}{541.024 \times 10^{-3} \text{ sec}} = 12.6 \text{ kbps}$$

11.13 The selective-reject approach would burden the server with the task of managing and maintaining large amounts of information about what has and has not been successfully transmitted to the clients; the go-back-N approach would be less of a burden on the server.

CHAPTER 12
TCP TRAFFIC CONTROL

12.1 The number of unacknowledged segments in the "pipeline" at any time is 5. Thus, once steady state is reached, the maximum achievable throughput is equal to the normalized theoretical maximum of 1.

12.2 This will depend on whether multiplexing or splitting occurs. If there is a one-to-one relationship between network connections and transport connections, then it will do no good to grant credit at the transport level in excess of the window size at the network level. If one transport connection is split among multiple network connections (each one dedicated to that single transport connection), then a practical upper bound on the transport credit is the sum of the network window sizes. If multiple transport connections are multiplexed on a single network connection, their aggregate credit should not exceed the network window size. Furthermore, the relative amount of credit will result in a form of priority mechanism.

12.3 In TCP, no provision is made. A later segment can provide a new credit allocation. Provision is made for misordered and lost credit allocations in the ISO transport protocol (TP) standard. In ISO TP, ACK/Credit messages (AK) are in separate PDUs, not part of a data PDU. Each AK TPDU contains a YR-TU-NR field, which is the sequence number of the next expected data TPDU, a CDT field, which grants credit, and a "subsequence number", which is used to assure that the credit grants are processed in the correct sequence. Further, each AK contains a "flow control confirmation" value which echoes the parameter values in the last AK received (YR-TU-NR, CDT, subsequence number). This can be used to deal with lost AKs.

12.4 The upper limit ensures that the maximum difference between sender and receiver can be no greater than 2^{31}. Without such a limit, TCP might not be able to tell when the 32-bit sequence number had rolled over from $2^{31} - 1$ back to 0.

12.5 **a.** $SRTT(n) = \alpha \times SRTT(0) + (1 - \alpha) \times RTT \times (\alpha^{n-1} + \alpha^{n-2} + \ldots \alpha + 1)$

$= \alpha \times SRTT(0) + (1 - \alpha) \times RTT \times (1 - \alpha^{n}) / (1- \alpha)$

$SRTT(19) = 1.1$ sec

b. $SRTT(19) = 2.9$ sec; in both cases, the convergence speed is slow, because in both cases, the initial SRTT(0) is improperly chosen.

12.6 When the 50-octet segment arrives at the recipient, it returns a credit of 1000 octets. However, the sender will now compute that there are 950 octets in transit in the network, so that the usable window is now only 50 octets. Thus, the sender will once again send a 50-octet segment, even though there is no longer a natural boundary to force it.

In general, whenever the acknowledgment of a small segment comes back, the usable window associated with that acknowledgment will cause another segment of the same small size to be sent, until some abnormality breaks the pattern. Once the condition occurs, there is no natural way for those credit allocations to be recombined; thus the breaking up of the usable window into small pieces will persist.

12.7 **a.** As segments arrive at the receiver, the amount of available buffer space contracts. As data from the buffer is consumed (passed on to an application), the amount of available buffer space expands. If SWS is not taken into account, the following procedure is followed: When a segment is received, the recipient should respond with an acknowledgment that provides credit equal to the available buffer space. The SWS avoidance algorithm introduces the following rule: When a segment is received, the recipient should not provide additional credit unless the following condition is met:

$$\text{available buffer space} \geq \text{MIN}\left(\frac{\text{buffersize}}{2}, \text{maximum segment size}\right)$$

The second term is easily explained: if the available buffer space is greater than the largest possible segment, then clearly SWS cannot occur. The first term is a reasonable guideline that states that if at least half of the buffer is free, the sender should be provided the available of credit.

b. The suggested strategy is referred to as the Nagle algorithm and can be stated as follows: If there is unacknowledged data, then the sender buffers all data until the outstanding data have been acknowledged or until a maximum-sized segment can be sent. Thus, the sender accumulates data locally to avoid SWS.

12.8 **a.** $E[X] = \mu_X = 1/4 = 0.25$

$Var[X] = E[X^2] - E^2[X] = (1/4) - (1/16) = 0.1875$

$\sigma_X = (0.1875)^{0.5} = 0.433$

$MDEV[X] = E[\,|X - \mu_X|\,] = (1/4)[(3/4) + (1/4) + (1/4) + (1/4)] = 0.612$

In this case, $\sigma_X < MDEV[X]$

b. $E[Y] = \mu_Y = 0.7$

$Var[Y] = E[Y^2] - E^2[Y] = 0.7 - 0.49 = 0.21$

$\sigma_X = (0.21)^{0.5} = 0.458$

$MDEV[X] = E[\,|Y - \mu_Y|\,] = (0.3)(0.7) + (0.7)(0.3) = 0.422$

In this case, $\sigma_X > MDEV[X]$

12.9 $SRTT(K + 1) = (1 - g)SRTT(K) + gRTT(K + 1)$

$SERR(K + 1) = RTT(K + 1) - SRTT(K)$

Substituting for SRTT(K) in the first equation from the second equation:

$SRTT(K + 1) = RTT(K + 1) - (1 - g)SERR(K + 1)$

Think of RTT(K + 1) as a prediction of the next measurement and SERR(K + 1) as the error in the last prediction. The above expression says we make a new prediction based on the old prediction plus some fraction of the prediction error.

12.10 TCP initializes the congestion window to 1, sends an initial segment, and waits. When the ACK arrives, it increases the congestion window to 2, sends 2 segments, and waits. When the 2 ACKS arrives, they each increase the congestion window by one, so that it can send 4 segments. In general, it takes $\log_2 N$ round trips before TCP can send N segments.

12.11 **a.** $W = (10^9 \times 0.06)/(576 \times 8) \approx 13{,}000$ segments

If the window size grows linearly from 1, it will take about 13,000 round trips, or about 13 minutes to get the correct window size.

b. $W = (10^9 \times 0.06) / (16{,}000 \times 8) \approx 460$ segments
In this case, it takes about 460 round trips, which is less than 30 seconds.

12.12 Recall from our discussion in Section 12.1 that a receiver may adopt a conservative flow control strategy by issuing credit only up to the limit of currently available buffer space, or an optimistic strategy by issuing credit for space that is currently unavailable but which the receiver anticipates will soon become available. In the latter case, buffer overflow is possible.

12.13 AAL5 accepts a stream of cells beginning with the first SDU=0 after an SDU=1 and continuing until a cell with SDU=1 is received, and assembles all of these cells, including the final SDU=1 cell, as a single segment. With PPD, some initial portion of an SDU=0 sequence may get through with the remainder of the SDU=0 sequence discarded. If the final SDU=1 cell is also discarded, then the receiving AAL5 will combine the first part of the partially discarded segment with the stream of cells that make up the next segment.

CHAPTER 13
TRAFFIC AND CONGESTION CONTROL IN ATM NETWORKS

13.1 Yes, but ATM does not include such sliding-window mechanisms. In some cases, a higher-level protocol above ATM will provide such mechanisms, but not in all cases.

13.2 a. We can demonstrate this by induction. We want to show that the maximum number of conforming back-to-back cells, N, satisfies:

$$N = \left\lfloor 1 + \frac{\tau}{T - \delta} \right\rfloor = 1 + \left\lfloor \frac{\tau}{T - \delta} \right\rfloor$$

First, suppose $\tau < T - \delta$. Then N = 1 and back-to-back cells are not allowed. To see this, suppose that the first cell arrives at $t_a(1) = 0$. Since the cell insertion time is δ, a second cell back to back would arrive at $t_a(2) = \delta$. But using the virtual scheduling algorithm, we see that the second cell must arrive no earlier than $T - \tau$. Therefore, we must have:

$t_a(2) > T - \tau$
$\delta > T - \tau$
$\tau > T - \delta$

which is not true. Therefore, N = 1.

Now suppose $T - \delta < \tau < 2(T - \delta)$. Then N = 2 and two consecutive cells are allowed, but not three or more. We can see that two consecutive cells are allowed since $T - \delta < \tau$ satisfies the required condition just derived, above. To see that three consecutive cells are not allowed, let us assume that three consecutive cells do arrive. Then, we have $t_a(1) = 0$, $t_a(2) = \delta$, and $t_a(3) = 2\delta$. Looking at the virtual scheduling algorithm (Figure 13.6a), we see that the theoretical arrival time for the third cell is TAT = 2T. Therefore the third cell must arrive no earlier than $2T - \tau$, and we must have:

$t_a(3) > 2T - \tau$
$2\delta > 2T - \tau$
$\tau > 2(T - \delta)$

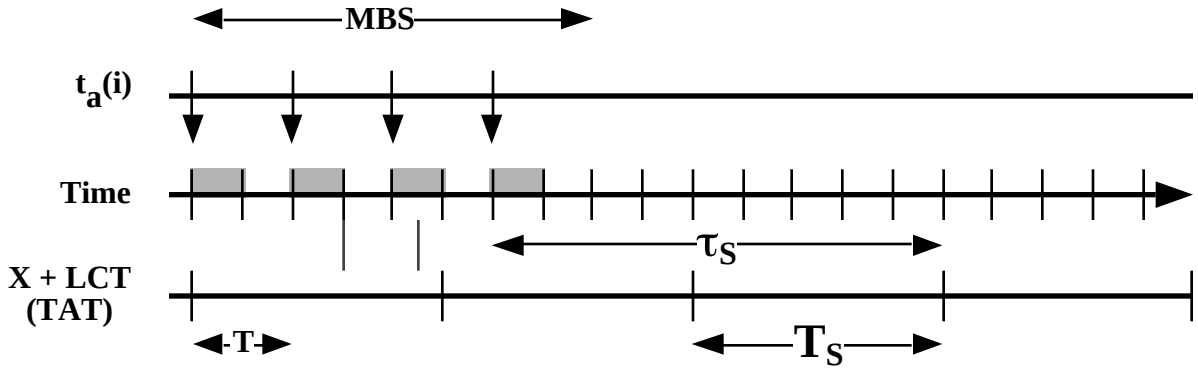
which is not true. Therefore N = 2.

This line of reasoning can be extended to longer strings of back-to-back cells, and by induction, Equation (13.1) is correct.

b. We want to show that the maximum number of cells, MBS, that may be transmitted at the peak cell rate satisfies:

$$MBS = \left\lfloor 1 + \frac{\tau_S}{T_S - T} \right\rfloor = 1 + \left\lfloor \frac{\tau_S}{T_S - T} \right\rfloor$$

The line of reasoning is the same as for part (a) of this problem. In this case, we assume that cells can arrive at a maximum of an interarrival time of T. Therefore, an MBS of 2 means that two cells can arrive at a spacing of T; an MBS of 3 means that 3 cells can arrive with successive spacings of T, and so on. Here is an example that shows the relationship among the relevant parameters. In this example, MBS = 4, $T_S = 5\delta$, $T = 2\delta$, and $\tau_S = 9\delta$.



c. Suppose that $\tau_S = (MBS - 1)(T_S - T)$ exactly. Then,

$$1 + \left\lfloor \frac{\tau_S}{T_S - T} \right\rfloor = 1 + \left\lfloor \frac{(MBS - 1)(T_S - T)}{T_S - T} \right\rfloor = 1 + (MBS - 1) = MBS$$

which satisfies Equation (13.2). Now suppose that we have the following:

$$(MBS - 1)(T_S - T) < \tau_S < MBS(T_S - T)$$

then the value $\frac{\tau_S}{T_S - T}$ is a number greater than the integer (MBS – 1) and less than the integer MBS. Therefore:

$$1 + \left\lfloor \frac{\tau_S}{T_S - T} \right\rfloor = 1 + (MBS - 1) = MBS$$

which still satisfies Equation (13.2). However if $\tau_S \geq MBS(T_S - T)$, then the equation is not satisfied.

13.3 Observe that if $t \geq (MBS \times T)$, then the first term of the inequality applies; otherwise, the second term applies.

13.4 We have ER = max[Fairshare, VCshare], where

Fairshare = Target_rate / #_connections

VCshare = CCR / LF = (CCR × Target_rate) / Input_rate

By the first equation, we know that ER ≥ Fairshare and ER ≥ VCshare

Case I: LF > 1

In this case VCshare > CCR. Therefore, ER > CCR

Case II: LF > 1

IIa Fairshare > VCshare

This condition holds if: Input_rate > CCR × #_connections
With this condition ER = Fairshare
Then ER > CCR if (Target_rate / #_connections) > CCR

IIb VCshare > Fairshare

This condition holds if: Input_rate < CCR × #_connections
With this condition ER = VCshare
Then ER > CCR if Target_rate > IR

Case III: LF = 1

In this case, VCshare = CCR

IIa Fairshare > VCshare

This condition holds if: Input_rate > CCR × #_connections
With this condition ER = Fairshare
Then ER > CCR if (Target_rate / #_connections) > CCR

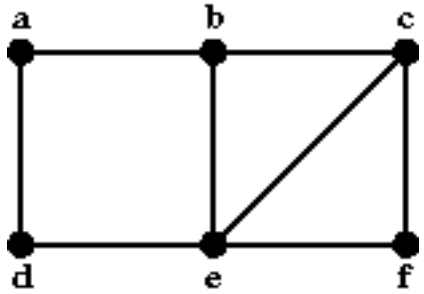
IIb VCshare > Fairshare

This condition holds if: Input_rate < CCR × #_connections
With this condition ER = VCshare = CCR

CHAPTER 14
OVERVIEW OF GRAPH THEORY AND LEAST-COST PATHS

14.1 $n(n-1)/2$

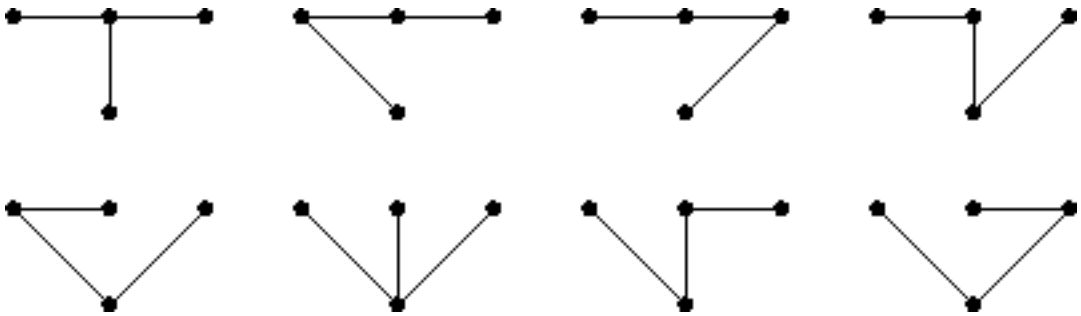
14.2 **a.**



b. (a, b, c, f); (a, b, c, e, f); (a, b, e, f); (a, b, e, c, f); (a, d, e, f); (a, d, e, b, c, f); (a, d, e, c, f);

c. 3

14.3



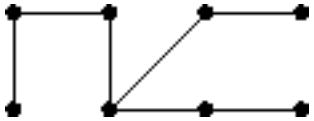
14.4 If G is a connected graph, then the removal of that edge makes the graph disconnected.

14.5 First, show that the graph T constructed is a tree. Then use induction on the level of T to show that T contains all vertices of G.

14.6 The input is a graph with vertices ordered $V_1, V_2, \ldots, V_N$.

1. [**Initialization**]. Set S to $\{V_1\}$ and set T to the graph consisting of V_1 and no edges. Designate V_1 as the root of the tree.
2. [**Add edges**]. Process the vertices in S in order. For each vertex x in S, process each adjacent vertex y in G in order; add edge (x, y) and vertex y to T provided that this does not produce a cycle in T. If no edges are added in this step, go to step 4.
3. [**Update S**]. Replace the contents of S with the children in T of S ordered consistent with the original ordering. Go to step 2.
4. [**Return Result**]. If the number of vertices in T is N, return CONNECTED; otherwise, return NOT_CONNECTED. Halt.

14.7 Start with V1
Iteration 1: Add V_5, V_8
Iteration 2: Add V_2, V_6, V_4
Iteration 3: Add V_3, V_7
Spanning Tree:



14.8
```
for n := 1 to N do
    begin
        d[n] := ∞;
        p[n] := –1
    end;
d[srce] := 0                {initialize Q to contain srce only 0}
insert srce at the head of Q;
{initialization over }

while Q is not empty do
    begin
        delete the head node j from Q;
        for each link jk that starts at j do
            begin
                newdist := d[j] + c[j,k];
                if newdist < d[k] then
                  begin
                      d[k] := newdist;
                      p[nk] := j
                      if k   Q then insert K at the tail of Q;
                  end
            end;
    end;
```

14.9 This proof is based on one in [BERT92]. Let us claim that
(1) L(i) ≤ L(j) for all i T and j T
(2) For each node j, L(j) is the shortest distance from s to j using paths with all nodes in T except possibly j.

Condition (1) is satisfied initially, and because w(i, j) ≥ 0 and L(i) = $\min_{j \ T}$ L(j), it is preserved by the formula in step 3 of the algorithm. Condition (2) then can be shown by induction. It holds initially. Suppose that condition (2) holds at the beginning of some iteration. Let i be the node added to T at that iteration, and let L(k) be the label of each node k at the beginning of the iteration. Condition (2) holds for i = j by the induction hypothesis, and it holds for all j T by condition (1) and the induction hypothesis. Finally for a node j T i, consider a path from s to j which is shortest among all those in which all nodes of the path belong to T i and let L'(j) be the distance. Let k be the last node of this path before node j. Since k is in T i, the length of this path from s to k is L(k). So we have

$$L'(j) = \min_{k \ T \ i}[w(k, j) + L(k)] = \min[\min_{k \ T}[w(k, j) + L(k)], w(i, j) + L(i)]$$

The induction hypothesis implies that $L(j) = \min_{k \in T}[w(k, j) + L(k)]$, so we have

$$L'(j) = \min[L(j), w(i, j) + L(i)]$$

Thus in step 3, L(j) is set to the shortest distance from s to j using paths with all nodes except j belonging to T $\cup$ i.

14.10 Consider the node i which has path length K+1, with the immediately preceding node on the path being j. The distance to node i is w(j, i) plus the distance to reach node j. This latter distance must be L(j), the distance to node j along the optimal route, because otherwise there would be a route with shorter distance found by going to j along the optimal route and then directly to i.

14.11 Not possible. A node will not be added to T until its least-cost route is found. As long as the least-cost route has not been found, the last node on that route will be eligible for entry into T before the node in question.

14.12 We show the results for starting from node 2.

	M	L(1)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{2}	3	2-1	3	2-3	2	2-4	∞	—	∞	—
2	{2, 4}	3	2-1	3	2-3	2	2-4	3	2-4-5	∞	—
3	{2, 4, 1}	3	2-1	3	2-3	2	2-4	3	2-4-5	∞	—
4	{2, 4, 1, 3}	3	2-1	3	2-3	2	2-4	3	2-4-5	8	2-3-6
5	{2, 4, 1, 3, 5}	3	2-1	3	2-3	2	2-4	3	2-4-5	5	2-4-5-6
6	{2, 4, 1, 3, 5, 6}	3	2-1	3	2-3	2	2-4	3	2-4-5	5	2-4-5-6

14.13 We show the results for starting from node 2.

h	$L_h(1)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	—	∞	—	∞	—	∞	—	∞	—
1	3	2-1	3	2-3	2	2-4	∞	—	∞	—
2	3	2-1	3	2-3	2	2-4	3	2-4-5	8	2-3-6
3	3	2-1	3	2-3	2	2-4	3	2-4-5	5	2-4-5-6
4	3	2-1	3	2-3	2	2-4	3	2-4-5	5	2-4-5-6

14.14 **a.** We provide a table for node 1 of network a; the figure is easily generated.

	M	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{1}	1	1-2	∞	—	4	1-4	∞	—	∞	—
2	{1,2}	1	1-2	4	1-2-3	4	1-4	2	1-2-5	∞	—
3	{1,2,5}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	6	1-2-5-6
4	{1,2,5,3}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6
5	{1,2,5,3,4}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6
6	{1,2,5,3,4,6}	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6

b. The table for network b is similar in construction but much larger. Here are the results for node A:

A to B: A-B
A to C: A-B-C
A to D: A-E-G-H-D
A to E: A-E
A to F: A-B-C-F
A to G: A-E-G
A to H: A-E-G-H
A to J: A-B-C-J
A to K: A--E-G-H-D-K

14.15

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	—	∞	—	∞	—	∞	—	∞	—
1	1	1-2	∞	—	4	1-4	∞	—	∞	—
2	1	1-2	4	1-2-3	4	1-4	2	1-2-5	∞	—
3	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	6	1-2-3-6
4	1	1-2	3	1-2-5-3	3	1-2-5-4	2	1-2-5	5	1-2-5-3-6

14.16 If there is a unique least-cost path, the two algorithms will yield the same result because they are both guaranteed to find the least-cost path. If there are two or more equal least-cost paths, the two algorithms may find different least-cost paths, depending on the order in which alternatives are explored.

14.17 This explanation is taken from [BERT92]. The Floyd-Warshall algorithm iterates on the set of nodes that are allowed as intermediate nodes on the paths. It starts like both Dijkstra's algorithm and the Bellman-Ford algorithm with single arc distances (i.e., no intermediate nodes) as starting estimates of shortest path lengths. It then calculates shortest paths under the constraint that only node 1 can be used as an intermediate node, and then with the constraint that only nodes 1 and 2 can be used, and so forth.

For n = 0, the initialization clearly gives the shortest path lengths subject to the constraint of no intermediate nodes on paths. Now, suppose for a given n, $L_n(i, j)$ in the above algorithm gives the shortest path lengths using nodes 1 to n as intermediate nodes. Then the shortest path length from i to j, allowing nodes 1 to n+1 as possible intermediate nodes, either contains node n+1 on the shortest path or doesn't contain node n+1. For the first case, the constrained shortest path from i to j goes from i to n+1 and then from n+1 to j, giving the length in the final term of the equation in step 2 of the problem. For the second case, the constrained shortest path is the same as the one using nodes 1 to n as possible

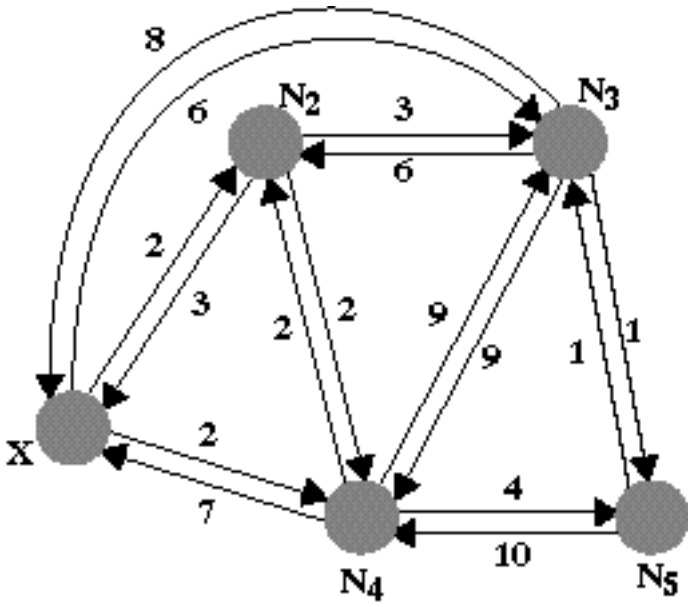
intermediate nodes, yielding the length of the first term in the equation in step 2 of the problem.

CHAPTER 15
INTERIOR ROUTING PROTOCOLS

15.1 No. That would violate the "separation of layers" concept. There is no need for IP to know how data is routed within a network. For the next hop, IP specifies another system on the same network and hands that address down to the network-layer protocol which then initiates the network-layer routing function.

15.2 When the unicast routing adapts to the shorter path, the delay-bandwidth between the sender and receiver decreases [than the case of the longer path], since shorter paths experience less delays. The new path does not have the capacity to contain all the packets with the window size filling the longer path, and so some packets are dropped and TCP cuts back its window size.

15.3 A digraph could be drawn in a number of ways, depending on what are defined to be nodes and what are defined to be edges. As Figure 14.10 illustrates, each node and each router can be depicted as a node. The following digraph is a simpler representation taken from the point of view of Host X:



15.4

B	C	A
3: N1	8: N1	6: N4, D, N2, B, N1
1: N2	8: N3, E, N4, D, N2	3: N4, D, N2
4: N2, G, N3	5: N3	2: N4, E, N3
3: N2, D, N4	6: N3, E, N4	1: N4
4: N2, D, N4, F, N5	6: N3, H, N5	2: N4, F, N5

15.5

Destination Network	Next Router	Metric L(A, j)
1	D	6
2	D	3

Destination Network	Next Router	Metric L(A, j)
1	D	6
2	D	3

3	D	6
4	—	1
5	F	5

A's routing table prior to update

3	E	2
4	—	1
5	F	2

A's routing table after update

15.6 First, a mechanism is needed for getting all nodes to agree to start the algorithm. Second, a mechanism is needed to abort the algorithm and start a new version if a link status or value changes as the algorithm is running.

15.7 **a.**

Router	Distance L(x, 5)	Next Hop R(x, 5)
A	3	B
B	2	D
C	3	B
D	1	—

b.

R	L(x, 5)	R(x, 5)	L(x, 5)	R(x, 5)	L(x, 5)	R(x, 5)	• • •	L(x, 5)	R(x, 5)	L(x, 5)	R(x, 5)
A	3	B	4	C	5	C	• • •	11	C	12	C
B	U	U	4	C	5	C	• • •	11	C	12	C
C	3	B	4	A	5	A	• • •	11	A	11	D
D	1	—	1	—	1	—	• • •	1	—	1	—
	Iteration 1		Iteration 2		Iteration 3		• • •	Iteration 9		Iteration 10	

U = Unreachable

15.8 Suppose A has a route to D through C, so A sends a poisoned reverse message to C that indicates that D is unreachable via A (set metric to infinity). B will receive this message and think that D is unreachable via A. But B can get to C itself and does not need to go through A to do so. If C is the best first hop for A to get to D, then C is also the best first hop for B to get to D across this network.

15.9 RIP runs on top of UDP, which provides an optional checksum for the data portion of the UDP datagram. OSPF runs on top of IP. The IP checksum only covers the IP header, so OSPF must add its own checksum. [STEV94]

15.10 Load balancing increases the chances of packets being delivered out of order, and possibly distorts the round-trip times calculated by TCP.

CHAPTER 16
EXTERIOR ROUTING PROTOCOLS AND MULTICAST

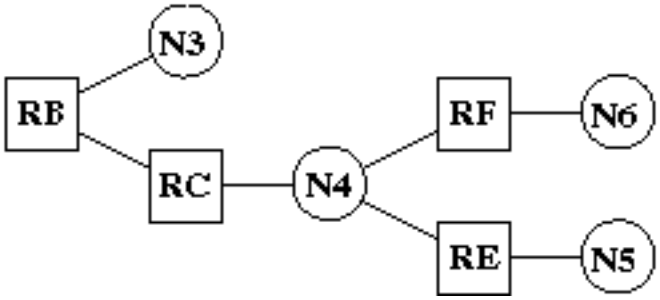
16.1 So that the queries are not propagated outside of the local network.

16.2 For convenience, flip the table on its side:

N1	N2	N3	N4	N5	N6	L1	L2	L3	L4	L5	Total
1	1	1	2	1	1	1	2	2	1	2	15

This is the least efficient method, but it is very robust: a packet will get through if there is at least one path from source to destination. Also, no prior routing information needs to be exchanged.

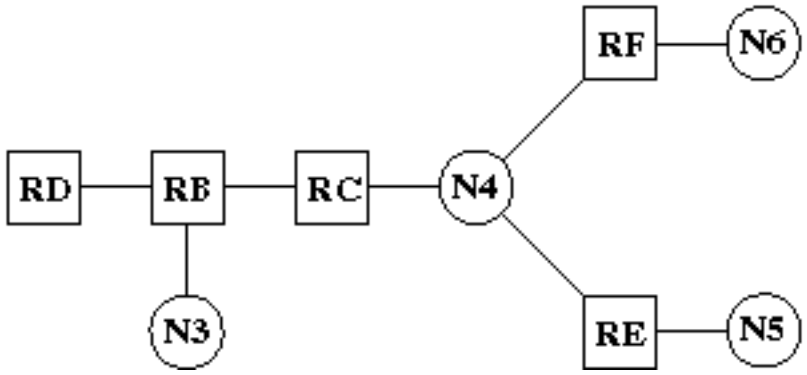
16.3 Root to the left:



16.4 a. In this table, we show the hop cost for each network or link:

N1	N3	N4	N5	N6	L3	L4	Total
0	1	12	1	1	3	4	22

b.



N1	N3	N4	N5	N6	L3	L5	Total
0	1	12	1	1	3	2	20

16.5 MOSPF works well in an environment where group members are relatively densely packed. In such an environment, bandwidth is likely to be plentiful, with most group members on shared LANs with high-speed links between the LANs. However, MOSPF does not scale well to large, sparsely-packed multicast groups.

Routing information is periodically flooded to all other routers in an area. This can consume considerable resources if the routers are widely dispersed; also the routers must maintain a lot of state information about group membership and location. The use of areas can cut down on this problem but the fundamental scaling issue remains.

16.6 PIM allows the a destination router to replace the group-shared tree with a shortest-path tree to any source. Thus, this is the shortest unicast path from destination to source, which is not necessarily the reverse of the shortest unicast path from source to destination.

16.7 First, suppose that all traffic must go through the RP-based tree. If the RP is placed badly with respect to the topology and distribution of the group receivers and senders, then the delay is likely to be large. However, for applications for which delay is important and the data rate is high enough, receivers' last hop routers may join directly to the source and receive packets over the shortest path. So, the RP placement is this case is not so crucial for good performance.

CHAPTER 17
INTEGRATED AND DIFFERENTIATED SERVICES

17.1 These answers are taken from RFC 1809, *Using the Flow Label Field in IPv6* (June 1995).

a. The IPv6 specification allows routers to ignore Flow Labels and also allows for the possibility that IPv6 datagrams may carry flow setup information in their options. Unknown Flow Labels may also occur if a router crashes and loses its state. During a recovery period, the router will receive datagrams with Flow Labels it does not know, but this is arguably not an error, but rather a part of the recovery period. Finally, if the controversial suggestion that each TCP connection be assigned a separate Flow Label is adopted, it may be necessary to manage Flow Labels using an LRU cache (to avoid Flow Label cache overflow in routers), in which case an active but infrequently used flow's state may have been intentionally discarded. In any case, it is clear that treating this situation as an error and, say dropping the datagram and sending an ICMP message, is inappropriate. Indeed, it seems likely that in most cases, simply forwarding the datagram as one would a datagram with a zero Flow Label would give better service to the flow than dropping the datagram.

b. An example is a router which has two paths to the datagram's destination, one via a high-bandwidth satellite link and the other via a low-bandwidth terrestrial link. A high bandwidth flow obviously should be routed via the high-bandwidth link, but if the router loses the flow state, the router may route the traffic via the low-bandwidth link, with the potential for the flow's traffic to swamp the low-bandwidth link. It seems likely, however, these situations will be exceptions rather than the rule.

17.2 These answers are taken from RFC 1809.

a. An internet may have partitioned since the flow was created. Or the deletion message may be lost before reaching all routers. Furthermore, the source may crash before it can send out a Flow Label deletion message.

b. The obvious mechanism is to use a timer. Routers should discard Flow Labels whose state has not been refreshed within some period of time. At the same time, a source that crashes must observe a quiet time, during which it creates no flows, until it knows that all Flow Labels from its previous life must have expired. (Sources can avoid quiet time restrictions by keeping information about active Flow Labels in stable storage that survives crashes). This is precisely how TCP initial sequence numbers are managed and it seems the same mechanism should work well for Flow Labels.

17.3 Again, RFC 1809:

The argument in favor of using Flow Labels on individual TCP connections is that even if the source does not request special service, a network provider's routers may be able to recognize a large amount of traffic and use the Flow Label field to establish a special route that gives the TCP connection better service (e.g., lower delay or bigger bandwidth). Another argument is to assist in efficient demux at the receiver (i.e., IP and TCP demuxing could be done once).

An argument against using Flow Labels in individual TCP connections is that it changes how we handle route caches in routers. Currently one can cache a route for a destination host, regardless of how many different sources are sending

to that destination host. Thus, if five sources each have two TCP connections sending data to a server, one cache entry containing the route to the server handles all ten TCPs' traffic. Putting Flow Labels in each datagram changes the cache into a Flow Label cache, in which there is a cache entry for every TCP connection. So there's a potential for cache explosion. There are ways to alleviate this problem, such as managing the Flow Label cache as an LRU cache, in which infrequently used Flow Labels get discarded (and then recovered later). It is not clear, however, whether this will cause cache thrashing.

Observe that there is no easy compromise between these positions. One cannot, for instance, let the application decide whether to use a Flow Label. Those who want different Flow Labels for every TCP connection assume that they may optimize a route without the application's knowledge. Forcing all applications to use Flow Labels will force routing vendors to deal with the cache explosion issue, even if we later discover that we don't want to optimize individual TCP connections.

17.4 From RFC 1809:
During its discussions, the End-to-End group realized this meant that if a router forwarded a datagram with an unknown Flow Label, it had to ignore the Priority field, because the priority values might have been redefined. (For instance, the priorities might have been inverted). The IPv6 community concluded this behavior was undesirable. Indeed, it seems likely that when the Flow Label is unknown, the router will be able to give much better service if it uses the Priority field to make a more informed routing decision.

17.5 **a.** $\lambda = \lambda 1 + \lambda 2 = 0.5$; Using the M / M / 1 equations in Table 8.6:
$T_r = T_s / (1 - \lambda T_s) = 1 / (1 - 0.5) = 2$; Then
$V = (4 - 2\times 2) + (4 - 2) = 2$

b. Using the M / M / 1 equations from Table 8.9:
$\rho_1 = \lambda_1 T_{s1} = 0.25 = \rho_2$; $\rho = \rho_1 + \rho_2 = 0.5$
$T_{r1} = T_{s1} + (\rho_1 T_{s1} + \rho_2 T_{s2}) / (1 - \rho_1) = 1.67$
$T_{r2} = T_{s2} + (T_{q1} - T_{s1}) / (1 - \rho) = 2.33$
$V = (4 - 2\times 1.67) + (4 - 2.33) = 2.33$

Therefore, the strict priority service is more efficient in the sense that it delivers a higher utility for the same throughput.

17.6 **a.** During a bust of S seconds, a total of MS octets are transmitted. A burst empties the bucket (b octets) and, during the burst, tokens for an additional rS octets are generated, for a total burst size of (b + rs). Thus,
$b + rS = MS$
$S = b / (M - r)$

b. $S = (250 \times 10^3) / (23 \times 10^6) \approx 11$ msec

17.7 Sum the equation over all session j:

$$S_i(\tau,t)\phi_j \geq S_j(\tau,t)\phi_i$$

$$S_i(\tau,t)\sum_j \phi_j \geq \phi_i \sum_j S_j(\tau,t)$$

$$S_i(\tau,t)\sum_j \phi_j \geq \phi_i C(t-\tau)$$

$$\frac{S_i(\tau,t)}{(t-\tau)} \geq \frac{\phi_i}{\sum_j \phi_j} C$$

The last inequality demonstrates that session i is guaranteed a rate of

$$g_i = \frac{\phi_i}{\sum_j \phi_j} C$$

17.8 **a.** If the traffic is fairly bursty, then TH_{min} should be sufficiently large to allow the link utilization to be maintained at an acceptably high level.

b. The difference between the two thresholds should be larger then the typical increase in the calculated average queue length in one RTT.

17.9 60 Mbits

CHAPTER 18
PROTOCOLS FOR QOS SUPPORT

18.1 a. **Problem:** IPv6 inserts a variable number of variable-length Internet-layer headers before the transport header, increasing the difficulty and cost of packet classification for QoS. **Solution:** Efficient classification of IPv6 data packets could be obtained using the Flow Label field of the iPv6 header.

b. **Problem:** IP-level security, under either IPv4 or IPv6, may encrypt the entire transport header, hiding the port numbers of data packets from intermediate routers. **Solution:** There must be some means of identifying source and destination IP users (equivalent to the TCP or UDP port numbers). With IP-level security, there is a parameter, called the Security Parameter Index (SPI) carried in the security header. While SPIs are allocated based on destination address, they will typically be associated with a particular sender. As a result, two senders to the same unicast destination will usually have different SPIs. In order to support the control of multiple independent flows between source and destination IP addresses, the SPI will be included as part of the FILTER_SPEC.

18.2 The diagram is a simplification. Recall that the text states that "Transmissions from all sources are forwarded to all destinations through this router."

18.3 a. The purpose of the label is to get the packet to the final router. Once the next-to-last router has decided to send the packet to the final router, the label no longer has any function, and need no longer be carried.

b. It reduces the processing required at the last router: it need not pop the label. [RFC 3031]

18.4 a. When the last label is popped from a packet's label stack (resulting in the stack being emptied), further processing of the packet is based on the packet's network layer header. The LSR which pops the last label off the stack must therefore be able to identify the packet's network layer protocol.

b. The identity of the network layer protocol must be inferable from the value of the label which is popped from the bottom of the stack, possibly along with the contents of the network layer header itself. Therefore, when the first label is pushed onto a network layer packet, either the label must be one which is used ONLY for packets of a particular network layer, or the label must be one which is used ONLY for a specified set of network layer protocols, where packets of the specified network layers can be distinguished by inspection of the network layer header. Furthermore, whenever that label is replaced by another label value during a packet's transit, the new value must also be one which meets the same criteria. If these conditions are not met, the LSR which pops the last label off a packet will not be able to identify the packet's network layer protocol.

c. The restrictions only apply to the bottom (last) label in the stack. [RFC3032]

18.5 RTP uses periodic status reporting among all group participants. When the number of participants becomes large, the period of the reporting is scaled up to reduce the overall bandwidth consumption of control messages. Since the period of reporting may be large (on the order of tens of seconds), it may convey coarse grain information about the network congestion. Hence this information cannot

be used to relief network congestion, which needs finer grained feedback. This information, however, can be used as a means for session-level (as opposed to packet level) flow control, where a source may adjust its transmission rate, and receivers may adjust their playback points according to the coarse grained information.

18.6 Define SSRC_r = receiver issuing this receiver report; SSRC_n = source receiving this receiver report; A = arrival time of report; LSR = value in "Time of last sender report" field; DLSR = value in "Delay since last sender report" field. Then

Total round-trip time = A – LSR

Round-trip propagation delay = A – LSR – DLSR

CHAPTER 19
OVERVIEW OF INFORMATION THEORY

19.1 $X = H(P_1 + P_2, P_3) = -(P_1 + P_2)\log(P_1 + P_2) - P_3\log(P_3)$

$Y = (P_1 + P_2)H[(P_1/(P_1 + P_2), (P_2/(P_1 + P_2)]$

$= (P_1 + P_2)[(P_1/(P_1 + P_2)\log(P_1/(P_1 + P_2) + (P_2/(P_1 + P_2)\log(P_2/(P_1 + P_2)]$

$= -P_1\log(P_1) + P_1\log(P_1 + P_2) - P_2\log(P_2) + P_2\log(P_1 + P_2)$

$X + Y = -P_1\log(P_1) - P_2\log(P_2) - P_3\log(P_3) = H(P_1, P_2, P_3)$

19.2 No, because the code for x_2=0001 is the prefix for x_5=00011. The sequence 000110101111000110 can be deciphered as:

0001 101011 1100 0110 = $x_2\ x_8\ x_4\ x_3$

00011 010 11110 00110 = $x_5\ x_1\ x_7\ x_6$

19.3

a.		**b.**		**c.**		**d.**			
x_1	00	x_1	1	x_1	00	x_1	10	x_8	00110
x_2	11	x_2	00	x_2	10	x_2	000	x_9	00111
x_3	010	x_3	010	x_3	010	x_3	011	x_{10}	01000
x_4	011	x_4	0111	x_4	110	x_4	110	x_{11}	01001
x_5	100	x_5	01100	x_5	0001	x_5	111	x_{12}	001010
x_6	101	x_6	01101	x_6	1111	x_6	0101	x_{13}	001011
				x_7	1110	x_7	00100		
				x_8	01101				
				x_9	01100				

19.4 Let N be the average code-word length of C and N' that of C'. Then

$N' - N = (P_iL_k + P_kL_i) - (P_iL_i + P_kL_k) = (P_i - P_k)(L_k - L_i) < 0$

Therefore C' is less optimal than C.

19.5 Use Equation 6.1: $P(X) = \sum_i P(X/E_i)P(E_i)$ and assume $P_A = 0.8$ and $P_B = 0.2$.

$P_A = Pr(A/A)P_A + Pr(A/B)P_B = 0.9 \times 0.8 + 0.4 \times 0.2 = 0.72 + 0.08 = 0.8$

$P_B = Pr(B/A)P_A + Pr(B/B)P_B = 0.1 \times 0.8 + 0.6 \times 0.2 = 0.08 + 0.12 = 0.2$

19.6 **Scheme 1:** This coding scheme could have not been generated by a Huffman algorithm since it is not optimal. x4 could have been represented by a 1 instead of 10, and x3 could have been represented by a 00 instead of 001.

Scheme 2: It is a Huffman code because it is a prefix code, it is optimal, and it forms a full binary tree.

$P(x_1) = 0.55$; $P(x_2) = 0.25$; $P(x_3) = 0.1$; $P(x_4) = 0.1$

Scheme 3: This coding scheme could have not been generated by a Huffman algorithm since it is not a prefix scheme. x_1 is a prefix for x_3.

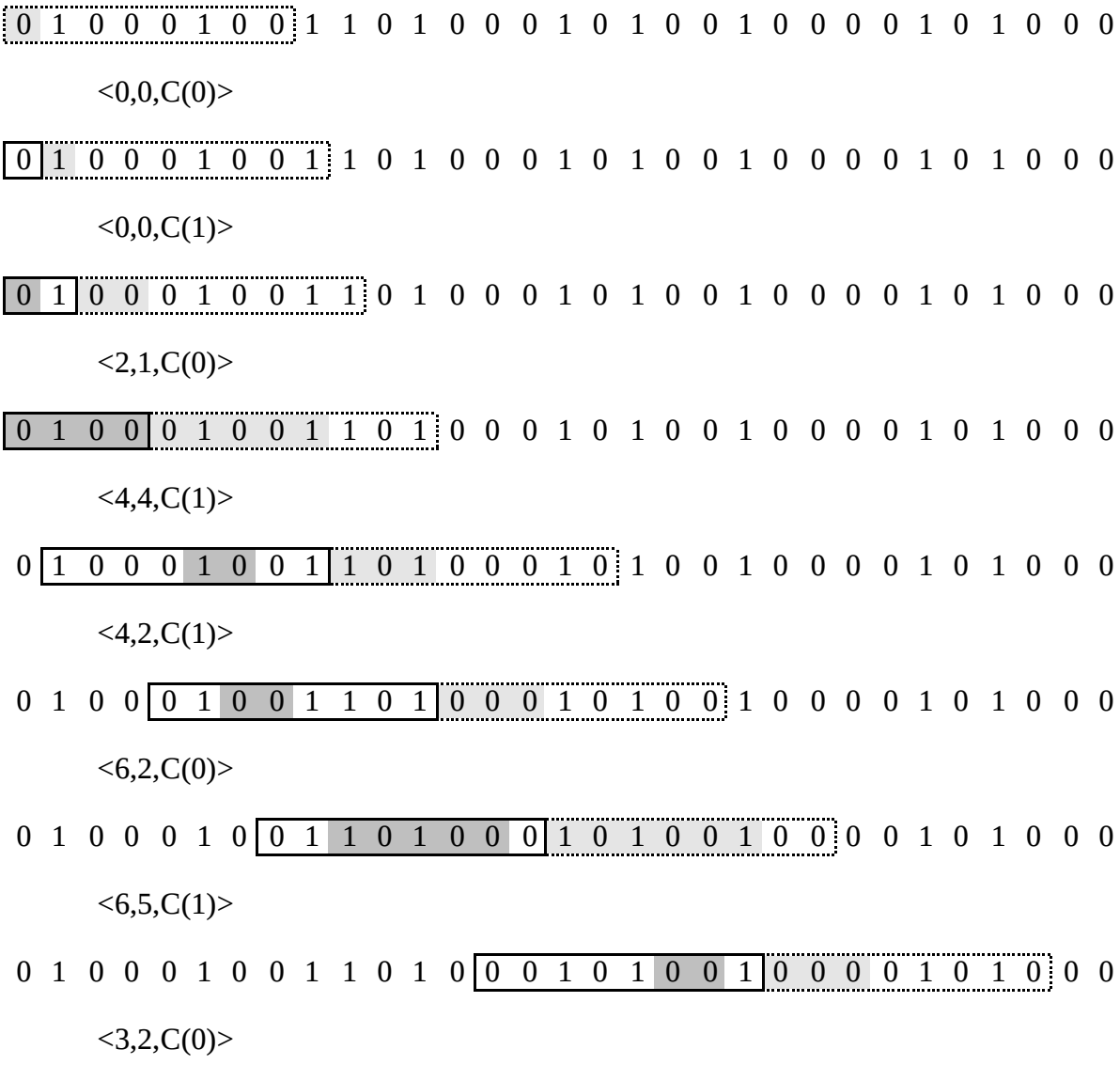
Scheme 4: It is a Huffman code because it is a prefix code, it is optimal, and it forms a full binary tree.

$P(x_1) = 0.3$; $P(x_2) = 0.2$; $P(x_3) = 0.1$; $P(x_4) = 0.4$

CHAPTER 20
LOSSLESS COMPRESSION

20.1 **Advantage:** MNP eliminates the necessity of using a special compression-indicating character because the three repeating characters both indicate that compression has occurred and identify the character that was compressed. **Disadvantages:** (1) MNP results in data expansion when sequence of only three repeating characters are encountered in the original data stream. (2) MNP encoding requires four characters compared to three characters for ordinary run-length encoding.

20.2



0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0

<5,3,C(1)>

0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0

<6,3,EOF>

20.3

Initial Dictionary

Index	Phrase
0	a
1	b

0 Decoding: *a*

Dictionary

Index	Phrase
0	a
1	b
2	a_

0 Decoding: a*a*

Dictionary

Index	Phrase
0	a
1	b
2	aa
3	a_

1 Decoding: aa*b*

Dictionary

Index	Phrase
0	a
1	b
2	aa
3	ab
4	b_

3 Decoding: aab*ab*

Dictionary

Index	Phrase
0	a
1	b
2	a*a*
3	a*b*
4	b*a*
5	ab_

5 Decoding: aabab*aba* Dictionary

Index	Phrase
0	a
1	b
2	a*a*
3	a*b*
4	b*a*
5	ab*a*
6	aba_

2 Decoding: aababab*aa* Dictionary

Index	Phrase
0	a
1	b
2	a*a*
3	a*b*
4	b*a*
5	ab*a*
6	aba*a*
7	aa_

0 0 1 3 5 2 => Final Decoding: aababababaaa

20.4 **a.** In LZ77, compression takes longer because the encoder has to search the buffer to find the best match, whereas the decoder's job is only to retrieve the symbols from the buffer, given an index and a length.

b. Similarly, for LZ78, the encoding process requires the dictionary to be searched for the best entry matching the input stream, whereas the decoder simply retrieves an entry given by an index. However, the difference might not be as significant as in LZ77, since both the encoder and decoder need to build a dictionary.

20.5 LZ77 is a better technique for encoding *aaaa…aaa*, assuming the length of the buffer is sufficiently large compared to k. In the best case, if the buffer size is k, the sequence can be encoded with two triplets: <0,0,C(a)> and <1,(k – 1),EOF>. LZ78 will need to generate a dictionary with entries like a, aa, aaa, etc., thus generating a longer bitstream, beside the overhead from using a dictionary. On the other hand, if the size of the LZ77 buffer is small compared to k, LZ78 would be slightly more efficient because LZ77 would encode a fixed number of symbols in every triplet (i.e., k symbols), while LZ78 will encode one additional symbol with each encoding.

20.6 a.

symbol	g	f	e	d	space	c	b	a
Probability	8/40	7/40	6/40	5/40	5/40	4/40	3/40	2/40
code	00	110	111	010	101	011	1000	1001

b.

a	1	6c	7	e	12	16f	17
1space	2	6space	8	12e	13	17f	18
b	3	d	9	13e	14	g	19
3b	4	9d	10	5f	15	19g	20
space	5	10space	11	f	16	20g	21
c	6						

c.

Symbol	P_i	Cumulative Probability	Interval	Binary Representation of Lower Bound	Code
a	0.05	0.05	[0, 0.05)	0.00000	00000
b	0.075	0.125	[0.05, 0.125)	0.00001	00001
c	0.1	0.225	[0.125, 0.225)	0.00100	0010
d	0.125	0.35	[0.225, 0.35)	0.00111	0011
e	0.15	0.5	[0.35, 0.5)	0.01011	01
f	0.175	0.675	[0.5, 0.675)	0.10000	100
g	0.2	0.875	[0.675, 0.875)	0.10101	101
space	0.125	1.0	[0.875, 1.0)	0.11100	11

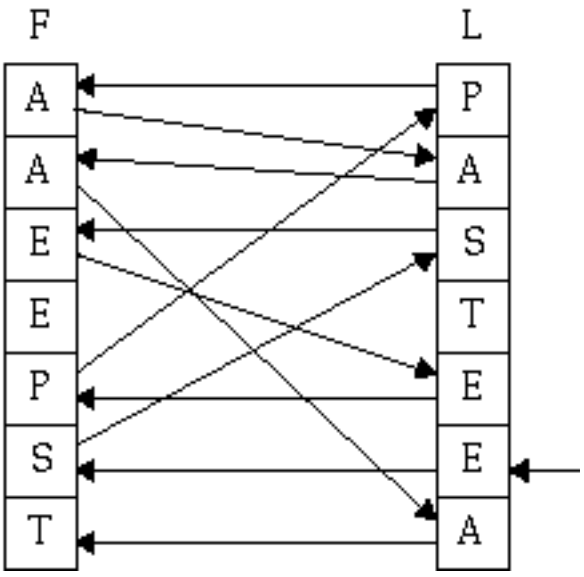
20.7 a. We need to exploit the following fact. Each row of the sorted matrix is a circular buffer that contains the original sequence in proper order, but rotated some number of positions. The following procedure will do the job. Label the final column in the sorted matrix L (the output string), the first column F, and the output integer I. Then perform these steps.

1. Given L, reconstruct F. This is easy: F contains the characters in L in alphabetical order.

2. Start with the character in row I, column L. By definition, this is the first character of the original string.

3. Go to the character in F in the current row. This is the next character in the string. This is so because each row is a circular buffer.

4. Go to the row that has the same character found in (3) in the Lth column. Repeat steps (3) and (4) until the entire original string is recovered. The following figure is an example.



In step 4, there may be more than one character in the Lth column matching the current character in the F column. In our example, the first time that an A is encountered in F, there are two choices for A in L. The ambiguity is resolved by observing that multiple instances of a character must appear in the same order in both F and L, although in different rows.

b. BWT alters the distribution of characters in a way that enhances compression. Each character in L is a prefix to the character string beginning with the character in F in the same row (except for the single case where L contains the last character in the string). BWT sorts the rows so that identical suffixes will be together. Often the same prefix character will occur for multiple instances of the same suffix. For example, if the sequence "hat" appears multiple times in a long string, all of these instances will be contiguous in rows beginning with "hat". Typically, the prefix character for almost all of these strings is "t". Thus, there could be a number of consecutive instances of "t" in L, providing an opportunity for compression.

CHAPTER 21
LOSSY COMPRESSION

21.1 The only change is to the dc component.

S(0)	S(1)	S(2)	S(3)	S(4)	S(5)	S(6)	S(7)
484	–129	–95	26	–73	4	–31	–11

21.2 **a.**

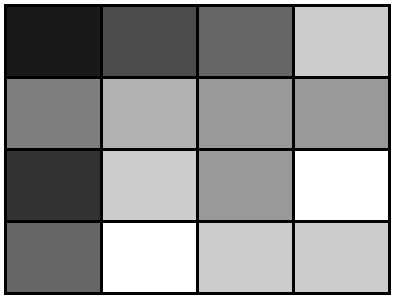
$$\mathbf{T_1} = \begin{bmatrix} 8 & 2 & 4 & 2 \\ 2 & 0 & -2 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} \qquad \mathbf{P}_1 = (\mathbf{W}')^{-1}\mathbf{T}_1\mathbf{W}^{-1}$$

$$\mathbf{P_1} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 8 & 2 & 4 & 2 \\ 2 & 0 & -2 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

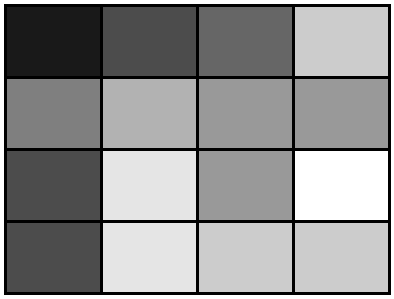
b.

$$= \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 14 & 6 & 8 & 4 \\ 0 & 4 & 2 & 2 \\ 4 & 4 & 2 & -2 \\ 0 & 0 & 2 & -2 \end{bmatrix} = \begin{bmatrix} 18 & 14 & 12 & 4 \\ 10 & 6 & 8 & 8 \\ 14 & 2 & 8 & 0 \\ 14 & 2 & 4 & 4 \end{bmatrix}$$

We use a gray scale with 21 levels, from 0 = white to 20 = black



P



$\mathbf{P}_1$

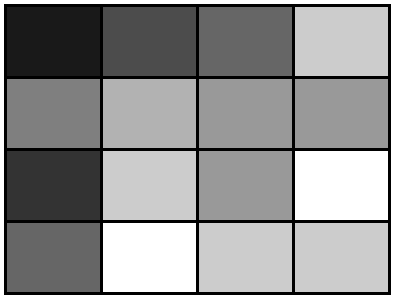
The two matrices are very similar.

c.

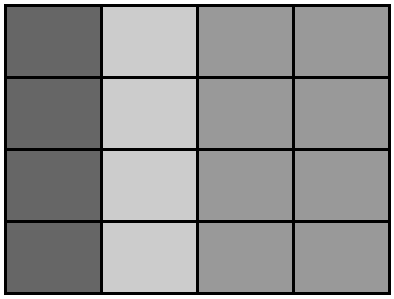
$$\mathbf{T_2} = \begin{bmatrix} 8 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{P}_2 = (\mathbf{W}')^{-1}\mathbf{T}_2\mathbf{W}^{-1}$$

$$\mathbf{P}_2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} 8 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} 12 & 4 & 8 & 8 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 12 & 4 & 8 & 8 \\ 12 & 4 & 8 & 8 \\ 12 & 4 & 8 & 8 \\ 12 & 4 & 8 & 8 \end{pmatrix}$$



P



$\mathbf{P}_2$

The two matrices are not very similar.

21.3 Predictor 0 is used for differential coding in the hierarchical mode of operation and for the first pixel in the top row. For the remaining pixels in the top row, A is the only preceding pixel (predictor 1). For the first pixel in each row but the top, B is the only preceding pixel (predictor 2). Beyond these obvious restrictions, any of the predictors can be used. Different images have different structures that can best be exploited by one of these eight modes of prediction. Predictor 3 is based only on the diagonal and predictor 7 is based only on the previous horizontal and vertical pixels. The remaining predictors use all three preceding pixels with different weightings.

21.4 At the receiver, the input frame is not available, only processed copies of each frames. Therefore, in order for the receiver to reproduce the processing of the sender, the sender must also used previous processed copies in the algorithm.