Q3)

10 Gbps = 10^{10} bps, meaning each bit is 10^{-10} sec (0.1 ns) wide. The length in the wire of such a bit is .1 ns $\times$ 2.3×10^8 m/sec = 0.023 m or 23mm

Q5)

(a) On a 100 Mbps network, each bit takes $1/10^8 = 10$ ns to transmit. One packet consists of 12000 bits, and so is delayed due to bandwidth (serialization) by 120 μs along each link. The packet is also delayed 10 μs on each of the two links due to propagation delay, for a total of 260 μs.

(b) With three switches and four links, the delay is

$$4 \times 120\mu s + 4 \times 10\mu s = 520\mu s$$

(c) With cut-through, the switch delays the packet by 200 bits = 2 μs. There is still one 120 μs delay waiting for the last bit, and 20 μs of propagation delay, so the total is 142 μs. To put it another way, the last bit still arrives 120 μs after the first bit; the first bit now faces two link delays and one switch delay but never has to wait for the last bit along the way.

Q6)

(a) 100×10^6bps $\times$ 10×10^{-6} sec = 1000 bits = 125 bytes.

(b) The first-bit delay is 520 μs through the store-and-forward switch, as in 16(a). 100×10^6bps $\times$ 520×10^{-6} sec = 52000 bits = 650 bytes.

(c) 1.5×10^6 bps $\times$ 50×10^{-3} sec = 75,000 bits = 9375 bytes.

(d) The path is *through* a satellite, *i.e.* between two ground stations, not *to* a satellite; this ground-to-satellite-to-ground path makes the total one-way travel distance $2 \times 35{,}900{,}000$ meters. With a propagation speed of $c = 3 \times 10^8$ meters/sec, the one-way propagation delay is thus $2 \times 35{,}900{,}000/c$ = 0.24 sec. Bandwidth$\times$delay is thus 1.5×10^6 bps $\times$ 0.24 sec = 360,000 bits $\approx$ 45 KBytes

Q7)

(a) Per-link transmit delay is 10^4 bits / 10^8 bps = 100 μs. Total transmission time including link and switch propagation delays = $2 \times 100 + 2 \times 20 + 35 =$ 275 μs.

(b) When sending as two packets, the time to transmit one packet is cut in half. Here is a table of times for various events:

T=0	start
T=50	A finishes sending packet 1, starts packet 2
T=70	packet 1 finishes arriving at S
T=105	packet 1 departs for B
T=100	A finishes sending packet 2
T=155	packet 2 departs for B
T=175	bit 1 of packet 2 arrives at B
T=225	last bit of packet 2 arrives at B

This is smaller than the answer to part (a) because packet 1 starts to make its way through the switch while packet 2 is still being transmitted on the first link, effectively getting a 50 μs head start. Smaller is faster, here.

Q8)

(a) Without compression the total time is 1 MB/$bandwidth$. When we compress the file, the total time is

$$compression\_time + compressed\_size/bandwidth$$

Equating these and rearranging, we get

$$bandwidth = compression\_size\_reduction/compression\_time$$

= 0.5 MB/1 sec = 0.5 MB/sec for the first case,
= 0.6 MB/2 sec = 0.3 MB/sec for the second case.

(b) Latency doesn't affect the answer because it would affect the compressed and uncompressed transmission equally.

Q10)

The time to send one 12000-bit packet is 12000 bits/100 Mbps = 120 μs. The length of cable needed to exactly contain such a packet is 120 μs $\times$ 2×10^8 m/sec = 24,000 meters.

12000 bits in 24000 meters is 50 bits per 100 m. With an extra 10 bits of delay in each 100 m, we have a total of 60 bits/100 m or 0.6 bits/m. A 12000-bit packet now fills 12000/(.6 bits/m) = 20,000 meters.

Q11)

(a) In the absence of any packet losses or duplications, when we are expecting the Nth packet we *get* the Nth packet, and so we can keep track of N locally at the receiver.

(b) The scheme outlined here is the stop-and-wait algorithm of Section 2.5; as is indicated there, a header with at least one bit of sequence number is needed (to distinguish between receiving a new packet and a duplication of the previous packet).

(c) With out-of-order delivery allowed, packets up to 1 minute apart must be distinguishable via sequence number. Otherwise a very old packet might arrive and be accepted as current. Sequence numbers would have to count as high as

$$bandwidth \times 1\,\text{minute}\,/packet\_size$$

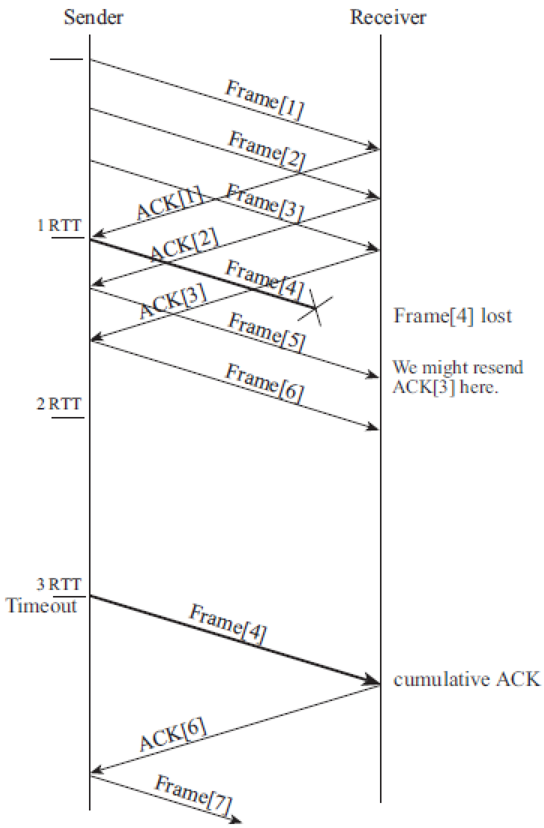
Q13)

(a) We take the message 11100011, append 000 to it, and divide by 1001 according to the method shown in Section 2.4.3. The remainder is 100; what we transmit is the original message with this remainder appended, or 1110 0011 100.

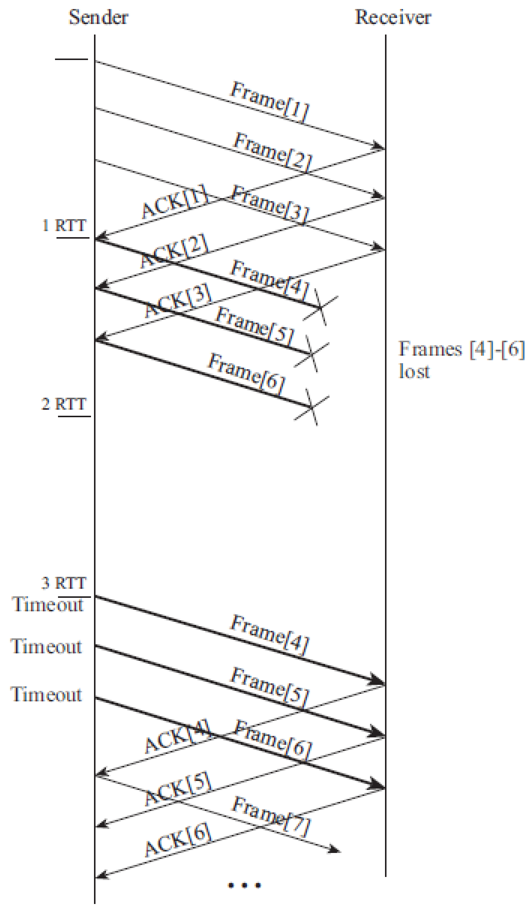
(b) Inverting the first bit of the transmission gives 0110 0011 100; dividing by 1001 ($x^3 + 1$) gives a remainder of 10; the fact that the remainder is non-zero tells us a bit error occurred.

Q17)

The right diagram, for part (b), shows each of frames 4-6 timing out after a 2×RTT timeout interval; a more realistic implementation (*e.g.* TCP) would probably revert to SWS=1 after losing packets, to address both congestion control and the lack of ACK clocking.



Sender
Receiver
Frame[1]
Frame[2]
ACK[1]
Frame[3]
1 RTT
ACK[2]
Frame[4]
ACK[3]
Frame[4] lost
Frame[5]
We might resend
ACK[3] here.
Frame[6]
2 RTT
3 RTT
Timeout
Frame[4]
cumulative ACK
ACK[6]
Frame[7]



Sender
Receiver
Frame[1]
Frame[2]
ACK[1]
Frame[3]
1 RTT
ACK[2]
Frame[4]
ACK[3]
Frame[5]
Frames [4]-[6]
lost
Frame[6]
2 RTT
3 RTT
Timeout
Frame[4]
Timeout
Frame[5]
Timeout
ACK[4]
Frame[6]
ACK[5]
Frame[7]
ACK[6]