Solutions to Select Exercises

CHAPTER 1

4. We will count the transfer as completed when the last data bit arrives at its destination

(a) 1.5 MB = 12582912 bits. 2 initial RTTs (160 ms) + 12,582,912/10,000,000 bps (transmit) + RTT/2 (propagation) $\approx$ 1.458 seconds.

(b) Number of packets required = 1.5 MB/1KB = 1536. To the above we add the time for 1535 RTTs (the number of RTTs between when packet 1 arrives and packet 1536 arrives), for a total of $1.458 + 122.8 = 124.258$ seconds.

(c) Dividing the 1536 packets by 20 gives 76.8. This will take 76.5 RTTs (half an RTT for the first batch to arrive, plus 76 RTTs between the first batch and the 77th partial batch), plus the initial 2 RTTs, for 6.28 seconds.

(d) Right after the handshaking is done we send one packet. One RTT after the handshaking we send two packets. At n RTTs past the initial handshaking we have sent $1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1$ packets. At $n = 10$ we have thus been able to send all 1536 packets; the last batch arrives 0.5 RTT later. Total time is $2 + 10.5$ RTTs, or 1 second.

6. Propagation delay is 50×10^3 m$/(2 \times 10^8$ m/s$) = 250\,\mu$s. 800 bits / 250 μs is 3.2 Mbps. For 512-byte packets, this rises to 16.4 Mbps.

14. **(a)** Propagation delay on the link is $(55 \times 10^9)/(3 \times 10^8) = 184$ seconds. Thus, the RTT is 368 seconds.

(b) The delay $\times$ bandwidth product for the link is $184 \times 128 \times 10^3 = 2.81$ MB.

(c) After a picture is taken, it must be transmitted on the link and be completely propagated before Mission Control can interpret it. Transmit delay for 5 MB of data is 41,943,040 bits/$128 \times 10^3 = 328$ seconds. Thus, the total time required is transmit delay + propagation delay $= 328 + 184 = 512$ seconds.

17. **(a)** For each link, it takes 1 Gbps / 5 kb $= 5\,\mu$s to transmit the packet on the link, after which it takes an additional $10\,\mu$s for the last bit to propagate across the link. Thus, for a LAN with only one switch that starts forwarding only after receiving the whole packet, the total transfer delay is two transmit delays + two propagation delays $= 30\,\mu$s.

(b) For three switched and thus four links, the total delay is four transmit delays + four propagation delays $= 60\,\mu$s.

(c) For cut-through, a switch need only decode the first 128 bits before beginning to forward. This takes 128 ns. This delay replaces the switch transmit delays in the previous answer for a total delay of one transmit delay + three cut-through decoding delays + four propagation delays $= 45.384\,\mu$s.

27. **(a)** $1920 \times 1080 \times 24 \times 30 = 1{,}492{,}992{,}000 \approx 1.5$ Gbps.

(b) $8 \times 8000 = 64$ Kbps.

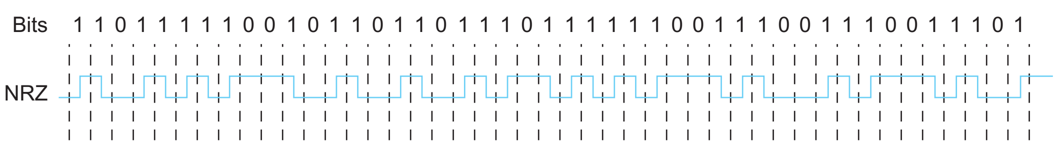
(c) $260 \times 50 = 13$ Kbps.

(d) $24 \times 88{,}200 = 216{,}800 \approx 2.1$ Mbps.

CHAPTER 2

3. The 4B/5B encoding of the given bit sequence is the following:

11011 11100 10110 11011 10111 11100 11100 11101



7. Let $\wedge$ mark each position where a stuffed 0 bit was removed. There was one error where the sever consecutive 1s are detected (err) At the end of the bit sequence, the end of frame was detected (eof).

01101011111$\wedge$101001111111$\underline{\mathbf{1}}_{err}$0 110 $\underline{\mathbf{01111110}}_{eof}$

19. **(a)** We take the message 1011 0010 0100 1011, append 8 zeros and divide by 1 0000 0111 $(x^8 + x^2 + x^1 + 1)$. The remainder is 1001 0011. We transmit the original message with this remainder appended, resulting in 1011 0010 0100 0011 1001 0011.

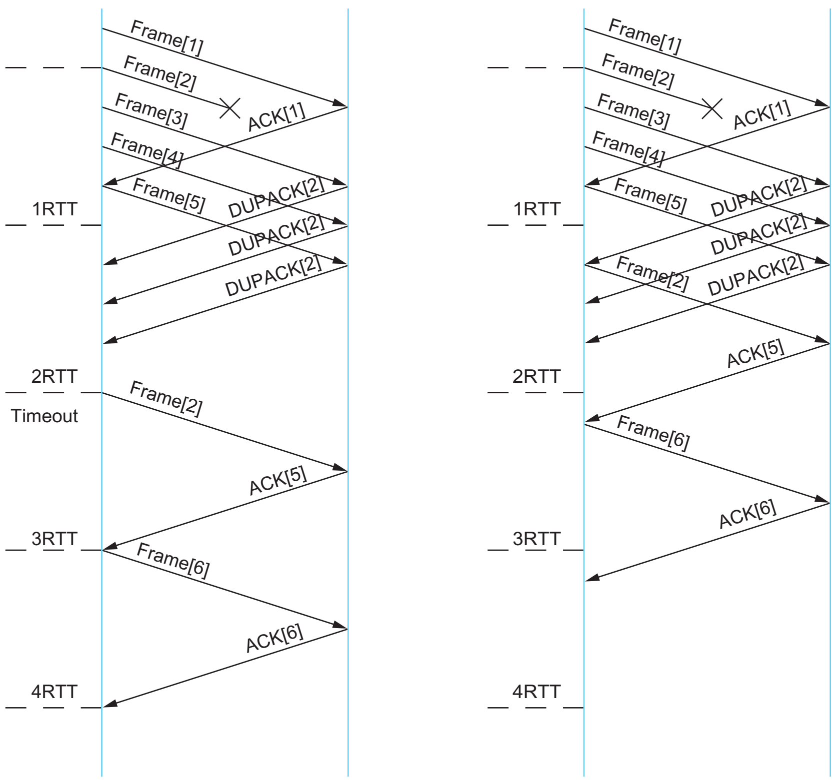
(b) Inverting the first bit gives 0011 0010 0100 1011 1001 0011. Dividing by 1 0000 0111 $(x^8 + x^2 + x^1 + 1)$ gives a a remainder of 1011 0110.

25. One-way latency of the link is 100 ms. (Bandwidth) $\times$ (roundtrip delay) is about 125 pps $\times$ 0.2 sec, or 25 packets. SWS should be this large.

(a) If RWS $=$ 1, the necessary sequence number space is 26. Therefore, 5 bits are needed.

(b) If RWS $=$ SWS, the sequence number space must cover twice the SWS, or up to 50. Therefore, 6 bits are needed.

32. The figure that follows gives the timeline for the first case. The second case reduces the total transaction time by roughly 1 RTT.



CHAPTER 3

2. The following table is cumulative; at each part the VCI tables consist of the entries at that part and also all previous entries. Note that at stage (d) we assume that VCI 0 on port 0 of switch 4 cannot be reused (it was used for a connection to H in part (a)). This would correspond to the case where VCIs are bidirectional, as they commonly are.

Exercise Part	Switch	Input		Output	
		Port	VCI	Port	VCI
(a)	1	0	0	1	0
	2	3	0	1	0
	4	3	0	0	0
(b)	2	0	0	1	1
	3	3	0	0	0
	4	3	1	1	0
(c)	1	1	1	2	0
	2	1	2	3	1
	4	2	0	3	2
(d)	1	1	2	3	0
	2	1	3	3	2
	4	0	1	3	3
(e)	2	0	1	2	0
	3	2	0	0	1
(f)	2	1	4	0	2
	3	0	2	1	0
	4	0	2	3	4

14. The following list shows the mapping between LANs and their designated bridges.
B1 dead
B2 A,B,D

B3 E,F,G,H
B4 I
B5 idle
B6 J
B7 C

16. All bridges see the packet from D to C. Only B3, B2, and B4 see the packet from C to D. Only B1, B2, and B3 see the packet from A to C.

B1	A-interface : A	B2-interface : D (not C)	
B2	B1-interface : A	B3-interface : C	B4-interface : D
B3	C-interface : C	B2-interface : A,D	
B4	D-interface : D	B2-interface : C (not A)	

27. Since the I/O bus speed is less than the memory bandwidth, it is the bottleneck. Effective bandwidth that the I/O bus can provide is 1000/2 Mbps because each packet crosses the I/O bus twice. Therefore, the number of interfaces is $(500/100) = 5$.

37. By definition, path MTU is 576 bytes. Maximum IP payload size is $576 - 20 = 556$ bytes. We need to transfer $1024 + 20 = 1044$ bytes in the IP payload. This would be fragmented into 2 fragments, the first of size 552 bytes (because the fragment needs to be a multiple of 8 bytes, so it can't be exactly 556) and the second of size $1044 - 552 = 492$ bytes. There are 2 packets in total if we use path MTU. In the previous setting we needed 3 packets.

47. **(a)**

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	2	∞	5	∞	∞
B	2	0	2	∞	1	∞
C	∞	2	0	2	∞	3
D	5	∞	2	0	∞	∞
E	∞	1	∞	∞	0	3
F	∞	∞	3	∞	3	0

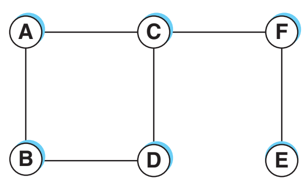
(b)

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	2	4	5	3	∞
B	2	0	2	4	1	4
C	4	2	0	2	3	3
D	5	4	2	0	∞	5
E	3	1	3	∞	0	3
F	∞	4	3	5	3	0

(c)

Information Stored at Node	Distance to Reach Node					
	A	B	C	D	E	F
A	0	2	4	5	3	6
B	2	0	2	4	1	4
C	4	2	0	2	3	3
D	5	4	2	0	5	5
E	3	1	3	5	0	3
F	6	4	3	5	3	0

53. The following is an example network topology.



56. Apply each subnet mask and, if the corresponding subnet number matches the SubnetNumber column, then use the entry in Next-Hop.

(a) Applying the subnet mask 255.255.254.0, we get 128.96.170.0. Use interface 0 as the next hop.

(b) Applying subnet mask 255.255.254.0, we get 128.96.166.0. (Next hop is Router 2.) Applying subnet mask 255.255.252.0, we get 128.96.164.0. (Next hop is Router 3.) However, 255.255.254.0 is a longer prefix, so use Router 2 as the next hop.

(c) None of the subnet number entries match, so use default Router R4.

(d) Applying subnet mask 255.255.254.0, we get 128.96.168.0. Use interface 1 as the next hop.

(e) Applying subnet mask 255.255.252.0, we get 128.96.164.0. Use Router 3 as the next hop.

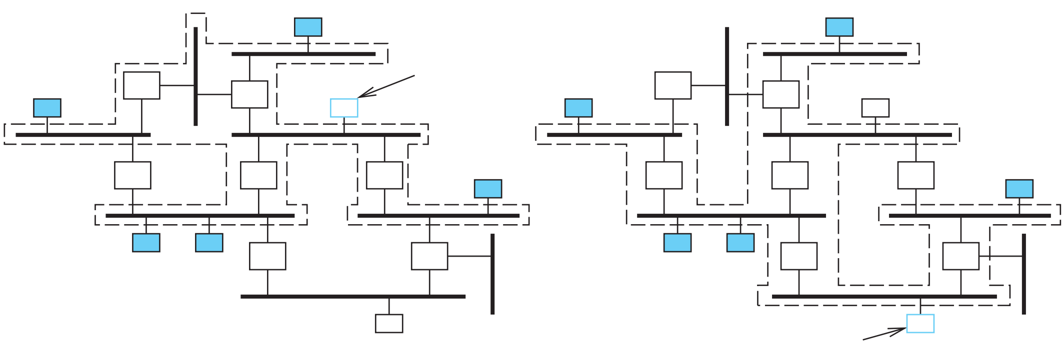
63.

Step	Confirmed	Tentative
1	(A,0,-)	
2	(A,0,-)	(B,1,B) (D,5,D)
3	(A,0,-) (B,1,B)	(D,4,B) (C,7,B)
4	(A,0,-) (B,1,B) (D,4,B)	(C,5,B) (E,7,B)
5	(A,0,-) (B,1,B) (D,4,B) (C,5,B)	(E,6,B)
6	(A,0,-) (B,1,B) (D,4,B) (C,5,B) (E,6,B)	

73. (a) F (b) B (c) E (d) A (e) D (f) C

CHAPTER 4

15. The following figures illustrate the multicast trees for sources D and E.



CHAPTER 5

10. The advertised window should be large enough to keep the pipe full; delay (RTT) $\times$ bandwidth here is 140 ms $\times$ 1 Gbps = 10 Mb = 17.5 MB of data. This requires 25 bits ($2^{25} = 33,554,432$) for the

AdvertisedWindow field. The sequence number field must not wrap around in the maximum segment lifetime. In 60 seconds, 7.5 GB can be transmitted. 33 bits allows a sequence space of 8.6 GB, and so will not wrap in 60 seconds.

13. **(a)** 2^{32} B / (5 GB) = 859 ms.

(b) 1000 ticks in 859 ms is once each 859 μs indicating wrap around in 3.7 Ms or approximately 43 days.

27. Using initial Deviation = 50 it took 20 iterations for TimeOut to fall below 300.0.

Iteration	SampleRTT	EstRTT	Dev	diff	TimeOut
0	200.0	90.0	50.0		
1	200.0	103.7	57.5	110.0	333.7
2	200.0	115.7	62.3	96.3	364.9
3	200.0	126.2	65.0	84.3	386.2
4	200.0	135.4	66.1	73.8	399.8
5	200.0	143.4	66.0	64.6	407.4
6	200.0	150.4	64.9	56.6	410.0
7	200.0	156.6	63.0	49.6	408.6
8	200.0	162.0	60.6	43.4	404.4
9	200.0	166.7	57.8	38.0	397.9
10	200.0	170.8	54.8	33.3	390.0
11	200.0	174.4	51.6	29.2	380.8
12	200.0	177.6	48.4	25.6	371.2
13	200.0	180.4	45.2	22.4	361.2
14	200.0	182.8	42.0	19.6	350.8
15	200.0	184.9	38.9	17.2	340.5
16	200.0	186.7	36.0	15.1	330.7
17	200.0	188.3	33.2	13.3	321.1
18	200.0	189.7	30.6	11.7	312.1
19	200.0	190.9	28.1	10.3	303.3
20	200.0	192.0	25.8	9.1	295.2

CHAPTER 6

11. (a) First we calculate the finishing times F_i. We don't need to worry about clock speed here since we may take $A_i = 0$ for all the packets. F_i thus becomes just the cumulative per-flow size: $F_i = F_{i-1} + P_i$.

Packet	Size	Flow	F_i
1	200	1	200
2	200	1	400
3	160	2	160
4	120	2	280
5	160	2	440
6	210	3	210
7	150	3	360
8	90	3	450

We now send in increasing order of F_i: Packet 3, Packet 1, Packet 6, Packet 4, Packet 7, Packet 2, Packet 5, Packet 8.

(b) To give flow 1 a weight of 2 we divide each of its F_i by 2: $F_i = F_{i-1} + P_i/2$. To give flow 2 a weight of 4 we divide each of its F_i by 4: $F_i = F_{i-1} + P_i/4$. To give flow 3 a weight of 3 we divide each of its F_i by 3: $F_i = F_{i-1} + P_i/3$. Again, we are using the fact that there is no waiting.

Packet	Size	Flow	Weighted F_i
1	200	1	100
2	200	1	200
3	160	2	40
4	120	2	70
5	160	2	110
6	210	3	70
7	150	3	120
8	90	3	150

Transmitting in increasing order of the weighted F_i we send as follows: Packet 3, Packet 4, Packet 6, Packet 1, Packet 5, Packet 7, Packet 8, Packet 2.

15. (a) For the ith arriving packet on a given flow we calculate its estimated finishing time F_i by the formula $F_i = \max\{A_i, F_{i-1}\} + 1$, where the clock used to measure the arrival times A_i runs slow by a factor equal to the number of active queues. The A_i clock is global; the sequence of F_i values calculated as above is local to each flow.

The following table lists all events by wall clock time. We identify packets by their flow and arrival time; thus, packet A4 is the packet that arrives on flow A at wall clock time 4 (ie., the third packet). The last three columns are the queues for each flow for the subsequent time interval, *including* the packet currently being transmitted. The number of such active queues determines the amount by which A_i is incremented on the subsequent line. Multiple packets appear on the same line if their F_i values are all the same; the F_i values are in italic when $F_i = F_{i-1} + 1$ (versus $F_i = A_i + 1$).

Wall Clock	A_i	Arrivals	F_i	Sent	A's Queue	B's Queue	C's Queue
1	1.0	A1,B1,C1	2.0	A1	A1	B1	C1
2	1.333	C2	3.0	B1		B1	C1,C2
3	1.833	A3	3.0	C1	A3		C1,C2
4	2.333	B4	3.333	A3	A3	B4	C2,C4
		C4	4.0				
5	2.666	A5	4.0	C2	A5	B4	C2,C4
6	3.0	A6	5.0	B4	A5,A6	B4	C4,C6
		C6	5.0				
7	3.333	B7	4.333	A5	A5,A6	B7	C4,C6,C7
		C7	6.0				
8	3.666	A8	6.0	C4	A6,A8	B7,B8	C4,C6,C7
		B8	5.333				
9	4	A9	7.0	B7	A6,A8,A9	B7,B8,B9	C6,C7
		B9	6.333				

(Continued)

Wall Clock	A_i	Arrivals	F_i	Sent	A's Queue	B's Queue	C's Queue
10	4.333			A6	A6,A8,A9	B8,B9	C6,C7
11	4.666	A11	8.0	C6	A8,A9,A11	B8,B9	C7
12	5	C12	7.0	B8	A8,A9,A11	B8,B9	C7,C12
13	5.333	B13	7.333	A8	A8,A9,A11	B9,B13	C7,C12
14	5.666			C7	A9,A11	B9,B13	C7,C12
15	6.0	B15	8.333	B9	A9,A11	B9,B13,B15	C12
16	6.333			A9	A9,A11	B13,B15	C12
17	6.666			C12	A11	B13,B15	C12
18	7			B13	A11	B13,B15	
19	7.5			A11	A11	B15	
20	8			B15		B15	

(b) For weighted fair queuing we have, for flow B,

$$F_i = max\{A_i, F_{i-1}\} + 0.5$$

For flows A and C, F_i is as before. Here is the table corresponding to the one above:

Wall Clock	A_i	Arrivals	F_i	Sent	A's Queue	B's Queue	C's Queue
1	1.0	A1,C1	2.0	B1	A1	B1	C1
		B1	1.5				
2	1.333	C2	3.0	A1			C1,C2
3	1.833	A3	3.0	C1	A1		C1,C2
4	2.333	B4	2.833	B4	A3	B4	C2,C4
		C4	*4.0*				
5	2.666	A5	4.0	A3	A3,A5		C2,C4
6	3.166	A6	5.0	C2	A5,A6		C2,C4,C6
		C6	5.0				
7	3.666	B7	4.167	A5	A5,A6	B7	C4,C6,C7
		C7	6.0				
8	4.0	A8	6.0	C4	A6,A8	B7,B8	C6,C7
		B8	4.666				

(Continued)

Wall Clock	A_i	Arrivals	F_i	Sent	A's Queue	B's Queue	C's Queue
9	4.333	A9	7.0	B7	A6,A8,A9	B7,B8,B9	C6,C7
		B9	5.166				
10	4.666			B8	A6,A8,A9	B8,B9	C6,C7
11	5.0	A11	8.0	A6	A6,A8,A9,A11	B9	C6,C7
12	5.333	C12	7.0	C6	A8,A9,A11	B9	C6,C7,C12
13	5.666	B13	6.166	B9	A8,A9,A11	B9,B13	C7,C12
14	6.0			A8	A9,A11	B13	C7,C12
15	6.333	B15	6.833	C7	A9,A11	B13,B15	C12
16	6.666			B13	A9,A11	B13,B15	C12
17	7.0			B15	A11	B15	C12
18	7.333			A9	A11		C12
19	7.833			C12	A11		C12
20	8.333			A11	A11		

35. **(a)** We have

$$\mathsf{TempP} = \mathsf{MaxP} \times \frac{\mathsf{AvgLen} - \mathsf{MinThreshold}}{\mathsf{MaxThreshold} - \mathsf{MinThreshold}}.$$

AvgLen is halfway between MinThreshold and MaxThreshold, which implies that the fraction here is $1/2$ and so $\mathsf{TempP} = \mathsf{MaxP}/2 = p/2$. We now have

$$\mathsf{P}_{\mathsf{count}} = \mathsf{TempP}/(1 - \mathsf{count} \times \mathsf{TempP}) = 1/(x - \mathsf{count}),$$

where $x = 2/p$. Therefore,

$$1 - \mathsf{P}_{\mathsf{count}} = \frac{x - (\mathsf{count} + 1)}{x - \mathsf{count}}.$$

Evaluating the product

$$(1 - \mathsf{P}_1) \times \cdots \times (1 - \mathsf{P}_n)$$

gives

$$\frac{x-2}{x-1} \cdot \frac{x-3}{x-2} \cdots \frac{x-(n+1)}{x-n} = \frac{x-(n+1)}{x-1},$$

where $x = 2/p$.

(b) From the result of previous question,

$$\alpha = \frac{x-(n+1)}{x-1}.$$

Therefore,

$$x = \frac{(n+1)-\alpha}{1-\alpha} = 2/p.$$

Accordingly,

$$p = \frac{2(1-\alpha)}{(n+1)-\alpha}$$

48. At every second, the bucket volume must not be negative. For a given bucket depth D and token rate r, we can calculate the bucket volume $v(t)$ at time t seconds and enforce $v(t)$ being non-negative:

$$\begin{aligned}
v(0) &= D-5+r = D-(5-r) \geq 0\\
v(1) &= D-5-5+2r = D-2(5-r) \geq 0\\
v(2) &= D-5-5-1+3r = D-(11-3r) \geq 0\\
v(3) &= D-5-5-1+4r = D-(11-4r) \geq 0\\
v(4) &= D-5-5-1-6+5r = D-(17-5r) \geq 0\\
v(5) &= D-5-5-1-6-1+6r = D-6(3-r) \geq 0
\end{aligned}$$

We define the functions $f_1(r), f_2(r), \ldots, f_6(r)$ as follows:

$$\begin{aligned}
f_1(r) &= 5-r\\
f_2(r) &= 2(5-r) = 2f_1(r) \geq f_1(r) \quad (\mathit{for}\ 1 \leq r \leq 5)\\
f_3(r) &= 11-3r \leq f_2(r) \quad (\mathit{for}\ r \geq 1)\\
f_4(r) &= 11-4r < f_3(r) \quad (\mathit{for}\ r \geq 1)\\
f_5(r) &= 17-5r\\
f_6(r) &= 6(3-r) \leq f_5(r) \quad (\mathit{for}\ r \geq 1)
\end{aligned}$$

First of all, for $r \geq 5$, $f_i(r) \leq 0$ for all i. This means if the token rate is faster than 5 packets per second any positive bucket depth will suffice (i.e., $D \geq 0$). For $1 \leq r \leq 5$, we only need to consider $f_2(r)$ and $f_5(r)$, since other functions are less than these functions.

One can easily find $f_2(r) - f_5(r) = 3r - 7$. Therefore, the bucket depth D is enforced by the following formula:

$$D \geq \begin{cases} f_5(r) = 17 - 5r & (r = 1, 2) \\ f_2(r) = 2(5 - r) & (r = 3, 4, 5) \\ 0 & (r \geq 5) \end{cases}$$

CHAPTER 7

2. Each string is preceded by a count of its length; the array of salaries is preceded by a count of the number of elements. That leads to the following sequence of integers and ASCII characters being sent:

```
4 M A R Y 4377 7 J A N U A R Y 7 2002 2 90000 150000 1
```

8.

INT	4	15
INT	4	29496729
INT	4	58993458

10.

```
15 be 00000000 00000000 00000000 00001111
15 le 00001111 00000000 00000000 00000000

29496729 be 00000001 11000010 00010101 10011001
29496729 le 10011001 00010101 11000010 00000001

58993458 be 00000011 10000100 00101011 00110010
58993458 le 00110010 00101011 10000100 00000011
```