Network Systems (201600146/201600197), Test 1

February 28, 2020, 13:45–15:45

Answers

1. P2P Systems

1 pt (a) A.

1 pt (b) D.

The value is stored on peer 5; so peer 9 will consecutively contact peers 11, 15, 2, and finally 5. That's a total of 4 peers contacted.
We also accepted answer C, since it wasn't 100% clear that the number asked for should *include* the peer that finally has the value.

1 pt (c) C.

In this case, peer 9 contacts peers 15, 2 and 5, that's a total of 3.
Note that it can't use the shortcut from 15 to 5, because it cannot know whether there is a peer number 3 or 4. If there would be a peer number 3 or 4, the value we're looking for would be stored there, rather than on peer 5, and taking the shortcut from 15 to 5 would thus be wrong.
Only for students who answered C in the previous question we accept B in this question, for consistency.

2. Application protocols

1 pt (a) D.

1 pt (b) D.

Originally, e-mail was only suitable for sending text.
Text is represented using 1 byte (8 bits) per letter. 8 bits have 2^8 possible values, not all of which are needed for representing letters and other characters in the so-called ASCII code. E-mail by itself was not designed to send mails which contain the other possible byte values which don't represent normal letters etc., but such byte values may occur in binary files such as pictures.
By using base64 encoding, the binary data from the image file is converted to bytes that are allowed in an e-mail.
This does not give compression, in fact it's the opposite: for every 3 bytes from the image, 4 bytes are used in the e-mail.

1 pt (c) C.

1 pt (d) C.

We intended answer C, since HTTP/2 makes it possible to already send e.g. pictures that it expects the client will need, before the client asks for it, thus saving an RTT.
We also accepted A, since HTTP/2 can compress the headers.
(But when we wrote the question, we thought of "data" referring only to the actual data in the files (HTML, images, etc.): for that data, HTTP/2 doesn't do more compression than 1.1 already does.)

3. Reliable communication performance

1.5 pt (a) Propagation delay = $2 \cdot 37\,500 \;/\; 300\,000 = 0.25$ s (250 ms).

Note that we need to go both to the satellite and back, so the distance is twice 37 500 km.

1.5 pt (b) Transmission delay = $8 \cdot 125 \;/\; 10\,000\,000 = 0.0001$ s (0.1 ms).

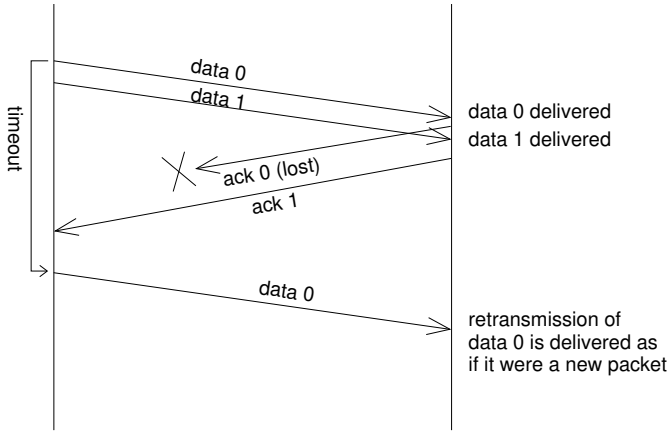
Note the factor of 8 to go from bytes to bits.

2 pt (c) $0.25 \cdot 10\,000\,000 = 2\,500\,000$ bits (2.5 Mbits)

2 pt (d) Using a stop-and-wait algorithm, after every transmission of a packet of 8 × 125 bits (taking 0.1 ms), 2 × the propagation delay (2 x 250 ms) has to be spent waiting for the ACK to come back, after which the next packet can be transmitted. So, 8 × 125 / 0.5 = 2 000 bits/s (2 kbit/s)

Strictly speaking, we have to wait one transmission time more than just the propagation time (since the ACK won't be sent until the entire packet has been received); but in this case, the transmission time is very small compared to the propagation time so it doesn't make much difference: 8 × 125 / (0.5+0.0001) = 1999.6 bits/s.

2 pt (e) There are many possibilities, for example:



2 pt (f) That should be the round-trip time (RTT) × link data rate / packet size
= 2 × 0.25 × 10 000 000 / (8 × 125) = 5000 packets

1 pt (g) One more than the answer to the previous question (because the RWS=1).

1 pt (h) D.

Suppose packets 2, 3, 4, 5, and 6 are sent, and packet 4 is lost.
If RWS=1, the receiver, after receiving packets 2 and 3, will only accept packet 4, which it doesn't get, so it will discard 5 and 6. Thus, not only the lost packet 4 has to be retransmitted, but also 5 and 6.
O.t.o.h., if RWS would be at least 3 in this example, the receiver could store packets 5 and 6, and only packet 4 would need to be retransmitted. Of course, the receiver would not deliver packets 5 and 6 to the higher protocol layer until it has also received packet 4, so it can deliver them in the correct order.

---

4. Information theory

3 pt (a) Use the formula: $H = 0.70 \log_2 \frac{1}{0.70} + 0.15 \log_2 \frac{1}{0.15} + 0.10 \log_2 \frac{1}{0.10} + 0.05 \log_2 \frac{1}{0.05} = 1.319$ bits per message.

1 pt (b) B.

1 pt (c) A.

Optimal source coding would use such dependencies to reduce the number of bits needed.
E.g., when coding an English text, you could assign a shorter bit pattern to the combination of letters TH than to the combination TQ, since there are many the combination words with TH, while the combination TQ very rarely occurs (even though T and Q independently occur often). This is more efficient than treating the letters independently.

3 pt (d) Correct answers are, respectively: E, G, D, G, G.

MC12: no choice possible because the code as given is already not uniquely decodable: if 00 is received, is that a single message 00, or twice the message 0 ?
MC14 and M15: no choice possible because the average codeword becomes too long, regardless of the choice made for the '`ill`' message.

1 pt (e) B.

Many students chose one of the channel coding options, but that's not correct: channel coding is adding extra

redundancy to be able to detect or correct errors.

2 pt (f) Calculate the channel capacity: $C = 10^5 \cdot \log_2(1+3) = 2 \cdot 10^5$ bit/s.
Divide by the entropy from question (a) to find 151625.96777 messages per second, say 151 thousand messages per second.

1 pt (g) G.

The messages will still be correctly encoded and decoded, but the average number of bits needed will be larger. We expected few '`ill`' messages, so we assigned a long codeword to them, and now we have to use that long codeword relatively often.

1 pt (h) E.

There's a limited number of valid codewords. The sender sends one of them; the receiver receives it but with some errors. The assumption is that bit errors are unlikely, so the receiver first looks whether by changing only one bit it can get to a valid codeword; if not, then it tries changing two bits, and so on. (Well, that's the principle — in reality more efficient algorithms are used than just trying.)

I was surprised to see very many students choose answer D; not only was the answer itself wrong (the CRC long division calculation is only for detecting errors, not correcting), but even the expansion given there for the abbreviation "CRC" was wrong.

3 pt (i)

```
1011 / 01110000 \ 01100
       1011
       ----
        1010
        1011
        ----
         00100
```

So the CRC is 100.

The most frequently made errors were:

- Forgetting to append three 0s to the message (to make "space" for the CRC, corresponding to multiplying by x^k in the polynome notation).
- Doing the first subtraction with 1011 aligned under the 0111 bits, rather than under the 1110 bits: this doesn't work because after this subtraction the leftmost bit becomes a, 1 making the number bigger rather than smaller.
  Think about what you would do in a normal decimal long division if the left-most digit would be a 0: you would just ignore that digit and start with next.

1 pt (j) A.

The light is transported inside the core (the center part of the glass fiber). The cladding (outer layer) is only needed to guarantee that the light does not escape from the core: thanks to the phenomenon of total internal reflection.

1 pt (k) C.

Many chose E, but that is wrong. When the surrounding medium (cladding, or air in this question) has a lower index of refraction, it makes total internal reflection possible, so the light can stay inside the core.
B.t.w., the question contained an (obvious) typo: the "index of reflection" doesn't exist, this should read "index of refraction".

1 pt (l) B.

Multipath propagation means there are multiple paths from the sender to the (same) receiver. It's not about multiple paths to multiple receivers (such as eavesdroppers). These multiple paths generally have different lengths, which makes reception harder as differently-delayed versions of the signal interfere with each other.

1 pt (m) D.

1 pt (n) E.

The extra byte 7E in the middle of the frame would be considered a frame separation marker by the receiver; thus, it would think the frames stops at that point and a new frame starts there, effectively splitting the original frame into two smaller frames (and losing one byte, namely the one that underwent the bit error).

---

5. Sharing a medium

1 pt (a) F.

Efficiency will increase as a node can transmit many packets with guaranteed no collisions. But when every node transmits 5 packets in a row, rather than one, it will take longer for another node to get a chance to transmit, so waiting time increases.

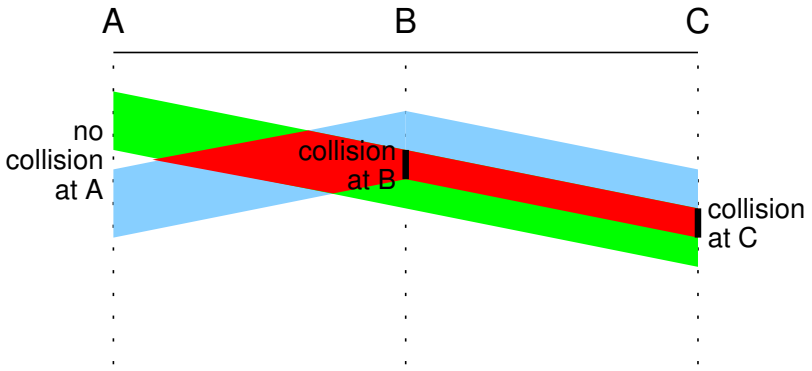
1 pt (b) E.

Since the nodes have something to transmit all the time, TDMA is best: just assign each node periodically a fixed amount of time. Then there is no overhead due to polling, no overhead due to collisions, and no wasted time due to no node transmitting (because every node always has data ready for transmission, so it will always use its allotted time).

1 pt (c) A.

ALOHA, basically transmitting randomly, is the best in this case.
CSMA variants can't work well since the nodes can't hear each other.
Polling would be wasteful since the nodes would have to continuously waste battery power listening to the master and sending back notifications when they are polled but don't have any data.
TDMA would force the nodes to have a good clock running all the time to know when their slot is, and cause unnecessarily long delays since the number of nodes is large so it takes a long time before it's your turn again.
Of course, ALOHA suffers from low efficiency, but since the nodes do not often have data to transmit, this can be accepted here.

2 pt (d)



There's a signal (green) starting at A and travelling to the right (not to the left, as it is given that A is at the end of the cable).
Furthermore, there's a signal (blue) transmitted by B, flowing both to the left and the right.
The overlap is indicated in red: at these places and times, a collision occurs.

Some points of attention:

- B must start transmitting before it receives the signal from A (otherwise, carrier sense would prevent it from transmitting).
- At B and C, the signals should overlap: collision.
- At A, there should be no overlap, as it is given that A does not detect a collision.
- The slope of the lines should be the same. This slope represents the propagation speed of the signal, as it is the ratio of (horizontally) the distance traveled by (vertically) the time needed for that.