Network Systems (201300179/201400431), Test 1

February 13, 2014, 13:45–15:15

Brief answers

---

1. A transatlantic link and reliable data transfer

4 pt (a)

transmission delay = 40000 / $20 \cdot 10^9 = 2 \cdot 10^{-6} = 2\ \mu$s.
propagation delay = $10000 \cdot 10^3 / 2 \cdot 10^8 = 0.05$ s.

Surprisingly many students made minor calculation errors in this and subsequent questions, in particular, being off by powers of 10. Even though this didn't cost you many points, it's still a pity; also, your future employer will not be so easy on you for being a factor of 10 off...

4 pt (b)

That's 200 000 packets. Each of them takes 2μs to send, followed by $2 \cdot 0.05$ seconds of waiting for the ack. That makes the total time $200\,000 \cdot 0.100002 = 20\,000.4$ s. (Strictly speaking, we don't need to wait for that last ack, so 20 000.35s is also correct.)

3 pt (c)

The number of bits we can send during 1 RTT is $0.1 \cdot 20 \cdot 10^9 = 2 \cdot 10^9$, which is 50 000 packets. So we need to be able to send 50 000 packets without seeing any ACKs, so SWS must at least be 50 000.
(Strictly speaking, we need 1 more, because the ack won't be sent until after a packet has been received entirely.)

3 pt (d)

No waiting for ACKs is needed, so the file can be sent at full speed, taking $8 \cdot 10^9 / 20 \cdot 10^9 = 0.4$ s to transmit, followed by the one-way propagation delay for the last bit, making the total 0.45 s.

Surprisingly many students tried to calculate this based on how many RTTs are needed using the window size calculated previously; however, that's more complicated and not needed.

3 pt (e)

No, because the sender will never send more than SWS packets beyond the last ACK it has received, so the receiver will also never see more than SWS packets beyond its last ack, so choosing RWS bigger makes no difference.

4 pt (f)

HTTP 1.0 needs 4 RTT (setup, fetch page, setup, fetch image); HTTP 1.1 needs 3 RTT (setup, fetch page, fetch image), so 1 RTT = 0.1 s, is saved.

4 pt (g)

Frame 0 is sent ok; ack 0 is lost. Frame 0 is sent again. Receiver expects frame 1, but with RWS=2 is also willing to receive new frame 0. It will accept the retransmission of frame 0 as if it were a new one.

There are many other possibilities; however, scenarios in which the sending node already sends frame 1 before having seen an acknowledgement for frame 0 are not correct, because SWS=1.

---

2. Information theory and error-correcting codes

4 pt (a)

0.784 bits.

Suprisingly many students had trouble converting 0.9 % into a fraction of 0.009, preferring 0.09 instead.

4 pt (b)

0 for Harmless, 10 for Suspicious, 110 for Terroristic, and 111 for Syria; average is 1.21 bits

4 pt (c)

The Shannon capacity of this channel is 988.59 bit/s. So as long as we stay below 988.59/0.784=1261 messages per second, the error rate can be made arbitrarily low.

4 pt (d)

Yes. In principle, when receiving a packet with wrong CRC, the receiver can search which valid message (including its CRC) differs least from the received one, just like with any error-detection code (although with practical error-correcting code, a more efficient algorithm is used). And because of the property that CRCs detect every error up to 2 bits, valid messages differ in at least 3 places (i.e., its Hamming distance is at least 3), so after a 1-bit error, the message is still closer to the original one than to any other valid message, so correction is indeed possible without ambiguity.

This was, intentionally, a more difficult question.

---

3. Peer-to-peer applications

Consider a peer-to-peer system as follows. There is one server, denoted S_F, that makes available a large file $\mathcal{F}$ of size F bytes to n peers. The upload rate from server S_F is u_F bytes per second. The download rate of each of the n clients is d bytes per second; the upload rate of each peer is u bytes per second.

For this scenario, it is known that the overall download time D_F for the file $\mathcal{F}$ is lower bounded as follows:

$$D_{\mathcal{F}} \geq \max\left\{\frac{F}{u_F}, \frac{F}{d}, \frac{F}{\frac{u_F}{n}+u}\right\}.$$

5 pt (a)

See book (note that the formula is simpler than in the book because the download and upload speeds are the same for all peers).

4 pt (b)

$$D_{\mathcal{F} \text{ and } \mathcal{G}} \geq \max\left\{ \frac{F}{u_F}, \frac{G}{u_G}, \frac{F+G}{d}, \frac{F+G}{\frac{u_F}{n} + \frac{u_G}{n} + u}, \frac{F}{\frac{u_F}{n} + u}, \frac{G}{\frac{u_G}{n} + u} \right\}$$

The first and second term are the times that each of the servers needs to upload its own file to the network (note that these uploads happen in parallel, so we don't add them, just take their maximum).

Third term is the time each client needs to download both files.

The fourth term is the time needed to send both files n times through the total available uplink bandwidth (which is $u_F + u_G + nu$).

The last two terms cover a rather extreme situation, namely if one file is so much bigger than the other, that even if all the client upload bandwidth is dedicated to the bigger file, still the uploader of the smallest file will finish his upload before the bigger file has reached all clients. (Note that he cannot contribute his upload bandwidth to uploading the *other* file.)

Any answer having the first 4 terms was worth a full score (since we kind of suggested 4 terms would be enough, and the last 2 terms are a rather extreme case). Those who gave the first 3 and the latter 2 terms, got 3 points.

3 pt (c)

Fault-tolerance, performance/scalability, goes against the idea of p2p where there is no "central authority"

End of this exam.