Homework 1

Due: 1 October 2015, 11:59pm

1 Sockets

You are designing a protocol and writing software for reliable message delivery. It is extremely important that when the sending software tells the user that a message is sent, the message must be stored safely at the receiver's machine.

a. You coworker recommends that you build the protocol on top of TCP, since it provides a reliable byte-stream, and acknowledgements are used at the protocol level to ensure that data have been received. He reasons that write() is a blocking operation, and that once it returns, you may presume the data has been received at the other end. Why is he wrong? Would it help if you waited until the *next* call to write() returned? If not, propose a way to know that the receiver has actually received the message.

b. Another coworker suggests that since your messages are short (about 20 bytes), you should just send them using UDP, and perform your own acknowledgments (also with UDP). Provide at least two upsides and two downsides of such a protocol.

c. How might a message be delivered twice by mistake using a simple protocol that just addresses the concerns in (a) and (b)? How would you fix that?

2 Errors

a. We learned that a 2D parity scheme can detect all 3-bit errors, but not all 4-bit errors. Assuming that all bitflips are equally likely, what is the probability that a 4 bit error is detected in the 8x8 scheme we saw in class? What about 16x16 (15x15 data bits, plus the parity bits)?

b. Can a 2D parity scheme detect all 5 bit errors? If not, show an error that can't be detected. If it can, prove it.

c. Suppose there is a code that adds three bits to every bit pair: a copy of the original two bits and a parity bit (computed off the original pair, not all four bits). For example, 01 becomes 01011, and 11 becomes 11110. How many bit errors can this code detect? How many bit errors can this code correct (*i.e.*, what is the largest k such that the code can correct all possible k-bit flips)? Justify your answer.

d. Suppose we want to transmit messages with a 4-bit CRC. Select a polynomial to use, and justify your choice. Then, encode the 12-bit message 1010001010100 with your polynomial. What bit-sequence would you send? Show that your scheme would (or would not) detect the error if the two middle bits of the original message were flipped.

3 Reliability

Suppose you have a network with 4 wireless links between A and B, and that every frame sent (either way) on these links has a uniform and independent chance of being dropped of 5%.

a. What is the chance that a packet will be successfully transmitted from A to B? (Assume no acknowledgments or retransmissions of any kind.)

b. Now, add acknowledgments at each link. If a packet is not acknowledged it is retransmitted, up to three times. What is the probability that a packet will be dropped at a single link (*i.e.*, that it will fail after three attempted retransmissions?)

c. Using the result from the calculation above, what is the chance that a packet will arrive at B, if all four links are using up to three retransmissions?

d. On average how many total link transmissions (data and ack frames) are send for each data packet from A to B, using this mechanism?

e. Suppose we hadn't added link layer acknowledgements, and simply had B send a network (or higher) layer acknowledgement back to A, using the original scheme of part (a). How many link layer frames would be sent per packet, on average? A will retransmit indefinitely (after appropriate timeouts), until it gets an acknowledgement from B.

f. When should link layers have reliability features like acknowledgements, despite the fact that TCP will build reliability on top anyway for those that care?