Homework 1

Due: Solution

1 Sockets

a. The write() system call only blocks until the data can be accepted by the kernel. It may still be buffered there, rather than sent, when write() returns. (It could also be "on the wire", but not yet received.) [3 pt]

No, it would not help. If the data from multiple write() operations can fit in the outgoing buffer, then the writes will return.

You should build explicit application level acknowledgements in the protocol. For example, the receiver might respond with "OK" after it has receieved an processed a message.

b. Upsides: 1) Data can arrive and be acknowledged in one round-trip time. No need for TCP handshake first. 2) Messages maintain their "packet-ness" when received as UDP messages. You don't need to frame the messages as you would in a TCP stream of multiple messages. [8 pt]

Downsides: 1) You must write your own reliability mechanism. 2) You must write your own congestion-control (and/or flow-control) mechanism. (And it should probably be "fair" to TCP streams.)

c. The receiver might crash after delivering the message, but before sending, "OK" causing the sender to retransmit later. Or, the sender might crash after sending the message, but before receiving the "OK", so it will retransmit when it restarts. [4 pt]

You could add sequence numbers to the messages. The receiver must remember what numbers it has seen. They should be commited to persistent storage, in case of a crash.

2 Errors

a. There are $\binom{64}{4} = 635376$ possible 4 bit errors. They are detected unless they occur in a "square". A square is completely described by two rows and two columns, so there are $\binom{8}{2}\binom{8}{2} = 784$ possible squares. Four bit errors will be detected with $1 - \frac{784}{635376} = 0.998766$ probability. [3 pt]

For 16x16, use $\binom{256}{4}$ and $\binom{16}{2}$. The probability of detecting a four bit error is 0.9999993.

b. Yes, it can. Recall that the parity scheme will detect any odd number of errors in a single row. Since five is odd, it's impossible to split the five errors up in such a way that all rows have an even number of the errors. So at least one row will have an odd number of errors, and be detected. [3 pt]

c. First we find the minimum Hamming distance among any two code words. If you look at the original words, 00, 01, 10, 11, there are four pairs with distance 1 (*e.g.*, 01→11) and two pairs with distance 2 (*e.g.*, 00 → 11). The pairs with distance 1 have different parity among the pairs, and as we are copying the original bits twice in the new code, the new distance will be [3 pt]

$(1 \times 2 + 1) = 3$. The two pairs with distance 2 have the same parity. Thus, the new distance will be $(2 \times 2 + 0) = 4$, and the minimum distance among any pairs is 3.

Now we can answer the question: with minimum HD $d = 3$, the code can detect any $d - 1 = 2$ bit errors (as you will not go from a valid code to another valid code by flipping 2 bits, and can correct any $\lfloor (d-1)/2 \rfloor = 1$ bit errors.

d. If you are sending a 4-bit CRC value, you need a degree-4 polynomial, which has 5 terms (upon division, it will produce remainder no longer than 4 bits.) There are three criteria for selecting a good polynomial. If the polynomial is of degree k, you want x^k and $x^0 = 1$ in order to detect all single bit errors. You can detect any odd number of errors if x^1 and x^0 are 1. These criteria admit: [3 pt]

$x^4 + x + 1 = 10011$
$x^4 + x^2 + x + 1 = 10111$
$x^4 + x^3 + x + 1 = 11011$
$x^4 + x^3 + x^2 + x + 1 = 11111$

Finally you also want to select an irreducible polynomial. Any polynomial with an even number of terms is divisible by $x + 1$.

$x^4 + x^2 + x + 1 = (x + 1)(x^3 + x^2 + 1)$
$x^4 + x^3 + x + 1 = (x + 1)(x^3 + 1)$

You should select 10011 or 11111. 10011 is the CRC-4-ITU international standard, so we'll use that. To find the extra bits, tack four zeros onto the message, and divide by our CRC polynomial. Dividing 10011 into 10100010101000000 yields 101.

```
             1011111010011
10011 | 10100010101000000
        10011
          111010101000000
          10011
           11100101000000
           10011
            1111101000000
            10011
             110001000000
             10011
              10111000000
              10011
                100000000
                10011
                   110000
                   10011
                    10110
                    10011
                      ---
                      101
```

So we send the message 10100010101000000 followed by 0101: 101000101010000000101.

This polynomial will detect any burst error less than length four, so flipping the middle bits would be detected. You could show that by division yielding a non-zero remainder.

3 Reliability in lower versus higher layers

a. The likelihood of transmission success for a single frame is always $1 - p$. Because each frame is independent, the probability that a packet will be succesfully transmitted at four links is $(1-p)^4 \approx \mathbf{0.814}$. [1 pt]

b. Let p_d be the probability of such an event. With three retransmissions, a transmission/acknowledgment pair (which we will call a handshake) will be attempted four times in total before ending. There are two interpretations of a dropped frame, which are both acceptable. These interpretations also change the answers for part (c). [2 pt]

(a) A frame is dropped when its data does not reach the receiver.

In this case, the probability of a drop is simply $p_d = p^4 = \mathbf{6.25 * 10^{-6}}$. The fate of acknowledgments is irrelevant, because each possibility suggests that the original data frame is dropped.

(b) A frame is dropped when the sender does not receive an acknowledgement of receipt.

If the likelihood that a handshake fails once is q, the likelihood that the frame will be dropped is q^4.

Now a handshake can fail either if the original frame is dropped or the frame has been received but the return acknowledgment is dropped. In other words, $q = p + p(1-p) = p(2-p)$. For $p = 0.05$, $q = 0.0975$. Therefore, the probability that a frame is dropped is $p_d = q^4 \approx \mathbf{9.0 * 10^{-5}}$.

c. The likelihood of success for a frame to be successfully delivered across a single link is $1 - p_d$. Because the likelihood for each frame to be delivered across each link is independent, the likelihood for a frame to be delivered across four links is [1 pt]

$$(1-p_d)^4 \approx \begin{cases} \mathbf{0.99998} & \text{case 2a} \\ \mathbf{0.9996} & \text{case 2b} \end{cases}$$

d. This question does **not** depend on the interpretation of "drop" in (b): frames continue to be sent until each link layer sender is assured of delivery through acknowledgment. Also, it is permissible to use some approximations for this and the next question. Any reasonable approximation will be accepted. [3 pt]

One example average answer:

We can reason that the total number of frames sent F is the number of frames required for succesful transmissions/acknowledgments (which is fixed at 8), plus the number of frames sent that were part of a failed handshake, which we call X. Then $\mathbf{E}[F] = 8 + \mathbf{E}[X]$.

Let m be the number of failed handshakes. Because the likelihood of failure is q, $\mathbf{E}[m] \approx 4q$, assuming that each link will only fail just once. Each failed handshake incurs a cost of

approximately 1.5 frames; one if the transmission fails, or two if the acknowledgment fails; these are almost equally likely. Because the likelihood of failure is (almost) independent of the number of frames dropped, we can write that $\mathbf{E}[X] \approx 1.5\mathbf{E}[m] = 6q = 0.58$. Thus, the average number of frames sent is approximately $\mathbf{E}[F] \approx \mathbf{8.58}$.

A full answer:

We first calculate the number of frames across a single link until an acknowledgment is succesfully returned. Let X_n be a random variable of the number of frames sent across a single link given n possible retransmissions. For X_0, we can write:

$$X_0 = \begin{cases} 2 & \text{success, likelihood } (1-p)^2 \\ 1 & \text{transmission fails, likelihood } p \\ 2 & \text{acknowledgement fails, likelihood } p(1-p) \end{cases}$$

Similarly, we can also write X_n as a recurrence relation:

$$X_n = \begin{cases} 2 & \text{success, likelihood } (1-p)^2 \\ 1 + X_{n-1} & \text{transmission fails, likelihood } p \\ 2 + X_{n-1} & \text{acknowledgement fails, likelihood } p(1-p) \end{cases}$$

Therefore, we can find the expectation for both X_0 and X_n.

$$\begin{aligned} \mathbf{E}[X_0] &= 2(1-p)^2 + p + 2p(1-p) \\ &= 2 - p \\ &= 1.95 \end{aligned}$$

Likewise,

$$\begin{aligned} \mathbf{E}[X_n] &= 2(1-p)^2 + (1 + \mathbf{E}[X_{n-1}])p + (2 + \mathbf{E}[X_{n-1}])p(1-p) \\ &= 2 - p + \mathbf{E}[X_{n-1}]p(2-p) \\ &= 2 - p + q\mathbf{E}[X_{n-1}] \end{aligned}$$

where $q = 0.0975$ is the likelihood (defined above) that a single handshake will fail. Thus, we can iteratively calculate $\mathbf{E}[X_1] \approx 2.14$, $\mathbf{E}[X_2] \approx 2.159$, $\mathbf{E}[X_3] \approx 2.1605$[1]

Failure at one link is very small (see answer to part (b)), so we can assume that the number of frames at each link is independent[2]. Thus each packet requires $4\mathbf{E}[X_3] = \mathbf{8.64}$ frames.

e. Let X_n be a random variable of the number of frames sent with n retransmissions allowed. Let r be the probability that both a single transmission and its acknowledgment succeed across multiple links. [3 pt]

We can reason about the expected value of the number of frames as follows: we can calculate the number of frames sent on this transmission attempt, and then if we fail with probability $1 - r$, we fall to the number of frames sent with $n - 1$ retransmissions allowed. Thus, we

[1] The recurrence converges exponentially quickly to $\mathbf{E}[X_\infty] = \frac{2-p}{1-q} \approx 2.16066$.

[2] More precisely, when we consider that a failure at a link stops the number of frames at successive links, the total expected value is $(\sum_{n=0}^{3}(1-q)^n)\mathbf{E}[X_3]$, but this correction is on the order of 0.01%.

can write $\mathbf{E}[X_n] = \mathbf{E}[X_0] + \mathbf{E}[X_{n-1}](1-r)$. At the limit where $n \to \infty$, we can solve for the expected number of frames with infinite number of retransmissions: $\mathbf{E}[X_\infty] = \frac{\mathbf{E}[X_0]}{r}$.

Now eight frames must be successfully sent for the transmission from A to B to complete; therefore $r = (1-p)^8 \approx 0.663$.

Also, the probability that the ith frame out of 8 in the series of transmissions is $p(1-p)^{i-1}$, where i frames were sent. Therefore, we can write[3]

$$\mathbf{E}[X_0] = \sum_{i=1}^{8} ip(1-p)^{i-1} + 8(1-p)^8 \approx 6.73$$

The total number of frames under this retransmission scheme is thus $\mathbf{E}[X_\infty] = \frac{\mathbf{E}[X_0]}{r} \approx \mathbf{10.15}$.

f. Often multiple link layers must be passed through for each application-layer packet: if a link layer is lossy enough so that the dropping the packet will incur the cost of a retransmission through other link layers, then it should implement reliability. Also, TCP and other protocols use end-level acknowledgments to monitor the reliability and congestion of a connection; if an acknowledgment is not returned when it could have been with more intermediate reliability, the connection state may deteriorate (this will be seen later.) [3 pt]

[3] This can be evaluated analytically: see arithmetico-geometric sum for details. The closed form is $1 + \frac{1-p}{p}(1-(1-p)^7)$.