University of Waterloo
ECE 358, Spring 2017
Pencil-n-Paper Assignment 3

Shiranka Miskin, Dhruv Lal, Sam Maier

Problem 1

Both Bob and Carol pick a 5th peer uniformly from the remaining 46 peers that aren't from their top 4. This means that every 30 seconds, there is a $\frac{1}{46} + \frac{1}{46} - \frac{1}{46^2} = \frac{91}{46^2}$ chance that Alice is selected (from the probability of two non mutually exclusive events occuring). Let t represent the number of times Alice is chosen across n trials. To determine how many trials are necessary for Alice to expect to receive chunk c, we must determine the lowest value of n such that $E[t] \geq 1$. We express $t = \sum_{i=1}^{n} Y_i$ where $Y_i = 1$ if Alice is chosen in the i^{th} trial and $Y_i = 0$ if she is not.

$$\begin{aligned}
E[t] &= E\left[\sum_{i=1}^{n} Y_i\right] \\
&= \sum_{i=1}^{n} E[Y_i] \\
&= \sum_{i=1}^{n} Pr\{Y_i = 1\} \\
&= \sum_{i=1}^{n} \frac{91}{46^2} \\
&= \frac{91n}{46^2} \\
\frac{91n}{46^2} &\geq 1 \\
n &\geq 23.253
\end{aligned}$$

Therefore Alice can expect to wait $n \cdot 30 = 24 \cdot 30 = 720$ seconds before she receives chunk c

Problem 2

UDP's checksum algorithm produces its checksum by summing up all 16 bit segments of the packet (including the header) under a one's compliment addition. For an error to pass by undetected, it must be the case that after some number of bits in those segments have been changed, that one's compliment sum results in the same value it had previously.

(a)

To determine if a 1-bit error can pass by undetected, we must determine if $x+y=x+z, z=y \pm 2^i, 0 \leq i \leq 15$ is possible under a 16-bit one's compliment addition.

If neither the addition of $x+y$ or $x+z$ produce a carry, it is impossible for $x+y=x+z$ to be the case, as by flipping one bit in y to create z, we are either adding or subtracting $2^i, 0 \leq i \leq 15$ from the result. The one's compliment does not factor in in this situation, and we can use the simple integer arithmetic to deduce that $x+y=x+z$ implies that y must equal z.

If both the addition of $x+y$ and $x+z$ produce a carry, this 16-bit one's compliment is equivalent to $x+y-2^{16}+1=x+z-2^{16}+1$ (moving the carry bit at the 16th position to the first position). Simple integer arithmetic now applies, and we can subtract the $-2^{16}+1$ from both sides to get $x+y=x+z$ in the same situation as the first case. We can therefore conclude that in this case $x+y=x+z$ can only be the case if $y=z$, which cannot happen under a bit flip.

If either the addition of $x+y$ or $x+z$ produces a carry bit while the other does not, then we can think of it as an addition of $x+y=x+z-2^{16}+1$ where both $x+y$ and $x+z$ alone do not produce carry bits, and $z=y \pm 2^i, 0 \leq i \leq 15$. If $x+z$ produces a carry while $x+y$ does not, then $x+z > x+y$, therefore our previous $\pm$ restriction becomes $z=y+2^i, 0 \leq i \leq 15$.

$$\begin{aligned} x+y &= x+(y+2^i)-2^{16}+1, 0 \leq i \leq 15 \\ y &= y+2^i-2^{16}+1, 0 \leq i \leq 15 \\ 2^{16}-2^i &= 1, 0 \leq i \leq 15 \end{aligned}$$

The smallest value of $2^{16}-2^i, 0 \leq i \leq 15$ is 32728, therefore this case is also impossible.

Since every case where a difference would proceed undetected has been proven to be impossible, we can conclude that UDP's checksum algorithm is able to detect any 1-bit error.

(b)

This is false. Consider the addition of two 16 bit segments:

```
0000000000000100
0000000000000001 +
-------------------
0000000000000101
      \/ (1's compliment)
1111111111111010
```

If the last bit in the first segment and the last bit in the second segment both flip, we will have:

```
0000000000000101
0000000000000000 +
------------------
0000000000000101
     \/ (1's compliment)
1111111111111010
```

The checksum will result in the same value, therefore the checksum algorithm would not detect an error.

Problem 3

Consider a network where every packet that is sent is corrupted in a detectable manner. The receiver would detect the error via the checksum, and constantly reply to the sender with NAKs. The sender would therefore be locked in its "Wait for ACK or NAK 0" state and never progress to the next packet. This would also happen if even though all the data packets are sent successfully, every ACK from the receiver happens to get corrupted. The sender would constantly resend the first packet and never make progress.

We can assert that the progress property is held if we assume that the corruption does not follow a regular pattern such as always occuring or occuring for every single ACK. Under zero corruption it is trivially clear that progress is always made. The two cases of corruption are corruption of the data packets and corruption of the ACK/NAK responses. If either the data packet or the response is corrupted, the sender resends the data. Due to our assumptions, we can claim that eventually it will be the case that both the data transfer as well as the response will successfully be sent. The sender will proceed to its "Wait for call 1 from above" state, and then by the same logic as previously we can assert that the next packet to be sent will also eventually be received by the receiver and the sender proceeds back to its original "Wait for call 0 from above" state. This process repeats until the entire contents of the data are sent and `deliver_data` completes successfully at the receiver.