UNIVERSITY COLLEGE LONDON
DEPARTMENT OF COMPUTER SCIENCE
Dr Kyle Jamieson and Dr Brad Karp

3035/GZ01 Networked Systems
Individual Coursework 1
Distributed: 27th October; Due: 11:00 AM 3rd November 2011

Answer all of the following problems. Either handwritten or typeset solutions are fine, but if you write by hand, please ensure your answers are legible. Please show all work! We cannot award credit for correct answers if their complete derivation isn't shown. Also state clearly any assumptions you make while solving a problem. This coursework is worth 8% of the total marks for 3035/GZ01.

Please monitor the course mailing list, `gz01@cs.ucl.ac.uk`, during the period between now and the due date for the coursework. Any announcements (e.g., helpful tips on how to work around unexpected problems encountered by others) will be sent to the lists.

Hand-in instructions: Hand your solutions in to the instructor at the beginning of lecture on the due date.

Collaboration: Collaboration is not permitted on this problem set; you may not discuss the problems or their solutions with anyone else (whether or not the other person is taking the class), apart from the instructors. All work you submit must be your own. You may of course refer to all lecture notes and readings, and any other materials you wish (textbooks, papers, or material found on the Internet).

1. **Linear codes**

   Dee Coder, a former 3035/GZ01 student, has graduated and is working for a local Internet Service Provider in town. As customers are complaining of poor line quality, one day her manager asks her to implement a (6, 3) block code defined by the following parity bit computations:

$$\begin{aligned} p_0 &= d_1 \oplus d_2 \\ p_1 &= d_0 \oplus d_2 \\ p_2 &= d_0 \oplus d_1 \end{aligned}$$

   The code takes three data bits d_0, d_1, and d_2, and computes three parity check bits p_0, p_1, and p_2.

   a. What is the rate of this code?

   [3 marks]

   b. What is $d_{\min}$ for this code? (Show your reasoning.)

   [3 marks]

c. First, Dee wants to use this code as an error **detecting** code.

   i. Up to how many bit errors per codeword can this code always detect? Give a short yet precise justification.

   [2 marks]

   ii. Give an example (*i.e.*, specify which bits are errored) of a four-bit error that a receiver could not detect, and explain why the receiver couldn't detect those errors.

   [3 marks]

d. Next, Dee tries using the code as an error **correcting** code.

   i. Up to how many bit errors per codeword can this code always correct? Give a short yet precise justification.

   [2 marks]

   ii. Give an example (*i.e.*, specify which bits are errored) of a three-bit error that a receiver could not correct, and explain why the receiver couldn't correct those errors.

   [2 marks]

Louis Reasoner looks at the code Dee is working with and is skeptical. "You should be using a *repetition code*," he says. "Just set

$$\begin{aligned} p_0 &= d_0 \\ p_1 &= d_1 \\ p_2 &= d_2 \end{aligned}$$

so that the parity bits just repeat the data. That will let you correct any single-bit error just by looking at the other bit.

e. Is Louis correct? Give a concise reason that proves his claim true or false.

[3 marks]

Dee implements her (6, 3) code as described above, but it doesn't help correct errors on customers' DSL lines. By sending (known) test data patterns and looking at received bits, Dee determines that her customers DSL lines are generating burst errors three bits long from time to time (which, as you've shown above, can't be corrected). After taking several days' worth of measurements, she observes that no contiguous 18 bits have more than one error burst, and there are no error bursts longer than three.

f. Can you improve Dee's error control system to allow her (6, 3) code to fix bit errors as described above? If so, explain how. If not, argue why not.

[4 marks]

2. **Ethernet**

a. For the first part of this question, suppose we are operating a 10 Mbps shared-bus Ethernet over a coaxial cable with signal propagation speed of 1.8×10^8 m/s. Furthermore, we would like to allow a very large number of users to share the Ethernet link efficiently.

i. If we set our contention window slot size $T = 10\mu s$, what is the maximum length of coaxial cable that we can use?

[2 marks]

ii. As the number of users increases to infinity, what minimum packet length ℓ should we use to ensure an Ethernet efficiency of at least 95% ?

[3 marks]

iii. Is the packet size in 2.a.ii large enough to guarantee collision detection at all senders?

[2 marks]

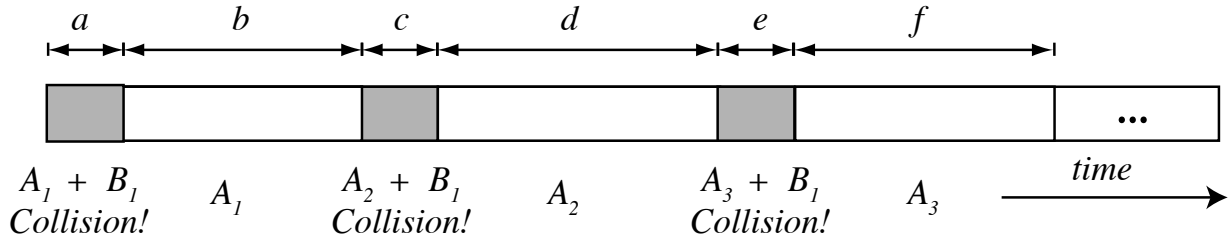

FIGURE 1: Frame transmission sequence for two senders A and B.

b. The second part of this question deals with an observed phenomenon called the Ethernet capture effect. Suppose we have two users A and B attempting to send frames on the shared Ethernet link. Both A and B have a large queue of frames to send, with A's frames numbered $\{A_1, A_2, \cdots, A_n\}$ and B's frames $\{B_1, B_2, \cdots, B_n\}$. A and B's frames will experience collisions, forcing their senders to contend for the shared medium. In this question, we will calculate how likely it is that A's packets always win the contention races. Figure 1 shows the scenario we will be using for the remaining questions.

   i. Let's start with the case where A sends A_1 and B send B_1 at the beginning of time interval a (shown in the figure). After the inevitable collision, both A and B set their contention window sizes to $CW = [0, \cdots, 2^m - 1]$ with $m = 1$ and hence randomly pick either a zero or one backoff slot. Suppose A wins the contention race, and backs off with delay $0 \times T$ while B backs off with $1 \times T$, where T is the slot size. As a result, A transmits A_1 at the beginning of time interval b. Because packet sizes are much larger than T at this stage, B becomes ready to transmit during A's transmission. But B carrier senses the link and defers because the medium is currently busy with A's transmission. After A completes, B senses an idle medium at the beginning of time interval c and begins to send B_1, while A attempts to send its next frame A_2 resulting in a new collision.

   What are the contention window sizes of senders A and B after the collision in c, and why?

   [2 marks]

   ii. What is the probability that A wins the contention race after the collision in c, and manages to transmit its frame A_2 in d? In other words, what is the probability that A's first choice of backoff time $k_A \times T$ is less than B's?

   [3 marks]

   iii. Suppose A wins the second backoff race after time interval c. It therefore transmits its frame A_2 at the beginning of d, but once it tries to transmit A_3 in e, the frame collides with B_1. Calculate the probability that A wins the third contention race immediately after the collision.

   [3 marks]

iv. Given that A has won the last $m-1$ contention races, show that the probability A will win the mth contention race is equal to $1-\frac{3}{2^{m+1}}$. What happens to this probability as m grows bigger?

[4 marks]

v. Through the Ethernet capture effect described above, the more contention races A wins the more likely it is to win future ones, and preclude B from sending its frame B_1. What eventually happens to frame B_1 and B's contention window?

[2 marks]

vi. Consider a scenario similar to the one described in 2.b.ii, but this time three different users A, B, and C are transmitting on the medium. As before, assume that user A has managed to transmit it's frame A_1 during time interval b, and assume a collision between A, B, and C's frames in interval c. What is the probability that A will win the contention race at the beginning of time interval d?

[2 marks]

vii. Referring to your answer in 2.b.vi and how Ethernet works in general, why isn't capture a problem for a shared-medium Ethernet connecting large numbers of active users? Hint: also think about the probability of other users winning the contention as their numbers increase.

[2 marks]

3. **Wireless 802.11**

a. Recall that in the 802.11 MAC protocol, a node transmits a data frame only after sensing an idle medium for a *DIFS* + *backoff* time. Upon successful receipt of the frame, the receiver sends an *ACK* frame back to the transmitter after waiting for a *SIFS* time. Suppose station STA1 has 200 data frames to send to STA2 and each frame is 1500 bytes in size (total size of the frame, including all the headers). The data rate chosen by STA1 to transmit is 54 Mbit/s and it does not change. *ACK* frames and any other protocol control data are sent at 6 Mbit/s. The probability that a data frame is received in error is 1%, and protocol control frames are always successful. Assume that the parameters used in the protocol are as follows: $DIFS = 34\mu s$, $SIFS = 16\mu s$, $ACK_{timeout} = 48\mu s$, $ACK_{size} = 14$ bytes, and there is no back-off time ($backoff = 0$)

i. How many frames, on average, does STA1 need to transmit in order to deliver the 200 frames to STA2?

[1 mark]

ii. Excluding all the overheads, how much time do these transmissions alone take over the air?

[1 mark]

iii. How many transmissions timeout waiting for an *ACK* and how long does this take alone, on average?

[1 mark]

iv. How many *DIFS* intervals are required to deliver all 200 frames, and how long does this take alone, on average?

[1 mark]

v. How many *SIFS* intervals are required to deliver all 200 frames, and how long does this take alone, on average?

[2 marks]

vi. Showing your work, calculate the total time it takes, on average, to successfully deliver all 200 frames from STA1 to STA2.

[2 marks]

vii. What is the protocol overhead (as percentage of the raw link speed)?

[2 marks]

b. 802.11n, the new IEEE standard, uses the so called Block Acknowledgement protocol to reduce the overhead of the Stop-and-Wait protocol in previous standards. A simplified version is as follows.

After sensing the medium idle for a $DIFS + backoff$ time, STA1 starts transmitting frames one after another with a batch size of 64 frames, waiting for a *SIFS* time between each frame. After it transmits all the frames, it waits for a *SIFS* time before transmitting a *BlockAckRequest* frame (26 bytes). Upon receipt of *BlockAckRequest* STA2 sends back a *BlockAck* frame (158 bytes) after waiting for a *SIFS* time. The *BlockAck* contains IDs of all the frames STA2 successfully received. STA1 removes all acked frames from its queue and re-transmits unsuccessful frames by repeating the protocol. Using the same assumptions as in 3.a answer the following questions.

i. Explaining your answer, what is the required number of batches to successfully deliver all the frames to STA2, on average?

[2 marks]

ii. How many *SIFS* intervals, both STA1 and STA2 wait, on average? Explain why?

[2 marks]

iii. What is the the number of *DIFS* intervals, on average, needed? Explain why?

[1 mark]

iv. Calculate the total time it takes, on average, to successfully deliver all 200 frames from STA1 to STA2

[2 marks]

v. Does the Block Ack protocol transmit data faster then the Stop-and-Wait protocol? Did the protocol overhead reduce or increase, and by how much?

[2 marks]

vi. Could the Block Ack protocol perform better in the given situation? If yes, show how could it be improved and give any tradeoffs involved. If no, explain why?

[2 marks]

c. Unlike CSMA/CD used in Ethernet, the 802.11 protocol uses CSMA/CA (Collision Avoidance). Give two reasons why 802.11 uses collision avoidance instead of collision detection?

[2 marks]

d. In wireless LANs, nodes can interfere with each other even when they are separated by a distance greater then their transmission range. Using a picture (sketch) explain why this happens and how 802.11 can eventually deliver data across the link despite this happening?

[2 marks]

4. **Reordering in rdt v3.0**

   Reliable Transfers plc employee Don U Lucit has just studied `rdt3.0`, the stop-and-wait reliable transfer protocol with a one-bit sequence number described in lecture. Don thinks that rdt3.0 would function even when the network reorders packets. Is he right? If so, argue informally why. If not, provide a counterexample timeline in the style of the timelines presented in lecture, showing sender and receiver states, and packet contents.

   [10 marks]

END OF PAPER