BLG337E: Principles of Computer Communications

Homework #2

Due Date: 20.11.2016, 23:00

You wrote a program to simulate line coding in HW#1. In this homework, you are asked to add two functions of data link layer to your code: (1) framing and (2) error checking. As in HW#1, we assume that a single sender-receiver pair is connected via a point-to-point link (see the first figure in HW #1).

In this homework, frames consist of two bytes:

- The first byte is used for frame sequence numbers. Each frame carries a sequence number which uniquely identifies the frame. The sequence numbers start from 0 and increase by 1 whenever a new frame is constructed. After the sequence number reaches 255, it wraps around and starts again from 0.
- The second byte includes 7 bits of user data followed by an **even parity** bit.

Your program will be executed as follows:

program.exe data_file_name BER

A sample program flow is given below as a pseudocode:

---

1. Open the ***data_file*** whose name is given by the command line argument ***data_file_name*** and contains the sender side digital data as a sequence of '0' and '1' characters to represent 0 and 1 bits, respectively (e.g., 0110101 … 000111).

2. Open the ***sent_frames_file*** whose name is constructed as **your student_id** followed by the postfix "**_sent_frames.txt**". For example, if your student id is **12345**, then the name of the ***sent_frames_file*** is "**12345_sent_frames.txt**".

3. Open the ***link_file*** whose name is constructed as **your student id** followed by the **line coding scheme** used (as in the HW#1). In this homework, you **have to** use **NRZ-L** line coding. Thus, if your student id is 12345, then the name of the ***link_file*** is "**12345_NRZ_L.txt**".

4. Read 7 bits from the ***data_file***. You can assume that the ***data_file*** contains **exactly 7*N** data bits where **N** can be any positive number (i.e., please don't assume any maximum size for the user data in your code!).

5. Assign next sequence number for the frame.

6. Calculate the even parity bit over **both** the sequence number field and user data.

7. Fill in the frame fields and construct the frame.

8. Write the frame into ***sent_frames_file*** as follows:

| 00000000 | 01100101 |
| 00000001 | 11010110 |
.....
.....
.....

- The first line contains a frame with sequence number 0 (8 ‘0’ bits) followed by 7 bits of user data (‘0110010’) and the even parity bit (‘1’)
- The second line contains a frame with sequence number 1 (7 ‘0’ bits followed by a ‘1’ bit), 7 bits of user data (‘1101011’) and the even parity bit (‘0’). Note that, the parity bit is calculated **not only taking into account the user data but also the sequence number field**.

9. For each bit of the frame do the following:

9.1. Based on the given **bit error rate (BER)**, decide whether the bit will be corrupted or not.

9.2. If the bit will be corrupted, then corrupt it (invert ‘1’ to ‘0’ or ‘0’ to ‘1’).

10. Convert the (possibly corrupted) frame into a **digital signal** by using **NRZ-L** line coding scheme (as you did in HW#1).

11. Put the **digital signal** into the the ***link_file*** (as you did in HW#1).

12. Go to **Step 4** if there are any more data.

13. Close ***data_file*** and ***sent_frames_file***.

---

14. Open the ***receiver_file*** which will store the receiver side digital data as a sequence of ‘0’ and ’1’ characters (e.g., 0110101 … 000111). The file name is constructed as **your student_id** followed by the postfix “**_rcv_data.txt**”.For example, if your student id is 12345, then the name of the ***receiver_file*** is “**12345_rcv_data.txt**”.

15. Open the ***rcvd_frames_file*** whose name is constructed as **your student_id** followed by the postfix “**_rcvd_frames.txt**”. For example, if your student id is **12345**, then the name of the **rcvd_*frames_file*** is “**12345_rcvd_frames.txt**”.

16. Read **a frame** from the ***link_file*** and convert the signals back into the digital data.

17. Check the parity bit of the frame. If the parity bit is correct, then write the **7-bit user data portion** of the frame to the **receiver_file**. Otherwise just **ignore** writing the user data to the ***receiver_file*** (frame is corrupted).

18. Write the received frame into ***rcvd_frames_file***:

18.1. Use the format described in **Step 8** to write the received frame.

18.2. If the frame is **corrupted**, then add to the end of the line the character sequence “ | PE | ” to mark that there is a parity error.

19. Go to **Step 16** if there are any more frames.

20. Close the ***link_file***, ***receiver_file***, and ***rcvd_frames_file***.

<u>You have to submit:</u>

1. The **source file** which is written in **c** programming language and named as **“student_id.c**”. The source file **MUST BE** compilable and runnable in **Linux**. Please write your development environment at the start of the source file as well as your name, surname and student ID, of course within comments!
2. **A PDF report** which is named as “**student_id.pdf**”.