Data Fragmentation Assignment

Deliver to: noha_gamal@mentor.com

Details:

You are responsible for testing a new feature used to send messages between server and client(s).

Messages can vary in length between 64 bytes to 16,000 bytes.

The user can specify the width of the channel between a client and a server (e.g. channel width = 100 bytes).

If the length of message to be sent is larger than the specified channel width (e.g. message length = 320 bytes), the message will be divided into fragments and will be transmitted one fragment at a time.

Note that each message must have a header (length = 8 bytes) and a footer (length = 4 bytes). Same for fragments. Then the payload length of each fragment will be as follows:

100 (channel width) – 4 (footer) – 8 (header) = 88 bytes

So message will be transmitted as follows:

- First fragment:

Header	Payload (88 bytes)	Footer

- Second fragment:

Header	Payload (88 bytes)	Footer

- Third fragment:

Header	Payload (88 bytes)	Footer

- Forth fragment:

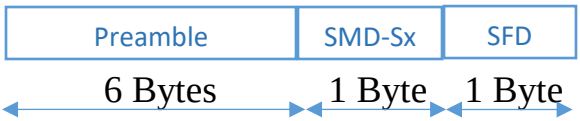
Header	Payload (56 bytes)	Footer

Note that both header and footer lengths are included in the channel width (e.g. 100 bytes).

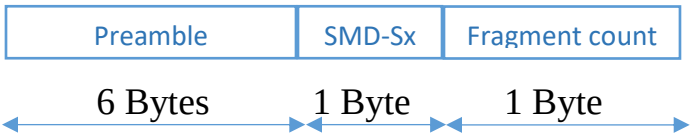
If the length of the message/fragment to be sent is less than the channel width, the message will be padded with zeros.

Header is 8 bytes formatted as follows:

- For
  - non-fragmented message (message with total length <= channel)
  - first-fragment of fragmented massage (message with total length > channel width)



- For other fragments



Note the following:

  - Preamble is 6 bytes of 0x07 byte value: 0x070707070707
  - SFD value is 0xD5
  - SMD-Sx is the ID of message, It is 2-bits length; so it takes values from 0 → 3 as follows.
    - Message 1, SMD-Sx = 0x00
    - Message 2, SMD-Sx = 0x01
    - Message 3, SMD-Sx = 0x10
    - Message 4, SMD-Sx = 0x11
    - Message 5, SMD-Sx = 0x00
    - And so on
  - Fragment count field is fragment ID (excluding first fragment) and It is also 2-bits length; so it takes values from 0 → 3 as follows.
    - Fragment 1, Fragment count = 0x00
    - Fragment 2, Fragment count = 0x01
    - Fragment 3, Fragment count = 0x10
    - Fragment 4, Fragment count = 0x11

    - Fragment 5, Fragment count = 0x00
    - And so on

Payload will be generated as follows:

- Stream of bytes with value 0xFF

Footer is 4 bytes, generated as follows:

- For non-fragmented massage or last fragment of fragmented massage , footer will be 0xAABBCCDD
- For other fragments, footer will be 0x11223344

Expected Deliverables:

1. An automated runnable code in C++ that applies the above algorithm
    a. Program inputs:
        i. Number of messages to be sent
        ii. Message length
        iii. Channel width
    b. Program output:
        i. Log in a file the messages to be streamed in the following format: *Message ID, Fragment ID, Message body*

        Note the following samples:

        1. Sample output for messages of length 6 bytes and channel width 20 bytes:
           1, 1, 07 07 07 07 07 07 00 D5 FF FF FF FF FF FF AA BB CC DD 00 00
           2, 1, 07 07 07 07 07 07 01 D5 FF FF FF FF FF FF AA BB CC DD 00 00
           and so on till number of messages achieved. Note that it's non-fragmented messages, fragment ID of non-fragmented message will be 1
        2. Sample output for messages of length 5 bytes and channel width 15 bytes:
           1, 1, 07 07 07 07 07 07 00 D5 FF FF FF 11 22 33 44
           1, 2, 07 07 07 07 07 07 00 00 FF FF AA BB CC DD 00
           2, 1, 07 07 07 07 07 07 01 D5 FF FF FF 11 22 33 44
           2, 2, 07 07 07 07 07 07 01 00 FF FF AA BB CC DD 00
           And so on till number of messages achieved. Note that it's fragmented messages
2. A document explaining your test design and how your code works.
3. The code should be runnable on Linux OS.
4. Make file and steps to run the program
5. Sample file for the output

Evaluation Criteria:

- Code structure, maintainability, coverage and usability.

Guidelines:

- Deadline: 3 days.