CSE123PROJECT1

Lijun Chen
A53071897

My frame structure is:

```
#define MAX_FRAME_SIZE 64
#define FRAME_PAYLOAD_SIZE 56
```

1. In common.h

```
struct Frame_t
{
   char recv_id[2];
   char send_id[2];
   char seqNum[1];
   char inAddition[2];
   char data[FRAME_PAYLOAD_SIZE];
   char crc[1];
};
```

The frame size is 64 bytes in which the payload size is 56 bytes. The most significant two bytes in frame is receiver id and then is sender id. Following them is one byte storing sequence number. And then is two bytes for additional functionality(I set it 1 when the frame is longer than 64 bytes, because in first implementation I concatenate them in receiver, but never use it since that the requirement says just to output them separately). The next 56 bytes are data and the last 1 byte I use to record crc-8.

My sender structure is:

```
struct Sender_t
{
   pthread_mutex_t buffer_mutex;
   pthread_cond_t buffer_cv;
   LLnode * input_cmdlist_head;
   LLnode * input_framelist_head;

   int send_id;
   uint8_t seqNum;
   uint8_t LAR;
   uint8_t LFS;
   uint8_t SWS;
   char* cached_FrameArray[8];
   int isCached_Frame[8];
   struct timeval lastSendTime_Frame[8];
   uint8_t seqQue[8];
};
```

Send_id is for recording the id for this sender. SeqNum is used to generate the sequence number from zero. Then three bytes to record LAR, LFS and SWS. cached_FrameArray is for buffering the frame that is sent but not received ack yet. isCached_Frame is for flaging the buffer to show if it is buffered. lastSendTime_Frame is for storing the timeout time for each frame buffered. seqQue is for recording the sequence number for each buffered frame.

My receiver structure is:

```
struct Receiver_t
{
   pthread_mutex_t buffer_mutex;
    pthread_cond_t buffer_cv;
    LLnode * input_framelist_head;

    int recv_id;
    uint8_t LAF;
    uint8_t LFR;
    uint8_t RWS;
    int isReceived_Frame[8];
    char* cached_FrameArray[8];
    char bufferedMessage[10000];
    int pendingMessage;
};
```

Recv_id is for recording the id for this receiver. Then three bytes to record LAF, LFR and RWS. isReceived_Frame is to show if in this location stored received frame. cached_FrameArray is to buffer received frame. bufferedMessage and pendingMessage are used for concatenating long message and not used.

2. In util.c and util.h, I implement the function to convert frame to char and to convert char to frame. Also, I implement the crc8Caculate function to calculate the crc in frame.

3. In sender.c
In void handle_input_cmds(Sender * sender, LLnode ** outgoing_frames_head_ptr), firstly, check whether the sending frame is in window. Then, if frame is longer than 64 bytes, I cut it and return remaining frame to cmdlist for next use. After doing this, I build a frame and transfer cmd data to frame. Then, I caculate the crc for frame. Next step, I caculate timeout time for each sending frame and buffer it. At last, I put it in outgoing_frames_head_ptr and send it and make LFS plus one.

In void handle_incoming_acks(Sender * sender, LLnode ** outgoing_frames_head_ptr), firstly, I calculate crc to make sure the data is correct. Then if I can find this frame in my buffered sent frame, I will release the buffer and make LAR plus one.

In struct timeval * sender_get_next_expiring_timeval(Sender * sender), each time when the sender thread wake up, I caculate the timeout time in my buffered frame array to get the time for next wake up.

In void handle_timedout_frames(Sender * sender,LLnode **outgoing_frames_head_ptr), I check if in the buffered frame array there is any frame that is timeout. If yes, then I resend it.

4. In receiver.c
void handle_incoming_msgs(Receiver * receiver, LLnode **outgoing_frames_head_ptr), I firstly check if the received message is corrupted by caculating crc. If it is correct, I check if it is duplicated sent by senders.If it's duplicate, then I just resend the ack to senders. If it is new coming message I buffered this frame and update LFR and LAF.

5. In testMy.sh
I send 300 packets (with corrupt probability of 20% and drop rate of 20%) and expecting receiver to print them out in order(longer sleep time at last for handle time out frames). It passed.