Notes for Discussion 3

1. The struct timeval object in C

```
Struct timeval{
    time_t tv_sec; //sec
    suseconds_t tv_usec; //microsec
}
```

When sending a frame, get the time and add 0.1s to it. The result records when the frame times out. Put this information into the sender queue along with the frame.

```
void calculate_timeout(struct timeval * timeout){
    gettimeofday(timeout,NULL);//use this to get the current time
    timeout->tv_usec+=100000;//0.1s
    if (timeout->tv_usec>=1000000){
        timeout->tv_usec-=1000000;//1s
        timeout->tv_sec+=1;
    }
}
```

Use time_val_usecdiff (already defined in util.c) to compute the difference for two timeval objects.

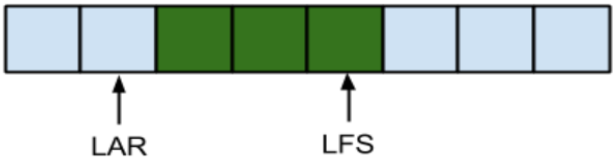
2. Sequence/ack number wrap around

The sequence/ack number in project 1 is 8 bits (unsigned char). When its value reaches 255, it should wrap back around to 0.

For sender:

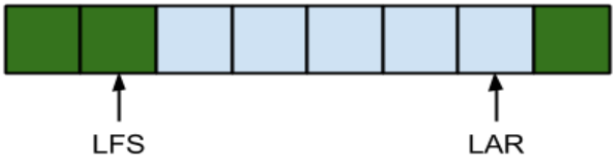
CASE 1: Usual Case
LAR <= LFS

LAR <= LFS && seqNo > LAR && seqNo <= LFS



CASE 2: Sequence Number Wrap Around
LAR > LFS

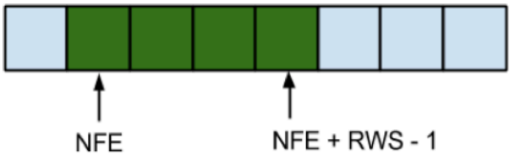
LAR > LFS && (seqNo > LAR || seqNo <= LFS)



For receiver:

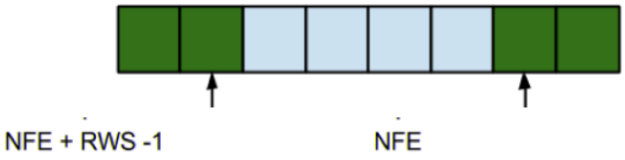
CASE 1: Usual Case
NFE + RWS - 1 >= NFE

NFE + RWS - 1 >= NFE && seqNo >= NFE && seqNo <= NFE + RWS - 1



CASE 2: Sequence Number Wrap Around
NFE + RWS - 1 < NFE

NFE + RWS - 1 < NFE && (seqNo >= NFE || seqNo <= NFE + RWS - 1)



3. Framing

If the size of the input message is greater than that of a frame payload, you should split the message into multiple pieces.

A skeleton code to do framing:

```
void ll_split_head (LLnode ** head_ptr, int payload_size){
    if (head_ptr == NULL || *head_ptr == NULL){
      return;
    }
    //get the message from the head of the linked list
    LLnode* head = *head_ptr;
    Cmd* head_cmd = (Cmd*) head -> value;
    char* msg = head_cmd -> message;
    //do not need to split
    if(strlen(msg) < payload_size){
      return;
    }
    int i;
    LLnode* curr;
    LLnode* next;
    Cmd* next_cmd;
    curr = head;
    for(i = payload_size; i < strlen(msg); i += payload_size){
         //TODO: malloc here
         char* cmd_msg = (char*) malloc((cut_size + 1) *
         sizeof(char)); // One extra byte for NULL character
         memset(cmd_msg, 0, (payload_size + 1) * sizeof(char));
         strncpy(cmd_msg, msg + i, payload_size);
         //TODO: fill the next_cmd
         //TODO: fill the next node and add it to the linked list
    }
    msg[payload_size] = '\0';//cut the original msg
}
```