

CSE 123 Project 1

Discussion Session 3

-the finer details

Agenda

- What to add in the structures?
- What is the Frame Format?
- Wrap Around
- How to Handle Big Messages
- What to do with messages out of the window?
- How to Buffer Messages?
- How to Handle Timeout?
- Refresher on Bit Operators in C
- CRC – Example and Implementation
- Putting it together – Sender and Receiver

What to add in the structures?

- Frame
 - Receiver and Sender Ids
 - Seq and Ack Numbers
 - CRC
- Sender
 - Last Frame Acknowledged
 - Last Frame Sent
 - Buffering of timestamps and payload
- Receiver
 - Last Acceptable Frame
 - Last Frame Received
 - Buffering Out of order frames

What is the Frame Format?

Frame Size is 64 Bytes . It is a fixed number

- Header (Must be of maximum of 16 bytes)
 - Receiver and Sender Ids
 - Seq and Ack Numbers
 - CRC
- Payload (Must be a minimum of 48 bytes)

Wrap Around

- Sequence Numbers can wrap around.
Example: 1,2,3....253,254,255,0,1,2.....repeat
- Implementation (More in the book)
 - Use a circular linked list OR
 - Use a circular array
 - Use the mode operation on SWS/RWS
example here we have $SWS=RWS=8$
Can use $SeqNo \% 8$, so that if it wraps around you don't have to worry too much.

Wrap Around

(Red=Out of Window, Green=In the Window)

- Initially

0	1	2	L F A	4	5	6	7	8	9	10	L F S	12	..	255
---	---	---	-------------	---	---	---	---	---	---	----	-------------	----	----	-----

- Finally

0	1	2	L F S	4	5	6	4	251	L F A	253	254	255
---	---	---	-------------	---	---	---	----	----	---	-----	-------------	-----	-----	-----

How to Handle Big Messages

Example: Imagine you type a really long message of size 480 characters and you can put only 48 characters in one frame.

- What you should do is read characters worth 8 Frames (i.e. 8×48 characters) from the "input_cmdlist_head" linked list. 8 because window size is specified as 8. Leave the rest (i.e. $(10-2) \times 48$ characters) in the "input_cmdlist_head" linked list. Now try sending the 8 frames. Must now read characters from "input_cmdlist_head" ONLY when you receive ACKs and want to move the sliding window forward.

How to Handle Big Messages

Implementation(One of the Ways)

- The problem is once you pop the "input_cmdlist_head" linked list, you have removed the entire cmd and so with it the entire message i.e. 480 characters.
- But you have to restore the additional $(10-2)*48$ characters somewhere so that you can read from it.
- One way to do it is you put back the $(10-2)*48$ worth of characters of the message into the "input_cmdlist_head" linked list.
- But remember you have to add it to the head of the "input_cmdlist_head" linked list so that these are the first ones that are read.
- Will have to write another function for adding to head of linked list.

How to Handle Big Messages Output

- Your output should be
 - <REC_1>: [part 1 of huge message]
 - <REC_1>: [part 2 of huge message]
 - <REC_1>: [part 3 of huge message]
- Not in any other way.

What to do with messages out of the window

Simple - Discard

How to Buffer Messages?

- Two popular approaches
 - Arrays
 - Linked List (Implementation provided in util.h)
- Need this at both the receiver and sender

How to Handle Timeout of Frames

```
//Declaration
struct timeval  currentTime
struct timeval  expiringTime
long diffOfTime_In_Usec

//Get timestamps
gettimeofday(&currentTime, NULL)
expiringTime = Expiring Time Stamp of the frame "x"

//Find Time Elapsed
diffOfTime_In_Usec = timeval_usecdiff(&currentTime,
                                       &expiringTime );

if (sleep_usec_time < 0) {
    //There is a      timeout for frame "x"
    //Resend frame"x"
}
```

Easy as that !

Refresher on Bit Operators in C

Assume integers X, Y, Z of an equal number of bytes, i.e. `int X, Y, Z;`

`X OR Y` -----> `X | Y`

`X AND Y` -----> `X & Y`

`X XOR Y` -----> `X ^ Y`

`NOT X` -----> `~X`

- Shift X by Y bits to the left -----> `X << Y`
Shift X by Y bits to the right -----> `X >> Y`
- Example: `Z = Z XOR (X shifted to the left by Y bits)`
`Z ^= X << Y;` // ^= operator is combo of XOR and assignment (=)
- Example: `X = (X AND (NOT Y))`
`X = X & (~Y);`
- See http://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htm
- And also <http://www.wikihow.com/Convert-Binary-to-Hexadecimal.htm>

CRC-Example

Example of CRC-8

- polynomial is $x^8 + x^2 + x + 1$
- express as 100000111
- Let's say we want to know 1111100000 divided by our polynomial

```
100000111 | 1111100000
            100000111
            -----
            111101110
            100000111
            -----
            11101001
```

Example of CRC-8

- Example of CRC-8
- polynomial is $x^8 + x^2 + x + 1$ ($k=8$)
- express as 100000111
- Let's say we want to know 1111100000 divided by our polynomial

```
100000111 | 1111100000
            100000111 XOR
            -----
            111101110 shift left by 1
            100000111 XOR
            -----
            11101001 shift left by 1
```

- stop since we are left with 8 bits

CRC-Pseudocode

Example of CRC-8

```
// Function returns the remainder from a CRC calculation on a char* array of length byte_len
char crc8(char* array, size_t byte_len){
    const char poly = shift_right_by_one(x^8 + x^2 + x^1 + 1);
    char crc = array[0];
    int i, j;
    for(i = 1; i < byte_len; i++){
        char next_byte = ith byte of array;
        for(j = 7; j >= 0; j--){ // Start at most significant bit of next byte and work our way down
            if(crc's most significant bit is a 0){
                shift_left_by_one(crc); // Shift out 0
                crc = crc OR get_bit(j, next_byte); // Shift in next bit
            }
            else{ // crc's most significant bit is a 1
                crc = shift_left_by_one((crc XOR (poly))); // Do first 8 bits of modulo 2 subtraction
                crc = crc OR (get_bit(j, next_byte) XOR 1); // Do final bit of modulo 2 subtraction
            } // and place in position 0 of crc register
        }
    }
    return crc;
}
```


Helper Functions / Macros

- You'll need to implement the following:
 - `char get_bit(int pos, char byte);` // Return a char with a value of 0 or 1
 - depending on whether the bit in the pos most significant bit is 0 or 1
 - So if byte has a value of 0x08 or 00001000, then for any value of pos between 0 and 7 other than 3, the return value should be 0 and otherwise 1.
- `shift_right_by_one` -- implement by using the right shift operator in C
- `shift_left_by_one` -- implement by using the left shift operator in C

Putting it together - Sender

- Pick messages from command linked list (sender->input_cmdlist_head) if there is space i.e. there are less than 8 messages in the buffer
- Calculate the CRC and form the header
- Send the frame based on SWP criteria i.e. Add to (outgoing_frames_head_ptr) linked list
- Receive ACKS from sender->input_framelist_head linked list
- If Timeouts again resend . Put buffered frames on outgoing_frames_head_ptr

Putting it together - Receiver

- When Receiving the Frame FIRST check if the CRC is right ; Else discard the frame
- Next check if the Frame is for me; Else discards it
- Check if the frame is in the Sliding Window ; Else Discard
- Send an ACK for it if it meet the other SWP criteria