Computer Networks Assignment - 2 (Part - II & Part - III)

Implementation of Reassembler:-

I have used a mapping of index and string as the buffer of the reassembler, declared some additional variables to handle some important cases of end of the file, and first unacceptable index (first index of the window), for all the pairs If the new index is less than or equal to acknowledgment index (means in order) then push in the byte stream and for overlapping bytes concatenated the string from ack_index to original index.

For all the new indices that are greater than the ack (next expected index), they are temporarily stored in the buffer of the reassembler and the bytes that lie beyond the stream's available capacity are discarded as soon as they are received.

while iterating over the buffer of the reassembler, bytes stored in it are simultaneously pushed into the buffer of the byte stream by checking if its index is equal to **ack** and the remaining capacity is not zero and also erased from the buffer of the reassembler, for those bytes whose index is less than **ack** It means they are already written to the byte stream and hence we will be directly erasing them from the buffer of reassembler.

All test cases of reassembler passed:-

```
      Start 16: fsm_stream_reassembler_cap
16/23 Test #16: fsm_stream_reassembler_cap ...........   Passed    0.20 sec
      Start 17: fsm_stream_reassembler_single
17/23 Test #17: fsm_stream_reassembler_single ........   Passed    0.00 sec
      Start 18: fsm_stream_reassembler_seq
18/23 Test #18: fsm_stream_reassembler_seq ...........   Passed    0.00 sec
      Start 19: fsm_stream_reassembler_dup
19/23 Test #19: fsm_stream_reassembler_dup ...........   Passed    0.01 sec
      Start 20: fsm_stream_reassembler_holes
20/23 Test #20: fsm_stream_reassembler_holes .........   Passed    0.00 sec
      Start 21: fsm_stream_reassembler_many
21/23 Test #21: fsm_stream_reassembler_many ..........   Passed    6.62 sec
      Start 22: fsm_stream_reassembler_overlapping
22/23 Test #22: fsm_stream_reassembler_overlapping ...   Passed    0.01 sec
      Start 23: fsm_stream_reassembler_win
23/23 Test #23: fsm_stream_reassembler_win ...........   Passed    8.20 sec
```

Implementation of TCP Receiver and Wrapping Integers:-

All Implementation of the TCP receiver is handled inside one function only which takes a TCP segment as input and extracts the TCP header information (payload, acknowledgment number, sequence number), **SYN** and **FIN** are the control flags of the beginning and end of the byte stream respectively and **_isn** is the initial sequence number as soon as the end of the byte stream is received it unwraps the 32-bit sequence number using checkpoint and push the payload into the buffer of reassembler and we store the number of bytes written then rest is handled by reassembler:-

How do the wrap and unwrap functions work?

Simply transform the "absolute" 64-bit sequence number (zero-indexed) into a WrappingInt32

wrapping process:- WrappingInt32(initial_sequence_number + uint32_t(n)); where n is absolute 64-bit sequence number.

Unwrapping: This includes transforming a WrappingInt32 into an "absolute" 64-bit sequence number (zero-indexed) as below:-

```
if (checkpoint > static_cast<uint64_t>(unwrapped_no))
{
    uint64_t i = shiftby32 * k;
    if (unwrapped_no + i > checkpoint + shiftby31)
        abs_sequenceNo = unwrapped_no + i-shiftby32;
    else if (unwrapped_no + i + shiftby31 < checkpoint)
        abs_sequenceNo = unwrapped_no + i+shiftby32;
    else
        abs_sequenceNo = unwrapped_no + i;
}
else
{   // if addition doesn't results in overflow
    abs_sequenceNo = unwrapped_no;
}

return abs_sequenceNo;
```

All test cases passed:-

```
akash@akash-Modern-14-B11MOU:~/Pictures/assignment2/build$ ctest
Test project /home/akash/Pictures/assignment2/build
      Start  1: wrapping_integers_cmp
 1/23 Test  #1: wrapping_integers_cmp ..............   Passed    0.00 sec
      Start  2: wrapping_integers_unwrap
 2/23 Test  #2: wrapping_integers_unwrap ............   Passed    0.00 sec
      Start  3: wrapping_integers_wrap
 3/23 Test  #3: wrapping_integers_wrap ..............   Passed    0.00 sec
      Start  4: wrapping_integers_roundtrip
 4/23 Test  #4: wrapping_integers_roundtrip .........   Passed    0.62 sec
      Start  5: byte_stream_construction
 5/23 Test  #5: byte_stream_construction ............   Passed    0.00 sec
      Start  6: byte_stream_one_write
 6/23 Test  #6: byte_stream_one_write ...............   Passed    0.00 sec
      Start  7: byte_stream_two_writes
 7/23 Test  #7: byte_stream_two_writes ..............   Passed    0.00 sec
      Start  8: byte_stream_capacity
 8/23 Test  #8: byte_stream_capacity ................   Passed    1.32 sec
      Start  9: byte_stream_many_writes
 9/23 Test  #9: byte_stream_many_writes .............   Passed    0.00 sec
      Start 10: recv_connect
10/23 Test #10: recv_connect ........................   Passed    0.00 sec
      Start 11: recv_transmit
11/23 Test #11: recv_transmit .......................   Passed    0.12 sec
      Start 12: recv_window
12/23 Test #12: recv_window .........................   Passed    0.00 sec
      Start 13: recv_reorder
13/23 Test #13: recv_reorder ........................   Passed    0.00 sec
      Start 14: recv_close
14/23 Test #14: recv_close ..........................   Passed    0.00 sec
      Start 15: recv_special
15/23 Test #15: recv_special ........................   Passed    0.00 sec
      Start 16: fsm_stream_reassembler_cap
16/23 Test #16: fsm_stream_reassembler_cap ...........   Passed    0.16 sec
      Start 17: fsm_stream_reassembler_single
17/23 Test #17: fsm_stream_reassembler_single ........   Passed    0.00 sec
      Start 18: fsm_stream_reassembler_seq
18/23 Test #18: fsm_stream_reassembler_seq ...........   Passed    0.00 sec
      Start 19: fsm_stream_reassembler_dup
19/23 Test #19: fsm_stream_reassembler_dup ...........   Passed    0.01 sec
      Start 20: fsm_stream_reassembler_holes
20/23 Test #20: fsm_stream_reassembler_holes .........   Passed    0.00 sec
      Start 21: fsm_stream_reassembler_many
21/23 Test #21: fsm_stream_reassembler_many ..........   Passed    6.01 sec
      Start 22: fsm_stream_reassembler_overlapping
22/23 Test #22: fsm_stream_reassembler_overlapping ...   Passed    0.00 sec
      Start 23: fsm_stream_reassembler_win
23/23 Test #23: fsm_stream_reassembler_win ...........   Passed    7.65 sec

100% tests passed, 0 tests failed out of 23

Total Test time (real) =  15.95 sec
```