Indraprastha Institute of Information Technology Delhi (IIITD)

ASSIGNMENT-2(Part II and Part III)

Name-PARAS DHIMAN || Roll no-2021482

Computer Networks - CSE232

Question:

```
paras@paras-HP-Laptop-15s-du1xxx:~/Downloads/2021482-Assignment2/assignment2/build$ make
[  1%] Building CXX object src/CMakeFiles/tcp_reciever.dir/stream_reassembler.cc.o
[  3%] Linking CXX static library libtcp_reciever.a
[ 23%] Built target tcp_reciever
[ 26%] Built target tcp_reciever_checks
[ 28%] Linking CXX executable wrapping_integers_cmp
[ 30%] Built target wrapping_integers_cmp
[ 31%] Linking CXX executable wrapping_integers_unwrap
[ 33%] Built target wrapping_integers_unwrap
[ 34%] Linking CXX executable wrapping_integers_wrap
[ 36%] Built target wrapping_integers_wrap
[ 38%] Linking CXX executable wrapping_integers_roundtrip
[ 39%] Built target wrapping_integers_roundtrip
[ 41%] Linking CXX executable byte_stream_construction
[ 42%] Built target byte_stream_construction
[ 44%] Linking CXX executable byte_stream_one_write
[ 46%] Built target byte_stream_one_write
[ 47%] Linking CXX executable byte_stream_two_writes
[ 49%] Built target byte_stream_two_writes
[ 50%] Linking CXX executable byte_stream_capacity
[ 52%] Built target byte_stream_capacity
[ 53%] Linking CXX executable byte_stream_many_writes
[ 55%] Built target byte_stream_many_writes
[ 57%] Linking CXX executable recv_connect
[ 58%] Built target recv_connect
[ 60%] Linking CXX executable recv_transmit
[ 61%] Built target recv_transmit
[ 63%] Linking CXX executable recv_window
[ 65%] Built target recv_window
[ 66%] Linking CXX executable recv_reorder
[ 68%] Built target recv_reorder
[ 69%] Linking CXX executable recv_close
[ 71%] Built target recv_close
[ 73%] Linking CXX executable recv_special
[ 74%] Built target recv_special
[ 76%] Linking CXX executable fsm_stream_reassembler_cap
[ 77%] Built target fsm_stream_reassembler_cap
[ 79%] Linking CXX executable fsm_stream_reassembler_single
[ 80%] Built target fsm_stream_reassembler_single
[ 82%] Linking CXX executable fsm_stream_reassembler_seq
[ 84%] Built target fsm_stream_reassembler_seq
[ 85%] Linking CXX executable fsm_stream_reassembler_dup
[ 87%] Built target fsm_stream_reassembler_dup
[ 88%] Linking CXX executable fsm_stream_reassembler_holes
[ 90%] Built target fsm_stream_reassembler_holes
[ 92%] Linking CXX executable fsm_stream_reassembler_many
[ 93%] Built target fsm_stream_reassembler_many
[ 95%] Linking CXX executable fsm_stream_reassembler_overlapping
[ 96%] Built target fsm_stream_reassembler_overlapping
[ 98%] Linking CXX executable fsm_stream_reassembler_win
[100%] Built target fsm_stream_reassembler_win
```

```
[ 98%] Linking CXX executable fsm_stream_reassembler_win
[100%] Built target fsm_stream_reassembler_win
paras@paras-HP-Laptop-15s-du1xxx:~/Downloads/2021482-Assignment2/assignment2/build$ ctest
Test project /home/paras/Downloads/2021482-Assignment2/assignment2/build
      Start  1: wrapping_integers_cmp
 1/23 Test  #1: wrapping_integers_cmp ..............   Passed    0.01 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    0.37 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    0.61 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    0.07 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    0.10 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    1.50 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    1.55 sec

100% tests passed, 0 tests failed out of 23

Total Test time (real) =   4.35 sec
paras@paras-HP-Laptop-15s-du1xxx:~/Downloads/2021482-Assignment2/assignment2/build$
```

Building a TCP Receiver - Reassembler

Introduction

This report focuses on Part II of the assignment, which involves building a reassembler to assemble segments received from the sender in the correct order. The `StreamReassembler` class is responsible for managing the reassembly process and storing the reassembled byte stream.

Part II: Building a Reassembler

Background

In the context of TCP communication, data is often divided into segments or substrings before transmission. These segments can arrive at the receiver out of order, overlap, or even be lost during transmission. It is the responsibility of the receiver to reassemble these segments into the correct order to reconstruct the original data stream.

Class Members

The `StreamReassembler` class has the following private members:

- `_output`: An instance of the `ByteStream` class for storing the reassembled byte stream.
- `_first_unassembled_index`: The index of the first byte in the stream that has not been assembled.
- `_stored_not_reassembled`: The count of bytes that have been received but not yet reassembled.
- `_eof`: A flag indicating the end of the entire data stream.
- `_buffer`: A deque used to temporarily store received bytes.
- `_buff_bitmap`: A deque of boolean values indicating whether each byte in `_buffer` has been processed.
- `_capacity`: The capacity of the reassembler, limiting both reassembled and unassembled bytes.

Constructor

The constructor initializes the `StreamReassembler` object with a given capacity.

Reassembly Logic

- `push_substring(const std::string &data, const size_t index, const bool eof)`: This function is responsible for receiving a substring (segment) of bytes, possibly out of order, and assembling any newly contiguous substrings. It also takes care of writing the assembled bytes into the output stream in the correct order.

Contiguous Checking

- `check_contiguous()`: This function checks for contiguous bytes in the buffer and writes them into the output stream. It ensures that bytes are written in order.

Other Member Functions

- `unassembled_bytes() const`: Returns the number of bytes in the substrings that are stored but not yet reassembled.
- `empty() const`: Checks if the internal state of the reassembler is empty (other than the output stream).
- `ack_index() const`: Returns the acknowledge index of the stream, which is the index of the next interested substring.

Reassembly Strategy

The reassembler follows a strategy to manage received segments effectively:

1. Received segments are stored in `_buffer` and marked in `_buff_bitmap`.
2. The `push_substring` function checks if the incoming segment can be assembled immediately or partially, based on the current state of the reassembler.
3. Contiguous bytes in `_buffer` are checked using `check_contiguous`, and if found, they are written into the output stream.
4. The reassembler handles cases where the end of the stream (`eof`) is reached and there are no more unassembled bytes.

Testing

To test the `StreamReassembler` class, we build the project using `make` and run the tests using `ctest .`

Conclusion

In conclusion, the `StreamReassembler` class is a critical component of building a TCP receiver. It effectively manages the reassembly of received segments, ensuring that bytes are assembled in the correct order and written to the output stream. Proper implementation and

testing of the `StreamReassembler` class are essential for the overall functionality of the TCP receiver.