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$
```

TCP Receiver Implementation Report

Introduction

The purpose of this report is to provide a comprehensive explanation of the implementation of the TCP Receiver, specifically Part 3 of the assignment. The TCP Receiver is a critical component of the Transmission Control Protocol (TCP), responsible for receiving and reassembling segments, computing acknowledgment numbers (ackno), and managing the window size.

Properties of TCP Receiver and Wrapping Integers

Before diving into the implementation details, let's briefly recap the properties of the TCP Receiver and the concept of Wrapping Integers, which are essential for understanding the implementation.

TCP Receiver Properties

- The TCP Receiver is responsible for reassembling segments received from the sender into a reliable byte stream.
- It computes the acknowledgment number (ackno) and window size to send back to the sender.
- It handles the reception of SYN and FIN flags.
- The receiver has a capacity limit for the maximum number of bytes it can store.
- It must track whether the first SYN message has been received.
- It must also track whether a FIN message has been received.

Wrapping Integers

Wrapping Integers are used to represent sequence numbers (seqno) and acknowledgment numbers (ackno) in TCP. They are necessary to handle wrapping and the initial sequence number (ISN). Wrapping Integers include the following key concepts:

- They are 32-bit integers expressed relative to an arbitrary ISN.
- Wrapping Integers help handle sequence numbers that wrap around after reaching the maximum value.
- TCP sequence numbers start at a random value known as the ISN.
- Wrapping Integers assist in transforming absolute 64-bit sequence numbers into 32-bit relative sequence numbers.

Now, let's delve into the implementation details of Part 3 of the assignment.

Class Members and Constructors

TCPReceiver Class

The `TCPReceiver` class is responsible for reassembling segments and managing acknowledgments and window size.

- `_reassembler`: The `StreamReassembler` instance responsible for reassembling received segments.
- `_capacity`: The maximum number of bytes that the receiver can store.
- `_synReceived`: A flag indicating whether the first SYN message has been received.
- `_finReceived`: A flag indicating whether a FIN message has been received.
- `_isn`: An instance of `WrappingInt32` representing the Initial Sequence Number (ISN).

Constructors

The constructor of `TCPReceiver` initializes the class members:

cpp
```
TCPReceiver(const size_t capacity)
    : _reassembler(capacity), _capacity(capacity),
_synReceived(false), _finReceived(false), _isn(0) {}
```

The constructor takes the maximum capacity as a parameter and initializes the reassembler, capacity, and flags.

Input Inference

The primary input for the TCP Receiver is the reception of TCP segments. Each received TCP segment contains a header with flags (e.g., SYN, FIN) and a payload with data. The receiver processes these segments to reassemble the byte stream.

Segment Processing

The `segment_received` method of the `TCPReceiver` class handles the processing of incoming TCP segments. Here is a summary of the key steps within this method:

- Extract the TCP header and payload from the received segment.
- Check if the SYN flag is set and whether it's the first SYN message. If so, set `_synReceived` and `_isn`.
- Check if the FIN flag is set, indicating the end of the stream.

- Calculate the absolute sequence number (abs_seqno) based on the received sequence number (seqno) and `_isn`.
- Calculate the stream index based on the absolute sequence number.
- Push the payload data into the reassembler with the calculated stream index.
- If the FIN flag is set, mark `eof` as true to indicate the end of the stream.

Output Inference

The TCP Receiver provides two main pieces of information as output:

Acknowledgment Number (ackno)

The acknowledgment number represents the beginning of the receiver's window, indicating the sequence number of the first byte in the stream that the receiver hasn't received. If the SYN has not been received, the ackno is empty.

The `ackno` is computed based on the acknowledgment index and whether the reassembler is empty. It ensures that the receiver acknowledges the correct sequence number, taking into account the end of the stream (FIN).

Window Size

The window size represents the available capacity in the output ByteStream. It is computed as the difference between the sequence number of the first byte that falls after the window and the sequence number of the beginning of the window (ackno). This calculation ensures that the sender does not send more data than the receiver can handle.

General Accounting

The TCP Receiver maintains several internal states and flags to keep track of the progress of segment reception. These include `_synReceived`, `_finReceived`, and the capacity limit `_capacity`. These states help ensure that the receiver correctly handles the reception of SYN and FIN flags and does not exceed its storage capacity.

Conclusion

In conclusion, this report has provided an overview of the implementation of the TCP Receiver as part of the assignment. It covers the class members, constructors, input inference (segment processing), output inference (acknowledgment number and window size), and general accounting to manage the receiver's internal states.

The TCP Receiver plays a crucial role in ensuring reliable communication between sender and receiver in a TCP connection, handling the complexities of sequence numbers, wrapping, and reassembling data segments.