Lab3 实验报告

王卫东　221900332

2024 年 10 月 29 日

一　Program Structure and Design

For the TCPSender class, I add 2 public methods to help locate the next absolute sequence number.

```
uint64_t get_next_abs_seqno_() const { return next_abs_seqno_; };
Wrap32 get_next_seqno() const { return isn_ + next_abs_seqno_; };
```

Below are the private variables and methods in the TCPSender class.

```
private:
  // Variables initialized in constructor
  ByteStream input_;
  Wrap32 isn_;
  uint64_t initial_RTO_ms_;
  uint64_t next_abs_seqno_ { 0 }; // Next absolute sequence number to send
  uint64_t window_size_ { 1 }; // Current window size
  uint64_t bytes_in_flight_ { 0 }; // Number of bytes in flight
  // std::queue<TCPSenderMessage> send_queue_ {}; // Queue of messages to send
  bool syn_sent_ { false }; // Has the SYN flag been sent?
  bool fin_sent_ { false }; // Has the FIN flag been sent?
  uint64_t consecutive_retransmissions_ { 0 }; // Number of consecutive
       retransmissions
  uint64_t rto_ { initial_RTO_ms_ }; // Retransmission Timeout
  uint64_t timer_elapsed_ { 0 }; // Time elapsed since last tick
  bool timer_running_ { false }; // Is the timer running?
  std::queue<TCPSenderMessage> segments_outstanding_ {}; // Outstanding segments
```

For make_empty_message, correctly setting the true value is enough.

```
TCPSenderMessage TCPSender::make_empty_message() const
{
  TCPSenderMessage msg;
```

```
    msg.seqno = isn_ + next_abs_seqno_;
    msg.RST = input_.has_error();
    return msg;
  }
```

For push function, we need to pay attention that the window size should be at least 1.Also, calculate the right payload size and update the next absolute sequence number.Meanwhile, we need to check whether to append the FIN flag.

```
  void TCPSender::push( const TransmitFunction& transmit )
  {
    uint64_t window = max( window_size_, static_cast<uint64_t>( 1 ) );
    while ( bytes_in_flight_ < window ) {
      TCPSenderMessage seg;
      if ( !syn_sent_ ) {
      seg.SYN = true;
      syn_sent_ = true;
      }
      auto payload_size = min( TCPConfig::MAX_PAYLOAD_SIZE,
                       min( window - bytes_in_flight_ - static_cast<uint64_t>( seg
                          .SYN ? 1 : 0 ),
                        input_.reader().bytes_buffered() ) );
      string payload;
      while ( payload.size() < payload_size ) {
        auto view = input_.reader().peek();
        if ( view.empty() ) {
          throw std::runtime_error( "Reader::peek()␣returned␣empty␣string_view" );
        }
        view = view.substr( 0, payload_size - payload.size() );
        payload += view;
        input_.reader().pop( view.size() );
      }
      seg.payload = payload;
      if ( !fin_sent_ && input_.reader().is_finished() && bytes_in_flight_ + seg.
          sequence_length() < window ) {
        seg.FIN = true;
        fin_sent_ = true;
      }
      if ( seg.sequence_length() == 0 )
        break; // empty message

      seg.seqno = get_next_seqno();
      seg.RST = input_.has_error();
      transmit( seg );
      if ( !timer_running_ ) {
```

```
      timer_running_ = true;
      timer_elapsed_ = 0;
    }
    segments_outstanding_.push( seg );
    bytes_in_flight_ += seg.sequence_length();
    next_abs_seqno_ += seg.sequence_length();
  }
}
```

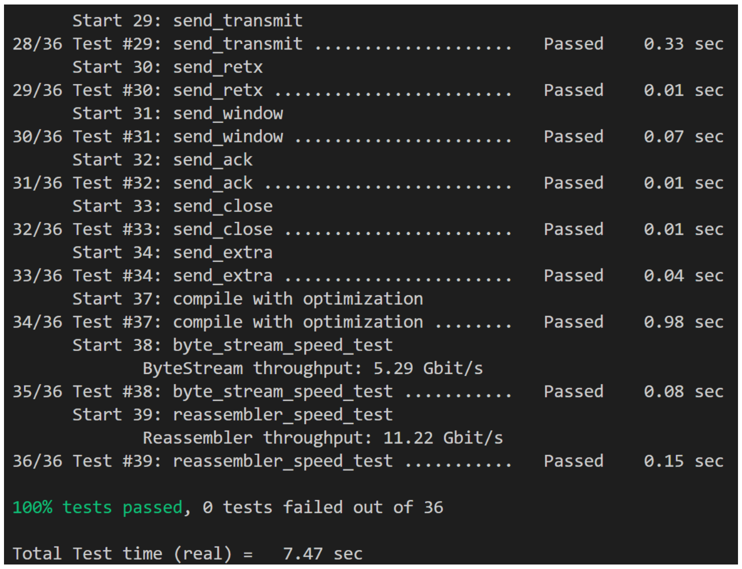
In the receive function, we need to handle the rst value and adjust the queue of outstanding segments.

```
void TCPSender::receive( const TCPReceiverMessage& msg )
{
  msg.RST ? input_.set_error() : void();
  if ( !msg.ackno.has_value() ) {
    window_size_ = msg.window_size;
    return;
  }
  auto abs_ackno = msg.ackno.value().unwrap( isn_, next_abs_seqno_ );
  if ( abs_ackno > next_abs_seqno_ )
    return;
  while ( !segments_outstanding_.empty() ) {
    auto& seg = segments_outstanding_.front();
    if ( seg.seqno.unwrap( isn_, next_abs_seqno_ ) + seg.sequence_length() <=
         abs_ackno ) {
      bytes_in_flight_ -= seg.sequence_length();
      segments_outstanding_.pop();
      timer_elapsed_ = 0;
      consecutive_retransmissions_ = 0;
      rto_ = initial_RTO_ms_;
    } else {
      break;
    }
  }
  if ( bytes_in_flight_ == 0 ) {
    timer_running_ = false;
  }
  window_size_ = msg.window_size;
}
```

In the tick function, we need to handle the timer and retransmission.

```
void TCPSender::tick( uint64_t ms_since_last_tick, const TransmitFunction&
     transmit )
{
  if ( !timer_running_ ) {
    return;
  }
  timer_elapsed_ += ms_since_last_tick;
  if ( timer_elapsed_ >= rto_ ) {
    transmit( segments_outstanding_.front() );
    if ( window_size_ > 0 ) {
     consecutive_retransmissions_++;
     rto_ *= 2;
    }
    timer_elapsed_ = 0;
  }
}
```

二 Experimental Results



```
      Start 29: send_transmit
28/36 Test #29: send_transmit ....................   Passed    0.33 sec
      Start 30: send_retx
29/36 Test #30: send_retx ........................   Passed    0.01 sec
      Start 31: send_window
30/36 Test #31: send_window ......................   Passed    0.07 sec
      Start 32: send_ack
31/36 Test #32: send_ack .........................   Passed    0.01 sec
      Start 33: send_close
32/36 Test #33: send_close .......................   Passed    0.01 sec
      Start 34: send_extra
33/36 Test #34: send_extra .......................   Passed    0.04 sec
      Start 37: compile with optimization
34/36 Test #37: compile with optimization ........   Passed    0.98 sec
      Start 38: byte_stream_speed_test
             ByteStream throughput: 5.29 Gbit/s
35/36 Test #38: byte_stream_speed_test ...........   Passed    0.08 sec
      Start 39: reassembler_speed_test
             Reassembler throughput: 11.22 Gbit/s
36/36 Test #39: reassembler_speed_test ...........   Passed    0.15 sec

100% tests passed, 0 tests failed out of 36

Total Test time (real) =   7.47 sec
```

图 1: passing lab3 tests

三 Challenge

Pay attention to the initialization of the window_size_ since it need to be 1 instead of 0.(to keep transmission and trigger the condition in the timer).
Besides, you need to set the true rst value in the sender function(make_empty_message)

and set error in the receive function(call set_error if rst in received message is true).