Lab2 实验报告

王卫东　221900332

2024 年 10 月 20 日

一　Program Structure and Design

For the wrap function, I use a single type cast:

```
Wrap32 Wrap32::wrap( uint64_t n, Wrap32 zero_point )
{
  return Wrap32{static_cast<uint32_t>(n) + zero_point.raw_value_};
}
```

As for the unwrap function, I respectively calculate the high and low bits of the 64-bit integer:

```
uint64_t Wrap32::unwrap( Wrap32 zero_point, uint64_t checkpoint ) const
{
  auto seqno_offset = raw_value_ - zero_point.raw_value_;
  auto bit1 = (checkpoint + (1 << 31)) & 0xFFFFFFFF00000000;
  auto bit2 = (checkpoint - (1 << 31)) & 0xFFFFFFFF00000000;
  auto res1 = seqno_offset | bit1;
  auto res2 = seqno_offset | bit2;
  if(max(res1,checkpoint) - min(res1,checkpoint) <= max(res2,checkpoint) - min(
      res2,checkpoint)){
    return res1;
  }
  return res2;
}
```

For the receive function, I simply set the correct _isn and insert the data into the reassambly buffer:

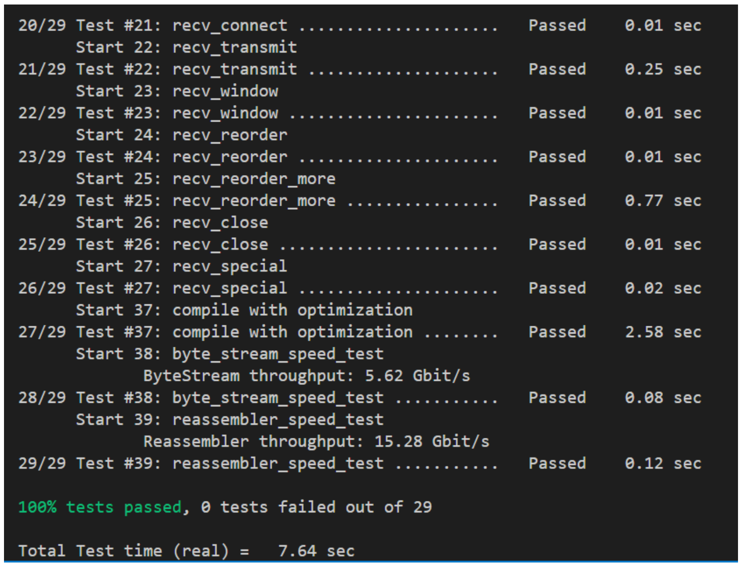
```
void TCPReceiver::receive( TCPSenderMessage message )
{
  if(message.RST) {
    reassembler_.reader().set_error();
```

```
      return;
    }
    if(!_isn.has_value()){
      if(!message.SYN){
        return;
      }
      _isn = message.seqno;
    }
    uint64_t checkpoint = reassembler_.writer().bytes_pushed();// +1?
    uint64_t abs_seqno = message.seqno.unwrap(_isn.value(), checkpoint);
    uint64_t stream_index = abs_seqno - 1 + message.SYN;
    reassembler_.insert(stream_index, message.payload, message.FIN);
  }
```

And in the send part, I set the true value of acknumber and window size:

```
  TCPReceiverMessage TCPReceiver::send() const
  {
    std::optional<Wrap32> ack;
    if(!_isn.has_value()) {
      ack = nullopt;
    } else {
      uint64_t abs_seq = reassembler_.writer().bytes_pushed() + 1 + (reassembler_.
           writer().is_closed() ? 1 : 0);
      ack.emplace(Wrap32::wrap(abs_seq, _isn.value()));
    }
    uint16_t capa = min(UINT16_MAX, static_cast<int>(reassembler_.writer().
         available_capacity()));
    bool rst = false;
    if(reassembler_.writer().has_error()) rst = true;
    return TCPReceiverMessage{ack,capa,rst};
  }
```

二 Experimental Results



```
20/29 Test #21: recv_connect .....................   Passed    0.01 sec
      Start 22: recv_transmit
21/29 Test #22: recv_transmit ....................   Passed    0.25 sec
      Start 23: recv_window
22/29 Test #23: recv_window ......................   Passed    0.01 sec
      Start 24: recv_reorder
23/29 Test #24: recv_reorder .....................   Passed    0.01 sec
      Start 25: recv_reorder_more
24/29 Test #25: recv_reorder_more ................   Passed    0.77 sec
      Start 26: recv_close
25/29 Test #26: recv_close .......................   Passed    0.01 sec
      Start 27: recv_special
26/29 Test #27: recv_special .....................   Passed    0.02 sec
      Start 37: compile with optimization
27/29 Test #37: compile with optimization ........   Passed    2.58 sec
      Start 38: byte_stream_speed_test
             ByteStream throughput: 5.62 Gbit/s
28/29 Test #38: byte_stream_speed_test ...........   Passed    0.08 sec
      Start 39: reassembler_speed_test
             Reassembler throughput: 15.28 Gbit/s
29/29 Test #39: reassembler_speed_test ...........   Passed    0.12 sec

100% tests passed, 0 tests failed out of 29

Total Test time (real) =   7.64 sec
```

图 1: passing lab1 tests

三 Challenge

Pay attention to the number calculated in the unwrap function to maintain a 2^{32} range window size.

Besides, do notice that we need to consider the SYN flag when calculating the stream index and the FIN bit when calculating the abs_seqno in the send function.

Finally, you need to set the true rst value in the send function and set error to Reader in the receive function when receiving a RST message.