Lab2实验报告

几种序号的相互转换

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

这个函数很简单，只需要以zero_point为基准，进行映射即可。

```cpp
uint64_t Wrap32::unwrap( Wrap32 zero_point, uint64_t checkpoint ) const
{
  // if(raw_value_>=zero_point.raw_value_)return checkpoint + raw_value_-zero_point.raw_value_;
  // else return checkpoint + zero_point.raw_value_ - raw_value_ + (static_cast<uint64_t>(1)<<32);
  uint64_t sub{static_cast<uint64_t>(raw_value_ - wrap(checkpoint,zero_point).raw_value_)};
  uint64_t res{sub + checkpoint};
  if((sub >=static_cast<uint64_t>(1)<<31) &&
     res>=(static_cast<uint64_t>(1)<<32))[[unlikely]]
  res -= static_cast<uint64_t>(1)<<32;

  return res;
}
```

这个函数稍显复杂。
我们先考虑rawvalue和checkpoint的距离。由于两者不是同一个参考系，因此需要先把checkpoint映射回seqno，然后再计算（这就是SUB）
很自然地，在64位的参照系下，rawvalue与checkpoint的距离就是sub（不考虑回环的情况下）。如果考虑回环，那么一旦sub超过了32位数的一半，那么在64位参考系中，选择绕回来和checkpoint的距离会小于不绕回的距离。因此绕回来即可。

碰到的error：我们还需要考虑并不总是能绕回来。如果盲目地 -= 2^{32} ，那么当res小于2^{32} 的情况下，一旦回环，得到的结果相当于在64位系统中进行了一个回环，得到的是一个非常接近于2^{64}的数，因此这时不能直接减

TCP Receiver

需要考虑很多边界条件的处理。
在测试的时候，出现了Recv_reorder测试超时，导致我不得不重写了ByteStream。使用了线

性的string代替了非线性的deque，并且使用了大量的string_view来加速。

```
  private:
    Reassembler reassembler_;
    bool SYN{false};
    Wrap32 ISN{ 0 };
   bool RSTflag{false};
```

```
void TCPReceiver::receive( TCPSenderMessage message )
{
  if(message.SYN)
  {
    SYN = true;
    ISN = {message.seqno};
  }
  // if(message.RST)
  // cerr<<"YEs,we have RST"<<endl;
  // cerr<<"now ,I come to here"<<endl;
  if(reassembler_.reader().has_error()||RSTflag||message.RST)[[unlikely]]
  {
    RSTflag = true;
    reassembler_.reader().set_error();
    return;
  }
  //cerr<<"\n\n Rst:"<<RSTflag<<endl;
  if(SYN)[[likely]]
  //if(SYN)[[likely]]
  {
   // std::cout<<"\n\nIAMHERE"<<std::endl;
    reassembler_.insert((message.seqno+message.SYN).unwrap(ISN,reassembler_.get__first_unassembled_index())-1,message.payload,message.FIN);
  }
  // cerr<<"i have receive"<<endl;
  return;
}
```

增加了几个成员变量来维护状态。如果message中含有SYN，那么就记录下SYN和ISN。如果message的RST为1或者数据流本身就关闭了，那么就设置一下（特别注意要设置error）

文档中对RST的描述不详，经过多次cerr调试以及对测试的推断，才能得到结果。

如果SYN打开了，我们才进行插入。

这里index需要加上SYN才能得到absolute seqno。

```cpp
TCPReceiverMessage TCPReceiver::send()
{
  // if(reassembler_.writer().is_closed())
  // {
  //   cerr<<"\n\nYES,writer have closed!"<<std::endl;
  // }
  if(reassembler_.reader().has_error())
  {
    RSTflag = true;
  }
  std::optional<Wrap32> res_ackno{Wrap32::wrap(reassembler_.get_first_unassembled_index(),ISN)+SYN+reassembler_.writer().is_closed()};
  uint16_t res_window_size { static_cast<uint16_t>(min(reassembler_.writer().available_capacity(),static_cast<uint64_t>(65535)))};
  if(!SYN)[[unlikely]]
  {
   // cerr<<"\n\nset Rst:"<<RSTflag<<endl;
    return
    {.window_size = res_window_size,
     .RST = RSTflag
     //.RST = reassembler_.writer().has_error()
    };
  }
 // cerr<<"\n\nset Rst:"<<RSTflag<<endl;
  return
  {

    .ackno = res_ackno,
    .window_size = res_window_size,
    .RST = RSTflag
    //.RST = reassembler_.writer().has_error()
  };
}
```

返回的res_ackno仍然需要注意+SYN才能得到seqno。

这里通过测试，推断出如果已经close了，那么需要+1.这应该是给FIN留下的位置。

窗口大小只需在可用空间和65535中取最小值即可。

最终根据SYN，决定结果是否需要填充ackno。

测试结果

```
      Start 23: recv_window
22/29 Test #23: recv_window ......................   Passed    0.03 sec
      Start 24: recv_reorder
23/29 Test #24: recv_reorder .....................   Passed    0.03 sec
      Start 25: recv_reorder_more
24/29 Test #25: recv_reorder_more ................   Passed    2.88 sec
      Start 26: recv_close
25/29 Test #26: recv_close .......................   Passed    0.03 sec
      Start 27: recv_special
26/29 Test #27: recv_special .....................   Passed    0.05 sec
      Start 37: compile with optimization
27/29 Test #37: compile with optimization ........   Passed    1.39 sec
      Start 38: byte_stream_speed_test
             ByteStream throughput: 1.66 Gbit/s
28/29 Test #38: byte_stream_speed_test ...........   Passed    0.24 sec
      Start 39: reassembler_speed_test
             Reassembler throughput: 1.95 Gbit/s
29/29 Test #39: reassembler_speed_test ...........   Passed    0.45 sec

100% tests passed, 0 tests failed out of 29

Total Test time (real) =  14.29 sec
Built target check2
wsll@wsll-virtual-machine:~/桌面/Conputer_Network_code/minnow$
```

经过对bytestream的重写，速度比之前提升了一个数量级。