check 2

2.1 Translating between 64-bit indexes and 32-bit seqnos

element	SYN	c	a	t	FIN
seqno	$2^{32}-2$	$2^{32}-1$	0	1	2
absolute seqno	0	1	2	3	4
stream index		0	1	2	

1.32位整数仅能表示4GiB的索引，一旦32位序列号溢出，流中的下一个字节的序列号将为零。

2.流中的第一个字节的序列号是一个随机的32位数字（而不是0），称为初始序列号（ISN）。原 因是：为了提高健壮性，TCP试图确保序列号不会被猜出来，也不太可能重复。

3.在TCP中，SYN（流的开始）和FIN（流的结束）控制标志也会被分配序列号。（SYN标志占用的序列号是ISN）因为除了确保接收所有字节的数据外，TCP还确保可靠地接收流的开始和结束。流中的每个数据字节也占用一个序列号。请记住，SYN和FIN不是流本身的一部分，也不是“字节”——它们表示字节流本身的开始和结束。

无符号32位整数超出极限数值后会循环到能表示的最小值，对于64位转32位，直接截断即可。

```
Wrap32 Wrap32::wrap( uint64_t n, Wrap32 zero_point )
{
  // Your code here.
  (void)n;
  (void)zero_point;
  return zero_point+static_cast<uint32_t>(n);
}

uint64_t Wrap32::unwrap( Wrap32 zero_point, uint64_t checkpoint ) const
{
  // Your code here.
  uint64_t abs_seqno=static_cast<uint64_t>(this->raw_value_-
zero_point.raw_value_);
  //near checkpoint
  uint64_t mod_times=checkpoint>>32;
  uint64_t the_mod=checkpoint<<32>>32;

```

```
  uint64_t boundary;
  if(the_mod<(1UL<<31)) boundary=mod_times;
  else boundary=mod_times+1;

  uint64_t absl=abs_seqno+((boundary==0?0:boundary-1)<<32);
  uint64_t absr=abs_seqno+(boundary<<32);

  if(checkpoint<(absl+absr)/2){
    abs_seqno=absl;
  }else{
    abs_seqno=absr;
  }

  (void)zero_point;
  (void)checkpoint;
  return abs_seqno;
}
```

2.2 Implementing the TCP receiver

1.段开头的序列号（seqno），如果设置了SYN标志，这是SYN标志的序列号。否则，它是有效载荷开头的的序列号。

2.是否设置了SYN标志。设置了标志就代表这一段数据是开头的数据。

3.有效负载：待传输的数据段（可能为空）。

4.是否设置了FIN标志。设置了标志就代表这一段数据是结尾的数据。

5..确认号（ackno）：TCP接收器所需的下一个序列号。如果TCPReceiver尚未收到初始序列号，则该字段为空。2.窗口大小。这是TCP接收器希望接收的序列号的数量，从ackno（如果存在）开始。最大值为65535（UINT16_MAX）

对于tcp_receiver.hh，向类中添加ISN变量，封装初始化序列号；添加变量fin，封装最后一个数据序列号。添加布尔变量is_set_ISN,标识是否设置初始化序号，添加布尔变量is_end，表示是否结束传输，Ressembler中的inset()函数有布尔变量is_last_substring，表示是否是最后字串，此处is_end与其有相同的含义。

tcp_receiver.hh

```
class TCPReceiver
{
protected:
  Wrap32 ISN;       //flag ISN
  Wrap32 fin;       //flag fin
  bool is_set_ISN;   //is set ISN
```

```
  bool is_end;      //is end?
public:
  /*
   * The TCPReceiver receives TCPSenderMessages, inserting their payload into the Reassembler
   * at the correct stream index.
   */
  void receive( TCPSenderMessage message, Reassembler& reassembler, Writer& inbound_stream );

  /* The TCPReceiver sends TCPReceiverMessages back to the TCPSender. */
  TCPReceiverMessage send( const Writer& inbound_stream ) const;
  TCPReceiver():ISN(0),fin(0),is_set_ISN(0),is_end(false){}
};
```

tcp_receiver.cc

```
void TCPReceiver::receive( TCPSenderMessage message, Reassembler& reassembler, Writer& inbound_stream )
{
  // Your code here.
  if(message.SYN==true){                //建立连接
    ISN=Wrap32(message.seqno);          //封装初始化序列号
    is_set_ISN=true;                    //标识设为true
    message.seqno=message.seqno+1;      //SYN标志位占用一个序列号，数据序列号+1
  }

  if(message.FIN==true){                //最后的数据子串
    is_end=true;                        //设置标识
    fin=Wrap32(message.seqno+message.payload.size());      //封装fin号
  }else{
    is_end=false;
  }

  if(is_set_ISN==true){                     //已建立连接并初始化完序列号
    //传入字符串
    /* insert()函数:
      void Reassembler::insert(uint64_t first_index,
                               string data,
                               bool is_last_substring,
                               Writer& output)
      message.seqno.unwrap(ISN,inbound_stream.bytes_pushed())-1: 解封装序列号
      message.payload: 数据串
      is_end: 是否最后子串
      inbound_stream: 推入流

    */
```

```
 reassembler.insert(message.seqno.unwrap(ISN,inbound_stream.bytes_pushed())-1,
                       message.payload,
                       is_end,
                       inbound_stream);
  }
  (void)message;
  (void)reassembler;
  (void)inbound_stream;
}

/*
  发送数据，需要发送ack确认号（下一份数据的起始序列号）
*/
TCPReceiverMessage TCPReceiver::send( const Writer& inbound_stream ) const
{
  // Your code here.
  TCPReceiverMessage tcpreceivermessage;
  //封装ack_no
  Wrap32 ackno=Wrap32::wrap((inbound_stream.bytes_pushed()+1),ISN);
  //流通道当前容量
  uint64_t cur_siez=inbound_stream.available_capacity();
  if(is_set_ISN==true){
    //若ackno==fin，由于fin占用一位，ackno+1
    tcpreceivermessage.ackno=ackno==fin?ackno+1:ackno;
  }

  //调整滑动窗口大小
  tcpreceivermessage.window_size=cur_siez>UINT16_MAX?UINT16_MAX:cur_siez;
  (void)inbound_stream;
  return tcpreceivermessage;
}
```

result

```
Test project /home/sgt/cs/minnow/build
      Start  1: compile with bug-checkers
 1/29 Test  #1: compile with bug-checkers ........   Passed    0.36 sec
      Start  3: byte_stream_basics
 2/29 Test  #3: byte_stream_basics ...............   Passed    0.03 sec
      Start  4: byte_stream_capacity
 3/29 Test  #4: byte_stream_capacity .............   Passed    0.02 sec
      Start  5: byte_stream_one_write
 4/29 Test  #5: byte_stream_one_write ............   Passed    0.02 sec
      Start  6: byte_stream_two_writes
 5/29 Test  #6: byte_stream_two_writes ...........   Passed    0.02 sec
      Start  7: byte_stream_many_writes
 6/29 Test  #7: byte_stream_many_writes ..........   Passed    0.07 sec
```

```
      Start  8: byte_stream_stress_test
 7/29 Test  #8: byte_stream_stress_test ..........   Passed    0.50 sec
      Start  9: reassembler_single
 8/29 Test  #9: reassembler_single ...............   Passed    0.02 sec
      Start 10: reassembler_cap
 9/29 Test #10: reassembler_cap ..................   Passed    0.03 sec
      Start 11: reassembler_seq
10/29 Test #11: reassembler_seq ..................   Passed    0.04 sec
      Start 12: reassembler_dup
11/29 Test #12: reassembler_dup ..................   Passed    0.06 sec
      Start 13: reassembler_holes
12/29 Test #13: reassembler_holes ................   Passed    0.03 sec
      Start 14: reassembler_overlapping
13/29 Test #14: reassembler_overlapping ..........   Passed    0.03 sec
      Start 15: reassembler_win
14/29 Test #15: reassembler_win ..................   Passed    5.63 sec
      Start 16: wrapping_integers_cmp
15/29 Test #16: wrapping_integers_cmp ............   Passed    0.02 sec
      Start 17: wrapping_integers_wrap
16/29 Test #17: wrapping_integers_wrap ...........   Passed    0.01 sec
      Start 18: wrapping_integers_unwrap
17/29 Test #18: wrapping_integers_unwrap .........   Passed    0.01 sec
      Start 19: wrapping_integers_roundtrip
18/29 Test #19: wrapping_integers_roundtrip ......   Passed    1.36 sec
      Start 20: wrapping_integers_extra
19/29 Test #20: wrapping_integers_extra ..........   Passed    0.28 sec
      Start 21: recv_connect
20/29 Test #21: recv_connect .....................   Passed    0.03 sec
      Start 22: recv_transmit
21/29 Test #22: recv_transmit ....................   Passed    0.49 sec
      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    0.04 sec
      Start 25: recv_reorder_more
24/29 Test #25: recv_reorder_more ................   Passed   11.29 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    0.04 sec
      Start 28: compile with optimization
27/29 Test #28: compile with optimization ........   Passed    0.11 sec
      Start 29: byte_stream_speed_test
             ByteStream throughput: 0.41 Gbit/s
28/29 Test #29: byte_stream_speed_test ...........   Passed    0.43 sec
      Start 30: reassembler_speed_test
             Reassembler throughput: 0.33 Gbit/s
29/29 Test #30: reassembler_speed_test ...........   Passed    0.72 sec
```

```
100% tests passed, 0 tests failed out of 29

Total Test time (real) =  21.80 sec
Built target check2
```