# check 3

## 2. The TCP Sender

TCP:一种网络协议，它在不可靠的网络数据流上实现一对流控制的可靠传输的字节流。每一层的对等体经行流控制。

TCPSender：

- 跟踪接收方窗口大小
- 读数据，创建新的TCP数据段，尽可能填满接收方窗口，发送数据。
- 跟踪已发送但未被确认的数据段
- 重发经过既定时间仍未被确认的数据段

### 2.1 How does the TCPSender know if a segment was lost?

- TCPSender 的tick函数计算时间的流逝，超时便重传。
- RTO使用_*initial*_ret'ran'smission_timeout进行初始化，RTO会随时间变化
- 实现超时重传计时器timer，只能使用tick
- 每次发送数据都要启动计时器，RTO事件后过期。
- 已调用tick并且计时器过期：
  - 重传最早的未被确认的段；这些数据要保存在特定的数据结构中
  - 如果窗口大小非零：
    - 跟踪连续重传的数据并增加它，TCPConnection要使用此信息判断是否连接无望并终止
    - RTO值加倍（指数回退），减缓重传速度
  - 重置计时器，在RTO时间后过期（考虑RTO的变化）
- 当接收方给发送方一个确认成功接收到新数据的确认(该确认反映了一个比之前任何确认都大的绝对序列号):
  - 将RTO设为初始值
  - 发送方有未完成的数据，重启计时器
  - 将连续重传计数设为0

## 2.2 Implementing the TCP sender

TCPSender的基本思想：

- 给定的数据字节流
- 分割成段
- 发送给接收方
- 没有很快得到确认，重新发送未被确认的数据

具体接口:

```
1 void push( Reader& outbound_stream );
```

从流中读取数据，只要有新的字节流过来并且有足够的空间，就尽可能多的生成TCPSenderMessage，每个TCPSenderMessage不能大于TCPConfig::MAX PAYLOAD SIZE（1452字节），TCPSenderMessage::sequence length() 用来计算一个TCPSenderMessage所占的序列号数量；如果接收方宣布接收窗口为0，发送方则需要假设窗口大小为1，发送数据，可能会受到接收方反馈的最新窗口大小，继续进行数据传输。

```
1 std::optional<TCPSenderMessage> maybe send();
```

TCPSender实际发送TCPSenderMessage的机会

```
1  void receive( const TCPReceiverMessage& msg );
```

从接收方接受消息，传递新的窗口边界。查看未完成的数据段并清楚已完成的数据段（确认过）。

```
1 void tick( const size_t ms_since_last_tick );
```

记录时间流逝情况。

```
1 void send_empty_message();
```

发送空消息，接收方要配合

tcp_sender.hh

```
lass TCPSender
{
  Wrap32 isn_;
  bool is_send_ISN; /*the flag of ISN*/
  bool is_send_FIN; /*the flag of FIN*/
  uint64_t initial_RTO_ms_;
  int cur_RTO_ms;            // current time
  bool is_set_timer;         // whether start timer
  TCPReceiverMessage recMsg; // the msg of TCPReceiver(ackno,window_size)
  uint64_t abs_seqno;
  size_t pre_window_size;
  std::deque<TCPSenderMessage> outstanding_byte; // send but not ack
  std::deque<TCPSenderMessage> ready_send_byte;  // ready to send;
  size_t outstanding_set_bytes;                  // the bytes of outstanding set
  size_t retransmission_count;                   // the count of restransmission

public:
  /* Construct TCP sender with given default Retransmission Timeout and possible ISN */
  TCPSender( uint64_t initial_RTO_ms, std::optional<Wrap32> fixed_isn );

  /* Push bytes from the outbound stream */
  void push( Reader& outbound_stream );

  /* Send a TCPSenderMessage if needed (or empty optional otherwise) */
  std::optional<TCPSenderMessage> maybe_send();

  /* Generate an empty TCPSenderMessage */
  TCPSenderMessage send_empty_message() const;

  /* Receive an act on a TCPReceiverMessage from the peer's receiver */
  void receive( const TCPReceiverMessage& msg );

  /* Time has passed by the given # of milliseconds since the last time the tick() method was called. */
  void tick( uint64_t ms_since_last_tick );

  /* Accessors for use in testing */
  uint64_t sequence_numbers_in_flight() const;  // How many sequence numbers are outstanding?
```

```
38   uint64_t consecutive_retransmissions() const; // How many consecutive
   *re*transmissions have happened?
39 };
```

tcp_sender.cc

```
1  /* TCPSender constructor (uses a random ISN if none given) */
2  TCPSender::TCPSender( uint64_t initial_RTO_ms, optional<Wrap32> fixed_isn )
3    : isn_( fixed_isn.value_or( Wrap32 { random_device()() } ) )
4    , is_send_ISN( false )
5    , is_send_FIN( false )
6    , initial_RTO_ms_( initial_RTO_ms )
7    , cur_RTO_ms( initial_RTO_ms )
8    , is_set_timer( false )
9    , recMsg()
10   , abs_seqno( 0 )
11   , pre_window_size( 1 )
12   , outstanding_byte()
13   , ready_send_byte()
14   , outstanding_set_bytes( 0 )
15   , retransmission_count( 0 )
16 {
17   recMsg.ackno = isn_;
18   recMsg.window_size = 1;
19 }
20
21 uint64_t TCPSender::sequence_numbers_in_flight() const
22 {
23   // Your code here.
24   return outstanding_set_bytes;
25 }
26
27 uint64_t TCPSender::consecutive_retransmissions() const
28 {
29   // Your code here.
30   return retransmission_count;
31 }
32
33 optional<TCPSenderMessage> TCPSender::maybe_send()
34 {
35   // Your code here.
36   if ( ready_send_byte.size() == 0 )
37     return nullopt;
38   TCPSenderMessage msg( ready_send_byte.front() );
39   ready_send_byte.pop_front();
40   is_set_timer = true;
41   return msg;
42 }
```

```cpp

void TCPSender::push( Reader& outbound_stream )
{
  // Your code here.
  while ( outstanding_set_bytes < recMsg.window_size ) {
    // read a msg
    TCPSenderMessage msg;
    // 1.whether have send?
    if ( !is_send_ISN ) {
      is_send_ISN = true;
      msg.SYN = true;
      msg.seqno = isn_;
    } else {
      msg.seqno = Wrap32::wrap( abs_seqno, isn_ );
    }

    // 2. set lenth
    size_t data_len
      = min( min( static_cast<size_t>( recMsg.window_size - outstanding_set_bytes ), TCPConfig::MAX_PAYLOAD_SIZE ),
             static_cast<size_t>( outbound_stream.bytes_buffered() ) );
    // 3. get data
    read( outbound_stream, data_len, msg.payload );
    // 4.
    if ( outbound_stream.is_finished() && msg.sequence_length() + outstanding_set_bytes < recMsg.window_size ) {
      if ( !is_send_FIN ) {
        is_send_FIN = true;
        msg.FIN = true;
      }
    }
    // 5.
    if ( msg.sequence_length() == 0 )
      break;
    else {
      outstanding_byte.push_back( msg );
      ready_send_byte.push_back( msg );
      outstanding_set_bytes += msg.sequence_length();
    }
    // 6.
    abs_seqno += msg.sequence_length();

    (void)outbound_stream;
  }
}

TCPSenderMessage TCPSender::send_empty_message() const
{
```

```
  // Your code here.
  TCPSenderMessage msg;
  msg.seqno = Wrap32::wrap( abs_seqno, isn_ );
  return msg;
}

void TCPSender::receive( const TCPReceiverMessage& msg )
{
  // Your code here.
  recMsg = msg;
  if ( recMsg.window_size == 0 )
    recMsg.window_size = 1;
  pre_window_size = msg.window_size;
  if ( msg.ackno.has_value() ) {
    if ( msg.ackno.value().unwrap( isn_, abs_seqno ) > abs_seqno )
      return;

    while ( outstanding_set_bytes != 0
            && outstanding_byte.front().seqno.unwrap( isn_, abs_seqno ) +
outstanding_byte.front().sequence_length()
                 <= msg.ackno.value().unwrap( isn_, abs_seqno ) ) {
      outstanding_set_bytes -= outstanding_byte.front().sequence_length();
      outstanding_byte.pop_front();
      if ( outstanding_set_bytes == 0 ) {
        is_set_timer = false;
      } else {
        is_set_timer = true;
      }
      cur_RTO_ms = initial_RTO_ms_;
      retransmission_count = 0;
    }
  }
  (void)msg;
}

void TCPSender::tick( const size_t ms_since_last_tick )
{
  // Your code here.
  if ( is_set_timer ) {
    cur_RTO_ms -= ms_since_last_tick;
  }
  if ( cur_RTO_ms <= 0 ) {
    // retransmission
    ready_send_byte.push_front( outstanding_byte.front() );
    retransmission_count++;
    // ARQ
    if ( pre_window_size > 0 ) {
      cur_RTO_ms = pow( 2, retransmission_count ) * initial_RTO_ms_;
```

```
    } else {
      cur_RTO_ms = initial_RTO_ms_;
    }
  }
  (void)ms_since_last_tick;
}
```

```
cmake --build build --target check3
Test project /home/sgt/cs/minnow/build
      Start  1: compile with bug-checkers
 1/36 Test  #1: compile with bug-checkers ........   Passed   14.53 sec
      Start  3: byte_stream_basics
 2/36 Test  #3: byte_stream_basics ...............   Passed    0.01 sec
      Start  4: byte_stream_capacity
 3/36 Test  #4: byte_stream_capacity .............   Passed    0.01 sec
      Start  5: byte_stream_one_write
 4/36 Test  #5: byte_stream_one_write ............   Passed    0.02 sec
      Start  6: byte_stream_two_writes
 5/36 Test  #6: byte_stream_two_writes ...........   Passed    0.02 sec
      Start  7: byte_stream_many_writes
 6/36 Test  #7: byte_stream_many_writes ..........   Passed    0.07 sec
      Start  8: byte_stream_stress_test
 7/36 Test  #8: byte_stream_stress_test ..........   Passed    0.47 sec
      Start  9: reassembler_single
 8/36 Test  #9: reassembler_single ...............   Passed    0.02 sec
      Start 10: reassembler_cap
 9/36 Test #10: reassembler_cap ..................   Passed    0.02 sec
      Start 11: reassembler_seq
10/36 Test #11: reassembler_seq ..................   Passed    0.03 sec
      Start 12: reassembler_dup
11/36 Test #12: reassembler_dup ..................   Passed    0.04 sec
      Start 13: reassembler_holes
12/36 Test #13: reassembler_holes ................   Passed    0.02 sec
      Start 14: reassembler_overlapping
13/36 Test #14: reassembler_overlapping ..........   Passed    0.02 sec
      Start 15: reassembler_win
14/36 Test #15: reassembler_win ..................   Passed    5.48 sec
      Start 16: wrapping_integers_cmp
15/36 Test #16: wrapping_integers_cmp ............   Passed    0.02 sec
      Start 17: wrapping_integers_wrap
16/36 Test #17: wrapping_integers_wrap ...........   Passed    0.01 sec
      Start 18: wrapping_integers_unwrap
17/36 Test #18: wrapping_integers_unwrap .........   Passed    0.01 sec
      Start 19: wrapping_integers_roundtrip
```

```
18/36 Test #19: wrapping_integers_roundtrip ......   Passed    1.40 sec
      Start 20: wrapping_integers_extra
19/36 Test #20: wrapping_integers_extra ..........   Passed    0.27 sec
      Start 21: recv_connect
20/36 Test #21: recv_connect .....................   Passed    0.02 sec
      Start 22: recv_transmit
21/36 Test #22: recv_transmit ....................   Passed    0.49 sec
      Start 23: recv_window
22/36 Test #23: recv_window ......................   Passed    0.02 sec
      Start 24: recv_reorder
23/36 Test #24: recv_reorder .....................   Passed    0.02 sec
      Start 25: recv_reorder_more
24/36 Test #25: recv_reorder_more ................   Passed   10.74 sec
      Start 26: recv_close
25/36 Test #26: recv_close .......................   Passed    0.02 sec
      Start 27: recv_special
26/36 Test #27: recv_special .....................   Passed    0.03 sec
      Start 28: send_connect
27/36 Test #28: send_connect .....................   Passed    0.02 sec
      Start 29: send_transmit
28/36 Test #29: send_transmit ....................   Passed    0.59 sec
      Start 30: send_retx
29/36 Test #30: send_retx ........................   Passed    0.02 sec
      Start 31: send_window
30/36 Test #31: send_window ......................   Passed    0.31 sec
      Start 32: send_ack
31/36 Test #32: send_ack .........................   Passed    0.02 sec
      Start 33: send_close
32/36 Test #33: send_close .......................   Passed    0.02 sec
      Start 34: send_extra
33/36 Test #34: send_extra .......................   Passed    0.10 sec
      Start 36: compile with optimization
34/36 Test #36: compile with optimization ........   Passed    3.06 sec
      Start 37: byte_stream_speed_test
             ByteStream throughput: 0.46 Gbit/s
35/36 Test #37: byte_stream_speed_test ...........   Passed    0.35 sec
      Start 38: reassembler_speed_test
             Reassembler throughput: 0.40 Gbit/s
36/36 Test #38: reassembler_speed_test ...........   Passed    0.61 sec

100% tests passed, 0 tests failed out of 36

Total Test time (real) =  38.90 sec
Built target check3
```