Lab 4 Writeup

My name: 王聪颖

My Student number : 211220180

This lab took me about 24 hours to do. I did attend the lab session.

1. Program Structure and Design:

TCPConnection中新添加的成员如下，作用见注释：

```
class TCPConnection {
  private:
    // ......
    // begin added new private member
    size_t _time_since_last_segment_received{0};
    // set a segment's ack, ackno, win
    void set_seg_ack_win(TCPSegment &seg);
    // fill TCPConnection::_segments_out from TCPSender::_segments_out,
return whether send at least one segment
    bool fill_segments_out();
    // judge whether inbound ended
    bool inbound_ended() const;
    // judge whether outbound ended
    bool outbound_ended() const;
    // send an empty segment with the rst flag set
    void send_rst_seg();
    // end added new private member
    // ......
}
```

重要函数介绍，详见注释：

```
void TCPConnection::segment_received(const TCPSegment &seg) {
    // rst bit
    if (seg.header().rst) {
        // LISTEN -> ignore rst
        if (LISTEN())
            return;
        // unclean shutdown
        _sender.stream_in().set_error();
        _receiver.stream_out().set_error();
        return;
    }
    // update _time_since_last_segment_received
    _time_since_last_segment_received = 0;
    // give the segment to the TCPReceiver
```

```
    _receiver.segment_received(seg);
    // condition of _linger_after_streams_finish
    if (inbound_ended() && !_sender.stream_in().eof())
        _linger_after_streams_finish = false;
    // ack bit
    bool send_one = false;
    if (seg.header().ack) {
        _sender.ack_received(seg.header().ackno, seg.header().win);
        send_one = fill_segments_out();
    }
    // incoming segment occupies sequence numbers
    if (seg.length_in_sequence_space() > 0 && !send_one) {
        _sender.fill_window();  // send possibly SYN
        send_one = fill_segments_out();
        // at least one segment is sent in reply
        if (!send_one) {
            _sender.send_empty_segment();
            fill_segments_out();
        }
    }
}

bool TCPConnection::active() const {
    // unclean shutdown
    if (_sender.stream_in().error() || _receiver.stream_out().error())
        return false;
    // clean shutdown
    bool checked = inbound_ended() && outbound_ended();
    if (checked) {
        if (!_linger_after_streams_finish)
            return false;
        else {
            if (time_since_last_segment_received() >= 10 * _cfg.rt_timeout)
                return false;
            else
                return true;
        }
    }
    return true;
}

//! \param[in] ms_since_last_tick number of milliseconds since the last
call to this method
void TCPConnection::tick(const size_t ms_since_last_tick) {
    _time_since_last_segment_received += ms_since_last_tick;
    // tell the TCPSender the passage of time
    _sender.tick(ms_since_last_tick);
    // abort the connection, and send a reset segment to the peer (an empty
segment with the rst flag set),
    // if the number of consecutive retransmissions is more than an upper
limit TCPConfig::MAX_RETX_ATTEMPTS
    if (_sender.consecutive_retransmissions() > _cfg.MAX_RETX_ATTEMPTS) {
        send_rst_seg();  // missed retransmission seg, instead, an empty
seg with RST=1
```

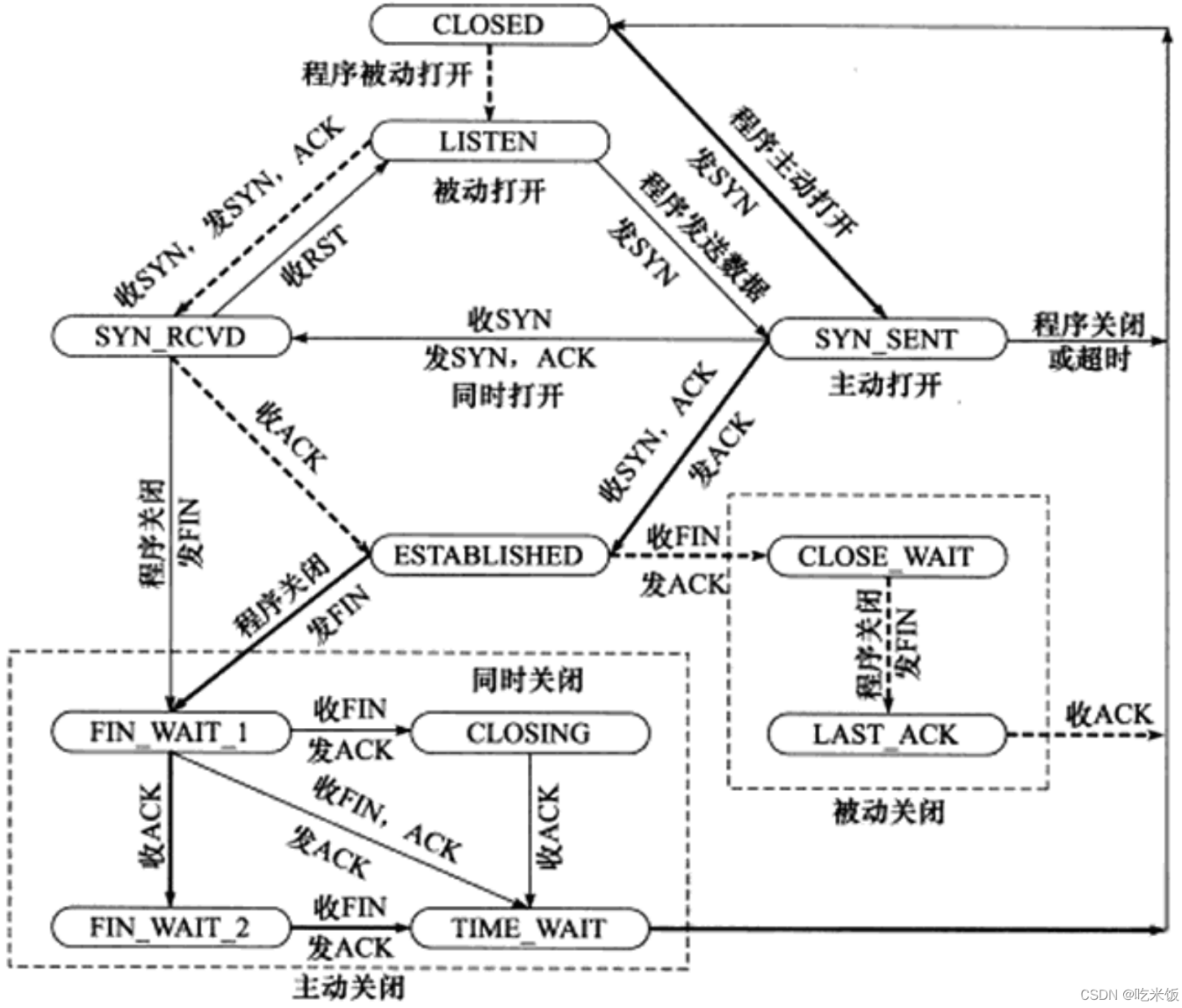
```
        _sender.stream_in().set_error();
        _receiver.stream_out().set_error();
        return;
    }
    fill_segments_out();
}

bool TCPConnection::inbound_ended() const { return
_receiver.stream_out().input_ended(); }

bool TCPConnection::outbound_ended() const {
    return _sender.stream_in().eof() && _sender.next_seqno_absolute() ==
_sender.stream_in().bytes_written() + 2 &&
           _sender.bytes_in_flight() == 0;
}
```

修改底层数据结构ByteStream，提高吞吐量：

```
class ByteStream {
  private:
    // Your code here -- add private members as necessary.
    BufferList _buffer = {};
    size_t _capacity;
    // size_t _size{0}; not defined, use the _buffer.size() instead to
simplify code and avoid bug
    size_t _total_written{0};
    size_t _total_read{0};
    bool _flag_input_ended{false};
}

size_t ByteStream::write(const string &data) {
    size_t write_len = min(remaining_capacity(), data.length());
    _buffer.append(BufferList(move(string{data.begin(), data.begin() +
write_len})));
    _total_written += write_len;
    return write_len;
}

//! \param[in] len bytes will be copied from the output side of the buffer
string ByteStream::peek_output(const size_t len) const {
    size_t peek_len = min(buffer_size(), len);
    string &&total_str = _buffer.concatenate();
    return string{total_str.begin(), total_str.begin() + peek_len};
}

//! \param[in] len bytes will be removed from the output side of the buffer
void ByteStream::pop_output(const size_t len) {
    size_t pop_len = min(buffer_size(), len);
    _buffer.remove_prefix(pop_len);
    _total_read += pop_len;
    return;
}
```

2. Implementation Challenges:

难度较大的部分是TCP的状态转移和active()判断，手册的状态转移描述不是很清晰，从网上找了TCP的官方状态转移进行参照



3. Remaining Bugs:

本次实验测试全部通过，无遗留Bug，且吞吐量满足要求

```
        Start 159: t_isnR_128K_8K_l
 157/162 Test #159: t_isnR_128K_8K_l .................   Passed    0.40 sec
        Start 160: t_isnR_128K_8K_L
 158/162 Test #160: t_isnR_128K_8K_L .................   Passed    0.27 sec
        Start 161: t_isnR_128K_8K_lL
 159/162 Test #161: t_isnR_128K_8K_lL ................   Passed    0.55 sec
        Start 162: t_isnD_128K_8K_l
 160/162 Test #162: t_isnD_128K_8K_l .................   Passed    0.30 sec
        Start 163: t_isnD_128K_8K_L
 161/162 Test #163: t_isnD_128K_8K_L .................   Passed    0.28 sec
        Start 164: t_isnD_128K_8K_lL
 162/162 Test #164: t_isnD_128K_8K_lL ................   Passed    1.40 sec

 100% tests passed, 0 tests failed out of 162

 Total Test time (real) =  39.63 sec
 [100%] Built target check_lab4
● oslab@oslab-virtual-machine:~/Desktop/lab4-2023autumn-HistoriaY/sponge/build$ ./apps/tcp_benchmark
 CPU-limited throughput               : 0.39 Gbit/s
 CPU-limited throughput with reordering: 0.38 Gbit/s
```