

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机网络基础 |
| 实验名称： | Lab5:the summit(TCP in full) |
| 姓 名： |  |
| 学 院： | 计算机学院 |
| 系： |  |
| 专 业： |  |
| 学 号： |  |
| 指导教师： |  |

年 月 日

**浙江大学实验报告**

实验名称： Lab5:the summit(TCP in full) 实验类型： 设计实验

同组学生： 实验地点：

# 实验背景：

* 学习掌握TCP的工作原理

# 实验目的：

* 学习掌握TCP的工作原理
* 学习掌握TCP connection的相关知识
* 学习掌握协议栈结构

# 实验内容

* 将TCPSender和TCPReceiver结合，实现一个TCP终端，同时收发数据。
* 实现TCP connection的状态管理，如连接和断开连接等。
* 整合网络接口、IP路由以及TCP并实现端到端的通信。

# 主要仪器设备

* 联网的PC机
* Linux虚拟机

# 操作方法与实验步骤

* 在开始整合代码之前，你需要了解以下TCPConnection的准则：

- **接收端**，TCPConnection的**segment\_received()**方法被调用时，它从网络中接收TCPSegment。当这种情况发生时，TCPConnection将查看segment并且：

1. 如果设置了RST标志，将入站流和出站流都设置为错误状态，并永久终止连接。
2. 把这个段交给TCPReceiver，这样它就可以在传入的段上检查它关心的字段：seqno、SYN、负载以及FIN。
3. 如果设置了ACK标志，则告诉TCPSender它关心的传入段的字段：ackno和window\_size。

4. 如果传入的段包含一个有效的序列号，TCPConnection确保至少有一个段作为应答被发送，以反应ackno和window\_size的更新。

5.有一个特殊情况，你将不得不在TCPConnection的**segment\_received()**方法中处理：响应一个“keep-alive”段。对方可能会选择发送一个带有无效序列号的段，以查看你的TCP是否仍然有效，你的TCPConnection应该回复这些“keep-alive”段。实现的代码如下：

# TCP KeepAlive 报文是一种没有任何数据，同时置ACK标志的报文，报文中的序列号为上次发生数据交互时TCP报文序列号减1。请注意，提供的代码中已实现了这部分代码的类似功能。

if (\_receiver.ackno().has\_value() and (seg.length\_in\_sequence\_space() == 0) and seg.header().seqno == \_receiver.ackno().value() - 1)

{

\_sender.send\_empty\_segment()

}

- **发送端**，TCPConnection将通过网络发送TCPSegment：

1. TCPSender通过调用tcp\_sender.cc中的函数，将一个TCPSegment数据包添加到待发送队列中，并设置一些字段（seqno，SYN，FIN）。
2. 在发送当前数据包之前，TCPConnection 会获取当前它自己的 TCPReceiver 的 ackno 和 window size，将其放置进待发送 TCPSegment 中（设置window\_size和ackno），并设置其 ACK 标志。

- **tick**，当时间流逝，TCPConnection有一个tick()方法，该方法将被操作系统定期调用，当这种情况发生时，TCPConnection需要：

1. 告诉TCPSender时间的流逝。
2. 如果连续重传的次数超过上限**TCPConfig::MAX\_RETX\_ATTEMPTS**，则终止连接，并发送一个重置段给对端（设置了RST标志的空段）。
3. 如有必要，结束连接。（什么时候是有必要呢？下一段内容会回答这个问题）

* TCPConnection的一个重要功能是决定TCP连接何时结束。

1. 接收方收到RST 标志或者发送方发送RST 标志后，设置当前 TCPConnection 的输入输出字节流的状态为错误状态，\_is\_active立即赋值为false。这种属于暴力退出（unclear shutdown），可能会导致尚未传输完成的数据丢失（例如仍然在网络中运输的数据包在接收方收到RST标志后被丢弃）。
2. 若想让双方都在数据流收发完整后退出，即实现在没有错误情况下的结束连接，需要尽可能地确保两个字节流的每一个都已完全可靠地传递给了对方。在这种情况下，需要满足四个条件（可以结合4次挥手加以理解）：

* 需要接收的数据（入站流）已经全部接收完毕。
* 需要发送的数据（出站流）已经全部发送完毕。
* 需要发送的数据对方已完全确认。
* 本地TCPConnection确信对端可以满足条件1-3后，有两种选项：
  + 在两个流结束后逗留 **linger**
  + 被动关闭

选择哪一种选项呢？TCPConnection中有一个名为**\_linger\_after\_streams\_finish**的成员变量，变量初始为true。如果在出站流发送结束前（即还没有到达出站流的EOF），入站流已经全部接收完毕，则需要将此变量设置为false。总结一下：

* 满足条件1-3 && **\_linger\_after\_streams\_finish**为false，立即结束连接，即\_is\_active立即赋值为false；
* 满足条件1-3 && **\_linger\_after\_streams\_finish**为true，需要等待足够的时间（10 \* **\_cfg.rt\_timeout**）后结束连接。

你可能会觉得理解状态转换的过程有点困难，不用担心，我们已经为这部分提供了一些代码（segment\_received中的某些代码），你只需要确保你对它有一定的理解就可以很好地完成这个实验。

* 完成TCPConnection编写后，运行**make check**命令来自动测试整个实验所需实现代码的正确性。注意，要先make。
* 重新测试webget，修改**webget.cc**文件，把TCPSocket替换为FullStackSocket，同时将**#include “socket.hh”**替换为**#include “tcp\_sponge\_socket.hh”**，并在你的**get\_URL()**方法的结尾加上**socket.wait\_until\_closed()**，重新进行测试。如果你遇到了什么问题，尝试运行**./apps/webget cs144.keithw.org /hasher/xyzzy**并跟踪其的发送和接收。
* 【选做】最终测试，开启两个终端，在一个终端上运行./apps/lab4 server cs144.keithw.org 3000的命令充当服务器，另一个终端运行./apps/lab4 client cs144.keithw.org 3001命令充当客户端，如果成功连接，请尝试向对端发送信息并看看接收情况。最后在两个终端分别运行ctrl D来结束连接。
* 温馨提示: 当你在开发代码的时候，可能会遇到无法解决的问题，下面给出解决的办法。

- 运行**cmake .. -DCMAKE\_BUILD\_TYPE=RelASan**命令配置build目录，使编译器能够检测内存错误和未定义的行为并给你很好的诊断。

- 你还可以使用valgrind工具。

- 你也可以运行**cmake .. -DCMAKE\_BUILD\_TYPE=Debug**命令配置并使用GNU调试器（**gdb**）。

- 你可以运行**make clean**和**cmake .. -DCMAKE\_BUILD\_TYPE=Release**命令重置构建系统。

- 如果你不知道如何修复遇到的问你题，你可以运行**rm -rf build**命令删除build目录，创建一个新的build目录并重新运行**cmake..**命令。

# 实验数据记录和处理

实现TCPConnection的关键代码截图

* 运行make check命令的运行结果截图
* 重新编写webget.cc的代码截图
* 重新测试webget的测试结果展示
* 最终测试中服务器的运行截图
* 最终测试中客户端的运行截图
* 【选做】在成功连接后端到端发送消息的运行截图
* 【选做】成功关闭连接的截图

# 实验数据记录和处理

* ACK标志的目的是什么？ackno是经常存在的吗？
* 请描述TCPConnection的代码中是如何整合TCPReceiver和TCPSender的。
* 【选做】在最终测试中服务器和客户端能互连吗？如果不能，你分析是什么原因？
* 【选做】在关闭连接的时候是否两端都能正常关闭？如果不能，你分析是原因？

# 实验数据记录和处理