TCP 协议实验

计 21 2012011401 张梦豪

一. 实验目的

通过编程实验，深入理解 TCP 协议原理，掌握连接状态控制、可靠传输等重要机制，还可以进一步探索拥塞控制算法等前沿问题。TCP 协议中的状态控制机制和拥塞控制算法是协议的核心部分。TCP 协议的复杂性主要源于它是一个有状态的协议，需要进行状态的维护和变迁。有限状态机可以很好的从逻辑上表示 TCP 协议的处理过程，理解和实现 TCP 协议状态机是本实验的重点内容。另外， TCP 协议还要向应用层提供编程接口，即网络编程中常用的 Socket 接口。通过实现这些接口函数，可以深入了解网络编程的原理，提高网络程序的设计和调试能力。

二. 实验说明

从设计实现的角度出发，TCP 协议主要考虑以下几个因素：

1）状态控制

2）滑动窗口机制

3）拥塞控制算法

4）性能问题（RTT 估计）

5）Socket 接口

TCP 协议是非常复杂的，将这些内容完全实现具有很大的难度，很难在一个实验中完成所有的内容。出于工作量和实验复杂度的考虑，本实验对 TCP 协议进行适当的简化，仅实现客户端角色的、“停—等”模式的 TCP 协议，能够正确的建立和拆除连接，接收和发送 TCP 段，并向应用层提供客户端需要的 Socket 接口。

一般情况下，TCP 协议通过一个数据结构来控制每个 TCP 连接的发送和接收动作，该数据结构被称为传输控制块（TCB）。TCP 为每个活动的连接维护一个 TCB 结构，所有 TCP 连接对应的 TCB 结构可以组织成一个 TCB 链表结构进行管理。TCB 中包含了有关 TCP 连接的所有信息，包括两端的 IP 地址和端口号、连接状态信息、发送和接收窗口信息等。TCB 的数据结构由学生来设计

完成，本实验不作具体规定，学生可以根据设计实现的需要来具体定义 TCB 结构。

更具体的实验原理性说明见实验书的实现说明部分。（P187）

三．实验内容及实现思路

（1）实验要求

本实验的要求分为以下几点：1）理解 TCP 协议的主要原理，针对客户端角色的、“停-等”模式的 TCP 协议，设计接收和发送流程；2）编程实现 TCP 段的接收流程，重点是段接收的有限状态机；3）编程实现 TCP 段的发送流程，完成 TCP 段的封装处理；4）编程实现客户端的 Socket 接口函数。为完成本实验的要求，学生需完成以下几项内容：

（2）接口函数说明

本实验需要学生实现一些接口函数，同时实验系统也为学生提供了一些接口函数。现分别对这两类接口函数作详细说明。

1）需要实现的接口函数

本实验中需要学生实现的接口函数较多，包括针对段交互的测试函数和针对 socket API 的测试函数两类。

<1> TCP 段输入函数

int stud_tcp_input(char* pBuffer, unsigned shortlen, unsigned int srcAddr, unsigned int

dstAddr)

参数：

char *pBuffer：指向接收缓冲区的指针，从 TCP 头开始。

unsigned short len：缓冲区数据长度。

unsigned int srcAddr：源 IP 地址。

unsigned int dstAddr：目的 IP 地址。

返回值：

如果成功则返回 0，否则返回-1。

<2> TCP 段输出函数：

void stud_tcp_output(char *pData, unsigned short len, unsigned char flag, unsigned short srcPort, unsigned short dstPort, unsigned int srcAddr, unsigned int dstAddr)

参数：

char *pData：数据指针。

unsigned short len：数据长度。

unsigned char flag：段类型。

unsigned short srcPort：源端口。

unsigned short dstPort：目的端口。

unsigned int srcAddr：源 IP 地址。

unsigned int dstAddr：目的 IP 地址。

其中，段类型 flag 为以下几种可能：

1> PACKET_TYPE_DATA：数据。

2> PACKET_TYPE_SYN：SYN 标志位打开。

3> PACKET_TYPE_SYN_ACK：SYN、ACK 标志位打开。

4> PACKET_TYPE_ACK：ACK 标志位打开。

5> PACKET_TYPE_FIN：FIN 标志位打开。

6> PACKET_TYPE_FIN_ACK：FIN、ACK 标志位打开。

<3> 获得 socket 描述符函数：

int stud_tcp_socket(int domain, int type, int protocol)

参数：

int domain：套接字标识符，默认为 INET。

int type：类型，默认为 SOCK_STREAM。

int protocol：协议，默认为 IPPROTO_TCP。

返回值：

如果正确建立连接则返回 socket 描述符，否则返回-1。

<4> TCP 建立连接函数：

int stud_tcp_connect(int sockfd, struct sockaddr_in* addr, int addrlen)

参数：

int sockfd：套接字标识符。

struct sockaddr_in* addr：socket 地址结构指针。

int addrlen socket：地址结构的大小。

返回值：

如果正确发送则返回 0，否则返回-1。

<5> TCP 段发送函数：

int sutd_tcp_send(int sockfd, const unsigned char* pData, unsigned short datalen, int flags)

参数：

int sockfd：套接字标识符。

const unsigned char* pData：数据缓冲区指针。

unsigned short datalen：数据长度。

int flags：标志。

返回值：

如果正确接收则返回 0，否则返回-1。

<6> TCP 段接收函数：

int stud_tcp_recv(int sockfd, const unsigned char* pData, unsigned short datalen, int

flags)

参数：

int sockfd：套接字标识符。

const unsigned char* pData：数据缓冲区指针。

unsigned short datalen：数据长度。

int flags：标志。

返回值：

如果正确接收则返回 0，否则返回-1。

<7> TCP 关闭连接函数：

int stud_tcp_close(int sockfd)

参数：

int sockfd：连接描述符。

返回值：

如果正常关闭则返回 0，否则返回-1。

2）系统提供的接口函数说明

<1> TCP 丢弃段函数：

void tcp_DiscardPkt(char* pBuffer, int type)

参数：

char* pBuffer：指向被丢弃的分组。

int type：分组丢弃的原因，有以下 3 种可能：

1> STUD_TCP_TEST_SEQNO_ERROR：序号错误。

2> STUD_TCP_TEST_SRCPORT_ERROR：源端口错误。

3> STUD_TCP_TEST_DSTPORT_ERRPR：目的端口错误。

<2> IP 分组发送函数：

void tcp_sendIpPkt(unsigned char* pData, uint16 len, unsigned int srcAddr, unsigned int dstAddr, uint8 ttl)

参数：

pData：IP 上层协议数据。

len：IP 上层协议数据长度。

srcAddr：源 IP 地址。

dstAddr：目的 IP 地址。

ttl：最大跳数。

<3> IP 数据分组主动接收函数：

int waitIpPacket(char *pBuffer, int timeout)

参数：

char *pBuffer：接收缓冲区的指针。

int timeout：超时时间。

返回值：

如果正确接收则返回接收到的数据长度，否则返回-1。

<4> 客户端获得本机 IPv4 地址：

UINT32 getIpv4Address()

<5> 客户端获得服务器 IPv4 地址：

UINT32 getServerIpv4Address()

3）全局变量

本实验中涉及的全局变量包括源端口号、目的端口号、TCP 序号和确认号。这些变量的取值可由学生自己定义，比如可以做如下定义：

int gSrcPort = 2008

int gDstPort = 2009

int gSeqNum = 1234

int gAckNum = 0

需要注意的是，序号不能从 0 开始定义，否则系统调用会出现错误。

（3）实验实现思路

从 CLOSED 状态开始，调用 Socket 函数 stud_tcp_connect ( )发送建立连接请求（SYN 报文）后转入 SYN-SENT 状态，等待服务器端的应答和建立连接请求；收到服务器端的 SYN+ACK 后，以 ACK 应答，完成“三次握手”的过程，转为 ESTABLISHED 状态。在 ESTABLISHED 状态，客户端与服务器端可以通过函数 stud_tcp_recv ( )和 stud_tcp_send ( )进行数据交互。完成数据传输后，客户端通过 stud_tcp_close ( )函数发送关闭连接请求（FIN 报文），转入 FIN-WAIT1 状态；收到应答 ACK 后进入 FIN-WAIT2 状态，此时状态为半关闭，等待服务器端关闭连接。收到服务器端的 FIN 报文后，需要发送确认 ACK，状态改变为 TIME-WAIT；等待一段时间后，转为初始的 CLOSED 状态，连接完全断开。

具体实现见 TCP.cpp。

四．思考问题

本节主要介绍了基于 NetRiver 实验系统的简单 TCP 实验的相关内容。请在实验的基础上思考以下问题。

（1） 试总结 TCP 中序号的处理方式。

解：在 TCP 中，序号主要有序列号 seq 和确认号 ack，其中，一般情况下有 ack=seq+1，seq=ack，即自己发出的 ack 等于自己收到的 seq+1，自己发出的 seq 等于自己收到的 ack。

（2） 试比较 TCP 校验和的计算与前面 IPv4 收发/转发实验中校验和的计算有何异同。

解：TCP 校验和的计算与前面 IPv4 收发/转发实验中校验和的计算都是采用把被校验的数据按 16 位进行累加，然后取反码，若数据字节长度为奇数，则在这数据尾部补一个字节的 0 以凑成偶数。但是两者参与运算的数据不一样，TCP 校验和的计算不仅包含整个 TCP/UDP 数据报，还覆盖了一个虚头部；IPv4 收发/转发实验中校验和计算的数据包括 IPv4 头部的所有数据，但不包括载荷数据以及头部校验和本身。

五．实验总结

通过本次实验，让我对 TCP 协议有了更加深刻的理解，对建立连接时的三次握手，断开连接时的四次握手以及中间数据交互的过程有了非常直观的感受。本实验应该是所有实验中最复杂的一个，做起来非常纷繁复杂，既需要考虑状态的转换，也需要考虑数据的发送，异常状况的处理，一不小心就会出错，实在需要细心，耐心和用心，总体做下来还是收获蛮大的。