1 定义的结构和变量

1.1 gSrcPort、gDstPort、gSeqNum、gAckNum

Netriver中要求定义的变量，在初始化TCB的时候给它需要的SrcPort、DstPort、Seq和Ack。

1.2 typedef uint8_t STATE、enum TCPStatus

STATE是TCB中表示客户端目前的状态的变量的类型，TCPStatus则是客户端能有的所有状态集合。

1.3 typedef struct TCPHead

表示TCP头的结构，里面有TCP头部的各个信息和转换头部主机序和网络序的函数ntoh、hton，以及debug用的展示头部信息的函数display。

p.s. 由于TCP的校验和包括头部、伪首部和数据，所以我们在TCPHead中也有一个data元素，存放TCP的数据，即使它不属于头部。

1.4 typedef struct MyTCB

表示TCB的结构，根据Netriver的描述，里面存放了srcAddr，dstAddr，srcPort，dstPort，seq，ack，status和expectedAck（期待收到的ack回复），sockfd（套接口描述符）和data（TCB缓存区）。

1.5 TCBNum、TCBTable、curTCB

TCBNum是指目前拥有的TCB的个数

TCBTable存放指向目前拥有的TCB的指针

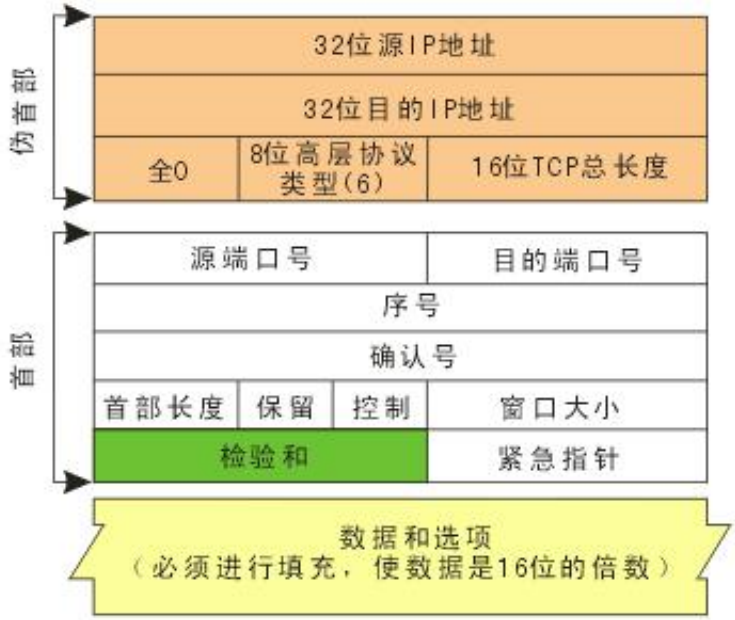
curTCB指针指向目前使用的TCB

2 函数及实现逻辑

2.1 unsigned int GetCheckSum(TCPHead* head, unsigned short len, unsigned int srcAddr, unsigned int dstAddr)

- 函数作用：计算TCP头部的校验和。
- 函数参数：head：指向TCP头部结构的指针；len：TCP帧的长度（包括数据）、srcAddr：源ip地址；dstAddr：目的ip地址
- 函数返回值：计算过的校验和

注意：TCP的校验和包括伪首部，伪首部的结构如下：



//填充字段值来自IP层 typedef struct tag_pseudo_header { u_int32_t source_address; //源IP地址 u_int32_t dest_address; //目的IP地址 u_int8_t placeholder; //必须置0,用于填充对齐 u_int8_t protocol; //8为协议号(IPPROTO_TCP=6,IPPROTO_UDP=17) u_int16_t tcplength; //UDP/TCP长度 }PseudoHeader_S;

这里的TCP总长度实际上是指TCP头的长度 + 数据部分长度，而且以字节为单位（不是4字节为单位了）！

2.2 int stud_tcp_input(char* pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dstAddr)

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 首先将pBuffer的内容拷贝到TCPHead中。
- 将TCPHead中头部部分的网络序转成主机序。
- 计算校验和，比较是否有问题。

  注意：这里计算的校验和是利用GetcheckNum得到的结果，不应该和0比，因为里面没有checkNum的值，而是直接和checkNum比，看是否相等，若相等则说明校验和正确。
- 检查序列号，和curTCB的ackExpect做比较
- 查看curTCB的status和TCPHead中的type，进行有限状态机的状态转换，每次转换后调用stud_tcp_output向服务端发送ack回复。

  注意：这里要进行curTCB部分数值的更新，ack应该是seqNum + len（数据长度）的值，而不是每次都是+1

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

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 先查看curTCB是否为空，若为空初始化curTCB的值
- 新建TCPHead，将pData中的数据拷贝到TCPHead的data段中，然后手动配置TCPHead的其他元素
- 将TCPHead的头部转成网络序
- 继续进行有限状态机的转换和curTCB中数值的更新
- 调用tcp_sendIpPkt发送IP packet

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

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 首先判断TCBNum是否为0，若为0向TCBTable中加入一个空指针，TCBNum + 1。这么做是因为我们用TCBNum初始化sockfd，但sockfd从1开始，所以我们这里要让TCBNum > 0
- 新建MyTCB，手动配置其元素值，并将指针加入TCBTable
- 返回sockfd

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

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 利用sockfd确定使用的curTCB
- 初始化curTCB中的源IP地址、源端口、目的IP地址、目的端口
- 调用stud_tcp_output发送SYN进行第一次握手
- 调用waitIpPacket等待服务端回复的第二次握手
- 收到回复pBuffer后交给stud_tcp_input，处理信息、状态机转换并进行回复完成第三次握手

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

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 利用sockfd确定使用的curTCB
- 判断TCB状态是否是ESTABLISHED
- 将pData中的数据存入curTCB的数据缓存区
- 调用stud_tcp_output发送数据到服务端
- 调用waitIpPacket等待服务端回复数据
- 收到回复pBuffer后交给stud_tcp_input，处理信息、状态机转换并进行回复

2.7 int stud_tcp_recv(int sockfd, unsigned char* pData, unsigned short datalen, int flags)

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 利用sockfd确定使用的curTCB
- 判断TCB状态是否是ESTABLISHED
- 调用waitIpPacket等待服务端给数据
- 收到回复pBuffer后交给stud_tcp_input，处理信息、状态机转换并进行回复

2.8 int stud_tcp_close(int sockfd)

函数作用、函数参数和函数返回值已经在指导书中说明，故这里只阐述实现逻辑。

- 利用sockfd确定使用的curTCB
- 判断TCB状态是否是ESTABLISHED
- 调用stud_tcp_output发送SYN_ACK数据给服务端第一次握手，进入状态FIN-WAIT1
- 调用waitIpPacket等待服务端回复数据进行第二次握手
- 收到回复pBuffer后交给stud_tcp_input，处理信息、状态机转换并进行回复完成第三次握手，进入状态FIN-WAIT2
- 调用waitIpPacket等待服务端发送数据FIN进行第四次握手
- 收到回复pBuffer后交给stud_tcp_input，处理信息、状态机转换并进行回复进入状态TIME-WAIT

3 实验过程的重点难点及遇到的问题

- 要学会打括号！ 移位运算优先级很低 指不定就和 + - 的顺序搞反了（debug感想）
- TCP伪首部的length是以字节为单位
- 网络序转主机序仅针对字节大小而言，所以对于一个字节的数据（headLen、Type等）就没必要调用ntoh、hton了（也没有适用的函数可以调用）
- 太奇怪了 stud_tcp_input的函数参数srcAddr和dstAddr要先从网络序转成主机序，在这里卡了很久....但是output以及之前所有lab中都没有需要处理srcAddr和dstAddr的情况，很疑惑是为什么。

4 感想和建议

这次试验和以往实验都不一样，难度和任务量都很大。

但确实在完成实验后更深入的体会到了TCP协议的过程。如果以后有机会的话，期待完成实验指导书中的Bonus部分，或许会对TCP协议有更深的理解和感悟。