计算机网络概论_第二次实验_Ipv4 收发

实验流程

1、 阅读 netRiver 实验指导书中关于滑动窗口部分的介绍。重点包括：frame 数据结构，frame_head 数据结构，frame_kind 枚举；三种滑动窗口测试函数在什么情况下被调用，以及传入的参数的意义，需要处理的情况（三种，包括 timeout，send，receive。其中 receive 还有 NAK 和 ACK 的区分），在各种情况下的处理要求。
2、 根据指导书中滑动窗口部分的介绍完成函数
3、 反复调试，直到成功通过测试

实验实现思路

本次实验数据结构非常简单，只要简单将 ip packet header 部分对应字段取出操作即可。重点在于校验和算法的实现。

问题

校验和并非 CRC 循环校验，而是 RFC1071 校验法。可能是上课讲到校验部分的时候开了小差，只记得到网络层之后，校验码可以软件实现，可是却没听到校验方法是 RFC1071 校验法，因此一开始按照 CRC 循环校验的步骤写了一下。

解决方法

上网查阅资料，发现 RFC1071 校验算法，一开始以为这是 CRC 的快速算法，后面渐渐发现这是两种不同的校验算法。因此参考了博客上关于 RFC1071 算法的解释和实现。

代码

```
#include "sysInclude.h"

extern void ip_DiscardPkt(char* pBuffer,int type);

extern void ip_SendtoLower(char*pBuffer,int length);

extern void ip_SendtoUp(char *pBuffer,int length);

extern unsigned int getIpv4Address();

int stud_ip_recv(char *pBuffer,unsigned short length)
{
    // get the variable
    char *tempIndex = pBuffer;
    unsigned short version = ((unsigned char)tempIndex[0]) >> 4;
    unsigned short IHL = tempIndex[0] & 15;
    // be careful to add 8 to index;
    tempIndex = tempIndex + 8;
    unsigned short ttl = ((unsigned char)tempIndex[0]);
    tempIndex += 2;
```

```
	unsigned int headerChecksum = ((unsigned short *)tempIndex)[0];
	tempIndex += 2;
	unsigned int srcip = ntohl(*((unsigned int *)tempIndex));
	tempIndex += 4;
	unsigned int dstip = ntohl(*((unsigned int *)tempIndex));

	if (version != 4){
		ip_DiscardPkt(pBuffer, STUD_IP_TEST_VERSION_ERROR);
		return 1;
	}
	if (IHL < 5){
		ip_DiscardPkt(pBuffer, STUD_IP_TEST_HEADLEN_ERROR);
		return 1;
	}
	if (ttl == 0){
		ip_DiscardPkt(pBuffer, STUD_IP_TEST_TTL_ERROR);
		return 1;
	}
	if (dstip != getIpv4Address() && dstip != 0xffffffff){
		ip_DiscardPkt(pBuffer, STUD_IP_TEST_DESTINATION_ERROR);
		return 1;
	}

	// Fast algo from RFC1071
	unsigned int checksum = 0;
	for (int i = 0; i < 20; i += 2){
		checksum += (pBuffer[i] & 0xff) << 8;
		checksum += (pBuffer[i + 1] & 0xff);
	}
	while (checksum >> 16)
		checksum = (checksum >> 16) + checksum;
	checksum = (~checksum) & 0xffff;
	if (checksum != 0){
		ip_DiscardPkt(pBuffer, STUD_IP_TEST_CHECKSUM_ERROR);
		return 1;
	}

	ip_SendtoUp(pBuffer, (int) length);
	return 0;
}

int stud_ip_Upsend(char *pBuffer,unsigned short len,unsigned int srcAddr,
		unsigned int dstAddr,byte protocol,byte ttl)
{
```

```
	char *ipPacket = new char[len + 20];

	memset(ipPacket, 0, len + 20);
	ipPacket[0] = 0x45; // version & headlen
	unsigned short totallen = htons(len + 20);
	// memcpy(buf + 2, &ttllen, sizeof(unsigned short));
	// use a code block
	{
		unsigned short *tmp = (unsigned short *)(ipPacket + 2);
		tmp[0] = totallen;
	}

	ipPacket[8] = ttl;
	ipPacket[9] = protocol;

	unsigned int srcadd = htonl(srcAddr);
	unsigned int dstadd = htonl(dstAddr);
	{
		unsigned int *tmp = (unsigned int *)(ipPacket + 12);
		tmp[0] = srcadd;
		tmp[1] = dstadd;
	}

	unsigned int checksum = 0;
	for (int i = 0; i < 20; i += 2){
		checksum += (ipPacket[i] & 0xff) << 8;
		checksum += (ipPacket[i + 1] & 0xff);
		checksum %= 65535;
		// printf("checksum: %x, num1: %x, num2: %x\n", checksum, (buf[i] & 0xff) << 8, (buf[i
+ 1] & 0xff));
	}
	unsigned short res = htons(0xffff - (unsigned short)checksum);
	{
		unsigned short *tmp = (unsigned short *)(ipPacket + 10);
		tmp[0] = res;
	}

	// copy the content in pBuffer to ipPacket content.
	memcpy(ipPacket + 20, pBuffer, len);
	ip_SendtoLower(ipPacket, (int)(len + 20));
	return 0;
}
```