IP收发实验报告

1800017781 程芷怡 元培学院

一、实验要求和接口

这个实验要求我们用两个函数 `int stud_ip_recv(char *pBuffer, unsigned short length)` 和 `stud_ip_Upsend(char *pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dst` 分别完成IP报文在版本号、TTL、headlen、校验和错误时的丢弃工作和正确接收工作，以及给定内容报文的正确发送工作。

系统给定的函数分别有：

`extern void ip_DiscardPkt(char *pBuffer, int type)` 丢弃IP报文， `pBuffer` 指针指向报文头部， `type` 说明丢弃原因。

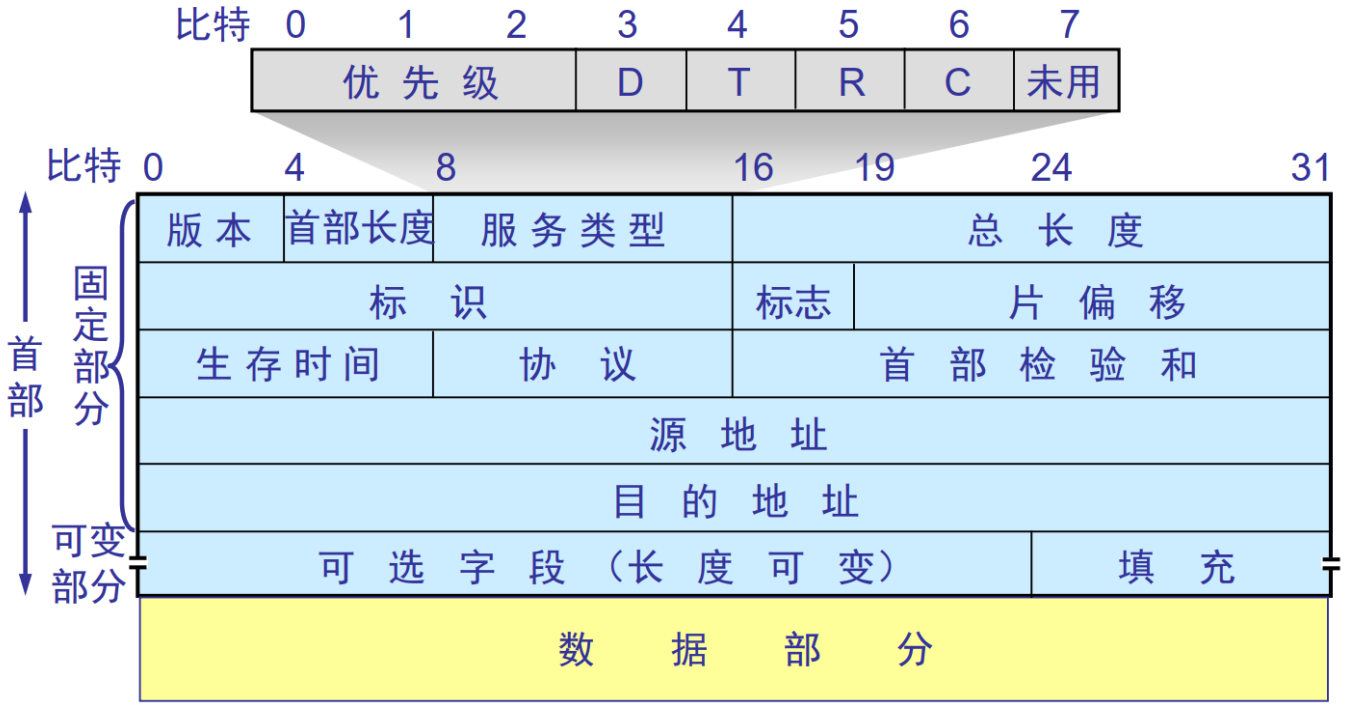
`extern void ip_SendtoLower(char *pBuffer, int length)` 发送IP报文。 `length` 指明报文总长度。

`extern void ip_SendtoUp(char *pBuffer, int length)` 把IP报文送交给上层。 `length` 指明报文总长度。

`extern unsigned int getIpv4Address()` 获取本机IP地址。

二、数据结构设计

- 由首部和数据两部分组成，首部占 20 字节



如图所示，由于IP控制协议要求报文分割一般以8bit的字节和16bit的字为单位，因此将 `pBuffer` 看作是字节数组，并避免了很多时候的 `htons` 、 `htonl` 、 `ntohs` 和 `ntohl` 的使用。

三、函数过程

1.IP接收

根据要求和上图所示，我们依次用字符操作和类型转换检查版本（为4）、头部长度（以 `unsigned long` 为单位，一定是5）和TTL（一定大于0）。由于据上图，各个数据的位置已经给定，因此不用进行大小端的转换。而在检查目的ip地址的时候要进行 `ntohl` 的转换。

最后是检查校验和。根据要求，以字为单位将每个位置的数据相加，之后取反还要加上超过16位的进位。这里的坑点在于， `sum` 是一个整型值，取反之后只有低16bit为0，需要用另一个 `unsigned short` 类型的数赋值或者判断 `sum` 是否为0xffff。

代码如下所示。

```
int stud_ip_recv(char *pBuffer, unsigned short length) {
    if (((unsigned int)pBuffer[0]&0xf0) != 0x40) {
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_VERSION_ERROR);
        return 1;
    }
    if ((((unsigned int)pBuffer[0])&0xf) < 5) {
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_HEADLEN_ERROR);
        return 1;
    }
    if ((unsigned int)pBuffer[8] == 0) {
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_TTL_ERROR);
        return 1;
    }
    if (ntohl(((unsigned long *)pBuffer)[4]) != getIpv4Address()) {
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_DESTINATION_ERROR);
        return 1;
    }
    unsigned int sum = 0;
    for (int i = 0; i < ((unsigned int)pBuffer[0]&0xf)*4; i += 2) {
        sum += (*(unsigned short *)(pBuffer + i));
    }
    unsigned short sum2 = (sum & 0xffff) + (sum >> 16);
    sum2 = ~sum2;
    if (sum2 != 0) {
        ip_DiscardPkt(pBuffer, STUD_IP_TEST_CHECKSUM_ERROR);
        return 1;
    }

    ip_SendtoUp(pBuffer+((unsigned int)pBuffer[0]&0xf)*4, length);
    return 0;
}
```

2.IP发送

根据上图，我们需要填充 `buffer` 数组的前20个字节，分别是版本号、头部长度、总长度、TTL、协议、源地址和目的地址。之后计算校验和，和上面类似，将计算得到的结果填入相应位置。这里因为有自动截断，所以没有上面那样的坑。

```
int stud_ip_Upsend(char *pBuffer, unsigned short len, unsigned int srcAddr, unsigned int dstAddr, byte protocol, byte ttl) {
    char buffer[len + 20];
    memset(buffer, 0, sizeof(buffer));

    buffer[0] = 0x45;
    ((unsigned short *)buffer)[1] = htons(len + 20);
    buffer[8] = ttl;
    buffer[9] = protocol;
    ((unsigned long *)buffer)[3] = htonl(srcAddr);
    ((unsigned long *)buffer)[4] = htonl(dstAddr);
    unsigned int sum = 0;
    for (int i = 0; i < 20; i += 2) {
        sum += htons(*(unsigned short *)(buffer + i));
    }
    sum = (sum & 0xffff) + (sum >> 16);
    sum = ~sum;
    ((unsigned short *)buffer)[5] = htons(sum);
    for (int i = 0; i < len; i++) {
        buffer[i+20] = pBuffer[i];
    }
    ip_SendtoLower(buffer, len+20);
    return 0;
}
```