1 定义的结构和变量

1.1 const int discardType[5]

用来存放丢弃该帧的原因，五种错误类型。

STUD_IP_TEST_CHECKSUM_ERROR IP 校验和出错

STUD_IP_TEST_TTL_ERROR TTL 值出错

STUD_IP_TEST_VERSION_ERROR IP 版本号错

STUD_IP_TEST_HEADLEN_ERROR 头部长度错

STUD_IP_TEST_DESTINATION_ERROR 目的地址错

1.3 typedef struct Window

```
typedef struct Window{ // 滑动窗口 用来缓存数据
    frame message;
    unsigned int size;
}
```

由于滑动窗口的大小是固定的，所以没有必要使用链表，直接用数组表示即可，我们这里定义滑动窗口数组的每个元素为Window。其中有一个存放frame的变量message和一个表示frame大小的变量size。

1.4 函数中的变量

将在每个函数中进行解释。

2 函数及实现逻辑

2.1 bool checkHeadSum(unsigned int* bufferHead, int length)

- 函数作用：对Ipv4分组头部的校验和进行检验。
- 函数参数：
    - sum表示校验和。反码算术运算求和：带进位的二进制加法运算，若最高位有进位，则结果+1。注意：最后一次运算若有溢出，就要回卷（在最低位+1）
    - bufferHead指Ipv4分组头部
    - length指bufferHead长度
- 函数的返回值：校验和是否有误

- 实现逻辑：反码算术运算求和：带进位的二进制加法运算，若最高位有进位，则结果+1，注意：最后一次运算若有溢出，就要回卷（在最低位+1）。

2.2 int checkSegment(unsigned int* pBuffer, unsigned int& headLen)

- 函数作用：对Ipv4分组头部进行检查，看是否属于五种错误类型中的一种。
- 函数参数：
  - bufferHead指Ipv4分组头部
- 函数的返回值：头部是否出现错误。
- 实现逻辑：先用ntohl将网络序转换成主机序，在分别检查校验和、TTL、IP版本号、头部长度和目的地址是否出错。

2.3 int stud_ip_recv(char* pBuffer, unsigned short length)

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。

- 先调用checkSegment检验是否出错。
- 如果出错，则根据错误类型调用discard函数丢弃该帧，并指明错误类型。
- 如果没有出错，调用SendtoUp函数传递给上一层。

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

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。

- 先调用malloc函数分配一定的存储空间给Buffer。
- 按照Ipv4分组头部的规则，和函数参数提供的信息，填充需要填充的位置和校验和。
- 注意将头部全部转成网络序。
- 最后将上一层的pBuffer内容全部拷贝到Buffer中，调用SendtoLower函数传递到下一层。

3 实验过程的重点难点及遇到的问题

- 关于网络序和主机序的转换：

3. 主机字节序与网络字节序的转换

```
#include <netinet.in.h>
unsigned long int htonl(unsigned long int hostlong);
unsigned short int htons(unsigned short int hostshort);
unsigned long int htonl(unsigned long int netlong);
unsigned short int htons(unsigned short int netshort);
```

htonl是按照int大小转换的，所以我们在将头部分组是指针类型是unsigned int *。

- 这里的位数是按照从右到左从低到高。

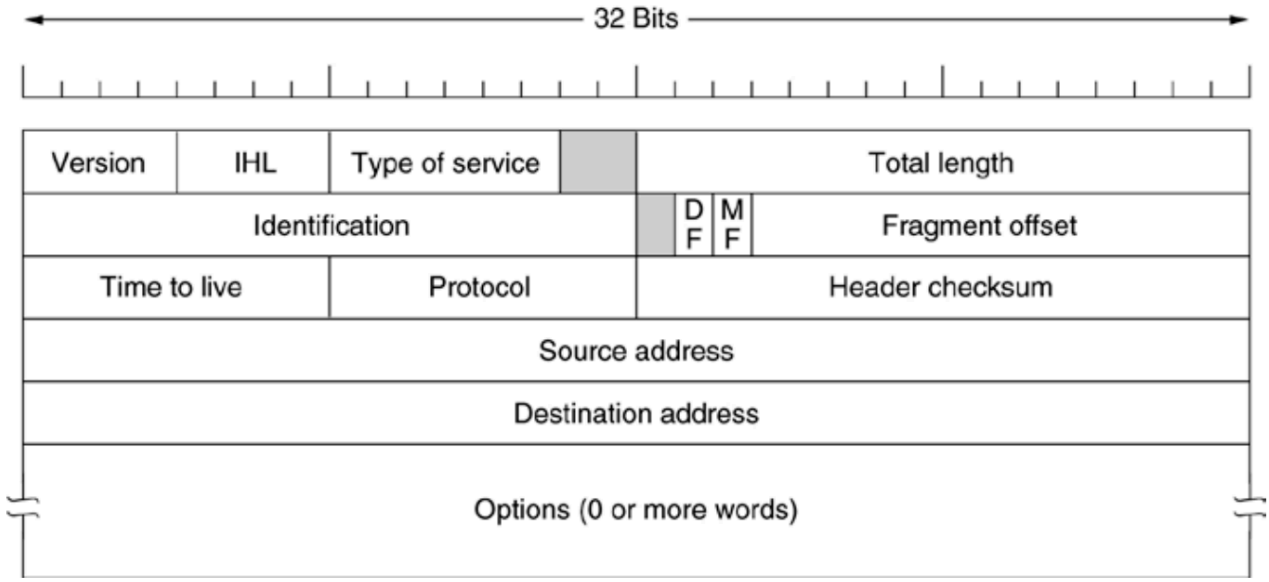

图 1.3　IPv4 分组头部格式

- 注意headLen是有固定的20字节，但下面也可能有可变字节，不能默认headLen就是20，要根据IHL中记录的数字确定headLen。
- 校验码要先取反再转为unsigned short，否则取反操作后结果默认为为short，结果不为0，没法和0比。
- TTL检验要转化成char检验是否 < 0。

4 感想和建议

个人感觉这个实验比滑动窗口实验要简单一些。但也让我深入了解了Ipv4协议的收发过程。而且在debug的过程中发现自己对于htonl等函数没有深度了解，即使是已经在滑动窗口实验使用过，也会因为大端字节和小端字节的问题纠结很久。