实验六：Ping 工具的实现

——1801090006 郭盈盈

2020.04.07

一、实验题目

请结合附件内容，编写一个程序，使其能够实现简单的 ping 功能，即**判断目标网站是否可以连接**，然后通过 Wireshark 进行抓包分析其 ICMP 协议，指出哪个数据包是 ping 的请求（request），哪个数据包是对这个请求的回应（reply）（如果 reply 数据包存在的话）。本次实验编程语言不限，提交时请上传**实验报告**、抓取的 **Wireshark 数据包**以及**源代码**（.cpp、.py 或.java 文件等，注意不要提交整个工程文件）。

二、相关知识

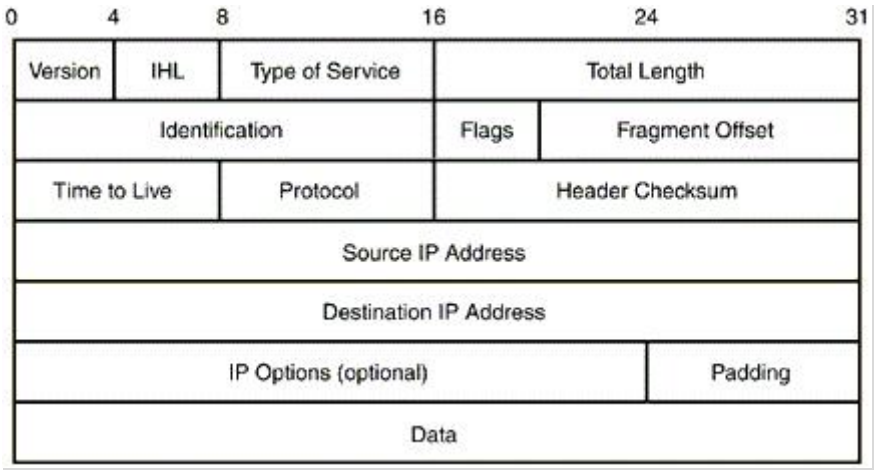
```
C:\Users\Administrator>ping eol.bnuz.edu.cn

正在 Ping eol.bnuz.edu.cn [59.38.32.70] 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。

59.38.32.70 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 0，丢失 = 4 (100% 丢失)，
```

由上面的执行结果可以看到，ping 命令执行后显示出被测试系统主机名和相应 IP 地址、返回给当前主机的 ICMP 报文顺序号、ttl 生存时间和往返时间 rtt（单位是毫秒，即千分之一秒）。要真正了解 ping 命令实现原理，就要了解 ping 命令所使用到的 TCP/IP 协议。ICMP(Internet Control Message,网际控制报文协议)是为网关和目标主机而提供的一种差错控制机制，使它们在遇到差错时能把错误报告给报文源发方。ICMP 协议是 IP 层的一个协议，但是由于差错报告在发送给报文源发方时可能也要经过若干子网，因此牵涉到路由选择等问题，所以 ICMP 报文需通过 IP 协议来发送。ICMP 数据报的数据发送前需要两级封装：首先添加 ICMP 报头形成 ICMP 报文，再添加 IP 报头形成 IP 数据报。由于 IP 层协议是一种点对点的协议，而非端对端的协议，它提供无连接的数据报服务，没有端口的概念，因此很少使用 bind()和 connect()函数，若有使用也只是用于设置 IP 地址。

ping 的过程是向目的 IP 发送一个 type=8 的 ICMP 响应请求报文，目标主机收到这个报文之后，会向源 IP（发送方，我）回复一个 type=0 的 ICMP 响应应答报文。

那上面的字节、往返时间、TTL 之类的信息取决于 IP 和 ICMP 的头部

IP 的头部



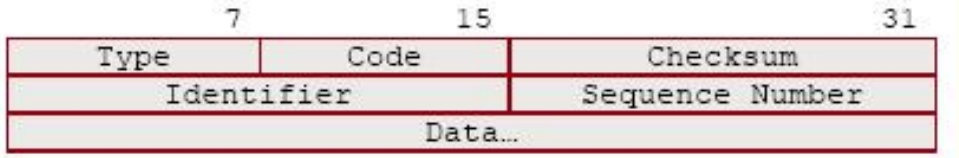
我们需要关注里面的几个信息：

首部长度(IHL)：因为 IP 的头部不是定长的，所以需要这个信息进行 IP 包的解析，从而找到 Data 字段的起始点。另外注意这个首部长度是以 4 个字节为单位的，所以首部实际长度是首部长度*4 字节。

TTL(生存时间)

数据(Data)：这部分是 IP 包的数据，也就是 ICMP 的报文内容。

ICMP 响应请求/应答报文头部



类型(Type)：type=8 表示响应请求报文，type=0 表示响应应答报文。

代码(Code)：与类型组合，表示具体的信息，参考这里。

检验和(Checksum)：这个是整个 ICMP 报文的检验和，包括类型、代码、...、数据。

标识符(Identifier)：这个一般填入本进程的标识符。

序号(Sequence Number）

一般而言，统计 ping 的往返时间的做法是，在 ICMP 报文的 Data 区域写入 4 个字节的时间戳。在收到应答报文时，取出这个时间戳与当前的时间对比即可。

三、实验步骤与结果分析

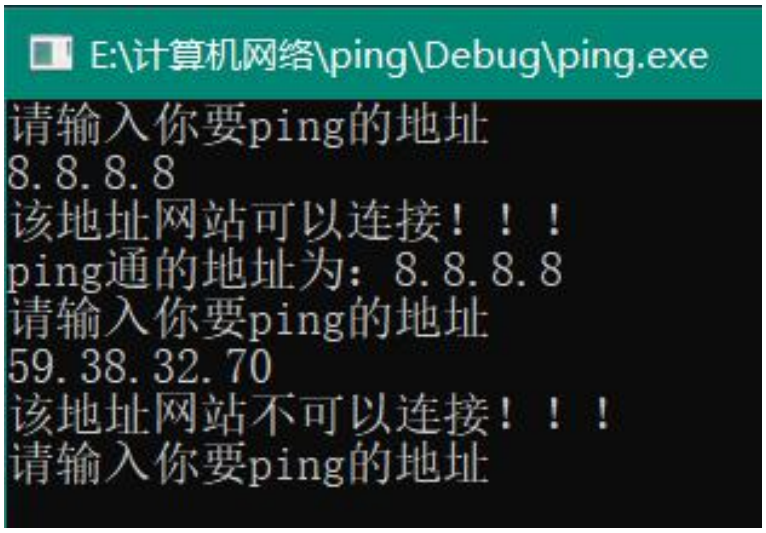
ICMP 结构体定义

```
struct icmp_header
  {
    unsigned char icmp_type; //信息类型
    unsigned char icmp_code;   //代码
    unsigned short icmp_checksum; //校验和
    unsigned short icmp_id;   //用来唯一标识此请求的 ID 号
    unsigned short icmp_suquence; //序列号
    unsigned long icmp_timestamp; //时间戳
  };
```

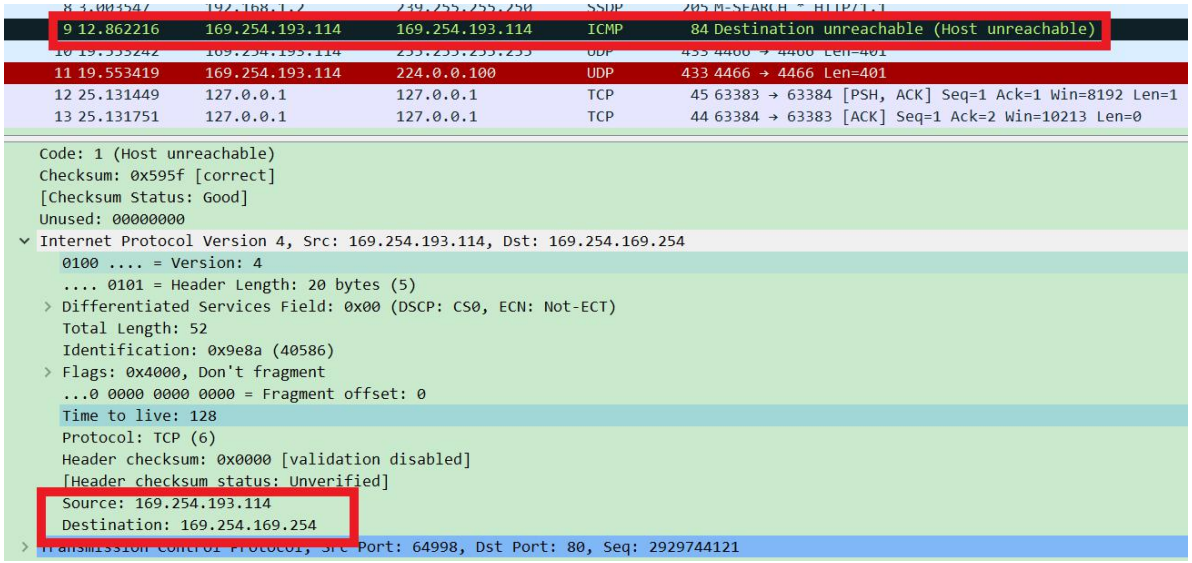
计算校验和

```
unsigned short chsum(struct icmp_header *picmp,int len){

    long sum=0;
    unsigned short *pusicmp=(unsigned short *)picmp;

    while(len>1)
    {
        sum += *(pusicmp++);
            if(sum & 0x80000000)
            {
                sum=(sum & 0xffff)+(sum>>16);
            }
            len-=2;
    }

    if(len)
    {
        sum += (unsigned short)*(unsigned char *)pusicmp;
    }

    while(sum>>16)
    {
        sum = (sum & 0xffff)+(sum>>16);
    }

    return (unsigned short)~sum;
}
```

运行结果



用 Wireshark 抓包



从截图中可以看到 ICMP 数据结构的值。

四、实验收获与总结

遇到的问题：

1. 无法打开包括文件:“stdafx.h”: No such file or directory
   解决方法：将“stdafx.h”换成“pch.h”（预编译头默认为“pch.h”）。
2. 未定义的标识符“_TCHAR*”
   解决方法：在头文件中添加#include <tchar.h>。
3. C4996 'inet_addr': Use inet_pton() or InetPton() instead or define _WINSOCK_DEPRECATED_NO_WARNINGS to disable deprecated API warnings ping
   解决方法：添加对_WINSOCK_DEPRECATED_NO_WARNINGS 的定义
   #define _WINSOCK_DEPRECATED_NO_WARNINGS 0
4. "char+*"+类型的实参与+"LPCWSTR"+类型的形参不兼容。
   解决方法：将配置属性->常规->字符集设置为：使用多字节字符集

收获：

通过对 ping 功能的实现，学习 ICMP 的结构和 IP 的首部结构，并解析他们。