Ping 程序

作者：吴玉波-berton

2015 年 12 月 23 日

## 一、设计目的

PING程序是我们使用的比较多的用于测试网络连通性的程序。PING程序基于ICMP，使用ICMP的回送请求和回送应答来工作。由计算机网络课程知道，ICMP是基于IP的一个协议，ICMP包通过IP的封装之后传递。

课程设计中选取PING程序的设计，其目的是通过PING程序的设计，能初步掌握TCP/IP网络协议的基本实现方法，对网络的实现机制有进一步的认识。

熟悉SOCKET的编程，包括基本的系统调用如SOCKET、BIND等；

## 二、设计内容

1、RAW模式的SOCKET编程

PING程序是面向用户的应用程序，该程序使用ICMP的封装机制，通过IP协议来工作。为了实现直接对IP和ICMP包进行操作，实验中使用RAW模式的SOCKET编程。

2、具体内容

1. 定义数据结构

定义IP数据报、ICMP包等相关的数据结构；

ICMP数据头结构：

typedef struct Icmp

{

unsigned char type; //类型

unsigned char code; //代码

unsigned short check\_sum; //检验和

unsigned short id; //标识符

unsigned short seq; sss//序列号

}IcmpHeader;

IP数据包头结构：

typedef struct iphdr

{

unsigned int headLen:4; //首部长度

unsigned int version:4; //版本

unsigned char tos; //区分服务

unsigned short totalLen; //总长度

unsigned short ident; //标识

unsigned short fragAndFlags; //标志与片偏移

unsigned char ttl; //生存时间

unsigned char proto; //协议

unsigned short checkSum; //检验和

unsigned int sourceIP; //源地址

unsigned int destIP; //目的地址

}IpHeader;

1. 程序实现

在LINUX环境下实现PING程序；

1. 程序功能
   1. ping ip 地址

如ping 192.168.1.1

* 1. ping 域名(进行DNS解析)

如ping www.baidu.com

* 1. 参数“ -n 数字”进行设置ping 的次数

如ping [www.baidu.com](http://www.baidu.com) –n 10

* 1. 参数 -t 无限循环

如ping 192.168.1.140 -t

* 1. 分析ping到的数据报

如最短时间，最长时间，平均时间和丢包率

6、ping ? 提供帮助提示

## 三、实验平台与语言

平台：linux 语言：C语言

## 四、功能模块实现

1. **总体设计方案**

流程图



主要代码：

// ICMP数据头结构

typedef struct Icmp

{

unsigned char type; //类型

unsigned char code; //代码

unsigned short check\_sum; //检验和

unsigned short id; //标识符

unsigned short seq; //序列号

}IcmpHeader;

//执行ping功能

int ping(const char \*ip, int send\_count)

{

int rawfd;

struct sockaddr\_in dest\_adr;

char icmp\_data[1024];

int size = sizeof(IcmpHeader)+32;

int r, i = 0, send, recv=0, lost=0;

char recv\_buf[1024];

int all\_time[1024] = {0};

//创建原始套接字

rawfd = socket(AF\_INET, SOCK\_RAW, IPPROTO\_ICMP);

if(rawfd == -1)

{

perror("create socket failed!");

return -1;

}

//设置目的地址与端口

dest\_adr.sin\_family = AF\_INET;

dest\_adr.sin\_port = htons(80);

inet\_aton(ip, &dest\_adr.sin\_addr);

//封装icmp数据包

pack\_icmp(icmp\_data, size);

printf("\n正 Ping %s 具有%d个字节的数据：\n", ip, size-sizeof(IcmpHeader));

if(send\_count == LOOP)

{

//无限循环

while(1)

{

if(ping\_one(rawfd,dest\_adr, ip, icmp\_data , all\_time) != -1)

{

recv++;

}

send++;

}

}

else

{

for(i = 0; i<send\_count; i++)

{

if(ping\_one(rawfd,dest\_adr, ip, icmp\_data , all\_time) != -1)

{

recv++;

}

send++;

}

}

printf("\n%s 的 Ping 统计信息：\n 数据包：已发送 = %d，已接收 = %d, 丢失

= %d<%.1f%% 丢失>, \n",ip, send, recv, lost, ((float)lost/(float)send)\*100);

printf("往返行程的估计时间：\n 最短=%dms 最长=%dms 平均 = %dms \n\n"

,min(all\_time, send), max(all\_time, send), average(all\_time, send));

close(rawfd);

return 0;

}

1. DNS域名解析功能实现

⑴、DNS 服务器

地址：202.96.134.133 端口：53

#define DNS\_PORT 53

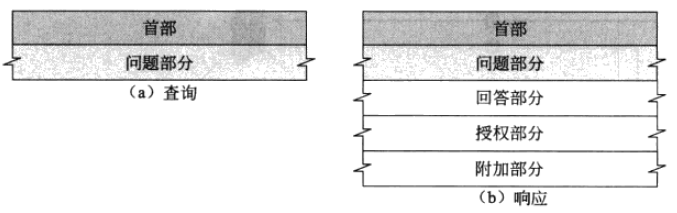
#define DNS\_IP "202.96.134.133"

#define DNS\_IP2 "8.8.8.8"

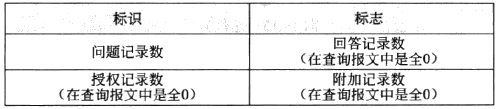
⑵、DNS 的实现基础：

通过UDP发送查询报文给DNS 服务器，然后从服务通过UDP 返回的回应报文中解析得到对应域名的IP。

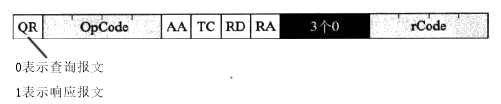
⑶、DNS 报文格式：



⑷、首部格式：



**其中标志字段：**16位



定义首部结构体：

typedef struct DNSheader

{

unsigned short id;

unsigned char qr\_opcode\_aa\_tc\_rd;

unsigned char ra\_zero\_rcode;

unsigned short qdcount;

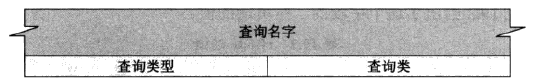
unsigned short ancount;

unsigned short nscount;

unsigned short arcount;

}DnsHeader;

⑸、问题记录格式：



**查询名字：**域名的可变长字段；其中计数字段指明每一节中的字符数。

**查询类型：**16位；值的意义如下表，查询ip 时为1

**查询类别：**16位；定义使用DNS的特定协议， 一般为1

查询名字打包代码：

//把域名打包成dns数据报的数据部分 如（3www5baidu3com）

//计算每段的数量

for(i = 0; i < name\_len; i++)

{

if(netname[i] == '.')

{

len[flg++] = i - s;

s += len[flg-1]+1;

}

}

len[flg] = i - s;

i = 0;

flg = 0;

data[i++] = len[flg++];

//加入每段字节的数量

for(; i < name\_len+1; i++)

{

if(netname[i-1] == '.')

{

data[i] = len[flg++];

}

else

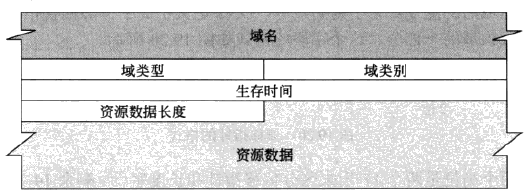
{

data[i] = netname[i-1];

}

}

**⑹、资源记录格式**



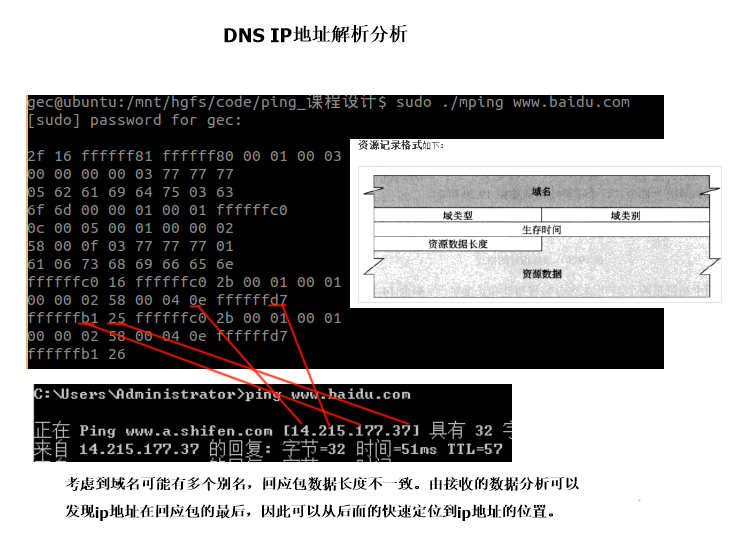
**资源数据：**可变长；值内容取决于类型字段的值，可以是数值、域名、偏移指针、字符串

**⑺、在资源数据中提取ip和原域名**：

**A、失败的解决方法：**

由于没有查到详细的解析资料，因此通过对整个报文每个字节进行分析，发现其格式的规律：ip地址放在报文的末尾，可以通过指针快速定位。

采样分析的域名有（[www.baidu.com](http://www.baidu.com) 和 www. sina.com），下图是对[www.baidu.com](http://www.baidu.com)的分析



代码如下：

//直接定位ip地址

Ipadr \*ip =(Ipadr \*) (sen\_buf+(r-8-4-8-2));

//如果ip地址长度不为4, 则返回

if(ip->len != 0x0400)

{

return -1;

}

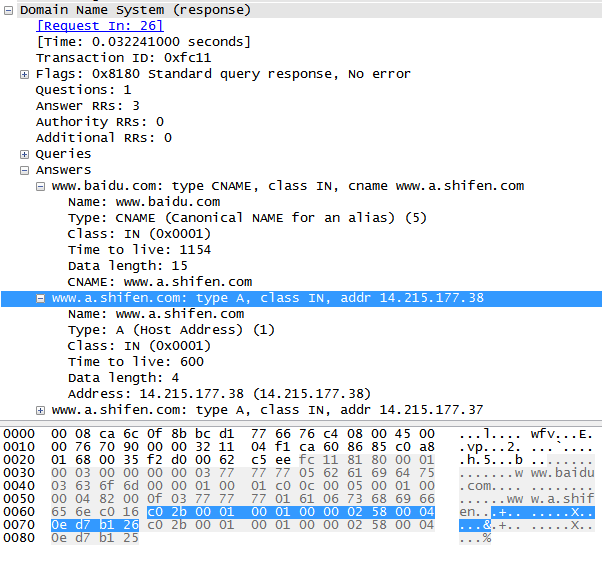
//次ip地址转为字符串

sprintf(get\_ip, "%u.%u.%u.%u", ip->a, ip->b, ip->c, ip->d);

失败原因：过于投机取巧，取得两个分析对象不够特殊，对DNS回答格式完全不解

**B、新的解决方法：**

通过用wireshark抓取DNS包进行分析，由于分析次数过多，此处以[www.baidu.com](http://www.baidu.com)为例



由抓取的dns包分析可得到回答部分的格式如下：

answer1：

name : 不定长，c0 0c指段偏移地址

type: 16 位 0005 是别名answer

class: 16位， 0001

ttl：32位

data length: 16位

cname： 别名，长度data length

answer 2

name：c0 2b 指向别名

type: 16位 0001 是ip answer

class : 16位 0001

ttl： 32位

data length：16位

address: 4个字节

answer 3

与回应2相似 ip 不一样

**ii、进一步分析**：

可以发现回答部分有两个类型（只是本设计的情况，DNS有很多种类型），type 字段为5时为别名回应，为1 时为ip回应，因此通过type和上面得到的格式来进行ip和别名的获取。

其中还发现，DNS为了减小数据报文的长度，回应部份重复的字段会省略，并通过偏移指针指向重复部分，由上面的分析可知是用c0（代替字段长度）来转义下一字节为偏移指针。

**iii、获取代码设计：**

int parse\_dns\_respone(unsigned char \*recv\_buf,unsigned char \*\*answer\_o, int data\_len ,char

\*get\_ip, const char \*netname)

{

int asw\_type;

int i,j, k;

static int fn = 0, fi = 0;

int r;

unsigned char cname[40];

unsigned char \*answer = \*answer\_o;

answer +=2;//查询域名

asw\_type = ntohs(\*((unsigned short\*)answer));

answer +=2;//type

answer +=2;//class

answer +=4;//ttl

if(asw\_type == 5) //域名包

{

bzero(cname, sizeof(cname));

//解析域名

parse\_dns\_name(recv\_buf, &answer, cname);

fn = 1; //标记已取得别名

}

else if(asw\_type == 1) //ip 回应包

{

//解析IP

parse\_dns\_ip(&answer ,get\_ip);

if(fn == 1)

printf("\n---- %s(%s)",cname , get\_ip);

else

printf("\n---- %s(%s)",netname , get\_ip);

fi = 1; //标记已取得IP

}

\*answer\_o = answer;

if(fn != 0 && fi != 0) return 2; //已取得别名和域名。则结果

else if(fi != 0) return 1;

return 0;

}

//解析IP

void parse\_dns\_ip(unsigned char \*\*answer\_o ,char \*get\_ip)

{

unsigned char \*answer = \*answer\_o;

Ipadr \*ip =(Ipadr \*)answer; //取提ip

sprintf(get\_ip, "%u.%u.%u.%u", ip->a, ip->b, ip->c, ip->d);

\*answer\_o = answer;

}

//解析域名

void parse\_dns\_name(unsigned char \*recv\_buf, unsigned char \*\*answer\_o, unsigned char \*cname)

{

int tmp\_len;

int d\_length;

int i, k;

unsigned char \*answer = \*answer\_o;

d\_length = ntohs(\*((unsigned short\*)answer));//总长度

answer +=2; //data length

i=0;

for(k=0; k<d\_length; k++)

{

tmp\_len = \*answer++; //名字段长度

if(i != 0)

cname[i++]='.';

if(tmp\_len == 0xc0) //CO转义为复字段

{

int tmp = \*answer++; //获得偏移指针

k ++;

i--;

while(1)

{

tmp\_len = recv\_buf[tmp++];//跳转到偏移位置

if(tmp\_len == 0) break; //偏移位置结果

cname[i++]='.';

//填充域名

get\_seg\_name(&recv\_buf[tmp], cname+i, tmp\_len);

tmp += tmp\_len;

i += tmp\_len;

}

continue;

}

if(tmp\_len == 0) break;

//填充域名

get\_seg\_name(answer, cname+i, tmp\_len);

k += tmp\_len;

i += tmp\_len;

//移动指针

answer += tmp\_len;

}

cname[i] = '\0';

\*answer\_o = answer;

}

**⑻、遇到的问题：**

* 1. 有多段连续的域名字段重复：

原来只考虑到有一个域名字段是重复的，但是有些是有多段连续的域名字段重复的，解决的方法是（直到len字节为0 才认为没有重复字段了）：

while(1)

{

tmp\_len = recv\_buf[tmp++];//跳转到偏移位置

if(tmp\_len == 0) break; //偏移位置结果

cname[i++]='.';

//填充域名

get\_seg\_name(&recv\_buf[tmp], cname+i, tmp\_len);

tmp += tmp\_len;

i += tmp\_len;

}

* 1. 有多个域名回答：

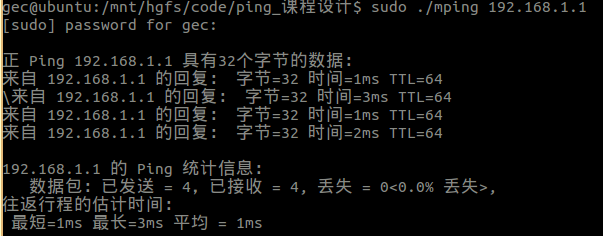
原来只考虑到只有一个域名名回答，但有些（如[www.sina.com](http://www.sina.com)）是有多个域名的，因此解决方法是域名回答可以多次解析，只有取得IP地址才结束，而不是原来只解析两个回答就认为拿到了IP地址

**⑼、本实现的不足：**

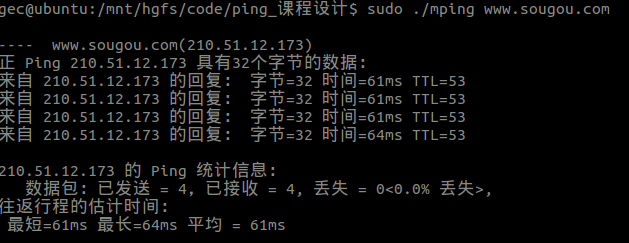
* + 没有完全掌握DNS 回应报文的格文。
  + 服务器单一，没有备用服务器。
  + 还存在未知域名不能解析的，未能确保能解析所有正确域名

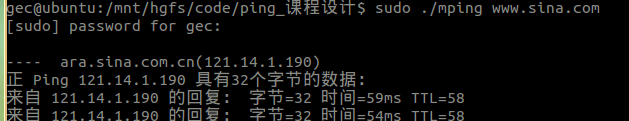
### （五）、结果分析

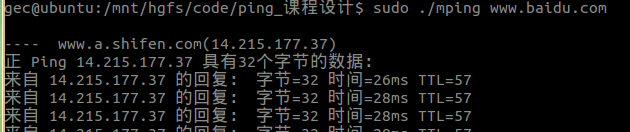
**Ping ip 地址 如 ping 192.168.1.1**

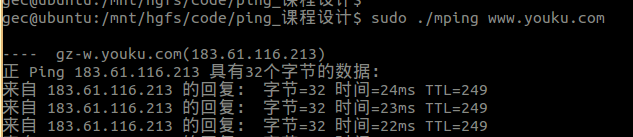


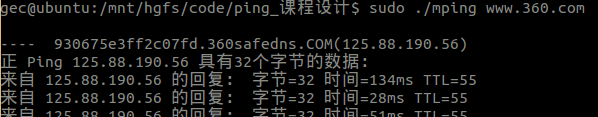
**Ping 域名 如：ping www.sougou.com**



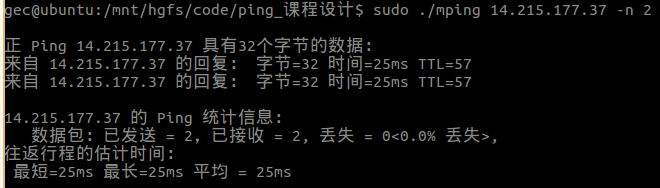








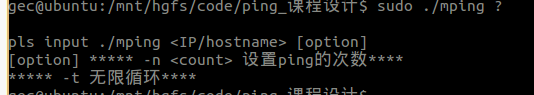
**Ping –n 如：ping 14.215.177.37 –n 2**

****

**Ping –t 如： ping** [**www.baidu.com**](http://www.baidu.com) **-t**



**Ping ?**



### （六）、心得体会

做的永远比想象中的难，修改了多次代码，刚开始想只要IP 地址，要愿意深入分析回应数据报，只是进过一些特例来定位ip 地址就好，想不到两个特例效果是一样的（[www.baidu.com](http://www.baidu.com)和[www.sina.com](http://www.sina.com)）, ip 地址都是在末尾前几个字节，但是，由于DNS 的去掉重复的功能，造成只有是只有2个IP 地址的域名才能有效，多于或少于两个的都不行，因此又得花时间重新进行分析，然后才想到利用抓包软件来协助分析，又花了不少心血修正这个BUG，程序员真不好做，坐到腰酸背痛。

但是，还情事还没那么顺利，拿多几个域名来试之后又发现了问题，具有多个别名的域名没办法正确解析，在原来的基础上又很难修改，因此决定把代码封装起来，封装多几个函数，然后在封装好的基础上解决多个别名的问题，但是在调试中又出现多个连绵的域名字段重复省略导致解析也来的域名完全的问题，又进行了一番修改。真不容易。

总结： 1.DNS的格式还需要找相关资料来学习。

2.以后写代码如果不是用于学习的，则应该找完成的框架进行修改，这样可以省时间，且写出来的程序也会比较稳定。

3.写程序前最好先做出详细方案，避免一些BUG。

4.网络知识的学习还有很多要学，网络编程要学的知识更加多。