Ping（gcc实现）

1、实验思路

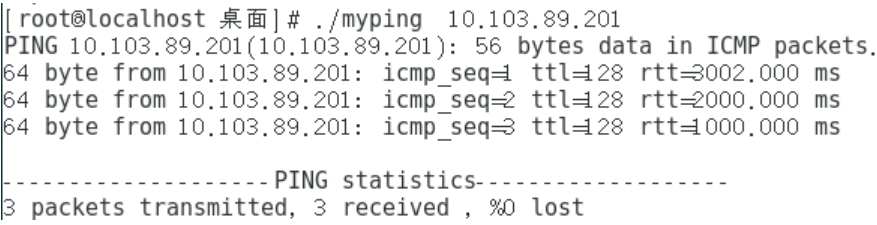
本实验使用gcc环境模拟实现Ping的功能，由于Ping是基于ICMP报文的，所以我们需要自己构造ICMP报文，由于牵扯到路由选择的问题，要将ICMP报文通过IP协议来发送。向对方发送使用sendto，接收使用recvfrom。ICMP报文的类型分为ICMP\_ECHO和ICMP\_ECHOREPLY，分别对应0和8，在构造ICMP报文发送时，需要指定icmp->type=ICMP\_ECHO和序号，同时使用校验和函数计算校验和并填充校验和字段。在接收对方的回应报文时，如果ICMP的type字段为ICMP\_ECHOREPLY以及序号等于我们定义的序号，则表明接收到的是自己发的ICMP的回应。然后将接收到的报文分别转换为IP格式和ICMP格式（跳过IP头部），通过IP结构体可以得到ttl，通过ICMP结构体可以得到seq序号。rtt的获取可以使用timeval结构体来计算得出。

2、实验环境及步骤

* Linux下gcc环境
* gcc -o myping myping.c
* ./myping 10.103.89.201（这是我的主机IP）
* ./myping 192.168.76.33（不存在的IP）

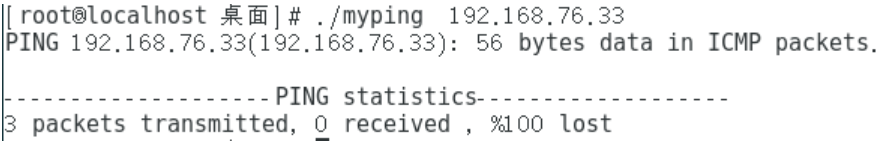
2、实验结果及分析

**1、./myping 10.103.89.201**



ping主机IP可以ping通，seq为自己设置的3个序号，信息显示均正确。

**2、./myping 192.168.76.33**



ping不存在的IP不能ping通。

从实验结果可以看出，该实验模拟实现了ping的功能。

4、实验程序

* **ICMP结构体**

**typedef** struct \_icmphdr**{**

unsigned char i\_type**;** //8位类型, 本实验用 8: ECHO 0:ECHO REPLY

unsigned char i\_code**;** //8位代码, 本实验置零

unsigned short i\_cksum**;** //16位校验和, 从TYPE开始,直到最后一位用户数据,如果为字节数为奇数则补充一位

unsigned short i\_id **;** //识别号（一般用进程号作为识别号）, 用于匹配ECHO和ECHO REPLY包

unsigned short i\_seq **;** //报文序列号, 用于标记ECHO报文顺序

unsigned int timestamp**;** //时间戳

**}**ICMP\_HEADER**;**

* **总体代码**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netinet/ip.h>

#include <netinet/udp.h>

#include <netinet/tcp.h>

#include <sys/types.h>

int main**()** **{**

int sock**,** bytes\_recieved**,** fromlen**,**n**,**id**=**1**,**on**=**1**,**s**;**

unsigned char buffer**[**65535**];**

struct sockaddr\_in from**;**

struct ip **\***ip**;**

struct tcphdr **\***tcp**;**

/\* 建立原始TCP包方式 收到IP+TCP信息包 \*/

sock **=** socket**(**AF\_INET**,** SOCK\_RAW**,**IPPROTO\_TCP**);**

printf**(**" The IPPROTO\_TCP value is %d \n"**,**IPPROTO\_TCP**);**

**if** **(**sock**>**0**)** **{**

printf**(**"Prepare caught!! \n"**);**

**}** **else**

**return(**0**);**

id**=**1**;**

**while(**1**)** **{**

fromlen **=** **sizeof(**from**);**//struct sockaddr\_in from

bytes\_recieved **=** recvfrom**(**sock**,** buffer**,** **sizeof(**buffer**),**0**,(**struct sockaddr **\*)&**from**,** **&**fromlen**);**//int bytes\_recieved

**for(**int i**=**0**;** i**<**bytes\_recieved**;** i**++)** printf**(**"%02X "**,**buffer**[**n**]);**

**if** **(**bytes\_recieved**>**0**)** **{**

printf**(**"OK!Start analytic data packet!\n"**);**

ip **=** **(**struct ip **\*)**buffer**;**

/\* tcp从整个IP/TCP buffer + (4\*ip->ip\_length) 地址处开始 \*/

tcp **=** **(**struct tcphdr **\*)(**buffer **+** **(**4**\***ip**->**ip\_hl**));**

**if** **(**ntohs**(**tcp**->**dest**)!=**23 **)** **{** /\*23为Telnet端口，也可改为其它端口\*/

printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\* ^TCP^ \*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**

printf**(**"\n ID=::: %d\n"**,**id**);**

printf**(**"Bytes received ::: %5d\n"**,**bytes\_recieved**);**

printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\* IP info begin\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**

printf**(**"IP header length ::: %d\n"**,**ip**->**ip\_hl**);**

printf**(**"IP sum size ::: %d\n"**,**ntohs**(**ip**->**ip\_len**));**

printf**(**"Protocol ::: %d\n"**,**ip**->**ip\_p**);**

printf**(**"IP\_source address ::: %s \n"**,**inet\_ntoa**(**ip**->**ip\_src**));**

printf**(**"IP\_dest address ::: %s \n"**,**inet\_ntoa**(**ip**->**ip\_dst**));**

**for** **(**n**=**0**;** n**<**4**\***ip**->**ip\_hl**;** n**++)** printf**(**"%02X "**,**buffer**[**n**]);**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* IP info end\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* TCP info begin\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**

printf**(**"Source port ::: %d\n"**,**ntohs**(**tcp**->**source**));**

printf**(**"Dest port ::: %d\n"**,**ntohs**(**tcp**->**dest**));**

**if(**ntohl**(**tcp**->**ack\_seq**)** **<** 0**){**

printf**(**"seq ::: %d\n"**,**ntohl**(**tcp**->**seq**));**

**}else{**

printf**(**"seq ::: %d\n"**,**ntohl**(**tcp**->**seq**));**

printf**(**"ack::: %d\n"**,**ntohl**(**tcp**->**ack\_seq**));**

**}**

printf**(**"Source address ::: %s\n"**,**inet\_ntoa**(**from**.**sin\_addr**));**

/\* 列出收到的TCP信息内容 一般TCP头20个字节 \*/

printf**(**"TCP 20 bytes is after the transfer of information:\n"**);**

**for** **(**n**=(**4**\***ip**->**ip\_hl**+**20**);** n**<**ntohs**(**ip**->**ip\_len**);** n**++)** **{**

printf**(**"%02X "**,**buffer**[**n**]);**

**if** **(**n**>**65**)** **break;**

**}**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* TCP info end\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**

id**=**id**+**1**;**

**}** /\*>23 end \*/

**}** /\*>0 end \*/

**}** /\*while end \*/

**}**