沈阳航空航天大学

**课 程 实 验 报 告**

课程名称：**网络程序设计**

实验题目：**实验二 利用原始套接字实现ping程序**

院（系）：计算机学院

专 业： 软件工程

班 级：1934010501

学 号：193401050107

姓 名：解雨洋

指导教师：王岩

完成时间：2022年4月30日

## 实验二 利用原始套接字实现ping程序

### 1、实验目的

熟悉并掌握利用原始套接字进行网络通讯程序设计的原理和方法。

### 2、实验内容

利用原始套接字实现Ping程序的编写。主要包含的功能有：

(1) 目的地址当程序运行后由用户指定；

(2) 仿照ping命令的-l length参数，自定义发送数据包的大小，也就是发送由length指定大小的ECHO数据包，默认为32字节，最小值为0，最大值是65500字节。

(3) 仿照ping命令的-n count参数，发送count指定的ECHO数据包数，默认值为4。

运行结果显示如下内容:

(1) 接收到的数据包的来源地址

(2) 数据包的顺序号

(3) 数据报的传输时间

(4) 生存周期

(5) IP包总长度

(6) 实际接收的字节数

(7) ECHO数据包设置的字节数。

### 3、工具

Golang 1.15.6

Goland 2020.3

### 4、程序代码

package main

import (

"bufio"

"bytes"

"encoding/binary"

"fmt"

"net"

"os"

"strconv"

"strings"

"time"

)

const (

MAX\_PG = 65500 //数据包最大字节数

)

//ICMP首部

type ICMP struct {

Type uint8 //类型

Code uint8 //代码

Checksum uint16 //检验和

Identifier uint16 //标识符

SeqenceNum uint16 //序号

}

var (

originBytes []byte // 数据包

)

func init() {

originBytes = make([]byte, MAX\_PG)

}

//ICMP校验和

func CheckSum(data []byte) (rt uint16) {

var (

sum uint32

length int = len(data)

index int = 0

)

for length > 1 { //2个字节凑一行二进制加法（模2和），保留溢出所以用32位加法

sum += uint32(data[index])<<8 + uint32(data[index+1])

index += 2

length -= 2

}

if length > 0 {

sum += uint32(data[index]) << 8 //剩余不够凑16个字节的放到高八位，后面补0

}

rt = uint16(sum) + uint16(sum>>16) //低16位与高16位相加

return ^rt //按位取反

}

func Ping(domain string, length, Count int) {

var (

icmp ICMP

laddr = net.IPAddr{IP: net.ParseIP("0.0.0.0")} // 得到本机的IP地址结构

raddr, \_ = net.ResolveIPAddr("ip", domain) // 解析域名得到 IP 地址结构

)

conn, err := net.DialIP("ip4:icmp", &laddr, raddr)

if err != nil {

fmt.Println(err.Error())

return

}

defer conn.Close()

icmp = ICMP{8, 0, 0, 0, 0} //类型：8，代码：0 ，为ping命令的请求

var buffer bytes.Buffer //大端构造ICMP报文

binary.Write(&buffer, binary.BigEndian, icmp)

binary.Write(&buffer, binary.BigEndian, originBytes[0:length])

b := buffer.Bytes()

binary.BigEndian.PutUint16(b[2:], CheckSum(b)) //放入首部校验和

fmt.Printf("\n正在 Ping %s [%s] 具有 %d字节的数据:\n", domain, raddr.String(), length)

recv := make([]byte, 1024)

for i := 1; i <= Count; i++ {

//向目标地址发送二进制报文包

if \_, err := conn.Write(b); err != nil {

fmt.Println("请求超时。")

continue

}

// 记录当前得时间

t\_start := time.Now()

conn.SetReadDeadline((time.Now().Add(time.Second \* 3)))

len, err := conn.Read(recv)

if err != nil {

fmt.Println("请求超时。")

continue

}

t\_end := time.Now()

dur := t\_end.Sub(t\_start).Milliseconds()

var TTL = recv[8]

fmt.Printf("来自 %s 的回复: 顺序号 = %d 时间 = %dms TTL = %d IP包总长度 = %d 实际字节数 = %d ECHO数据包字节数 = %d\n", raddr.String(), i, dur, TTL, len, len-20, len-28)

time.Sleep(time.Second)

}

}

func main() {

var length, count = 32, 4

reader := bufio.NewReader(os.Stdin)

res, \_, err := reader.ReadLine()

if err != nil {

fmt.Println("reader.ReadLine() error:", err)

}

s := strings.Split(string(res), " ")

for i, v := range s {

if v == "-l" {

length, \_ = strconv.Atoi(s[i+1])

if length < 0 {

fmt.Println("错误: 最小值为0")

return

}

}

if v == "-n" {

count, \_ = strconv.Atoi(s[i+1])

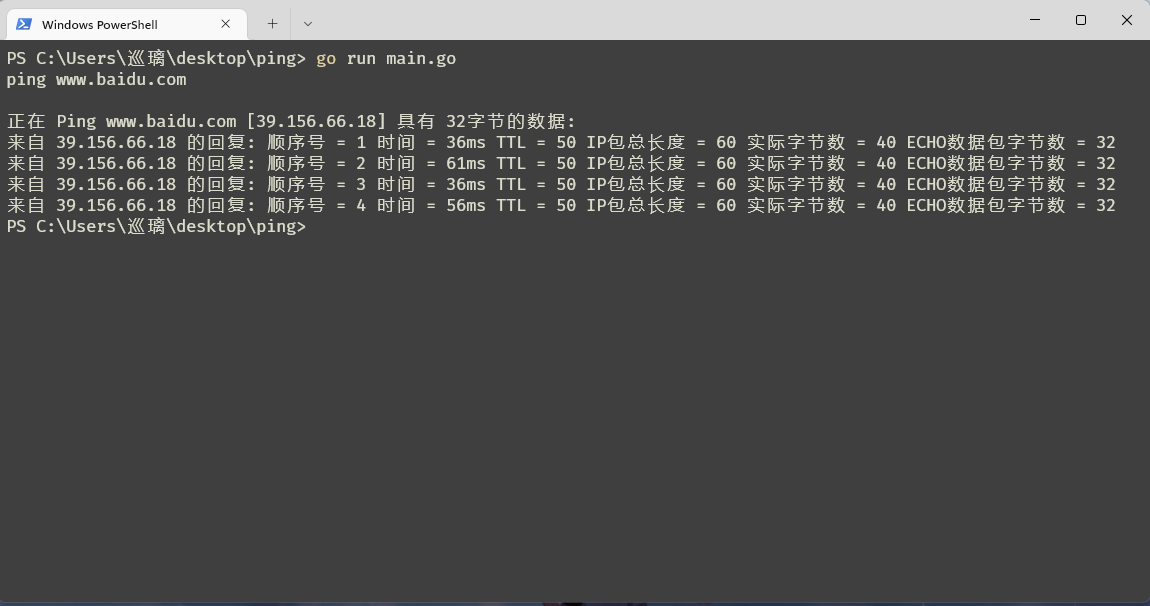
}

}

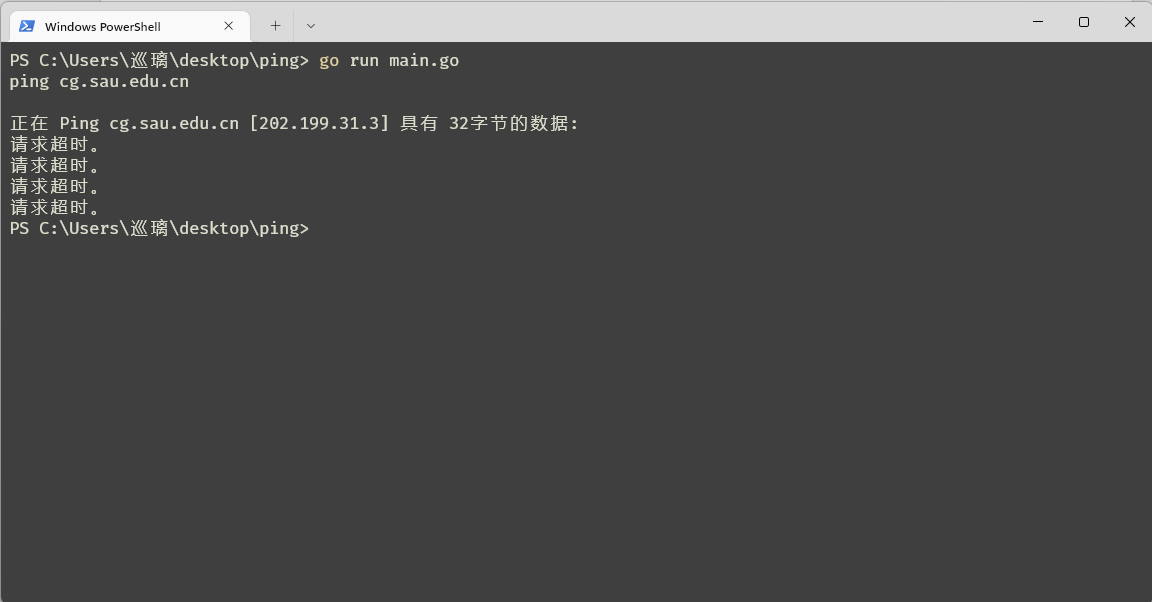
Ping(s[1], length, count)

}

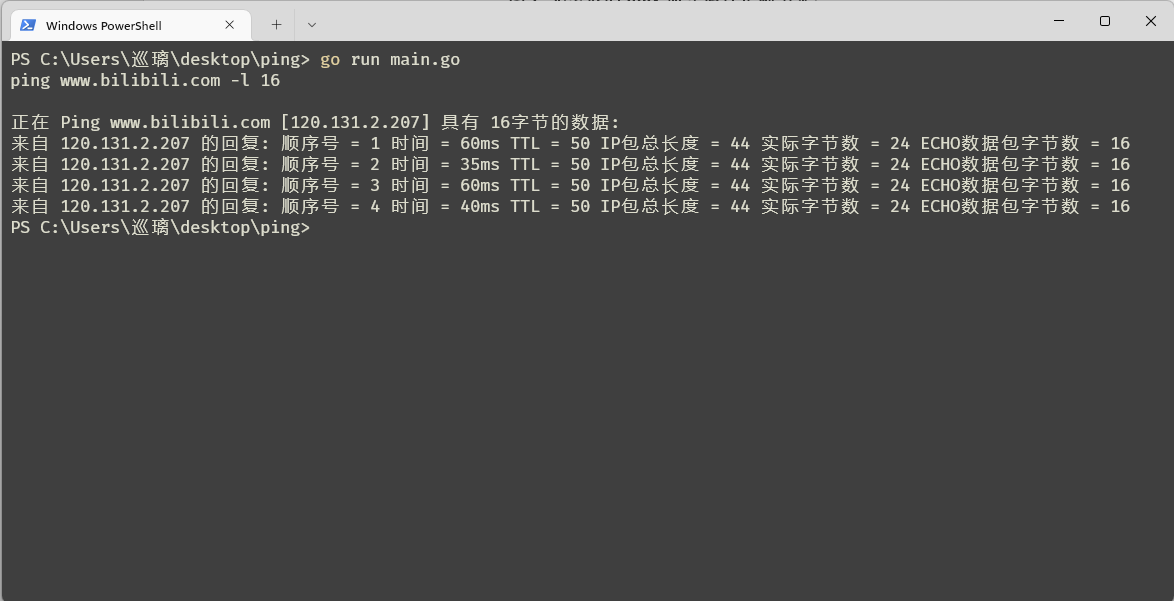
### 5、程序运行结果



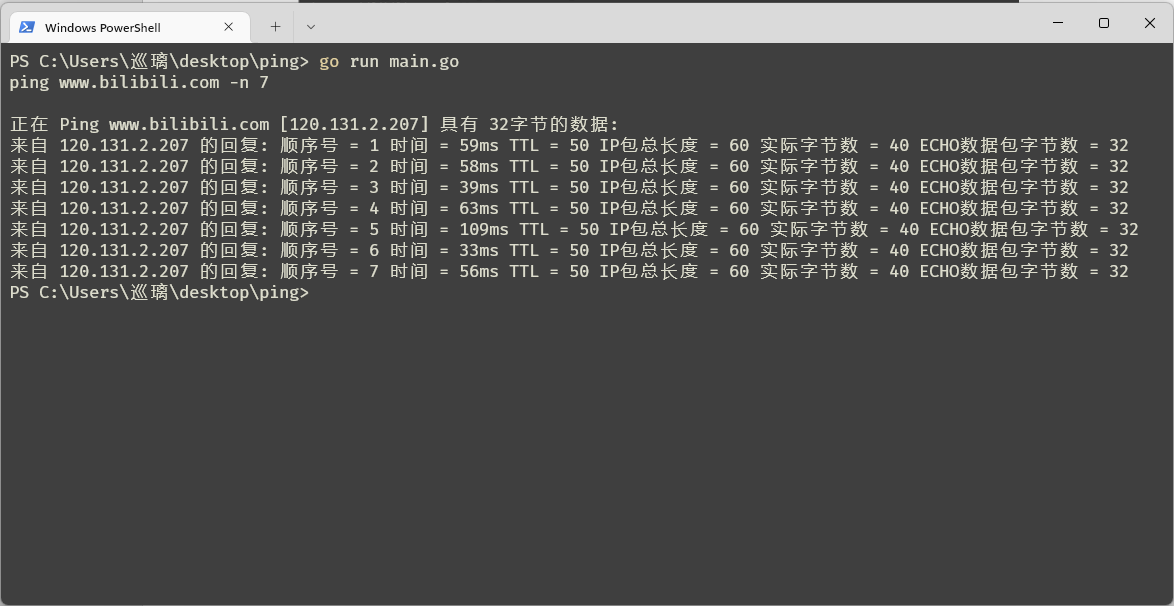
**图1 无参数的ping命令运行正确结果**

****

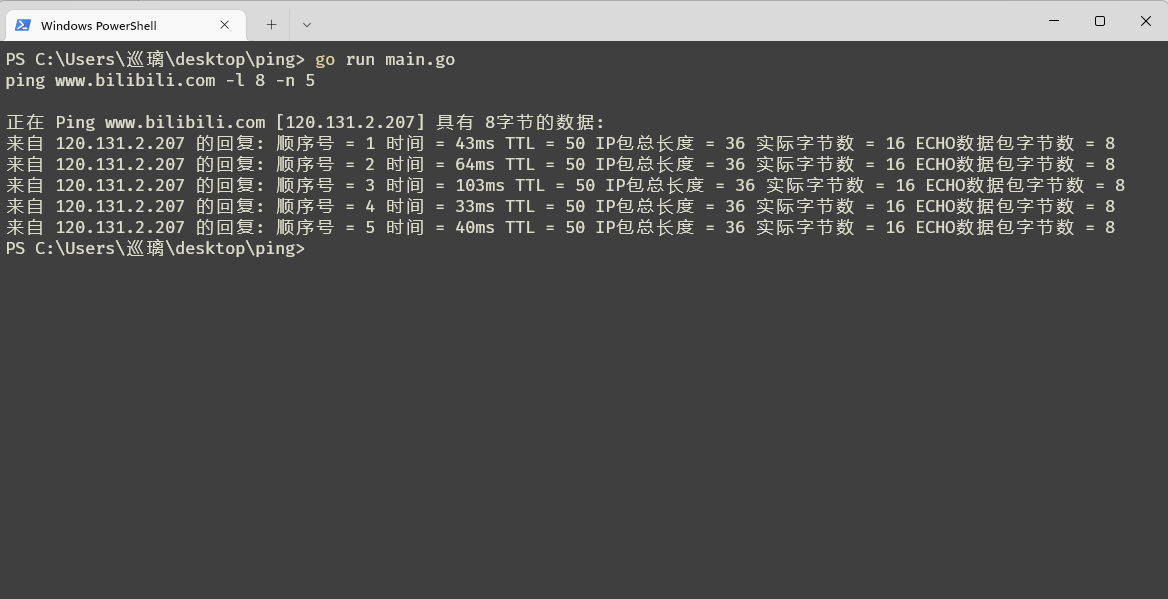
**图2 无参数的ping命令运行超时结果**

****

**图3 有length参数的ping命令运行正确结果**

****

**图4 有count参数的ping命令运行正确结果**

****

**图5 有count参数和length参数的ping命令运行正确结果**