基于 ICMP 的 ping 程序

第五章实验 1 报告

赖泽强组

1 实验目的

- 分析 icmp 协议报文，理解和掌握 ICMP 协议报文头各字段的含义和作用。
- 熟悉原始套接字编程，了解网络结构和网络传输底层协议。
- 理论结合实现，提高实践能力。

2 实验内容

用三种语言 C++,Java,Python 实现基于 ICMP 的 ping 程序，参考 windows 自带的 ping，具体要求如下所示。

- 在命令提示符下输入 ping ***.***.***.***。
  - ***为目的主机的 IP 地址
- 返回信息的格式：
  - 来自***.***.***.***的回复：字节大小，时间

3 实验原理

ping 程序是用来探测主机到主机之间是否可通信，如果不能 ping 到某台主机，表明不能和这台主机建立连接。ping 使用的是 ICMP 协议，它发送 ICMP 回送请求消息给目的主机。ICMP 协议规定：目的主机必须返回 ICMP 回送应答消息给源主机。如果源主机在一定时间内收到应答，则认为主机可达。

ICMP 协议通过 IP 协议发送的，IP 协议是一种无连接的，不可靠的数据包协议。因此，保证数据送达的工作应该由其他的模块来完成。其中一个重要的模块就是 ICMP(网络控制报文)协议。

当传送 IP 数据包发生错误，比如主机不可达，路由不可达等等，ICMP 协议将会把错误信息封包，然后传送回给主机。给主机一个处理错误的机会，这也就是为什么说建立在 IP 层以上的协议是可能做到安全的原因。ICMP 数据包由 8bit 的错误类型和 8bit 的代码和 16bit 的校验和组成。而前 16bit 就组成了 ICMP 所要传递的信息。

PING 利用 ICMP 协议包来侦测另一个主机是否可达。其原理是用类型码为 0 的 ICMP 发请求，受到请求的主机则用类型码为 8 的 ICMP 回应。ping 程序来计算间隔时间，并计算有多少个包被送达。用户就可以判断网络大致的情况。

4 实验环境

4.1 开发环境

语言	环境
C++	macOS 10.14.4 Apple LLVM version 10.0.1 (clang-1001.0.46.3)
Java	Windows 10 1903 IntelliJ IDEA 2018.3.3
Python	Windows 10 1903 PyCharm 2018.3.3

4.2 部署环境

可在 Windows，Linux，macOS 三平台正确编译运行。

5 实验步骤

5.1 命令行参数的设置

进行信息的格式化输入。若输入域名，通过 SOCKET 自带 API 的 gethostbyname 函数将主机名转成 ipv4 地址格式

```
1  data_type = 8 # ICMP Echo Request
2      data_code = 0 # must be zero
3      data_checksum = 0 # "...with value 0 substituted for this field..."
4      data_ID = 0 #Identifier
5      data_Sequence = 1 #Sequence number
6      payload_body = b'abcdefghijklmnopqrstuvwabcdefghi' #data
7      dst_addr = socket.gethostbyname(host)
8      print(" 正 在  Ping  {0}  [{1}]  具 有  32  字 节 的 数
9  据:".format(host,dst_addr))
```

5.2 数据校验

传入 data（十六进制）以每两个字节通过 ord 转十进制，第一个字节在低位，第二个字节在高位，将高于 16 位的部分与低十六位相加，将主机字节序转网络字节序，返回校验和

```
1  n = len(data)
2      m = n % 2
3      sum = 0
4      for i in range(0, n - m ,2):
5          sum += (data[i]) + ((data[i+1]) << 8)
6      if m:
7          sum += (data[-1])
8      sum = (sum >> 16) + (sum & 0xffff)
```

```
    sum += (sum >> 16)
    answer = ~sum & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer
```

5.3 使用 ICMP 协议创建 RAW SOCKET

dst_addr 为目的 IP 地址，imcp_packet 为要发送的包

```
def raw_socket(dst_addr,imcp_packet):

    rawsocket=socket.socket(socket.AF_INET,socket.SOCK_RAW,socket.getprot
    obyname("icmp"))
    send_request_ping_time = time.time()
    rawsocket.sendto(imcp_packet,(dst_addr,80))
    return send_request_ping_time,rawsocket,dst_addr
```

5.4 格式化发送包

把字节打包成二进制数据，通过之前的 checksum 函数获取校验和，打包成新的二进数据并返回作为发送包的标准格式

```
def
request_ping(data_type,data_code,data_checksum,data_ID,data_Sequence,payl
oad_body):
imcp_packet=struct.pack('>BBHHH32s',data_type,data_code,data_checksum,dat
a_ID,data_Sequence,payload_body)
    icmp_chesksum = chesksum(imcp_packet)
imcp_packet=struct.pack('>BBHHH32s',data_type,data_code,icmp_chesksum,dat
a_ID,data_Sequence,payload_body)
    return imcp_packet
```

5.5 创建发送包并开始发送

这部分的详细过程是通过之前的 request_ping 函数循环创建四个发送包，reply_ping 的返回值为往返时间，对时间长短进行判断是否超时，并统计平均时间等信息。

```
for i in range(0,4):
        icmp_packet                                                    =
request_ping(data_type,data_code,data_checksum,data_ID,data_Sequence    +
i,payload_body)
        send_request_ping_time,rawsocket,addr                          =
raw_socket(dst_addr,icmp_packet)
        times = reply_ping(send_request_ping_time,rawsocket,data_Sequence
+ i)
        if times > 0:
            if times<min:
                min=times
            if times>max:
```

```
                max=times
            alltimes=alltimes+times
            print("来自 {0} 的回复: 字节=32 时间
={1}ms".format(addr,int(times*1000)))
            num=num+1
            time.sleep(0.7)
        else:
            print("请求超时。")
    print("\n")
```

5.6 进行时间和回传包统计

在每次发送包时记录下往返时间，计算其最大，最小和平均值，统计信息

```
print("          数据包: 已发送=4, 已接收={0}, 丢失={1}({2}%丢
失)".format(num,int((4-num)/4),25*(4-num)))
    print("往返行程的估计时间(以毫秒为单位): ")
    print("          最短={0}ms, 最长={1}ms, 平均
={2}ms".format(int(min*1000),int(max*1000), int((alltimes/4)*1000)))
```

6 实验总结

这个实验主要是通过能否 ping 通某个主机从而判断主机与主机之间是否可通信。如果不能 ping 到某台主机，表明不能与该主机建立连接。由 ping 的结果检验网络情况。通过这个实验使我们更加了解 ping 程序的实现原理与效果。

附录

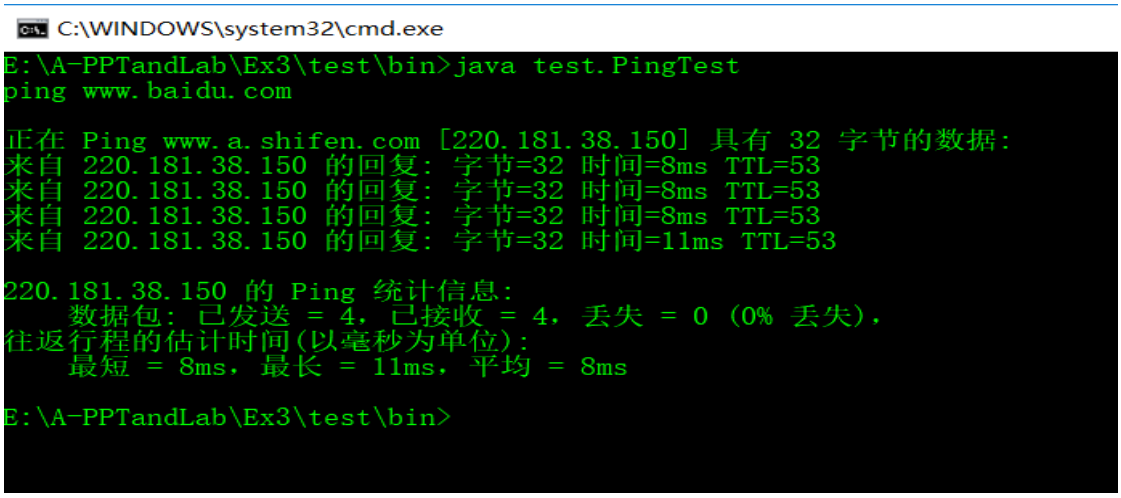

图 1 Java 运行截图

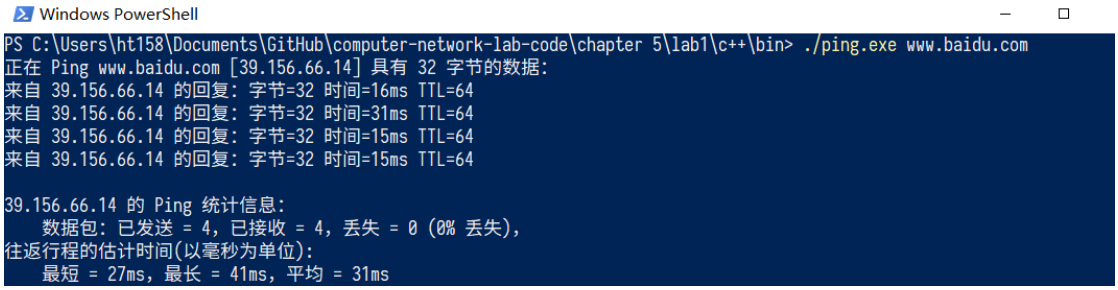

图 2 C++运行截图

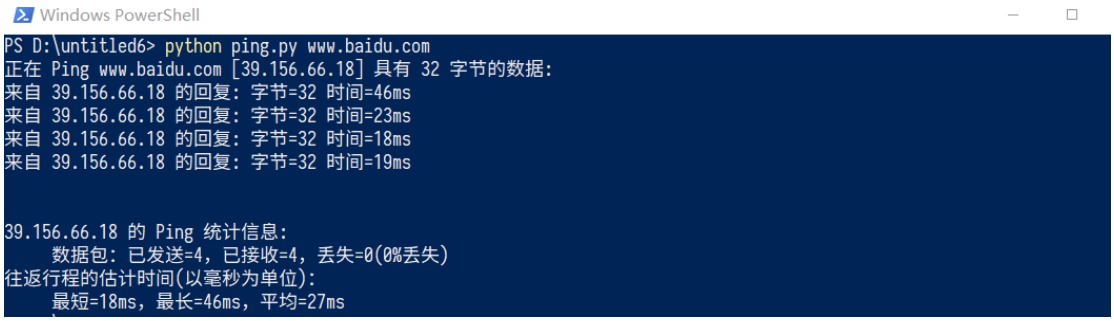

图 3 Python 运行截图