**实验报告**

**实验名称**

实现TRACEROUTE命令

**实验目的及内容**

目的: 要求学生掌握Socket编程技术,以及ICMP协议   
内容: i. 要求学生掌握利用Socket进行编程的技术ii. 不能采用现有的工具,必须自己一步一步,根   
据协议进行操作iii. 要求每一次操作,必须点击下一步才能继续iv. 了解Traceroute报文的格式和步骤,要求符合ICMP协议并组建报文v. 在一秒钟内,如果收到,则为成功,如果收不 到,则失败vi. 必须采用图形界面,查看每次收到回应的结果 vii.可以通过程序,查看经过了哪些节点

**实验方案与步骤**

Traceroute程序使用ICMP报文和和IP首部中的TTL字段（生存周期）。  
每个处理数据报的路由器都要把TTL的值减去1或者减去数据报在路由器中停留的秒数。由于大多数的路由器转发数据报的延时都小于1秒钟，因此TTL最终成为一个跳站的计数器，所经过的每个路由器都将其值减1.  
TTL字段的目的是防止数据报在选路时无休止的在网络中流动。例如，当路由器瘫痪或者两个路由器之间的连接丢失时，选路协议有时回去检查丢失的路由器并一直进行下去。TTL字段就是在这些寻暖传递的数据报上加上一个生存上限。  
当路由器收到一份IP数据报，如果TTL字段是0或者1，则路由器不转发该数据报（接收到这种数据报的目的主机可以将它交给应用程序，这是因为不需要转发该数据报。但是，在通常情况下系统不应该接收TTL字段为0的数据报）。通常情况下是，路由器将该数据报丢弃，并给信源主机发送一份ICMP超时信息。Traceroute程序的关键在于，这份ICMP超时信息包含了该路由器的地址。  
那么，Traceroute就通过发送一份TTL字段为1的IP数据报给目的主机。处理这份数据报的第一个路由器将TTL值减去1，丢弃该数据报，并回发一份超时ICMP报文。这样就得到了该路径中的第一个路由器的IP地址。然后Traceroute发送一份TTL为2的数据报，这样就得到了第二个路由器的IP地址。那么，继续这个过程，直到达到目的主机。即使目的主机接收到一份TTL值为1的数据报也不会丢弃该数据报并产生一份ICMP报文，因为已经到达最终目的地。这个时候，Traceroute程序发送一份UDP数据报给目的主机，但选择一个不可能的值作为目的端口号（大于30000），使得目的主机的任何一个程序都不可能使用该端口。因为，当该数据报到达时，将使目的主机的UDP模块产生一份“端口不可达”错误的ICMP报文，这样Traceroute程序要做的就是区分搜到的ICMP报文是是超时还是端口不可达，以判断什么时候结束。

我使用 wireshark 观察了 OS X 系统自带的 traceroute 程序的工作过程，并仿照它写了这个程序。

1. 买一台装有 linux 或类似系统的个人电脑

2. 安装 Python 运行环境

3. 安装 git

4. 打开终端

5. 输入 $ git clone http://github.com/aheadlead/computer-network-exp

6. 输入 $ ./computer-network-exp/traceroute/traceroute\_gui.py

7. 在 ip address 栏位后面输入一个目的IP地址

8. 点击后面的按钮

9. 观察现象

**源程序**

文件 traceroute.py

#!/usr/bin/env python3

# coding=utf-8

import sys

import socket

import struct

import select

import time

def traceroute(IP, ttl):

PORT = 33434

MAX\_HOPS = 64

if ttl <= MAX\_HOPS:

print\_buffer = [''] \* 7

print\_buffer[0] = str(ttl)

for cycle in range(3):

send\_sock = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM,

proto=socket.getprotobyname('udp'))

recv\_sock = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_RAW,

proto=socket.getprotobyname('icmp'))

send\_sock.setsockopt(socket.SOL\_IP, socket.IP\_TTL, ttl)

start\_time = time.time()

send\_sock.sendto(('\x00'\*24).encode('utf-8'), (IP, PORT + ttl))

if select.select([recv\_sock], [], [], 1.0)[0]:

raw\_packet, addr = recv\_sock.recvfrom(1024)

stop\_time = time.time()

print\_buffer[4+cycle] = '%.2f' % (1000\*(stop\_time - start\_time))

print\_buffer[1] = addr[0] + '\t'

ip\_header\_length = (raw\_packet[0] & 0x0F)\*4

icmp\_packet\_header = raw\_packet[ip\_header\_length:ip\_header\_length+4]

Type, Code = struct.Struct('!bb2x').unpack(icmp\_packet\_header)

print\_buffer[2] = str(Type)

print\_buffer[3] = str(Code)

else:

print\_buffer[4+cycle] = '\*'

print\_buffer[1] = '?'+' '\*14

send\_sock.close()

recv\_sock.close()

return '\t'.join(print\_buffer) + '\n'

文件 traceroute\_gui.py

#!/usr/bin/env python3

# coding=utf-8

import os

import platform

import re

import subprocess

from tkinter import \*

from tkinter.messagebox import \*

from traceroute import traceroute

if platform.system() != 'Windows':

if os.geteuid() != 0:

print('root permission needed')

exit(1)

else:

print('STOP: This program is incompatible with Microsoft Windows.')

exit(1)

ttl = 1

root = Tk()

root.geometry('{}x{}'.format(470, 400))

dest\_ip\_label = Label(root, text='ip address')

dest\_ip\_label.grid(row=0, column=0, sticky='W')

dest\_ip\_entry = Entry(root)

dest\_ip\_entry.grid(row=0, column=1)

result\_text = Text(root, width=65)

result\_text.grid(row=1, columnspan=4)

next\_svar = StringVar(value='traceroute')

def go():

global ttl

ip\_pattern = r'(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$'

if re.match(ip\_pattern, dest\_ip\_entry.get()) is None:

showerror(title='error', message='invaild ip address')

else:

r = traceroute(dest\_ip\_entry.get(), ttl)

if ttl == 1:

result\_text.delete(1.0, END)

result\_text.insert(END, 'ttl\taddress\t\ttype\tcode\t1\t2\t3\n')

result\_text.insert(END, r)

if '\t3\t3' in r:

ttl = 1

next\_svar.set('traceroute')

showinfo(title='done', message='traceroute finished')

else:

ttl += 1

next\_svar.set('next')

next\_button = Button(root, textvariable=next\_svar, command=go)

next\_button.grid(row=0, column=2)

root.mainloop()

**总结**

通过实验实现TRACEROUTE命令，我学习到了ICMP协议的工作过程，了解了Socket编程技术的魅力，感受到了计算机网络体系的美妙，为毕业后找到理想的工作，建设我社会主义新中国，奠定了坚实的基础。