

**《计算机网络》课程设计**

**题 目 中小型网络工程设计与实现**

**基于SOCKET的局域网通信软件的设计与实现**

**姓名\_\_\_\_\_\_申博文\_\_\_\_\_\_\_\_\_**

**学号\_\_\_\_201883290091\_\_\_\_\_**

**专业\_\_\_计算机科学与技术\_\_\_**

**二Ｏ 二一 年 一 月 六 日**

中小型网络工程设计与实现

申博文

计算机与软件学院，计算机科学与技术，2018级5班

**摘 要：**企业局域网的规划与组建，为各种网络应用程序提供了基本的操作平台，使信息可以在正确的时间正确地传输到不同的部门，例如集成的企业信息管理和办公室自动化。企业局域网工程建设中主要应用了网络技术中的重要分支局域网技术来建设与管理。本设计主要以中小型局域网网络建设过程用到的实施方案为方向，对企业需求进行分析，设计网络拓扑结构图并对网络进行划分，最后给出各设施的配置情况。最后，通过Cisco Packet Tracer进行的仿真实验，验证了本设计的有效性。

**关键词**：计算机网络；局域网；网络管理

1、需求分析

为了使企业内部做到高效的信息交换、资源共享，为员工提供准确、可靠、快捷的各种数据和信息，提高工作效率，实现资源共享，降低运作及管理成本，企业网络的建设是企业向信息化发展的必然选择。因此，企业对网络的需求快速增加，包括子网划分、VLAN、所提供的服务（WWW、FTP、MAIL、DNS等等）。

1.1 网络选择

Internet组织机构定义了A、B、C三类地址。A类网络以0开头，有126个，每个网络最多有16777214台主机；B类网络以10开头，有16384个，每个网络最多有65534台主机；C类网络以110开头，有2097152个，每个网络最多有254台主机；根据需求，该企业需要的主机数量大于254，C类网络无法满足其需要，因此选择B类网络，本文中假设企业的B类网络号为129.0.0.0/16。

1.2 子网划分

一个B类地址可以包含大量网络节点，若将其归为同一个广播域，则会因大面积广播通信导致网络的阻塞。为解决此问题，可以把B类地址分成更小的子网，每个子网具有一个新的网络地址，其中的主机数更少，广播域也相应地更小。划分子网后，通过使用子网掩码，把子网隐藏起来，使得从外部看网络没有变化。

1.3 虚拟局域网

虚拟局域网（VLAN）建立在交换技术的基础上，将网络结点按工作性质与需要划分成若干个“逻辑工作组”，即虚拟网络。VLAN的实现技术有四种：用交换机端口（Port）号定义虚拟网络、用MAC地址定义虚拟网络、用IP广播组定义虚拟网络、用网络层地址定义虚拟网络。

1.4 设计需求

本网络工程设计需要在部门之间实现子网和VLAN的划分，以减小广播域。划分后，不同用户的数据流会被互相隔离，用户扩展有了极大的灵活性，网络的安全也得到了保障。另一方面，需要使用三层交换机实现VLAN间的通信，确保部门间一定的连通性。同时，需要提供WWW服务、文件传输（FTP）服务、电子邮件（Email）服务等。为了提高安全性，对外提供服务的服务器要用防火墙保护，内部计算机用双层防火墙保护。

2、概要设计

网络拓扑结构中企业被分为三个建筑，即行政楼、销售部和生产厂区，每栋建筑各层分别有一台二层交换机连接本层的所有主机，同时再连至该建筑的三级交换机。划分子网和VLAN后，各部门的主机不能直接与其他部门通信，但是可以通过三层交换机转发进行不同VLAN间通信。此外，销售部和生产厂区的三层交换机与行政楼中心交换机相连。同时，行政楼三层交换机上连接内网交换器和外网交换器。内网交换机上连有DNS、EMAIL、FTP、WWW等服务器，外网交换机上连有访问外网的路由器。网络拓扑结构图如图 1所示，反映了企业局域网的连接情况，物理结构图如图 2所示，直观反映设计与需求的契合程度。

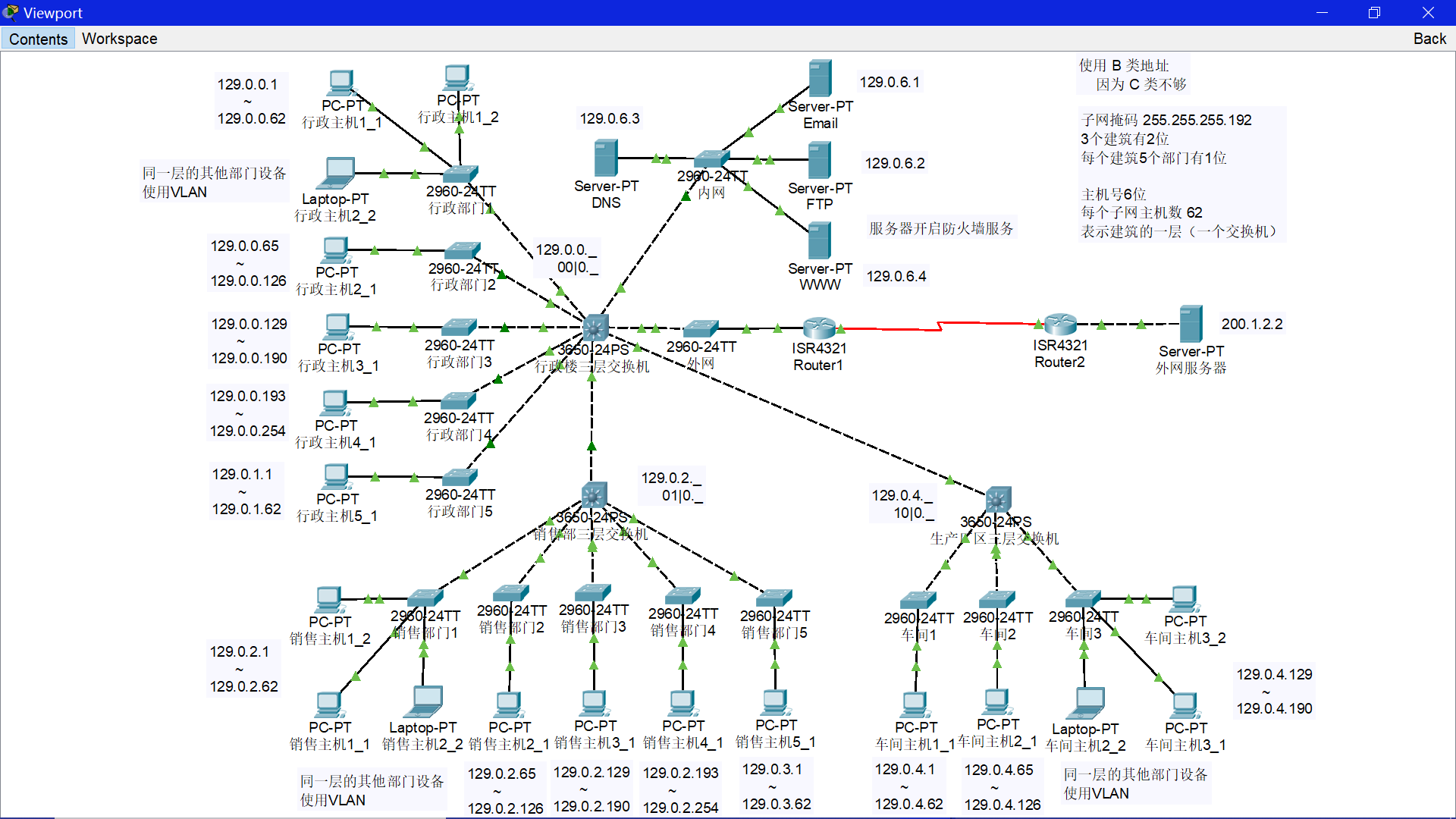


图 1 企业网络拓扑结构图

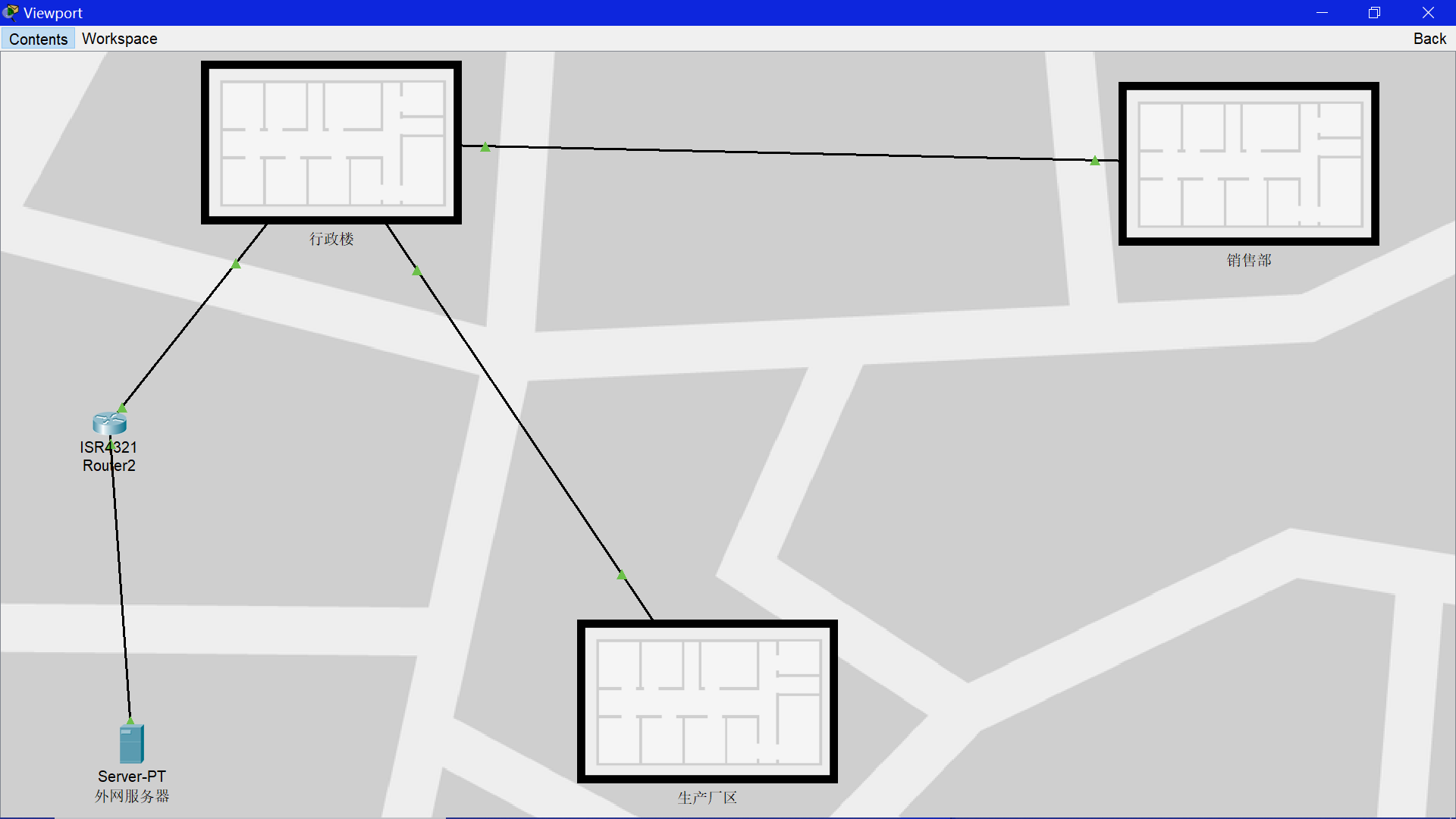
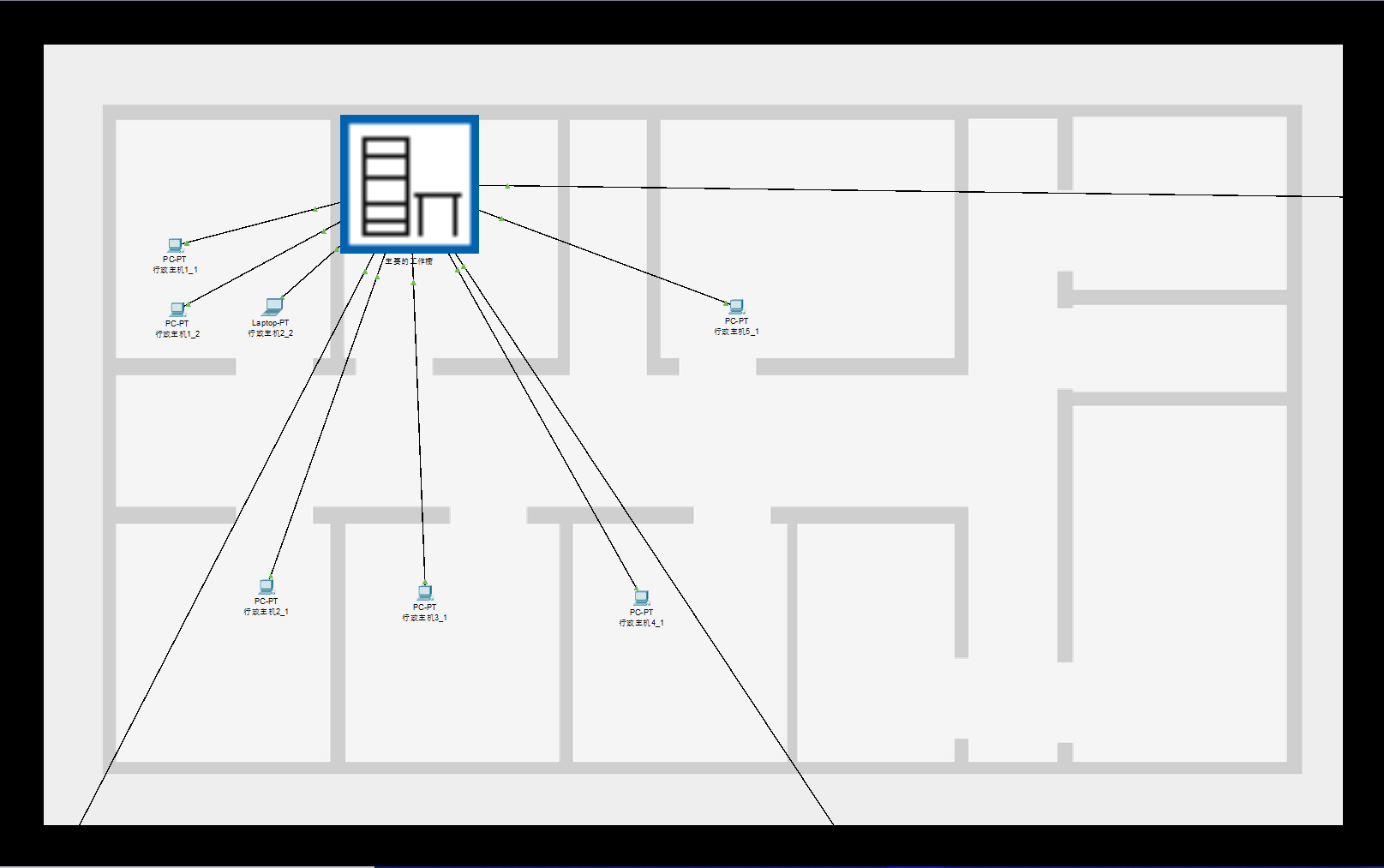
 

图 2 企业网络物理结构图

2.2 VLAN划分与子网规划

根据设计需求，行政楼的5个部门，每个部门约24人，假设为每人分配2个IP地址，即每个部门需要至少位主机号；同理，销售部门需要位主机号；考虑到生产厂区的人均设备数量较少，假设为每人分配1个IP地址，则每个车间需要位主机号。同时，行政楼、销售部和生产厂区3个建筑需要位掩码进行区分，建筑内的5个部门需要位掩码进行区分，所以总共需要5位进行区分的掩码。综上，VLAN分配与子网划分情况如表 1所示。

表 1 VLAN分配与子网划分

|  |  |  |  |
| --- | --- | --- | --- |
| VLAN序号 | 位置 | 网段 | 子网掩码 |
| VLAN2 | 内网 | 129.0.6.0 - 129.0.6.62 | 255.255.255.192 |
| VLAN3 | 外网 | 129.0.8.0 - 129.0.8.62 | 255.255.255.192 |
| VLAN11 | 行政部门1 | 129.0.0.1 - 129.0.0.62 | 255.255.255.192 |
| VLAN12 | 行政部门2 | 129.0.0.65 - 129.0.0.126 | 255.255.255.192 |
| VLAN13 | 行政部门3 | 129.0.0.129 - 129.0.0.190 | 255.255.255.192 |
| VLAN14 | 行政部门4 | 129.0.0.193 - 129.0.0.254 | 255.255.255.192 |
| VLAN15 | 行政部门5 | 129.0.1.1 - 129.0.1.62 | 255.255.255.192 |
| VLAN21 | 销售部门1 | 129.0.1.1 - 129.0.1.62 | 255.255.255.192 |
| VLAN22 | 销售部门2 | 129.0.1.65 - 129.0.1.126 | 255.255.255.192 |
| VLAN23 | 销售部门3 | 129.0.1.129 - 129.0.1.190 | 255.255.255.192 |
| VLAN24 | 销售部门4 | 129.0.1.193 - 129.0.1.254 | 255.255.255.192 |
| VLAN25 | 销售部门5 | 129.0.2.1 - 129.0.2.62 | 255.255.255.192 |
| VLAN31 | 车间1 | 129.0.4.1 - 129.0.4.62 | 255.255.255.192 |
| VLAN32 | 车间2 | 129.0.4.65 - 129.0.4.126 | 255.255.255.192 |
| VLAN33 | 车间3 | 129.0.4.129 - 129.0.4.190 | 255.255.255.192 |

2.3 服务器设计

为了满足企业需求的FTP、EMAIL和WWW服务，需要在内网中设置数个服务器。其中DNS服务器提供域名和对应的IP地址间的解析，是EMAIL和WWW等服务的基础。内容如图 3所示。

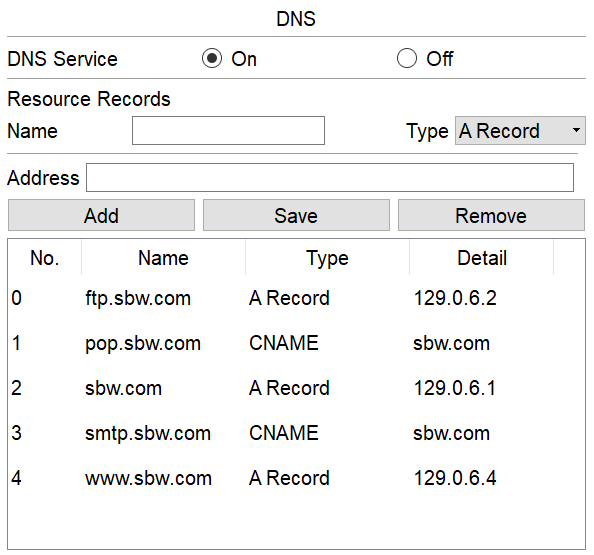


图 3 DNS服务器的域名记录列表

EMAIL服务器：邮件服务器是一种用来负责电子邮件收发管理的设备。支持使用简单邮件传送协议（SMTP）发送电子邮件，使用邮局协议3（POP3）接收电子邮件。服务器域名使用DNS服务器解析的sbw.com，并且存储邮件用户列表，如图 4所示。

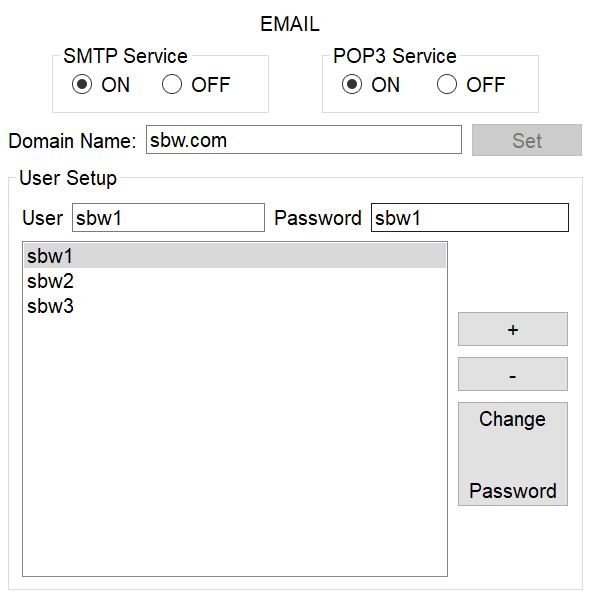


图 4 EMAIL服务器域名及用户列表

WWW服务器：服务器开启HTTP服务，客户端通过统一资源定位符URL向WWW服务器发送请求，WWW服务器通过DNS解析URL中定位的资源，给客户程序回送相应的文档。WWW服务器内容如图 5所示。

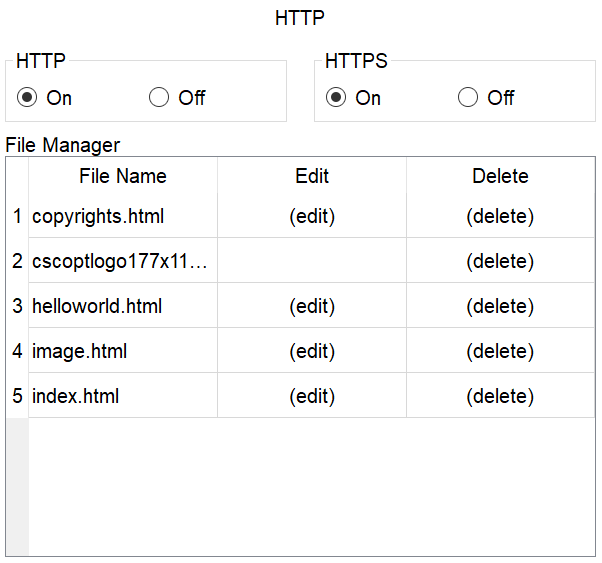


图 5 WWW服务器文件列表

FTP服务器：FTP即文件传输协议，是一种基于TCP的协议。通过FTP协议，用户可以在FTP服务器中进行文件的上传或下载等操作。FTP服务器是在互联网上提供文件存储和访问服务的计算机，它们依照FTP协议提供服务。服务器内容如图 6所示。

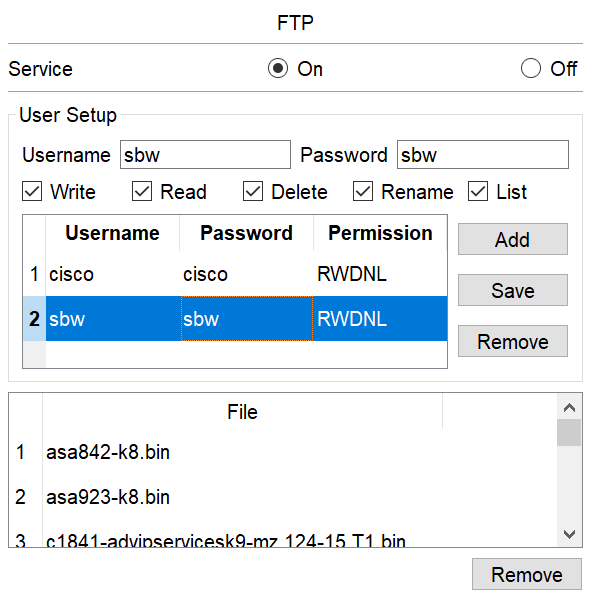


图 6 FTP服务器用户列表和文件列表

3、详细设计

行政楼、销售部、生产厂区的三层交换机配置相似，为节省空间，此处以表 2 行政楼三层交换机配置清单为例。

表 2 行政楼三层交换机配置清单

|  |  |
| --- | --- |
| interface Vlan2  mac-address 0007.ec16.0901  ip address 129.0.6.62 255.255.255.192  interface Vlan3  mac-address 0007.ec16.0902  ip address 129.0.8.62 255.255.255.192  interface Vlan11  mac-address 0007.ec16.0904  ip address 129.0.0.62 255.255.255.192  interface Vlan12  mac-address 0007.ec16.0905  ip address 129.0.0.126 255.255.255.192  interface Vlan13  mac-address 0007.ec16.0906  ip address 129.0.0.190 255.255.255.192  interface Vlan14  mac-address 0007.ec16.0907  ip address 129.0.0.254 255.255.255.192  interface Vlan15  mac-address 0007.ec16.0908  ip address 129.0.1.62 255.255.255.192  interface Vlan21  mac-address 0007.ec16.090a  ip address 129.0.2.62 255.255.255.192  interface Vlan22  mac-address 0007.ec16.090b  ip address 129.0.2.126 255.255.255.192  interface Vlan23  mac-address 0007.ec16.090c  ip address 129.0.2.190 255.255.255.192 | interface Vlan24  mac-address 0007.ec16.090d  ip address 129.0.2.254 255.255.255.192  interface Vlan25  mac-address 0007.ec16.090e  ip address 129.0.3.62 255.255.255.192  interface Vlan31  mac-address 0007.ec16.0911  ip address 129.0.4.62 255.255.255.192  interface Vlan32  mac-address 0007.ec16.0912  ip address 129.0.4.126 255.255.255.192  interface Vlan33  mac-address 0007.ec16.0913  ip address 129.0.4.190 255.255.255.192  interface GigabitEthernet1/0/1  switchport trunk encapsulation dot1q  switchport mode trunk  ……  interface GigabitEthernet1/0/9  switchport trunk encapsulation dot1q  switchport mode trunk  router rip  ip classless  ip route 200.1.2.0 255.255.255.0 129.0.8.1  ip route 200.1.1.0 255.255.255.0 129.0.8.1 |

各个建筑楼层的双层交换机配置相似，此处以表 3 行政楼1交换机配置清单为例。

表 3 行政楼1交换机配置清单

|  |  |
| --- | --- |
| interface FastEthernet0/1  switchport mode trunk  interface FastEthernet0/2  switchport access vlan 11 | interface FastEthernet0/3  switchport access vlan 11  interface FastEthernet0/4  switchport access vlan 12 |

内外网连接通过路由器实现，其中企业内的路由器如表 4所示。

表 4 内外网连接路由器配置清单

|  |  |
| --- | --- |
| interface GigabitEthernet0/0/0.3  encapsulation dot1Q 3  ip address 129.0.8.1 255.255.255.192  interface Serial0/1/0  ip address 200.1.1.1 255.255.255.0  clock rate 64000 | router rip  ip classless  ip route 200.1.2.0 255.255.255.0 200.1.1.2  ip route 129.0.0.0 255.255.0.0 129.0.8.62 |

4、调试分析

测试各个主机间的连通性如图 7所示。行政主机1\_1在同一VLAN中的主机时，没有经过三层交换机，从可以看出；而在ping其他VLAN中的主机时，被三层交换机转发，即。

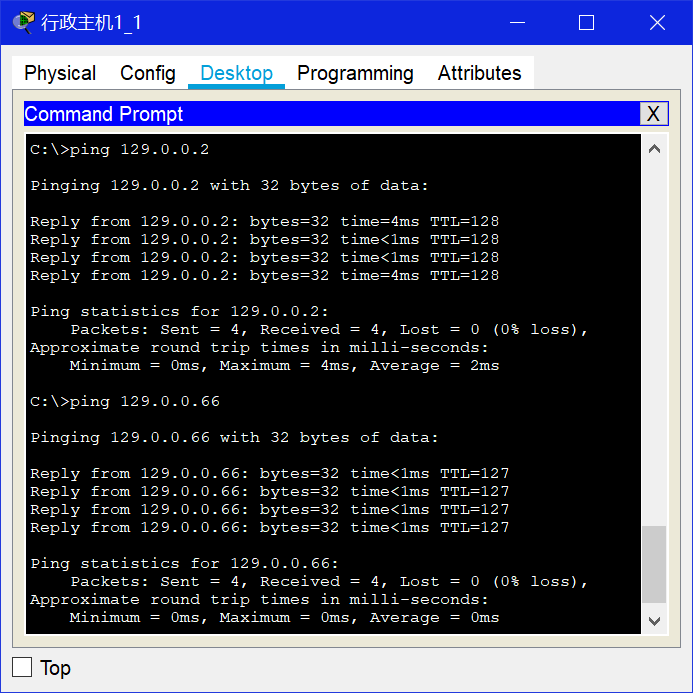
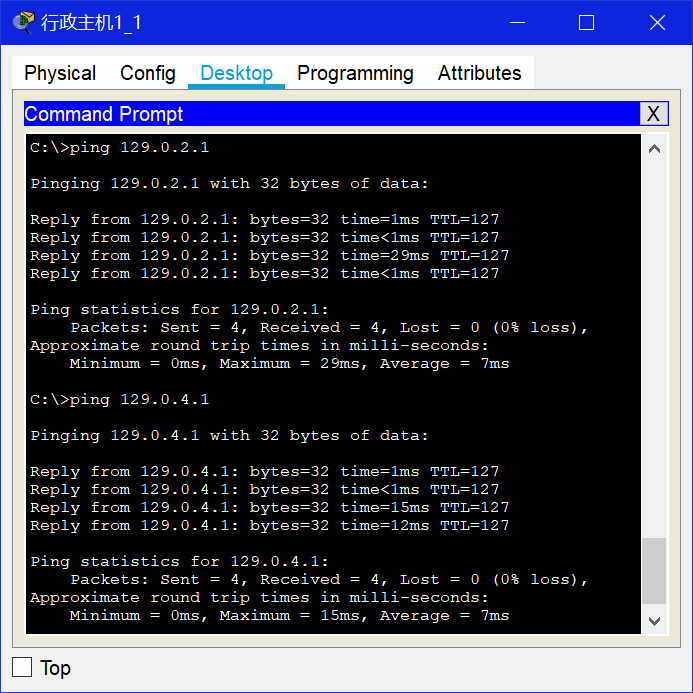
 

图 7 行政主机1\_1（129.0.0.1）与同部门（左上），同建筑其他部门（左下），其他建筑内主机（右）的通信（ping）测试

测试主机与内外网的连通性，通过行政主机1\_1访问内网提供的www.sbw.com和模拟外网200.1.2.2（不属于企业的C类地址）。结果如图 8所示，内外网站均能正确访问。

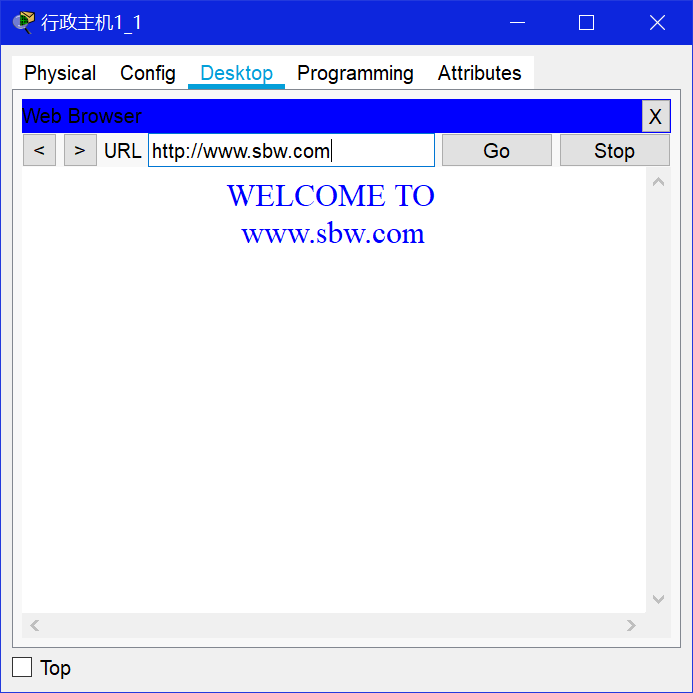
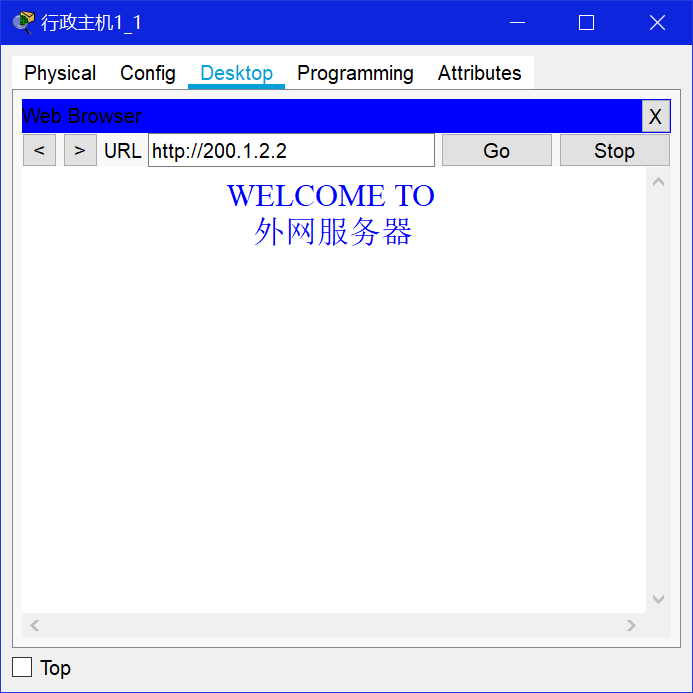
 

图 8 行政主机1\_1（129.0.0.1）访问内网WWW服务（左）和外网（右）的测试

测试EMAIL服务前，需要现在测试主机上设置EMAIL，包含用户名、密码、EMAIL地址、收发邮件服务器等，如图 9所示。

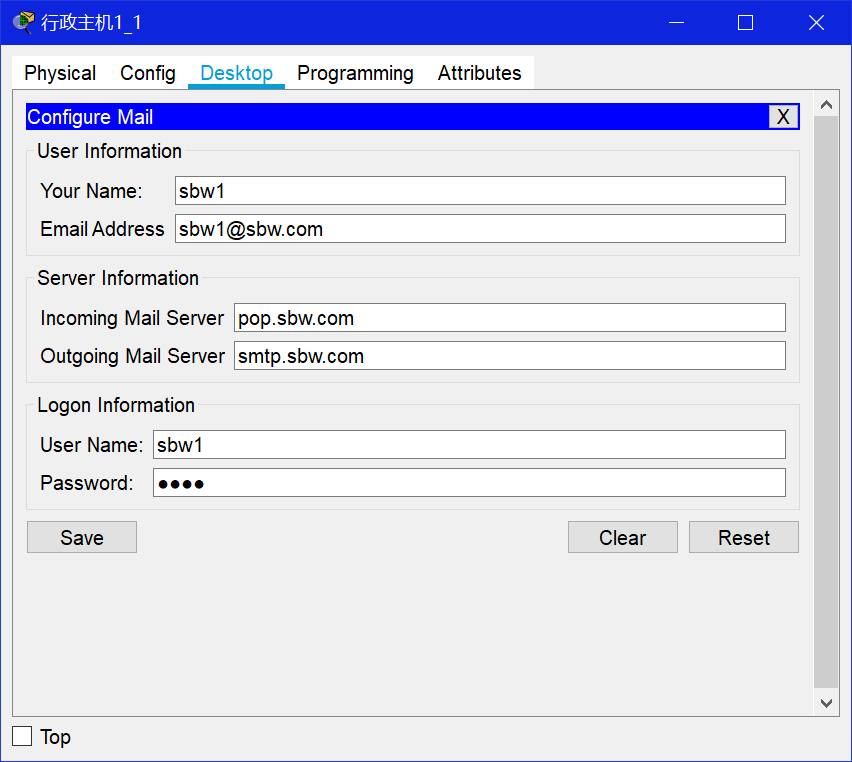
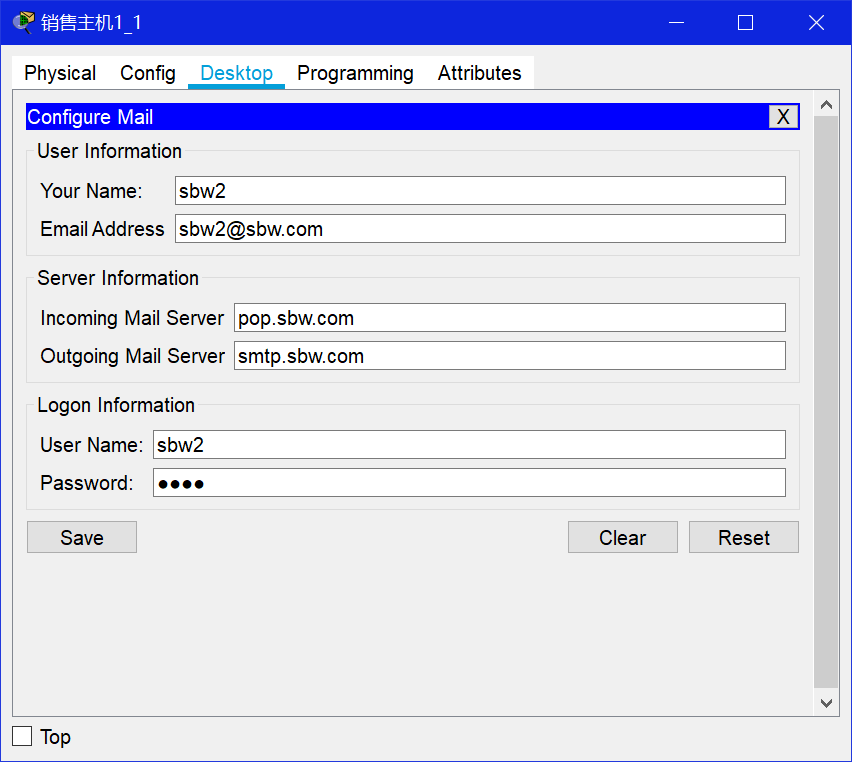
 

图 9 行政主机1\_1（129.0.0.1, sbw1@sbw.com）与销售主机1\_1（129.0.2.1, sbw2@sbw.com）的EMAIL设置

测试过程中，先令行政主机1\_1向销售主机1\_1发送邮件，此时邮件将被传输至EMAIL服务器。在获得“Send Success”提示后，令销售主机1\_1从服务器端接收邮件，并回复行政主机1\_1的来信。测试结果如图 9所示，主机间可以正常进行EMAIL收发。

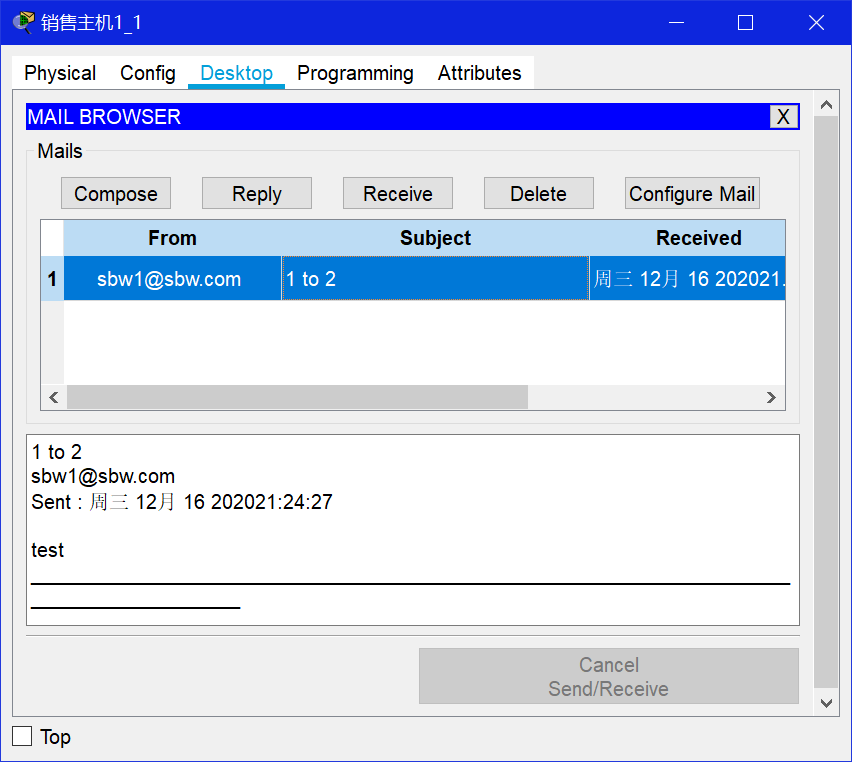
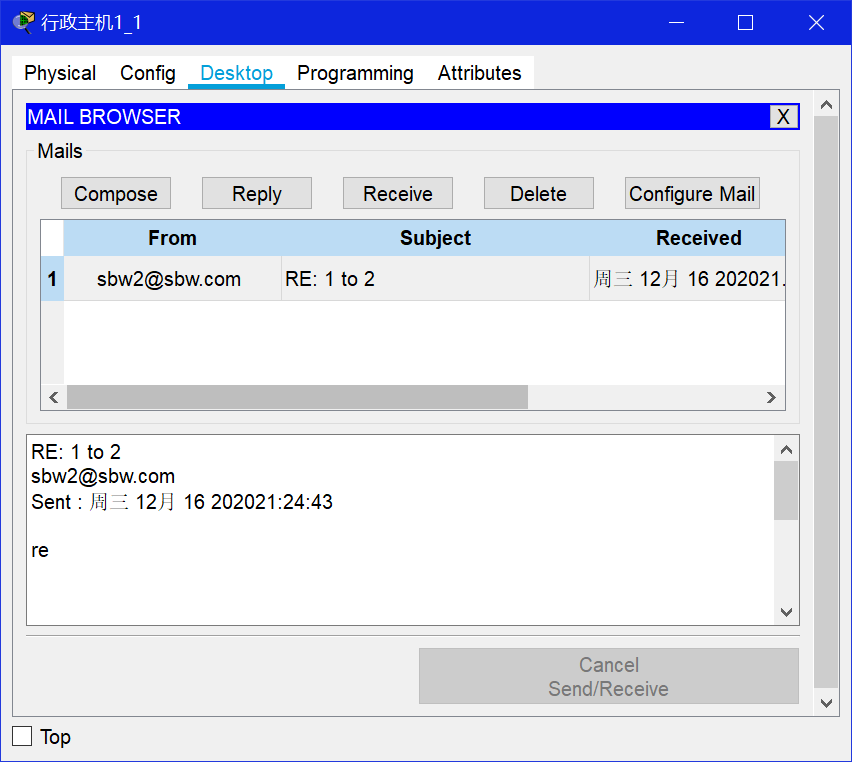
 

图 10 行政主机1\_1（129.0.0.1, sbw1@sbw.com）与销售主机1\_1（129.0.2.1, sbw2@sbw.com）的Email发信（左）回信（右）测试

FTP服务需要通过命令“ftp [地址]”的方式启动。成功连接后需输入用户名、密码（需要与FTP服务器中对应），输入正确后可以登录，并进行操作。图 11中测试了FTP服务器的访问以及使用了dir命令查看服务器中的文件列表。

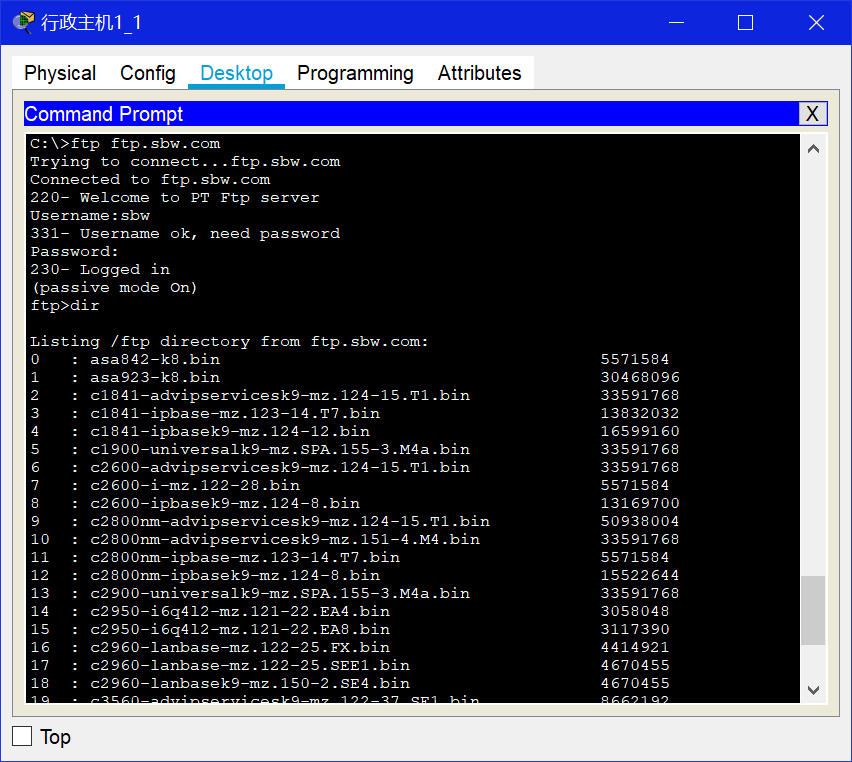


图 11 行政主机1\_1（129.0.0.1）访问FTP服务器，并查看文件列表

经过上述测试，可以看出此网络设计基本符合企业需求，可以为其提供较好的局域网通信服务。

**参考文献**

[1] 谢希仁. 计算机网络[M]. 电子工业出版社, 2013.

[2] JamesF.Kurose, KeithW.Ross. 计算机网络:自顶向下方法[M]. 机械工业出版社, 2009.

[3] Fu W H , Lu A X . VLAN Technology Application Research Based on Campus Network Security[J]. Applied Mechanics & Materials.

**基于SOCKET的局域网通信软件的设计与实现**

申博文

计算机与软件学院，计算机科学与技术，18级5班

**摘 要：**现代企业需要可靠的通信手段与工具，实现其内部的高效信息传递与交流。围绕企业局域网通信，结合企业的需求与市场的实际情况,本课题设计并实现了基于python和Socket技术的C/S模式企业局域网通信系统。本系统支持企业局域网内的私聊与群聊功能，提供简洁的客户端图形用户界面，为企业提供高效安全的内部沟通，提高企业工作效率，促进企业发展。模拟测试结果表明，本课题实现的企业局域网通信软件可以基本满足企业内部局域网通信需求，整体运行状态良好，测试结果符合预期。

**关键词**：计算机网络；套接字；局域网通信

1、需求分析

企业局域网通信软件是面向企业内部沟通的通信系统，使用客户机/服务器（C/S）模式为企业提供高效的管理以及稳定的内部通信。企业员工通过本软件能够使用私聊功能来进行工作上沟通交流、通过群聊功能的进行工作汇报或者通知信息等。

1.1 客户端需求

本企业局域网内部通信软件客户端系统主要提供消息显示、消息编辑与发送、在线用户显示、聊天对象选择等功能。由于客户端需要经常被用户使用，需要提供直观便捷的图形用户界面（Graphical User Interface, GUI）。

用户登录系统后，聊天对象默认群聊。用户可以在消息编辑区编辑文字消息，并将消息发送给服务器，通过服务器将信息转发给其他在线的用户。所有系统中的在线用户和服务器均可以接收到群聊消息。

用户通过点击在线用户列表，可以将聊天对象选择为某一个用户，进入私聊模式。系统根据私聊另一方的IP地址和端口号，通过服务器向对方发送文字信息。私聊信息只有私聊双方和服务器可以查看到。在私聊模式下，可以通过点击“群聊”切换回群聊模式。

1.2 服务器需求

本企业局域网内部通信软件客户端系统主要提供消息转发、在线用户存储等功能。由于服务器开启后较少被用户直接使用，仅为其设计基本的命令行界面（Command Line Interface, CLI）。

1.3 客户机与服务器的连接

本系统中使用TCP连接。TCP是一种面向连接的保证可靠传输的协议。发送方和接收方的成对的两个socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个socket等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输并且双方都可以进行发送或接收操作，任何一方断线都会造成无法进行通话。

2、概要设计

系统结构图如图 12所示，其中TCP协议用来控制网络设备间的通信，两端设备按作用分为客户端和服务端。服务端为客户端提供服务，通常等待客户端的请求消息，有客户端请求到达后，及时提供服务和返回响应消息；客户端向服务端主动发出请求，并接收服务的响应消息。

客户端主面板由三部分构成，主要为消息显示区，在线用户区和消息编辑区。客户端向服务器发送请求用户列表的信息后，将服务器回复的用户信息显示在用户列表中，显示在线人数和在线用户列表。用户列表中除了显示用户名以外还可以显示用户IP地址。系统启动时默认为群聊状态，通过点击列表中的某个在线用户可以与其进行私聊，点击“群聊”后切换为群聊模式。

服务器的主要功能有被动监听、建立连接、管理在线用户（增加和移除）和消息收发。服务器需要在客户机启动前启动，否则客户机无法建立连接。类似的，若服务器关闭，所有客户机的连接都会中断。

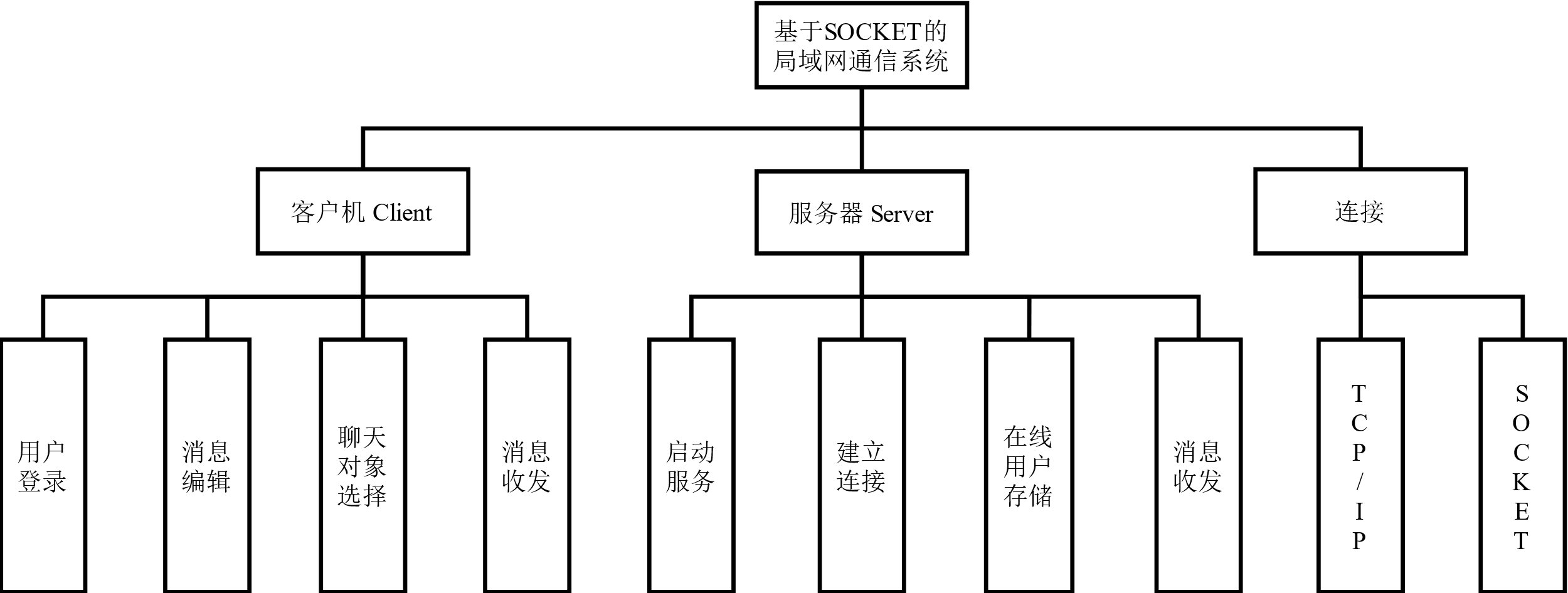


图 12 系统结构图

3、详细设计

系统序列如图 13所示，用户首先使用登录界面（Login UI），输入登录信息后进入通信界面（Chat UI），此时，客户机与服务器建立连接，开始进行基于SOCKET的通信。用户退出客户端后，与服务器的连接断开。

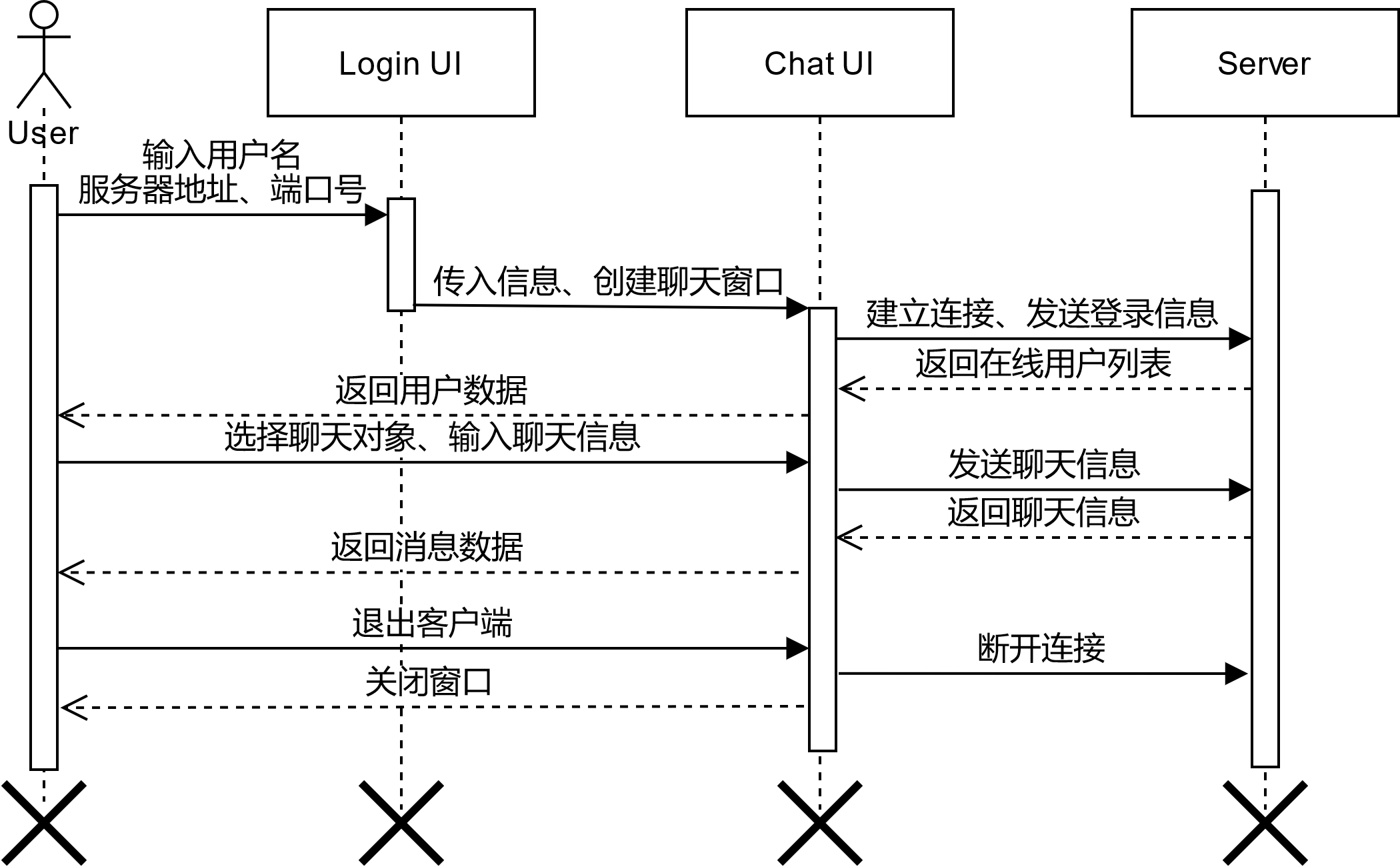


图 13 系统序列图

客户端界面、发送群聊与私聊消息的场景如图 14所示，可以看出由用户sbw1发送的群聊消息“1 to all”（蓝色与绿色）被所有在线用户接收，而用户sbw1向用户sbw2发送的私聊消息“1 to 2”（红色）仅被用户sbw1和sbw2接收，用户sbw3没有收到。

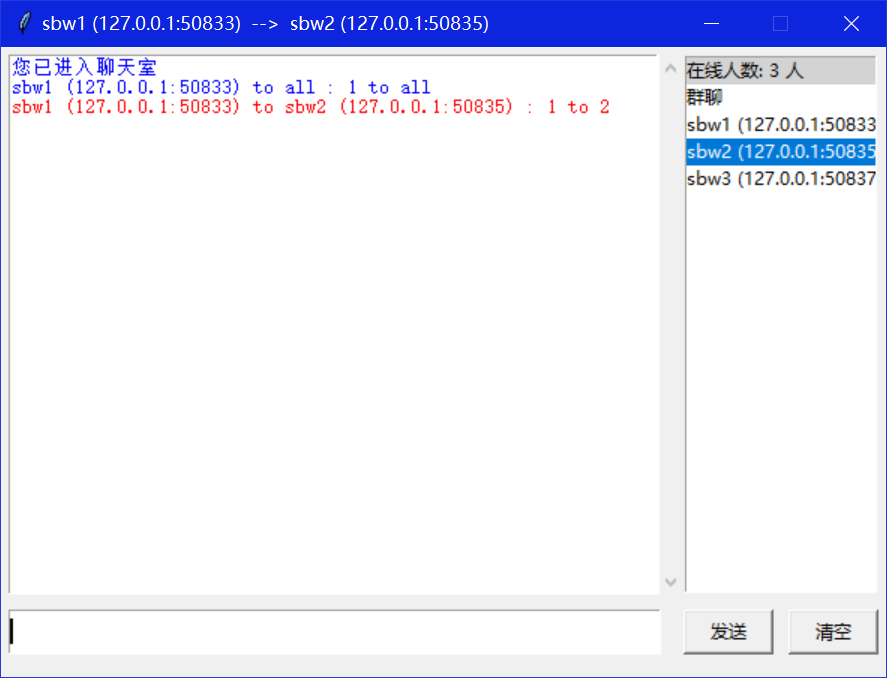
  

图 14 客户端群聊与私聊场景，从左至右分别为用户sbw1、sbw2和sbw3

服务器界面和在线用户管理的场景如图 15所示，可以看用户的连接建立与断开情况，以及剩余在线用户列表。



图 15 服务器命令行界面

4、实现

客户端关键代码如下（省略UI部分）：

class **Login\_UI**():

    def **\_\_init\_\_**(self, main\_window):

*# region UI …*

*# endregion*

*# 登录按钮事件*

    def **login**(self, \*args):

        global SERVER\_IP, SERVER\_PORT, USERNAME

        SERVER\_IP, SERVER\_PORT = self.ip\_entry.get().split(':')  *# 获取IP和端口号*

        SERVER\_PORT = int(SERVER\_PORT)  *# 端口号需要为int类型*

        USERNAME = self.username\_entry.get()

        self.main\_window.destroy()  *# 关闭窗口*

class **Chat\_UI**():

    def **\_\_init\_\_**(self, main\_window):

*# region UI …*

*# endregion*

        self.my\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        self.my\_socket.connect((SERVER\_IP, SERVER\_PORT))

        addr = self.my\_socket.getsockname()  *# 获取客户端ip和端口号*

        addr = addr[0] + ':' + str(addr[1])  *# SERVER\_IP:port*

        USERNAME = USERNAME + ' (' + addr + ')'

        self.main\_window.title(USERNAME + ' 群聊')

        self.my\_socket.send(USERNAME.encode())

        r = threading.Thread(target=self.recv)

        r.start()  *# 开始线程接收信息*

*# 清空按钮事件*

    def **clear**(self, \*args):

        self.message\_text.set('')

*# 发送按钮事件*

    def **send**(self, \*args):

        self.online\_users.append('群聊')

        if self.chatter not in self.online\_users:

            tkinter.messagebox.showerror('发送失败', message='请选择聊天对象')

            return False

        if self.chatter == USERNAME:

            tkinter.messagebox.showerror('发送失败', message='请不要选择自己')

            return False

        msg = USERNAME + ':;' + self.message\_entry.get() + ':;' + self.chatter

        self.my\_socket.send(msg.encode())

        self.clear()  *# 发送后清空文本框*

        return True

*# 选择聊天对象事件*

    def **select\_chatter**(self, \*args):

        index = self.online\_users\_listbox.curselection()[0]

        if index == 0 or index == 1:

            self.main\_window.title(USERNAME + ' 群聊')

            self.chatter = '群聊'

        else:

            self.chatter = self.online\_users\_listbox.get(index)

            self.main\_window.title(USERNAME + '  -->  ' + self.chatter)

*# 刷新在线列表*

    def **refresh\_list**(self, receive\_data):

        self.online\_users\_listbox.delete(0, tkinter.END)  *# 清空列表框*

        online\_count = ('在线人数: ' + str(**len**(receive\_data)) + ' 人')

        self.online\_users\_listbox.insert(tkinter.END, online\_count)

        self.online\_users\_listbox.itemconfig(

            tkinter.END, fg='black', bg="lightgray")

        self.online\_users\_listbox.insert(tkinter.END, '群聊')

        self.online\_users\_listbox.itemconfig(tkinter.END, fg='black')

        for data in receive\_data:

            self.online\_users\_listbox.insert(

                tkinter.END, data)

            self.online\_users\_listbox.itemconfig(

                tkinter.END, fg='black')

*# 接收服务端发送的信息*

    def **recv**(self):

        while True:

            receive\_data = self.my\_socket.recv(1024)

            receive\_data = receive\_data.decode()

            try:

*# 接收到在线用户列表*

                receive\_data = json.loads(receive\_data)

                self.online\_users = receive\_data

                self.refresh\_list(receive\_data)

            except:

*# 接收到消息*

                receive\_data = receive\_data.split(':;')

                data1 = receive\_data[0]  *# 发送信息的用户名*

                data2 = receive\_data[1].strip()  *# 消息*

                data3 = receive\_data[2]  *# 聊天对象*

                if data3 == '群聊':

                    data2 = data1 + ' to all : ' + data2+'\n'

                    if data1 == USERNAME:  *# 自己发送的*

                        self.message\_aera.insert(tkinter.END, data2, 'blue')

                    else:  *# 接收的*

                        self.message\_aera.insert(

                            tkinter.END, data2, 'green')

                elif data1 == USERNAME or data3 == USERNAME:  *# 私聊*

                    data2 = data1 + ' to '+data3+' : ' + data2+'\n'

                    self.message\_aera.insert(

                        tkinter.END, data2, 'red')

                self.message\_aera.see(tkinter.END)

    def **\_\_del\_\_**(self):

        self.my\_socket.close()  *# 关闭 TCP 连接*

服务器端关键代码：

class **TCP\_Server**():

    def **\_\_init\_\_**(self):

        self.que = queue.Queue()  *# 客户端发送信息队列*

        self.users = []  *# 在线用户信息  [conn, username, addr]*

        self.lock = threading.Lock()  *# 创建锁, 防止多个线程写入数据的顺序打乱*

        self.my\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        self.my\_socket.bind((IP, PORT))

        self.my\_socket.listen(5)

**print**('tcp server online...')

*# 接收客户端发送信息*

    def **tcp\_connect**(self, conn, addr):

        username = conn.recv(1024)

        username = username.decode()

        self.users.append((conn, username, addr))

**print**('新连接:', addr, ':', username)

        online\_list = self.onlines()

        self.recv(addr, online\_list)

        try:

            while True:

                data = conn.recv(1024)

                data = data.decode()

                self.recv(addr, data)  *# 保存信息到队列*

            conn.close()

        except:

**print**(username + ' 断开连接')

            self.delUsers(conn, addr)  *# 移除断开用户*

            conn.close()

*# 判断断开用户在users中是第几位并移出列表, 刷新客户端的在线用户显示*

    def **delUsers**(self, conn, addr):

        index = 0

        for user in self.users:

            if user[0] == conn:

                self.users.pop(index)

**print**('剩余在线用户: ')

                online\_list = self.onlines()

                self.recv(addr, online\_list)

**print**(online\_list)

                break

            index += 1

*# 将接收到的信息存入队列*

    def **recv**(self, addr, data):

        self.lock.acquire()

        try:

            self.que.put((addr, data))

        finally:

            self.lock.release()

*# 将队列que中的消息发送给所有连接到的用户*

    def **sendData**(self):

        while True:

            if not self.que.empty():

                data = ''

                message = self.que.get()

                if **isinstance**(message[1], str):

                    for i in **range**(**len**(self.users)):

                        for j in **range**(**len**(self.users)):

                            if message[0] == self.users[j][2]:

                                data = message[1]

                                break

                        self.users[i][0].send(data.encode())

                data = data.split(':;')[0]

                if **isinstance**(message[1], list):

                    data = json.dumps(message[1])

                    for i in **range**(**len**(self.users)):

                        self.users[i][0].send(data.encode())

*# 将在线用户存入online列表并返回*

    def **onlines**(self):

        online = []

        for user in self.users:

            online.append(user[1])

        return online

    def **\_\_del\_\_**(self):

        self.my\_socket.close()

**参考文献**

[1] 谢希仁. 计算机网络[M]. 电子工业出版社, 2013.

[2] JamesF.Kurose, KeithW.Ross. 计算机网络:自顶向下方法[M]. 机械工业出版社, 2009.

[3] 夏玲, 王伟平. 客户端与服务器端的Socket通信[J]. 电脑知识与技术:学术交流, 2009.

[4] 梁明华. 基于Socket技术的局域网聊天系统设计与实现[J]. 2015.