xx 学院本科生毕业论文

计算机远程控制软件的设计与开发

Computer Remote? Control Software Design and Development

院　　系	计算机科学与工程学院
专　　业	计算机科学与技术
学 生 班 级	xx 级 xx 班
姓　　名	xx
学　　号	xx
指导教师单位	计算机科学与工程学院
指导教师姓名	xxx
指导教师职称	讲师

计算机远程控制软件的设计与开发

计算机科学与技术 2009 级 00 班　xxx

指导教师　xxx

摘要

随着网络技术的飞速发展，远程控制协助软件越来越受到人们的重视。计算机远程控制技术是计算机系统管理人员在异地通过计算机网络连接目标计算机，由本地计算机对远程计算机进行管理和维护的行为。基于 JAVA 与 Socket 编程技术结合的 C/S 远程监控系统软件突破了空间的限制，使用者不用亲自到目标地点，直接通过网络就能实现对被控机器的监控，并且有着友好的操作界面。

本系统采用 Java 网络编程和 Java 图形编程实现。本系统在开发过程中，将计算机网络技术与远程控制理论基础相结合。本远程控制软件包括远程资源管理器、远程监控、远程 CMD 控制台三大模块，实现了对被控机磁盘文件的上传、下载、删除，实现对鼠标、键盘的模拟以及屏幕截取，实现了远程执行 DOS 命令，实现了远程关机。本系统从设计最初到具体实现、优化、测试都严格遵循软件工程的思想。

关键词：Java；Socket；C/S；远程控制；屏幕截取；DOS 命令。

Computer Remote Control Software Design and Development

Computer Science and Technology xxx-xx xxxx

Supervisor xxx

Abstract

With the rapid development of network technology, the remote control to assist software by more and more people's attention. Computer remote control technology is a computer system management personnel in different through the computer network connection target computer, the local computer to the remote computer management and maintenance of behavior. Based on JAVA and Socket programming technology in combination with C/S remote monitoring system software broke through the limitation of space, users need not personally to the target location, directly through the network can achieve for the controlled machine monitoring, and have a friendly operation interface.

This system uses Java network programming and Java graphical programming realization. This system in the development process, the computer network technology and remote control theory foundation combined. The remote control software including remote resource manager, remote monitoring, remote CMD console three modules, realize the accused of machine disk file upload, download, delete, realize the simulation of the mouse, keyboard and screen capture, realized the remote implement DOS command, realized the remote shutdown. This system from design to implementation first, optimization and testing are strictly follow the concept of software engineering.

Keywords: Java； Socket; C/S; Remote control; Screen capture; DOS command.

目录

1 引言

——现如今，随着网络的快速发展，越来越多的企业都建立了自己的内部网络。面对众多的部门联网计算机，对于人数相对偏少的网络管理员来说，如果每台计算机都需要亲临维护既浪费时间，工作效率也极低，因此他们希望对整个网络上的计算机能实现远程控制操作；也希望能实现远程传输文件操作；同时还能查看实时的计算机运行状态和进行一些相关操作；希望能够传输文件；希望能够防止病毒的蔓延、非法程序的拷贝、杜绝某些用户的越权或非法操作等。因此，对于一个网络管理员来说，一个合适的远程协控制软件是至关重要的。

——我们熟知的远程控制技术，最早始于 DOS 时代，当时并没有现在的条件与技术，而且也网络不发达，市场没有更高的要求，所以远程控制技术没有引起许多人的注意。但如今，随着网络的飞速发展，随着人们对电脑的管理及技术支持的需要的不断提高，远程操作及控制技术越来越引起人们的关注。远程控制一般支持下以下网络类型：LAN、WAN、拨号方式、互联网方式。除此之外，实现远程控制的方式还有通过串口、并口、红外端口等通信方式。对于传统的远程控制软件，一般使用 NETBEUI、NETBIOS、IPX/SPX、TCP/IP 等协议来实现远程控制。随着网络技术的发展，越来越多的远程控制软件提供通过 C/S 模式以 Java 语言来开远程控制软件。

对于现如今流行的远程控制软件，一般分两个部分：客户端 Client，和服务器端 Server。实用方法是，先将客户端安装到主控计算机上，将服务端程序安装在被控制电脑上，然后在主控端电脑上执行 Client 端程序，并且向被控端电脑中的 Server 端程序发出信号，建立基于 TCP 协议的远程服务连接，然后通过这个远程服务，使用各种远程控制功能发送远程控制命令，Server 端响应接收到的命令并执行相应的操作，我们称这种远程控制方式为基于远程服务连接的远程控制。通过远程控制软件，我们可以进行很多方面的远程控制，包括截取目标电脑屏幕图像、窗口以及进程目录；记录并提取远端键盘事件；可以打开、关闭目标电脑的任意目录并实现资源共享；管理远端电脑的文件和文件夹；关闭或者重新启动远端电脑中的操作系统等。

上面主要表述的一般是一对一的基于远程服务的远程控制实现原理，其实，如今最实用的远程控制软件最理想的模式应该是一对多，即一台控制机可以控制多台电脑。这对于如今的意义就好像一个大型的企业，如果控制端能够直接控制全部电脑，即显示全

部电脑目录，再进行一些列的操作，如远程桌面协助，远程资源管理器，都将带来极大的方便。

一般的远程控制程序的优点在于，方便技术人员进行远程维护或协助，技术人员再也不用亲临到实地操作，只需通过互联网，就可以方便的进行诸如应用程序的上传，部署，或是对远程故障机的协助操作等等，极大的节省了人力物力，大大的提高了工作效率。

当然，凡是有利必有弊，随着互联网的普及，远程控制技术也是如今黑客恶意攻击的主要手段，诸如一些木马控制程序等，不仅侵犯了他人的隐私，也严重的违反了国家法律。但是远程控制技术对于人类日常生活也是很重要的，如何利用好这门技术造福人类，这也是我研究这门课题的意义。

本软件就是基于此而设计开发的，能实现以下的基本的远程功能：

(1) 查看被控制端的文件目录清单，即远程资源管理器；

(2) 拷贝被控制端的文件到控制端，同时也能上传文件到被控端，或删除被控端文件；

(3) 强迫被控制端重新启动或关机；

(4) 直接执行任何可执行命令，打开应用程序；

(5) 控制被控制端的屏幕，在本地直接操作被控制端计算机；

而且做了一些必要的安全性考虑。

1.1 远程控制软件的研究现状和前景

在飞速发展的今天，包括涵盖远程办公、远程教育、远程维护、远程协助以及企业管理等都属于远程控制涉及的应用领域。随着应用领域的越来越广泛，使用的价值也不断提高，市场需求也越来越明显。目前比较出名的远程控制软件例如PeerYou，VCN，Oray等都具有高效的信息交互和传输能力，以及实现跨平台的高效远程控制。C/S模式远程控制软件充分利用网络资源，以普通PC机为控制设备，通过面向对象以及模块化的程序设计，有着低成本、高实用性、可靠性和可扩展性的优点。综合以上原因，我决定通过一定的理论知识与实践来尝试学习此类知识，并且使用Java编程语言结合C/S模式实现一款远程控制软件。

1.2 课题研究的目的和意义

通过学习远程控制软件的相关技术知识来更深刻的理解 TCP/IP 协议以及 Socket 编程原理；通过实现代码的编写来达到对 Java 语言编程能力的锻炼以及增加对程序代码优化的经验。

1.3 课题研究的主要内容

基于Windows的远程控制软件开发毕业设计的主要任务是要求做出从系统角度出发的基于C/S开发模式与远程控制技术的实用软件。

使用Socket网络编程技术及Java程序开发语言。实现基本的远程控制要求，界面简洁友好。采用面向对象开发技术，严格遵循软件工程设计思想。

要求：

(1) 基于C/S模式架构；

(2) 实现截屏、执行远程CMD命令，远程关机等功能；

(3) 软件安全、稳定、可靠；

(4) 至少能在两台计算机之间进行控制演示；

(5) 界面简洁友好。

2 系统分析

2.1 系统功能分析

本远程控制软件由Server（服务器）和Client（客户端）两部分组成，并且需要使用者在服务端与客户端同时运行相应程序来实现，具体步奏如下：

第一步，服务器端运行相应程序，使服务器端口处于监听状态，这里本远程控制软件使用了三个端口（30018,30011,30012）；启动服务器后，被控端计算机则处于等待连接状态。

第二步，运行客户端程序，输入被控端IP，服务端准备响应，然后程序自动连接到指定IP的远程计算机。到目前为止，整个连接步奏则完成，这个程序之间已经建立了基于TCP协议的远程服务连接，进入使用界面后便可进行相应操作。

2.2 系统软件模型

本系统采用C/S程序开发模式设计，由Server端（服务端）与Client端（客户端）两部分组成，下面是结构图：

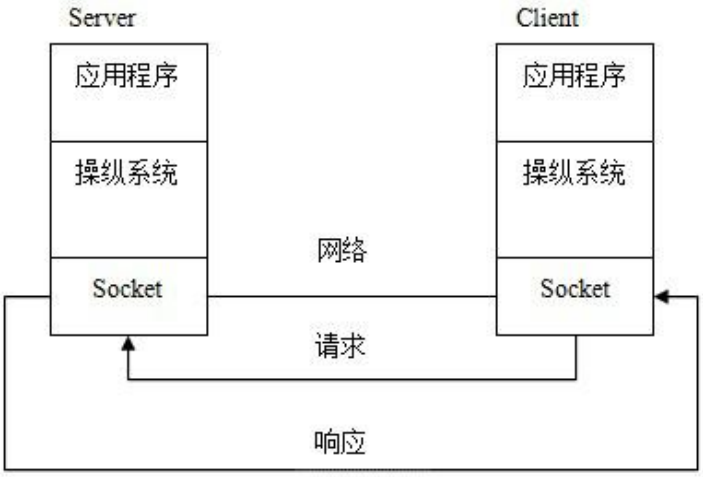


图 2.1 程序结构图

Fig.2.1 The chart of program structure

对客户端的设计，一直遵循面向对象的设计思想，坚持界面友好，易操作；对于服务端的设计则偏于简洁实用，启动后显示界面位于屏幕的右下角，只进行操作的反馈，详尽的设计过程将在接下来介绍。

2.3 系统的开发平台

2.3.1 Eclipse 介绍

本软件实用的主要开发工具之一Eclipse 是一个开放源代码的、基于Java的可扩展开发平台。它本身就是使用Java语言开发的，其本身就包含了括Java开发工具（Java Development Kit，JDK），使用Eclipse进行Java程序开发不仅方便管理，而且也能大大提高开发效率。

2.3.2 Java 语言

早期的Java是使用在家用电器等小型系统的编程语言，被称为Oak。用来解决家用电器的控制和通讯问题，如电视机、电话、闹钟、烤面包机等。由于这些智能化家电的市场需求没有预期的高，Sun放弃了该项计划。就在Oak将要失败的时候，随着互联网的发展，Sun看到了Oak在计算机网络上的广阔应用前景，于是现在广为流传使用的Java诞生了。

Java 编程语言的风格非常接近C、C++。Java是一个完全面向对象的程序设计语言，它继承了 C++ 语言面向对象技术的同时舍弃了C++语言中容易引起错误的指针（以引用取代）、运算符重载（operator overloading）、多重继承（以接口取代）等特性，增加了垃圾回收器功能用于回收不再被引用的对象所占据的内存空间，使得程序员不用再为内存管理而担忧。在 Java SE 1.5 版本中，Java 又引入了泛型编程（Generic Programming）、类型安全的枚举、不定长参数和自动装/拆箱等语言特性。

Java 不同于一般的编译执行计算机语言和解释执行计算机语言。它首先将源代码编译成二进制字节码（byte code），然后依赖各种不同平台上的虚拟机来解释执行字节码，从而实现了“一次编译、到处执行”的跨平台特性。不过，每次的编译执行需要消耗一定的时间，这同时也在一定程度上降低了 Java 程序的运行效率。但在 J2SE 1.4.2 发布后，Java 的执行速度有了大幅提升。

与传统程序不同，Sun 公司在推出 Java 之际就将其作为一种开放的技术。全球数以万计的 Java 开发公司被要求所设计的 Java 软件必须相互兼容。“Java 语言靠群体的力量而非公司的力量”是 Sun 公司的口号之一，并获得了广大软件开发商的认同。这与微软公司所倡导的注重精英和封闭式的模式完全不同。

Sun 公司对 Java 编程语言的解释是：Java 编程语言是个简单、面向对象、分布式、解释性、健壮、安全与系统无关、可移植、高性能、多线程和动态的语言。

Java 平台是基于 Java 语言的平台。这样的平台目前非常流行，因此微软公司推出了与之竞争的.NET平台以及模仿 Java 的 C#语言。

3 主要技术支持

C/S是当今比较流行与具有发展潜力的技术之一。使用C/S模式设计的客户、服务器应用系统具有系统结构优化、资源利用率高、整体运算速度快的优点，因而得到了广泛的应用。Socket网络编程技术也是当今主要的进程之间的通信方式，它利用客户/服务器模式巧妙地解决了进程之间建立通信连接的问题。基于C/S模式于Socket网络编程技术，本远程控制程序才能得以实现。

3.1 Socket

20世纪80年代初，美国政府的高级研究工程机构(ARPA)给加利福尼亚大学Berkeley分校提供资金，让他们在UNIX操作系统下实现TCP/IP协议。在这个项目中，研究人员为

TCP/IP网络通信开发了一个应用程序接口(API)。这个API就称为Socket接口。今天，Socket接口是TCP/IP网络最通用的API，也是在Internet上进行应用开发最为通用的API。

事实上，Socket（套接字）就是在计算机之间提供了一个通信端口。通过这个端口，一台计算机可以与任何具备套接字的网间计算机进行通信。一个Socke只通信的一端，在这一端上可以找到与其对应的一个名字。一个正在被使用的套接口都有它的类型和与其相关的进程，Socket存在于整个通信域中，与对应的并且在相同域的Socket进行数据交换，即通信。应用程序在网络上传输，接收的信息都通过这个套接口来实现。在应用开发种就像使用文件句柄一样，可以对Socket句柄进行读写操作。

开始使用套接字编程之前，首先必须了解什么是网间进程通讯，什么是服务方式，以及C/S软件开发模式。

进程间通信的最初概念来源于单机系统。由于每个进程都在自己的地址区域范围内运行，为了保证进程间能相互通信而又不干涉他们自己的工作，操作系统被要求提供相应的设施，如UNIX BSD中的管道(Pipe)、命名管道(Named Pipe)和软中断信号(Signal)、UNIX System V的消息（Message）、共享存储区（Shared Memory）和信号量（Semaphore）等，但都仅限于用在本机进程之间的通信。网间域中的计算机要通信必须解决不同计算机中进程相互通讯的问题。为此，首先要解决的是网间进程标识问题，同一计算机上，不同进程可以用进程号（Process ID）作为唯一标识，但到了网络环境下，不同的计算机中很可能存在拥有相同进程号的进程，比如A计算机存在进程号为101，B计算机也很可能存在进程号名为101的进程。其次，用于网络通讯的协议众多，不同的协议也有自己独有的辨识方式，因此，要实现网间域内计算机通讯还要解决众多协议识别的问题。

在网络的分层模型中，各层严格遵循着单向依赖，各层之间分工明确，但又相互协作，他们之间的协作主要体现在各相邻层边缘的应用上。“服务”是描述相邻层之间关系的抽象概念，即网络中各层向紧邻上层提供的一组操作。下层是服务的提供者，上层是请求服务的用户。服务的表现形式是原语（Primitive），如系统调用或库函数等。系统调用是操作系统内核向网络应用程序或高层协议提供的服务原语。在国际标准化组织（ISO）的术语中，网络层及其以下各层又称为通信子网，只是实现了点对点之间的通信，没有程序或进程的概念。而传输层实现的是“端到端”通信，引进网间进程通信概念，同时也要解决差错控制、流量控制、数据排序(报文排序)及连接管理等问题。为此提供不同的服务方式：面向连接（虚电路）的服务或无连接的服务。

面向连接服务是电话系统服务模式的抽象,即每一次完整的数据传输都要经过建立、连接、数据传输及终止连接的过程。在数据传输过程中，各数据分组不携带目的地址，而使用连接号（Connect ID）。本质上，连接是一个通信管道，收发数据顺序一直，内容相同。其中TCP协议就提供面向连接的可靠通信协议。

无连接的服务是邮政系统服务的抽象，每个分组都携带完整的目的地址，各分组在系统中独立传送。无连接服务不能保证分组的先后顺序,不进行分组出错的恢复与重传,不保证传输的可靠性。提供无连接的数据报服务的常用协议是UDP协议。

在TCP/IP网络应用中,两个进程之间的相互通信主要基于C/S模式(Client/Server)。即客户端向服务端发出请求，服务端接收待来自客户端的请求好，调用相应的服务。C/S模式的建立基于以下两点：首先，建立网络的起因是网络中软硬件资源、运算能力和信息不均等，需要共享，从而造就拥有众多资源的主机提供服务，资源较少的客户请求服务这一非对等作用；其次，网间进程通信完全是异步的，相互通信的进程间既不存在父子关系，又不共享内存缓冲区，因此需要一种机制为希望通信的进程间建立联系，为两者的数据交换提供同步，这就是基于客户机/服务器式的TCP/IP。

3.2 Socket 分类

TCP/IP的Socket提供下列三种类型套接字。

(1) 原始套接字

该接口允许对较低层协议，如IP，ICMP的直接访问。它通常用于检验新的协议实现或访问现有服务配置中的新设备

(2) TCP流式套接字

如果我们需要一个可靠的连接，用来使数据按顺序、无错的发送到目标端，就需要流式套接字。流式套接字提供一种可靠的面向连接地传输方法。数据无重复、无差错，并且按发送端发出数据的顺序进行接收。不管对单个的数据报或者是一整个数据包，流式套接字都提供一种协议的历史传输——TCP。除此之外，在数据进行传输时，如果一端的连接断开，则另一端的应用程序会接到通知。流式套接字内设流量控制，避免数据流超限；数据被看成字节流，无长度限制。

(3) UDP数据报套接字

数据报套接字是提供一种非连接、不可靠的通信方式。在这里的“不可靠”是指发送的数据不能得到保障，也不保障数据按原来发出时的顺序到达目标端。数据包以独立

包形式被发送，不提供无错保证，数据可能丢失或重复，并且顺序混乱。事实上，一份数据可能不止一次被发送。对于基于Java的Socket网络编程的TCP/IP实现，数据报套接字使用用户数据报协议（UDP）。虽然在通常情况下，在同一台计算机上的两个进程或在轻负载的局域网所连接的两台计算机的进程之间进行通信时，可能不会出现数据包丢失或没按照顺序到达及又重复发送的情况，但在编写实用UDP协议进行进程间通信的程序是，应考虑到这些情况，并且能做出应对措施。当然，如果为非常复杂的网络（如Internet）编写通信应用程序，就应该考虑到数据报套接字的不可靠性。如果应用程序不能很好的处理这些问题，很可能导致程序崩溃。尽管如此，数据报套接字在发送数据包或者记录形数据时仍然有用。另外，数据报套接字还提供向多个目的地发送广播数据包的能力。

3.3 基本套接字函数调用

大多数的数据报套接字应用程序都使用一个规定的事件序列来完成客户应用程序与服务器之间的通信，如下图所示：

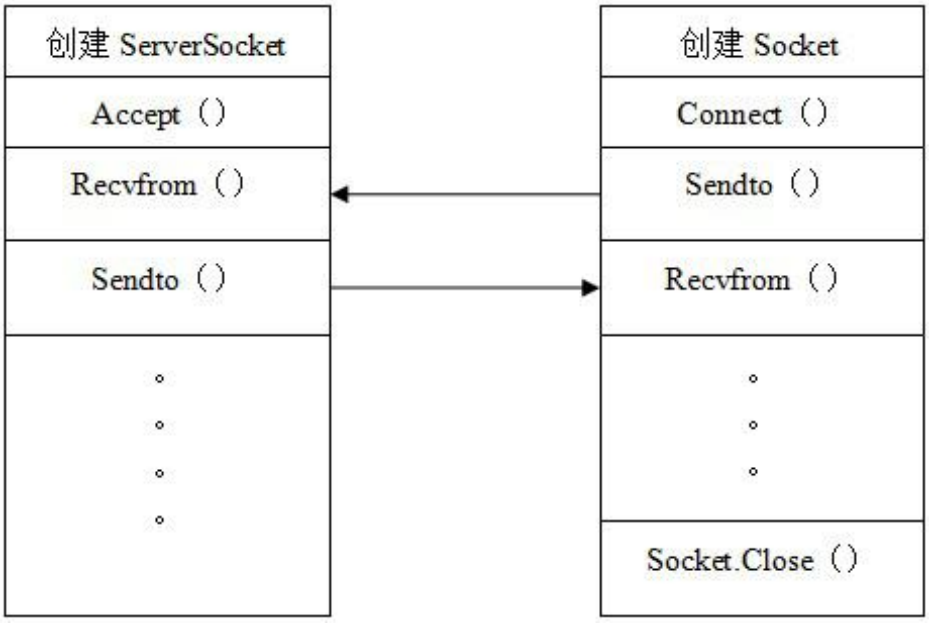


图 3.1 信息交互

Fig.3.1 Information interaction

首先服务端与客户端都要创建一个 Socket（套接字），然后服务端 ServerSocket 绑定端口和 IP，这样客户端就能使用同一端口表示服务器套接字，然后服务器与客户端建立通信，即 Sendto（）和 Recvfrom（）信息交互。下面步奏介绍了套接字使用的步奏。

3.3.1 创建套接字

Java 对 Socket 操作进行了很好的封装，即 java.net.ServerSocket 类，此类实现服务器套接字。服务器套接字等待请求通过网络传入。它基于请求执行某些操作，然后可能向请求者返回结果。在这里我们使用到的构造方法为 ServerSocket(int port)；该构造方法创建以本机 IP 为 IP 地址，以 port 为端口的套接字，参数 port - 端口号；或者为 0，表示使用任何空闲端口。

3.3.2 建立套接字连接

等待连接我们使用了 java.net.ServerSocket 类的 accept（）方法，该方法侦听并接受到此套接字的连接。此方法在连接传入之前一直阻塞。一旦接受到请求，则创建新的套接字与之连接交互信息。

3.3.3 数据传输

有了套接字连接后，我们就可以进行任意的数据传输了。在自定义了 Sendto（）与 Recvfrom（）方法后则可进行任意数据传输。Java 对传输流进行了很好的封装，这里我们主要用到 java.io 以及 javax.imageio 里面的流操作类：

(1) javax.imageio. ImageIO 类

该类包含一些用来查找 ImageReader 和 ImageWriter 以及执行简单编码和解码的静态便捷方法。主要用于远程图像的传输。

read(URL input)方法：返回一个 BufferedImage，作为使用 ImageReader（它是从当前已注册 ImageReader 中自动选择的）解码所提供 URL 的结果。InputStream 是从 URL 中获得的，它被封装在 ImageInputStream 中。

write(RenderedImage im, String formatName, ImageOutputStream output)方法：使用支持给定格式的任意 ImageWriter 将一个图像写入 ImageOutputStream。从当前流指针开始将图像写入 ImageOutputStream，并覆盖该点之后的现有流数据（如果有）。此方法在写入操作完成后不会关闭提供的 ImageOutputStream；一个严谨的程序，应该在实用完后对流进行关闭，减轻系统负担。

(2) java.io. InputStream 类

read()方法：从输入流中读取数据的下一个字节。返回 0 到 255 范围内的 int 字节值。如果因为已经到达流末尾而没有可用的字节，则返回值 -1。在输入数据可用、检测到流末尾或者抛出异常前，此方法一直阻塞。

(3) java.io. OutputStream 类

write(int b)方法：将一个 integer（数组长度）写入此流。

(4) java.io. File 类

主要用于文件或目录的操作。

File(String pathname)方法：通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例。如果给定字符串是空字符串，那么结果是空抽象路径名。

delete()方法：删除此抽象路径名表示的文件或目录。如果此路径名表示一个目录，则该目录必须为空才能删除。

exists()方法：测试此抽象路径名表示的文件或目录是否存在。

isDirectory()方法：测试此抽象路径名表示的文件是否是一个目录。

isFile()方法：测试此抽象路径名表示的文件是否是一个标准文件。

list()方法：返回一个字符串数组，这些字符串指定此抽象路径名表示的目录中的文件和目录。

mkdirs()方法：创建此抽象路径名指定的目录，包括所有必需但不存在的父目录。

listRoots()方法：列出可用的文件系统根。

(5) java.lang.Runtime 类

exec(String command)方法：在单独的进程中执行指定的字符串命令。参数 command：一条指定的系统命令。

(6) java.awt.Robot 类

使用该类提供的一系列方法可以模拟鼠标、键盘操作事件以及屏幕截图，达到远程监控的目的。

createScreenCapture(Rectangle screenRect)方法：创建包含从屏幕中读取的像素的图像。该图像不包括鼠标光标。

mouseMove(int x, int y)方法：将鼠标指针移动到给定屏幕坐标。

mouseWheel(int wheelAmt)方法：在配有滚轮的鼠标上旋转滚轮。

keyPress(int keycode)方法：按下给定的键。应该使用 keyRelease 方法释放该键。

keyRelease(int keycode)方法：释放给定的键。

mousePress(int buttons)方法:按下一个或多个鼠标按钮。应该使用 mouseRelease 方法释放鼠标按钮。

mouseRelease(int buttons)方法：释放一个或多个鼠标按钮。

以上为数据交互使用到的 Java 类，Java 很好的封装了这些操作，使得程序的设计简单又结构性强，更好的体现了 Java 的面向对象特性。

3.3.4 关闭套接字

在数据交互完成后,都要改关闭不在使用的Socket,本远程控制软件使用了 close()方法关闭此套接字。 所有当前阻塞于此套接字上的 I/O 操作中的线程都将抛出 SocketException。 套接字被关闭后，便不可在以后的网络连接中使用（即无法重新连接或重新绑定）。关闭此套接字也将会关闭该套接字的 InputStream 和 OutputStream。如果此套接字有一个与之关联的通道，则关闭该通道。

3.4 C/S 结构特点及发展

C/S 又称 Client/Server 或客户/服务器模式。它是软件系统体系结构，通过它可以充分利用两端硬件环境的优势，将任务合理分配到 Client 端和 Server 端来实现，降低了系统的通讯开销。目前大多数应用软件系统都是 Client/Server 形式的两层结构，由于现在的软件应用系统正在向分布式的 Web 应用发展,Web 和 Client/Server 应用都可以进行同样的业务处理,应用不同的模块共享逻辑组件;但两种应用都有着自己的优势，怎么使用取舍取决于对业务的需求。

传统的 C/S 体系结构虽然采用的是开放模式，但这只是系统开发一级的开放性，在特定的应用中无论是 Client 端还是 Server 端都还需要特定的软件支持。由于没能提供用户真正期望的开放环境，C/S 结构的软件需要针对不同的操作系统系统开发不同版本的软件， 加之产品的更新换代十分快，已经很难适应百台电脑以上局域网用户同时使用。而且代价高， 效率低。但是 Java 的平台无关性可以很好的解决这个问题，实用 Java 开发的应用程序，不管在什么样的操作系统中都能得到很好的支持。

用 C/S 模式设计本系统的优势在于：

(1) 应用服务器运行数据负荷较轻。最简单的 C/S 体系结构的数据库应用由两部分组成，即客户应用程序和服务器程序。二者可分别称为客户端与服务端。运行服务端的

机器，也称为应用服务器。一旦服务器程序被启动，就随时等待响应客户程序发来的请求；客户应用程序运行在用户自己的电脑上，对应服务器，可称为客户电脑，当需要对远程控制端进行操作时，客户程序就自动地寻找服务器程序，并向其发出请求，服务器程序根据预定的规则作出应答，送回结果，应用服务器运行数据负荷较轻。

(2) 服务端对于业务逻辑进行和很好的封装。对于客户端传来的请求，服务端进行很好的处理，这里我们把处理响应的业务逻辑都封装在服务端里，使得客户端显得非常“瘦小”，是软件实用更加灵活。

除此之外，C/S 模式设计的软件能充分发挥客户端 PC 的处理能力，很多工作可以在客户端处理后再提交给服务器。对应的优点就是客户端响应速度快。

3.5 TCP/IP 体系结构

TCP/IP 这个协议遵守一个四层的模型概念:应用层、传输层、网络互联层和网络层

3.5.1 网络层

TCP/IP 模型的基层是网络接口层。负责数据帧的发送和接收，帧是独立的网络信息传输单元。网络接口层将帧放在网上，或从网上把帧取下来。 实际上 TCP/IP 参考模型没有真正描述这一层的实现，只是要求能够提供给其上层—网络互连层一个访问接口，以便在其上传递 IP 分组。由于这一层次未被定义，所以其具体的实现方法将随着网络类型的不同而不同。

3.5.2 互联层

网络互连层是整个 TCP/IP 协议栈的核心。它的功能是把分组发往目标网络或主机。同时，为了尽快地发送分组，可能需要沿不同的路径同时进行分组传递。因此，分组到达的顺序和发送的顺序可能不同，这就需要上层必须对分组进行排序。

网络互连层定义了分组格式和协议，即 IP 协议（Internet Protocol）。

网络互连层除了需要完成路由的功能外，也可以完成将不同类型的网络（异构网）互连的任务。除此之外，网络互连层还需要完成拥塞控制的功能。

互联协议将数据包封装成 internet 数据报，并运行必要的路由算法。

这里有四个互联协议：

(1) 网际协议 IP：负责在主机和网络之间寻址和路由数据包。

(2) 地址解析协议 ARP：获得同一物理网络中的硬件主机地址。

(3) 网际控制消息协议 ICMP：发送消息，并报告有关数据包的传送错误。

(4) 互联组管理协议 IGMP：被 IP 主机拿来向本地多路广播路由器报告主机组成员。

3.5.3 传输层

传输协议在计算机之间提供通信会话。传输协议的选择根据数据传输方式而定。两个传输协议：

(1) 传输控制协议 TCP：为应用程序提供可靠的通信连接。适合于一次传输大批数据的情况。并适用于要求得到响应的应用程序。

(2) 用户数据报协议 UDP：提供了无连接通信，且不对传送包进行可靠的保证。适合于一次传输小量数据，可靠性则由应用层来负责。

3.5.4 应用层

TCP/IP 模型将 OSI 参考模型中的会话层和表示层的功能合并到应用层实现。

应用层面向不同的网络应用引入了不同的应用层协议。其中，有基于 TCP 协议的，如文件传输协议（File Transfer Protocol，FTP）、虚拟终端协议（TELNET）、超文本链接协议（Hyper Text Transfer Protocol，HTTP），也有基于 UDP 协议的。

3.6 多线程

在一个应用程序中，一些独立运行的程序片断被称作“线程”（Thread），利用它编程的概念就叫作“多线程技术”。多线程技术一个常见的例子就是用户界面。利用线程当用户按下按钮后，方法与主界面流程可以一起往下执行，而不是等待方法执行后才继续往下。

在计算机编程中，一个基本的概念就是同时对多个任务加以控制。许多程序设计问题都要求程序能够同步的处理工作，即工作能同时进行，而不是等待一个方法执行完后再接着执行主进程。对于不同的需求，可以通过多种途径达到这个目的。最开始的时候，那些掌握机器低级语言的程序员编写一些“中断服务例程”，主进程的暂停是通过硬件级的中断来实现的。尽管这个方法很有用，但这样的程序很难移植，由此造成了另一类的代价高昂问题。中断对那些实时性很强的任务来说是很有必要的。但对于其他许多问

题，只要求将问题划分进入独立运行的程序片断中，使整个程序能更迅速地响应用户的请求。

最开始，线程只是用于分配单个处理器的处理时间的一种工具。但假如操作系统本身支持多个处理器，那么每个线程都可分配给一个不同的处理器来处理，真正进入“并行运算”状态。从程序设计语言的角度看，多线程操作最有价值的特性之一就是程序员不必关心到底使用了多少个处理器。程序在逻辑意义上被分割为数个线程；假如机器本身安装了多个处理器，那么程序会运行得更快，毋需作出任何特殊的调校。如果仅局限于如上的操作，那多线程就太简单了。其实不然，实用多线程必须解决一个重要的问题：资源共享。如果有多个线程同时运行，而且它们都需要使用到这个资源，就会遇到资源共享的问题。例如两个进程不能同时调用同一台打印机打印，必须等一个进程结束运行后另一个进程在进行调用。为解决这个问题，对那些可共享的资源来说（比如打印机），它们在使用期间必须进入锁定状态。所以一个线程可将资源锁定，在完成了它的任务后，再解开（释放）这个锁，使其他线程可以接着使用同样的资源，这就是所谓的进程同步问题。

多线程是为了同步完成多项任务，不是为了提高运行效率，而是为了提高资源使用效率来提高系统的效率。线程是在同一时间需要完成多项任务的时候实现的。

使用多线程的好处：

(1) 使用线程可以把占据长时间的程序中的任务放到后台去处理，用户界面可以更加吸引人，比如，我在单击了远程控制后，任然能进行资源管理器操作，而不是等远程操作结束后才能进行资源管理器操作。

(2) 程序的运行速度可能加快。因为是“同步执行”的，不需要等待，所以可以加快程序的运行速度。

(3) 在一些需要阻塞等待操作的程序中，线程是很实用的，如数据传输，等待连接，等待输入等，并不一定要上述方法都执行，主进程才能继续执行，这样带来的方便是非常大的。

3.7 Java 远程控制的基本原理

(1) Socket 、ServerSocket

Socket 位于 java.net.包中，这是一个对于网络通信来说及其重要的类，无论是那种语言，都会有 Socket 网络编程的应用方法，存在的差异也仅仅是在表示和组织上

有所不同，Socket 中文称它为套接字，Java API 中可以查看相应的介绍，主要由 ServerSocket 和 Socket 之间建立连接。一个 ServerSocket 构造一对通信套接字方法如下：

```
ServerSocket sr=new ServerSocket(port);
Socket sc=new Socket("ip",port);
```

其中 ServerSocket 的 accpet（）方法十分重要，当一个服务套接字建立之后它会一直阻塞等待一个套接字的请求，直到建立连接。

部分计算机网络的书籍对套接字有这样的定义为端口+IP 地址；一个套接字是由一个由一个 IP 地址和一个端口组成的，在网络通信中的底层实现也的确如此，要建立进程之间的通信就必须使用一个未被占用的端口进行等待连接，在连接端口之后才能在该端口上通信，以实现进程间的通信。

(2) InputStream 、OutputStream

当一个套接字连接成功后就可以获得基于这个套接字的输入、输出流，一切数据的发送和接受都离不开输出、输入流，我们可以通过流封装的方法选择你要输入或者输出的数据类型，通过转换流可以把输入、输出流转换到我们需要的数据传输类型。

(3) 认识 RPC

RPC 是英文远程方法调用的缩写，见名知意，就是在本地计算机上调用远程计算机上的方法或者过程。

既然我们已经能通过 Socket 进行网间进程通讯，进行数据传输，那么，往更深一层去想，如果是传输命令让计算机做出响应呢？因此，实现这一假设的过程，即为远程控制软件设计的基础思想。

在这里 Java 提供了一个非常好的命令响应的封装类——RunTime 类，该类提供方法执行诸如 CMD 控制命令。

```
Runtime ec=Runtime.getRuntime();
ec.exec("命令");
```

Exec()方法就可以放入你想要运行的命令这样你就可以在机器上运行你的指令了。那么到目前为止，一个基于 Java 语言开发的远程控制程序的雏形就展现在我们面前了，试想，如果我们能远程调用 CMD 命令已经相当于无所不能了。

以上就是一个简易的 Java 远程控制的实现，当然本系统的设计要比此步奏复杂许多，但是基本原理相同。

4　C/S 模式远程控制系统的设计实现

4.1 主要实现功能

(1) 查看被控制端的文件目录清单；

(2) 下载、上传、删除文件；

(3) 强迫被控制端重新启动或关机；

(4) 直接执行任何可执行命令，打开应用程序；

(5) 控制被控制端的屏幕，在本地直接操作被控制端计算机；

4.2 Client（监控端）设计

Client 的界面设计使用的是 Java 提供的 GUI，具体界面如下图

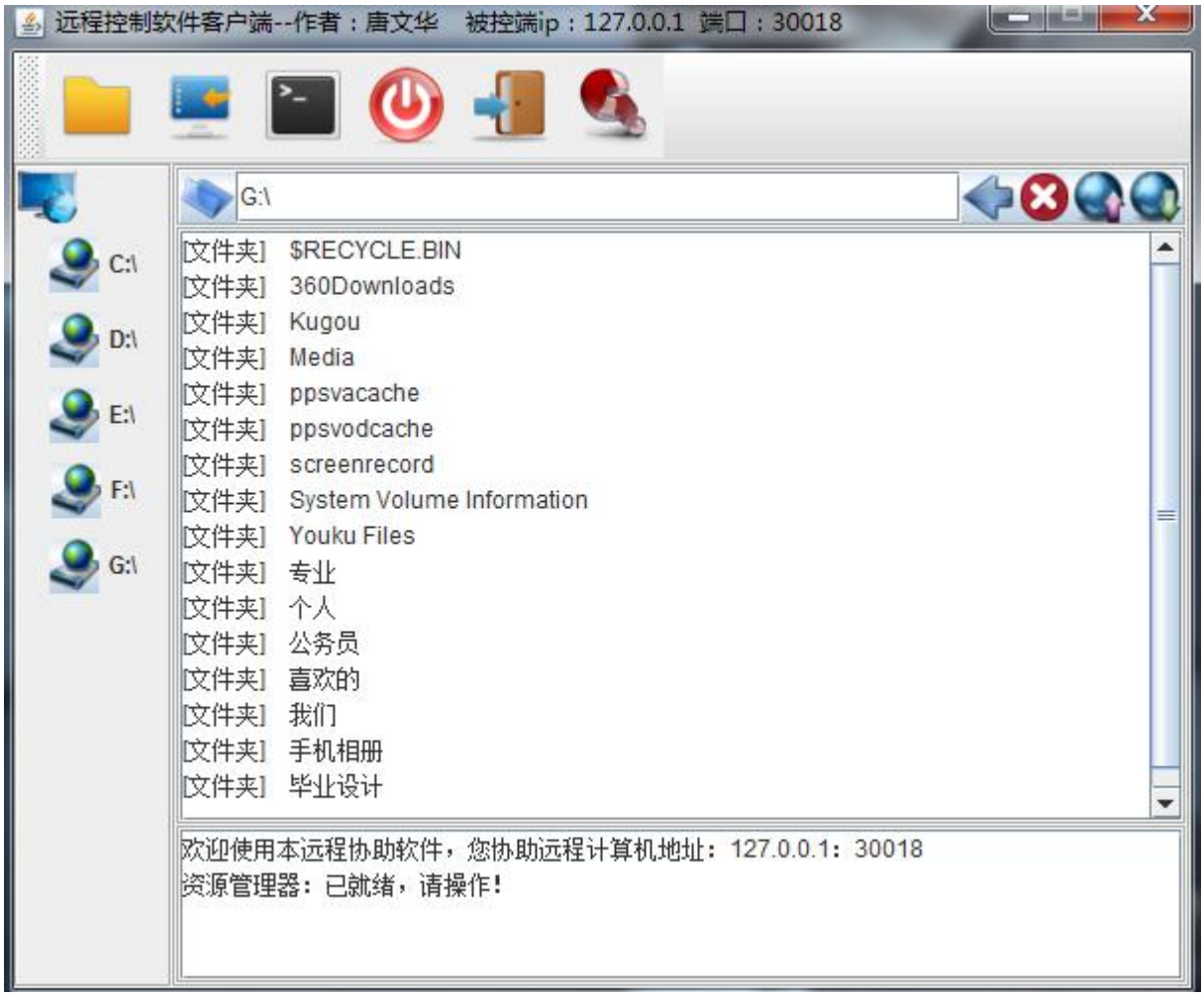


图 4.1 客户端

Fig.4.1 Client

主要包括登录、远程资源管理器、远程控制监视、远程控制台、远程关机、退出以及帮助。主要用于发送相应命令以及处理相应返回信息，具体流程如下图所示：

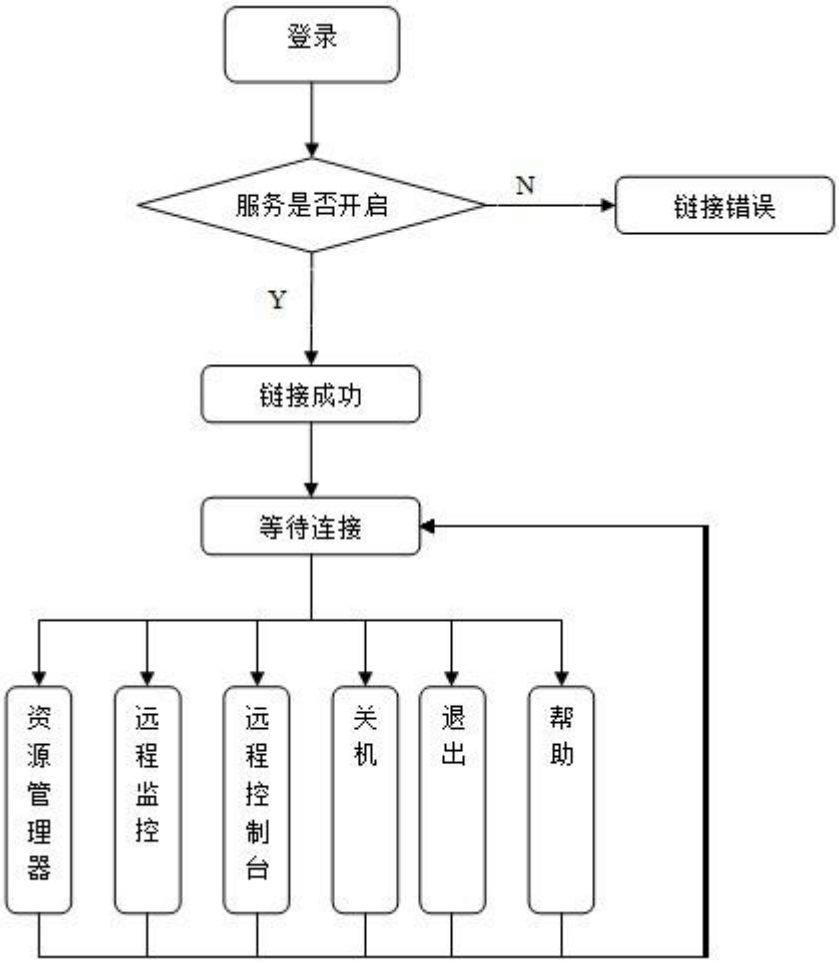


图 4.2 客户端流程

Fig.4.2 The client processes

4.2.1 文件操作 client.files

文件操作包含客户端的上传文件（SendFile）、下载文件（DownFile），文件一般操作（FileControl）以及初始化被控端磁盘（getIniDrivers）。

(1) 初始化被控端磁盘（getIniDrivers）

作为资源管理器第一个步骤，首先要初始化磁盘，既获得被控端磁盘驱动情况，具体体现在操作界面右侧的磁盘目录树上，主要实现方法为程序启动之初，接受由 Server 端发来的磁盘信息，然后解析为磁盘字符数组，再体现到操作界面上。该类主要在初始化的时候调用。

主要用到 public String[] getDrivers()方法，该方法接收来自服务器的数据，并转换为字符串数组。然后初始化磁盘数到显示界面，代码如下：

```
for (int i = 0; i < countc; i++) {
    pci[i] = new JPanel();
    cc.add(pci[i]);
    cc1[i] = new JButton(new ImageIcon(
            "src\\client\\imges\\hd-network.png"));
    cc1[i].setPreferredSize(new Dimension(25, 28));
    cc1[i].setRequestFocusEnabled(false); // 设置不需要焦点
    cc1[i].setBorderPainted(false);// 不绘制边框
    cc1[i].addActionListener(this);
    pci[i].add(cc1[i]);
    cc2[i] = new JLabel(drivers[i]);
    pci[i].add(cc2[i]);
}
```

其中 drivers[i]就是调用初始化磁盘方法返回的字符数组。

(2) 文件一般操作（FileControl）

作为资源管理器的重要类，包含了文件操作命令的发向 Server 端，以及接受 Server 端返回的操作结果，并且解析为文件数组反馈到操作界面。主要流程为下图所示：

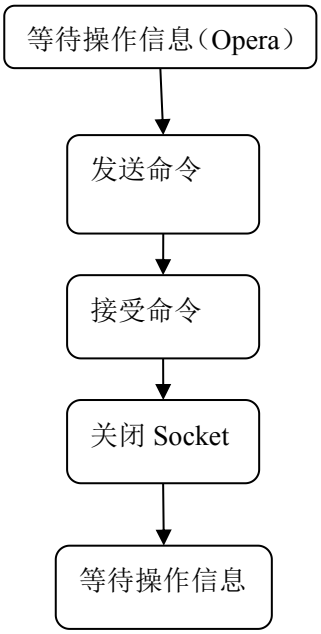


图 4.3 操作流程

Fig.4.3 Operation processes

该类中包含文件操作的相关方法,比如public String[][] getFiles(String opera)方法，public boolean deleteFile(String opera)，public boolean upFile(String opera, String upload)和public boolean downFile(String opera, String download)方法。

public String[][] getFiles(String opera)为根据地址获得目录列表的方法，返回字符数组，字符数组包括文件类型，文件名称、文件绝对地址，部分代码如下：

```
InputStream is = socket.getInputStream();
DataInputStream dis = new DataInputStream(is);
String info = "";
info = dis.readUTF();
is.close();
socket.close();
if (info != null && info != "") {
String s[] = info.trim().split(";");
if (s[0].equals("tj")) {
files = new String[s.length - 1][3];
for (int i = 0; i < s.length - 1; i++) {
String[] temp = s[i + 1].trim().split(",");
if (temp.length >= 3) {
files[i][0] = temp[0];
files[i][1] = temp[1];
files[i][2] = temp[2];
} else {
String t = m.info.getText();
m.info.setText(t + "\n 文件信息：port（" + port+ "）文件读出失败!");
}
}
return files;
```

```
}
}
```

该部分代码完成的功能是获取字符串，并根据分好分割成字符数组，每一个字符数组单元再更具逗号再分成二维字符串数组，最后返回这个二维字符数组 files，由前台显示出目录。

public boolean deleteFile(String opera)方法根据地址删除服务端相应文件，部分代码如下：

```
OutputStream os;
os = socket.getOutputStream();
DataOutputStream dos = new DataOutputStream(os);
dos.writeUTF(opera);
dos.flush();
```

完成功能为将调用该方法的传入的地址参数发送到服务端，服务端完成删除并反馈

public boolean upFile(String opera, String upload)方法为上传方法，参数 Opera 为上传目录地址，upload 为客户端上传文件地址，部分代码如下：

```
try {
        if (opera == null || opera == "") {
            socket.close();
            return false;
        }
        OutputStream os;
        os = socket.getOutputStream();
        DataOutputStream dos = new DataOutputStream(os);
        dos.writeUTF(opera);
        dos.flush();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    new SendFile(socket, upload, this.m).start();
```

其中最为重要的是最后一行 new SendFile(socket, upload, this.m).start()，启

用了一个新的线程进行传输。

public boolean downFile(String opera, String download)方法为下载服务端文件，其中参数 opera 为下载文件地址，download 为保存到本机地址，部分代码如下：

```
try {
        if (opera == null || opera == "") {
            socket.close();
            return false;
        }
        OutputStream os;
        os = socket.getOutputStream();
        DataOutputStream dos = new DataOutputStream(os);
        dos.writeUTF(opera);
        dos.flush();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    new DownFile(socket, download, this.m).start();
```

最后一行 new DownFile(socket, download, this.m).start();启用了一个新的线程下载文件。

4.2.2 远程控制台 client.cmd

主要包括 2 部分：产生一个新的 CMD 控制台窗口，包括命令行输入，帮助，命令行显示，单击和回车时间监听；发送命令到服务器，接受服务器返回的处理结果，并显示在窗口中。

(1)窗体：

图 4.4 CMD 操作

Fig.4.4 CMD operation

(2)命令的发送与接收

命令的发送方法为 public boolean cmdOpera(String opera)，该方法实现了命令的传入，即参数 opera，命令的发送与返回值接收处理并反馈到前台。部分代码如下：

```
OutputStream os;
os = socket.getOutputStream();
DataOutputStream dos = new DataOutputStream(os);
dos.writeUTF("cmd;"+opera);
dos.flush();
```

该部分为命令的发送，返回结果的接收为：

```
InputStream is = socket.getInputStream();
DataInputStream dis = new DataInputStream(is);
String info = "";
info = dis.readUTF();
is.close();
socket.close();
if (info != null && info != "") {
    str.append("\n<--返回信息-->\n"+info);
    this.info.setText(str.toString());
    return true;
}
```

接收到的结果直接显示在嵌套 TextArea 控件上，方法为 this.info.setText (str.toString());

4.2.3 远程监控 client.view

用于远程监控的主要操作，包括远程图像接收（RecivImg）、发送事件信息（SendOperate）以及主要显示窗体（CWindow）

(1) 图像接收

通过接收服务端发来的经过压缩的流，并解压为图像流，最终显示到窗体中。为了传输更为稳定，我们采用了压缩流，这里体现在客户端的为解压。部分代码为：

```
socket = new Socket(rec.getIp(), this.port);
DataInputStream dis = new
DataInputStream(socket.getInputStream());
ZipInputStream zis = new ZipInputStream(dis);
Image image = null;
try {
    zis.getNextEntry();// 读取下一个 ZIP 文件条目并将流定位到该条目数据的开始处
    image = ImageIO.read(zis);// 把 ZIP 流转换为图片
    ImageIcon img = new ImageIcon(image);
```

```
                if (img != null) {
                    rec.jlabel.setIcon(new ImageIcon(image));
                    rec.scroll.setViewportView(rec.jlabel);
                    rec.validate();
                }
            } catch (IOException ioe) {
                System.out.print("1");
            }
```

用到的是 Java 提供的 Zip 流进行压缩，显示用到的是 JLable 控件，即把接收到的图片设置为 JLable 的背景，实现方法为 rec.jlabel.setIcon(new ImageIcon(image))；最重要的是图像传输用到的是 ImageIO 流。为了使图像连贯，我们设置接收图片的间隔为 50 毫秒，具体方法为 rec.jlabel.setIcon(new ImageIcon(image))。

(2) 发送事件信息

主要是封装了信息发送的方法，具体事件在窗体单击等事件发生 时候才进行非法调用。部分代码如下：

```
        Socket socket = new Socket(this.ip, this.port);
        OutputStream os = socket.getOutputStream();
        os.write((this.operateStr).getBytes());
        os.flush();
        socket.close();
```

(3) 窗体

布局很简洁，只有一个 JLable 作为图像显示背景，原理为不断用接收来的新图像信息替换 JLable 原有的背景，达到动态视频监视的效果。主要实现的事件有鼠标下压、鼠标释放、鼠标拖拽、鼠标移动、键盘下压、键盘抬起等事件。当事件触发的时候边通过发送消息类的方法传递到远程被控端，由被控端方法响应操作达到远程控制的目的。

鼠标下压事件监听代码：

```
public void mousePressed(MouseEvent e) {
            super.mousePressed(e);
            int x = (int) e.getX() + (int)
CWindow.this.scroll.getHorizontalScrollBar().getValue();
```

```
                int y = (int) e.getY() + (int)
CWindow.this.scroll.getVerticalScrollBar().getValue();
                String operateStr ="mousePressed,"+ x + "," + y + "," +
e.getModifiers();
                SendOperate sender=new SendOperate(CWindow.this.ip,
(operateStr));
                new Thread(sender).start();
            }
```

主要发送的信息为 operateStr，包含操作类型（mousePressed），鼠标坐标 x，y，最后启用新的线程传输。鼠标释放方法与下压事件类似，不同的是传输字符串为操作类型（mouseReleased），鼠标坐标 x，y；对于鼠标的操作还有 mouseDragged（拖拽），mouseMoved（移动）。

键盘监听事件：

```
            public void keyPressed(KeyEvent e) {
                super.keyPressed(e);
                String operateStr ="keyPress,"+ e.getKeyCode();
                SendOperate sender=new SendOperate(CWindow.this.ip,
(operateStr));
                new Thread(sender).start();
            }
            public void keyReleased(KeyEvent e) {
                super.keyReleased(e);
                String operateStr ="keyReleas,"+ e.getKeyCode();
                SendOperate sender=new SendOperate(CWindow.this.ip,
(operateStr));
                new Thread(sender).start();
            }
```

实现键盘单击事件监听，传输的字符串为操作类型（keyReleas、keyPress），加键值，发送到服务端后由服务端解析处理。

4.2.4 关机

通过消息发送出 ShutDown 命令，由被控端 CMD 命令响应，并做出关机。部分代码如下：

```
if (JOptionPane.showConfirmDialog(null, "您确定要关闭远程控制端计算机？", "提醒",
        JOptionPane.YES_NO_OPTION) == 0) {
    si.sendOpera("cmd;Shutdown/s");
}
```

主要通过 CMD 命令 ShutDown 实现关机，传输操作的方法为 sendOpera("cmd;Shutdown －s");

4.2.5 退出

返回登录界面或关闭程序。部分代码为：

```
if (JOptionPane.showConfirmDialog(null, "是否返回登录", "提醒",
        JOptionPane.YES_NO_OPTION) == 0) {
    si.sendOpera("ini");
    Main.this.dispose();
    System.out.println("窗体关闭");
    System.gc();
    new Login();
} else {
    Main.this.dispose();
    System.out.println("窗体关闭");
    System.gc();
}
```

4.2.6 帮助

(1) CMD 帮助

列出常用的 CMD 命令。

(2) 一般帮助

列出软件相关信息。

4.3 Server（被控端）设计

被控端程序是远程控制软件的主体。远程控制软件的具体业务逻辑都是在被控制端实现的，主控端只负责传送要执行的命令和显示返回的结果。而几乎所有的操作控制都时在被控端本地实现的。具体设计思想如下图：

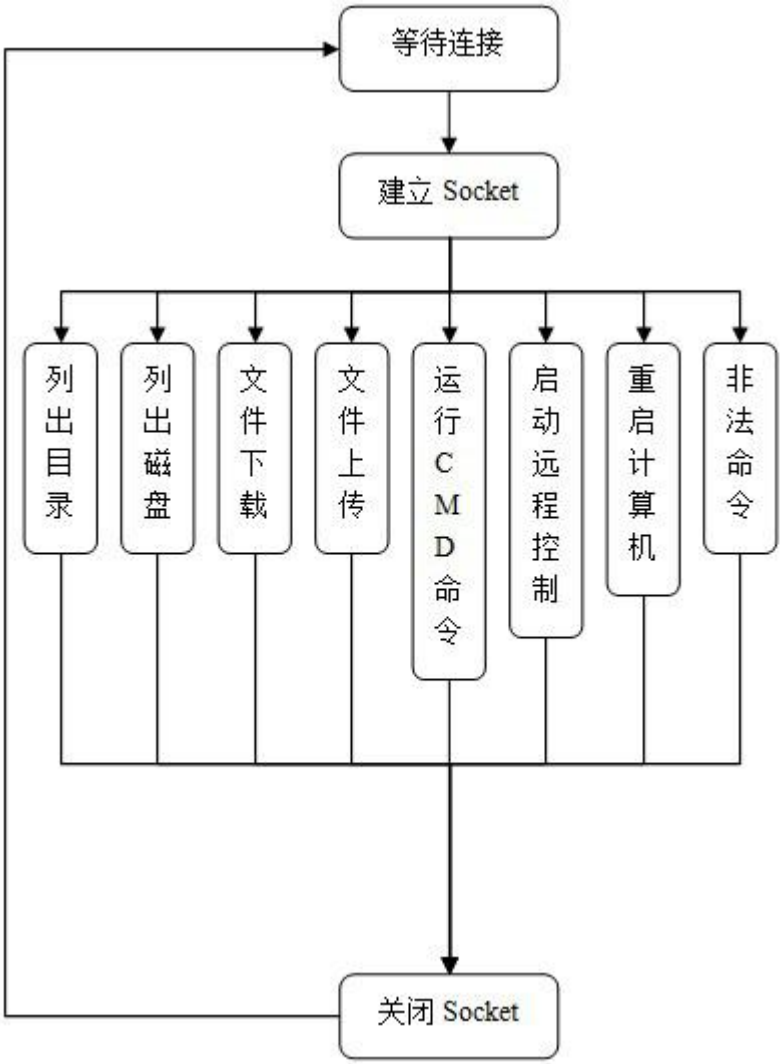


图 4.5 被控端流程

Fig.4.5 Server processes

4.3.1 文件操作（server.files）

(1) Dofile.java

文件操作基础类，包括 deletefile(String delpath, StringBuffer sb)：删除文件或文件夹；

readfile(String filepath)：读取文件或文件目录；只需要被控端接收到文件或文件目录地址，调用方法即可。部分代码如下：

```
        if (!file.isDirectory()) {
            file.delete();
            msg.append(temp.toString() + "" + delpath + ",删除成功!,");
        } else if (file.isDirectory()) {
            String[] filelist = file.list();
            msg.append("," + temp.toString() + "开始删除!," +
temp.toString()
                    + delpath + ",");
            temp.append("[scWhiteSpase]");
            for (int i = 0; i < filelist.length; i++) {
                File delfile = new File(delpath + "\\" + filelist[i]);
                if (!delfile.isDirectory()) {
                    delfile.delete();
                    msg.append(temp.toString() + "-"
                            + delfile.getAbsolutePath() + "删除成功!,");
                } else if (delfile.isDirectory()) {
                    deletefile(delpath + "\\" + filelist[i], temp);
                }
            }
            temp.delete(0, 14);
            file.delete();
            msg.append(temp.toString() + delpath + "," + temp.toString()
                    + "删除完成!" + ",");
        }
```

首先判断该地址是目录还是文件 file.isDirectory()，文件则直接删除，目录则先便利目录子文件或目录，若文件则直接删除，若仍为目录则递归调用删除方法。

读取目录代码：

```
if (file.exists() && !file.isDirectory()) {
        StringBuffer files = new StringBuffer();// type name path
        files.append("文件,");
        files.append(file.getName() + ",");
        files.append(file.getAbsolutePath() + ";");
        this.setFilePath(files.toString());
        return true;
    } else if (file.exists() && file.isDirectory()) {
        String[] filelist = file.list();
        StringBuffer files = new StringBuffer();// type name path
        if(filelist==null){
            msg.append("未找到文件！");
            return false;
        }
        for (int i = 0; i < filelist.length; i++) {
            File readfile = new File(filepath + "\\" + filelist[i]);
            if (readfile.isDirectory()) {
                files.append("文件夹,");
                files.append(readfile.getName() + ",");
                files.append(readfile.getAbsolutePath() + ";");
            } else if (!readfile.isDirectory()) {
                files.append("文    件,");
                files.append(readfile.getName() + ",");
                files.append(readfile.getAbsolutePath() + ";");
            }
        }
        this.setFilePath(files.toString());
        return true;
    }
```

遍历地址标注的子目录或子文件，人后分别判断类型如：文件或文件夹，文件或文件夹安分号隔开，文件或文件夹信息用逗号隔开，把最后组成的字符串传送回客户端解析成目录。

(2) FilesOpera.ava

响应操作命令的基本类，原理是接收控制端传入才操作命令，解析后分别调用相应的方法，如：delete；然后获得返回值并反馈给控制端。

基本响应的方法有：

if ("tj".equals(s[0]) && s[1] != "")响应客户端发送的打开方法，调用readfile();

if ("sc".equals(s[0]) && s[1] !="")响应客户端的删除方法,调用deletefile();

if ("xz".equals(s[0]) && s[1] != "")响应客户端的下载方法，调用新的线程进行传送new SendFile(socket, path).start();

if ("up".equals(s[0]) && s[1] != "")响应和护短的上传方法，调用线程进行上传new DownFile(socket, path).start();

(3) IniDrivers.java

主要是读取被控端磁盘数，并反馈给控制端，只在初始化的时候调用。部分代码为：

```
File roots[] = File.listRoots();
StringBuffer temp = new StringBuffer();
for (int i = 0; i < roots.length; i++) {
    temp.append(roots[i].toString() + ";");
}
this.drivers = temp.toString();
```

完成的功能是便利本机的磁盘驱动，记录下盘符，行车字符串传送至客户端进行解析。

(4) DownFile.java

下载文件的操作，类似于控制端的下载文件。部分代码为：

```
byte[] buf = new byte[this.bufferSize];
long fileLength = 0L;
this.savePath += inputStream.readUTF();
DataOutputStream fileOut = new DataOutputStream(
```

```
            new BufferedOutputStream(new BufferedOutputStream(
                    new FileOutputStream(this.savePath))));
        fileLength = inputStream.readLong();
        str.append("\n 文件的长度为:" + fileLength + "\n");
        str.append("开始接收文件!\n");
        while (true) {
            int read = 0;
            if (inputStream != null) {
                read = inputStream.read(buf);
            }
            if (read == -1) {
                break;
            }
            fileOut.write(buf, 0, read);
        }
```

主要经过数据流转换成套一个缓冲流，最后文件输出流输出文件。

(5) UpFile.java

上传文件的操作，类似于被控端的上传文件。部分代码：

```
        DataInputStream fis = new DataInputStream(new
    BufferedInputStream(new FileInputStream(this.filePath)));
      DataOutputStream ps = new
DataOutputStream(this.socket.getOutputStream());
        ps.writeUTF(fi.getName());
        ps.flush();
        ps.writeLong(fi.length());
        ps.flush();
        int bufferSize = 1024;
        byte[] buf = new byte[bufferSize];
        while (true)
        {
```

```
            int read = 0;
            if (fis != null) {
              read = fis.read(buf);
            }
            if (read == -1) {
              break;
            }
            ps.write(buf, 0, read);
          }
```

完成功能为使用文件流读出文件，在套一层缓冲流和一层数据传输流，最后传输到客户端。

4.3.2 CMD 控制台（server.cmd）

主要原理是使用了 Java 提供的 java.lang.Runtime 类的 exec(String command)方法，command 为命令，在单独的进程中执行指定的字符串命令。然后通过流把执行结果传输回控制端。部分代码：

```
Process pro = null;
    try {
        pro = Runtime.getRuntime().exec("cmd /c " + this.msg);
    } catch (Exception e) {
        e.printStackTrace();
    }
    InputStreamReader ipsr = new InputStreamReader(pro.getInputStream());
    BufferedReader br = new BufferedReader(ipsr);
    String cmdResult = "";
    String msgs = null;
    try {
        while ((msgs = br.readLine()) != null)
            cmdResult = cmdResult + msgs + "\n";
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
```

主要是新建一个进程执行客户端传来的命令，Runtime.getRuntime().exec("cmd /c " + this.msg)；然后读取进程的执行结果，pro.getInputStream()读出流，转换传输到客户端解析显示。

4.3.3 远程监控实现（server.view）

主要使用的是 java.awt.Robot 类。此类用于为测试自动化、自运行演示程序和其他需要控制鼠标和键盘的应用程序生成本机系统输入事件。Robot 的主要目的是便于 Java 平台实现自动测试。

使用该类生成输入事件与将事件发送到 AWT 事件队列或 AWT 组件的区别在于：事件是在平台的本机输入队列中生成的。例如，Robot.mouseMove 将实际移动鼠标光标，而不是只生成鼠标移动事件。

因此，我们从控制端获得的命令后，加以解析判断后调用 Robot 封装的相应方法，达到远程控制的目的。

鼠标移动：robot.mouseMove(x, y);

鼠标释放：robot.mousePress();

鼠标下压：robot.mouseRelease();

由于方法中代码比较多，更多代码请参见附录。

4.4 软件测试与分析

测试是确保系统质量的重要手段，是开发过程中最后一个阶段。软件测试主要包含黑盒测试与白盒测试，就是在指定条件下对软件进行操作并评估操作结果的过程。在软件测试过程中应该使用各种测试用例来促使错误的发生。总的来说，软件测试是在运行过程中发现软件中存在的毛病。目的检测软件是否能满足需求，并且是否运行稳定。

4.4.1 软件测试的重要性

系统测试在软件的整个生命周期中占据重要的地位，在传统的瀑布模型中，系统测试学存在于运行维护阶段之前，是系统交付使用前进行质量保证的重要手段。如今，软件工程界逐渐趋向于一种新的观点，即认为软件生命周期的每一阶段中都应包含测试过

程，用来检测本阶段是否达到预期的目标，尽可能早的发现错误，并加以修正。如果早期存在的错误没有被发现，那么遗留到最后可能会造成不可修复的损失。

实际上，不乱采用什么技术、方法开发的软件，都存在错误。采用新的语言、先进的开发方式、完善的开发过程，可以减少错误的引入，但是绝不可能完全避免错误，这些存在的错误需要进行不断的测试来找出。测试是所有工程学科的基本组成单元，是软件开发的重要部分。自有程序设计的那天起测试就一直伴随着。据统计，在软件开发项目过程中，软件测试工作量往往占总工作量四成以上。而在总成本中，用在测试上的开销要占五成左右。如果把维护阶段也算在内，评估整个软件生存周期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含有许多测试工作。

4.4.2 测试实例的研究与选择

按照软件内部逻辑到外部功能的角度划分，一般软件的测试方法有白盒测试、灰盒测试以及黑盒测试；从执行程序的角度划分，分为静态测试和动态测试；从软件开发阶段划分，有单元测试、集成测试、确认测试、系统测试以及验收测试。而本远程控制软件测试的主要采用黑盒测试。

白盒测试：一种是以程序的内部逻辑结构为依据而设计测试用例的方法，因而又称结构测试或玻璃盒测试，将系统看成一个透明的白盒子，按照程序的内部结构和处理逻辑来选定测试用例，对系统的逻辑路径及过程进行测试，检查它与设计是否相符。白盒测试就是要选取足够的测试用例，对源代码实行比较充分的覆盖，以便尽可能多地发现程序中的错误。主要有两种方法：一种称为逻辑覆盖法，另一种称为路径覆盖法。

黑盒测试：也称功能测试，数据驱动测试等，它将待测对象堪称是一个黑盒子，在完全不考虑程序的内部结构和特性的情况下，只依据规格说明书检查程序的功能是否能正常使用。

黑盒测试主要是根据输入条件和输出条件的确定测试数据，来检查程序是否能产生正确的输出。进行黑盒测试主要有下面几种方法：等价分类法、边界值分析法、猜错法、因果图法。

本远程控制软件主要使用的是黑盒测试方法。在测试当中遵循了实时测试以及使用完整测试用例的原则。通过测试达到以下测试目的：

(1) 功能检查：检查功能是否正确，是否遗漏或实现不了应该实现的功能等。

(2) 接口检查：检查能否正确地接受信息或输出信息。

(3) 性能检查：检查性能需求能否得到满足。

(4) 初始化、终止检查：检查是否能进行正确地初始化或终止。

4.4.3 测试环境与测试条件

处理器：Inter(R) Core (TM)2 Duo T5750

内存：2GB

硬盘：250G

操作系统：Windows XP

4.4.4 系统部分模块测试情况

(1) 登录

启动服务端，输入错误的 IP 地址，如“xxxx”，单击确认后查看结果。

预期结果：磁盘不能初始化，信息栏提示“port：30018 链接失败!”

实际结果：与预期结果相符。

(2) 删除驱动

单击盘符选择驱动，如“c：\”，单击【删除】按钮，查看结果。

预期结果：提示“不能删除磁盘！”，并且不进行如何操作。

实际结果：与预期结果相符。

(3) 地址栏为空测试

地址栏为空，分别单击【打开】，【后退】，【删除】，【上传】，【下载】。

预期结果：弹出提示框，“地址栏不能为空”。

实际结果：与预期结果相符。

(4) 远程控制测试

单击远程控制，分别对远程左面进行单击、右键、打击字符操作。如：右键新建一个 TXT 文档，打开后输入 hello ，然后保存。

预期结果：文档新建成功，字符输入成功，保存成功。

实际结果：与预期结果相符。

(5) CMD 测试

单击 CMD 控制台，输入 CMD 命令，如 IPCONFIG/ALL。

预期结果：返回 IP 信息。

实际结果：与预期结果相符。

5 总结与展望

通过对本程序的设计、编写过程，使我加深了对网络编程学习的了解，并且通过查阅相关资料，把网络编程与 Java 结合，让我了解到了 Java 与网络编程的更多方面，不仅加深了 C/S 设计模式的理解，也体会到了 C/S 模式开发带来的好处，同时也加深了对系统结构方面的熟悉。

在设计之初，由于对网络编程的不了解，只能通过一个个小实验去体会网络编程涵盖各方面的意义；之后对网络编程以及 Java 提供的类有着一定的了解后，便尝试着把系统划分成一个个小模块，通过不断的学习和实验，最终把一个个小模块完成，实现从远程资源管理器、上传、下载、删除、远程 CMD、到最后的远程视频监控，不断的优化代码，使其简洁高效，最后在突破局域网限制，进行互联网测试……一步步走来，到最后融合成一个完整的系统，都是不断学习与尝试的结果。

由于时间的关系，并没有很多时间来使程序更加完善，比如上传与下载文件并没有使用压缩的技术，而且对于互联网远程控制对网上的要求也可以再优化——例如并没有实现一些文档的打开或者在线编辑功能。因此，本远程控制软件还有待完善，但对于如今阶段来说，已经按要求完成了原来设定下来的目标，对于改进的功能，将于以后有机会再不断完善。

致谢

通过近五个月来的忙碌和学习，本次毕业论文设计已接近尾声。作为一个本科生的毕业设计，由于经验的匮乏，难免有许多考虑不周全的地方，在这里衷心感谢指导教师周培春老师对我的指导、评审以及意见修改。正是有周培春老师为我指明开发方向，并不断给我建议与提醒，更主要的是，他不仅没有催促我缓慢的进度，而是选择对我的支持与信任，非常感熊周培春老师！同时我还要感谢在我学习期间给我极大关心和支持的各位老师以及关心我的同学和朋友们，特别感谢培养我长大含辛茹苦的父母，谢谢你们！

本设计编写过程中，参考了《黑天鹅远程控制软件》以及《JDK_API_1_6_zh_CN.CHM》等帮助手册，对于这些作者，我也衷心的感谢。同时，我也感谢玉林师范学院这几年来对我的栽培。本次毕业设计中有很多细节问题，需要我们去耐心地查阅书籍，浏览资料；设计中需要用到设计软件的地方，也需要我们耐心的学习，掌握其使用的要领，运用到设计当中去。通过这次毕业设计，我的动手能力和资料收集能力也得到了提升，这对我日后的工作和生活有着非常积极的影响。

参考文献

[1] 薛宝赏.远程控制 新手也能行[J].电脑爱好者：普及版，2009，(12)：42-44.

[2] 华龙.计算机远程控制中 Desktop 工具（RDP）原理解析与实现[J].电子商务，2012，(9)：64-65.

[3] 张礼程.计算机远程控制技巧以及应用方法研究[J].计算机光盘软件与应用，2012，(16)：129-130.

[4] 杨材.计算机远程控制软件设计与实现[J].电脑编程技巧与维护，2012，(15)：86-89.

[5] 郭玉芝.基于 Socket 的实验室教师机远程控制系统的设计与研究[D].中国海洋大学，2011.

[6] 阮丽江.基于 Windows 的远程控制软件设计与实现[D].郑州大学，2008.

[7] 陈宏.基于嵌入式 WinCE 的 UDP 通信[D].电子科技大学，2011.

[8] 马春梅.计算机远程控制系统的设计与实现[D].天津：天津大学，2005.

[9] 刘洪宇. Java 难因互搏妥协而前途明朗[N].中国计算机报，2010-10-25，029.

[10] 张晨阳.远程控制“看我的”[J].网管员世界，2012，(15)：119-119.

[11] 张海藩.软件工程导论(第四版)[M].北京：清华大学出版社，2002，5-10.

[12] 郑魁敬，袁磊，周鑫.基于 C/S 结构的数控设备网络监控系统[J].机械设 计，2012，29(8)：5-10.

[13] Christina Ballantyne.Online Evaluations of Teaching： An Examination of Current Practice and Considerations for the Future[J]. New Directions for Teaching and Learning，
2003(96) :103-112.

[14] 夏利民.网络编程技术与实例[M].东南大学出版社，2001，77-85..

[15] 叶核亚.Java 程序设计实用教程[M].电子工业出版社，2007.

[16] 王达.计算机网络远程控制[M].北京：清华大学出版社，2003.

[17] 李宝拴.局域网内远程控制计算机系统的设计与实现[J].电脑知识与技术，2010，6(31)：8752-8754.

[18] 蒋宁,单连成,于涧,杨雪华.图像局部更新在计算机远程控制技术中的应用[J].沈阳师范大学学报（自然科学版）.2007，25(1)：57-60.

附　录

1 使用说明书：

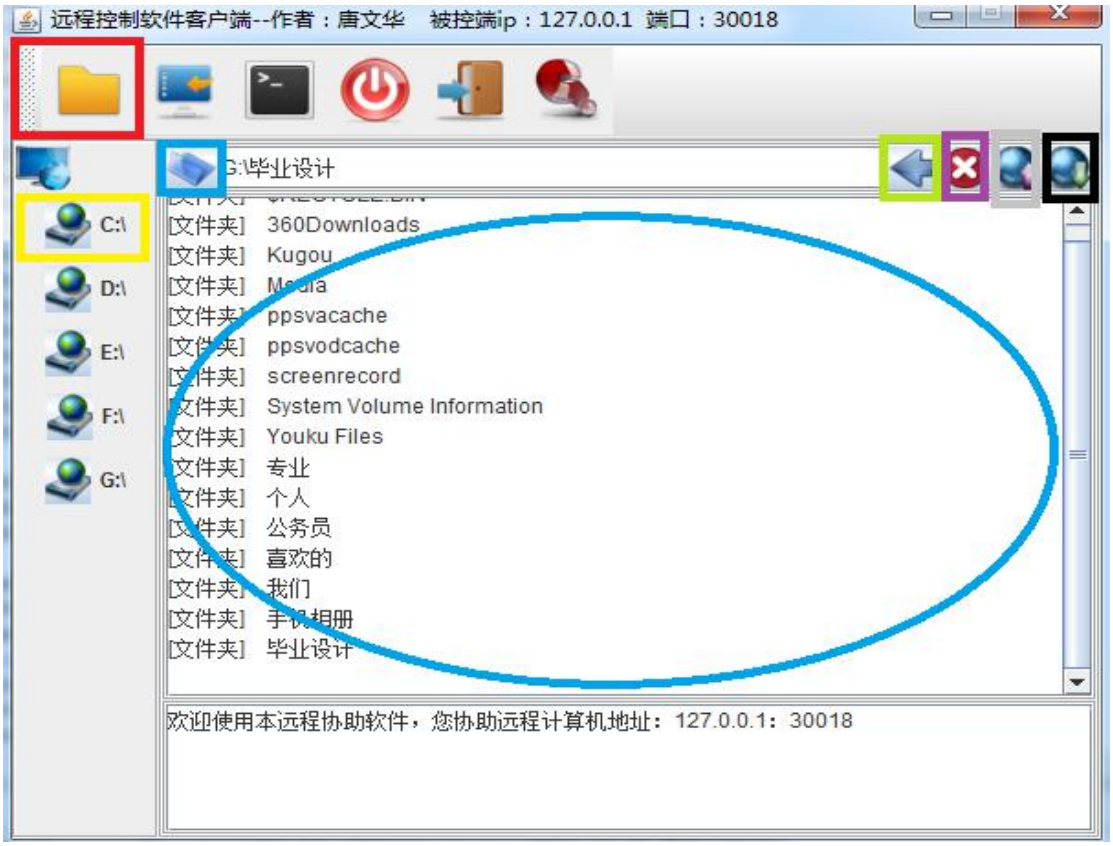
（1）登录

附图-附-1 登陆图

Fig. XXXXXX

首先要启动 Server 端，但后在控制端开启软件出现登录界面，输入被控端 IP 单击【链接】即可；如果被控端使用路由器上网，侧需要在路由器配置端口映射，端口号为：30018、30011 与 30012，具体步奏参考路由器使用手册。

（2）远程资源管理器

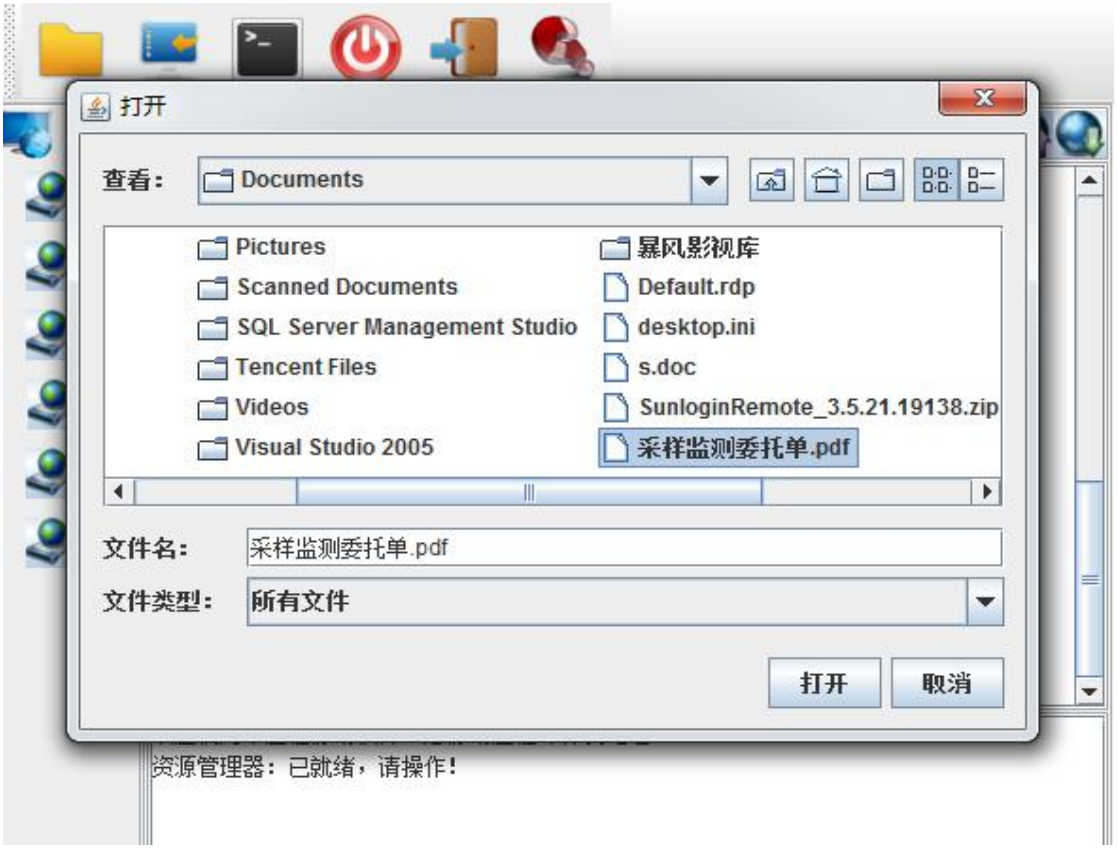


附图-附-2 _客户端

XXXXXXXXX

参见图附 2,红色方框为资源管理器重置图标，单击后能复位资源管理器；黄色方框标注的为远程磁盘盘符，单击后显示该磁盘下的文件盒目录；蓝色方框为查询按钮，能够查询地址栏显示地址的文件或目录；绿色方框为返回上一级菜单按钮；紫色的为删除按钮，删除地址栏标注的文件或目录，不能删除驱动，不能删除错误地址的文件或目录。

灰色的为上传按钮，单击后会会出现文件选窗口，如下图：

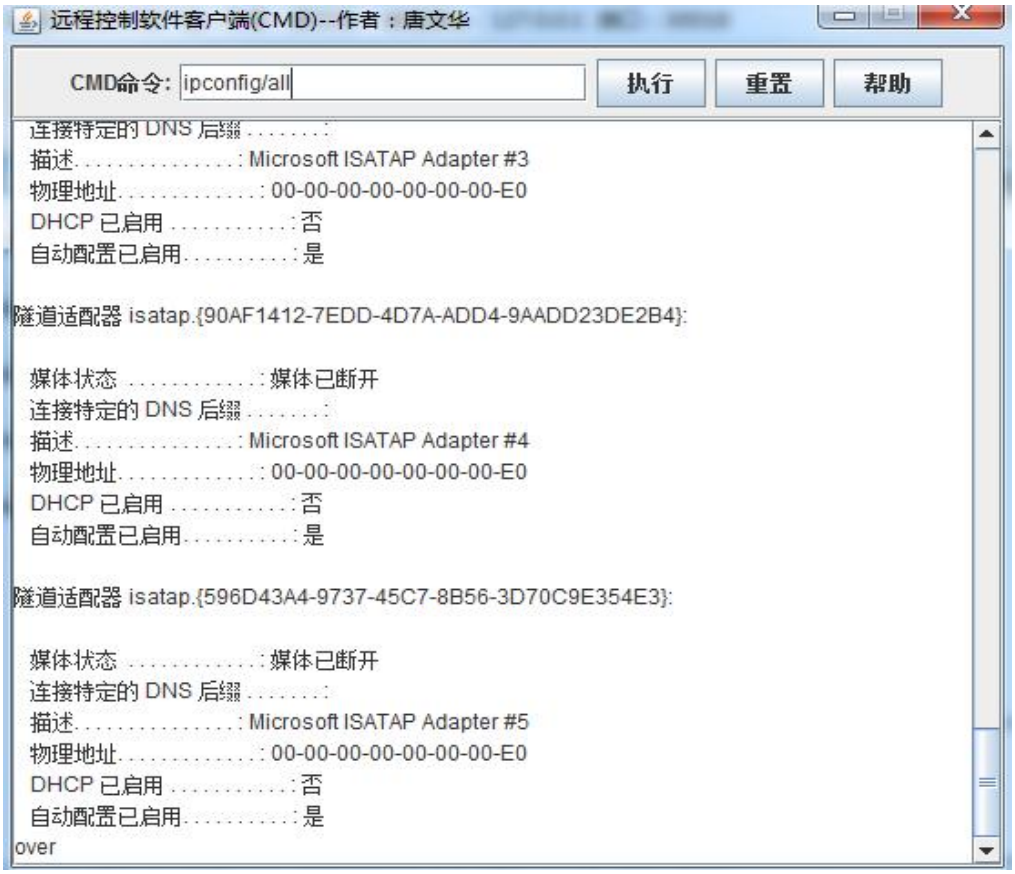


附图–附-3 _下载操作

XXXXXXXXXXXX

选择文件后便会上传到地址栏标注的地址；黑色的为下载按钮，选择文件后单击也会弹出存放地址选择框，单击打开后即下载保存。

（3）远程 CMD 控制台



附图–附4 _CMD 操作

XXXXXXXXXXX

单击 CMD 图标后显示 CMD 操作窗口，在输入框输入 CMD 命令后单击回车或执行，都能在下方的显示卡看见执行结果，单击重置侧清空 CMD 命令输入框，单击帮助侧弹出 CMD 常用命令帮助。

（4）远程监控

单击远程监控图标则开始远程监控，实现鼠标事件以及键盘事件。

（5）退出

单击对出会有两种选择，一是返回登录界面，二是直接退出。

（6）帮助

软件版本说明以及作者联系方式。

2 部分源代码

（1）图像传输压缩过程

```
public void run() {
        while (true) {
            try {
                System.out.println("等待接收截屏信息");
                socket = serverSocket.accept();
```

```
                zip = new ZipOutputStream(new DataOutputStream(
                        socket.getOutputStream()));
                zip.setLevel(9);
                try {
                    img = robot.createScreenCapture(rect);
                    zip.putNextEntry(new ZipEntry("test.jpg"));
                    ImageIO.write(img, "jpg", zip);
                    if (zip != null)
                        zip.close();
                    System.out.println("被控端：connect");
                } catch (IOException ioe) {
                    System.out.println("被控端：disconnect");
                }
            } catch (IOException ioe) {
                System.out.println("错误1");
            } finally {
                if (socket != null) {
                    try {
                        socket.close();
                    } catch (IOException e) {
                    }
                }
            }
        }
    }
```

（2）CMD 实现方法

```
public void run() {
        Process pro = null;
        try {
            pro = Runtime.getRuntime().exec("cmd /c " + this.msg);
        } catch (Exception e) {
            e.printStackTrace();
        }
        InputStreamReader ipsr = new InputStreamReader(pro.getInputStream());
        BufferedReader br = new BufferedReader(ipsr);
```

```
String cmdResult = "";
String msgs = null;
try {
    while ((msgs = br.readLine()) != null)
        cmdResult = cmdResult + msgs + "\n";
} catch (Exception e) {
    e.printStackTrace();
}
String sendmsg = cmdResult + "over";
try {

    os = socket.getOutputStream();
     dos = new DataOutputStream(os);
    dos.writeUTF(sendmsg);
    dos.flush();
    dos.close();
    os.close();
} catch (UnknownHostException e) {
    System.out.println("文件未找到！");
} catch (ConnectException e) {
    System.out.println("链接超时！");
} catch (IOException e) {
    System.out.println("不可打开文件！");
} finally {
    try {
        if (this.dos != null)
            this.dos.close();
        if (this.os != null)
            this.os.close();
        if (this.socket != null)
            this.socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
    }
```

（3）实现鼠标键盘事件方法

```
public void run() {
        while (true) {
            try {
                Socket socket = serverSocket.accept();
                InputStream is = socket.getInputStream();
                int r;
                String info = "";
                while ((r = is.read()) != -1) {
                    info += "" + (char) r;
                }
                //System.out.println(info);
                is.close();
                if (info != null) {
                    String s[] = info.trim().split(",");
                    if ("mouseClicked".equals(s[0].trim())) {
                        int type = Integer.parseInt(s[s.length - 1].trim());
                        if (s.length == 4) {
                            int x = Integer.parseInt(s[1].trim());
                            int y = Integer.parseInt(s[2].trim());
                            robot.mouseMove(x, y);
                            robot.mousePress(type);
                            robot.mouseRelease(type);
                            System.out.println("mouseClicked:MOUSE move to " + x
                                    + "," + y + " AND execute TYPE IS click "
                                    + type);
                        }
                    } else if ("mousePressed".equals(s[0].trim())) {
                        int type = Integer.parseInt(s[s.length - 1].trim());
                        if (s.length == 4) {
                            int x = Integer.parseInt(s[1].trim());
                            int y = Integer.parseInt(s[2].trim());
                            robot.mouseMove(x, y);
                            robot.mousePress(type);
```

```
                System.out.println("mousePressed:MOUSE move to " + x
                        + "," + y + " AND execute TYPE IS press "
                        + type);
            }
        } else if ("mouseReleased".equals(s[0].trim())) {
            int type = Integer.parseInt(s[s.length - 1].trim());
            if (s.length == 4) {
                int x = Integer.parseInt(s[1].trim());
                int y = Integer.parseInt(s[2].trim());
                robot.mouseMove(x, y);
                robot.mouseRelease(type);
                System.out.println("mouseReleased:MOUSE move to " + x
                        + "," + y
                        + " AND execute TYPE IS release    " + type);
            }
        } else if ("mouseDragged".equals(s[0].trim())) {
            if (s.length == 4) {
                int x = Integer.parseInt(s[1].trim());
                int y = Integer.parseInt(s[2].trim());
                robot.mouseMove(x, y);
                System.out.println("mouseDragged:MOUSE move to " + x
                        + "," + y);
            }
        } else if ("mouseMoved".equals(s[0].trim())) {
            if (s.length == 3) {
                int x = Integer.parseInt(s[1].trim());
                int y = Integer.parseInt(s[2].trim());
                robot.mouseMove(x, y);
                System.out.println("mouseMoved:MOUSE move to " + x
                        + "," + y);
            }
        } else if ("keyPress".equals(s[0].trim())) {
            if (s.length == 2) {
                int keycode = Integer.parseInt(s[1]);
                robot.keyPress(keycode);
```

```
                    }
                } else if ("keyRelease".equals(s[0].trim())) {
                    if (s.length == 2) {
                        int keycode = Integer.parseInt(s[1]);
                        robot.keyRelease(keycode);
                    }
                } else if ("keyTyped".equals(s[0].trim())) {
                    if (s.length == 2) {
                        int keycode = Integer.parseInt(s[1]);
                        robot.keyPress(keycode);
                        robot.keyRelease(keycode);
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```