

**实验6**

**FTP 工作原理研究与分析**

**班 级： 20级计合2班**

**学 生 姓 名 学 号： 孙嘉祎 202000800154**

**黄欣灵202000800082**

**李静 202000800136**

**崔家源 202000800127**

**2022年12月15日**

目 录

[**1.实验目的** 3](#_Toc122082016)

[**2.实验器材** 3](#_Toc122082017)

[**3.实验步骤** 3](#_Toc122082018)

[3.1建立客户端和服务器 3](#_Toc122082019)

[3.1.1创建虚拟机 3](#_Toc122082020)

[3.1.2配置服务器 4](#_Toc122082021)

[3.1.3配置客户机 8](#_Toc122082022)

[3.2设置连接 8](#_Toc122082023)

[3.2.1被动模式连接参数 8](#_Toc122082024)

[3.2.2主动模式连接参数 10](#_Toc122082025)

[3.3被动模式 10](#_Toc122082026)

[3.4主动模式 14](#_Toc122082027)

[3.5（补充）主动工作模式下的上传数据 16](#_Toc122082028)

[**4.试验总结** 18](#_Toc122082029)

[**5.非预期现象及原因探讨** 19](#_Toc122082030)

[5.1描述 19](#_Toc122082031)

[5.2原因猜测 19](#_Toc122082032)

[**6.成员分工&致谢** 24](#_Toc122082033)

**1.实验目的**

**注意：本次实验会观察到一些奇怪现象，稍后我会推测原因并给出解释**

理解 FTP 的工作原理，掌握通过工具分析 FTP 工作原理的方法。

（1） 掌握 FTP 服务器和客户端软件的使用；

（2） 研究生成并捕获 FTP 相应网络数据的方法；

（3） 通过工具分析捕获数据，研究 FTP 工作原理；

（4） 通过网络数据捕获软件分析 FTP 工作过程，研究分析 FTP 主动和被动工作模式

的区别；

（5） 撰写研究报告。

**2.实验器材**

硬件：具有硬件虚拟化功能的PC兼容机一台

平台：Windows10专业工作站版2004（19041.1237）

软件：Vmware Workstation 16，filezilla FTP，wireshark网络审计实用程序

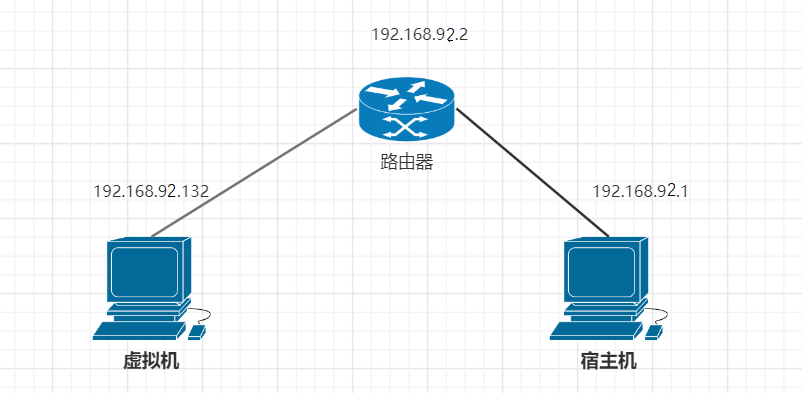
**3.实验步骤**

3.1建立客户端和服务器

3.1.1创建虚拟机

由于硬件只有一份，所以需要通过虚拟机模拟客户机

本试验的虚拟机与宿主机采用NAPT方式进行网络连接，并在虚拟内网上建立虚拟路由器。网络拓扑图如图所示（路由器是虚拟的，并非实际硬件）



本试验用物理机作为服务器，虚拟机作为客户机

3.1.2配置服务器

Filezilla唯一官方下载地址，其他的任何网址都不是，请擦亮眼睛认准！

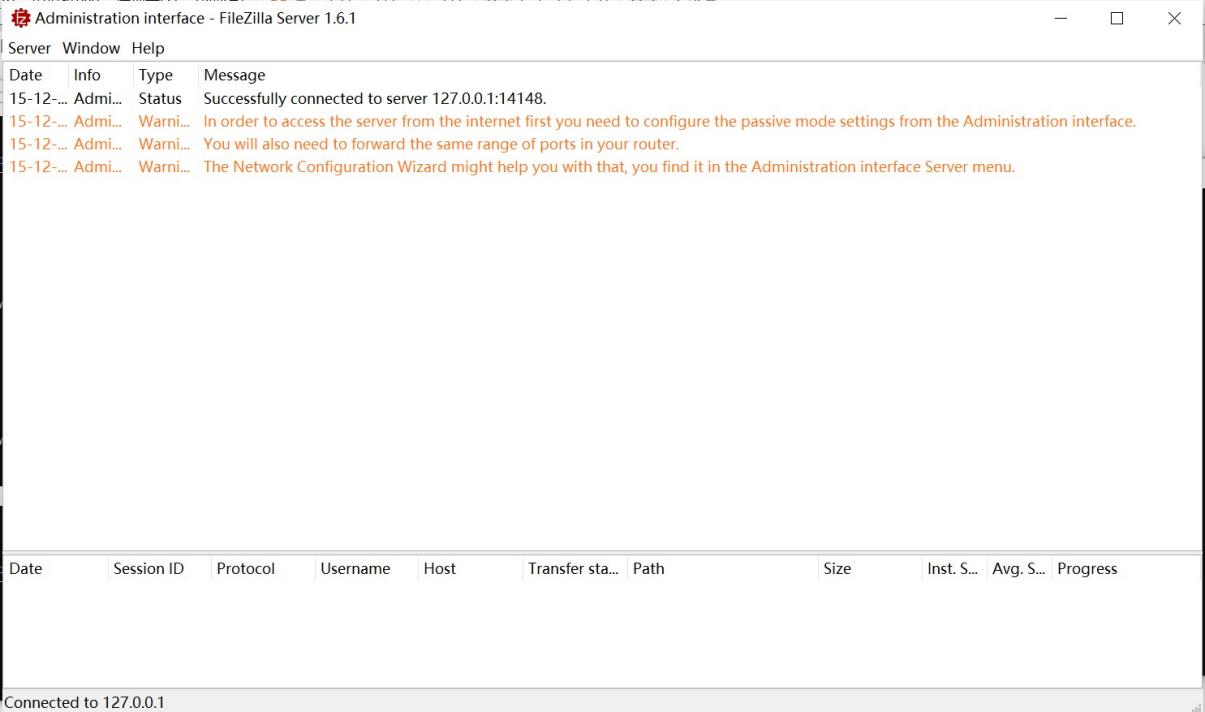
客户机：[Download FileZilla Client for Windows (64bit x86) (filezilla-project.org)](https://filezilla-project.org/download.php?type=client)

服务器：[Download FileZilla Server for Windows (64bit x86) (filezilla-project.org)](https://filezilla-project.org/download.php?type=server)

我不是打广告，因下载地址错误导致误用虚假、诈骗、挂马、后门软件，后果自负！

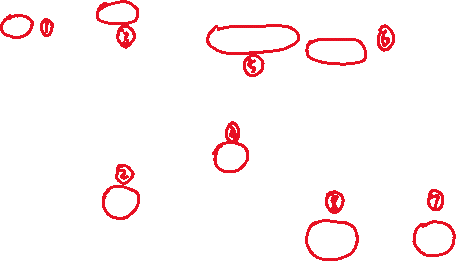
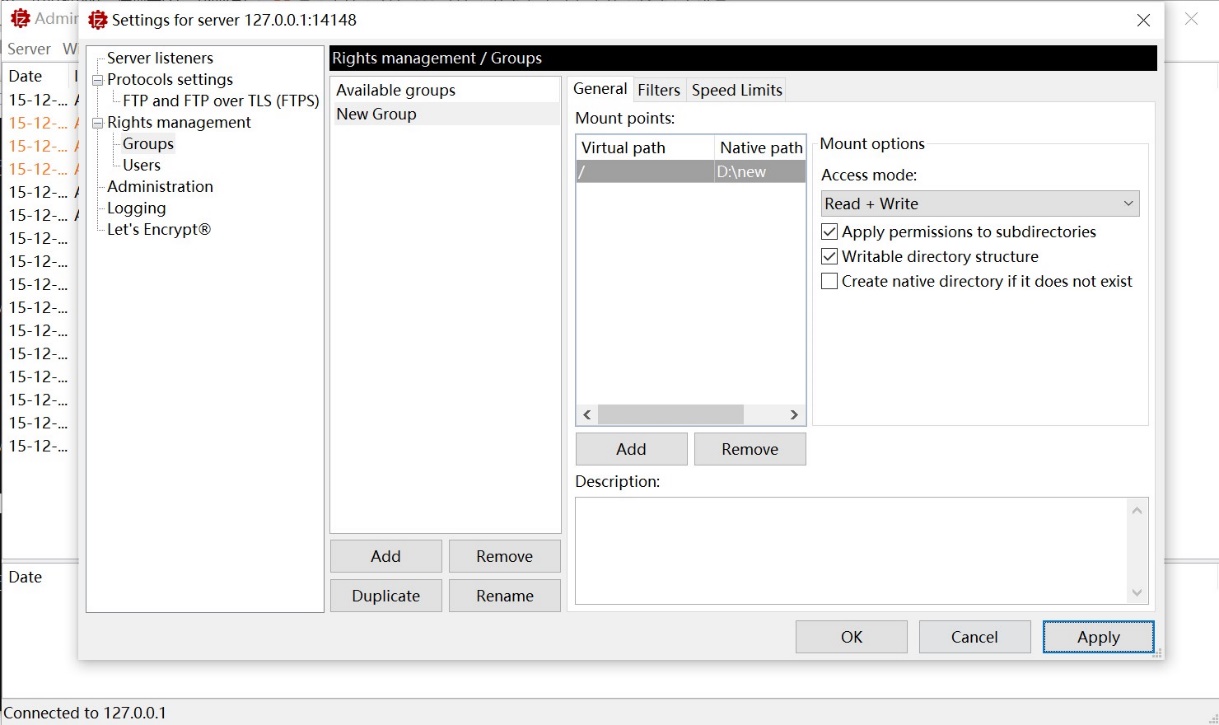
物理机安装FTP的服务程序。安装过程全部使用默认参数（无论他问你什么都是下一步），故不在赘述

安装完打开，一路下一部：

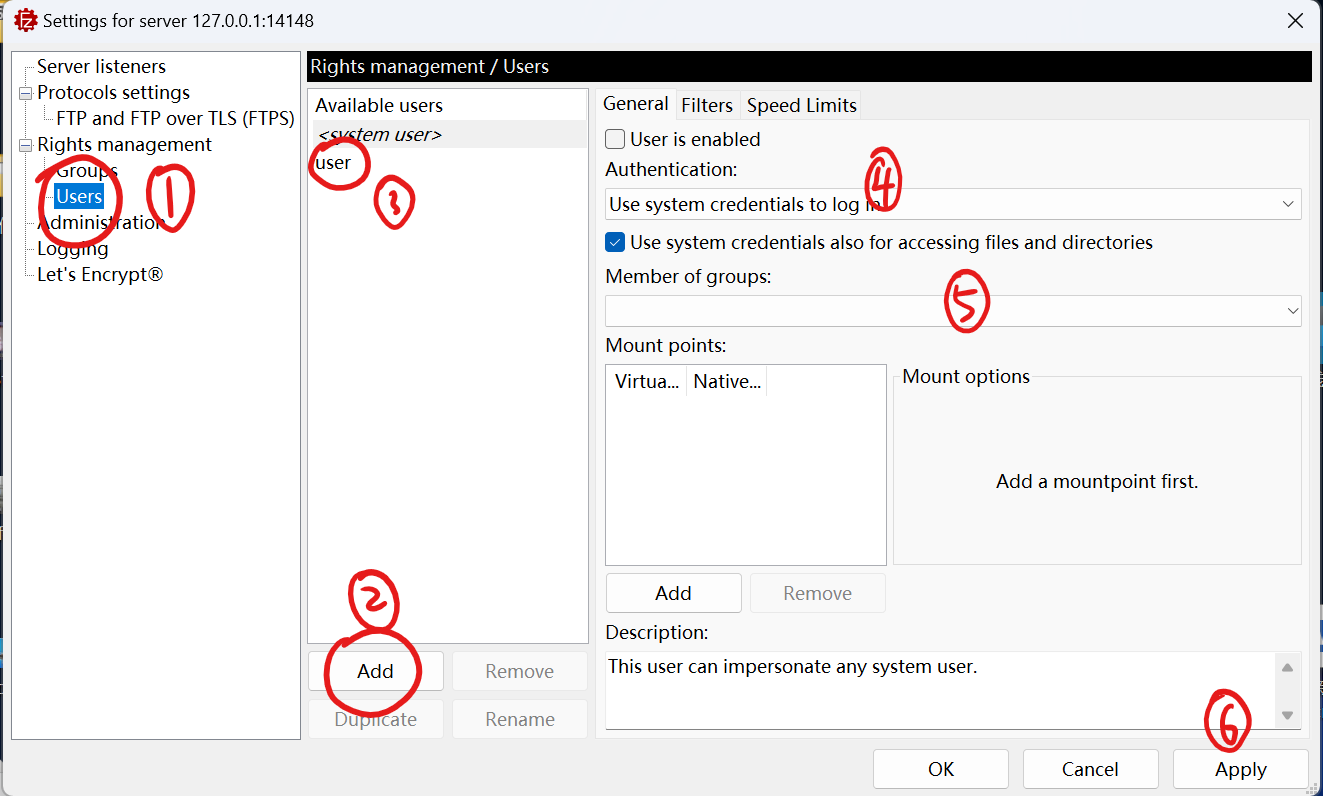


在服务器端建立FTP使用的文件夹，以D:\new为例（建议不要放C盘，否则需要处理UAC问题对小白不友好，会处理的话当我没说）

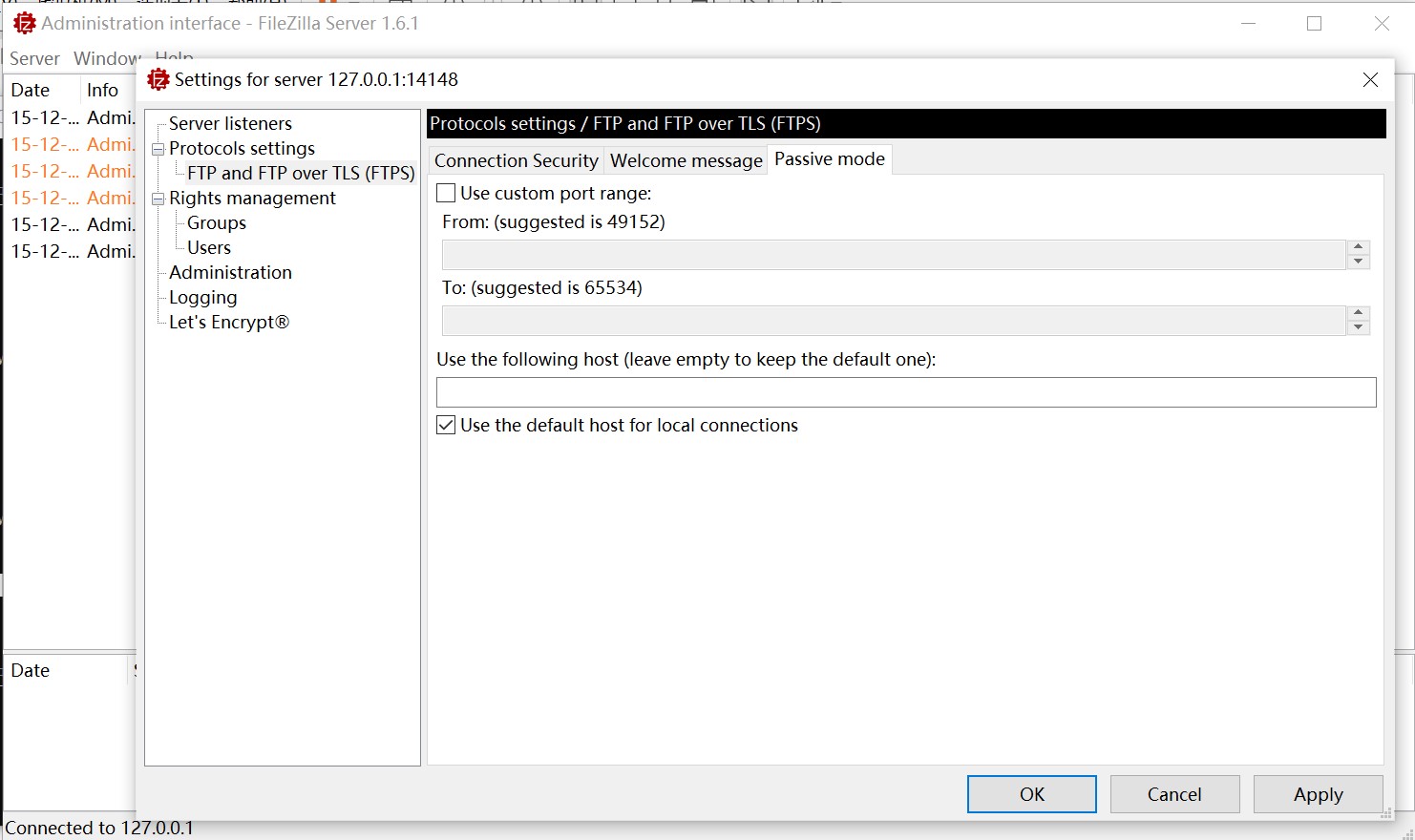
点击Server-config server，选择groups节点，add，名称随意，virtualpath写/（一个字都不能错），nativepath写D:\new， accessmode选择可读写，apply即可



点击user节点，add名称随意，authentication选use password，密码随意（请你记住否则后果自负），group选择刚才的，apply即可（忘记截图了，图片来源于网络，侵删）



FTP连接时使用的端口号范围，保持默认，记住这个端口号





打开计算机的control panel，导航到control panel——system and security——windows defender firewall：（当时装的英文版镜像也懒得的打中文语言包了，凑合着看吧）

Graphical user interface, text, application, email

Description automatically generated

点击allow an app or feature throw firewall：然后依次点击change settings，allow another app，browse（弹UAC请允许），找到安装的filezillaserver.exe（认准！）确定，然后把多出来的private和public全选，确定即可

Graphical user interface, application

Description automatically generated



导航到Control Panel\System and Security\Windows Defender Firewall，点击advanced settings，导航到inbound rules，把filezilla开头的条目全部enable

点击new rule：类型选port，下一部，端口范围写你刚才记住的，然后一直下一步，起名随便

Graphical user interface, text, application, email

Description automatically generated

3.1.3配置客户机

切换到虚拟机安装客户端，默认设置一路下一步即可，不再赘述，也不截图了

3.2设置连接

做两组试验，被动模式和主动模式，预先配置好连接参数：

3.2.1被动模式连接参数

打开客户机，file——site manager——new site

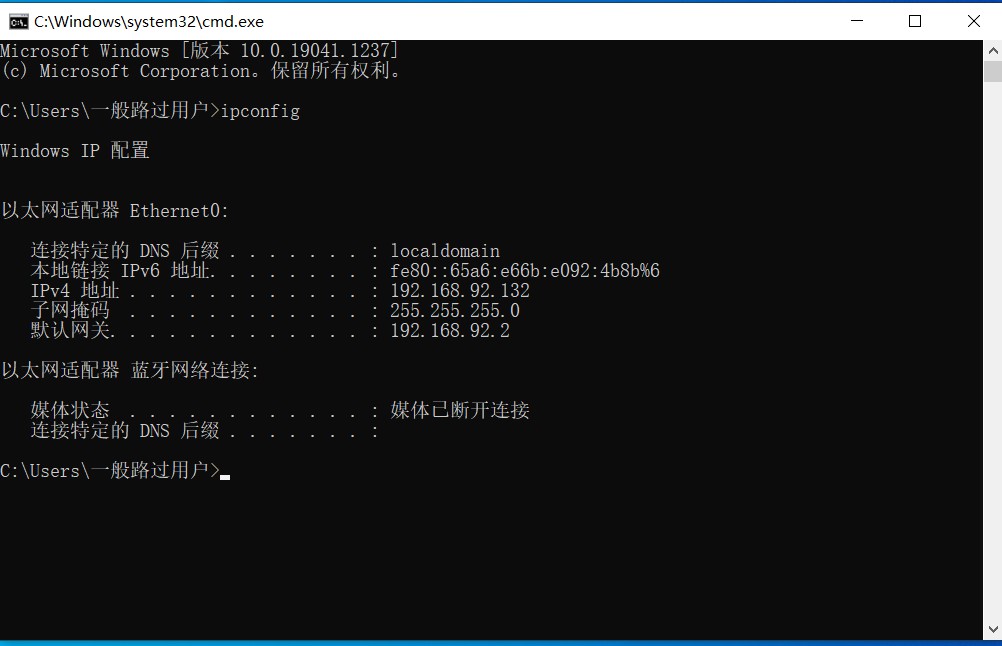
名称随意，用户名密码填刚刚设置的，IP地址填主机的（参考一开始的网络拓扑图）端口填21，加密方式选明文传输，transfer settings选项卡模式选passive

Graphical user interface, application

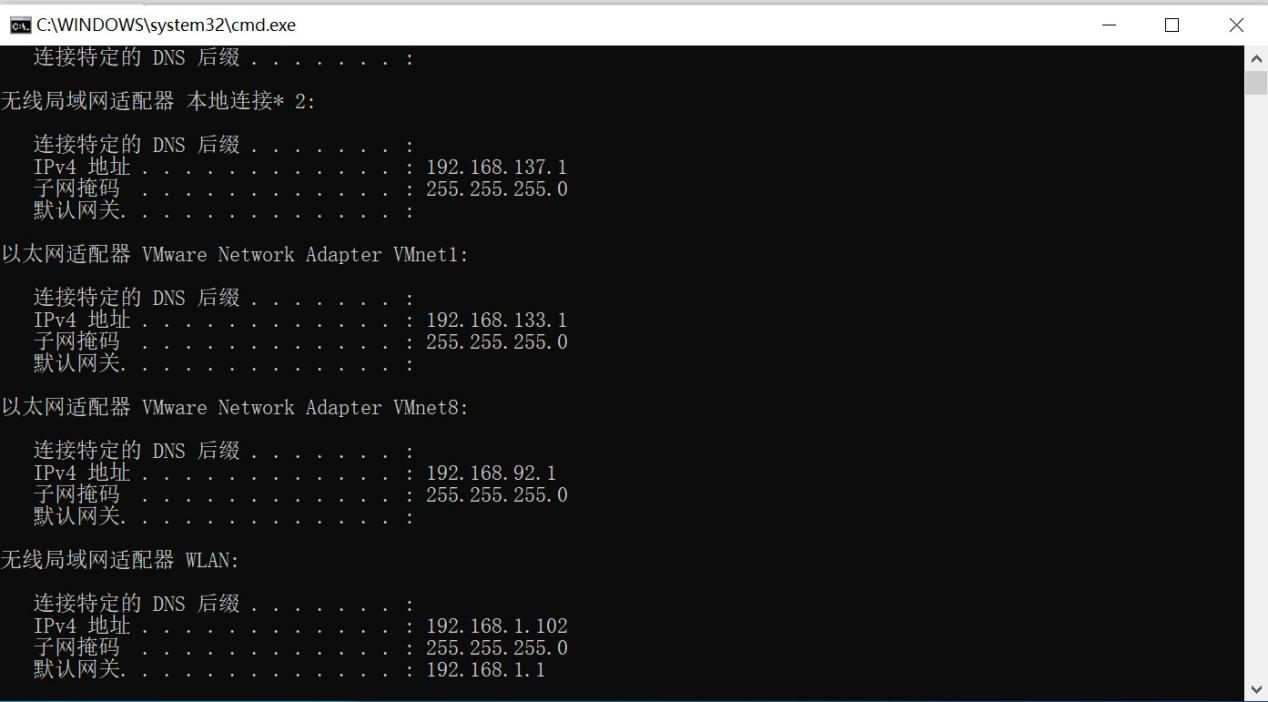
Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

客户端的IP地址（请根据实际情况自行修改）



服务器的IP地址

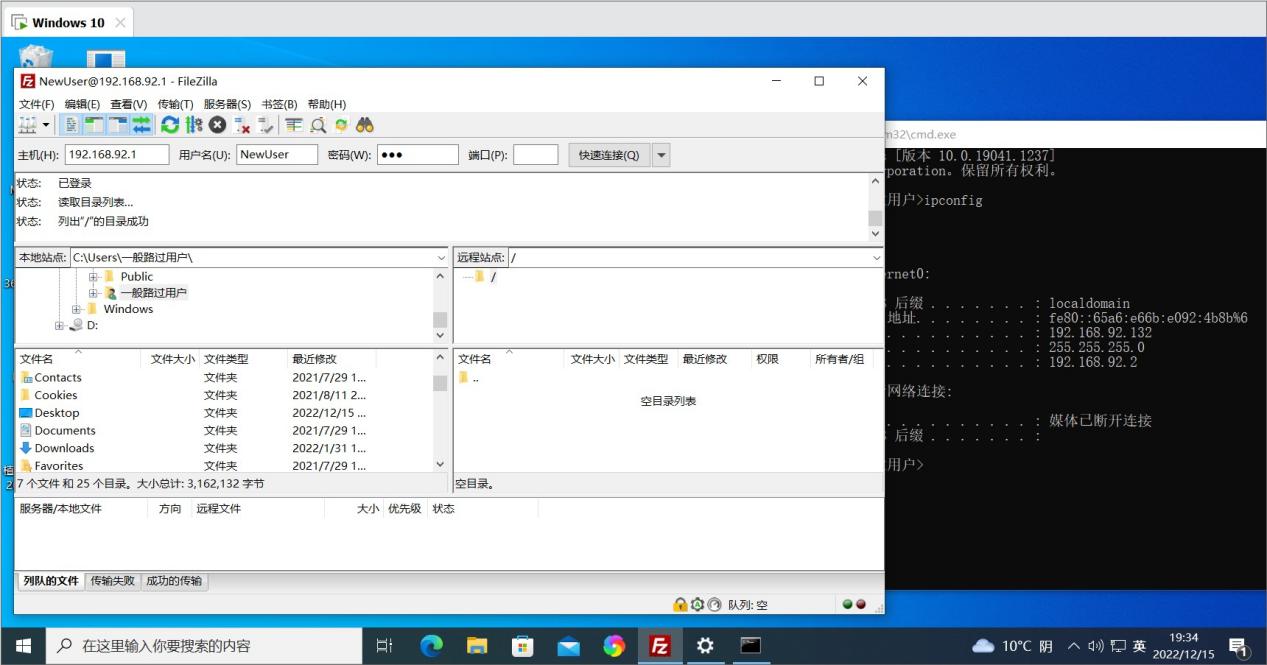


3.2.2主动模式连接参数

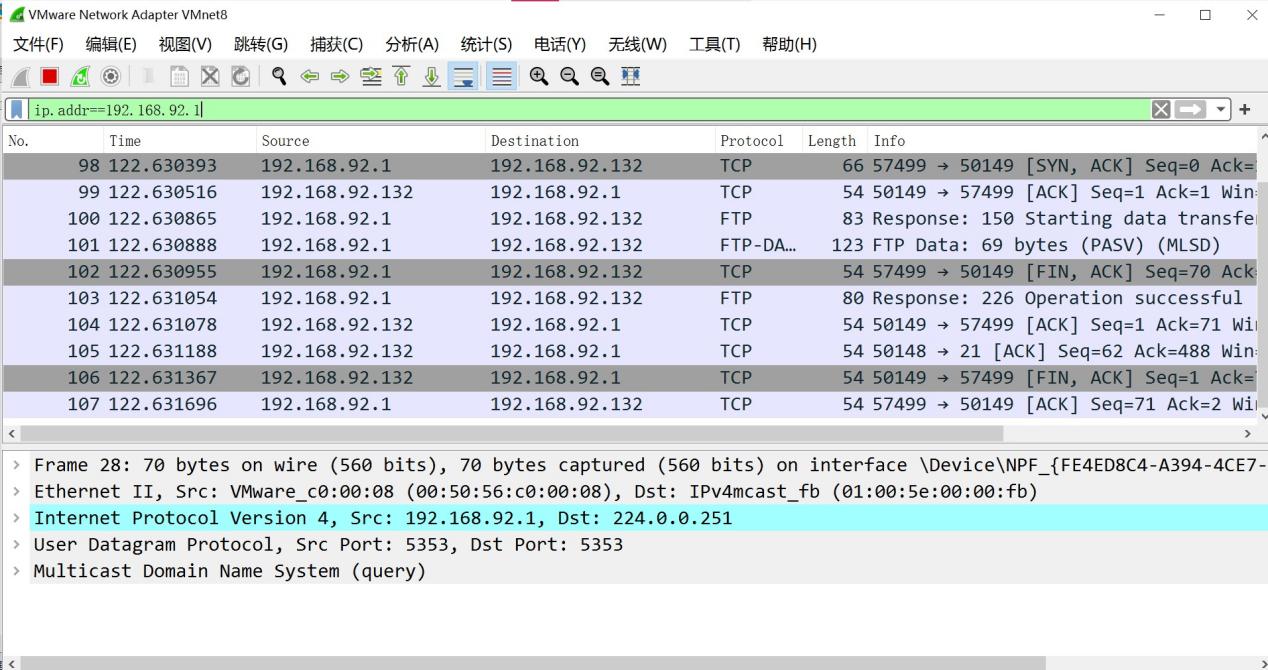
同上，只是transfer settings里面填active

3.3被动模式

选好连接参数，客户端成功连接上服务器



连接时产生的数据报

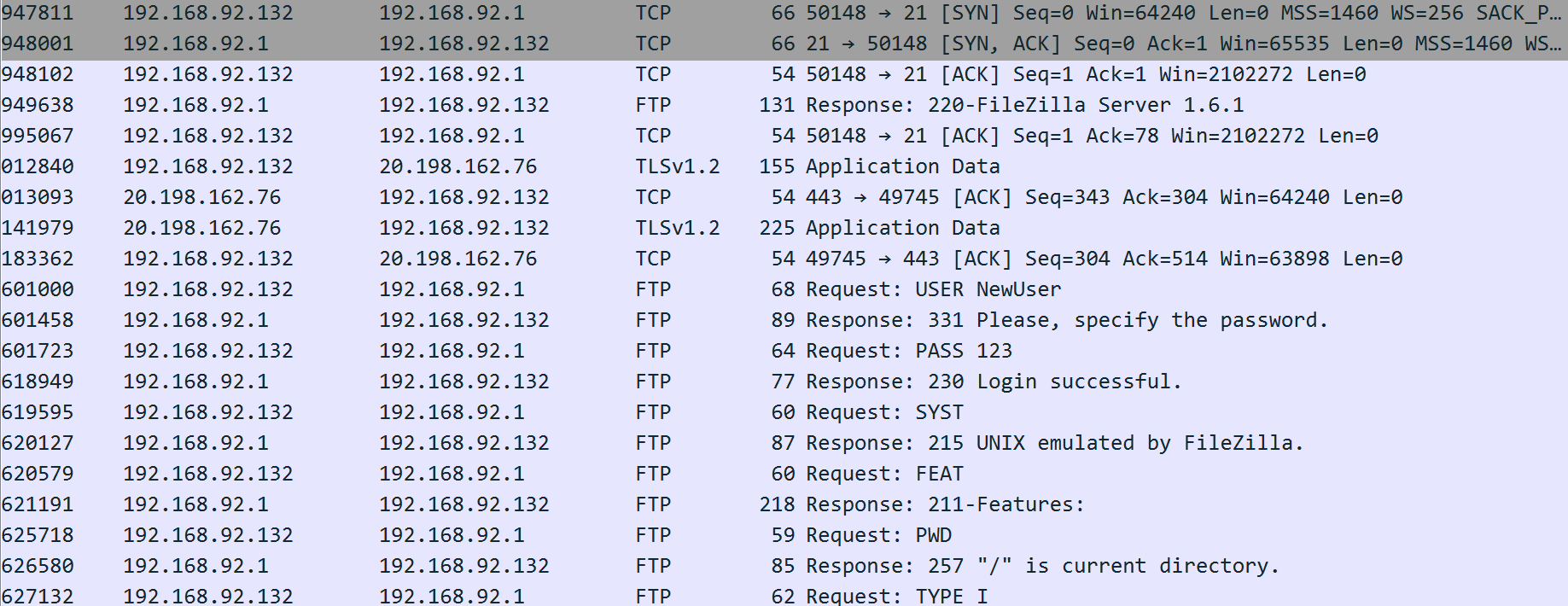
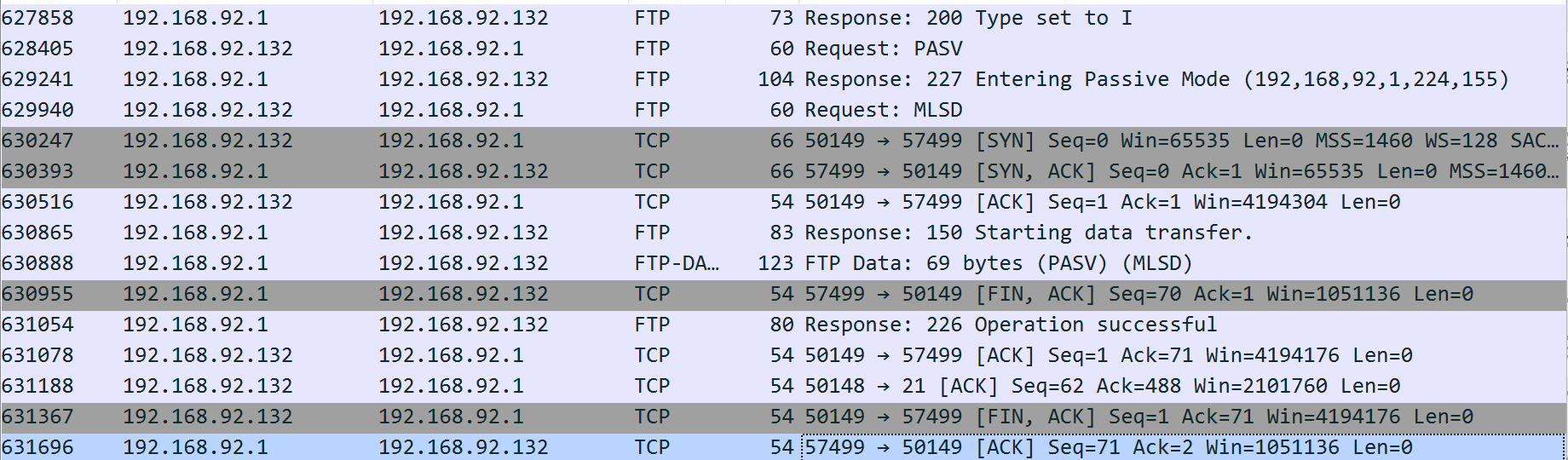


分析这张图我们可以看到，FTP的控制线程（这里尊重事实，称其为线程，而不是课本上说的进程）服务器端口21，客户机端口50418。

接下来尝试下载文件，我们在服务器放一个文本文件，内容是“it is a happy test”，到客户机完成下载，同时观察审计内容

**注意：这里观察到了两组FTP连接：查询文件时建立一次，真正传输数据又建立了一次。这是因为filezilla本身的设计：长时间不操作会断开FTP，再次操作会重新建立（可以设置保持连接）**

第一组的时序图：



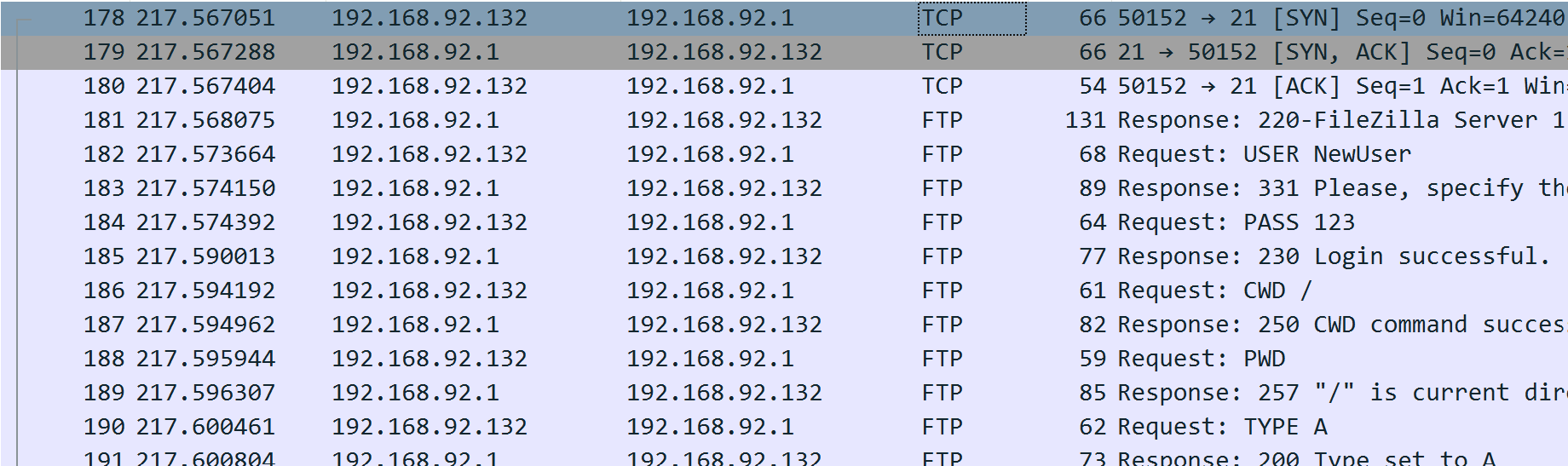
第一组的有效PDU，可以看出来只是查询文件信息

Graphical user interface

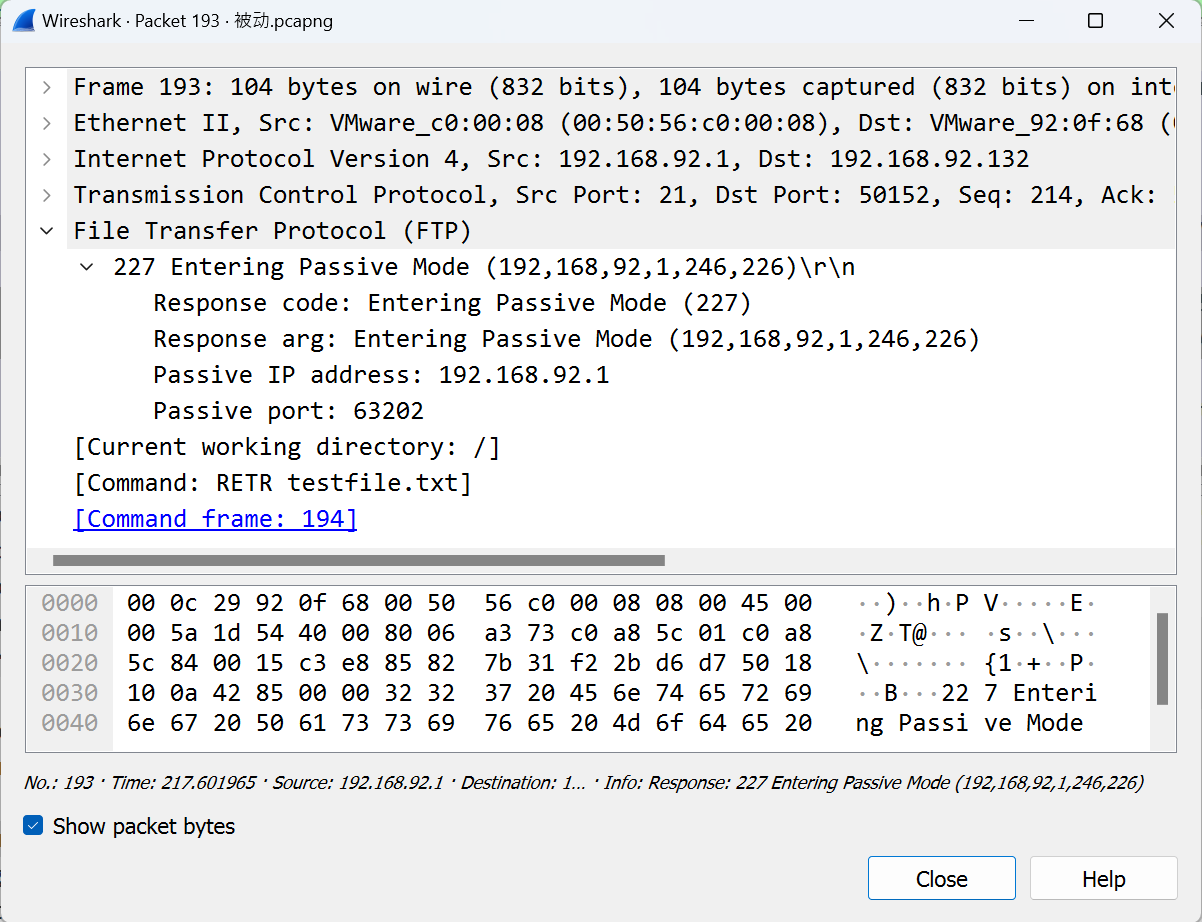
Description automatically generated with low confidence

接下来进入正式的数据传输

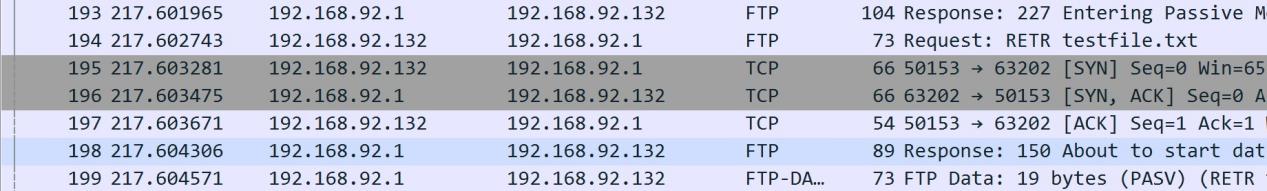
下图为控制连接，从端口号和握手报文可以看出客户机端口号50152连接到服务器端口21



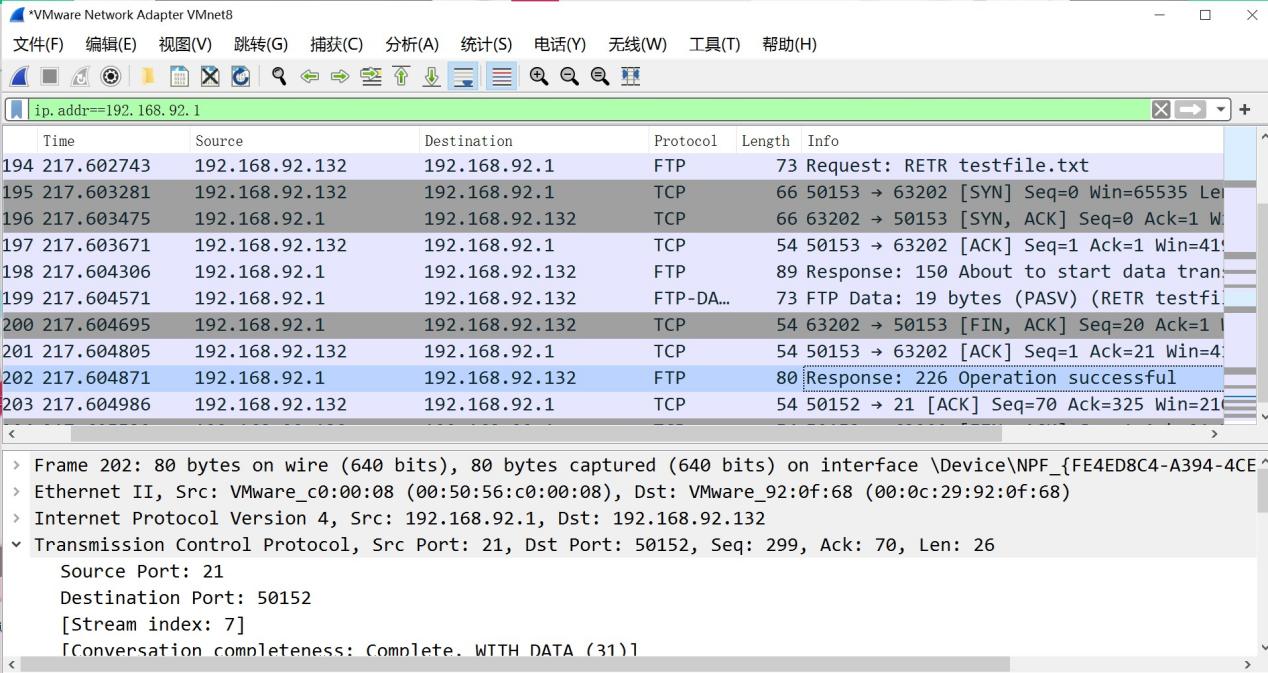
服务器告诉客户机数据连接将要使用的端口：



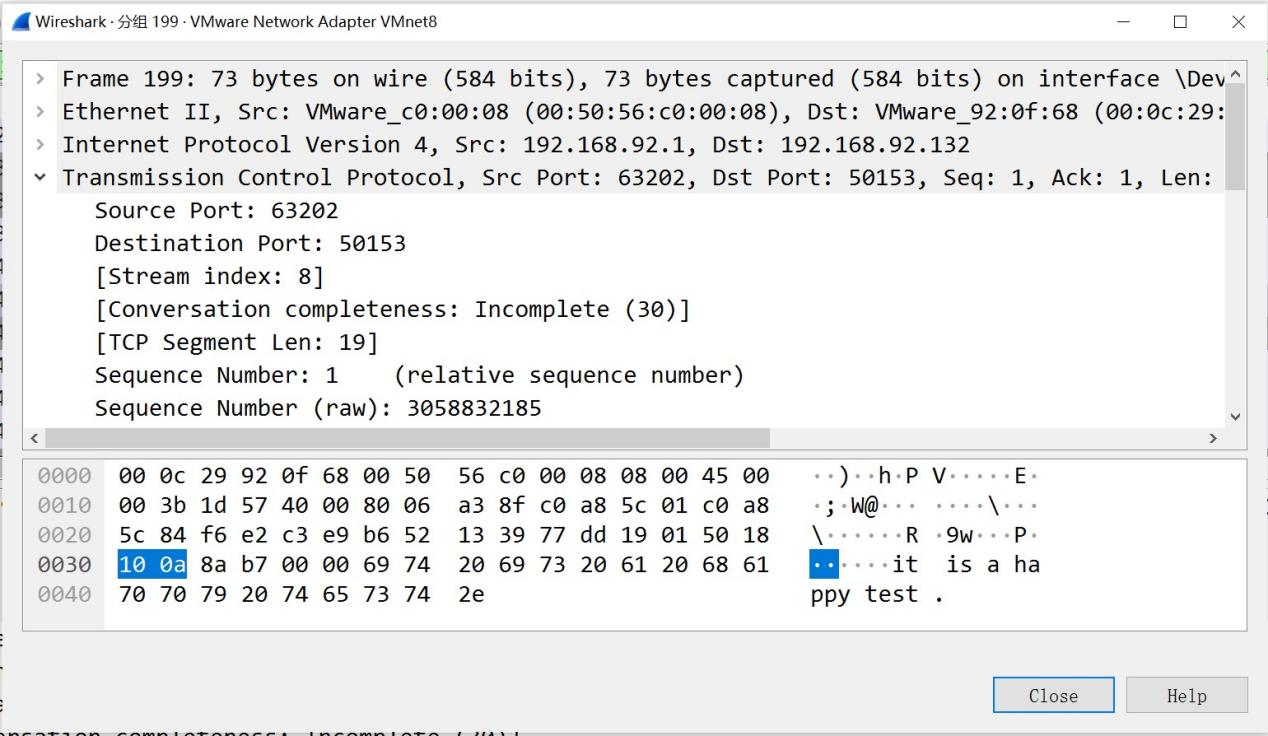
下图是数据链接的时序图，服务器端口63202与客户机端口50153连接进行数据传输。注意观察握手报文：



传输完成后关闭数据连接的挥手过程：



传输FTP服务器文件夹中下载的文件内容的数据报在传输层的信息，可见内容就是我们的文件



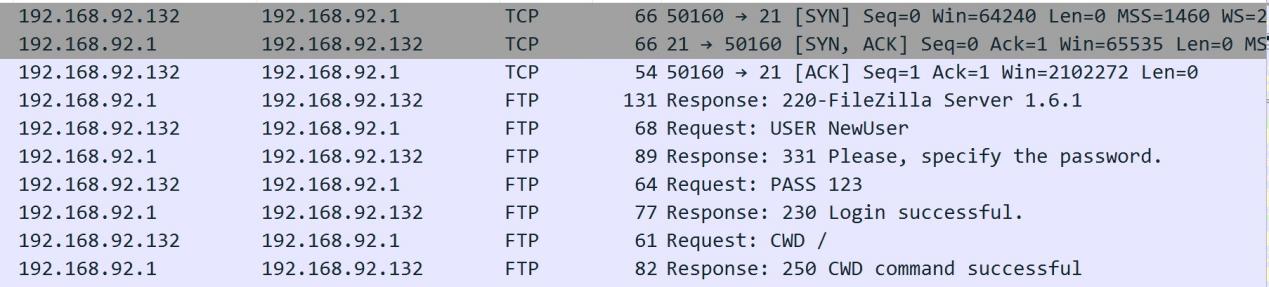
由此总结本次被动模式FTP的传输过程：

1. 服务器监听21端口，客户机使用随机端口（这里是50152）连接，这条路作为控制连接
2. 传输文件前，服务器用enter passive mode告诉客户端使用什么端口（这里是63202）
3. 客户端使用N+1端口（这里是50153）连接被告知的端口（这里是63202），完成数据传输后关闭连接

3.4主动模式

切换到主动模式的连接参数，连接服务器。

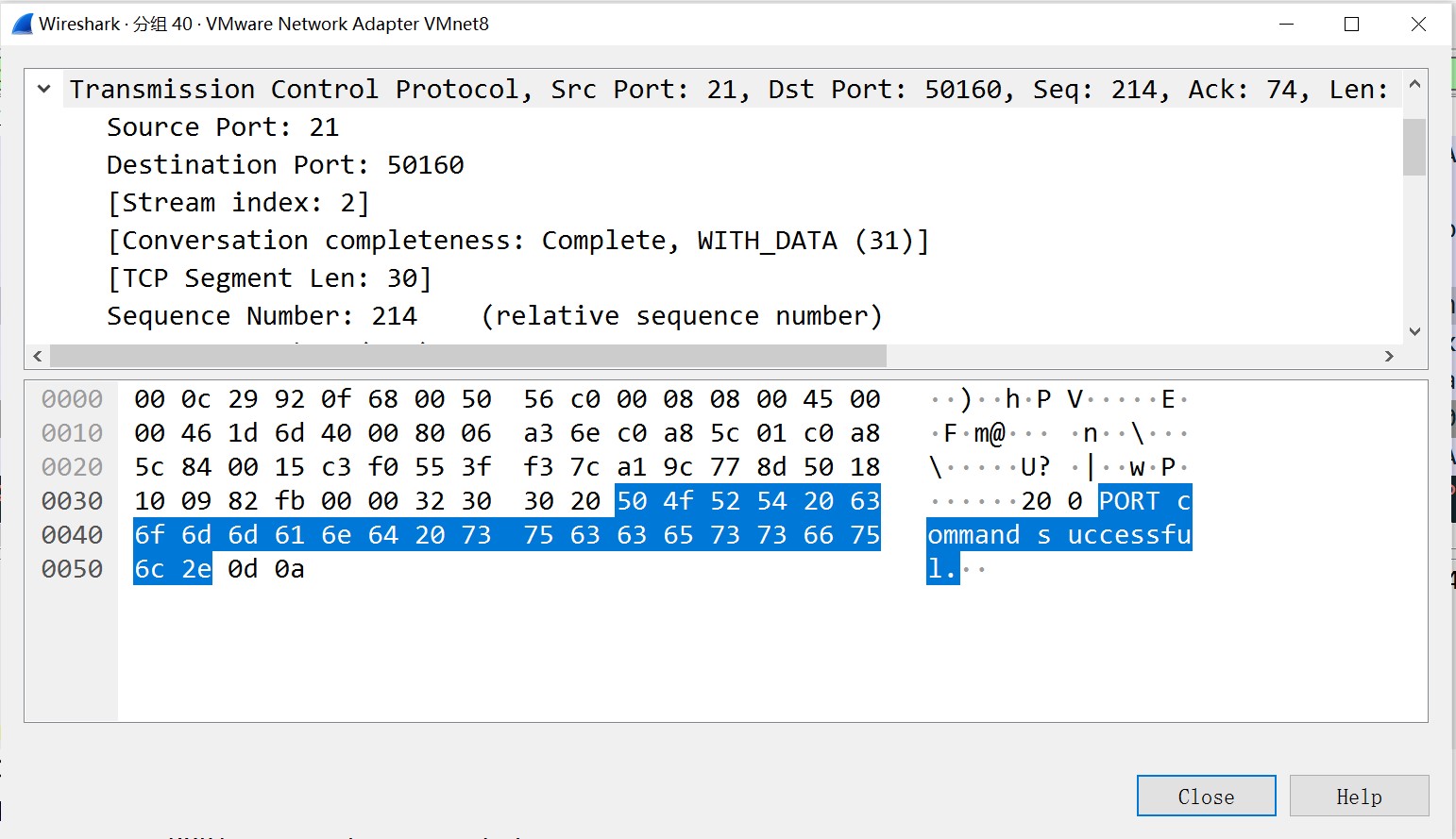
下图是连接的时序图和握手报文，控制连接 客户机端口50160与服务器端口21连接



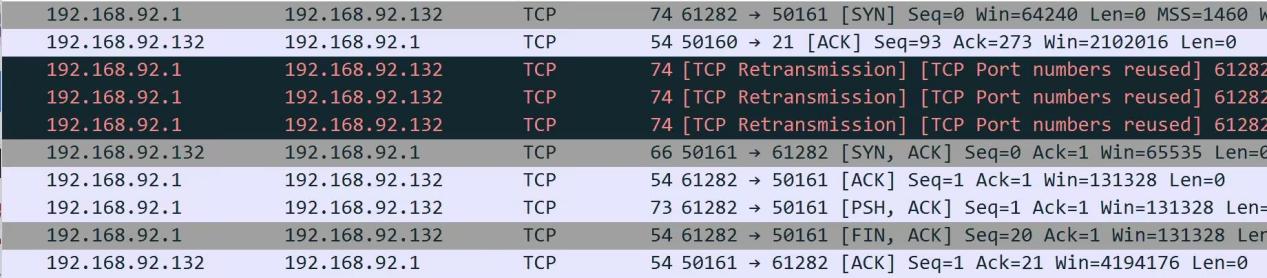
客户端使用PORT命令通过控制连接把数据端口号50161发送给服务器



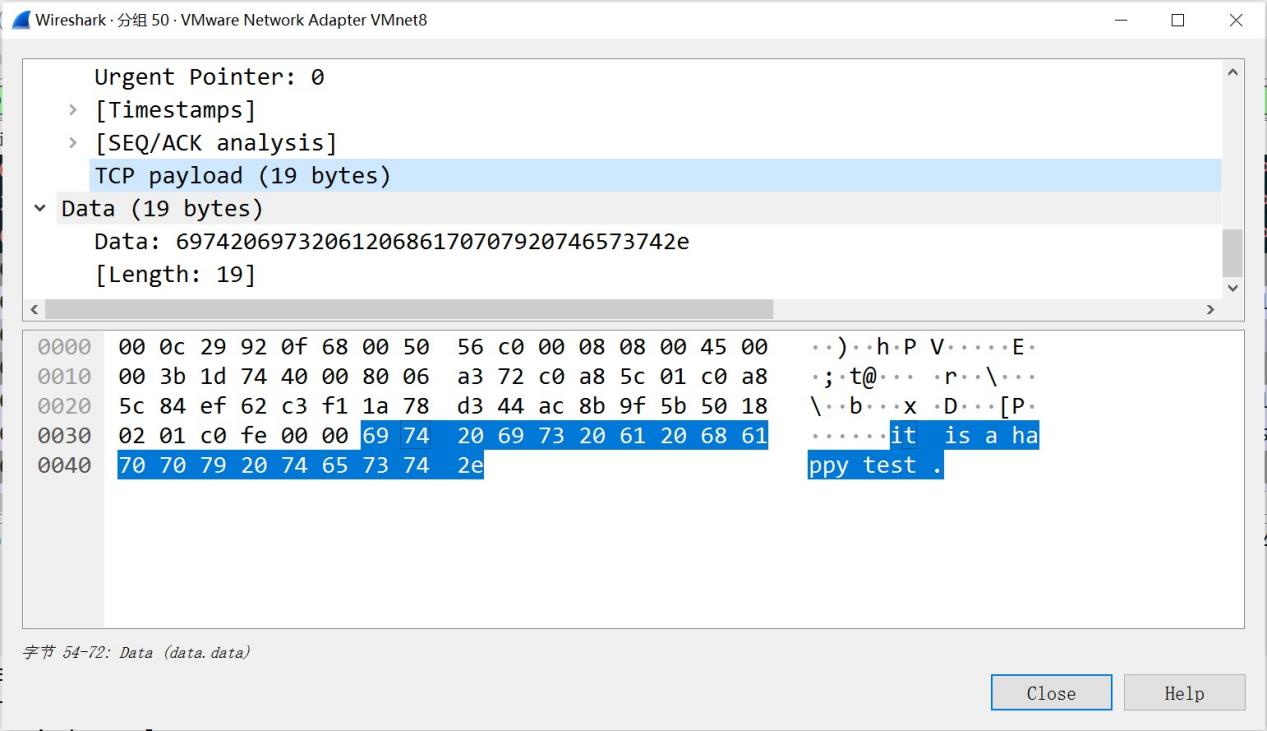
服务器确认：port命令发送成功



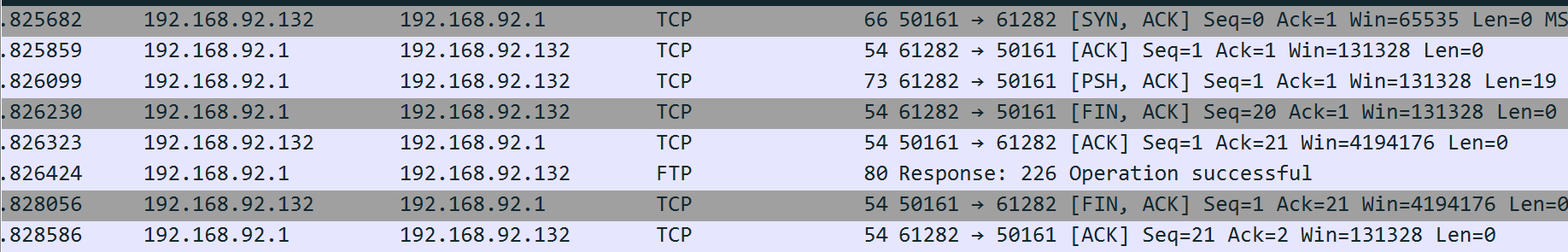
建立数据传输连接的时序图：注意观察握手报文，此时服务器的61282连接到客户机的50161（为什么服务器不用20，后面我会解释）



服务器端口与客户机端口50161连接进行数据传输，内容是文件内容。在主动模式下数据传输走的是纯TCP报文，而不显示为FTP，这是程序本身的设计



操作结束，服务器从21端口发送给客户机50160端口告知传输完成，关闭数据连接， 注意观察挥手报文：

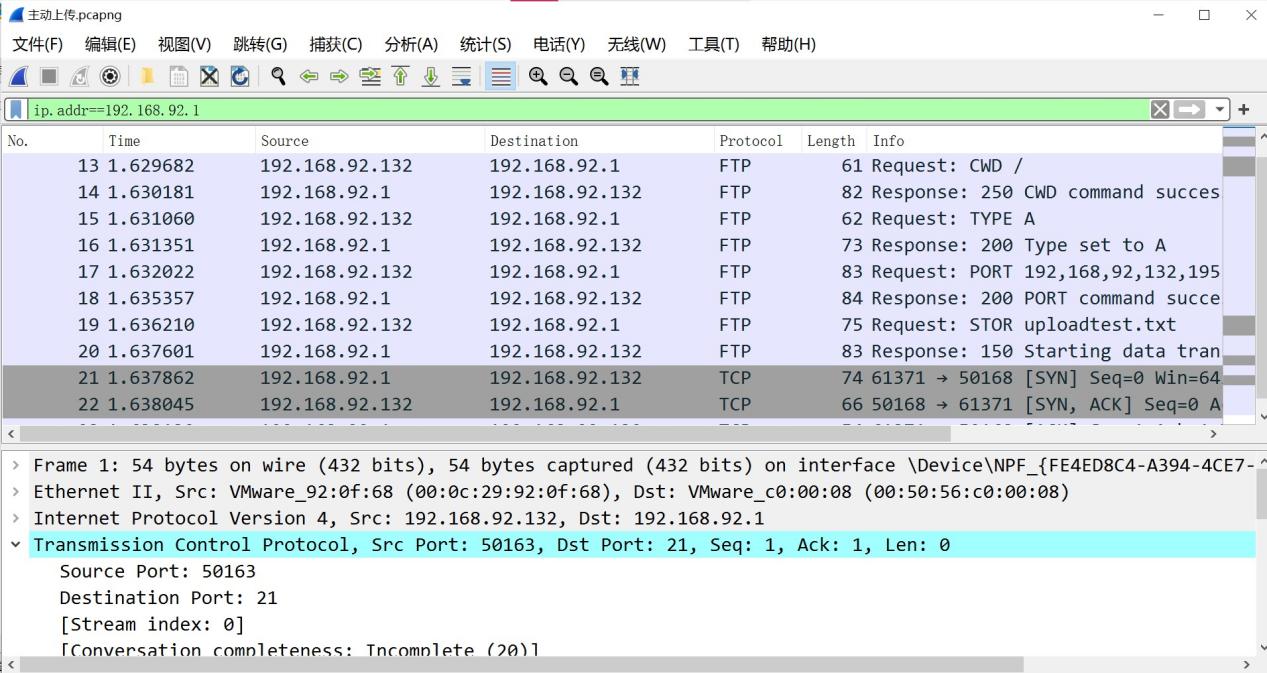


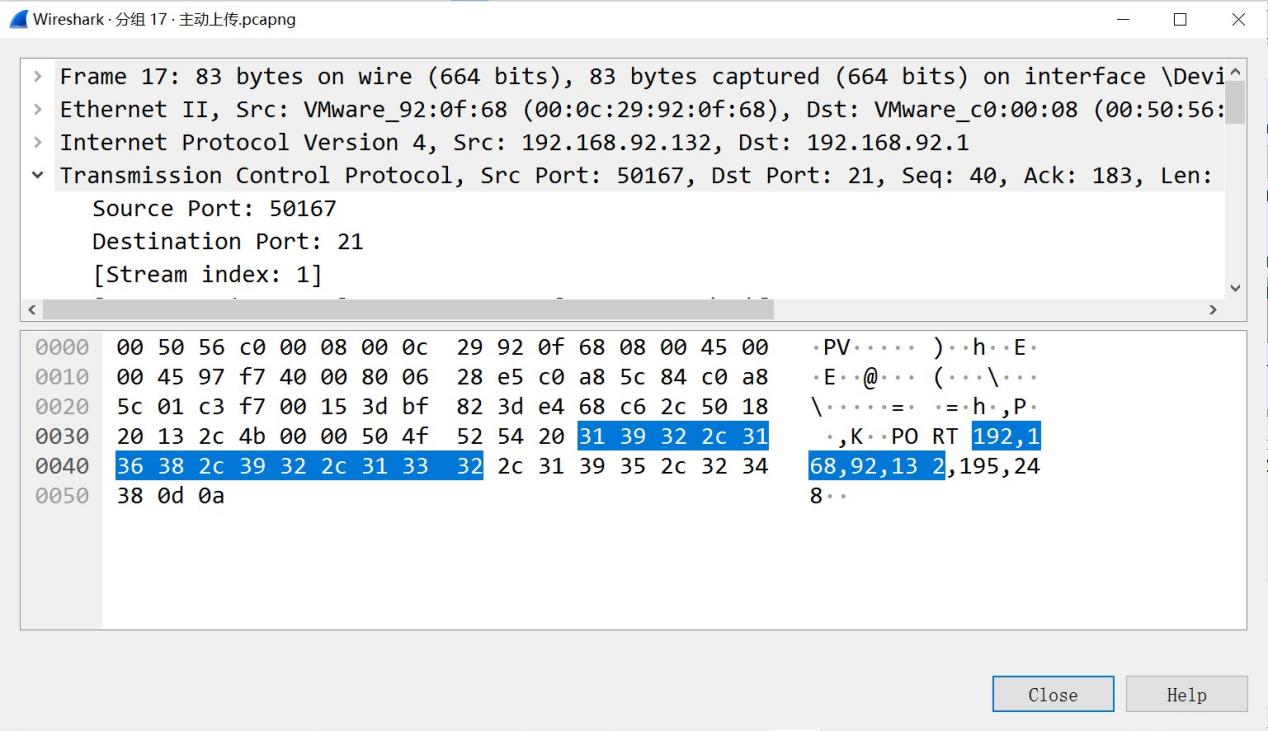
总结：主动模式的传输过程：

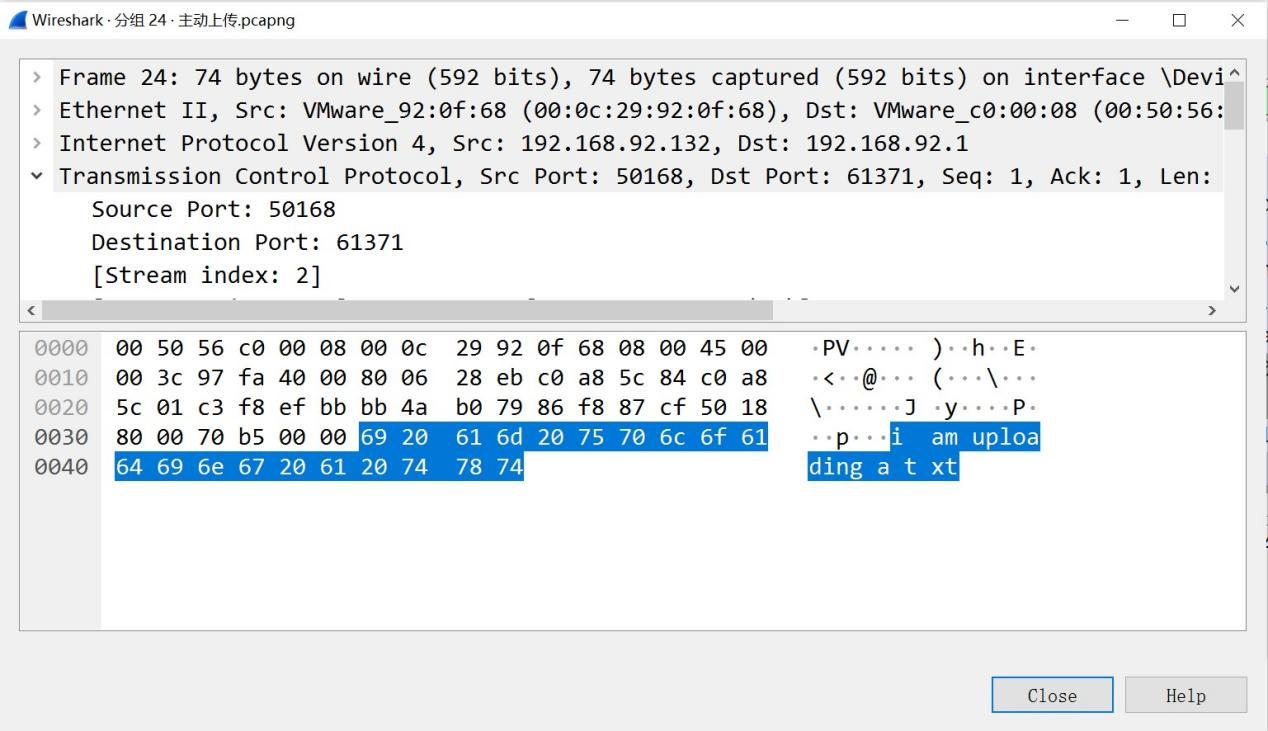
1. 服务器打开21端口
2. 客户机使用随机端口（50160）连接到21作为数据连接
3. 客户机开放N+1端口（50161），并且告诉服务器使用主动模式，并告知端口号
4. 服务器使用某个端口（多数情况下是20，但不一定）连接N+1端口（50161），使用纯TCP完成数据传输
5. 关闭数据连接

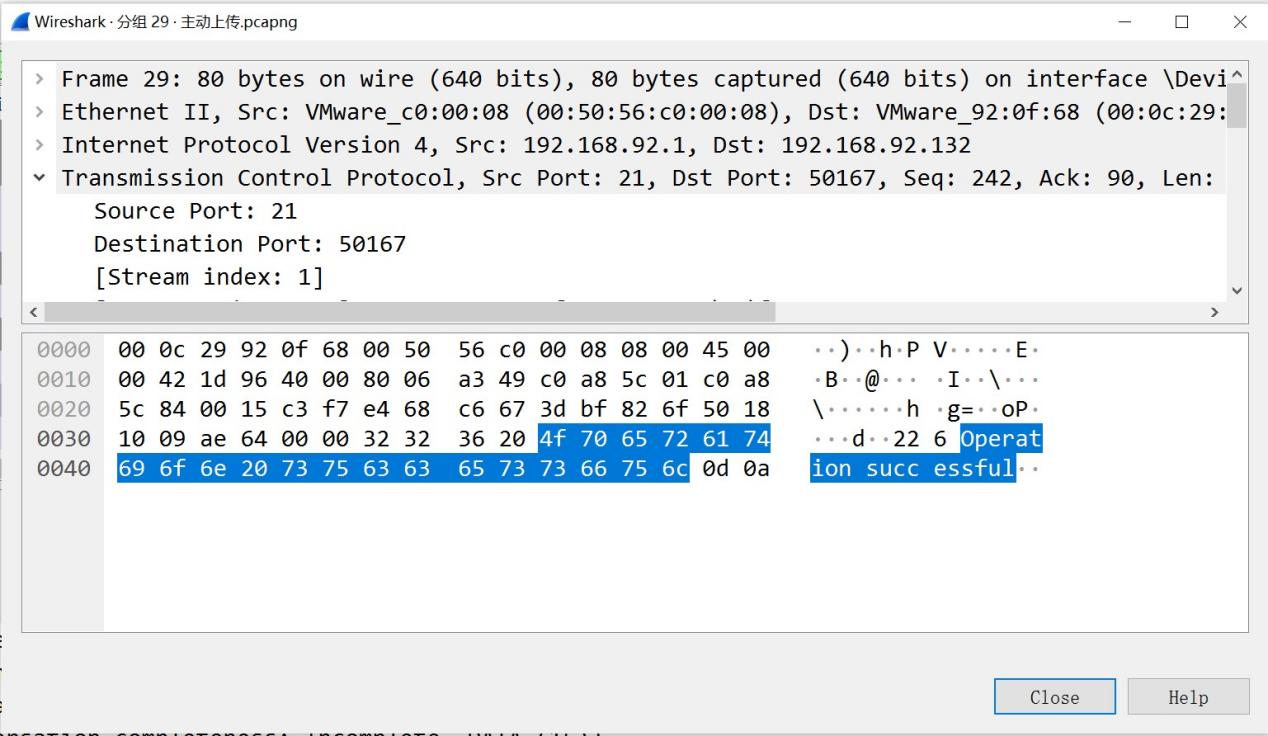
3.5（补充）主动工作模式下的上传数据

原理和主动模式下载一样的，这里只演示过程，不再解释原因









**4.试验总结**

被动模式FTP的传输过程：

（1） 服务器监听21端口，客户机使用随机端口连接，这条路作为控制连接

（2） 传输文件前，服务器用enter passive mode告诉客户端使用什么端口

（3） 客户端使用N+1端口连接被告知的端口，完成数据传输后关闭连接

主动模式的传输过程：

（1） 服务器打开21端口

（2） 客户机使用随机端口连接到21作为数据连接

（3） 客户机开放N+1端口，并且告诉服务器使用主动模式，并告知端口号

（4） 服务器使用某个端口（多数情况下是20，但不一定）连接N+1端口，使用纯TCP完成数据传输

（5） 关闭数据连接

**5.非预期现象及原因探讨**

5.1描述

奇怪的现象：主动模式不绑定20端口

5.2原因猜测

1.filezilla本身的设计问题：

因为我只用过Java编写过网络程序，因此下面的猜测基于Java的API，尽管filezilla是一个C语言程序，但我认为道理应该是一样的

主动模式是客户端建立一个新的ServerSocket绑定新端口（往往是N+1），告诉服务器新的端口，然后让服务器建立Socket与之连接。传统的主动模式，这个Socket绑定20端口。但是从程序设计的角度说，哪怕是不绑定20端口，只要连接建立，理论上也能实现功能。

众所周知Java创建Socket对象有两种方法，一种是Socket(String address, int port);这个构造器不指定本地端口，而是随机分配的。另一种方法是构造器Socket();然后使用 bind(InetSocketAddress address);方法绑定本地端口，最后使用connect(InetSocketAddress address, int timeout);建立连接，这种方法可以设置本地端口。

所以很有可能filezilla的代码就是第一种方式

如何验证：

看源码，反正这玩意开源，奈何本人水平有限，看不懂这么复杂的C程序。。。（我也没写过）

不过我们可以反证法，毕竟验证第二个原因还是比较简单的

2.20端口被系统占用，绑定失败：

众所周知微软操作系统自带文件共享功能。有可能，这个20端口，她已经，名（被）花（占）有（用）主（了）。

我曾经想用微软操作系统的IIS功能完成这个实验，但配置过于繁琐，遂放弃。也许那个方法能观察到20端口。

如何验证：

自己写程序，尝试绑定20端口，如果绑定失败，就可以证明这个原因成立

我们准备了两段代码，一个模拟服务器，一个模拟客户机，其中客户机程序将尝试绑定20端口（这两个程序的通讯内容不重要，本人习惯用一些身边的名词作为测试程序的参数，我们只是为了研究程序。测试数据的实际含义无关紧要）

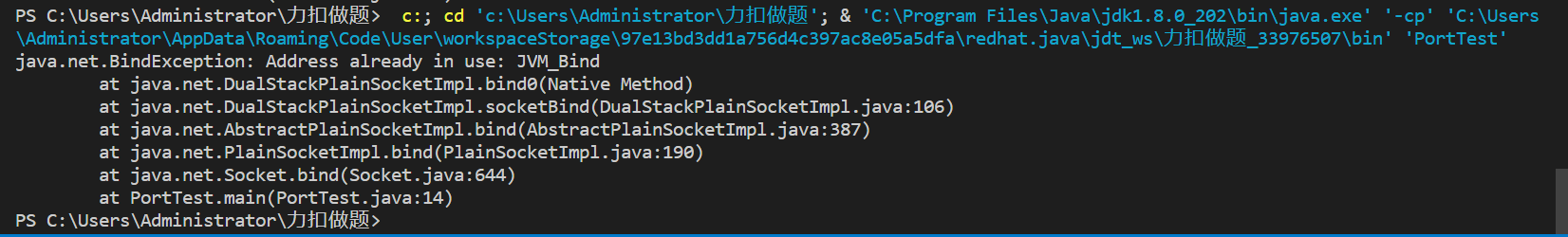
服务器：

|  |  |
| --- | --- |
| ServerTest.java | Java |
| import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.Scanner;  public class ServerTest {      public static void main(String[] args) {          ServerSocket serverSocket=null;          try {              //bind              serverSocket=new ServerSocket(9000);              System.out.println("OK");              while(true)              //listen              {                  //socket是可以线程复用的：(socket克隆)                  Socket socket=serverSocket.accept();                  new Thread(doWithSocket(socket)).run();              }          } catch (IOException e) {              // TODO Auto-generated catch block              e.printStackTrace();          }          finally          {              if(serverSocket!=null)              {                  try {                      serverSocket.close();                  } catch (IOException e) {                      // TODO Auto-generated catch block                      e.printStackTrace();                  }              }          }      }      private static Runnable doWithSocket(Socket socket) {          Runnable runnable=new Runnable() {              public void run()              {                  try {                      InputStream in=socket.getInputStream();                      OutputStream out=socket.getOutputStream();                      Scanner scanner=new Scanner(in,"UTF-8");                      PrintWriter printWriter=new PrintWriter(new OutputStreamWriter(out, "UTF-8"),true);                      //输入和输出是互不干扰的                      printWriter.println("输出");                      boolean done=false;                      while(!done&&scanner.hasNextLine())                      {                          //这地方scanner做了包装，一旦超时直接抛出IO异常                          String s=scanner.nextLine();                          System.out.println("输入："+s);                          if(s.equals("BYE"))                          {                              done=true;                          }                      }                      scanner.close();                  } catch (IOException e) {                      // TODO Auto-generated catch block                      e.printStackTrace();                  }              }          };          return runnable;      }  } | |

客户机：

|  |  |
| --- | --- |
| PortTest.java | Java |
| import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  import java.net.InetAddress;  import java.net.InetSocketAddress;  import java.net.Socket;  public class PortTest {      public static void main(String[] args) {          try {              Socket socket=new Socket();              InetAddress address=InetAddress.getByName("localhost");              InetSocketAddress socketAddress=new InetSocketAddress(address, 20);              socket.bind(socketAddress);              socket.connect(new InetSocketAddress(InetAddress.getByName("localhost"), 9000), 1000);                InputStreamReader reader=new InputStreamReader(socket.getInputStream(),"UTF-8");              OutputStreamWriter writer=new OutputStreamWriter(socket.getOutputStream(),"UTF-8");              writer.write("沐思遥");              int len=0;              char[] cbuf=new char[1024];              while((len=reader.read(cbuf))!=-1)              {                  System.out.print(new String(cbuf,0,len));              }            } catch (IOException e) {              // TODO Auto-generated catch block              e.printStackTrace();          }      }  } | |

多次运行这两个程序，部分情况下运行成功，部分情况下报错：



很显然这个异常是端口占用。

至此得出结论：观察不到20端口，大概率是由于这个端口被系统保留

（至于为什么是大概率，因为上面的程序仍然有运行成功的时候，所以要想100%定位问题还需要分析filezilla源码，这个等大佬帮忙吧，本人能力有限）

**6.成员分工&致谢**

直接参与提交的成员有：孙嘉祎，李静，黄欣灵，崔家源

提供硬件的成员有：黄欣灵

感谢刘猛老师的指导和对异常现象的分析解释，感谢沐思遥发现并提出20端口未启用的相关问题