计算机网络原理实验报告

学院 计算机学院 专业 计算机科学与技术 班级 10012006

学号 2020303245 姓名 夏卓 实验时间： 2022/12/3

一、 实验名称：

ICMP 协议分析与验证

二、 实验目的：

在分析 ping 命令实现实例代码基础上，理解该命令的实现原理；通过构造并发送 ICMP ECHO 请求报文，在目标计算机上对 ICMP ECHO 请求报文实施接收和解析，深刻理解 ICMP 协议的工作原理。

三、 实验环境：

Win10,Intelx86，wireshark，Visual Studio 2019

四、 实验内容及步骤：

实验内容：

（1）分析 ping 命令实现的实例代码。

（2）在发送端构造 ICMP ECHO 请求报文并发送给接收端。

（3）在接收端接收 ICMP ECHO 请求报文并解析与显示首部各个字段的值。

实验步骤：

（1）分析 ping 命令实现的实例代码，并总结分析其实现原理和工作流程。

ping 是用来探测本机与网络中另一主机之间是否可达的命令，其基于 ICMP 协议来工作。

运行时，ping 命令会发送一份 ICMP 回显请求报文给目标主机，并等待目标主机返回 ICMP 回显应答。因为 ICMP 协议会要求目标主机在收到消息之后，必须返回 ICMP 应答消息给源主机，如果源主机在一定时间内收到了目标主机的应答，则表明两台主机之间网络是可达的。

（2）在发送端构造 ICMP ECHO 请求报文并发送给接收端。

本次实验中，主机 A 的 IP 为：192.168.26.124；主机 B 的 IP 为：192.168.26.31。

现在要监测主机 A 和主机 B 之间网络是否可达，首先需要在主机 A 上输入命令：ping 192.168.26.31。此时，ping 命令会在主机 A 上构建一个 ICMP 的请求数据包,然后 ICMP 协议会将这个数据包以及目标 IP(192.168.26.31）等信息一同交给 IP 层协议。IP 层协议得到这些信息后，将源地址（即本机 IP：192.168.26.124）、目标地址（即目标 IP：192.168.26.31）、再加上

一些其它的控制信息，一起构建成一个 IP 数据包。

IP 数据包构建完成后，还需要加上 MAC 地址，因此，还需要通过 ARP 映射表找出目标 IP 所对应的 MAC 地址。当拿到了目标主机的 MAC 地址和本机 MAC 后，一并交给数据链路层，组装成一个数据帧，依据以太网的介质访问规则，将它们传送出出去。

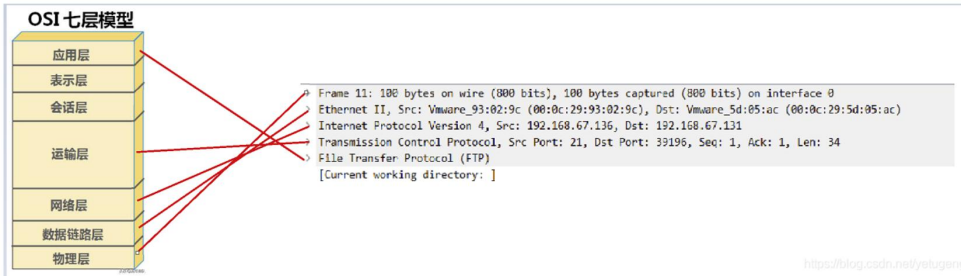
```
C:\Users\xz276>ping 192.168.26.31

正在 Ping 192.168.26.31 具有 32 字节的数据:
来自 192.168.26.31 的回复: 字节=32 时间=41ms TTL=128
来自 192.168.26.31 的回复: 字节=32 时间=9ms TTL=128
来自 192.168.26.31 的回复: 字节=32 时间=5ms TTL=128
来自 192.168.26.31 的回复: 字节=32 时间=16ms TTL=128

192.168.26.31 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 5ms，最长 = 41ms，平均 = 17ms
```

（3）在接收端接收 ICMP ECHO 请求报文，并解析其 ICMP 报文首部各个字段，在屏幕上显示解析结果。

在主机 A 和主机 B 利用 wireshark 进行抓包分析，显示过滤器中可以输入二者的 ip 地址或直接输入 icmp 协议

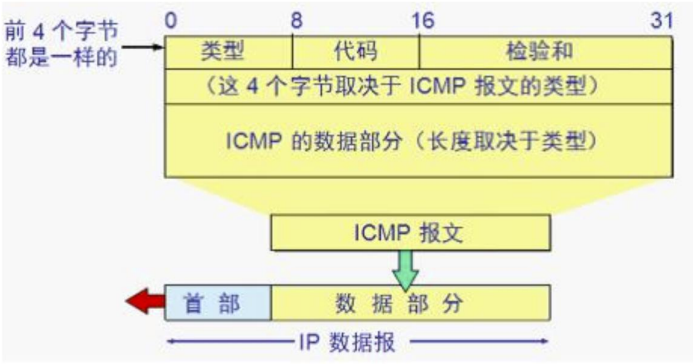
```
ip.addr == 192.168.26.124  && ip.addr == 192.168.26.31
```

利用 wireshark 自带的分组详情界面，可以很方便的查看 ICMP 报文首部的各个字段。



五、 实验结果：

首先分析 ICMP 的报文格式：



由图可知，ICMP 数据包由 8bit 的类型字段和 8bit 的代码字段以及 16bit 的校验字段再加上选项数据组成。

ICMP 的类型大致可分为以下两种：

ICMP报文类型	类型的值	ICMP的报文类型
差错报文类型	3	终点不可达
	4	源点抑制
	11	时间超过
	12	参数问题
	5	改变路由
询问报文	8	回送请求
	0	回送回答
	13	时间戳请求
	14	时间戳会带

本次实验所用到的 ICMP 类型属于第二种：查询报文类型。

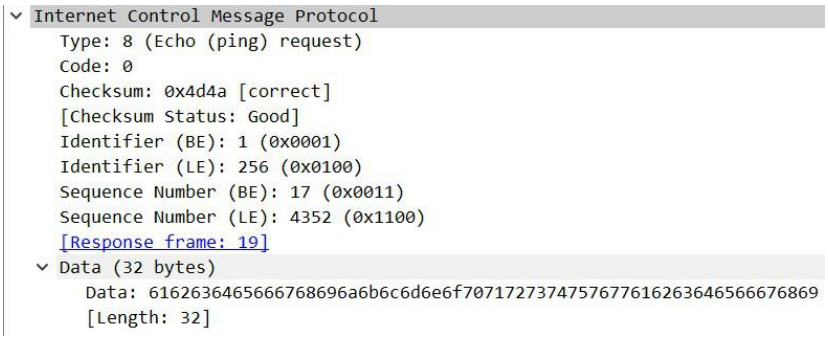
利用 wireshark 抓包结果如下所示：

发送端发送的第一个 ICMP ECHO 请求：

```
Internet Protocol Version 4, Src: 192.168.26.124, Dst: 192.168.26.31
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x27b8 (10168)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.26.124
    Destination Address: 192.168.26.31
> Internet Protocol Version 4, Src: 192.168.26.124, Dst: 192.168.26.31
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d4a [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 17 (0x0011)
    Sequence Number (LE): 4352 (0x1100)
    [Response frame: 7]
  v Data (32 bytes)
      Data: 61626364656667686 96a6b6c6d6e6f707172737475767761626364656667686 9
      [Length: 32]
```

接收端接收的第一个 ICMP ECHO 请求：

```
v Internet Protocol Version 4, Src: 192.168.26.124, Dst: 192.168.26.31
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x27b8 (10168)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x9d1d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.26.124
    Destination Address: 192.168.26.31
```



可以看到接收到的 IP 报文首部和 ICMP 报文首部各个字段都与发送报文完全一致（注意到虽然 IP 首部的校验和不一样但是该字段是 validation disabled 的，因此是无效字段）

在 IP 报文首部可以得出如下信息：

源 IP 地址：192.168.26.124，目的 IP 地址：192.168.26.31，总长度为 60，TTL 为 64，协议类型为 ICMP。

在 ICMP 首部字段可以得出如下信息：

Type = 8（说明是回送请求），Code = 0，校验和为 0x4d4a。

六、 实验总结

思考题：

Traceroute 是用来侦测源主机到目标主机之间所经过路由情况的常用工具。其原理就是利用 ICMP 的规则，制造一些错误的事件出来，然后根据错误的事件来评估网络路由情况。

具体做法就是：

Traceroute 会设置特殊的 TTL 值，来追踪源主机和目标主机之间的路由数。首先它给目标主机发送一个 TTL=1 的 UDP 数据包，那么这个数据包一旦在路上遇到一个路由器，TTL 就变成了 0（TTL 规则是每经过一个路由器都会减 1），因为 TTL=0 了，所以路由器就会把这个数据包丢掉，然后产生一个错误类型（超时）的 ICMP 数据包回发给源主机，也就是差错包。这个时候源主机就拿到了第一个路由节点的 IP 和相关信息了。

接着，源主机再给目标主机发一个 TTL=2 的 UDP 数据包，依旧上述流程走一遍，就知道第二个路由节点的 IP 和耗时情况等信息了。

如此反复进行，Traceroute 就可以拿到从主机 A 到主机 B 之间所有路由器的信息了。最后，为了检测目标主机，会设置一个不可能达到的目标端口号，使目标主机产生一份“端口不可达”的错误 ICMP 报文返回给源主机。

教师评语：

成绩：_____教师签名：_____批阅日期：_____

计算机网络原理实验报告

学院 计算机学院 专业 计算机科学与技术 班级 10012006

学号 2020303245 姓名 夏卓 实验时间： 2022/12/3

一、 实验名称：

FTP 客户端多进程编程

二、 实验目的：

通过设计和实现一个 FTP 客户端系统，深刻理解 FTP 协议工作原理，重点掌握 FTP 协议设计与实现中控制连接和数据连接建立过程，两个连接通信模式特点。

三、 实验环境：

Win10,Intelx86，wireshark，Visual Studio 2019

四、 实验内容及步骤：

实验内容：

（1）FTP 客户端系统的设计，理解 FTP 协议中数据连接建立两种方式区别：被动模式和主动模式；

（2）FTP 客户端系统的实现，涉及控制连接、数据连接建立；通过在控制连接传输命令，数据连接传输数据，利用多进程编程，实现一个 FTP 客户端系统。

实验步骤：

（1）首先在 FTP 客户端和服务器之间建立控制连接。

```
//创建控制连接 socket
SocketControl = socket(AF_INET, SOCK_STREAM, 0);
if (SocketControl == INVALID_SOCKET)
{
    printf("creat TCP Control socket error !");
    return -1;
}
//定义 FTP 服务器控制连接地址和端口号
sockaddr_in server_addr;
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(21);
server_addr.sin_addr.s_addr = inet_addr(argv[1]);
//向 FTP 服务器发送控制连接请求
printf("FTP control connect......");
int nConnect = connect(SocketControl, (sockaddr *)&server_addr, sizeof(server_addr));
if (nConnect == SOCKET_ERROR)
{
    printf("client could not establish the FTP control connection with server\n");
    return -1;
}
```

（2）FTP 客户端通过控制连接向服务器发送账号信息（用户名+密码），进行身份认证。

```
//向服务器发送 USER 命令
printf("FTP->USER:");
memset(CmdBuf, 0, MAX_SIZE);
gets(CmdBuf); //输入用户名并保存
memset(command, 0, MAX_SIZE);
memcpy(command, "USER ", strlen("USER "));
memcpy(command + strlen("USER "), CmdBuf, strlen(CmdBuf));
memcpy(command + strlen("USER ") + strlen(CmdBuf), "\r\n", 2);
if (!SendCommand())
    return -1;
//获得 USER 命令的应答信息
if (RecvReply())
{
    if (nReplyCode == 230 || nReplyCode == 331)
        printf("%s", ReplyMsg);
```

```
if (nReplyCode == 331)
{
    //向 FTP 服务器发送 PASS 命令
    printf(" FTP > PASS:");
    memset(CmdBuf, 0, MAX_SIZE);
    for (int i = 0; i < MAX_SIZE; i++)
    {
        CmdBuf[i] = getch(); //输入用户密码
        if (CmdBuf[i] == '\r')
        {
            CmdBuf[i] = '\0';
            break;
        }
        else
            printf("*");
    }
    printf("\n");
    memset(command, 0, MAX_SIZE);
    memcpy(command, "PASS ", strlen("PASS "));
    memcpy(command + strlen("PASS "), CmdBuf, strlen(CmdBuf));
    memcpy(command + strlen("PASS ") + strlen(CmdBuf), "\r\n", 2);
    if (!SendCommand())
        return -1;
```

（3）FTP 客户端通过控制连接向服务器发送 passive 命令，说明采用被动模式建立数据连接。

```
//向 FTP 服务器发送 PASV 命令
memset(command, 0, MAX_SIZE);
memcpy(command, "PASV", strlen("PASV"));
memcpy(command + strlen("PASV"), "\r\n", 2);
if (!SendCommand())
{
    return false;
}
```

（4）FTP 客户端与服务器之间通过被动模式建立数据连接。

```
//与 FTP 服务器发送建立数据 TCP 连接请求
int nConnect = connect(SocketData, (sockaddr *)&server_addr, sizeof(server_addr));
if (nConnect == SOCKET_ERROR)
{
    printf("create data TCP connection error : %d\n", WSAGetLastError());
    return false;
}
return true;
```

（5）FTP 客户端向服务器发送 dir 命令，服务器对该命令进行处理，并向客户端发送处理结果；

```
//向 FTP 服务器发送 LIST 命令
printf("ftp>LIST\r\n");
char ftpserver[MAX_SIZE];
memset(ftpserver, 0, MAX_SIZE);
memcpy(ftpserver, argv[1], strlen(argv[1]));
if (!DataConnect(ftpserver))
    return -1;
memset(command, 0, MAX_SIZE);
memcpy(command, "LIST", strlen("LIST"));
memcpy(command + strlen("LIST"), "\r\n", 2);
if (!SendCommand())
    return -1;
```

（6）FTP 客户端接收服务器发送来的处理结果（获得服务器当前目录下的列表信息），并在屏幕上显示。

```
//获取 LIST 命令的应答信息
if (RecvReply())
{
    if (nReplyCode == 226)
        cout << ReplyMsg;
    else
    {
        printf("LIST 命令应答错误 !\r\n");
        closesocket(SocketControl);
        return -1;
    }
}
```

（7）释放数据连接。

```
//获取 LIST 命令的目录信息
char ListBuffer[MAX_SIZE];
while (true)
{
    memset(ListBuffer, 0, MAX_SIZE);
    nRecv = recv(SocketData, ListBuffer, MAX_SIZE, 0);
    if (nRecv == SOCKET_ERROR)
    {
        printf("数据接收错误 !\r\n");
        closesocket(SocketData);
        return -1;
    }
    if (nRecv <= 0)
        break;
    cout << ListBuffer;
}
closesocket(SocketData);
```

（8）FTP 客户端向服务器发送 quit 命令，并释放控制连接。

```
//向 FTP 服务器发送 quit 命令
printf("FTP->QUIT:");
memset(command, 0, MAX_SIZE);
memcpy(command, "QUIT", strlen("QUIT"));
memcpy(command + strlen("QUIT"), "\r\n", 2);
if (!SendCommand())
    return -1;
```

（9）通信结束。

五、 实验结果：

客户端 IP 地址：192.168.26.124，目的 IP 地址：192.168.26.31

```
(base) PS D:\大学\大三上\计网\实验\实验4> .\ftp_client 192.168.26.31
FTP control connect......220 Xlight FTP 3.9 就绪...

FTP->USER:123
331 需要密码 123
331 FTP > PASS:******
230 登录成功
ftp>LIST
server data Port:50006
150 正在打开文本模式数据连接为 /bin/ls (289 比特).
drw-rw-rw- 1 ftp ftp                0 Dec 02 19:21 .
drw-rw-rw- 1 ftp ftp                0 Dec 02 19:21 ..
-rw-rw-rw- 1 ftp ftp                0 Dec 02 19:14 54.txt
-rw-rw-rw- 1 ftp ftp                5 Dec 02 19:21 casc.txt
-rw-rw-rw- 1 ftp ftp                0 Dec 02 19:13 kmt.txt
226 传送完毕 (0.000 KB/s).
FTP->QUIT:221 再见
```

捕获的结果如下所示（图中乱码处是由于不支持中文字符显示造成的，并不是传输或实验的问题）：

No.	Time	Source	Destination	Protocol	Length	Info
95	2022-12-03 10:52:04.984118	192.168.26.124	192.168.26.31	TCP	74	2679 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM…
96	2022-12-03 10:52:04.988471	192.168.26.31	192.168.26.124	TCP	66	21 → 2679 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=25…
97	2022-12-03 10:52:04.988632	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [ACK] Seq=1 Ack=1 Win=131328 Len=0
98	2022-12-03 10:52:05.159911	192.168.26.31	192.168.26.124	FTP	82	Response: 220 Xlight FTP 3.9 ■■■■...
100	2022-12-03 10:52:05.208484	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [ACK] Seq=1 Ack=29 Win=131328 Len=0
129	2022-12-03 10:52:07.820386	192.168.26.124	192.168.26.31	FTP	64	Request: USER 123
130	2022-12-03 10:52:07.868070	192.168.26.31	192.168.26.124	TCP	54	21 → 2679 [ACK] Seq=29 Ack=11 Win=131328 Len=0
132	2022-12-03 10:52:08.169021	192.168.26.31	192.168.26.124	FTP	72	Response: 331 ■■ç□□□□ 123
133	2022-12-03 10:52:08.214223	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [ACK] Seq=11 Ack=47 Win=131328 Len=0
162	2022-12-03 10:52:10.482304	192.168.26.124	192.168.26.31	FTP	67	Request: PASS 123456
163	2022-12-03 10:52:10.532689	192.168.26.31	192.168.26.124	TCP	54	21 → 2679 [ACK] Seq=47 Ack=24 Win=131328 Len=0
165	2022-12-03 10:52:10.678721	192.168.26.31	192.168.26.124	FTP	68	Response: 230 ■■¾■ɹ■
166	2022-12-03 10:52:10.679832	192.168.26.124	192.168.26.31	FTP	60	Request: PASV
168	2022-12-03 10:52:10.733403	192.168.26.31	192.168.26.124	TCP	54	21 → 2679 [ACK] Seq=61 Ack=30 Win=131328 Len=0
169	2022-12-03 10:52:10.759290	192.168.26.31	192.168.26.124	FTP	104	Response: 227 Entering Passive Mode (192,168,26,31,195,86)
170	2022-12-03 10:52:10.760127	192.168.26.124	192.168.26.31	TCP	74	2681 → 50006 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_P…
171	2022-12-03 10:52:10.765792	192.168.26.31	192.168.26.124	TCP	66	50006 → 2681 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460 WS…
172	2022-12-03 10:52:10.766111	192.168.26.124	192.168.26.31	TCP	54	2681 → 50006 [ACK] Seq=1 Ack=1 Win=131328 Len=0
173	2022-12-03 10:52:10.766296	192.168.26.124	192.168.26.31	FTP	60	Request: LIST
175	2022-12-03 10:52:10.795774	192.168.26.31	192.168.26.124	FTP	106	Response: 150 ■■■ż□□ɹ■ĝ‘■■■l■■Ĭ /bin/ls (289 ■■■■).
176	2022-12-03 10:52:10.795774	192.168.26.31	192.168.26.124	FTP-DATA	343	FTP Data: 289 bytes (PASV) (LIST)
177	2022-12-03 10:52:10.795774	192.168.26.31	192.168.26.124	TCP	54	50006 → 2681 [FIN, ACK] Seq=290 Ack=1 Win=32768 Len=0
178	2022-12-03 10:52:10.796113	192.168.26.124	192.168.26.31	TCP	54	2681 → 50006 [ACK] Seq=1 Ack=291 Win=131072 Len=0
179	2022-12-03 10:52:10.798607	192.168.26.124	192.168.26.31	TCP	54	2681 → 50006 [FIN, ACK] Seq=1 Ack=291 Win=131072 Len=0
180	2022-12-03 10:52:10.803333	192.168.26.31	192.168.26.124	TCP	54	50006 → 2681 [ACK] Seq=291 Ack=2 Win=32768 Len=0
182	2022-12-03 10:52:10.850268	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [ACK] Seq=36 Ack=163 Win=131072 Len=0
183	2022-12-03 10:52:10.858058	192.168.26.31	192.168.26.124	FTP	82	Response: 226 ■■■■■■■ (0.000 KB/s).
184	2022-12-03 10:52:10.859666	192.168.26.124	192.168.26.31	FTP	60	Request: QUIT
185	2022-12-03 10:52:10.881302	192.168.26.31	192.168.26.124	FTP	64	Response: 221 ■□□
186	2022-12-03 10:52:10.881884	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [FIN, ACK] Seq=42 Ack=201 Win=131072 Len=0
187	2022-12-03 10:52:10.886453	192.168.26.31	192.168.26.124	TCP	54	21 → 2679 [ACK] Seq=201 Ack=43 Win=131328 Len=0
188	2022-12-03 10:52:10.930825	192.168.26.31	192.168.26.124	TCP	54	21 → 2679 [FIN, ACK] Seq=201 Ack=43 Win=131328 Len=0
189	2022-12-03 10:52:10.930888	192.168.26.124	192.168.26.31	TCP	54	2679 → 21 [ACK] Seq=43 Ack=202 Win=131072 Len=0

整个 FTP 通信流程大致如下：

首先客户端与服务器进行三次握手，建立 TCP 连接。

然后服务器响应 FTP 连接，客户端输入用户名，服务器端提示输入密码，客户端输入密码。

之后客户端请求被动模式，服务器端响应被动模式，并将监听的数据端口 50006（195*256+86）发送给客户端。数据连接也要经过三次握手来建立。

之后客户端发起 list 请求，服务器端传输数据包给客户端，客户端收到数据后要进行确认。传输完毕后经过四次挥手释放数据端口的连接，服务端提示传输完毕。

最后客户端请求断开 FTP 连接，服务端响应 FTP 断开，并通过控制连接的四次挥手断开连接。

另外，分析 FTP-DATA 数据包，可以看到传输过程中 dir 命令的执行结果，即文件夹下的文件目录：

```
Internet Protocol Version 4, Src: 192.168.26.31, Dst: 192.168.26.124
Transmission Control Protocol, Src Port: 52580, Dst Port: 2318, Seq: 1, Ack: 1, Len: 289
FTP Data (289 bytes data)
[Setup frame: 86]
[Setup method: PASV]
[Command: LIST]
Command frame: 90
[Current working directory: ]
Line-based text data (5 lines)
   drw-rw-rw- 1 ftp ftp               0 Dec 02 19:21 .\r\n
   drw-rw-rw- 1 ftp ftp               0 Dec 02 19:21 ..\r\n
   -rw-rw-rw- 1 ftp ftp               0 Dec 02 19:14 54.txt\r\n
   -rw-rw-rw- 1 ftp ftp               5 Dec 02 19:21 casc.txt\r\n
   -rw-rw-rw- 1 ftp ftp               0 Dec 02 19:13 kmt.txt\r\n
```

可以看到，在数据连接上，FTP 服务器发送给 FTP 客户端数据字节个数为 289 bytes

六、 实验总结

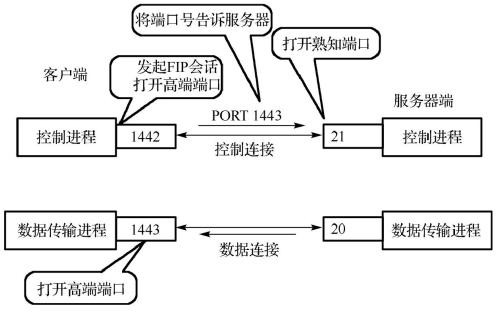
主动模式和被动模式的不同之处：

1. 被动模式是服务端发送自己监听的数据端口（包括 IP），等待客户端来连接；而主动模式是客户端发送自己监听的数据端口（包括 IP），等待服务端主动过来连接。

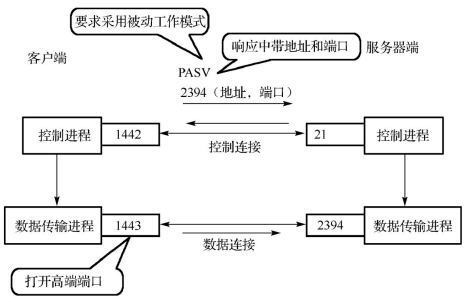
2. 被动模式下，服务端发送的端口范围可以在配置文件里控制，而主动模式下，客户端发送的端口是随机的，没有办法控制。

3. 在被动模式下，双方都可以开启防火墙，服务端在 iptables 上打开 21 和相应的数据端口即可；而在主动模式下，服务端 iptables 只需要开启 21 端口等待客户端的控制连接，客户端则不能打开防火墙，否则服务端的数据通道连接不上客户端监听的随机端口。

详细通信流程如下所示：



FTP协议主动工作模式流程



FTP协议被动工作模式流程

教师评语：

成绩：_____教师签名：_____批阅日期：_____