第27章 FTP：文件传送协议

27.1 引言

FTP是另一个常见的应用程序。它是用于文件传输的 Internet标准。我们必须分清文件传送（file transfer）和文件存取 (file access) 之间的区别，前者是 FTP提供的，后者是如 NFS（Sun的网络文件系统，第 29章）等应用系统提供的。由 FTP提供的文件传送是将一个完整的文件从一个系统复制到另一个系统中。要使用 FTP，就需要有登录服务器的注册帐号，或者通过允许匿名FTP的服务器来使用（本章我们将给出这样的一个例子）。

与Telnet类似，FTP最早的设计是用于两台不同的主机，这两个主机可能运行在不同的操作系统下、使用不同的文件结构、并可能使用不同字符集。但不同的是，Telnet获得异构性是强制两端都采用同一个标准：使用7比特ASCII码的NVT。而FTP是采用另一种方法来处理不同系统间的差异。FTP支持有限数量的文件类型（ASCII，二进制，等等）和文件结构（面向字节流或记录）。

参考文献959 [Postel 和 Reynolds 1985] 是FTP的正式规范。该文献叙述了近年来文件传输的历史演变。

27.2 FTP协议

FTP与我们已描述的另一种应用不同，它采用两个 TCP连接来传输一个文件。

1) 控制连接以通常的客户服务器方式建立。服务器以被动方式打开众所周知的用于FTP的端口（21），等待客户的连接。客户则以主动方式打开 TCP端口21，来建立连接。控制连接始终等待客户与服务器之间的通信。该连接将命令从客户传给服务器，并传回服务器的应答。

   由于命令通常是由用户键入的，所以IP对控制连接的服务类型就是“最大限度地减小迟延”。

2) 每当一个文件在客户与服务器之间传输时，就创建一个数据连接。（其他时间也可以创建，后面我们将说到）。

   由于该连接用于传输目的，所以IP对数据连接的服务特点就是“最大限度提高吞吐量”。

图27-1描述了客户与服务器以及它们之间的连接情况

从图中可以看出，交互式用户通常不处理在控制连接中转换的命令和应答。这些细节均由两个协议解释器来完成。标有“用户接口”的方框功能是按用户所需提供各种交互界面（全屏幕菜单选择，逐行输入命令，等等），并把它们转换成在控制连接上发送的 FTP命令。类似地，从控制连接上传回的服务器应答也被转换成用户所需的交互格式。

从图中还可以看出，正是这两个协议解释器根据需要激活文件传送功能。

27.2.1 数据表示

FTP协议规范提供了控制文件传送与存储的多种选择。在以下四个方面中每一个方面都必须作出一个选择。

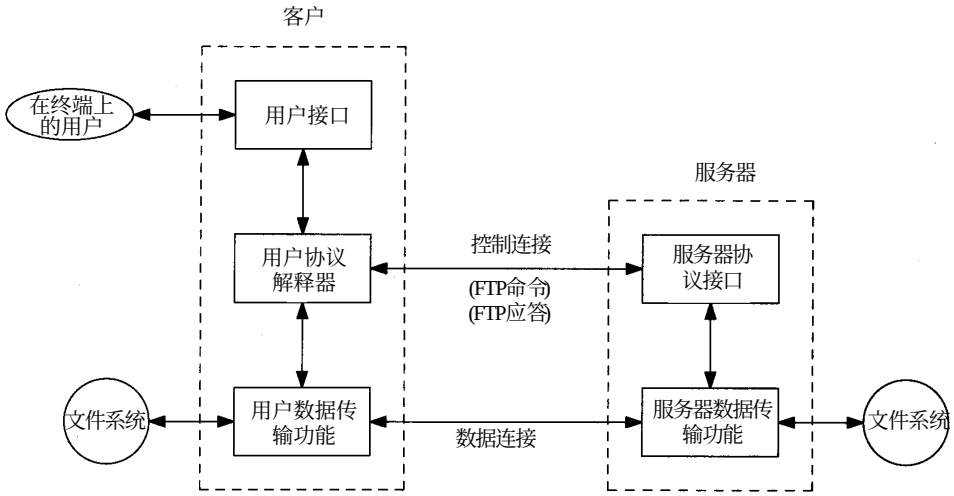

图27-1 文件传输中的处理过程

1. 文件类型

(a) ASCII码文件类型 （默认选择）文本文件以NVT ASCII码形式在数据连接中传输。这要求发方将本地文本文件转换成NVT ASCII码形式，而收方则将NVT ASCII码再还原成本地文本文件。其中，用NVT ASCII码传输的每行都带有一个回车，而后是一个换行。这意味着收方必须扫描每个字节，查找CR、LF对（我们在第15.2节见过的关于TFIP的ASCII码文件传输情况与此相同）。

(b) EBCDIC文件类型 该文本文件传输方式要求两端都是EBCDIC系统。

(c) 图像文件类型（也称为二进制文件类型） 数据发送呈现为一个连续的比特流。通常用于传输二进制文件。

(d) 本地文件类型 该方式在具有不同字节大小的主机间传输二进制文件。每一字节的比特数由发方规定。对使用8 bit字节的系统来说，本地文件以8 bit字节传输就等同于图像文件传输。

2. 格式控制

该选项只对ASCII和EBCDIC文件类型有效。

(a) 非打印 （默认选择）文件中不含有垂直格式信息。

(b) 远程登录格式控制 文件含有向打印机解释的远程登录垂直格式控制。

(c) Fortran 回车控制 每行首字符是Fortran格式控制符。

3. 结构

（a） 文件结构 （默认选择）文件被认为是一个连续的字节流。不存在内部的文件结构。

（b） 记录结构 该结构只用于文本文件（ASCII或EBCDIC）。

（c） 页结构 每页都带有页号发送，以便收方能随机地存储各页。该结构由 TOPS-20操作系统提供（主机需求RFC不提倡采用该结构）。

4. 传输方式

它规定文件在数据连接中如何传输。

（a） 流方式 （默认选择）文件以字节流的形式传输。对于文件结构，发方在文件尾提示关闭数据连接。对于记录结构，有专用的两字节序列码标志记录结束和文件结束。

（b） 块方式 文件以一系列块来传输，每块前面都带有一个或多个首部字节。

（c） 压缩方式 一个简单的全长编码压缩方法，压缩连续出现的相同字节。在文本文件

中常用来压缩空白串，在二进制文件中常用来压缩 0字节（这种方式很少使用，也不受支持。现在有一些更好的文件压缩方法来支持 FTP）。

如果算一下所有这些选择的排列组合数，那么对传输和存储一个文件来说就有 72种不同的方式。幸运的是，其中很多选择不是废弃了，就是不为多数实现环境所支持，所以我们可以忽略掉它们。

通常由Unix实现的FTP 客户和服务器把我们的选择限制如下：

- 类型：ASCII或图像。
- 格式控制：只允许非打印。
- 结构：只允许文件结构。
- 传输方式：只允许流方式。

这就限制我们只能取一、两种方式： ASCII或图像（二进制）。

该实现满足主机需求 RFC的最小需求（该 RFC也要求能支持记录结构，但只有操作系统支持它才行，而Unix不行）。

很多非Unix的实现提供了处理它们自己文件格式的 FTP功能。主机需求RFC指出“ FTP协议有很多特征，虽然其中一些通常不实现，但对 FTP中的每一个特征来说，都存在着至少一种实现 ”。

27.2.2 FTP命令

命令和应答在客户和服务器的控制连接上以 NVT ASCII码形式传送。这就要求在每行结尾都要返回CR、LF对（也就是每个命令或每个应答）。

从客户发向服务器的 Telnet命令（以IAC打头）只有中断进程（ <IAC,IP>）和Telnet的同步信号（紧急方式下 <IAC,DM>）。我们将看到这两条 Telnet命令被用来中止正在进行的文件传输，或在传输过程中查询服务器。另外，如果服务器接受了客户端的一个带选项的 Telnet命令（WILL，WONT，DO或DONT），它将以DONT 或WONT响应。

这些命令都是3或4个字节的大写ASCII字符，其中一些带选项参数。从客户向服务器发送的FTP命令超过30种。图27-2给出了一些常用命令，其中大部分将在本章再次遇到。

命　令	说　明
ABOR	放弃先前的FTP命令和数据传输
LIST *filelist*	列表显示文件或目录
PASS *password*	服务器上的口令
PORT *n*1,*n*2,*n*3,*n*4,*n*5,*n*6	客户端IP地址（*n*1.*n*2.*n*3.*n*4）和端口（*n*5×256+*n*6）
QUIT	从服务器注销
RETR *filename*	检索（取）一个文件
STOR *filename*	存储（放）一个文件
SYST	服务器返回系统类型
TYPE *type*	说明文件类型：A表示ASCII码，I表示图像
USER *username*	服务器上用户名

图27-2　常用的FTP命令

下节我们将通过一些例子看到，在用户交互类型和控制连接上传送的 FTP命令之间有时是一对一的。但也有些操作下，一个用户命令产生控制连接上多个 FTP命令。

27.2.3 FTP应答

应答都是ASCII码形式的3位数字，并跟有报文选项。其原因是软件系统需要根据数字代码来决定如何应答，而选项串是面向人工处理的。由于客户通常都要输出数字应答和报文串，一个可交互的用户可以通过阅读报文串（而不必记忆所有数字回答代码的含义）来确定应答的含义。

应答3位码中每一位数字都有不同的含义（我们将在第 28章看到简单邮件传送输协议，SMTP，使用相同的命令和应答约定）。

图27-3给出了应答代码第1位和第2位的含义。

应答	说　　明
1yz	肯定预备应答。它仅仅是在发送另一个命令前期待另一个应答时启动
2yz	肯定完成应答。一个新命令可以发送
3yz	肯定中介应答。该命令已被接受，但另一个命令必须被发送
4yz	暂态否定完成应答。请求的动作没有发生，但差错状态是暂时的，所以命令可以过后再发
5yz	永久性否定完成应答。命令不被接受，并且不再重试
x0z	语法错误
x1z	信息
x2z	连接。应答指控制或数据连接
x3z	鉴别和记帐。应答用于注册或记帐命令
x4z	未指明
x5z	文件系统状态

图27-3 应答代码3位数字中第1位和第2位的含义

第3位数字给出差错报文的附加含义。例如，这里是一些典型的应答，都带有一个可能的报文串。

- 125 数据连接已经打开；传输开始。
- 200 就绪命令。
- 214 帮助报文（面向用户）。
- 331 用户名就绪，要求输入口令。
- 425 不能打开数据连接。
- 452 错写文件。
- 500 语法错误（未认可的命令）。
- 501 语法错误（无效参数）。
- 502 未实现的MODE(方式命令)类型。

通常每个FTP命令都产生一行回答。例如， QUIT命令可以产生如下应答：

```
221 Goodbye.
```

如果需要产生一条多行应答，第1行在3位数字应答代码之后包含一个连字号，而不是空格，最后一行包含相同的3位数字应答代码，后跟一个空格符。例如，HELP命令可以产生如下应答：

```
214- The following commands are recognized (* =>'s unimplemented).
   USER    PORT    STOR    MSAM*   RNTO    NLST    MKD     CDUP
   PASS    PASV    APPE    MRSQ*   ABOR    SITE    XMKD    XCUP
   ACCT*   TYPE    MLFL*   MRCP*   DELE    SYST    RMD     STOU
   SMNT*   STRU    MAIL*   ALLO    CWD     STAT    XRMD    SIZE
```

```
   REIN*    MODE     MSND*    REST     XCWD     HELP     PWD      MDTM
   QUIT     RETR     MSOM*    RNFR     LIST     NOOP     XPWD
214 Direct comments to ftp-bugs@bsdi.tuc.noao.edu.
```

27.2.4 连接管理

数据连接有以下三大用途：

1) 从客户向服务器发送一个文件。

2) 从服务器向客户发送一个文件。

3) 从服务器向客户发送文件或目录列表。

FTP服务器把文件列表从数据连接上发回，而不是控制连接上的多行应答。这就避免了行的有限性对目录大小的限制，而且更易于客户将目录列表以文件形式保存，而不是把列表显示在终端上。

我们已说过，控制连接一直保持到客户-服务器连接的全过程，但数据连接可以根据需要随时来，随时走。那么需要怎样为数据连接选端口号，以及谁来负责主动打开和被动打开？

首先，我们前面说过通用传输方式（Unix环境下唯一的传输方式）是流方式，并且文件结尾是以关闭数据连接为标志。这意味着对每一个文件传输或目录列表来说都要建立一个全新的数据连接。其一般过程如下：

1) 正由于是客户发出命令要求建立数据连接，所以数据连接是在客户的控制下建立的。

2) 客户通常在客户端主机上为所在数据连接端选择一个临时端口号。客户从该端口发布一个被动的打开。

3) 客户使用PORT命令从控制连接上把端口号发向服务器。

4) 服务器在控制连接上接收端口号，并向客户端主机上的端口发布一个主动的打开。服务器的数据连接端一直使用端口20。

图27-4给出了第3步执行时的连接状态。假设客户用于控制连接的临时端口是1173，客户用于数据连接的临时端口是1174。客户发出的命令是PORT命令，其参数是6个ASCII中的十进制数字，它们之间由逗点隔开。前面4个数字指明客户上的IP地址，服务器将向它发出主动打开（本例中是140.252.13.34），而后两位指明16 bit端口地址。由于16 bit端口地址是从这两个数字中得来，所以其值在本例中就是4×256+150 = 1174。

图27-5给出了服务器向客户所在数据连接端发布主动打开时的连接状态。服务器的端点是端口20。

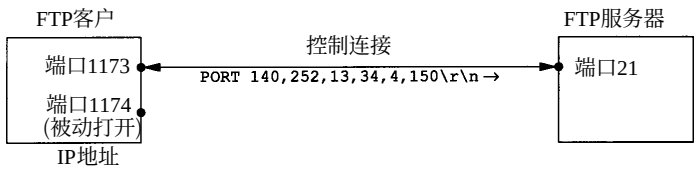

图27-4 在FTP控制连接上通过的PORT命令

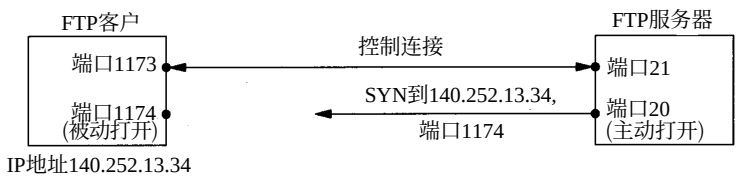

图27-5 主动打开数据连接的FTP服务器

服务器总是执行数据连接的主动打开。通常服务器也执行数据连接的主动关闭，除非当客户向服务器发送流形式的文件时，需要客户来关闭连接（它给服务器一个文件结束的通知）。

客户也有可能不发出PORT命令，而由服务器向正被客户使用的同一个端口号发出主动打开，来结束控制连接。这是可行的，因为服务器面向这两个连接的端口号是不同的：一个是20，另一个是21。不过，下节我们将看到为什么现有实现通常不这样做。

27.3 FTP的例子

现在看一些使用FTP的例子：它对数据连接的管理，采用NVT ASCII码的文本文件如何发送，FTP使用Telnet同步信号来中止进行中的文件传输，最后是常用的“匿名FTP”。

27.3.1 连接管理：临时数据端口

先看一下FTP的连接管理，它只在服务器上用简单FTP会话显示一个文件。我们用-d标志（debug）来运行svr4主机上的客户。这告诉它要打印控制连接上变换的命令和应答。所有前面冠以--->的行是从客户上发向服务器的，所有以3位数字开头的行都是服务器的应答。客户的交互提示是ftp>。

```
svr4 %ftp -d bsdi                              -d 选项用作排错输出
Connected to bsdi.                             客户执行控制连接的主动打开
220 bsdi FTP server (Version 5.60) ready       服务器响应就绪
Name (bsdi:rstevens):                          客户提示我们输入
---> USER rstevens                             键入RETURN，客户发送默认信息
331 Password required for rstevens.
Password:                                      键入口令；它不需要回显
---> PASS XXXXXXXX                                  客户以明文发送它
230 User rstevens logged in.
ftp> dir hello.c                               要求列出一个文件的目录
---> PORT 140,252,13,34,4,150                       见图27-4
200 PORT Command successful.
---> LIST hello.c
150 Opening ASCII mode data connection for /bin/ls.
-rw-r--r--   1 rstevens  staff  38 Jul 17 12:47 hello.c
226 Transfer complete.
remote: hello.c                                客户输出
56 bytes received in 0.03 seconds (1.8 Kbytes/s)
ftp> quit                                      我们已完成
---> QUIT
221 Goodbye
```

当FTP客户提示我们注册姓名时，它打印了默认值（我们在客户上的注册名）。当我们敲RETURN键时，默认值被发送出去。

对一个文件列出目录的要求引发一个数据连接的建立和使用。本例体现了我们在图27-4和图27-5中给出的程序。客户要求TCP为其数据连接的终端提供一个临时端口号，并用PORT命令发送这个端口号（1174）给服务器。我们也看到一个交互用户命令（dir）成为两个FTP命令（PORT和LIST）。

图27-6是控制连接上分组交换的时间系列（已除去了控制连接的建立和结束，以及所有窗口大小的通知）。我们关注该图中数据连接在哪儿被打开、使用和过后的关闭。

图27-7是数据连接的时间系列。图中的起始时间与图 27-6中的相同。已除去了所有窗口大小通知，但留下服务类型字段，以说明数据连接使用另一个服务类型（最大吞吐量），而不同于控制连接（最小时延）（服务类型(TOS)值在图3-2中）。

在时间系列上，FTP服务器执行数据连接的主动打开，从端口 20（称为ftp-data）到来自PORT命令的端口号（1174）。本例中还可以看到服务器在哪儿向数据连接上执行写操作，服务器对数据连接执行主动的关闭，这就告诉客户列表已完成。

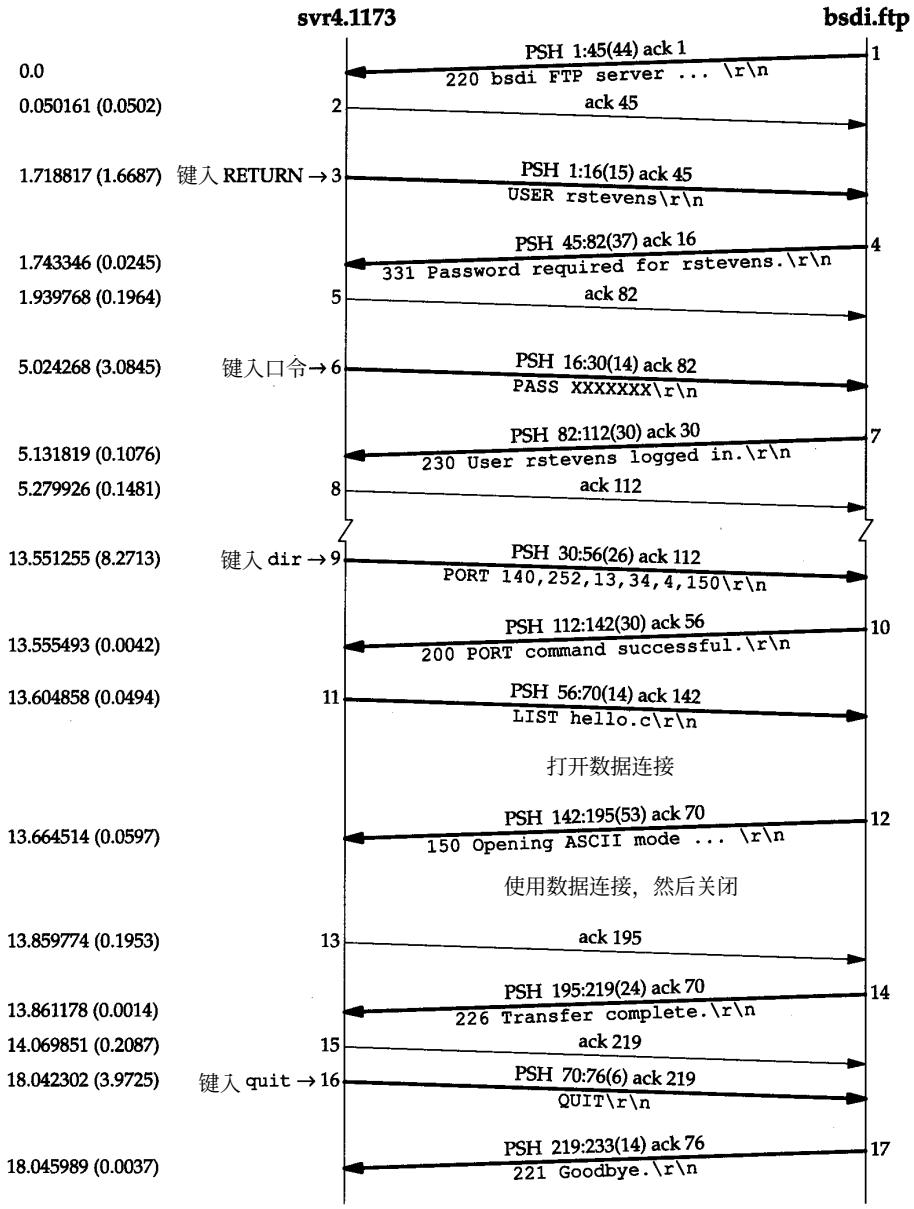

图27-6 FTP控制连接示例

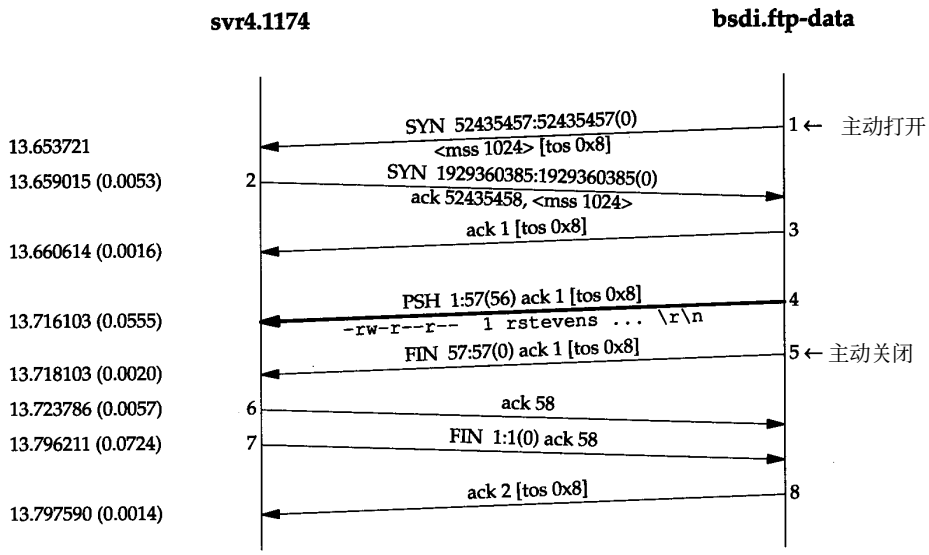

图27-7 FTP数据连接示例

27.3.2 连接管理：默认数据端口

如果客户没有向服务器发出PORT命令，来指明客户数据连接端的端口号，服务器就用与控制连接正在用的相同的端口号给数据连接。这会给使用流方式（ Unix FTP客户和服务器一直使用）的客户带来一些问题。正如下面所示：

> Host Requirements RFC建议使用流方式的FTP客户在每次使用数据连接前发一个PORT命令来启用一个非默认的端口号。

回到先前的例子（图27-6），如果我们要求在列出第1个目录后几秒钟再列出另一个目录，那该怎么办？客户将要求其内核选择另一个临时端口号（可能是 1175），下一个数据连接将建立在svr4端口1175和bsdi端口20之间。但在图27-7中服务器执行数据连接的主动打开，我们在18.6节说明了服务器将不把端口 20分配给新的数据连接，这是因为本地端口号已被更早的连接使用，而且还处于2MSL等待状态。

服务器通过指明我们在18.6节中提到的SO_REUSEADDR选项，来解决这个问题。这让它把端口20分配给新连接，而新连接将从处于 2MSL等待状态的端口（1174）处得到一个不一样的外部端口号（1175），这样一切都解决了。

如果客户不发送PORT命令，而在客户上指明一个临时端口号，那么情况将改变。我们可以通过执行用户命令sendport给FTP来使之发生。Unix FTP客户用这个命令在每个数据连接使用之前关闭向服务器发送PORT命令。

图27-8给出了用于两个连续LIST命令的数据连接时间系列。控制连接起自主机 svr4上的端口1176，所以在没有PORT命令的情况下，客户和服务器给数据连接使用相同的端口号（除去了窗口通知和服务类型值）。

事件序列如下：

1) 控制连接是建立在客户端口1176到服务器端口21上的（这里我们不展示）。

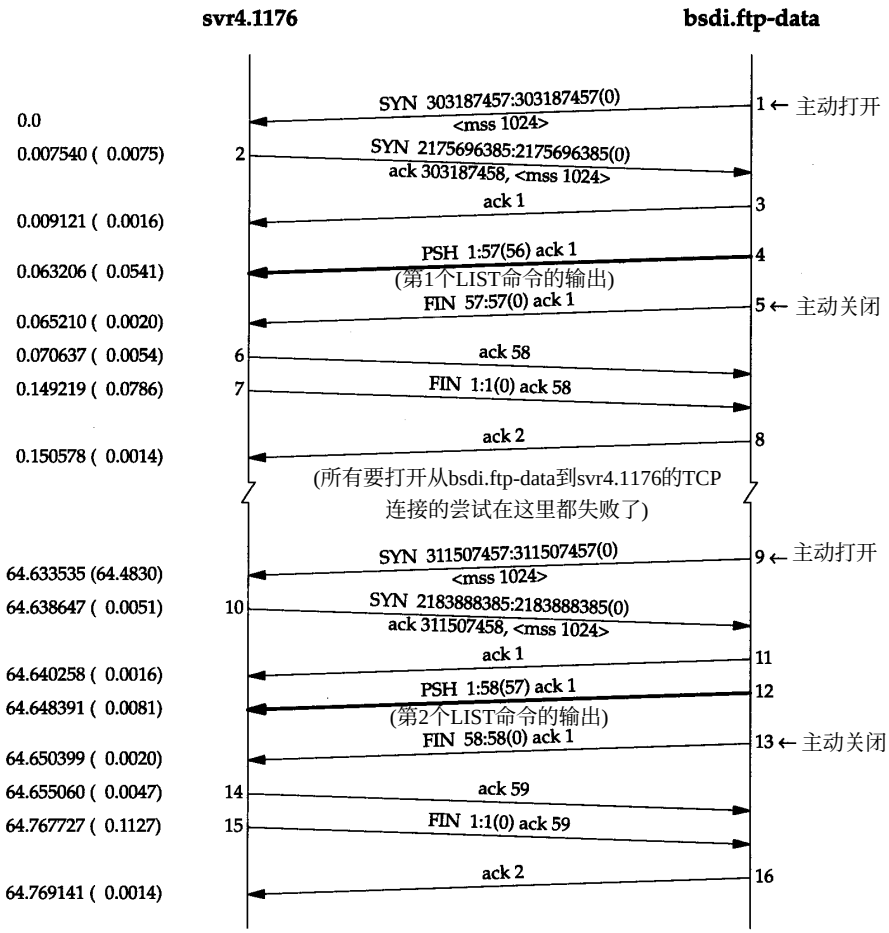

图27-8 两个连续LIST命令的数据连接

2) 当客户为端口1176上的数据连接做被动打开时，由于该端口已被客户上的控制连接使用，所以必须确定SO_REUSEADDR选项。

3) 服务器给端口20到端口1176的数据连接（报文段1）做主动打开。即便端口1176已在客户上被使用，客户仍会接受它（报文段2），这是因为下面这一对插口是不同的：

```
<svr4, 1176, bsdi, 21>
<svr4, 1176, bsdi, 20>
```

（在bsdi上的端口号是不同的）。TCP通过查看源IP地址、源端口号、目的IP地址、目的端口号分用各呼入报文段，只要这4个元素中的一个不同，就行。

4) 服务器对数据连接（报文段 5）做主动的关闭，即把这对插口置入服务器上的一个2MSL等待。

```
<svr4, 1176, bsdi, 20>
```

5) 客户在控制连接上发送另一个 LIST命令（这里我们不展示）。在此之前，客户在端口1176上为其数据连接端做一个被动打开。客户必须再一次指明 SO_REUSEADDR，这是因为端口号1176已在使用。

6) 服务器给从端口20到端口1176的数据连接发出一个主动打开。在此之前，服务器必须指明SO_REUSEADDR，这是因为本地端口（20）与处于2MSL等待状态的连接是相关联的，但从18.6节所示可知，该连接将不成功。其原由是这个连接用插口对 (socket pair)与步骤4中的仍处于2MSL等待状态的插口对相同。TCP规定禁止服务器发送同步信息（SYN）。这样就没办法让服务器跨过插口对的2MSL等待状态来重用相同的插口对。

在这一步伯克利软件分发（BSD）服务器每隔5秒就重试一次连接请求，直到满18次，总共90秒。我们看到报文段9将在大约1分钟后成功（我们在第18章提到过，SVR4使用一个30秒的MSL，以两个MSL来达到持续1分钟的等待）。我们没看到在这个时间系列上的这些失败有任何同步（SYN）信息，这是因为主动打开失败，服务器的TCP不再发送一个SYN。

Host Requirements RFC建议使用PORT命令的原因是在两个相继使用数据连接之间避免出现这个2MSL。通过不停地改变某一端的端口号，我们所说的这个问题就不会出现。

27.3.3 文本文件传输：NVT ASCII表示还是图像表示

让我们查证一下默认的文本文件传输使用 NVT ASCII码。这次不指定 -d标志，所以不看客户命令，但注意到客户还将打印服务器的响应：

```
sun % ftp bsdi
Connected to bsdi.
220 bsdi FTP server (Version 5.60) ready.
Name (bsdi:retevens);                                    键入RETURN
331 Password required for rstevens.
Passord :                                                键入口令
230 User rstevens logged in.
ftp> get hello.c                                         取一个文件
200 PORT command successful.
150 Opening ASCII mode data connection for hello.c (38 bytes).
226 Transfer complete.                                   服务器说明文件含有38字节
local: hello.c remote: hello.c                           由客户输出
42 bytes received in 0.0037 seconds (11 Kbytes/s)        字节传过数据连接
ftp> quit
221 Goodbye.
Sun % ls -l hello.c
-rw-rw-r--   1 rstevens     38 Jul 18   08:48 hello.c   但文件还含有38字节
sun % wc -l hello.c                                      在文件中记行数
    4 hello.c
```

因为文件有4行，所以从数据连接上传输42个字节。Unix下的每一新行符（\n）被服务器转换成NVT ASCII码的2字节行结尾序列（\r\n）来传输，然后再由客户转换成原先形式来存储。

新客户试图确定服务器是否是相同类型的系统，一旦相同，就可以用二进制码（图像文件类型）来传输文件，而不是ASCII码。这可以获得两个方面的好处：

1) 发方和收方不必查看每一字节（很大的节约）。

2) 如果主机操作系统使用比2字节的NVT ASCII码序列更少的字节来作行尾，就会传输更少的字节数（很小的节约）。

我们可以看到使用一个BSD/386客户和服务器的最优效果。启动排错（ debug）方式来看

客户FTP命令：

```
bsdi &ftp -d slip                                   指明 -d来看客户命令
Connected to slip.
220 slip FTP server (Version 5.60) ready.
Name (slip:rstevens):                               我们键入RETURN
---> USER rstevns
331 Password required for rstevens.
Password :                                          我们键入自己的口令
---> PASS XXXX
230 User rstevns logged in .
---> SYST                                           这由客户服务器的应答自动发送
215 UNIX Type: L8 Version : BSD-199103
Remote system type is UNIX.                         由客户发出的信息
Using binary mode to transfer files.                由客户发出的信息
ftp> get hello.c                                    取一个文件
---> TYPE I                                         由客户自动发送
200 Type set to I.
---> PORT  140,252,13,66,4,84                       端口号=4×256+84=1108
200 PORT command successful.
---> RETR hello.c
150 Opening BINARY mode data connection for hello.c (38 bytes) .
226 Transefer complete .
38 bytes received in 0.035 seconds (1.1 Kbytes/这时只有38个字节
ftp> quit
---> QUIT
221 Goodbye.
```

注册到服务器后，客户FTP自动发出SYST命令，服务器将用自己的系统类型来响应。如果应答起自字符串“ 215 UNIX Type : L8 ”，并且如果客户在每字节为8 bit的Unix系统上运行，那么二进制方式（图像）将被所有文件传输所使用，除非被用户改变。

当我们取文件hello.c时，客户自动发出命令TYPE I把文件类型定成图像。这样在数据连接上只有38字节被传输。

Host Requirements RFC指出一个FTP服务器必须支持SYST命令（这曾是RFC 959中的一个选项）。但支持它的使用文本的系统（见封 2）仅仅是BSD/386和AIX 3.2.2。SunOS 4.1.3和Solaris 2.x 用500（不能理解的命令）来应答。SVR4采用极不大众化的应答行为500，并关闭控制连接！

27.3.4 异常中止一个文件的传输：Telnet 同步信号

现在看一下FTP客户是怎样异常中止一个来自服务器的文件传输。异常中止从客户传向服务器的文件很容易 —— 只要客户停止在数据连接上发送数据，并发出 ABOR命令到控制连接上的服务器即可。而异常中止接收就复杂多了，这是因为客户要告知服务器立即停止发送数据。我们前面提到要使用Telnet同步信号，下面的例子就是这样。

我们先发起一个接收，并在它开始后键入中断键。这里是交互会话，其中初始注册被略去：

```
ftp> get a.out                                  取一个大文件
---> TYPE I                                     客户和服务器都是8 bit字节的Unix系统
200 Type set to I.
---> PORT 140,252,13,66,4,99
200 PORT command successful.
---> RETR a.out
150 Opening BINARY mode data connection for a.out (28672 bytes).
^?                                              键入的中断键
receive aborted                                 由客户输出
waiting for remote to finish abort              由客户输出
426 Transfer aborted. Data connection closed.
226 Abort successful
1536 bytes received in 1.7 seconds (0.89 Kbytes/s)
```

在我们键入中断键之后，客户立即告知我们它将发起异常中止，并正在等待服务器完成。服务器发出两个应答：426和226。这两个应答都是由Unix 服务器在收到来自客户的紧急数据和ABOR命令时发出的。

图27-9和图27-10展示了会话时间系列。我们已把控制连接（实线）和数据连接（虚线）合在一起来说明它们之间的关系。

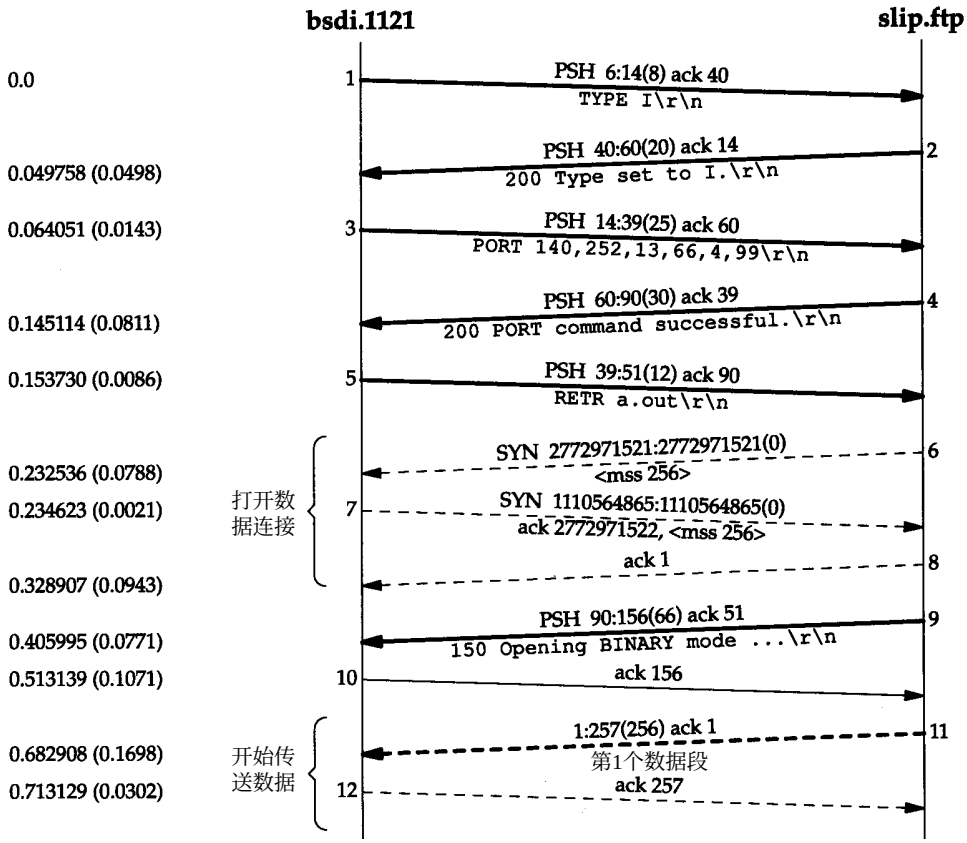

图27-9 异常中止一个文件的传输（前半部）

图27-9的前面12个报文段是我们所期望的。通过控制连接的命令和应答建立起文件传输，数据连接被打开，第1个报文段的数据从服务器发往客户。

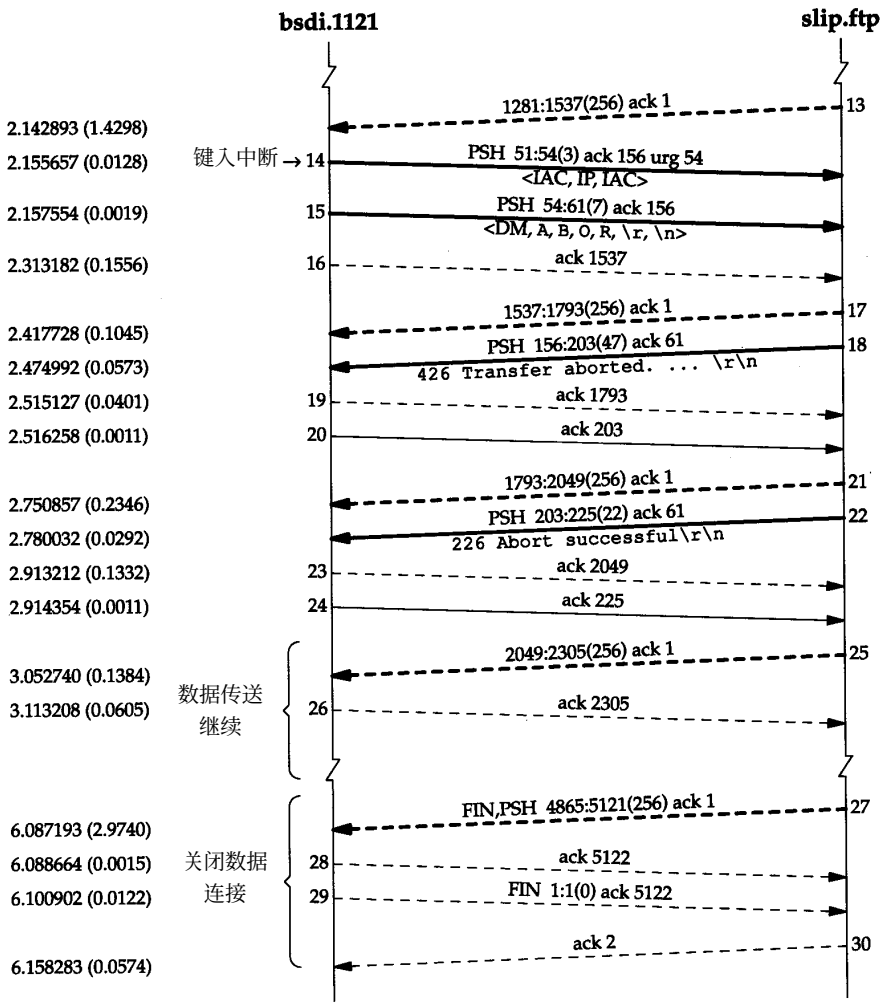

图27-10 异常中止一个文件的传输（后半部）

在图27-10中，报文段13是数据连接上来自服务器的第 6个数据报文段，后跟由我们键入的中断键产生的报文段14。客户发出10个字节来异常中止传输：

```
<IAC,IP,IAC,DM,A,B,O,R,\r,\n>
```

由于20.8节中详细讨论过这个问题，我们看到有两个报文段（ 14和15）涉及到TCP的紧急指针（我们在图26-17中看过对Telnet问题也做相同的处理）。Host Requirements RFC指出紧急指针应指向紧急数据的最后一个字节，而多数伯克利的派生实现使之指向紧急数据最后一个字节后面的第一个字节。了解到紧急指针将（错误地）指向下一个要写的字节（数据标志，DM。在序号为54处），FTP客户进程特意写前3个字节作为紧急数据。首先写下的 3字节紧急数据与紧急指针一起被立即发送，紧接着是后面 7个字节（BSD FTP 服务器不会出现由客户使用的紧急指针的解释问题。当服务器收到控制连接上的紧急数据时，它读下一个 FTP命令，寻找ABOR或STAT，忽略嵌入的Telnet命令）。

注意到尽管服务器指出传输被异常中止（报文段 18，在控制连接上），客户进程还要在数据连接上再接收14个报文段的数据（序列号是 1537~5120）。这些报文段可能在收到异常中止

时，还在服务器上的网络设备驱动器中排队，但客户打印“收到 1536字节”，意思是在发出异常中止后（报文段14和15），略去收到的所有数据报文段。

一旦Telnet用户键入中断键，我们在图 26-17中看到Unix客户在默认情况下不发出中断进程命令作为紧急数据。因为几乎没有机会用流控制来中止从客户进程到服务器进程的数据流，所以我们说这样就行了。FTP的客户进程也通过控制连接发送一个中断进程命令，因为两个连接正在被使用，因此没有机会用流控制来中止控制连接。为什么 FTP发送中断进程命令作为紧急数据而Telnet不呢？答案在于FTP使用两个连接，而 Telnet只使用一个，在某些操作系统上要求一个进程同时监控两个连接的输入是困难的。 FTP假设这些临界的操作系统至少提供紧急数据在控制连接上已到达的通知，而后让服务器从处理数据连接切换到控制连接上来。

27.3.5 匿名FTP

FTP的一种形式很常用，我们下面给出它的例子。它被称为匿名 FTP，当有服务器支持时，允许任何人注册并使用FTP来传输文件。使用这个技术可以提供大量的自由信息。

怎样找出你正在搜寻的站点是一个完全不同的问题。我们将在 30.4节简要介绍。

我们将把匿名FTP用在站点`ftp.uu.net`上（一个常用的匿名FTP站点）来取本书的勘误表文件。要使用匿名FTP，须使用“anonymous”（复习数遍就能正确地拼写）用户名来注册。当提示输入口令时，我们键入自己的电子邮箱地址。

```
sun % ftp ftp.uu.net
Connected to ftp.uu.net
220 ftp.UU.NET FTP server (Version 2.0WU(13) Fri Apr 9 20:44:32 EDT 1993) ready
Name (ftp.uu.net:rstevens): anonymous
331 Guest login ok, send your complete e-mail addraess as password.
Password :                                  键入rstevens@noao.edu；它没有回显
230-
230-                    Welcome to the UUNET archive.
230-    A service of UUNET Technologies Inc, Falls Church, Virginia
230-    For information about UUNET, call +1 703 204 8000, or see the files
230-    in /uunet-info
                                            还有一些问候行
230 Guest login ok, access restrictions apply.
ftp> cd published/books                     换成需要的目录
250 CWD command successful.
ftp> binary                                 我们将传送一个二进制文件
200 Type set to I.
ftp> get stevens.tcpipivl.errata.Z          取文件
200 PORT command successful.
150 Opening BINARY mode data connection for stevens.tcpipivl.errata.Z (150 bytes).
226 Transfer complete.                      (你可能得到一个不同的文件大小)
local: stevens.tcpipivl.errata.Z remote: stevens.tcpipivl.errata.Z
105 bytes received in 4.1 seconds (0.83 Kbytes/s)
ftp> quit
221 Goodbye.
sun % uncompress stevens.tcpipivl.errata.Z
sun % more stevens.tcpipivl.errata
```

不压缩是因为很多现行匿名FTP文件是用Unix compress(1)程序压缩的，这样导致文件带有.Z的扩展名。这些文件必须使用二进制文件类型来传输，而不是ASCII码文件类型。

27.3.6 来自一个未知IP地址的匿名FTP

可以把一些使用匿名FTP的域名系统（DNS）特征和选路特征结合在一起。在14.5节中我们谈到DNS中指针查询现象——取一个IP地址并返回其主机名。不幸的是并非所有系统管理员都能正确地创立涉及指针查询的名服务器。他们经常记得把新主机加入名字到地址匹配的文件中，却忘了把他们加入到地址到名字匹配的文件中。对此，可用traceroute经常看到这种现象，即它打印一个IP地址，而不是主机名。

有些匿名FTP服务器要求客户有一个有效域名。这就允许服务器来记录正在执行传输的主机域名。由于服务器在来自客户IP数据报中收到的关于客户的唯一标识是客户的IP地址，所以服务器能叫DNS来做指针查询，并获得客户的域名。如果负责客户主机的名服务器没有正确地创立，指针查询将失败。

要看清这个错误，我们来做以下诸步骤：

1) 把主机slip（见封2的图）的IP地址换成140.252.13.67。这是给作者子网的一个有效IP地址，但没有涉及到noao.edu域的域名服务器。

2) 把在bsdi上SLIP连接的目的IP地址换成140.252.13.67。

3) 把将数据报引向140.252.13.67的sun上的路由表入口加入路由器bsdi（回忆一下我们在9.2节中关于这个选路表的讨论）。

从Internet上仍然可以访问我们的主机slip，这是因为在10.4节中路由器gateway和netb正好把所有目的是子网140.252.13的所有数据报都发送给路由器sun。路由器sun知道利用我们在上述第3步建立的路由表入口来如何处理这些数据报。我们所创建的是拥有完整Internet连接性的主机，但没有有效的域名。结果，指针查询IP地址140.252.13.67将失败。

现在给一个我们所知的服务器使用匿名FTP，需要一个有效的域名：

```
slip % ftp ftp.uu.net
Connected to ftp.uu.net.

220 ftp.UU.NET FTP server (Version 2.0WU(13) Fri Apr 9 20:44:32 EDT 1993) ready.
Name (ftp.uu.net:rstevens): anonymous

530- Sorry, we're unable to map your IP address 140.252.13.67 to a hostname
530- in the DNS.  This is probably because your nameserver does not have a
530- PTR record for your address in its tables, or because your reverse
530- nameservers are not registered.  We refuse service to hosts whose
530- names we cannot resolve.  If this is simply because your nameserver is
530- hard to reach or slow to respond then try again in a minute or so, and
530- perhaps our nameserver will have your hostname in its cache by then.
530- If not, try reaching us from a host that is in the DNS or have your
530- system administrator fix your servers.
530 User anonymous access denied..

Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.

ftp> quit
221 Goodbye.
```

来自服务器的出错应答是无需加以说明的。

27.4 小结

FTP是文件传输的Internet标准。与多数其他TCP应用不同，它在客户进程和服务器进程之间使用两个TCP连接——一个控制连接，它一直持续到客户进程与服务器进程之间的会话完成为止；另一个按需可以随时创建和撤消的数据连接。

FTP使用的关于数据连接的连接管理让我们更详细地了解 TCP连接管理需求。我们看到TCP在不发出PORT命令的客户进程上对2MSL等待状态的作用。

FTP使用NVT ASCII码做跨越控制连接的所有远程登录命令和应答。数据传输的默认方式通常也是NVT ASCII码。我们看到较新的Unix客户进程会自动发送命令来查看服务器是否是8 bit字节的Unix主机，并且如果是，那么就使用二进制方式来传输所有文件，那将带来更高的效率。

我们也展示了匿名FTP的一个例子，它是在Internet上分发软件的常用形式。

习题

27.1 图27-8中，如果客户对第2个数据连接做一次主动打开，而不是由服务器来做，那将发生什么变化？

27.2 在本章FTP客户例子中，我们加入诸如由客户输出行的行注释。如果不看源代码，我们如何确定这些不是来自服务器？

```
local: hello.c remote: hello.c
42 bytes received in 0.0037 seconds (11 Kbytes/s)
```