# *TFTP*协议 （RFC 1350）

### 文档的现状

这个*RFC*文档被网络协会列为准*IAB*标准，需要进一步的讨论和修改。通过*IAB*标准可以查看这个协议的状态。可以任意的发布本协议。

### 概况

*TFTP*是一个传输文件的简单协议*,*可以从它的名字看出。 每个非结尾数据报被单独的确认。 本文档描述*TFTP*的协议和种类。 也解释一些设计*TFTP*协议的原因。

### 背景

这个协议原本由*Noel Chiappa*设计， 由*Noel Chiappa*、*Bob Baldwin*和*Dave Clark*重新设计， 并由*Steve Szymanski*校验。现在的这个版本，*Larry Allen, Noel Chiappa, Dave Clark, Geoff Cooper, Mike Greenwald, Liza Martin, David Reed, Craig Milo Rogers, Kathy Yellick*和原本作者进行了修订。转发和确认机制受到*TCP*的影响， 错误机制受*PARC*的*EFTP*异常消息。

*1992*年*5*月，重新修订了本协议的*BUG*，其他的小问题由*Noel Chiappa*来做。

*(*美国）国防高级研究计划局支持此协议的研究， 由海军研究办公室监制 号码为：*N00014-75-C-0661*。

## 1. 目的

TFTP是一个传输文件的简单协议，它已经基于UDP协议实现了,因此可以在不同网络的主机间进行传送。此协议设计的时候是进行小文件传输的。因此它容易实现，不具备通常的FTP的许多功能，它只能从文件服务器上获得或写入文件，不能列出目录，不进行认证，它传输8位数据。

传输中有三种模式：netascii，这是8位的ASCII码形式，另一种是octet，这是8位源数据类型；最后一种mail已经不再支持，它将返回的数据直接返回给用户而不是保存为文件。可以由协作的主机间自己定义模式。

参考[4]可以总结出更多的建议。

## 2. 概况

任何传输起自一个读取或写入文件的请求，这个请求也是连接请求。如果服务器批准此请求，连接就建立了，数据以定长512字节传输。每个数据包包括一块数据，发出下一个数据包以前必须得到对上一个数据包的确认。如果一个数据包的大小小于512字节，则表示传输结束。如果数据包在传输过程中丢失，发出方会在超时后重新传输最后一个未被确认的数据包。通信的双方都是数据的发出者与接收者，一方传输数据接收应答，另一方发出应答接收数据。

大部分的错误会导致连接中断，错误由一个错误的数据包引起。这个包不会被确认，也不会被重新发送，因此另一方无法接收到, 这种连接中断可以由超时机制来检查。错误主要是由下面三种情况引起的：不能满足请求；收到的数据包内容错误，而这种错误不能由延时或重发解释；对需要资源的访问丢失（如硬盘满）。TFTP只在一种作物情况下不中断连接，这种情况是源端口不正确，在这种情况下，指示错误的包会被发送到源机。这个协议限制很多，这是都是为了实现起来比较方便而进行的。例如， 固定的长度方便本地存储， 对每一个数据报进行确认方便流式控制和对接受的数据包进行排序。

## *3.*与其它协议的联系

因为TFTP使用UDP，而UDP使用IP。因此一个TFTP包中会有以下几段：IP头，UDP报头，TFTP头，剩下的就是TFTP数据了。此外， 数据报还需要像LNI， ARPA的头部以便在在本地传输。如图3-1所示。TFTP在IP头中不指定任何数据，但是它使用UDP中的源和目标端口以及包长度域。由TFTP使用的包标记（TID）在数据包层被用做端口，因此TID必须介于0到65,535之间。对它的初始化我们在后面讨论。TFTP头中包括两上字节的操作码，操作码指出了包的类型, 操作码和其他类型的操作码我们在后面的章节中进行讨论。

-----------------------------------------------------------

| Local Medium | Internet | Datagram | TFTP |

-------------------------------------------------------------------------------

图*3-1:*包头次序

## *4.*初始连接

初始连接时候需要发出WRQ（请求写入远程系统）或RRQ（请求读取远程系统），收到一个确定应答，是确定可以写出的包或应该读取的第一块数据。通常确认包包括要确认的包的包号，每个数据包都与一个块号相对应，块号从1开始而且是连续的。因此对于写入请求的确定是一个比较特殊的情况，因此它的包的包号是0。如果收到的包是一个错误的包，则这个请求被拒绝。创建连接时，通信双方随机选择一个TID，是随机选择的，因此两次选择同一个TID的可能性就很小了。每个包包括两个TID，发送者TID和接收者TID。这些TID用于在UDP(或其他数据包协议)通信时选择端口，请求主机选择TID的方法上面已经说过了，在第一次请求的时候它会将请求发到TID 69，也就是服务器的69端口上。应答时，服务器使用一个选择好的TID作为源TID，并用上一个包中的TID作为目的ID进行发送。这两个被选择的TID在随后的通信中会被一直使用。下例是一个写入的例子，其中WRQ，ACK和DATA代表写入请求，确认和数据。

1. 主机A向主机B发出WRQ，其中端口为69

2. B机向A机发出ACK，块号为0，包括B和A的TID

此时连接建立，第一个数据包以序列号1从主机开始发出。以后两台主机要保证以开始时确定的TID进行通信。如果源ID与原来确定的ID不一样，这个包会被认识为发送到了错误的地址而被抛弃。错误消息包是被发送到发送不正确包的端口，因而不影响继续传输文件，这种情况仅适合接受到包端口不正确。如果支持协议不允许，这种情况就不会发生。

关于上述情况，下面这个例子演示了一个正确的协议操作。设想发送方发出一个请求，这个请求在网络的那个设备中被复制成两个包，接收方先后接收到两个包。接收方会认为为这是两个独立的请求，会返回两个应答。当这两个应答其中之一被接收到时，连接已经建立。第二个应答再到达时，这个包会被抛弃，而不会因为接收到第二个应答包而导致第一个建立的连接失败。

## 5. TFTP包(作到这里了)

TFTP支持五种类型的包，在上面都涉及到了：

opcode operation

1 Read request (RRQ)

2 Write request (WRQ)

3 Data (DATA)

4 Acknowledgment (ACK)

5 Error (ERROR)

包头中包括了这个包所指定的操作码。

2 bytes string 1 byte string 1 byte

------------------------------------------------

| Opcode | Filename | 0 | Mode | 0 |

------------------------------------------------

Figure 5-1: RRQ/WRQ包

RRQ和WRQ包（代码分别为1和2）的格式如上所示。文件名是NETASCII码字符，以0结束。 而MODE域包括了字符串"netascii"，"octet"或"mail"，名称不分大小写。接收到NETASCII格式数据的主机必须将数据转换为本地格式。OCTET模式用于传输在源机上以8位格式存储的文件。假设每个机器都存在一个8位的格式，这样的假设是最一般的。比如DEC-20，这是一种36位机，我们可以假设它是4个8位外加另外4位而构成。如果机器收到OCTET格式文件，返回时必须与原来文件完全一样。在使用MAIL模式时，用户可以在FILE处使用接收人地址，这个地址可以是用户名或用户名@主机的形式，如果是后一种形式，允许主机使用电子邮件传输此文件。如果使用MAIL类型，包必须以WRQ开始，否则它与NETASCII完全一样。

我们的讨论建立在发送方和接收方都在相同模式的情况下，但是双方可以以不同的模式进行传输。例如一个机器可以是一台存储服务器，这样一台服务器需要将NETASCII格式转换为自己的格式。另外，我们可以设想DEC-20这种机器，它使用36位字长，用户这边可以使用特殊的机制一次读取36位，而服务器却可以仍然使用8位格式。在这两种情况下，我们看到了两台机器使用不同格式的情况。可以在两台主机间定义其它的传输方式，但是定义要小心，因为这种传输方式不为人知，而且也没有权威机构为其指定名称或定义它的模式。

*2 bytes 2 bytes n bytes  
----------------------------------  
| Opcode | Block # | Data |  
----------------------------------  
Figure 5-2: DATA*包

数据在数据包中传输，其格式如上图所示。数据包的*OP*码为*3*，它还包括有一个数据块号和数据。数据块号域从*1*开始编码，每个数据块加*1*，这样接收方可以确定这个包是新数据还是已经接收过的数据。数据域从*0*字节到*512*字节。如果数据域是*512*字节则它不是最后一个包，如果小于*512*字节则表示这个包是最后一个包。除了*ACK*和用于中断的包外，其它的包均得到确认。发出新的数据包等于确认上次的*ACK*包。*WRQ*和*DATA*包由*ACK*或*ERROR*数据包确认，而*RRQ*和数据包由*DATA*或*ERROR*数据包确认。下图即是一个*ACK*包，操作码为*4*。其中的包号为要确认的数据包的包号。

*2 bytes 2 bytes  
---------------------  
| Opcode | Block # |  
---------------------  
Figure 5-3: ACK*包

*WRQ*数据包被*ACK*数据包确认，*WRQ*数据包的包号为*0*。

*2 bytes 2 bytes string 1 byte  
-----------------------------------------  
| Opcode | ErrorCode | ErrMsg | 0 |  
-----------------------------------------  
Figure 5-4: ERROR*包

一个*ERROR*包，它的操作码是*5*，它的格式如上所示。此包可以被其它任何类型的包确认。错误码指定错误的类型。错误的值和错误的意义在附录中。错误信息是供程序员使用的*,*应该是*netascii*类型。像其他的字符， 它的中断连接标志是*srting*域为*0*。

## *6.*正常终止

传输的结束由*DATA*数据标记，数据区为*0-511*个字符。这个包可以被*ACK*数据包确认。接收方在发出对最后数据包的确认后可以断开连接，当然，适当的等待是比较好的，如果最后的确定包丢失可以再次传输。如果发出确认后仍然收到最后数据包，可以确定最后的确认丢失。发送最后一个*DATA*包的主机必须等待对此包的确认或超时。如果响应是*ACK*，传输完成。如果发送方超时并不准备重新发送并且接收方有问题或网络有问题时，发送也正常结束。也有可能这种情况传输是不成功的，但无论如何连接都将被关闭。

## 7. 早终结

如果请求不能被满足，或者在传输中发生错误，然后发送了*ERROR*包。这仅是一种传输友好的方式，这种包不会被确认也不会被重新传输，因此这种包可能永远不会被接收到。因此需要用超时来侦测错误。

*I.*附录

包头的次序

*2 bytes  
----------------------------------------------------------  
| Local Medium | Internet | Datagram | TFTP Opcode |  
----------------------------------------------------------*

*TFTP*格式

*Type Op #*没有包头的格式

*2 bytes string 1 byte string 1 byte  
-----------------------------------------------  
RRQ/ | 01/02 | Filename | 0 | Mode | 0 |  
WRQ -----------------------------------------------*

*2 bytes 2 bytes n bytes  
---------------------------------  
DATA | 03 | Block # | Data |  
---------------------------------*

*2 bytes 2 bytes  
-------------------  
ACK | 04 | Block # |  
--------------------*

*2 bytes 2 bytes string 1 byte  
----------------------------------------  
ERROR | 05 | ErrorCode | ErrMsg | 0 |  
----------------------------------------*

读文件的初始连接

*1.*主机*A*发*RRQ*到*A*，包括源*=A*的*ID*和目的*=69  
2.*主机*B*发送*DATA*，其中包号*=1*，这个包被传送到*A*，携带源*B*的*TID*，目的*A*的*TID*

错误码

*Value   Meaning  
  
0*未定义，请参阅错误信息（如果提示这种信息的话） *1*文件未找到 *2*访问非法 *3*磁盘满或超过分配的配额 *4*非法的*TFTP*操作 *5*未知的传输*ID  
6*文件已经存在 *7*没有类似的用户

*Internet*用户数据报头

（这个只是提供便利，*TFTP*不一定非要在*UDP*上实现。）

*Format*

*0 1 2 3  
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
| Source Port | Destination Port |  
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
| Length | Checksum |  
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+*

域的值

*Source Port*由传输发起方选择 *Dest. Port*由目的地选择（如果是*RRQ*或*WRQ*，其值为*69*） *Length*包括*UDP*包头的包长度 *Checksum*校验码，如果是*0*，则未使用校验。参考*2*描述了计算校验码的规则

注意：*TFTP*将传输标记*TID*传送给*UDP*作为源和目的端口

参考文献

*[1] USA Standard Code for Information Interchange, USASI X3.4 – 1968.*

*[2] Postel, J.,*“*Telnet Protocol Specification*”*RFC 764, USC/Information Sciences Institute, June, 1980.*

*[3] Braden, R., Editor,*“*Requirements for Internet Hosts – Application and Support*”*, RFC 1123, USC/Information Sciences Institute, October 1989*

安全问题

因为*TFTP*没有监权控制机制，因此服务器安全问题应该多加考虑。通常*TFTP*允许下载数据而不允许上传数据。

本文档作者的联系方式

*Karen R. Sollins*

*Massachusetts Institute of Technology*

*Laboratory for Computer Science*

*545 Technology Square*

*Cambridge, MA 02139-1986*

*Phone: (617) 253-6006*

*Email: SOLLINS@LCS.MIT.EDU*