IEN 133

The TFTP Protocol

January 29, 1980

Karen R. Sollins

Summary

TFTP is a very simple protocol used to transfer files. It is from this that its name comes, Trivial File Transfer Protocol or TFTP. Each nonterminal packet is acknowledged separately. This document describes the protocol and its types of packets. The document also explains the reasons behind some of the design decisions.

Acknowledgements

The protocol was originally designed by Noel Chiappa, and was redesigned by him, Bob Baldwin and Dave Clark, with comments from Steve Szymanski. The original version of this document was written by Bob Baldwin. The current version of the document includes modifications suggested by Noel Chiappa, Dave Clark, Liza Martin and the author. The acknowledgement and retransmission scheme was inspired by TCP, and the error mechanism was suggested by PARC's EFTP abort message.

1: Purpose

TFTP is a simple protocol to transfer files, and therefore was named the Trivial File Transfer Protocol or TFTP. It is built on top of the Internet User Datagram protocol (UDP or Datagram) [2] so it may be used to move files between machines on different networks. It is designed to be small and easy to implement. Therefore, it lacks most of the features of a regular FTP. The only thing it can do is read and write files (or mail) from/to a remote server. It cannot list directories, and currently has no provisions for user authentication. In common with other Internet protocols, it passes 8 bit bytes of data.

Three modes of transfer are currently supported: netascii (1); binary, raw 8 bit bytes; mail, netascii characters sent to a user rather than a file. Additional modes can be defined by pairs of cooperating hosts.

2: Overview of the Protocol

Any transfer begins with a request to read or write a file, which also serves to request a connection. If the server grants the request, the connection is opened and the file is sent in fixed length blocks of 512 bytes. Each data packet contains one block of data, and must be

---

(1) This is ascii as defined in "USA Standard Code for Information Interchange" [1] with the modifications specified in "Telnet Protocol Specification" [3]. Note that it is 8 bit ascii. The term "netascii" will be used throughout this document to mean this particular version of ascii.

acknowledged by an acknowledgment packet before the next packet can be sent. A packet of less than 512 bytes signals termination of a transfer. If a packet gets lost in the network, the intended recipient will timeout and may retransmit his last packet (which may be data or an acknowledgment), thus causing the sender of the lost packet to retransmit that lost packet. The sender has to keep just one packet on hand for retransmission, since the lock step acknowledgment guarantees that all older packets have been received. Notice that both machines involved in a transfer are considered senders and receivers. One sends data and receives acknowledgments, the other sends acknowledgments and receives data.

Most errors cause termination of the connection. An error is signalled by sending an error packet. This packet is not acknowledged, and not retransmitted (i.e., a TFTP server or user may terminate after sending an error message), so the other end of the connection may not get it. Therefore timeouts are used to detect such a termination when the error packet has been lost. Errors are caused by three types of events: not being able to satisfy the request (e.g., file not found, or access violation), receiving a packet which cannot be explained by a delay or duplication in the network (e.g. an incorrectly formed packet), and losing access to a necessary resource (e.g., disc full, or source file truncated during transfer).

TFTP recognizes only one type of error that does not cause termination, the source port of a received packet being incorrect. In

this case an error packet is sent to the originating host. See the section on the Initial Connection Protocol for more details.

This protocol is very restrictive, but that makes it easier to implement. For example, the fixed length blocks make allocation straight forward, and the lock step acknowledgement provides flow control and eliminates the need to reassemble files.

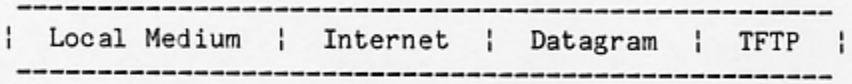
3: Relation to other Protocols

As mentioned TFTP is designed to be implemented on top of the Datagram protocol. Since Datagram is implemented on the Internet protocol, packets will have an Internet header, a Datagram header, and a TFTP header. Additionally, the packets may have a header (LNI, ARPA header, etc.) to allow them through the local transport medium. As shown in Figure 1, the order of the contents of a packet will be local medium header, if used, Internet header, Datagram header, TFTP header, followed by the remainder of the TFTP packet. (This may or may not be data depending on the type of packet as specified in the TFTP header.) TFTP does not specify any of the values in the Internet header.

The source and destination port fields of the Datagram header (its format is given in the appendix) are used by TFTP and the length field reflects the size of the TFTP packet. The transfer identifiers (TID's) used by TFTP are passed to the Datagram layer to be used as ports. Therefore for they must be between 0 and 65,535. The initialization of TID's is discussed in the section on initial connection protocol.

The TFTP header consists of a 2 byte opcode field which indicates the packet's type (e.g., DATA, ERROR, etc.) These opcodes and the formats of the various types of packets are discussed further in the section on TFTP packets.

Figure 1. Order of Headers



4: Initial Connection Protocol

A transfer is established by sending a request (WRQ to write onto a foreign file system, or RRQ to read from it), and receiving a positive reply, an acknowledgment packet for write, or the first data packet for read. In general an acknowledgment packet will contain the block number of the data packet being acknowledged. Each data packet has associated with it a block number; block numbers are consecutive and begin with one. Since the positive response to a write request is an acknowledgment packet, in this special case the block number will be zero. (Normally, since an acknowledgment packet is acknowledging a data packet, the acknowledgment packet will contain the block number of the data packet being acknowledged.) If the reply is an error packet, then the request is denied for the reason stated in the error packet.

In order to create a connection, TID's to be used for the duration of the connection are chosen by the two ends of that connection. The TID's chosen for a connection should be randomly chosen, so that the

probability that the same number is chosen twice in immediate succession is very low. Every packet has associated with it two TID's, the source TID and the destination TID. A requesting host chooses its source TID as described above, and sends its initial request to the known TID 69 (105 octal) on the serving host. The response to the request, under normal operation, uses a TID chosen by the server as its source TID and the TID chosen for the previous message by the requestor as its destination TID. The two chosen TID's are then used for the remainder of the transfer.

As an example, the following shows the steps used to establish a connection to write a file. Note that WRQ, ACK, and DATA are the names of the write request, acknowledgment, and data types of packets respectively. The Appendix contains a similar example for reading a file.

1. Host A sends a "WRQ" to host B with
   source= A's TID, destination= 69.
2. Host B sends a "ACK" (with block number= 0) to host A with
   source= B's TID, destination= A's TID.
3. Host A sends a "DATA" (with block number= 1) to host B with
   source= A's TID, destination= B's TID.
4. Host B sends a "ACK" (with block number= 1) to host A with
   source= B's TID, destination= A's TID.

In step three, and in all succeeding steps, the hosts should make sure that the source TID matches the value that was agreed on in step 2.

If it doesn't match, an error packet should be sent to the originator, but the connection should not be aborted. The following example demonstates the problem this and the randomly chosen TID's are trying to solve.

Host A sends a request to host B. Somewhere in the network, the request packet is duplicated, and as a result two acknowledgments are returned to host A, with different TID's chosen on host B in repsonse to the two requests. When the first response arrives, host A continues the connection. When the second response to the request arrives, it should be rejected, but there is no reason to terminate the first connection. Therefore, if different TID's are chosen on host B and host A checks the source TID's of the messages it receives, the first connection can be maintained while the second is rejected.

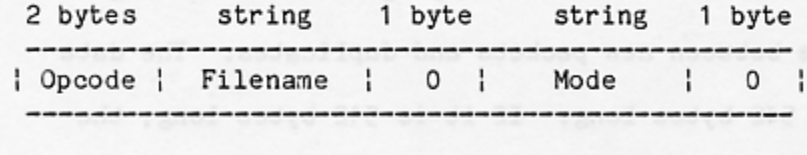
5: TFTP Packets

TFTP supports five types of packets, all of which have been mentioned above:

opcode	operation
1	Read request (RRQ)
2	Write request (WRQ)
3	Acknowledgment (ACK)
4	Data (DATA)
5	Error (ERROR)

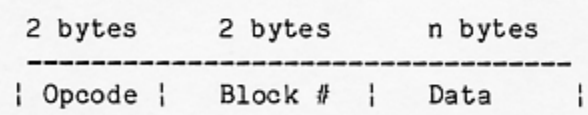
The TFTP header of a packet contains the opcode associated with that packet.

Figure 2. RRQ/WRQ



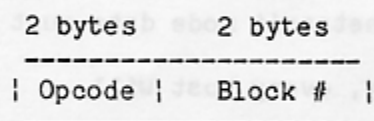
RRQ and WRQ packets (opcodes 1 and 2 respectively) have the format shown in Figure 2. The file name is a sequence of bytes in netascii terminated by a zero byte. The mode field contains the string "netascii", "binary", or "mail" in netascii indicating the three modes defined in the protocol. A host which receives netascii mode data must translate the data to its own format. Presumably, every host will translate its character set to and from netascii. Binary mode allows the two communicating hosts to impose their own interpretation on the data being transmitted; between similar machines binary mode can be used to avoid the conversion overhead. If a host receives a binary file and then returns it, the returned file must be identical to the file it received. Mail mode uses the name of a mail recipient in place of a file and must begin with a WRQ. Otherwise it is identical to netascii mode.

Figure 3. DATA



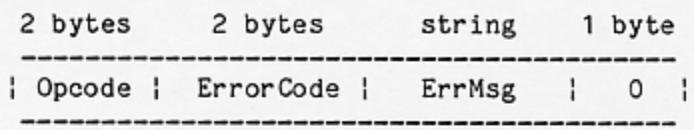
Data is actually transferred in DATA packets depicted in Figure 3. DATA packets (opcode = 4) have a block number and data field. The block numbers on data packets begin with one and increase by one for each new block of data. This restriction allows the program to use a single number to discriminate between new packets and duplicates. The data field is from zero to 512 bytes long. If it is 512 bytes long, the block is not the last block of data; if it is from zero to 511 bytes long, it signals the last data packet. (See the section on Normal Termination for details.)

Figure 4. ACK packet



All packets other than those used for termination are acknowledged individually. Sending a DATA packet is an acknowledgment for the ACK packet of the previous DATA packet. The WRQ and DATA packets are acknowledged by ACK or ERROR packets, while RRQ and ACK packets are acknowledged by DATA or ERROR packets. Figure 4 depicts an ACK packet; the opcode is 3. The block number in an ACK echoes the block number of the DATA packet being acknowledged. A WRQ is acknowledged with an ACK packet having a block number of zero.

Figure 5. ERROR packet



An ERROR packet (opcode 5) takes the form depicted in Figure 5. An ERROR packet can be the acknowledgment of any other type of packet. The error code is a small integer indicating the nature of the error. A table of its values and meanings is given in the appendix. The error message is intended for human consumption, and should be in netascii. Like all other strings, it is terminated with a zero byte.

6: Normal Termination

The end of a transfer is marked by a DATA packet that contains between 0 and 511 bytes of data (i.e. Datagram length < 516). This packet is acknowledged by an ACK packet like all other DATA packets. The final ACK packet is never retransmitted; the host acknowledging the final DATA packet may terminate its side of the connection on sending the final ACK. On the other hand, the host sending the last DATA must retransmit it until the packet is acknowledged or the sending host times out. If the response is an ACK, the transmission was completed successfully. If it is an ERROR (unknown transfer ID), or the sender of the data times out and is not prepared to retransmit any more, the transfer may still have been completed successfully, after which the acknowledger may have experienced a problem. It is also possible in

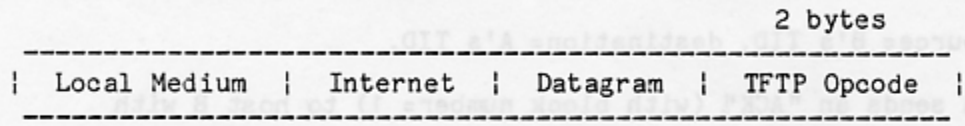
this case that the transfer was unsuccessful. In any case, the connection has been closed.

7: Premature Termination

If a request can not be granted, or some error occurs during the transfer, then an ERROR packet (opcode 5) is sent. This is only a courtesy since it will not be retransmitted or acknowledged, so it may never be received. Timeouts must also be used to detect errors.

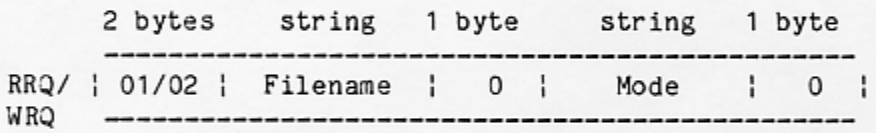
APPENDIX

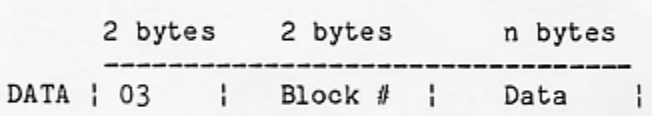
Order of Headers

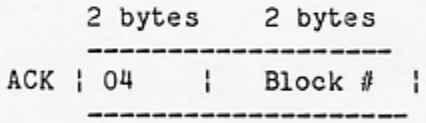


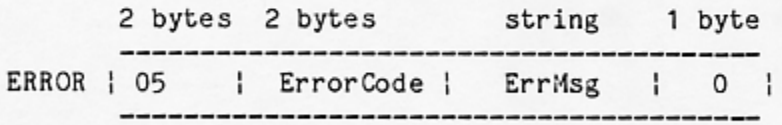
TFTP Formats

Type	Op #	Format without header









Initial Connection Protocol for reading a file

1. Host A sends a "RRQ" to host B with

   source= A's TID, destination= 69.

2. Host B sends a "DATA" (with block number= 1) to host A with

   source= B's TID, destination= A's TID.

3. Host A sends an "ACK" (with block number= 1) to host B with

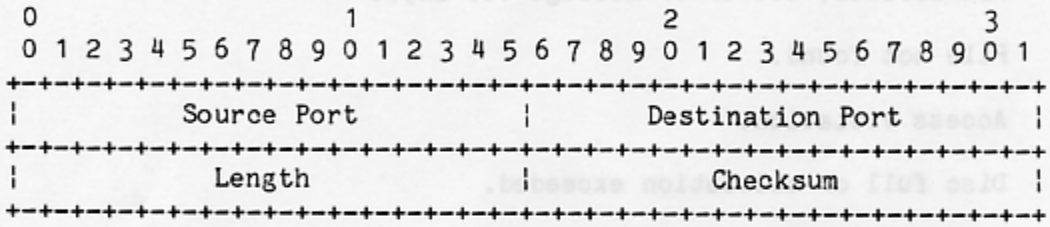
   source= A's TID, destination= B's TID.

Error Codes

Value	Meaning
0	Not defined, see error message (if any).
1	File not found.
2	Access violation.
3	Disc full or allocation exceeded.
4	Illegal TFTP operation.
5	Unknown transfer ID.

Internet User Datagram Header

Format



Values of Fields

Source Port	Picked by originator of packet.
Dest. Port	Picked by destination machine (69 for RRQ or WRQ).
Length	Number of bytes in packet after Datagram header.
Checksum	Reference 2 describes rules for computing checksum. Field contains zero if unused.

Note: TFTP passes transfer identifiers (TID's) to the Internet User Datagram protocol to be used as the source and destination ports.

References

1. USA Standard Code for Information Interchange, USASI X3.4-1968.

2. Postel, Jon., "User Datagram Protocol," IEN 88, May 2, 1979.

3. "Telnet Protocol Specification," RFC552, NIC 18639, August, 1973.