Network Working Group K. Sollins

Request For Comments: 1350 MIT

STD: 33 July 1992

Obsoletes: RFC 783

THE TFTP PROTOCOL (REVISION 2)

Status of this Memo

This RFC specifies an IAB standards track protocol for the Internet

community, and requests discussion and suggestions for improvements.

Please refer to the current edition of the "IAB Official Protocol

Standards" for the standardization state and status of this protocol.

Distribution of this memo is unlimited.

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.

Acknowlegements

The protocol was originally designed by Noel Chiappa, and was

redesigned by him, Bob Baldwin and Dave Clark, with comments from

Steve Szymanski. The current revision of the document includes

modifications stemming from discussions with and suggestions from

Larry Allen, Noel Chiappa, Dave Clark, Geoff Cooper, Mike Greenwald,

Liza Martin, David Reed, Craig Milo Rogers (of USC-ISI), Kathy

Yellick, and the author. The acknowledgement and retransmission

scheme was inspired by TCP, and the error mechanism was suggested by

PARC's EFTP abort message.

The May, 1992 revision to fix the "Sorcerer's Apprentice" protocol

bug [4] and other minor document problems was done by Noel Chiappa.

This research was supported by the Advanced Research Projects Agency

of the Department of Defense and was monitored by the Office of Naval

Research under contract number N00014-75-C-0661.

1. Purpose

TFTP is a simple protocol to transfer files, and therefore was named

the Trivial File Transfer Protocol or TFTP. It has been implemented

on top of the Internet User Datagram protocol (UDP or Datagram) [2]

Sollins [Page 1]

RFC 1350 TFTP Revision 2 July 1992

so it may be used to move files between machines on different

networks implementing UDP. (This should not exclude the possibility

of implementing TFTP on top of other datagram protocols.) 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 (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.); octet (This replaces the "binary" mode

of previous versions of this document.) raw 8 bit bytes; mail,

netascii characters sent to a user rather than a file. (The mail

mode is obsolete and should not be implemented or used.) Additional

modes can be defined by pairs of cooperating hosts.

Reference [4] (section 4.2) should be consulted for further valuable

directives and suggestions on TFTP.

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 acknowledged by an acknowledgment packet before the

next packet can be sent. A data 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

Sollins [Page 2]

RFC 1350 TFTP Revision 2 July 1992

caused by three types of events: not being able to satisfy the

request (e.g., file not found, access violation, or no such user),

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., disk full or access

denied during a transfer).

TFTP recognizes only one error condition 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.

This protocol is very restrictive, in order to simplify

implementation. For example, the fixed length blocks make allocation

straight forward, and the lock step acknowledgement provides flow

control and eliminates the need to reorder incoming data packets.

3. Relation to other Protocols

As mentioned TFTP is designed to be implemented on top of the

Datagram protocol (UDP). 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 3-1, 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. On the other hand, 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

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.

Sollins [Page 3]

RFC 1350 TFTP Revision 2 July 1992

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

| Local Medium | Internet | Datagram | TFTP |

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

Figure 3-1: 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 has been denied.

In order to create a connection, each end of the connection chooses a

TID for itself, to be used for the duration 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 the two

TID's of the ends of the connection, the source TID and the

destination TID. These TID's are handed to the supporting UDP (or

other datagram protocol) as the source and destination ports. A

requesting host chooses its source TID as described above, and sends

its initial request to the known TID 69 decimal (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.

Sollins [Page 4]

RFC 1350 TFTP Revision 2 July 1992

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.

At this point the connection has been established and the first data

packet can be sent by Host A with a sequence number of 1. In the

next step, and in all succeeding steps, the hosts should make sure

that the source TID matches the value that was agreed on in steps 1

and 2. If a source TID does not match, the packet should be

discarded as erroneously sent from somewhere else. An error packet

should be sent to the source of the incorrect packet, while not

disturbing the transfer. This can be done only if the TFTP in fact

receives a packet with an incorrect TID. If the supporting protocols

do not allow it, this particular error condition will not arise.

The following example demonstrates a correct operation of the

protocol in which the above situation can occur. 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 response 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 for the two

connections 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 by returning an error packet.

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 Data (DATA)

4 Acknowledgment (ACK)

5 Error (ERROR)

The TFTP header of a packet contains the opcode associated with

that packet.

Sollins [Page 5]

RFC 1350 TFTP Revision 2 July 1992

2 bytes string 1 byte string 1 byte

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

| Opcode | Filename | 0 | Mode | 0 |

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

Figure 5-1: RRQ/WRQ packet

RRQ and WRQ packets (opcodes 1 and 2 respectively) have the format

shown in Figure 5-1. The file name is a sequence of bytes in

netascii terminated by a zero byte. The mode field contains the

string "netascii", "octet", or "mail" (or any combination of upper

and lower case, such as "NETASCII", NetAscii", etc.) 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. Octet mode is used to transfer a file that is in the 8-bit

format of the machine from which the file is being transferred. It

is assumed that each type of machine has a single 8-bit format that

is more common, and that that format is chosen. For example, on a

DEC-20, a 36 bit machine, this is four 8-bit bytes to a word with

four bits of breakage. If a host receives a octet file and then

returns it, the returned file must be identical to the original.

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.

The mail recipient string should be of the form "username" or

"username@hostname". If the second form is used, it allows the

option of mail forwarding by a relay computer.

The discussion above assumes that both the sender and recipient are

operating in the same mode, but there is no reason that this has to

be the case. For example, one might build a storage server. There

is no reason that such a machine needs to translate netascii into its

own form of text. Rather, the sender might send files in netascii,

but the storage server might simply store them without translation in

8-bit format. Another such situation is a problem that currently

exists on DEC-20 systems. Neither netascii nor octet accesses all

the bits in a word. One might create a special mode for such a

machine which read all the bits in a word, but in which the receiver

stored the information in 8-bit format. When such a file is

retrieved from the storage site, it must be restored to its original

form to be useful, so the reverse mode must also be implemented. The

user site will have to remember some information to achieve this. In

both of these examples, the request packets would specify octet mode

to the foreign host, but the local host would be in some other mode.

No such machine or application specific modes have been specified in

TFTP, but one would be compatible with this specification.

It is also possible to define other modes for cooperating pairs of

Sollins [Page 6]

RFC 1350 TFTP Revision 2 July 1992

hosts, although this must be done with care. There is no requirement

that any other hosts implement these. There is no central authority

that will define these modes or assign them names.

2 bytes 2 bytes n bytes

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

| Opcode | Block # | Data |

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

Figure 5-2: DATA packet

Data is actually transferred in DATA packets depicted in Figure 5-2.

DATA packets (opcode = 3) 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 end of the transfer. (See the section

on Normal Termination for details.)

All packets other than duplicate ACK's and those used for

termination are acknowledged unless a timeout occurs [4]. Sending a

DATA packet is an acknowledgment for the first ACK packet of the

previous DATA packet. The WRQ and DATA packets are acknowledged by

ACK or ERROR packets, while RRQ

2 bytes 2 bytes

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

| Opcode | Block # |

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

Figure 5-3: ACK packet

and ACK packets are acknowledged by DATA or ERROR packets. Figure

5-3 depicts an ACK packet; the opcode is 4. 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.

Sollins [Page 7]

RFC 1350 TFTP Revision 2 July 1992

2 bytes 2 bytes string 1 byte

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

| Opcode | ErrorCode | ErrMsg | 0 |

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

Figure 5-4: ERROR packet

An ERROR packet (opcode 5) takes the form depicted in Figure 5-4. An

ERROR packet can be the acknowledgment of any other type of packet.

The error code is an integer indicating the nature of the error. A

table of values and meanings is given in the appendix. (Note that

several error codes have been added to this version of this

document.) 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 host acknowledging the final DATA packet may terminate its side

of the connection on sending the final ACK. On the other hand,

dallying is encouraged. This means that the host sending the final

ACK will wait for a while before terminating in order to retransmit

the final ACK if it has been lost. The acknowledger will know that

the ACK has been lost if it receives the final DATA packet again.

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 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 or network 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.

Sollins [Page 8]

RFC 1350 TFTP Revision 2 July 1992

I. Appendix

Order of Headers

2 bytes

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

| Local Medium | Internet | Datagram | TFTP Opcode |

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

TFTP Formats

Type Op # Format without header

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 |

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

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.

Sollins [Page 9]

RFC 1350 TFTP Revision 2 July 1992

Error Codes

Value Meaning

0 Not defined, see error message (if any).

1 File not found.

2 Access violation.

3 Disk full or allocation exceeded.

4 Illegal TFTP operation.

5 Unknown transfer ID.

6 File already exists.

7 No such user.

Internet User Datagram Header [2]

(This has been included only for convenience. TFTP need not be

implemented on top of the Internet User Datagram Protocol.)

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 |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

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 UDP packet, including UDP header.

Checksum Reference 2 describes rules for computing checksum.

(The implementor of this should be sure that the

correct algorithm is used here.)

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.

Sollins [Page 10]

RFC 1350 TFTP Revision 2 July 1992

References

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

[2] Postel, J., "User Datagram Protocol," RFC 768, USC/Information

Sciences Institute, 28 August 1980.

[3] Postel, J., "Telnet Protocol Specification," RFC 764,

USC/Information Sciences Institute, June, 1980.

[4] Braden, R., Editor, "Requirements for Internet Hosts --

Application and Support", RFC 1123, USC/Information Sciences

Institute, October 1989.

Security Considerations

Since TFTP includes no login or access control mechanisms, care must

be taken in the rights granted to a TFTP server process so as not to

violate the security of the server hosts file system. TFTP is often

installed with controls such that only files that have public read

access are available via TFTP and writing files via TFTP is

disallowed.

Author's Address

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

Sollins [Page 11]