WIKIPEDIA

Trivial File Transfer Protocol

Trivial File Transfer Protocol (**TFTP**) is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host. One of its primary uses is in the early stages of nodes booting from a local area network. TFTP has been used for this application because it is very simple to implement.

TFTP was first standardized in 1981[1] and the current specification for the protocol can be found in RFC 1350.

Contents

- Overview
- Details
- Security considerations
- IETF standards documentation
- See also
- References

Overview

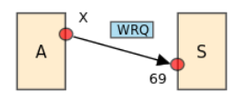
Due to its simple design, TFTP can be easily implemented by code with a small memory footprint. It is therefore the protocol of choice for the initial stages of any network booting strategy like BOOTP, PXE, BSDP, etc., when targeting from highly resourced computers to very low resourced Single-board computers (SBC) and System on a Chip (SoC). It is also used to transfer firmware images and configuration files to network appliances like routers, firewalls, IP phones, etc. Today, TFTP is virtually unused for Internet transfers.

TFTP's design was influenced from the earlier protocol EFTP, which was part of the PARC Universal Packet protocol suite. TFTP was first defined in 1980 by IEN 133.[2] In June 1981 The TFTP Protocol (Revision 2) was published as RFC 783 and later updated in July 1992 by RFC 1350 which fixed among other things the Sorcerer's Apprentice Syndrome. In March 1995 the TFTP Option Extension RFC 1782 updated later in May 1998 by RFC 2347, defined the option negotiation mechanism which establishes the framework for file transfer options to be negotiated prior to the transfer using a mechanism which is consistent with TFTP's original specification.

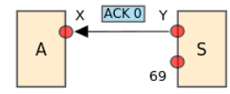
TFTP is a simple protocol for transferring files, implemented on top of the UDP/IP protocols using well-known port number 69. TFTP was designed to be small and easy to implement, and therefore it lacks most of the advanced features offered by more robust file transfer protocols. TFTP only reads and writes files from or to a remote server. It cannot list, delete, or rename files or directories and it has no provisions for user authentication. Today TFTP is generally only used on local area networks (LAN).

Details

In TFTP, a transfer is initiated by the client issuing a request to read or write a particular file on the server. The request can optionally include a set of negotiated transfer parameters proposed by the client under the terms specified by RFC 2347. If the server grants the request, the file is sent in fixed length blocks of 512 bytes by default or the number specified in the blocksize negotiated option defined by RFC 2348. Each block of transferred data, which is usually carried within a single IP packet in order to avoid IP fragmentation, must be acknowledged by an acknowledgment packet before the next block can be sent. A data packet of less than 512 bytes or the agreed blocksize option signals termination of a transfer. If a packet gets lost in the network, the intended recipient will timeout and may retransmit their 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 correctly received. Notice that both devices involved in a transfer are considered senders and receivers. One sends data and receives acknowledgments, the other sends acknowledgments and receives data.



(W1) Host A requests to write



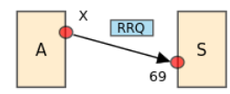
(W2) Server S acknowledges request

(W3) Host A sends numbered data packets

TFTP defines three modes of transfer: netascii, octet, and mail.

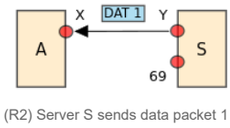
1. Netascii is a modified form of ASCII, defined in RFC 764. It consists of an 8-bit extension of the 7-bit ASCII character space from 0x20 to 0x7F (the printable characters and the space) and eight of the control characters. The allowed control characters include the null (0x00), the line feed (LF, 0x0A), and the carriage return (CR, 0x0D). Netascii also requires that the end of line marker on a host be translated to the character pair CR LF for transmission, and that any CR must be followed by either a LF or the null.
2. Octet allows for the transfer of arbitrary raw 8-bit bytes, with the received file resulting byte-per-byte identical to the one sent. More correctly, if a host receives an octet file and then returns it, the returned file must be identical to the original.[3]
3. Mail transfer mode uses Netascii transfer, but the file is sent to an email recipient by specifying that recipient's email address as the file name. RFC 1350 declared this mode of transfer obsolete.

TFTP uses UDP as its transport protocol. A transfer request is always initiated targeting port 69, but the data transfer ports are chosen independently by the sender and receiver during the transfer initialization. The ports are chosen at random according to the parameters of the networking stack, typically from the range of ephemeral ports.[4]



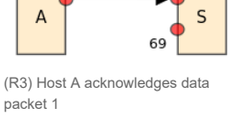
(R1) Host A requests to read

1. The initiating host A sends an RRQ (read request) or WRQ (write request) packet to host S at port number 69, containing the filename, transfer mode, and optionally any negotiated option under the terms of RFC 2347.
2. S replies with an option ACK if options were used, and an ACK (acknowledgement) packet to WRQ and directly with a DATA packet to RRQ. Packet is sent from a randomly allocated ephemeral port, and all future packets to host S should be directed to this port.
3. The source host sends numbered DATA packets to the destination host, all but the last containing a full-sized block of data (512 bytes default). The destination host replies with numbered ACK packets for all DATA packets.
4. The final DATA packet must contain less than a full-sized block of data to signal that it is the last. If the size of the transferred file is an exact multiple of the block-size, the source sends a final DATA packet containing 0 bytes of data.
5. Receiver responds to each DATA with associated numbered ACK. Sender responds to the first received ACK of a block with DATA of the next block.
6. If an ACK is not eventually received, a retransmit timer re-sends DATA packet.



(R2) Server S sends data packet 1

TFTP has always been associated to network booting. One of the first attempts in this regard was the Bootstrap Loading using TFTP standard RFC 906, published in 1984, which established the 1981 published Trivial File Transfer Protocol standard RFC 783 to be used as the standard file transfer protocol for bootstrap loading. It was followed shortly after by the Bootstrap Protocol standard RFC 951 (BOOTP), published in 1985, which allowed a disk-less client machine to discover its own IP address, the address of a TFTP server, and the name of a Network Bootstrap Program (NBP) to be TFTP transferred, loaded into memory, and executed. Dynamic Host Configuration Protocol standard RFC 2131 (DHCP) published in 1997 improved BOOTP capabilities. Finally, the Preboot Execution Environment (PXE) version 2.0 was released in December 1998, and the update 2.1 was made public in September 1999 counting on TFTP as its file transfer protocol.[5] Intel has recently decided to widely support PXE within the new UEFI specification extending the TFTP support to all EFI/UEFI environments.[6][7]



(R3) Host A acknowledges data packet 1

The original protocol has a transfer file size limit of 512 bytes/block x 65535 blocks = 32 MB. In 1998 this limit was extended to 65535 bytes/block x 65535 blocks = 4 GB by TFTP Blocksize Option RFC 2348. If the defined blocksize produces an IP packet size that exceeds the minimum MTU at any point of the network path, IP fragmentation and reassembly will occur not only adding more overhead[8] but also leading to total transfer failure when the minimalist IP stack implementation in a host's BOOTP or PXE ROM does not (or fails to properly) implement IP fragmentation and reassembly.[9] If TFTP packets should be kept within the standard Ethernet MTU (1500), the blocksize value is calculated as 1500 minus headers of TFTP (4 bytes), UDP (8 bytes) and IP (20 bytes) = 1468 bytes/block, this gives a limit of 1468 bytes/block x 65535 blocks = 92 MB. Today most servers and clients support block number roll-over (block counter going back to 0 or 1[10] after 65535) which gives an essentially unlimited transfer file size.

Since TFTP utilizes UDP, it has to supply its own transport and session support. Each file transferred via TFTP constitutes an independent exchange. Classically, this transfer is performed in lock-step, with only one packet (either a block of data, or an 'acknowledgement') alternatively in flight on the network at any time. Due to this single data block strategy instead of sending a larger amount of uninterrupted data blocks before pausing the transfer to wait for the corresponding acknowledge (windowing), TFTP provides low throughput especially over high latency links. Microsoft introduced windowed TFTP in Windows 2008 as part of their Windows Deployment Services (WDS), in January 2015 TFTP Windowsize Option RFC 7440 was published. This substantially improves performance for things like PXE booting without the IP fragmentation side effect sometimes observed on Blocksize Option RFC 2348[11]

Security considerations

TFTP includes no login or access control mechanisms. Care must be taken when using TFTP for file transfers where authentication, access control, confidentiality, or integrity checking are needed. Note that those security services could be supplied above or below the layer at which TFTP runs. Care must also be taken in the rights granted to a TFTP server process so as not to violate the security of the server's file system. TFTP is often installed with controls such that only files that have public read access are available via TFTP. Also listing, deleting, renaming, and writing files via TFTP are typically disallowed. TFTP file transfers are not recommended where the inherent protocol limitations could raise insurmountable liability concerns.[12]

IETF standards documentation

RFC Number	Title	Published	Author	Obsolete and Update Information
RFC 783 (https://datatracker.ietf.org/doc/html/rfc783)	The TFTP Protocol (Revision 1)	June 1981	K. Sollins	Obsoleted by - RFC 1350 (https://datatracker.ietf.org/doc/html/rfc1350)
RFC 906 (https://datatracker.ietf.org/doc/html/rfc906)	Bootstrap Loading using TFTP	June 1984	Ross Finlayson	—
RFC 951 (https://datatracker.ietf.org/doc/html/rfc951)	Bootstrap Protocol	Sep.1985	Bill Croft	Updated by RFC 1395 (https://datatracker.ietf.org/doc/html/rfc1395), 1497 (https://datatracker.ietf.org/doc/html/rfc1497), 1532 (https://datatracker.ietf.org/doc/html/rfc1532), 1542 (https://datatracker.ietf.org/doc/html/rfc1542), 5494 (https://datatracker.ietf.org/doc/html/rfc5494)
RFC 1350 (https://datatracker.ietf.org/doc/html/rfc1350)	The TFTP Protocol (Revision 2)	July 1992	K. Sollins	Updated by RFC 1782 (https://datatracker.ietf.org/doc/html/rfc1782), 1783 (https://datatracker.ietf.org/doc/html/rfc1783), 1784 (https://datatracker.ietf.org/doc/html/rfc1784), 1785 (https://datatracker.ietf.org/doc/html/rfc1785), 2347 (https://datatracker.ietf.org/doc/html/rfc2347), 2348 (https://datatracker.ietf.org/doc/html/rfc2348), 2349 (https://datatracker.ietf.org/doc/html/rfc2349)
RFC 1782 (https://datatracker.ietf.org/doc/html/rfc1782)	TFTP Option Extension	March 1995	G. Malkin	Obsoleted by - RFC 2347 (https://datatracker.ietf.org/doc/html/rfc2347)
RFC 2131 (https://datatracker.ietf.org/doc/html/rfc2131)	Dynamic Host Configuration Protocol	March 1997	R. Droms	Updated by RFC 3396 (https://datatracker.ietf.org/doc/html/rfc3396), 4361 (https://datatracker.ietf.org/doc/html/rfc4361), 5494 (https://datatracker.ietf.org/doc/html/rfc5494), 6842 (https://datatracker.ietf.org/doc/html/rfc6842)
RFC 2347 (https://datatracker.ietf.org/doc/html/rfc2347)	TFTP Option Extension	May 1998	G. Malkin	—
RFC 2348 (https://datatracker.ietf.org/doc/html/rfc2348)	TFTP Blocksize Option	May 1998	G. Malkin	—
RFC 2349 (https://datatracker.ietf.org/doc/html/rfc2349)	TFTP Timeout Interval and Transfer Size Options	May 1998	G. Malkin	—
RFC 5505 (https://datatracker.ietf.org/doc/html/rfc5505)	Principles of Internet Host Configuration	May 2009	B. Aboba	—
RFC 7440 (https://datatracker.ietf.org/doc/html/rfc7440)	TFTP Windowsize Option	Jan 2015	P. Masotta	—

See also

- Simple File Transfer Protocol

References

1. RFC 783
2. Karen R. Sollins (1980-01-29). *The TFTP Protocol* (http://www.rfc-editor.org/ien/ien133.txt). IETF. IEN 133. Retrieved 2010-05-01.
3. RFC 1350, page 5.
4. Karen R.Sollins (July 1992). *The TFTP Protocol (Revision 2)* (https://datatracker.ietf.org/doc/html/rfc1350). IETF. doi:10.17487/RFC1350 (https://doi.org/10.17487%2FRFC1350). RFC 1350 (https://datatracker.ietf.org/doc/html/rfc1350). Retrieved 2010-05-01.
5. "Preboot Execution Environment (PXE) Specification - Version 2.1" (https://web.archive.org/web/20131102003141/http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf) (PDF). Intel Corporation. 1999-09-20. Archived from the original (http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf) (PDF) on 2013-11-02. Retrieved 2014-02-08.
6. "Unified Extensible Firmware Interface Specification" (http://www.uefi.org/sites/default/files/resources/2_4_Errata_A.pdf) (PDF). UEFI. 2013-12-02. Retrieved 2014-04-04.
7. "UEFI PXE Boot Performance Analysis" (https://web.archive.org/web/20140808044632/https://uefidk.com/sites/default/files/Intel_UEFI_PXE_Boot_Performance_Analysis.pdf) (PDF). Intel Corporation. 2014-02-02. Archived from the original (https://uefidk.com/sites/default/files/Intel_UEFI_PXE_Boot_Performance_Analysis.pdf) (PDF) on 2014-08-08. Retrieved 2014-04-04.
8. RFC 2348, page 3.
9. RFC 5505, page 7.
10. "Extending TFTP" (https://www.compuphase.com/tftp.htm). CompuPhase. Retrieved 2018-12-12.
11. RFC 7440, page 1.
12. RFC 7440, page 7.