TFTP
Trivial File Transfer Protocol

References:
RFC 783, 1350

TFTP Usage and Design

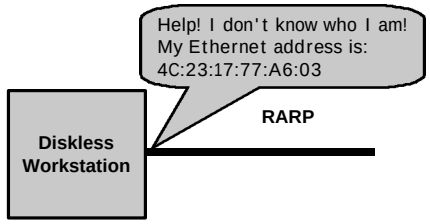
- Transfer files between processes.
- Minimal overhead (no security).
- Designed for UDP, although could be used with many transport protocols.

TFTP Usage and Design (cont.)

- Easy to implement
- Small - possible to include in firmware
- Used to bootstrap workstations and network devices.

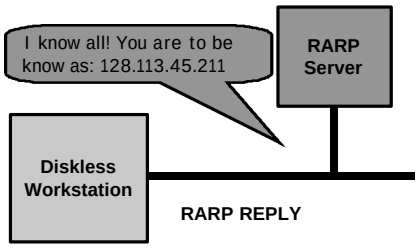
Diskless Workstation Booting 1

The call for help



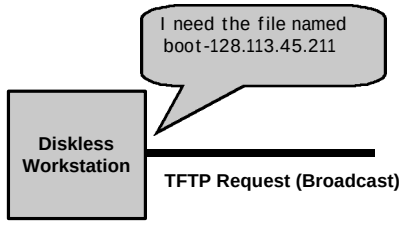
Diskless Workstation Booting 2

The answer from the all-knowing



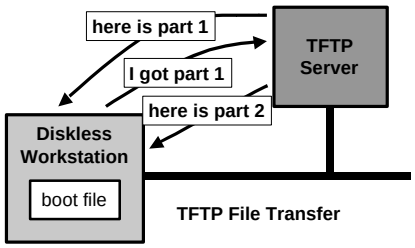
Diskless Workstation Booting 3

The request for instructions



Diskless Workstation Booting 4

The dialog



TFTP Protocol

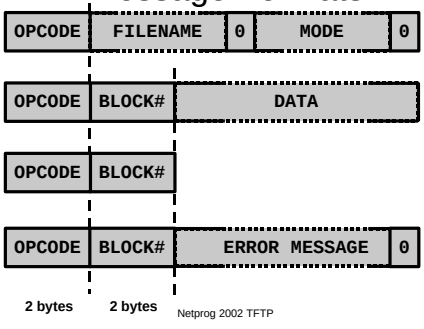
5 message types:

- Read request
- Write request
- Data
- ACK (acknowledgment)
- Error

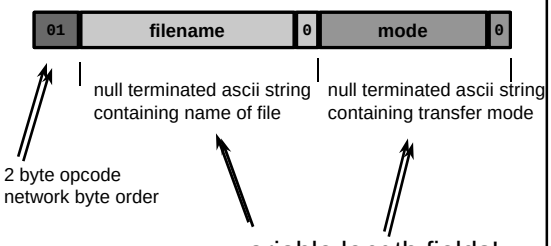
Messages

- Each is an independent UDP Datagram
- Each has a 2 byte opcode (1st 2 bytes)
- The rest depends on the opcode.

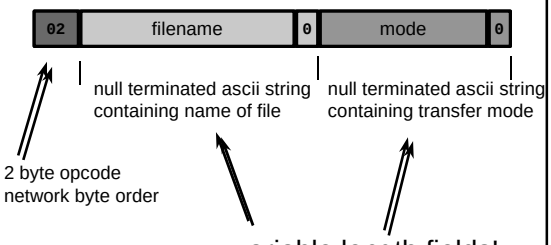
Message Formats



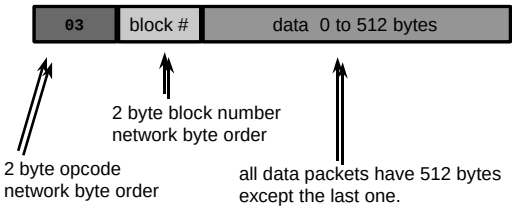
Read Request



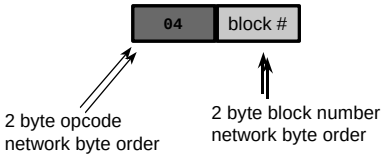
Write Request



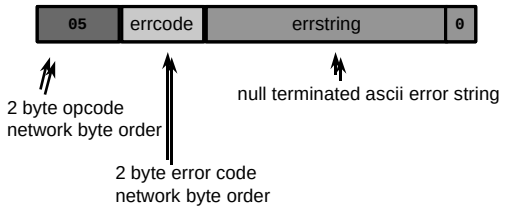
TFTP Data Packet



TFTP Acknowledgment



TFTP Error Packet



TFTP Error Codes

0 - not defined
1 - File not found
2 - Access violation
3 - Disk full
4 - Illegal TFTP operation
5 - Unknown port
6 - File already exists
7 - No such user

TFTP transfer modes

- "netascii" : for transferring text files.
  - all lines end with \r\n (CR,LF).
  - provides standard format for transferring text files.
  - both ends responsible for converting to/from netascii format.
- "octet" : for transferring binary files.
  - no translation done.

NetAscii Transfer Mode

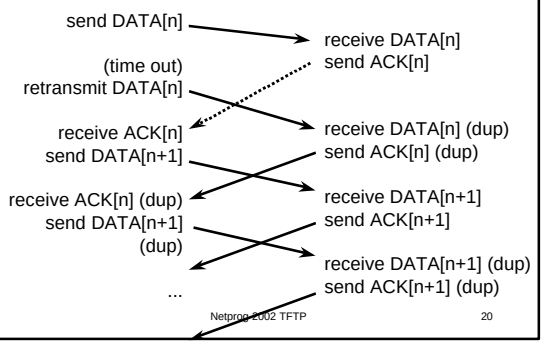
Unix - end of line marker is just '**\n**'

- receiving a file
  - you need to remove '**\r**' before storing data.
- sending a file
  - you need to replace every '**\n**' with "**\r\n**" before sending

Lost Data Packets - Original Protocol Specification

- Sender uses a timeout with retransmission.
  - sender could be client or server.
- Duplicate data packets must be recognized and ACK retransmitted.
- This original protocol suffers from the "sorcerer's apprentice syndrome".

Sorcerer's Apprentice Syndrome



The Fix

- Sender should not resend a data packet in response to a duplicate ACK.
- If sender receives ACK[n] - don't send DATA[n+1] if the ACK was a duplicate.

Concurrency

- TFTP servers use a "well known address" (UDP port number).
- How would you implement a concurrent server?
  - forking (alone) may lead to problems!
  - Can provide concurrency without forking, but it requires lots of bookkeeping.

TFTP Concurrency

- According to the protocol, the server may create a *new udp port* and send the initial response from this new port.
- The client should recognize this, and send all subsequent messages to the new port.

RRQ (read request)

- Client sends RRQ
- Server sends back data chunk #0
- Client acks chunk #0
- Server sends data chunk #1
- ...

WRQ (write request)

- Client sends WRQ
- Server sends back ack #0
- Client data chunk #1 (the first chunk!)
- Server acks data chunk #1
- …

there is no data chunk #0!

When is it over?

- There is no *length of file* field sent!
- All data messages *except the last one* contain 512 bytes of data.
  - message length is 2 + 2 + 512 = 516
- The last data message might contain 0 bytes of data!

Issues

What if more than 65535 chunks are sent?

- 65536 blocks x 512 bytes/block = 33,554,432 bytes.

- The RFC does not address this issue!
- Remember that the network can duplicate packets!