CpSc 360: Distributed and Network Programming

THE TFTP PROTOCOL (REVISION 2)

James Wang

http://www.faqs.org/rfcs/rfc1350.html

What is TFTP

- TFTP (Trivial File Transfer Protocol) is a very simple protocol used to transfer files.
- Each nonterminal packet is acknowledged separately.
- It has been implemented on top of the Internet User Datagram protocol (UDP or Datagram)
- It may be used to move files between machines on different networks implementing UDP.
- 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.

2

Protocol Overview

- 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.

3

Packet Retransmission

- 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.

4

Error Handling

- 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, 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).
  - 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.

5

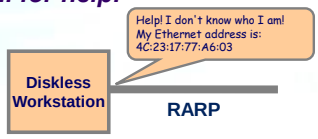
TFTP Usage and Design

- Transfer files between processes.
- Minimal overhead (no security).
- Designed for UDP, although could be used with many transport protocols.
- Easy to implement
- Small - possible to include in firmware
- Used to bootstrap workstations and network devices.

6

Diskless Workstation Booting

The call for help:

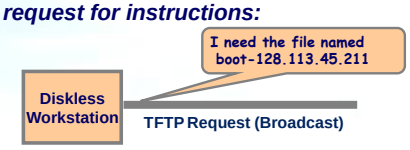


The answer from the all-knowing:

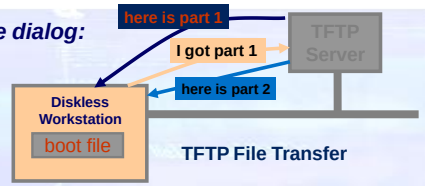
7

Diskless Workstation Booting (Conti.)

The request for instructions:



The dialog:



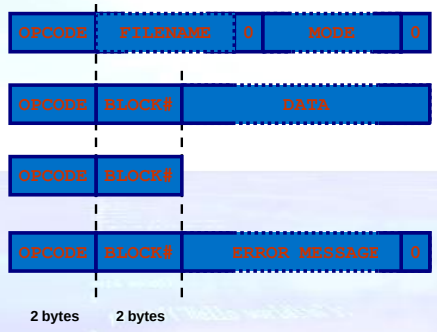
8

TFTP Messages

- Each message is an independent UDP Datagram.
- Each has a 2 byte opcode (1st 2 bytes)
- The structure of the rest of the datagram depends on the opcode.
- Message types:
  - Read request
  - Write request
  - Data
  - ACK (acknowledgment)
  - Error

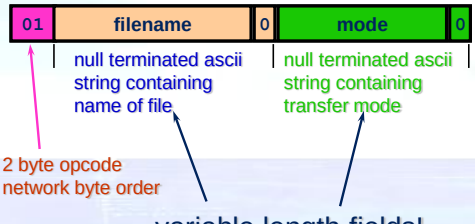
9

Message Formats



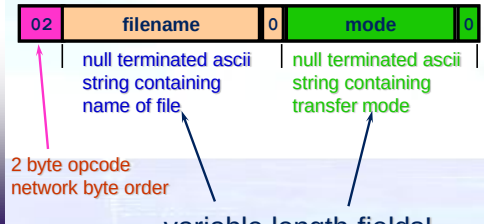
10

Read Request



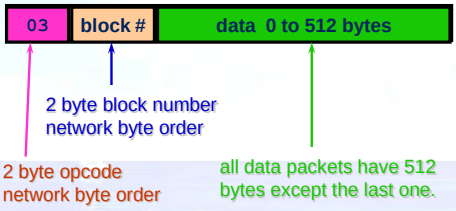
11

Write Request



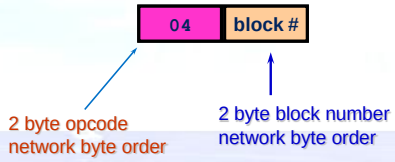
12

TFTP Data Packet



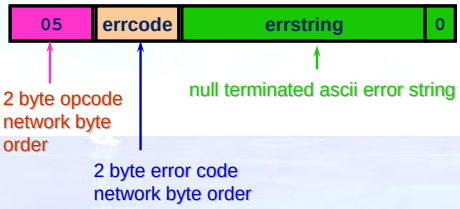
13

TFTP Acknowledgment



14

TFTP Error Packet



15

TFTP Error Codes (16 bit int)

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

16

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.

17

NetAscii Transfer Mode

For certain OS, such as 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

18

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.
- **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.**

19

RRQ (read request)

- **Client sends RRQ**
- **Server sends back data chunk #1**
- **Client acks chunk #1**
- **Server sends data chunk #2**
- **...**

20

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!

21

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!**
- **What if more than 65535 chunks are sent?**
  - 65536 blocks x 512 bytes/block = 33,554,432 bytes.

22