

TFTP
Trivial File Transfer Protocol

References:
RFC 783, 1350

Netprog 2000 TFTP

1

TFTP Usage and Design

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

Netprog 2000 TFTP

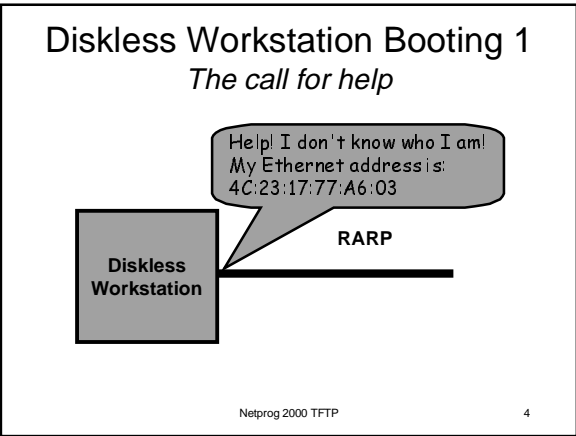
2

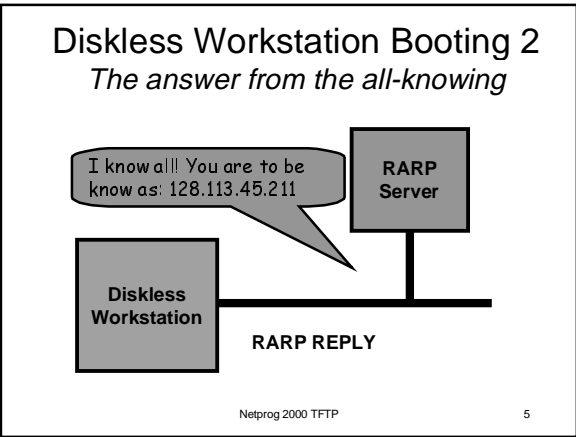
**TFTP Usage and Design
(cont.)**

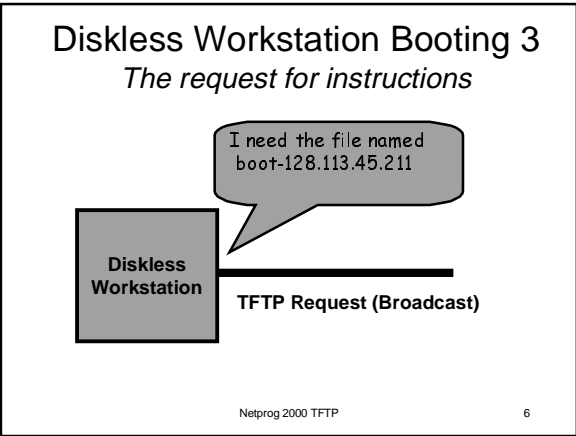
- Easy to implement
- Small - possible to include in firmware
- Often uses to bootstrap workstations and network devices.

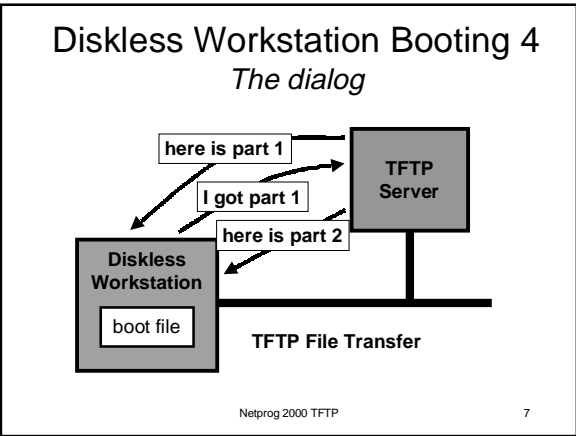
Netprog 2000 TFTP

3









TFTP Protocol

5 message types:

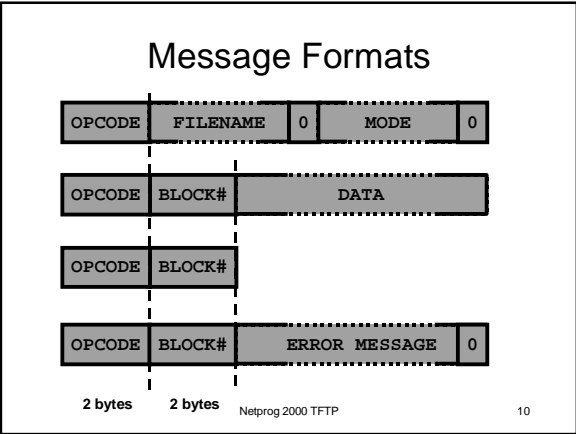
- Read request
- Write request
- Data
- ACK (acknowledgment)
- Error

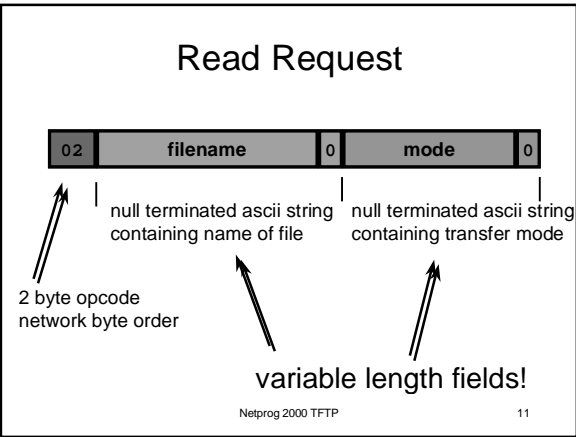
Netprog 2000 TFTP 8

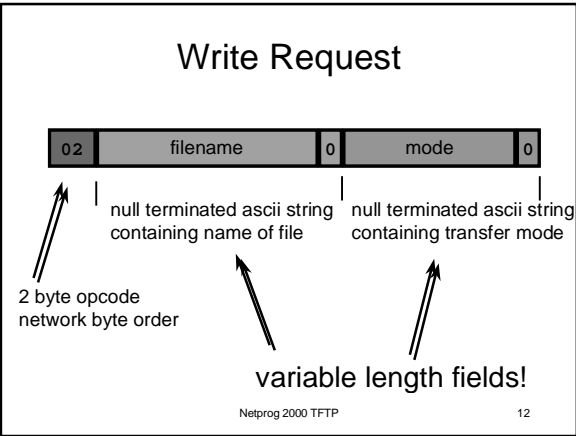
Messages

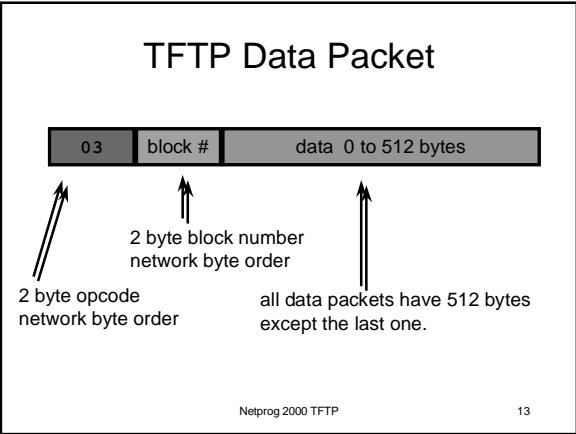
- Each is an independent UDP Datagram
- Each has a 2 byte opcode (1st 2 bytes)
- The rest depends on the opcode.

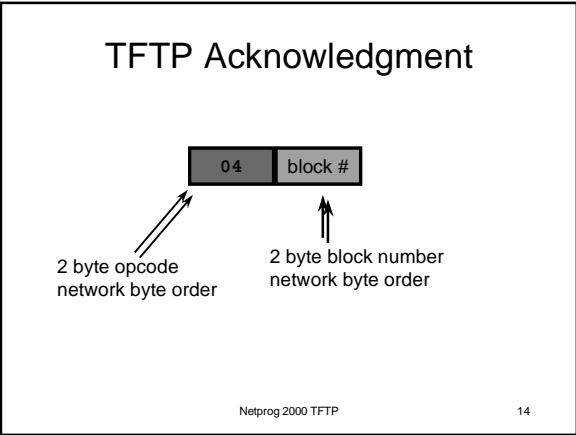
Netprog 2000 TFTP 9

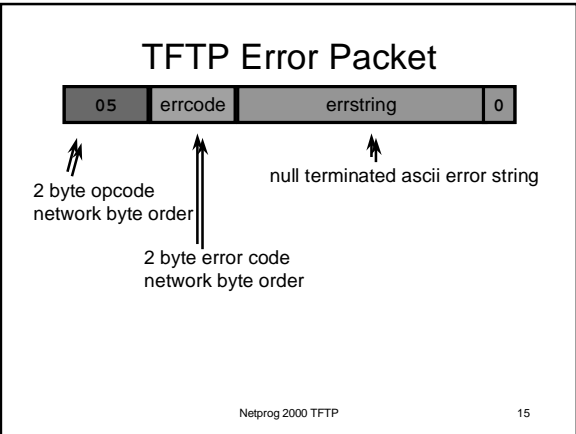












TFTP Error Codes

- 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

Netprog 2000 TFTP

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.

Netprog 2000 TFTP

17

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

Netprog 2000 TFTP

18

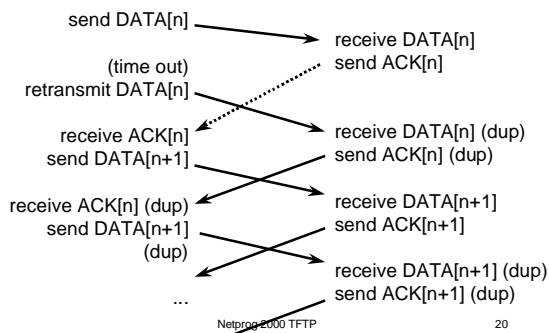
Lost Data Packets - Original Protocol Specification

- 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".

Netprog 2000 TFTP

19

Sorcerer's Apprentice Syndrome



Netprog 2000 TFTP

20

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.

Netprog 2000 TFTP

21

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.

Netprog 2000 TFTP

22

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.

Netprog 2000 TFTP

23

RRQ (read request)

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

Netprog 2000 TFTP

24

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!

Netprog 2000 TFTP

25

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!

Netprog 2000 TFTP

26

Issues

What if more than 65535 chunks are sent?

- $65536 \text{ blocks} \times 512 \text{ bytes/block} = 33,554,432$ bytes.

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

Netprog 2000 TFTP

27

Timeouts

- Set up an alarm to go off after a few seconds.
- Call `recvfrom` (or `recv` or `read`).
 - Check for error and `EINTR`
 - Can also set a flag in the `SIGALRM` signal handler.

Netprog 2000 TFTP

28

Timeout & Retransmission Parameters

- Reasonable Values:
 - wait no more than 5 seconds
 - retransmit no more than 5 times
- If no response - give up!

Netprog 2000 TFTP

29

Avoiding Sorcerer's Apprentice Syndrome

- Sender should timeout and retransmit.
- Sender should ignore duplicate ACKs.
 - don't retransmit data!
- Receiver should transmit ACK whenever data is received.
 - could be duplicate ACK, that's OK.

Netprog 2000 TFTP

30

Building Messages

- The messages are built in memory.
- Entire message is given to `sendto`.
- Opcode, block# are *binary, network byte order, 2-byte integers*.

Netprog 2000 TFTP

31

Suggestions

- Write a function that builds a message.
`buildmsg(char *buf, int op, int block, ...`
 - one place in the code to worry about network byte order!
- Write a function that extracts fields from a message.

Netprog 2000 TFTP

32

Stuffing binary values into a buffer

```
short int opcode; char *buffer;
```

Using memcpy - need to convert to NBO first:

```
tmp = htons(opcode);  
memcpy(buffer, (char *) &tmp, 2);
```

Netprog 2000 TFTP

33

Advanced Stuffing Techniques

(a great name for a band!)

Stuffing NBO short in to a buffer:

```
*((short int *) buffer) = htons(opcode);
```

Extracting a NBO short from a buffer

```
opcode = ntohs( *((short int *) buffer+2));
```

doesn't have to be the beginning of the buffer!

Netprog 2000 TFTP

34
