

# TFTP

## Trivial File Transfer Protocol

References:

RFC 783

# 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
- Often uses to bootstrap workstations and network devices.

# TFTP Protocol

- 5 message types
  - Read request
  - Write request
  - Data
  - ACK (acknowledgment)
  - Error

# Read Request



null terminated ascii string  
containing name of file

null terminated ascii string  
containing transfer mode

2 byte opcode  
network byte order

variable length fields!

# Write Request



null terminated ascii string  
containing name of file

null terminated ascii string  
containing transfer mode

2 byte opcode  
network byte order

variable length fields!

# TFTP Data Packet

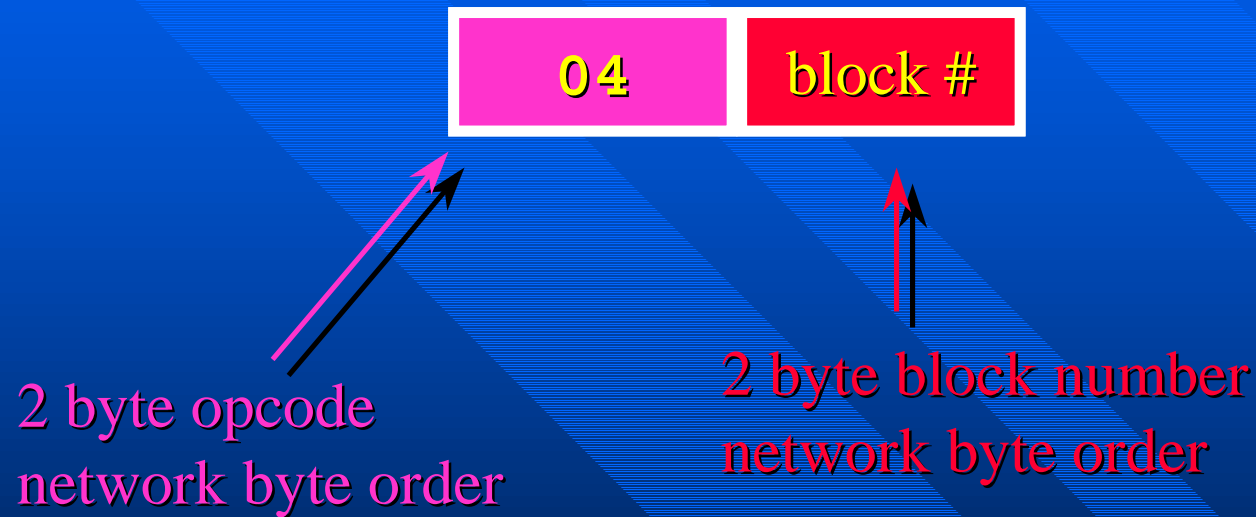


2 byte opcode  
network byte order

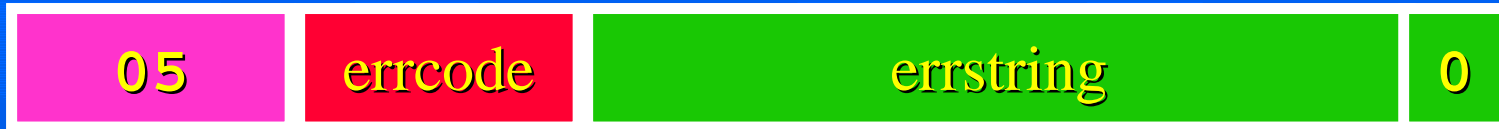
2 byte block number  
network byte order

all data packets have 512 bytes  
except the last one.

# TFTP Acknowledgment



# TFTP Error Packet



2 byte opcode  
network byte order

2 byte error code  
network byte order

null terminated ascii error string

## 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

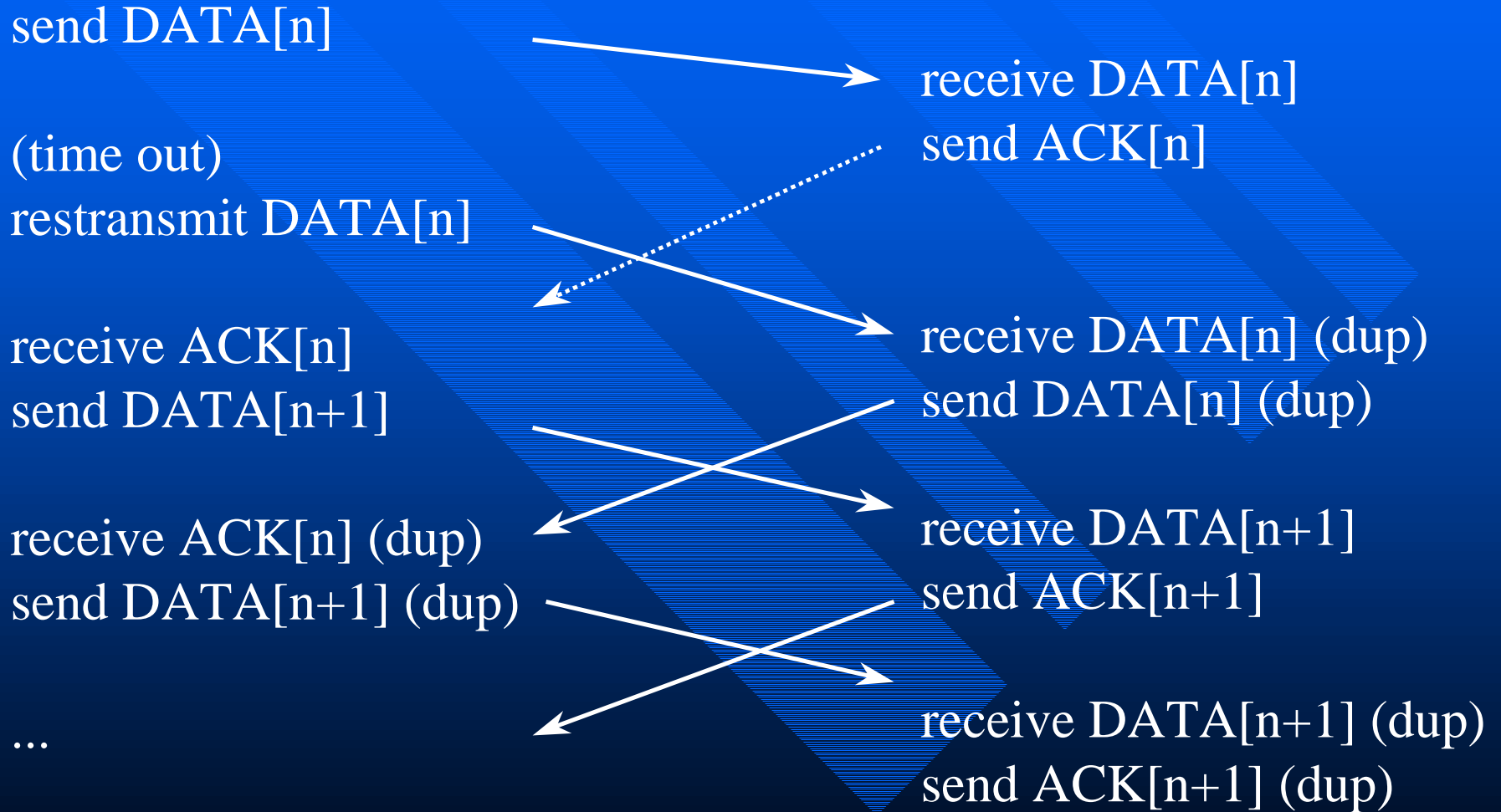
# TFTP transfer modes

- “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.

# Lost Data Packets - Original Protocol Specification

- Sender uses a timeout with retransmission.
  - sender could be client or server.
- Duplicate data packets must be recognized (ignored) 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.