Trivial File Transfer Protocol

Network Programming

Motivation

- Suppose you want to send a file
- Too big to wrap into one packet
- Transport layer doesn't know your file system
- Is it text? Binary? Rare case of Presentation layer being noticeable…
- Discussion: What features should we incorporate? What should we leave out?

RFC 1350

- RFC 1350 "The TFTP Protocol (Revision 2)" will be used for Assignment 1
- TFTP inherently offers very little, the entire RFC is only 10 pages!
  - This is why it's "trivial"
- The more commonly used File Transfer Protocol (FTP) is in RFC 959 and is 70 pages.

Requirements

- Before getting into how TFTP works, it is worth noting that it’s typically implemented over UDP
- Could use any datagram protocol
- Rely on UDP for "sessions" even though we’re going to manage them at the Application Layer

Header

- TFTP consists of a TFTP Header and TFTP data, which sits on top of the Transport information (in our case UDP header)
- What's the TFTP header?
  - 2 byte "op code"

Op Codes

- 1: Read Request (RRQ)
- 2: Write Request (WRQ)
- 3: Data (DATA)
- 4: Acknowledgement (ACK)
- 5: Error (ERROR)

How to Read an Op Code

- Option 1: Pointers and Type Casting

```
char buffer[BUF_LEN];
unsigned short int opcode;
unsigned short int * opcode_ptr;
…
//message read into buffer by this point
opcode_ptr = (unsigned short int *)buffer;
opcode = ntohs(*opcode_ptr);
```

- Option 2: Bitmasking

```
opcode = ntohs(buffer[0] << 8)|buffer[1])
```

RRQ/WRQ structure

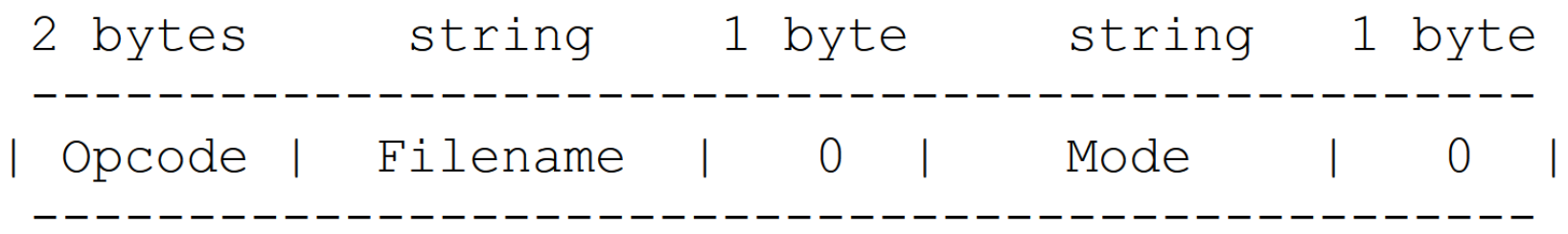


Figure 5-1: RRQ/WRQ packet

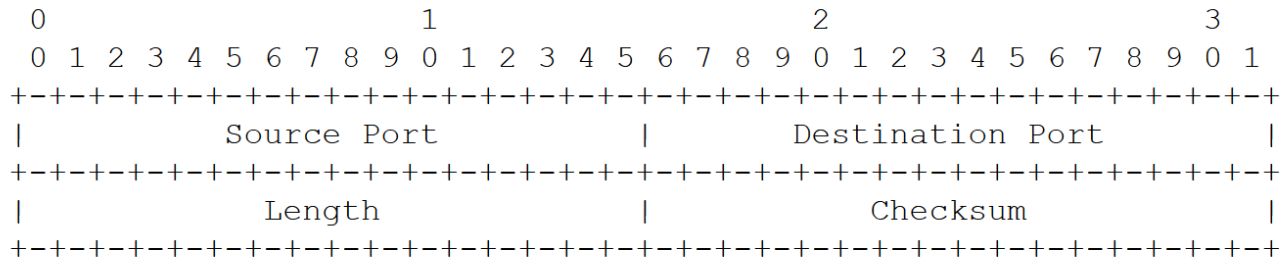
TIDs

- “On the other hand, the source and destination port fields of the Datagram header (its format is given in the appendix) are used by TFTP and the length field reflects the size of the TFTP packet. The transfer identifiers (TID's) used by TFTP are passed to the Datagram layer to be used as ports; therefore they must be between 0 and 65,535. The initialization of TID's is discussed in the section on initial connection protocol.” –RFC 1350, §3

Initiating (Simplified)

- Server receives a RRQ/WRQ with the Transfer ID (TID) of the requesting host
- Either the server sends…
  - DATA block 1 with matching Destination TID and its own Source TID
  - Or sends ERROR with the appropriate code
- How to select TIDs?
  - Read §4
  - What if the server TID was always UDP 69 (or our server's TFTP listening port)?

How Do We Share TIDs?



Values of Fields

Source Port	Picked by originator of packet.
Dest. Port	Picked by destination machine (69 for RRQ or WRQ).
Length	Number of bytes in UDP packet, including UDP header.
Checksum	Reference 2 describes rules for computing checksum. (The implementor of this should be sure that the correct algorithm is used here.) Field contains zero if unused.

Note: TFTP passes transfer identifiers (TID's) to the Internet User Datagram protocol to be used as the source and destination ports.

Mode?

- “netascii” is just ASCII (sometimes called plaintext or text mode)
  - Receiver must translate
  - Just deals with line endings, but in binary data we might have an octet that looks like CR or LF…
- “mail” sends to a host/relay with the intention of emailing data.
  - WRQ only
  - We won’t use this … why not?

Mode?

- “octet” is a binary format – this will be a stream of bits.
  - Use the sender format
  - RFC mentions the 36 bit format of a DEC-20

DATA structure

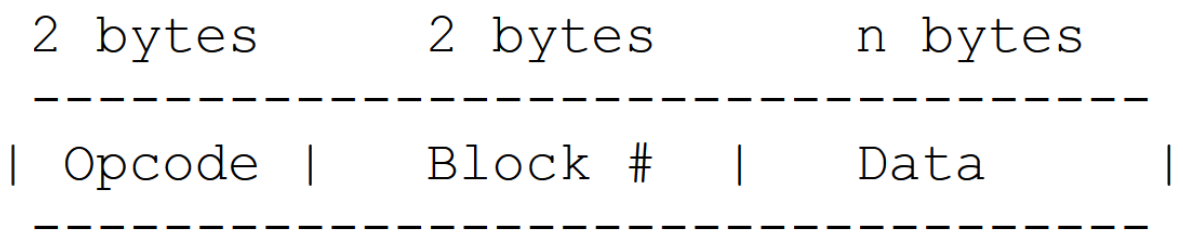


Figure 5-2: DATA packet

How to Read a Block Number

- Option 1: Could reuse the trick from reading op code, but change the pointer to be 2 bytes further.

```
block_ptr = ntohs(opcode_ptr + 1)
```

- Why isn't this opcode_ptr + 2?
- Option 2: Could use bitmasking

```
block_ptr = ntohs(buffer[2]<<8|buffer[3])
```

Terminating

- If n<512, this is the last block
  - That’s it!

ACK Strcture

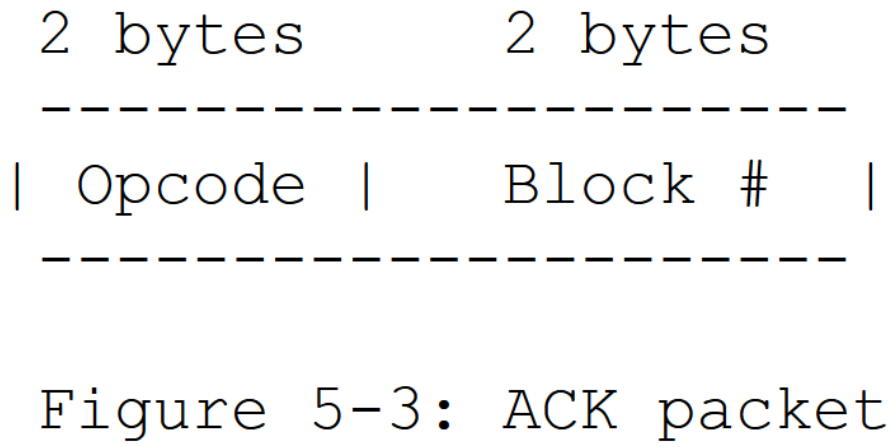


Figure 5-3: ACK packet

ERROR Structure

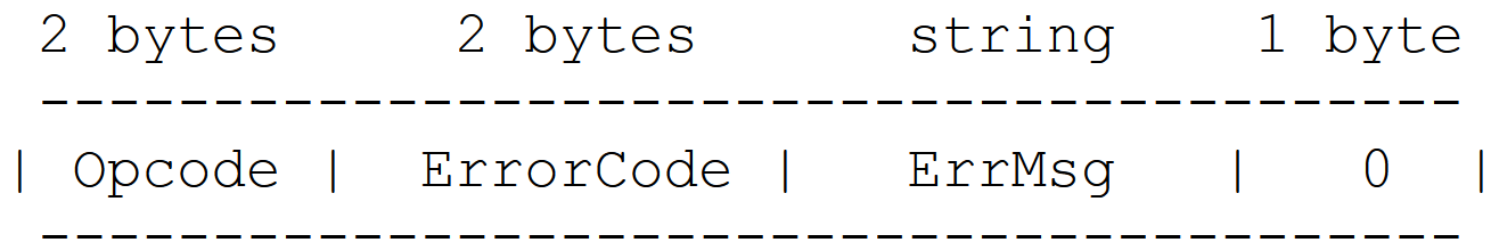


Figure 5-4: ERROR packet

Error Codes

Value	Meaning
0	Not defined, see error message (if any).
1	File not found.
2	Access violation.
3	Disk full or allocation exceeded.
4	Illegal TFTP operation.
5	Unknown transfer ID.
6	File already exists.
7	No such user.

Limitations

- No retransmission
- ACK system ensures we don't waste outbound network bandwidth but that's it
- How big a file can we send?
  - A) 65536*512 = 33,554,432 bytes = 32MB
  - B) 65535*512 = 33,553,920 bytes
  - C) (65535 * 512) – 1 = 33,553,919 bytes
- RFC 2348 (which we will not support) can raise this limit by a lot

Reliability & Timeout

- What if the receiver had a timeout T_R, and if no data was received after T_R since last ACK was sent, but n=512?
  - Resend the ACK
- What if the sender had a timeout T_S, and if no ACK was received after T_S…
  - Resend last DATA?
  - Kill “connection”?

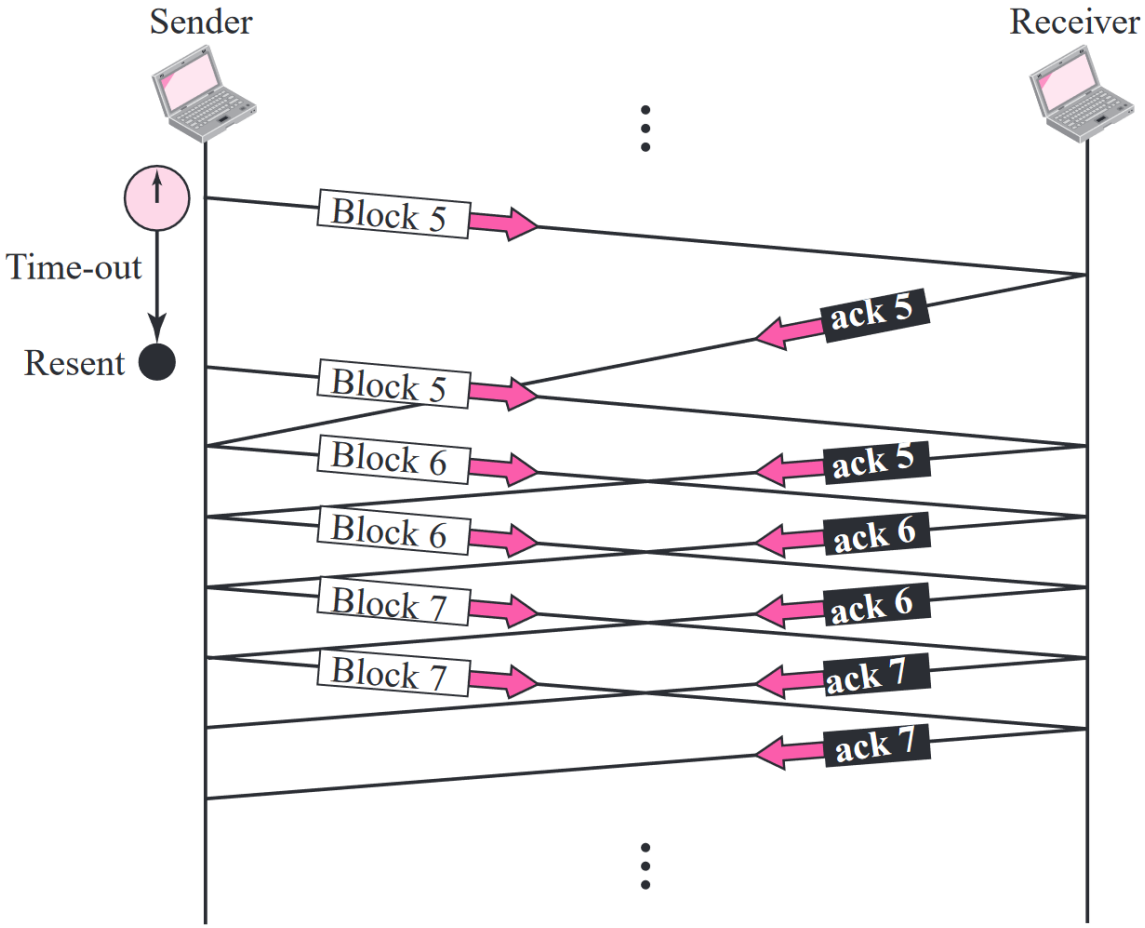
The Sorcerer's Apprentice

- *Der Zauberlehrling*, Goethe (1797)
- "The Sorcerer's Apprentice", *Fantasia* (1940)

- RFC 1123, IETF, Braden et al. (1989)

The Sorcerer's Apprentice

Figure 21.17 *Sorcerer's apprentice bug*



From Wikipedia

- “The TFTP specification said that any time *any* packet was received, the receiver was *required* to send the appropriate reply packet. Thus, the receipt of a block of data triggered the sending of an 'acknowledgement', and the receipt of an acknowledgement triggered the sending of the next data block.”

Reference

- Sorcerer's apprentice figure taken from the 4th Ed. of "TCP/IP Protocol Suite" by Behrouz A. Forouzan, 2010, pg 649

Video

- From Disney's "Fantasia"
- Sorcerer's Apprentice clip
- https://youtu.be/2DX2yVucz24?t=343