15-441 Project 1 Overview

1/25/06

Mike Cui

Your Assignment Is …

To implement a TFTP server.

Detailed handout on course website.

What exactly does this involve ?

What is TFTP ?

- The Trivial File Transfer Protocol
  - Basic file transfer protocol
  - Supports only Get & Put operations
- Major uses :
  - Netbooting workstations
  - Example of a simple but useful protocol
- Defined by a standards body document
  - RFC 1350 (1992)
  - RFC 1123 (1989) (bug fix)
  - RFC 783 (1981) (obsolete)

The Standard

- Defines
  - Message types & formats
  - Sequence of messages
  - Connection set-up and termination
- Written in very rigid style
  - Not necessarily easy to understand

Packets

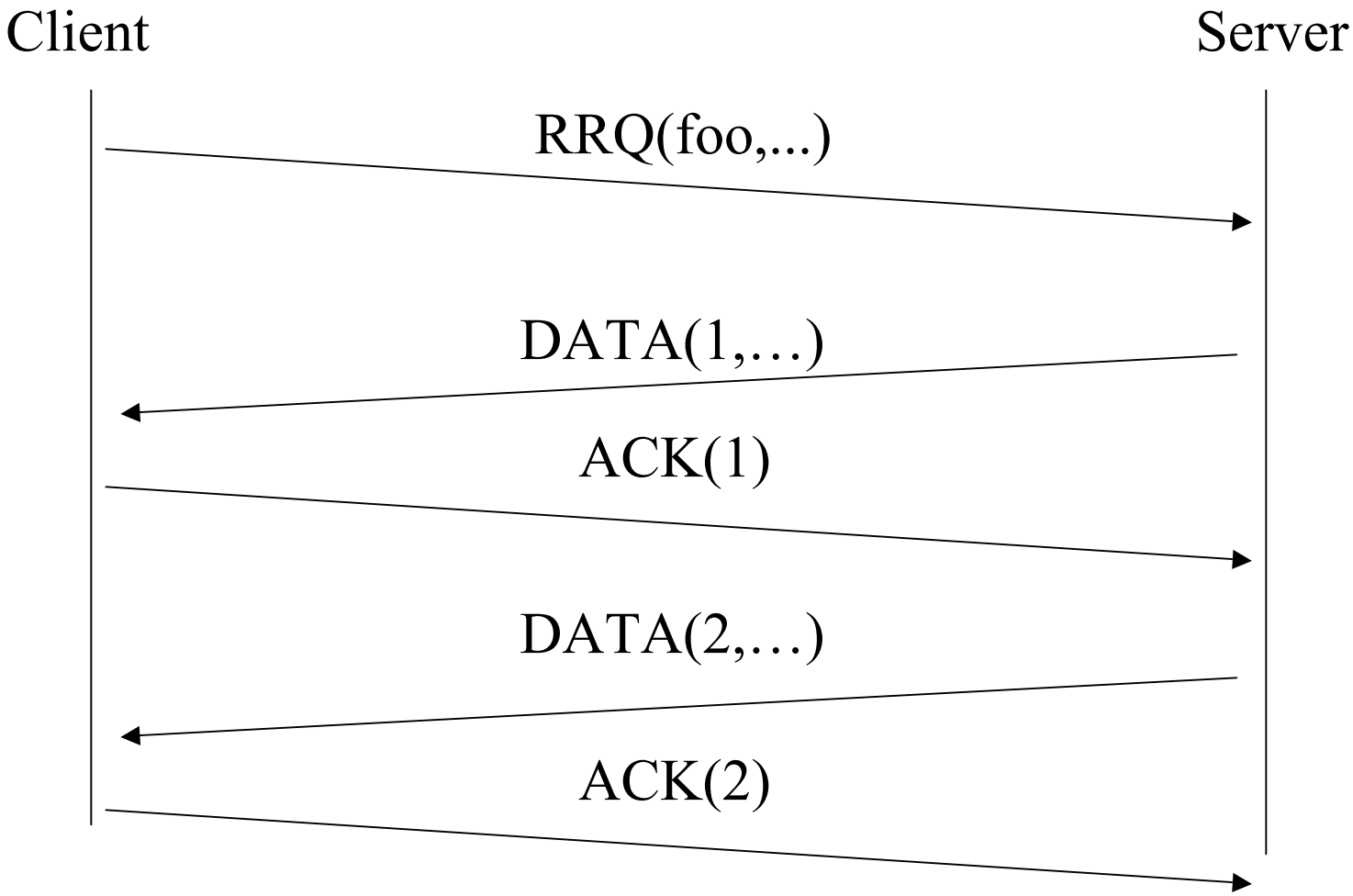
- Sent over User Datagram Protocol (UDP)
  - Single message (datagram)
  - See chapter 5.1 for details
- Only 5 types :
  1. RRQ (filename, mode)
  2. WRQ (filename, mode)
  3. DATA (block number, data bytes)
  4. ACK (block number)
  5. ERROR (error code, error message)
- Largest packet is limited to 516 bytes
  - DATA packets, specified by the RFC
  - RRQ/WRQ packets, specified by us for this project

Protocol

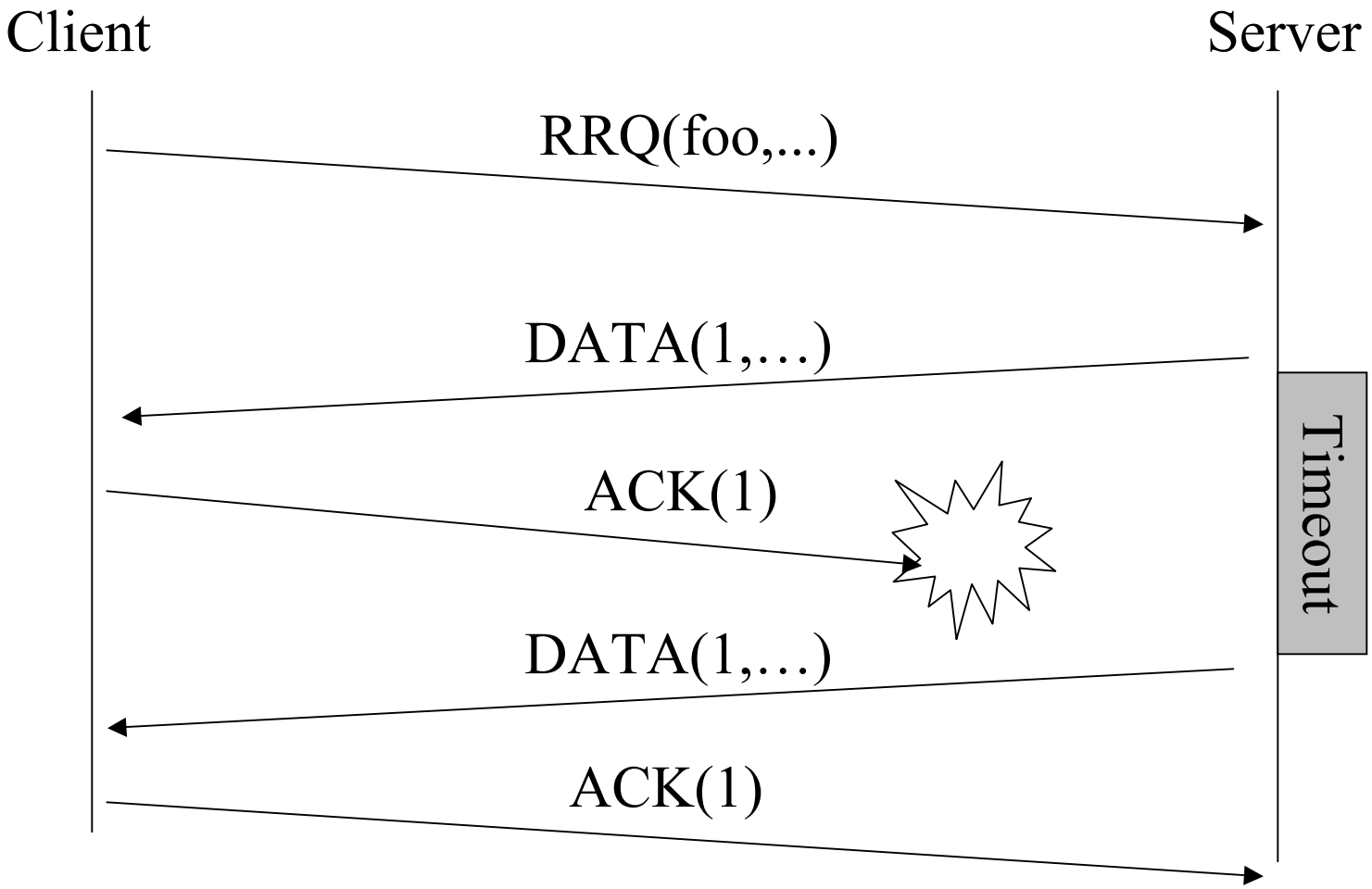
- Stop and wait protocol - send a message, wait for reply
- Send DATA(1,…) in response to RRQ
- Send ACK(n) in response to DATA(n)
- Send DATA(n+1,...) in response to ACK(n)

- What happens when a message is lost?
  - Sender retransmits DATA or RRQ
  - How do you know when to stop?

Get Example



Get Example (lost packet)



Hints

- Protocol Issues
- UDP
- Network Byte Order
- Debugging tools
- Project Planning

General Protocol Issues

- TFTP uses the well known UDP port 69
  - Usually only superuser can bind to ports < 1024
  - Use a different port instead
- Responses to RRQ/WRQ *must be* sent out on a different port than the well known port.
  - *Must* to create another socket
  - `bind` to port `0` will pick any free port
- Each side can consider the connection terminated when it sends to or receives from the other side an ERROR packet.
  - What if the ERROR packet is lost ?

UDP

- `socket(AF_INET, SOCK_DGRAM, 0)` creates a UDP socket
- `bind()` assigns the socket an address and port
- `recv()/recvfrom()` gets an entire packet addressed to the port assigned by `bind()`
  - Or (optionally) blocks until an entire packet arrives
  - No short-count, or EOF
  - Packet is truncated if buffer isn't large enough
- `recvfrom()` also fills in the source address
- `connect()` sets the default destination
  - Just a shortcut, no "connection" is actually made!
- `send()` sends a packet to the destination set by `connect()`
  - Packets might reach destination 0 or more times
- `sendto()` can specify a destination
- `accept()/listen()` not applicable

More on UDP

- `man udp`
- UDP server example in the reference section of Project handout

Network Byte Order

- Network functions deal in bytes
- Multi-byte structures in a message are more complicated (eg: integers)
  - One host could be big-endian, the other little-endian
  - Choose one byte order for messages on the wire (pages 536-538), which is big-endian
- Provide conversion functions for common types
  - `Long  : htonl, ntohl`
  - `Short : htons, ntohs`

Debugging Tools

- TFTP clients
  - `tftp` installed on Andrew Linux & Solaris

```
% tftp quark.weh.andrew.cmu.edu 3000
tftp> binary
tftp> get foo.c
tftp> put bar.sml
```

  - `trace` prints the packets sent & received
- `netstat`
  - List open sockets
- `gdb`

Project Planning

- Start early !
- Should already have read the RFC by now
  - Read it again
- This project may be larger than your previous ones.
  - Expect about 750-1000 lines of C-code
  - Most of the complexity will be in exceptional handling.
- Think about the corner cases early
- Use office hours

Questions ?