Application Level Protocol Design

Distributed Software Systems

Application Layer Protocol Design

- Steps in design
  - Services
  - Protocol Data Unit (PDU) structure and encoding
  - Protocol
  - Client, Server, interaction with environment (DNS, NFS, etc.)

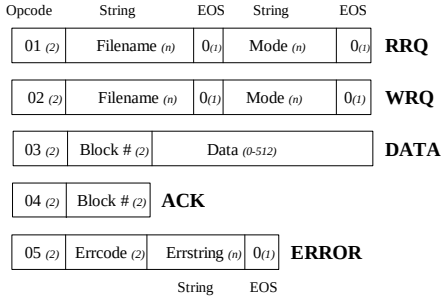
Trivial FTP (TFTP)

- RFC 1350
- a simple file transfer protocol, often used for booting from a remote file system
- no access control, directory manipulation, etc.
- UDP transport
- files are netascii or binary

Services & PDU design

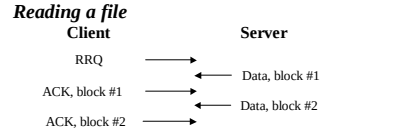
- Services
  - Read, write a file
- PDUs
  - RRQ, read request: read a file
  - WRQ, write request: write a file
  - DATA
  - ACK
  - ERROR

PDU structure and encoding

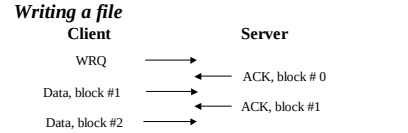


TFTP Protcol

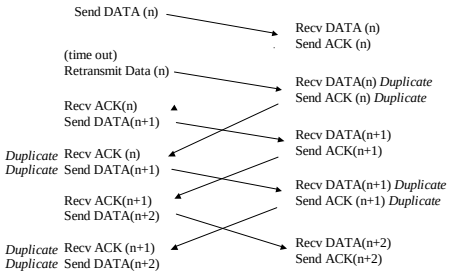
Reading a file



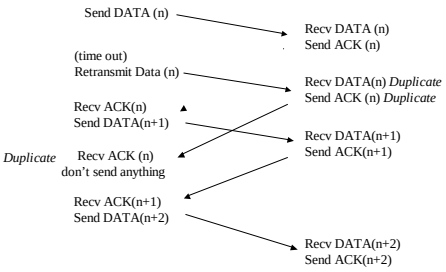
Writing a file



Sorcerer's apprentice syndrome



Correction



Implementation Issues

- How does server keep track of client address?
  - New socket created by child process on server side
  - Child process *bind*s socket to a local address
  - All subsequent messages from client sent to new address

Implementation cont'd

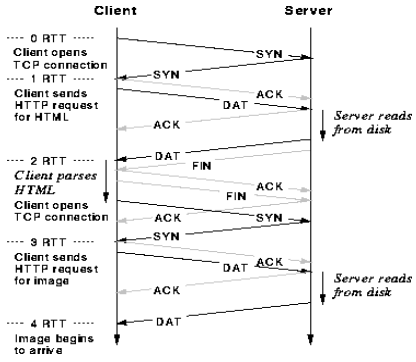
- Data formats
  - netascii -responsibility of the client & server to convert from netascii to local representation (and vice versa)
- Protocol processing
  - Finite state machine
- Security

HTTP 1.0

- Services
  - read web pages, append to web pages, write pages, etc.
  - GET, HEAD, POST, PUT, DELETE, etc.
- PDU design and encoding
  - ASCII request followed by MIME-like response
  - format specified in BNF

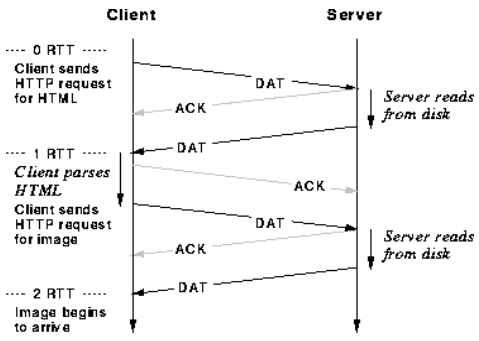
HTTP 1.0 cont'd

- Protocol
  - TCP used as transport
  - new connection for every file retrieved
    - poor performance
- Other issues
  - active content, caching, proxy servers, security



HTTP 1.1

- Several changes
  - Persistent connections - default behavior
  - Host header field in request
  - Connection header
  - See also link on class web page to a more detailed discussion
- Assignment
  - GETLIST method (not part of HTTP 1.1)
    - server should maintain a list of embedded files
    - *you have to design the request/response formats*



Application-level Protocols

- Session-layer
  - communication between processes (as opposed to hosts)
  - Example:
    - aspects of TFTP related to maintaining "pseudo-connections"
    - error-recovery and checkpointing for long-lived connections

Presentation Layer

- Handle issues related to different data representations on communicating hosts
  - big endian vs little endian, 32 bit vs 64 bit, different formats for characters, etc.
  - Usually handled in one of two ways
    - Canonical representation of data
      - XDR, ASN.1, netascii (TFTP)
    - "receiver makes right"
- Also handles encryption and decryption

MIME

- Internet email standard (RFC 822) specifies ASCII message format
- Multipurpose Internet Mail Extensions
  - continue to use the RFC 822 format but allow non-ASCII messages
  - New message headers
  - binary messages encoded using **base64** encoding or **quoted-printable** encoding