Aalto-yliopisto

Assignment 1: Chunked File Transfer (CFT)

Design a protocol
Specify the protocol
Implement the protocol

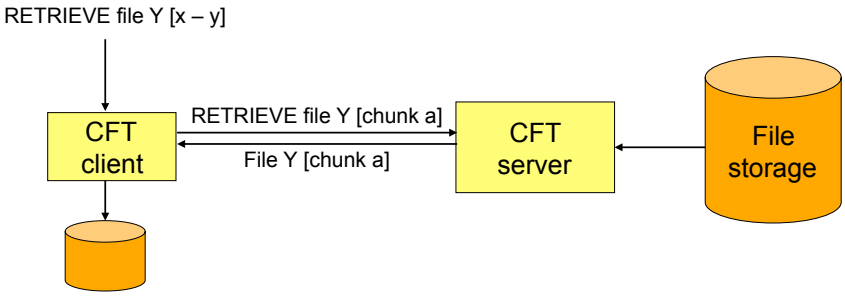
Aalto-yliopisto

File Pull: CFT

- Scenario: a file server offers files for partial download
  - Like in the web (e.g., HTTP Range: header)
  - Example: large source distributions
  - Files may be structured (you don't need to worry about the structure)
  - Clients can request all or parts of a file
    - "RETRIEVE 16384 – 32768"
    - "RETRIEVE 0 – 65536"
    - "RETRIEVE 131072 – "
    - "RETRIEVE 0 –"
  - File content retrieval shall be according to a certain chunk size
    - Fixed or dynamic (up to you)
  - The chunk size and the retrieval part need not agree
    - E.g., chunk size 1024 bytes
    - RETRIEVE 300 – 1500
    - Needs to be handled by the receiving client (not a protocol issue)
  - Clients allow for complementary download in several passes

Aalto-yliopisto

File Pull: CFT



Aalto-yliopisto

File Pull for CFT

- "Reliable" transfer of a file part from one source to a destination
  - Individual transfers
- Client mode operation
  - Initiate a transfer of part of a file from a server: receive data
- Server mode of operation
  - Wait for requests from a client
- File transmission shall take place in chunks that are individually requested
  - They may be in parallel, pipelined, or serially
- File chunk transmission should be reliable
- File chunk transmission should be somewhat performant
- File identification to be conveyed (i.e., the file name)
- File chunk identification to be chosen and conveyed (per file)
  - Needs to be persistent across multiple download
- File and chunk validation information (e.g., a checksum)
- Other information?
- Support "simulated" packet loss
  - Independently on both sender and receiver side

Aalto-yliopisto

Some Issues to Consider

- How do chunk retrieval and delivering individual chunks interact?
  - How to pace chunk retrieval and data delivery
- How to transmit data reasonably efficiently?
  - Keeping the link/path busy
- How to do error handling?
  - File does not exist?
  - Chunk beyond EOF?
- How to deal with failed file transfers?
  - What is a failed file transfer?
  - How and when do you declare something failed?
- Client side handling
  - How to know the file size if no range is specified by the user?
  - Complementary downloads across multiple runs? (like FTP)
    - What happens if the file changes on the server side? How to find out?
  - Chunk and retrieval demands do not match?

Aalto-yliopisto

CFT: Design and Specification

- Document (and motivate!) your design decisions
  - There are many possible approaches
- Write up a short specification for your protocol
  - Include sufficient detail so that one can understand and implement from it
  - Litmus test
    - Design together in your group
    - One or two of your group writes part of the spec
    - The other(s) try to understand it
    - Be critical: ask yourself what is really written there (as opposed to what might be meant)
  - No need to exaggerate on the spec though
- Do a short version for group discussion first: 3 – 4 slides
  - Send to us by 30 March 2010, 16:00 EET
  - Group discussions on 30 March, 16 – 18 E110/E111
- Update and complete your spec based upon feedback
  - Hand in the written spec by 9 April 2010 (hard deadline)
  - Try to implement your spec by 16 April 2010 (soft); needed for 2nd assignment

Aalto-yliopisto

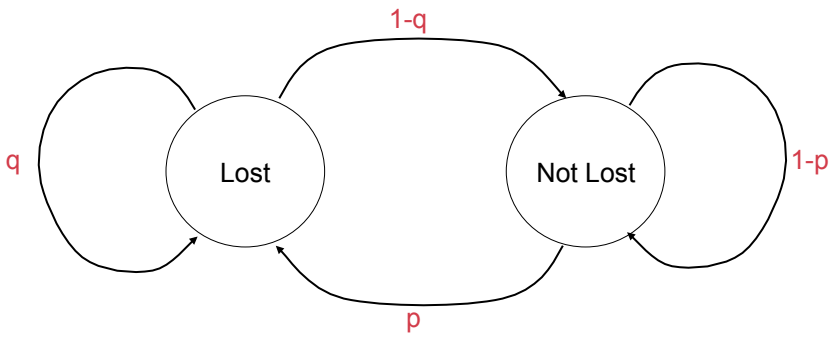
CFT: Implementation

- Realize your protocol specification in some language
- Write a single program that can act as both sender and receiver
  - Distinguished by command line options
- Simulate your own packet losses
  - Trashing packets in your code before sending or after receiving
- Test it!
  - Does it “comply” with your spec
- Document what you did and what you learned
  - How is your program structured?
  - Which were the major implementation issues?
  - Did you have to adjust your spec during the implementation?
  - What would you do differently if you started all over again?

Aalto-yliopisto

Packet loss simulation

- Choose a simple Markov chain
  - Then, we can play with dependent and independent losses



```
cft [-s] [-t <port>] [-p <p>] [-q <q>]
cft <host> [-t <port>] [-p <p>] [-q <q>] [-r start:end] <file>
```

-s:	server mode: accept incoming files from any host Operate in client mode if "–s" is not specified
<host>	the host to send to or request from (hostname or IPv4 address)
-t:	specify the port number to use (use a default if not given)
-p, -q:	specify the loss probabilities for the Markov chain model if only one is specified, assume p=q; if neither is specified assume no loss
-r:	range of data to retrieve; all if not specified
<file>	the name of the file(s) to send

Further options may be useful; up to you.

Remember to do report errors (locally and across the network) as needed.

You may want to do something useful if the user aborts either process (Ctrl-C).