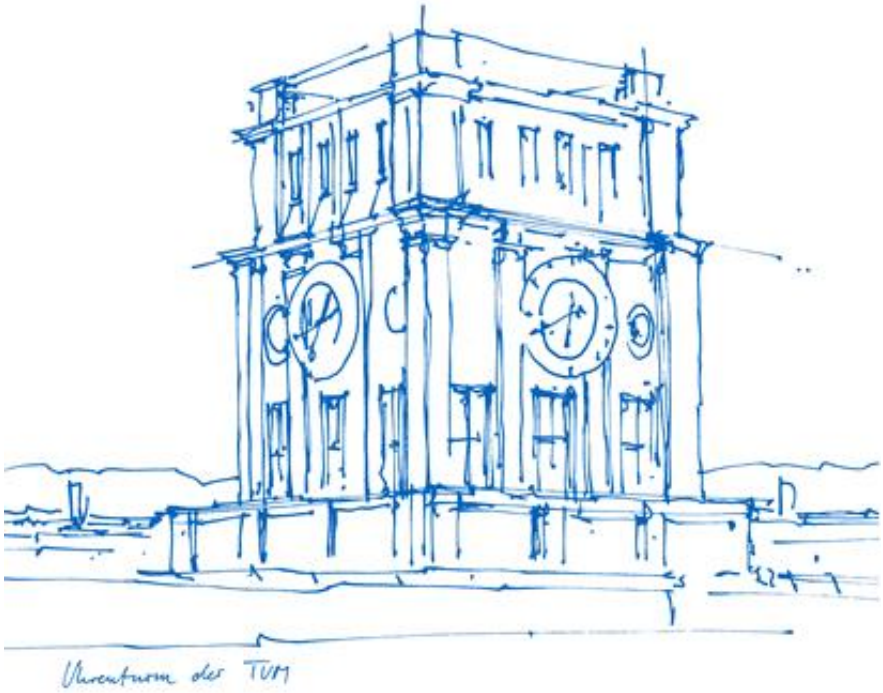
Assignment 1: Cloud Sync

Design a protocol

Specify the protocol

Implement the protocol



Scenario

- Cloud synchronization
- Client push protocol
- Client listens at a directory and automatically upload files to the server

Specification

- Use UDP: ensure reliability
- State transition
- Reliability
- Data encoding
- Connection management
- File management
  - Server decides whether it has the file/chunk or not and accepts the file upload
  - Checksum: make sure that entire file is uploaded, data integrity
- Chunk handling
  - Example: 90% of file uploaded, network connection aborts, don‘t start from beginning
- Keep in mind: Scalability
  - Client uploads multiple files at the same time

Issues

- How to do error handling?
  - File disappears
  - Loss of connection
- How to deal with failed file transfers?
  - What is a failed file transfer?
  - How and when do you declare something failed?
- Client side handling
  - Complementary uploads across multiple runs?
    - What happens if the file changes on the client side? How to find out?
  - What to do in case of long latency?
- Server side handling
  - Upload by client: file name already exists
  - Running out of memory?

Implementation

- Realize the protocol specification in your favoured language

- Write a single program that can act as both sender and receiver
  - Distinguished by command line options

- Simulate the file synchronization
  - Sufficient logging to keep track of transferred bytes

- Test it!
  - Does it “comply” with your specification

- Document what you did and what you learned
  - Which were the major implementation issues?
  - Did you have to adjust your specification during the implementation?

Interface

Client: csync [-h <hostname|ip-addr>] [-p <port>] [-f <directory-path>]

Server: csync [-s] [-p <port>]

-s	Server mode: accept incoming connections from any host
-p	Specify the port number (use a default if not given)
-f	Upload all files in that directory to the server
-h	Remote host

Regulations

- 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 it
  - Litmus test
    - Design together in your group
    - One or two of our group members writes a part of the specification
    - The other(s) try to understand it, be critical!
- Do a draft version of your protocol specification
  - Amount: paper (max. 2 pages) and key functionality for discussion: 3 – 4 slides
  - Send to us by 29th April 2018, 23:59:59
  - Group discussions on 30th April 2018
- Update and complete your specification based upon feedback
  - Submission by 14th May 2018, 23:59:59