## Network traffic

* Sending JSON objects with binary data if needed
* Wherever possible only the data of the changed block(s) is sent
* Mandatory Operations
  + Login
    - Send: op=login, name=x, pw=y
    - Receive: op=login, res=true/false, VFSList=VFSid1, VFSid2,...
  + Logout
    - Send: op=logout
  + Register
    - Send: op=reg, name=x, pw=y
    - Receive: op=reg, res=true/false, VFSList=VFSid1, VFSid2,…
  + Add VFS
    - Send: op=addVFS, data=b
    - Receive: op=addVFS, res = true/false, VFSid=123
  + Remove VFS
    - Send: op=remVFS, VFSid =123
    - Receive: op=remVFS, res = true/false
  + Retrieve VFS
    - Send: op=retrVFS, VFSid=123
    - Receive: op=retrVFS, data=b
  + Change of block(s)
    - Send: op=change, VFSid=123, {(offset=0, data=b), (offset=4, data=b) ,…}, changedFiles=/ab/c, /ab/d,…
    - Receive: op=change, versionid = 123
  + Update to current version
    - Send: op=curr, VFSid=123, oldVersionid=123
    - Receive: op=curr, data=b, currVersionid=124
* Optional Operations
  + Revert to version
    - Send: op=rev, VFSid=123, versionid=123
    - Receive: op=rev, data=b
  + Allow other users to access file/folder
    - Send: op=acc, type=add/remove, VFSid=123, user=xy, path=/abc/
    - Receive: op=acc, res=true/false

🡪 Sharing of VFS needed: If file/folder access is granted, server checks whether user sees VFS for the first time and therefore has to be notified  
Server sends: op=share, VFSid=123

## Client behaviour

* First step is to register/login; nothing can be done without having done that successfully
* Newly created VFS are automatically added to the account
* Existing VFS can be added to the account

🡪 New VFS on the server, even if already existing (not linked to this one/no sharing)

* Keeping track of the paths of the VFS files locally. If not existing, it can be downloaded from the server
* In offline mode, changes are queued to be sent to the server as soon as connected again
* Each client has to remember the versionIds of the local VFS files
* Polling changes form the server is done on a regular basis (i.e. 30s intervals). Additionally the client can have a button to allow the user to update immediately.
* After update from server received, the client has to check whether the form has to be updated by looking at the list of changed files

## Server behaviour

* Keeps track of users/pw
* Keeps track of VFS that a user has added to account
* Has the VFS files in the file system and makes the changes on them
  + Optional: The execution of changes on a VFS file could be delayed and only executed after a predefined amount of changes have happened to reduce the number of stored VFS file versions
* Each VFS update triggers creation of a new (increasing) versionid. The according change to the new version is stored, so that a client can receive all changes from a certain version to the current.
* (Optional: For each change the previous block data that has been overwritten could be stored too. This would make reverting to a previous version possible..)

## Open questions

* How and where to handle conflicts