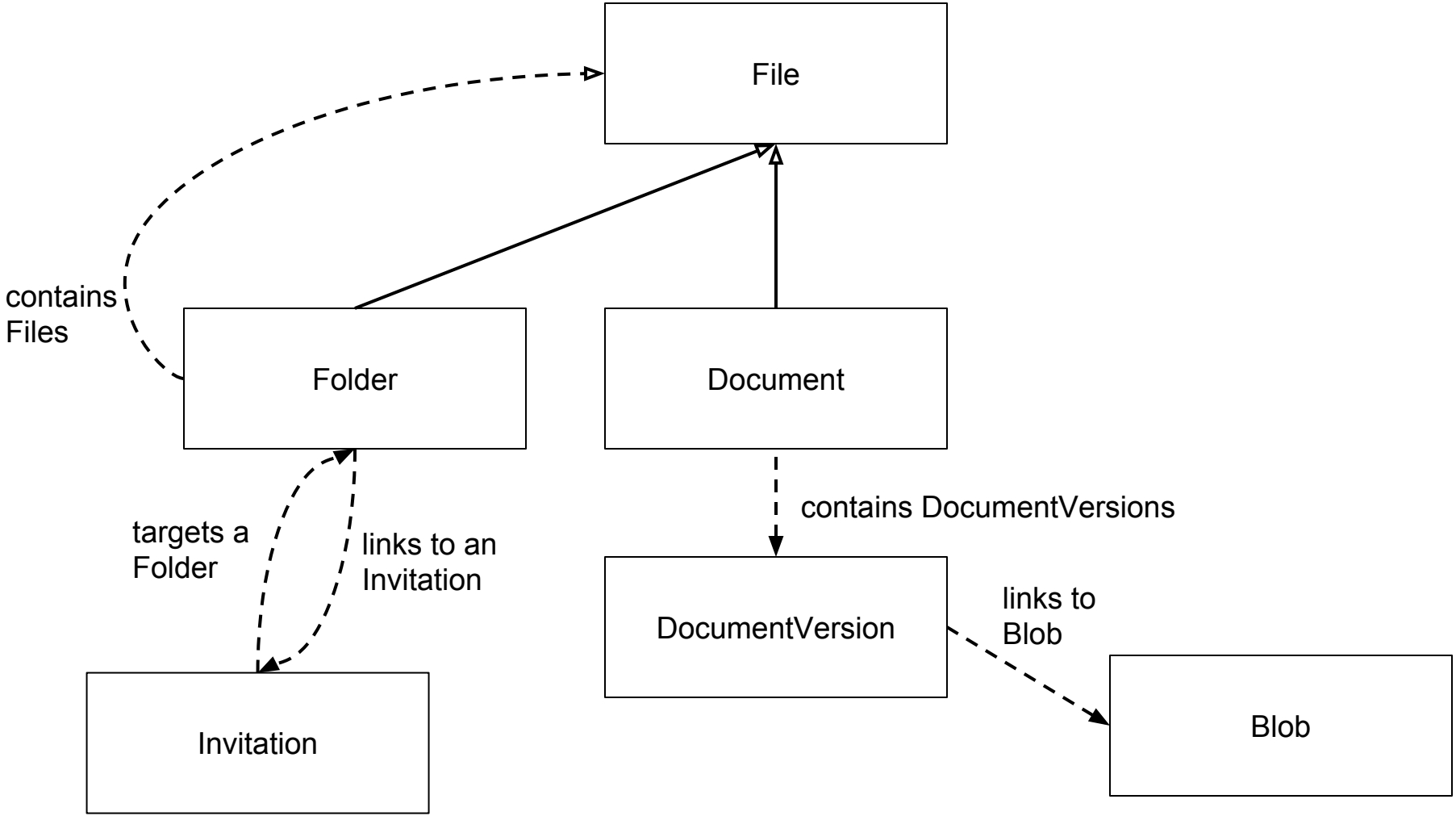
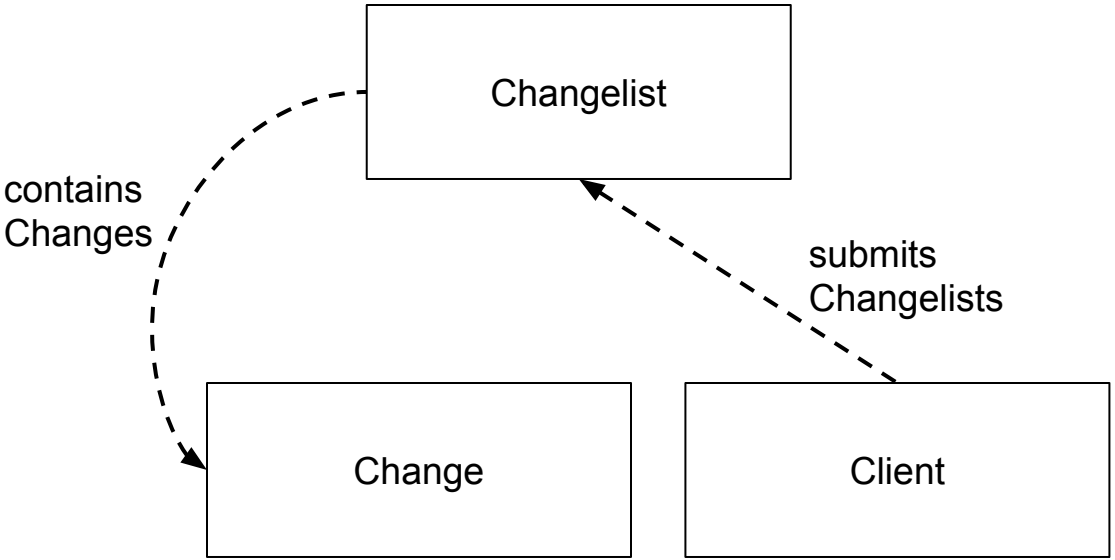
FooBox architecture

File synchronisation and backup system


File
contains
Files
Folder
Document
targets a
Folder
links to an
Invitation
contains DocumentVersions
DocumentVersion
links to
Blob
Invitation
Blob



Example:
Changelist #4 submitted by wenjia

Add /3/MYFOLDER [IsFolder]
Add /3/MYFOLDER/MY DOCUMENT.TXT [DisplayName = “My Document.txt”]
Delete /3/MYFOLDER/OLDDOCS
SetDisplayName /3/MYFOLDER/FILE.DAT [DisplayName = “fILe.DAt”]

Changelist submission

- Client sends to server:
  - **Base changelist ID**: the changelist it is currently synced up to
  - **List of client changes**
- Client receives from server:
  - Submission status
  - List of blobs that need to be uploaded (if any)
  - List of changes that have occurred since the base changelist

Changelist submission

- Server performs the following:
  1. Computes the total merged changes that have occurred since the client's base changelist.
  2. If merged changes conflict with client changes, returns **Conflict**.
  3. Transforms the client changes with respect to the merged changes (*MakeSequentialByPreserving*)
  4. If required file data is missing, returns **UploadRequired**.
  5. Locks the relevant tables and applies the client changes to the database file system. If any quota limit is exceeded, returns an **Error**.
  6. Returns **Success**, along with the merged changes.

Client sync loop

- Client performs the following every 3 seconds:
  1. Determines what changes have occurred in the last 3 seconds
  2. Submits a changelist to the server, uploading any required blobs and handling conflicts
  3. Adds the changes returned by the server to *PendingChanges*
  4. Processes *PendingChanges* by updating the local file system and downloading documents
- Step 2 occurs even if there are no local changes, because the server may have changes to send us
- This process can be paused and resumed at any time - the client will simply resume processing pending changes at the correct location

Conflict handling

- **Conflicts** arise when multiple users or multiple clients make changes to the same documents/folders
- Goal of conflict handling is to **preserve** as much data as possible, and to **inform** the user

Conflict handling

- **Definition**: two sets of changes (A and B) conflict if they do not commute, i.e. applying A then B to the file system produces a different result to applying B then A
- Example:
  - A: **Add** /3/FooBar [IsFolder]
  - B: **Add** /3/FooBar
  - Applying A then B results in a document /3/FooBar
  - Applying B then A results in a folder /3/FooBar

Conflict handling

- **Solution**: if a client submission fails due to a conflict, the client (locally) renames any conflicting documents/folders and tries again
- Example:
  - A: **Add** /3/FooBar [IsFolder]
  - B: **Add** /3/FooBar
  - A submits first, so B's submission fails
  - B renames "FooBar" to "FooBar (wenjia's conflicted copy 2014-10-30 15-02-53-133)" and tries again