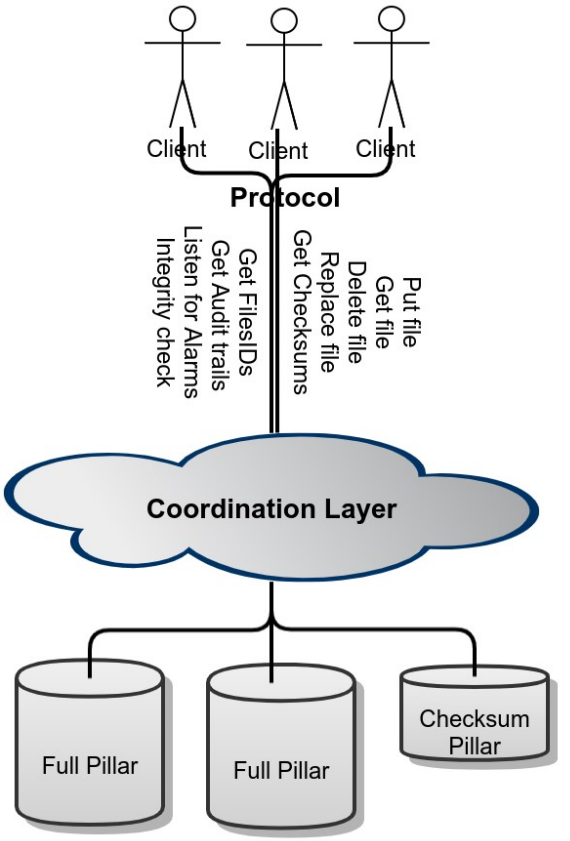
Bitrepository.org

Client training - presentation

What is the bitrepository?

- Distributed storage system
  - Designed for longterm bit-preservation
  - Multiple independent copies
- Uses messagebased asynchronous communication
- Designed to minimize common-mode errors:
  - Implementation (language, OS, etc.)
  - Organizational
  - Storage technology (disk, tapes, optical media)

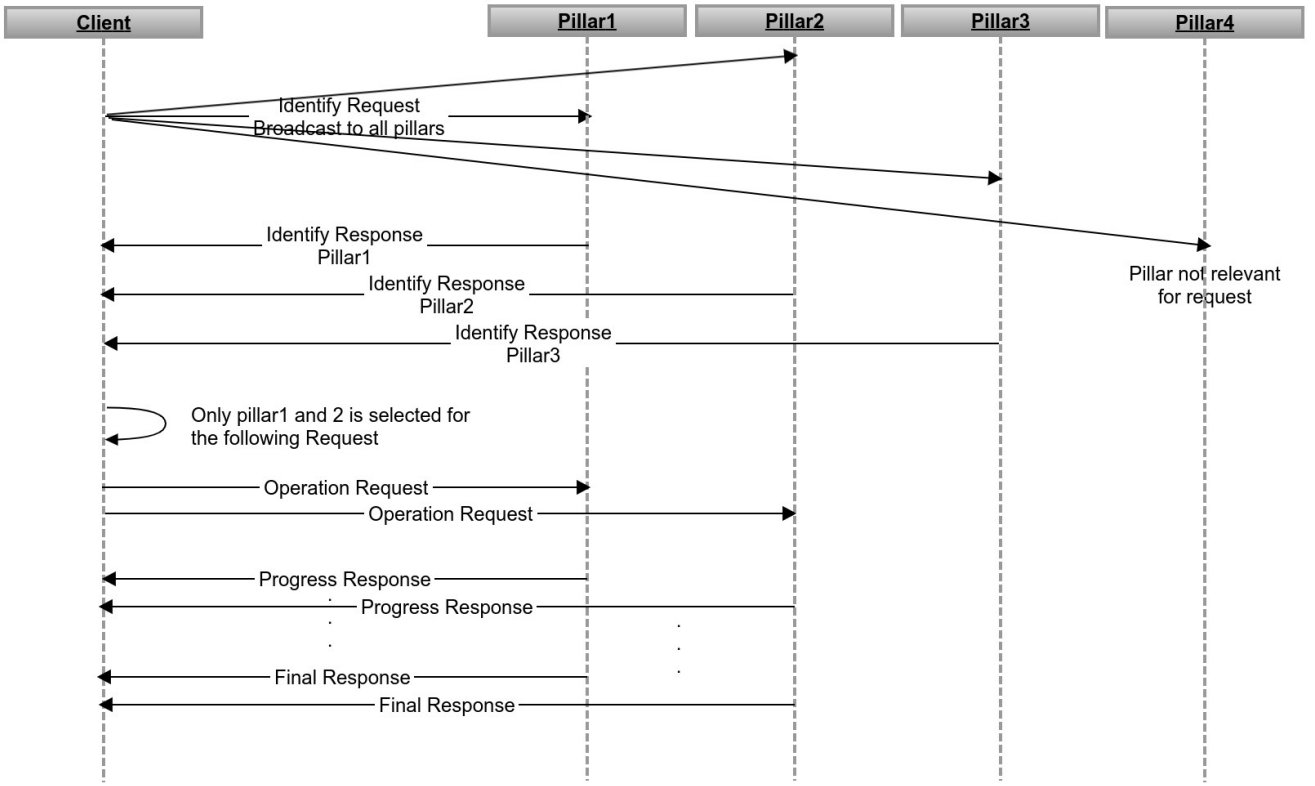


Bitrepository infrastructure

- To function the bitrepository components requires two major bits of infrastructure:
  - Messagebus
    - Transmission of messages between components
  - FileExchange
    - High level abstraction for moving files
    - Initial concept http/https (webdav)
- Enables all other components to be hidden behind firewalls with minimal open ports.

Messageflow

Normal message flow



Conversation

- Operations are handled as a ’conversation’
- A conversation starts with a identification phase
  - The identification phase is for discovery of components that the conversation can continue with
- After a succesful indentification phase, the operation/performing phase can start
- Each conversation have a unique ID

FileExchange

- Common ground to exchange files between components (clients, pillars)
  - Conceptually not limited to a specific protocol
  - Clients responsibility to specify fileexchange url in communication

Security and permission model

- Security is centered around x509 certificates
- RepositorySettings defines the systems full trust.
  - Trusted certificates
  - Permission matrix for each certificate
- Messages are cryptographically signed to:
  - Authenticate a message
  - Authorize an operation

Client library

- One for each protocol primitive
  - PutFileClient, GetChecksumsClient, etc.
- Abstracts most communication complexities away
  - Message signing
  - Messageflow
  - Error senarios
- Interaction with operations are event driven
- A single client can handle multiple concurrent operations

Client library - events

- Good case events
  - IDENTIFY_REQUEST_SENT
    - Start of a conversation (identification phase)
  - COMPONENT_IDENTIFIED
    - One for each identified component during identification phase
  - IDENTIFICATION_COMPLETE
    - Indication that the identification phase is complete
  - REQUEST_SENT
    - Start of the operation/performing phase
  - PROGRESS
    - A component have delivered a progress response
  - COMPONENT_COMPLETE
    - A component have sucessfully completed its operation
  - COMPLETE
    - The entire operation completed sucessfully

- Bad case events
  - COMPONENT_FAILED
    - A component failed its operation
  - FAILED
    - The entire operation completed in a failed stated
  - IDENTIFY_TIMEOUT
    - The identification phase timed out (not all expected contributors responded in time)
  - WARNING
    - Something is not quite ok, could for instance be a invalid message.

Client library – factories

- The client library have helper classes to help:
  - Load settings
  - Create MessageBus connection
  - Create the various clients
- Settings are loaded in a static way
  - Provides easy access to settings (the upside)
  - Limits a single JVM to interact with a single bitrepository (the caveat)
- MessageBus connection is worry free:
  - Setup is handled by client factory
  - Closing can be done by MessageBusManager, but should only be done when no more clients are needed in the JVM.

Client library – result paging

- Operations such as GetFileIDs, GetChecksums and GetAuditTrails employs paging to handle long lists of data.
  - Ensures that components scales to millions of files without having to handle all of it at once.
  - One request may therefore only deliver a partial result.
- Paging is done using timestamps
  - GetAuditTrails also by sequence number
- Timestamp paging is inclusive
  - I.e. the last result of the previous operation should be found in the start of the current.
- The component with the smallest pagesize (maxResults) wins
  - i.e. if a pillar is capable of delivering 10.000 results but a client only asks for 1000 pr request, the maximum number of results will be 1000.

Client traning - code

- The code for the training can be found at
  - https://github.com/bitrepository/client-training

Exercises

1) Get client-training project up and running
2) Implementation of GetFileIDs
3) Implementation of GetFile
4) Implementation of PutFile

Exercises

- The exercises will be based on the client-training repository
- The master branch contains runnable code, but only with partial implementation (Operations Get, Put and GetFileIDs are missing, only GetChecksums exists)
- The branch ‘solution-proposal’ have working implementation of all the implemented operations
- To make the code run, configuration files are needed (RepositorySettings.xml, ReferenceSettings.xml, client-certificate.pem), they should be obtained from a working BitRepository

Exercise 1

- Clone the repository: https://github.com/bitrepository/client-training
  - The project contains boiler plate code for a commandline program, helper classes and a simple implementation for the GetChecksums functionality
- Build the code (master branch)
- Configure the commandline client (product of the build)
- Run a get checksums request on an existing file

Exercise 2

- Implement the GetFileID
  - Start implementation in GetFileIDsAction class
- The implementation should:
  - Request FileIDs from just a single pillar
  - Implement paging of results. Use a page size of 100 results / operation
  - Print list of FileIDs to stdout

Exercise 3

- Implement the GetFile
  - Start implementation in GetAction class
- The implementation should:
  - Request a file from the fastest pillar
  - Use a unique location on the FileExchange (to avoid collitions)
  - Clean up on the FileExchange after the file have been downloaded
  - Not allow to overwrite an existing file on the filesystem

Exercise 4

- Implement the PutFile
  - Start implementation in PutAction class
- The implementation should:
  - Use a unique location on the FileExchange
  - Not upload the file to the FileExchange before identification have completed
  - Cleanup on the FileExchange after the operation