Access Control in a Decentralized Collaboration Platform

Nicolas Ritter

Professor:
Bryan Ford
DEDIS

Supervisor:
Kirill Nikitin
DEDIS
Introduction - Peerdoc platform
Introduction - The cost of centralization

- Having to share potentially sensitive data with a third party which may or may not be trustworthy
- Having to rely on a central server, which is a single point of failure
- Not having local control and ownership of the data

A decentralized, peer-to-peer approach removes the central server in favor of peers keeping a local state of the document. But this comes with challenges...
Challenges of decentralized access control

- No central authority to check users’ permissions
- Possibility of network partition
- Modifications might not be received in the right order

The state of the system needs to eventually converge regardless of these challenges.
Goals

- **Access control**
  - Users need permissions to edit/view a document
  - Permissions can be added/removed

- **Recovering from partitions and dynamically joining the network**
  - Catching up on the state of the document

- **General improvements**
  - Database
  - Communications
  - Switching between documents
Structure of the system

ABTU implemented by Damien Aymon

Web interface by Rehan Mulakhel

Changes from previous work:

- Access control
- Back-end database
- Redone network communications
Background

Operational transformation: Modifications to the document are expressed in terms of operations (e.g. “insert ‘a’ at position 1”)

ABTU algorithm: ABTU orders and integrates text operations from multiple sources which can be concurrent

Optimistic acceptance: Operations are applied optimistically, and rolled back if necessary
Access control design

- Access control operations and text operations do not wait on each other
- Text operations are accepted/rejected based on the local access control state for the document
- Access control operations specify the point at which they become effective (relative to text operations)
Access control operation

From: user1
Target: user3
Permissions: 4
Timestamp: [1,0,0,0]

From: user2
Target: user4
Permissions: 6
Timestamp: [1,1,0,0]

From: user1
Target: user3
Permissions: 0
Timestamp: [2,1,0,0]
Access control operation - Permissions

4 - Read-only

6 - Read/write

7 - Administrator

0 - None (removal)
Operation canceling

Permissions removed for A

Origin → Operation from A → Operation from A → Operation from B
Operation canceling

Permissions removed for A

Origin -> Operation from A -> Operation from A -> Operation from B
Operation canceling

Permissions removed for A

Origin ➔ Operation from A ➔ Operation from B
Joining or recovering from partitions

Catch-up mechanism based on statuses:

- A status contains the state of the local vector clock
- Upon receiving a status, a peer sends its source the operations they lack
- Peers send their status when joining, or when another peer is ahead of them
- This allows peers to catch up with the state of the document when joining or when a network partition is reconnected
Concurrent operations

This can happen when operations are generated simultaneously at different sites, but also when there is a network partition.

ABTU handles these cases for text operations, but what about access control? We need *deterministic* rules for ordering concurrent access control operations.
Priority rules for access control

- Priority to operations that are closer to the origin (time 0)
- Stricter permissions override higher permissions in case of conflict
- If possible, execute an operation from peer i before an operation which removes peer i’s rights
- Use lexicographic order on the source of the operation as last resort
Integrating concurrent operation - example
Integrating concurrent operation - example
Catch-up performance
Future work

- **Performance**
  - Optimize communication between back-end and front-end and between peers
  - State snapshots instead of keeping track of the entire history of operations
  - Reduce database writes

- **Encryption**
  - Document-specific symmetric key
  - Ability to change the key when a user is removed

- **Interface and usability**
  - Logging in
  - Sharing documents