replikativ: open P2P replication with CRDTs

Christian Weilbach - Universität Heidelberg
christian@replikativ.io

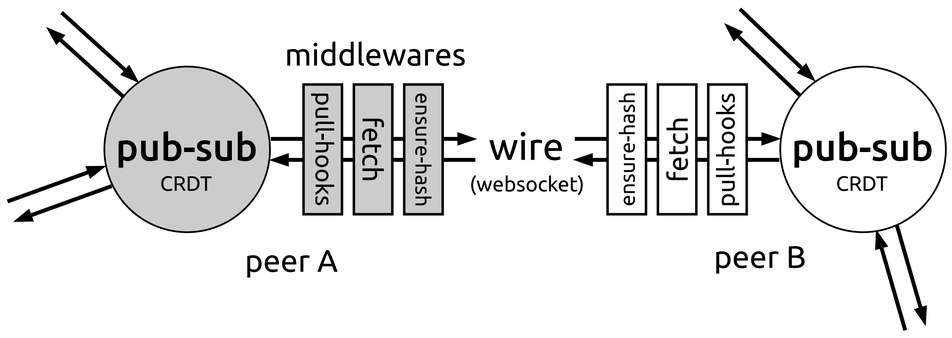
Konrad Kühne - Universität Heidelberg
konrad@replikativ.io

Annette Bieniusa - TU-Kaiserslautern
bieniusa@cs.uni-kl.de

Motivation

- Open exchange of state between apps possible (not only API)
- *Offline* availability
- CRDT[2] semantics, e.g. for *sets* and *maps*
- All logic frontend, *no network topology* management, *no backend* necessary
- Cross-platform: JVM, Android (planned), JavaScript (Browser, nodejs, iOS),
- Joint system for *database* and *binary* data
- Familiar *git-like* API with CDVCS: explore tradeoffs between *availability* and *conflict-resolution,* reason about causal history

P2P Networking



- *Global namespace* for users and CRDT ids
- Flexible *authentication*, from trust-based to public-private key authenticated (planned)
- *Gossip-based* broadcast as open P2P basis, MST techniques like plum tree planned
- Efficient *anti-entropy* [3]
- Cloud extension to the *edges* (Browser & mobile)

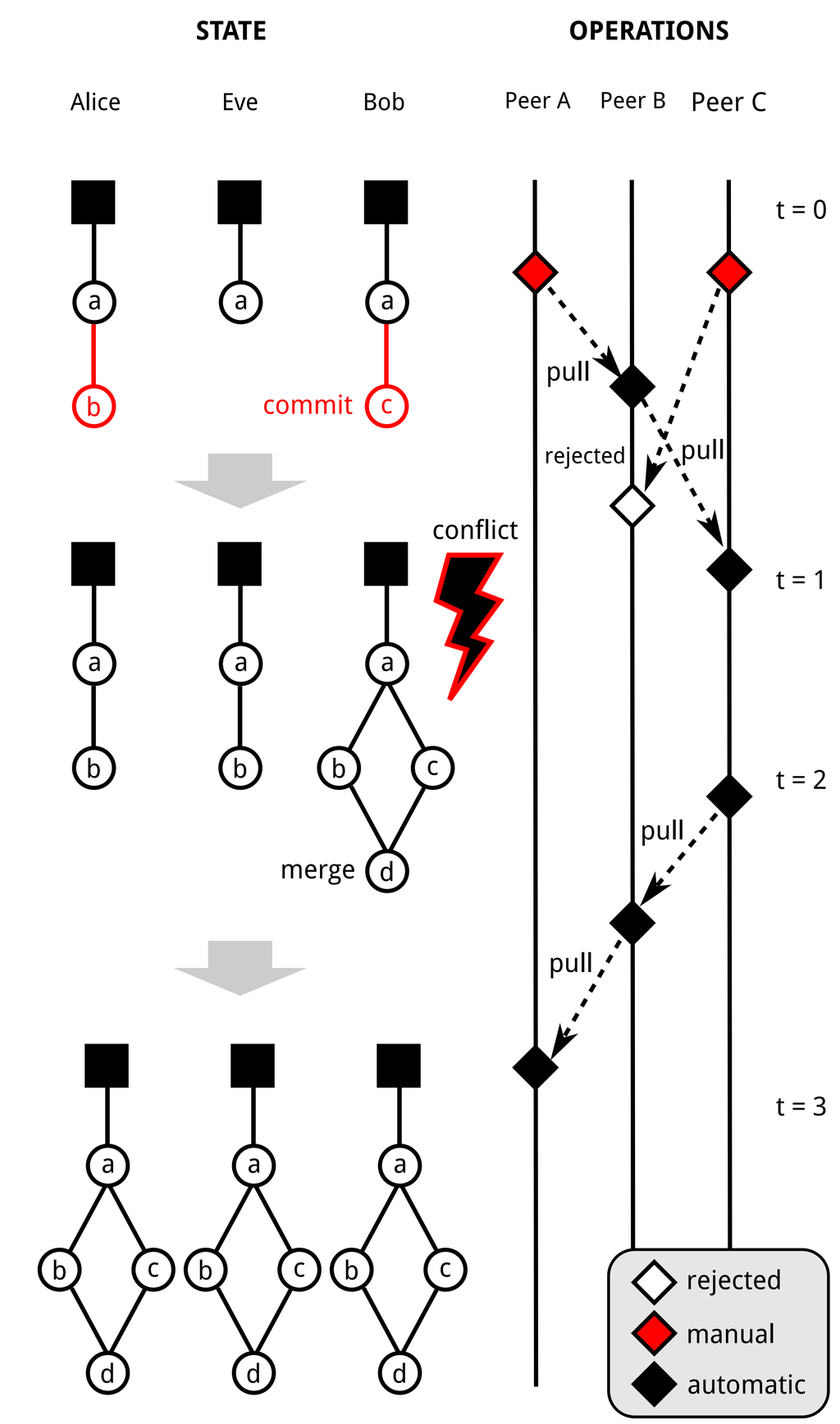
Confluent DVCS [1]

In a DVCS (distributed version control system) like *git,* the order of events is captured by an add-only, monotonic DAG of commits which represents an identity. The graph is monotonically growing and can be readily implemented as a CRDT [2]. To track the identity, we need to point to the head(s) in the graph. In a downstream update operation with head **a**, e.g. one reflecting a commit **b**, the heads are now {**a, b**}. This is resolved in a DVCS by a *lowest common ancestor* search (LCA). Whenever we want to resolve its value, i.e. its history, we need to remove all stale ancestors and either have only one head or a conflict of multiple ones. We therefore remove stale ancestors in the set of heads on downstream operations, so we do not need to use a CRDT for the set of heads. Combining the DAG, the set and LCA completes our CRDT which we refer to as confluent DVCS or CDVCS.

Coordination example

- Everybody forks the master repository and commits locally, even when *offline*
- A central peer pulls in *non-conflicting* changes
- every user *pulls* and *merges* from the central repository
- Allows naive local prototyping + conflict-resolution with *reduced availability* in conflict cases
- Works well for *low write frequency* (apps)
- Different, decoupled tradeoffs in combination with *conflict-free* CRDTs and other coordination schemes to *gradually scale up*

Strong consistency with CDVCS and (semi-)automatic merging in topiq [4]:



References

[1] C. Weilbach, K. Kühne and A. Bieniusa. Decoupling conflicts for configurable resolution in an open replication system. arXiv:1508.05545v2. Jan. 2016.
[2] M. Shapiro, N. Preguica, C. Baquero, and M. Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Rapport de recherche RR-7506, INRIA, Jan. 2011.
[3] Paulo Sérgio Almeida, Ali Shoker and Carlos Baquero. Efficient State-based CRDTs by Delta-Mutation. arXiv:1410.2803v2. Mar. 2015
[4] https://topiq.es