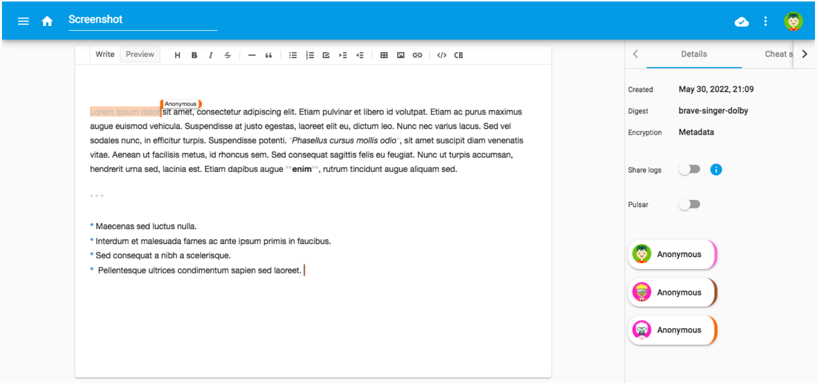
Conflict-free Replicated Data Types (CRDTs)

An Overview

Matthieu Nicolas (`matthieu.nicolas@inria.fr`)

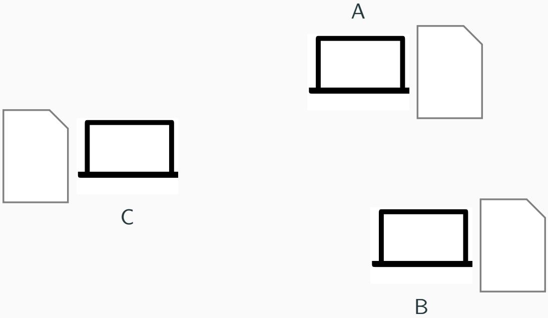
02/05/2024

MUTE [Nic+17]

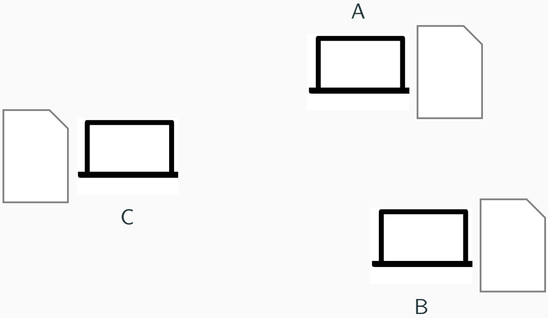


- Peer-to-Peer (P2P) application [Kle+19]
- Allow to edit collaboratively text documents
- Ensure ownership and privacy of data

Data replication in P2P systems

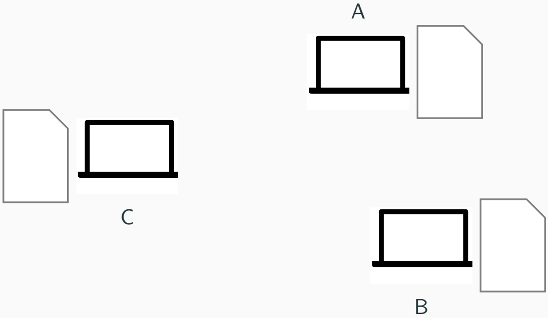


Data replication in P2P systems



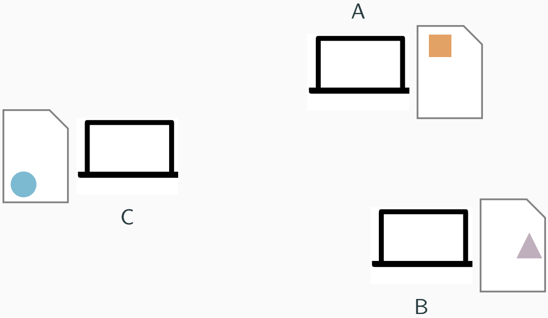
- Nodes may be disconnected

Data replication in P2P systems



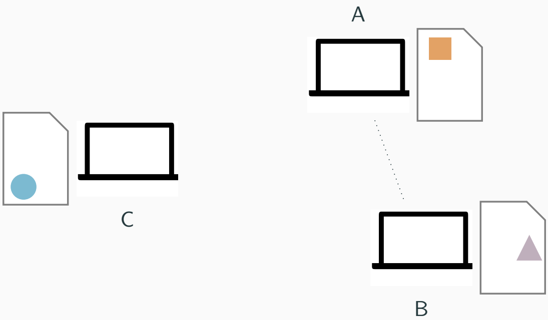
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



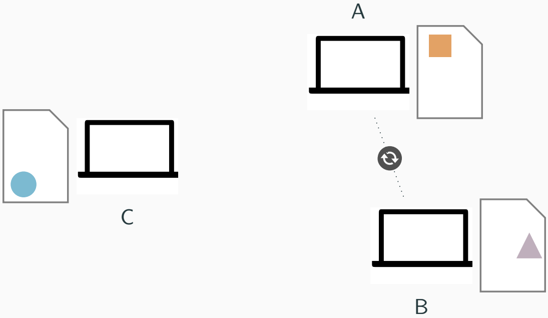
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



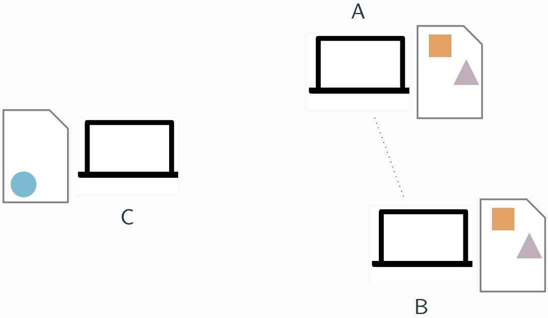
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



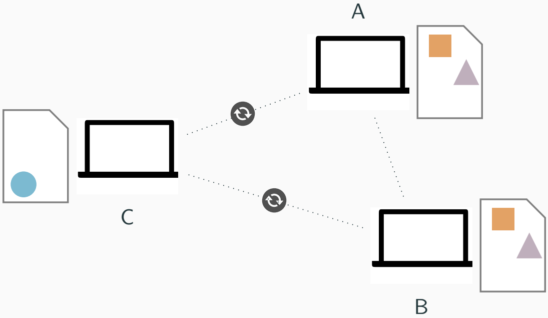
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



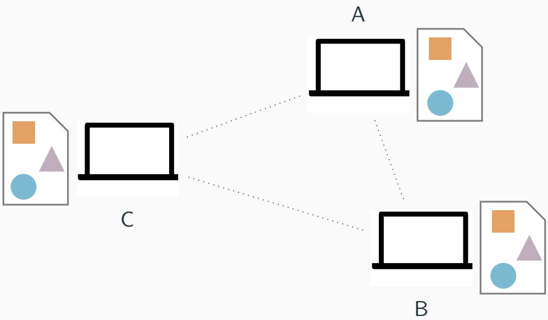
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



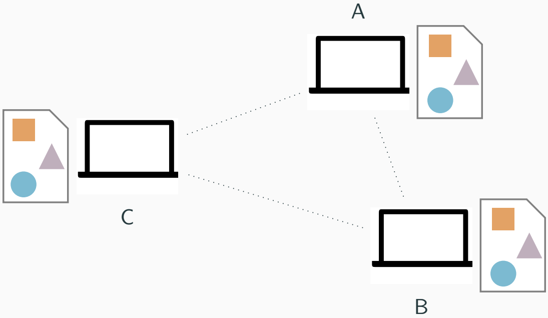
- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

Data replication in P2P systems



- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)
- Must ensure Eventual Consistency [Ter+95]...
- ...Despite different integration orders of updates

Data replication in P2P systems



- Nodes may be disconnected
- Allow nodes to work without prior or current synchronous coordination (i.e. consensus)

- Must ensure Eventual Consistency [Ter+95]...
- ...Despite different integration orders of updates

Require *conflict resolution mechanisms*

Conflict-free Replicated Data Types (CRDTs) are a family of conflict resolution mechanisms

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- New specifications of existing Data Types, e.g. *Set* or *Sequence*
- Embed natively conflict resolution mechanisms

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- New specifications of existing Data Types, e.g. *Set* or *Sequence*
- Embed natively conflict resolution mechanisms

Properties of CRDTs

- Enable modifications without coordination
- Ensure Strong Eventual Consistency

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- New specifications of existing Data Types, e.g. *Set* or *Sequence*
- Embed natively conflict resolution mechanisms

Properties of CRDTs

- Enable modifications without coordination
- Ensure Strong Eventual Consistency

Strong Eventual Consistency

Nodes that integrate the same set of updates reach equivalent states, without additional actions or messages

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- New specifications of existing Data Types, e.g. *Set* or *Sequence*
- Embed natively conflict resolution mechanisms

Properties of CRDTs

- Enable modifications without coordination
- Ensure Strong Eventual Consistency

Strong Eventual Consistency

Nodes that integrate the same set of updates reach equivalent states, without additional actions or messages

- Rely on the lattice theory . . .
- . . . More specifically, CRDTs are join-semilattice

Design of CRDTs

- Several CRDTs may be designed for a given data type . . .
- . . . Each offering different trade-offs

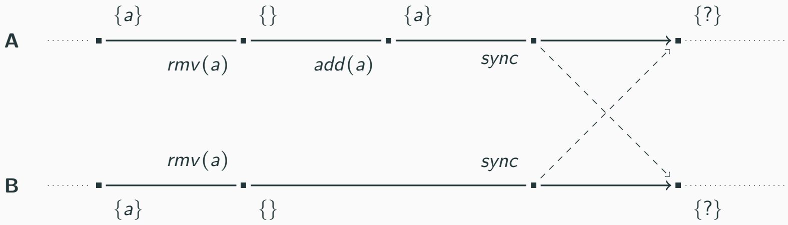
What impact the design of a given CRDT are its [Pre18]

- Conflict Resolution Semantics
- Synchronisation Model

- Impact their overhead in terms of computation, memory and bandwidth

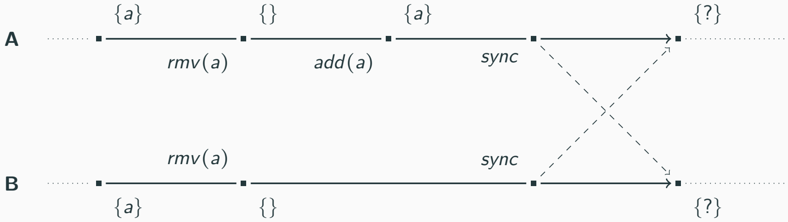
Conflict Resolution Semantics

- Distributed setting allows new scenarios



- Results of these executions are undefined
- Designing a CRDT consists in defining its behaviour in such cases

Conflict Resolution Semantics - Case study of the Set



Several semantics were proposed:

- *Add-Wins*: $add(a)$ has priority over concurrent $rmv(a) \implies \{a\}$
- *Remove-Wins*: $rmv(a)$ has priority over concurrent $add(a) \implies \{\}$
- *Causal-Length* [YR20]: The last action of the longest chain of sequential updates determines the presence (or not) of the element $\implies \{a\}$

Synchronisation Models

To converge

- Nodes have to propagate changes ...
- ... And integrate those of others

Several approaches proposed [Sha+11]

- State-based synchronisation
- Operation-based synchronisation

Synchronisation Models

To converge

- Nodes have to propagate changes . . .
- . . . And integrate those of others

Several approaches proposed [Sha+11]

- State-based synchronisation
- Operation-based synchronisation
- Delta-based synchronisation [ASB18]
    - "Best of the two worlds" approach

Synchronisation Models

To converge

- Nodes have to propagate changes ...
- ... And integrate those of others

Several approaches proposed [Sha+11]

- State-based synchronisation
- Operation-based synchronisation
- Delta-based synchronisation [ASB18]
    - "Best of the two worlds" approach

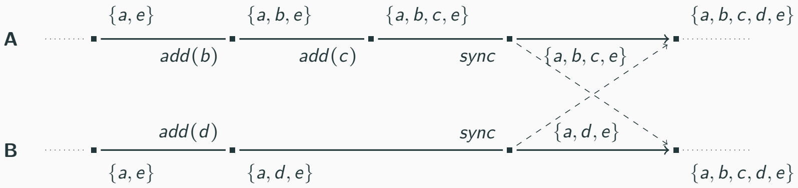
An aparté about lattice theory

Properties of join-semilattices

- States of the join-semilattices are partially ordered according to relation $\leq$
- Updates produce new states by inflation, i.e. greater to previous ones according to $\leq$
- Exists a function *join* that, given any pair of states, generates the minimal state greater or equal to both given states according to $\leq$
- Exists a set of minimal states of the join-semilattice, the *irreducible elements*

State-based synchronisation

- Send periodically current state to other nodes



- Upon reception, computes new state by merging received state with current one using `merge` function
- With `merge`, a commutative, associative and idempotent function

State-based synchronisation - ???

Strengths

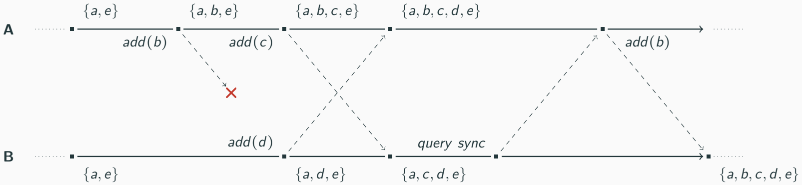
- No assumptions on the network reliability
- i.e. messages may be lost, re-ordered or duplicated w/o impact

Limits

- States difficult to design
    - e.g. how to represent efficiently deletion of elements?
- States expensive to broadcast
- `merge` expensive

Operation-based synchronisation

- Encode updates as arbitrary messages, called *operations*
- An operation correponds to one or several irreducible elements



- Upon reception, apply operations on current state
- Concurrent operations must be commutative

Operation-based synchronisation - ???

Strengths

- Designing operations is straightforward
- Operations usually cheap to broadcast and apply

Limits

- Hides/delegates complexity to delivery of operations
    - i.e. requires specific delivery order of operations
    - e.g. insertion of an element before its deletion
- Have to pair Op-based CRDTs with a delivery service to handle network failures
    - To re-order and/or de-duplicate operations
    - To retrieve lost operations using anti-entropy mechanisms

Bibliographie i

[Kle+19] Martin Kleppmann et al. **"Local-First Software: You Own Your Data, in Spite of the Cloud".** In: *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2019. Athens, Greece: Association for Computing Machinery, 2019, pp. 154–178. ISBN: 9781450369954. DOI: 10.1145/3359591.3359737. URL: https://doi.org/10.1145/3359591.3359737.

[Nic+17] Matthieu Nicolas et al. **"MUTE: A Peer-to-Peer Web-based Real-time Collaborative Editor".** In: *ECSCW 2017 - 15th European Conference on Computer-Supported Cooperative Work*. Vol. 1. Proceedings of 15th European Conference on Computer-Supported Cooperative Work - Panels, Posters and Demos 3. Sheffield, United Kingdom: EUSSET, Aug. 2017, pp. 1–4. DOI: 10.18420/ecscw2017_p5. URL: https://hal.inria.fr/hal-01655438.

[Ter+95] Douglas B Terry et al. **"Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System".** In: *SIGOPS Oper. Syst. Rev.* 29.5 (Dec. 1995), pp. 172–182. ISSN: 0163-5980. DOI: 10.1145/224057.224070. URL: https://doi.org/10.1145/224057.224070.

[Sha+11] Marc Shapiro et al. **"Conflict-Free Replicated Data Types".** In: *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems*. SSS 2011. 2011, pp. 386–400. DOI: 10.1007/978-3-642-24550-3_29.

[Pre18] Nuno M. Preguiça. **"Conflict-free Replicated Data Types: An Overview".** In: *CoRR* abs/1806.10254 (2018). arXiv: 1806.10254. URL: http://arxiv.org/abs/1806.10254.

[YR20] Weihai Yu et al. **"A Low-Cost Set CRDT Based on Causal Lengths".** In: *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*. New York, NY, USA: Association for Computing Machinery, 2020. ISBN: 9781450375245. URL: https://doi.org/10.1145/3380787.3393678.

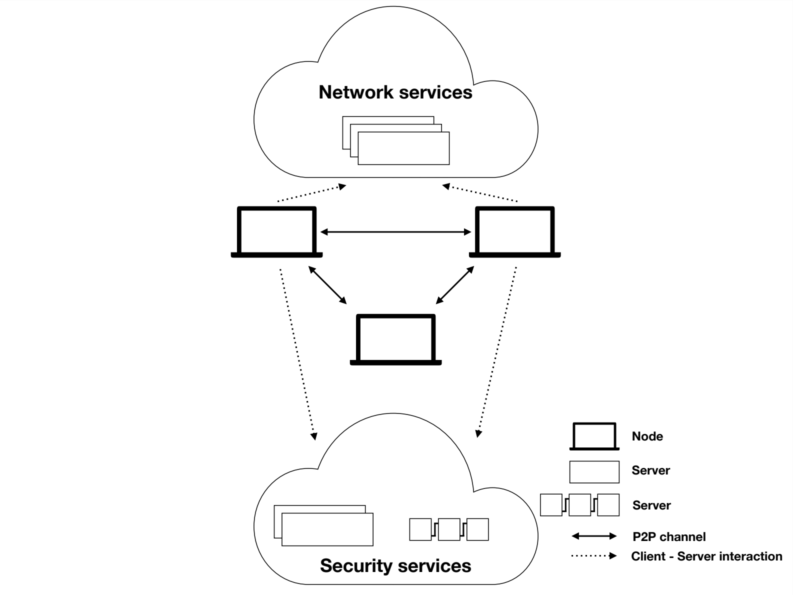
[ASB18] Paulo Sérgio Almeida et al. **"Delta state replicated data types".** In: *Journal of Parallel and Distributed Computing* 111 (Jan. 2018), pp. 162–173. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2017.08.003. URL: http://dx.doi.org/10.1016/j.jpdc.2017.08.003.

Back-up slides

Synchronisation Models - Recap

	State-based	Op-based	Delta-based
Integrate updates by merging states	✓	✗	✓
Integrate updates by irreducible elts	✗	✓	✓
Handle natively network failures	✓	✗	✓
Suited for real-time systems	✗	✓	✓

MUTE System Architecture



MUTE Software Architecture