Just-Right Consistency

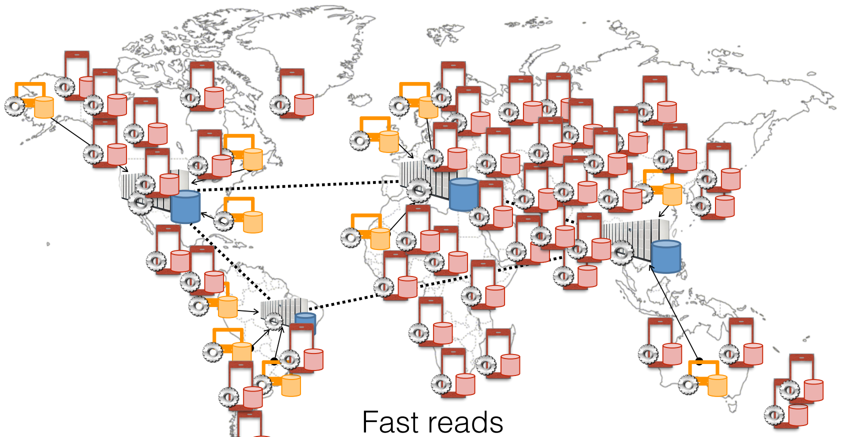
Living on the edge, safely

Marc Shapiro, Sorbonne-Université—LIP6 & Inria
Annette Bieniusa, U. Kaiserslautern
Nuno Preguiça, U. Nova Lisboa
Christopher Meiklejohn, U. Catholique de Louvain
Valter Balegas, U. Nova Lisboa

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

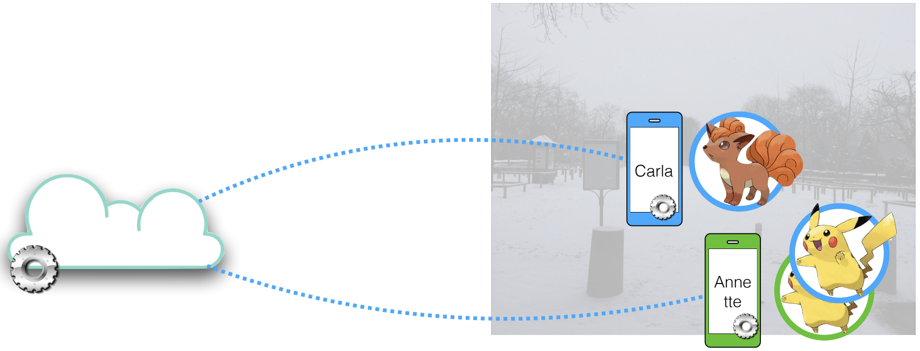
Inria informatics mathematics

Far edge



Fast reads
Replicate updates
consistent ∩ available = ∅

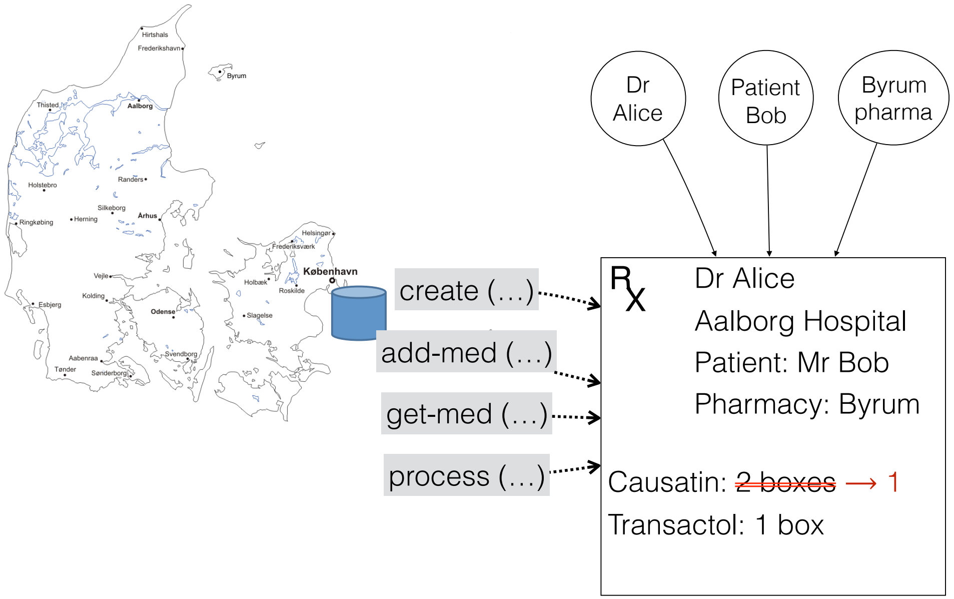
Consistency: Pokémon Go



Data at the edge, cloud-centric
✔availability, ✗latency, ✗bandwidth

- poor interaction
- anomalies

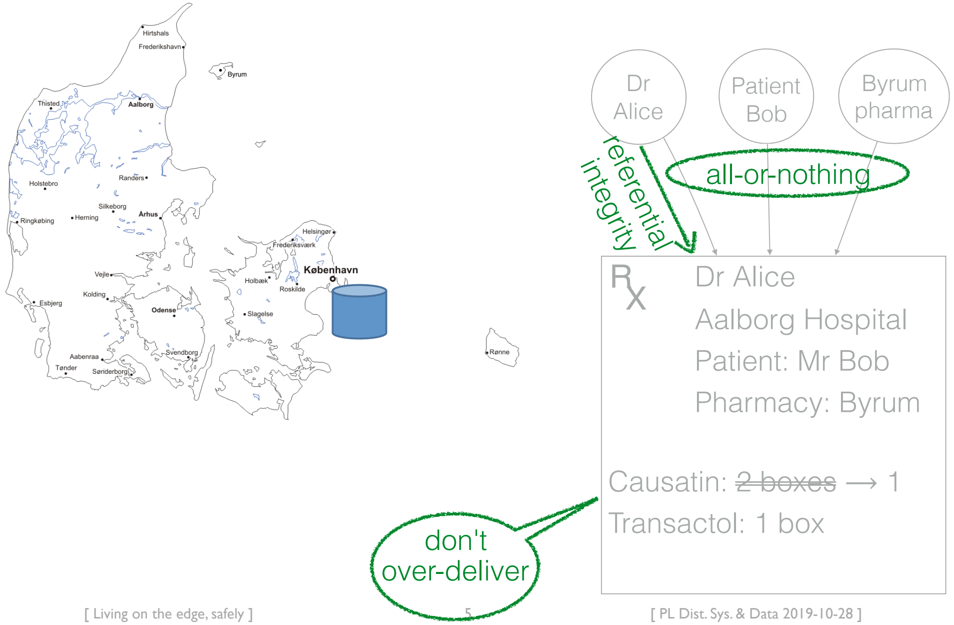
FMK *Fælles Medicinkort*



℞ Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: ~~2 boxes~~ → 1
Transactol: 1 box

Correctness invariants



R_X Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: ~~2 boxes~~ ⟶ 1
Transactol: 1 box

What is the right consistency model?

No “one-size-fits-all” consistency model

- All-CP: over-conservative
- All-AP: risk anomalies

Consistency options?

- Hard to choose the right one
- What happens when switching?

Just-Right Consistency

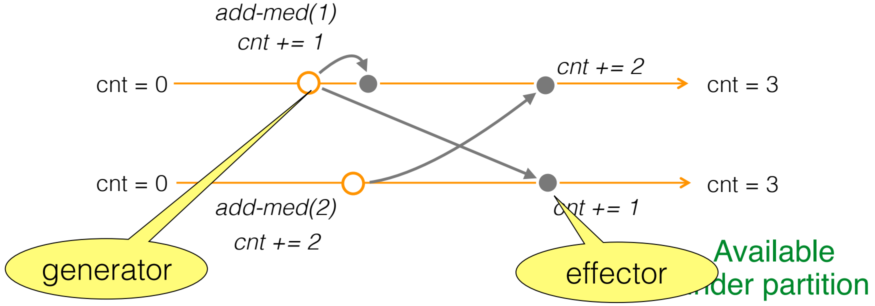
Sequential version is correct

Tailor consistency to *application invariants*

As available as possible & as consistent as necessary

- Asynchronous by default
- Synchronise (only) when <u>required</u> by application invariants
- Co-design application & protocol
- Correct by construction

Conflict-free Replicated Data Type



Concurrent, asynchronous updates

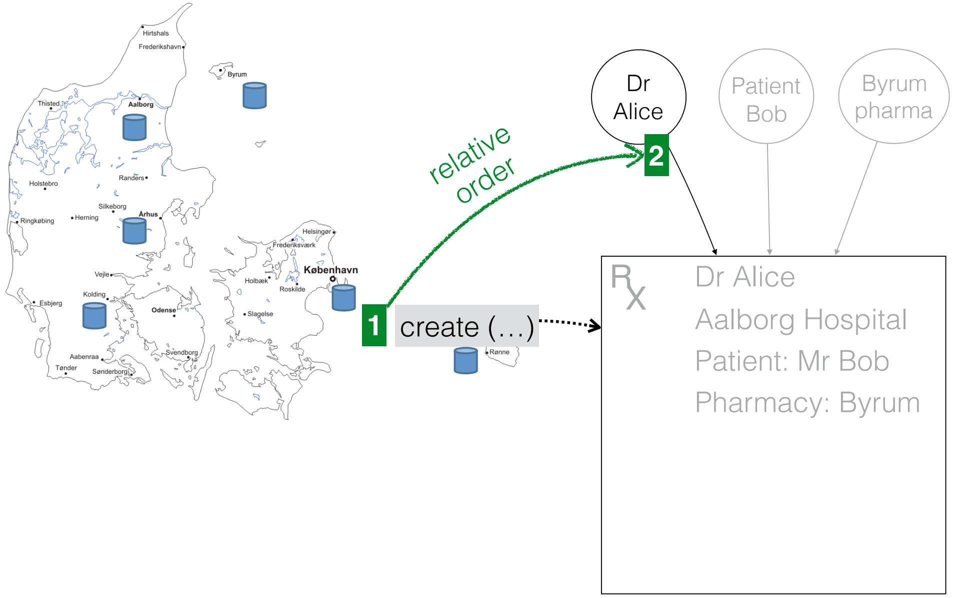
- Standard register model: assignments ⟹ CP
- AP ⟹ concurrent updates + merge

CRDT: register, counter, set, map, sequence

- Plug-in replacement for sequential type

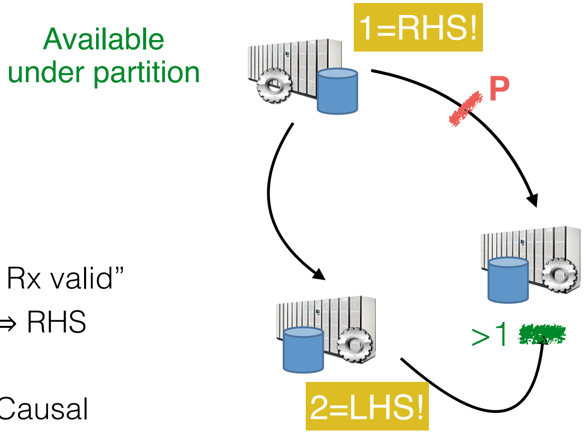
Rx.Patient: write_once_reg. Rx.Meds: set

Ordered-item invariant



with CC animation

Causal consistency



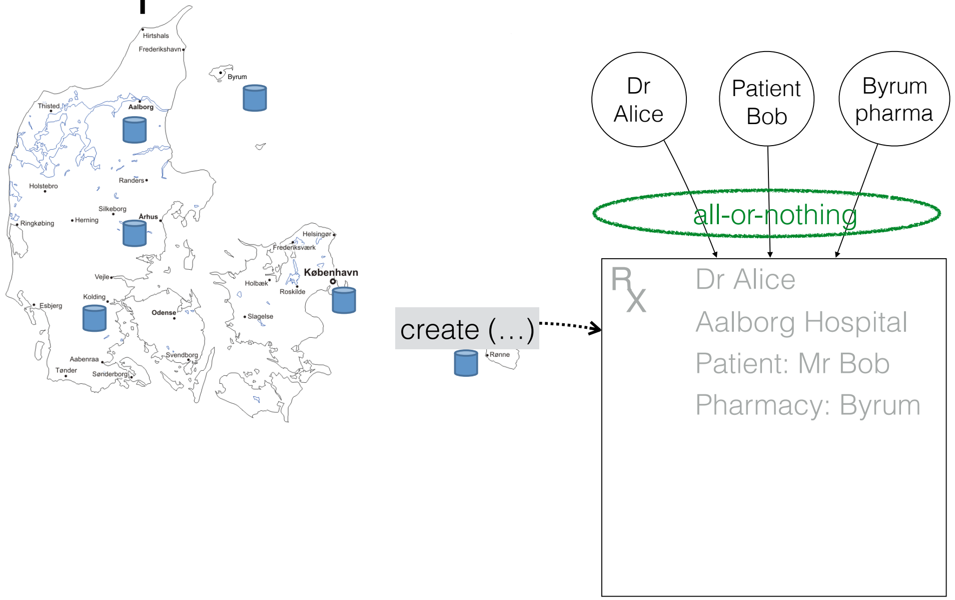
create-p before *add-med*

- "Bob points to Rx ⟹ Rx valid"
- General case: LHS ⟹ RHS
- pattern: RHS!; LHS!

Deliver in the right order: Causal Consistency

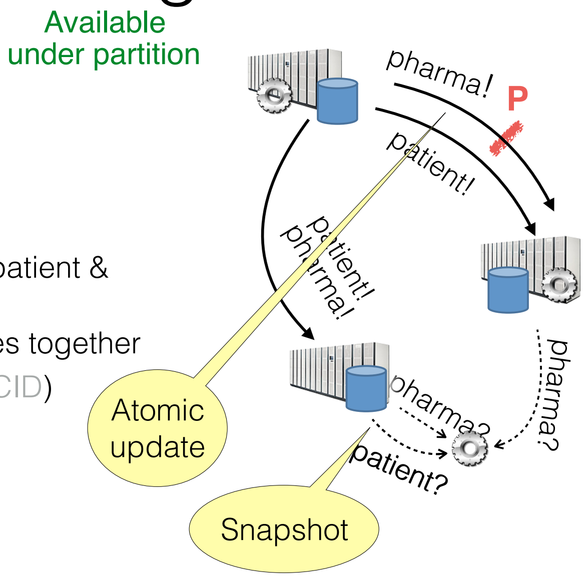
AP compatible

Equivalent-item invariant



with animations

ll-or-nothing bundle



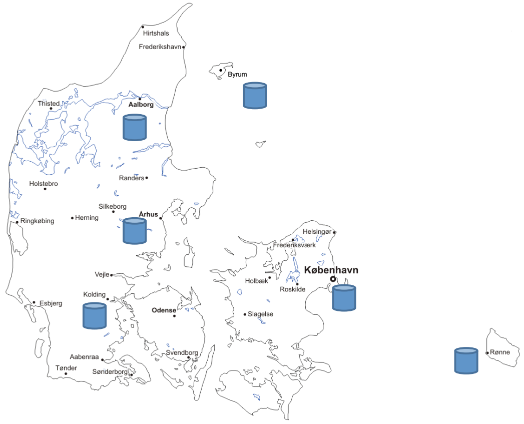
create-p updates doctor, patient & pharmacy record

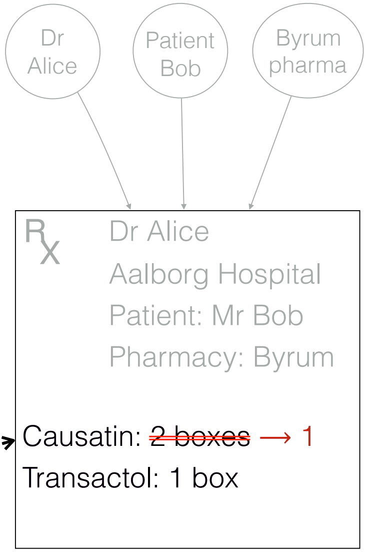
Transmit, read joint updates together

= All-or-Nothing (A of ACID)

AP-compatible

Item-value invariant





Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: ~~2 boxes~~ ⟶ 1
Transactol: 1 box

process (…)

precondition-modify

CISE result

Assume:

- Causal consistency, atomic effectors
- Initial state satisfies invariant
- *u* and *v* maintain the invariant in isolation
  (⟹ *u; v* maintains invariant)

If

- u_{eff}, v_{eff} commute
- u_{pre} stable under v_{eff}, v_{pre} stable under u_{eff}

Then *u* || *v* maintains the invariant

- Otherwise, disallow *u* || *v*

Sequential ⟶ concurrent

Safe sequential code

What invariant?

CRDTs

Synchronise only when necessary

Precondition-modify

Precondition

```
assert (remaining ≥ 0);
If (remaining ≥ requested)
    remaining −= requested;
assert (remaining ≥ 0);
```

Invariant

✔

```
assert (remaining ≥ 0);
remaining += 100;
assert (remaining ≥ 0);
```

May precondition be negated by concurrent update?

If precondition of *u* is stable under *v*, and vice-versa:
then *u* || *v* OK.

otherwise

- weaken invariant, Available under partition
- or synchronisation required.

Precondition-modify

```
assert (remaining ≥ 0);
If (remaining ≥ requested)
    remaining −= requested;
assert (remaining ≥ 0);
```

✘

```
assert (remaining ≥ 0);
If (remaining ≥ requested)
    remaining −= requested;
assert (remaining ≥ 0);
```

May precondition be negated by concurrent update?

If precondition of *u* is stable under *v*, and vice-versa:
then *u* || *v* OK. Available under partition

otherwise

- weaken invariant,
- or synchronisation required.

Sequential ⟶ concurrent

Assume

- Causal consistency, atomic effectors
- Sequentially-correct program

Transformations:

- Replace sequential data types with CRDTs
  - If not possible, synchronise access
- Verify precondition stability
- If not stable
  - either weaken invariant
  - or synchronise

Just-Right Consistency

Tailor consistency to application invariants

- (possibly unknown)

Three types of invariants:

- Ordered updates ⟹ Causal, AP
- Joint updates ⟹ Bundled, AP
- CAP-sensitive: precondition-modify
  - Mutually stable ⟹ concurrent OK. AP.
  - Otherwise, concurrency control. CP

Baseline: Correct app under strong consistency

- Identify, maintain programming patterns

AntidoteDB

CRDT data model

- Register, counter, set, map, sequence
- Extends sequential semantics

Transactional Causal Consistency Plus (TCC+)

- ≜ Relative Order + Bundles + CRDTs
- Strongest AP model

CISE: verify precondition stability

Open source, well engineered, growing community

LIGHTKONE
Lightweight computation for networks at the edge

European Commission

RainbowFS

ANR