Transaction Manifests

Marc Mosko

marc.mosko@sri.com

ICNRG, IETF 119, Brisbane

Wednesday 20, 2024

Can ICN be transactional

- Typically, ICN is considered in pub/sub or pre-pub
- Distributed transactions do exist, especially in DLTs.
- Consider a permissioned DLT with size N and K << N bookkeepers
    - In a DLT, they base their decision on the block hash history
    - In ICN, what would that be?

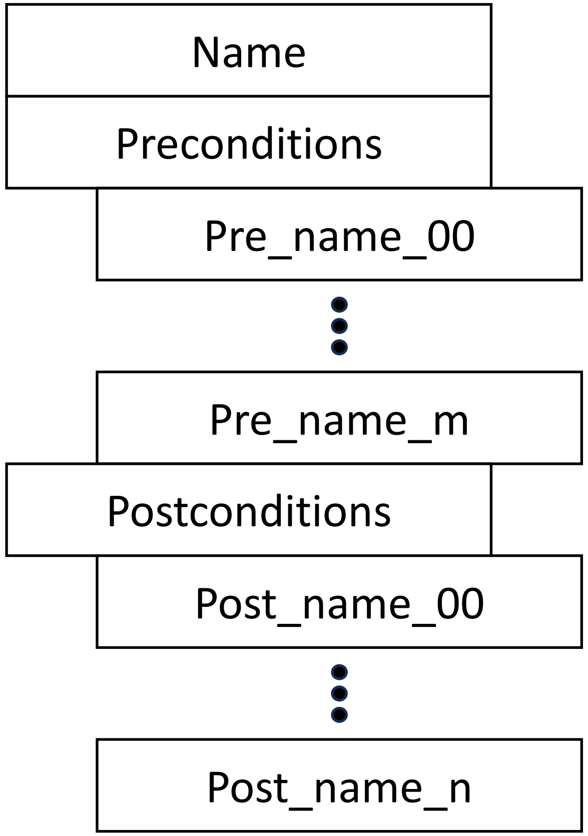
- We discuss a data object, the Transaction Manifest, as a concept.
- There needs to be a client-to-bookkeeper and bookkeeper-to-bookkeeper protocol to realize transactions.

TM vs FLIC

- FLIC describes a single object that is re-constructed by traversing the manifest in-order.
- A TM describes a set of names that must be considered together.
    - The TM names likely point to FLIC root manifests.
    - In the subsequent examples, I show TM entries pointing directly to objects.

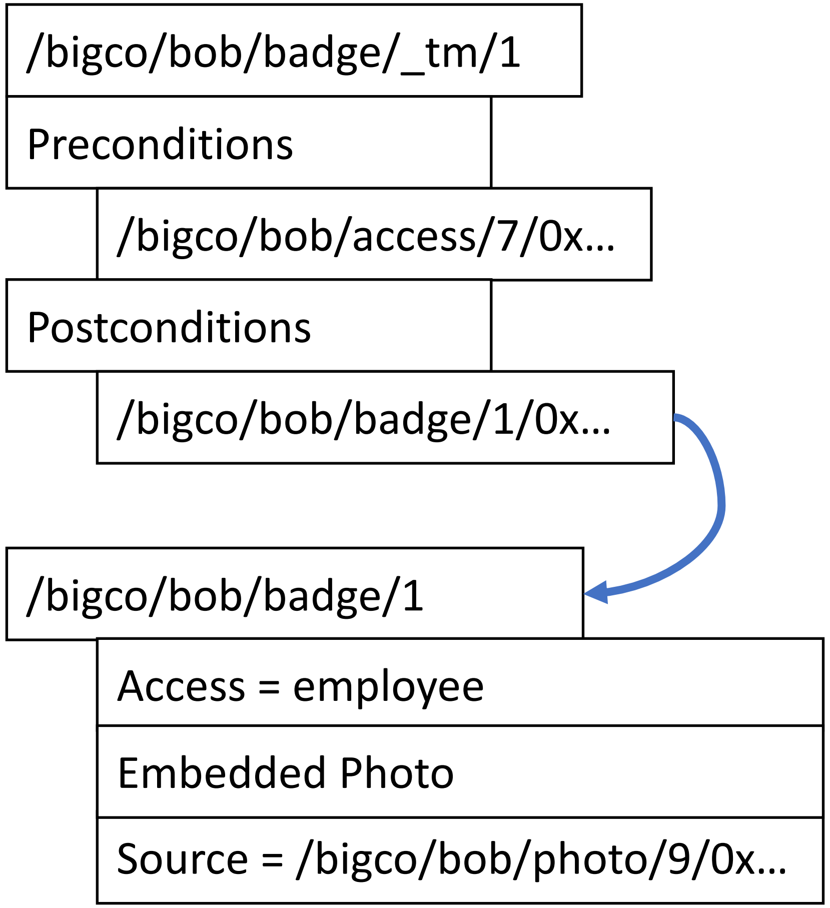
Transaction Manifest

- In a database, a transaction typically locks the input records and then writes the output records.
- A transaction manifest emulates this by specifying the input state and output state.
- An unconditional write has no preconditions.
- A transactional write with a null precondition uses a special name.



Not all input is part of a transaction

- Preconditions only name "latest version" required inputs.
- Example:
  - An employee database for producing badges needs a photo and door access level.
  - The door access level "latest version."
  - The photo may be any that matches the employee.
  - The badge TM would name the door access input object and the badge output object.
  - The badge output object would reference the photo name.



TMs do not stand on their own

- A set of bookkeepers
  - Systems like Hyperledger offer global ordering via Orderer nodes and SmartBFT (v3.0) or earlier CRT.
  - TMs are partial orders that maintain consistency without global order.
    - Partial order transactions exist in DB literature, need to review.
- Bookkeeper Job
  - Bookkeepers must ensure that a transaction has current pre-conditions, current post-conditions, and no conflicts in post-conditions.
  - TMs are a form of write-ahead log (WAL), as used in DBs like PostgreSql.
  - Nested transactions require more features (see later slide).

TMs and Repos and Caches, Oh My!

- A repository …
  - Should not respond with a post-condition unless it also has all the pre-conditions. (use a NAK?)
  - Should be able to return the TM that wrote (post-conditioned) an object.
- A cache …
  - Should respond with whatever it is asked for.
  - Applications should use non-cacheable discovery and full names from manifests.

Naming TMs

- User-specific naming
  - /bigco/alice/partsdb/tm/5
- DB (collection)-specific naming
  - /bigco/partsdb/users/alice/tm/8
- One cannot use post-condition naming unless each TM only writes one post-condition, which is likely not sufficient.

Distributed TMs

- What happens if a TM uses names that belong to different bookkeepers?
  - For example, an order database needs to reserve certain parts from the parts database.
- Hierarchical bookkeepers (nested transactions)!
  - Alice submits order to order BKs
  - Order BK provisionally approve order, submit to parts BKs.
  - Parts BKs provisionally approve part reservations.
  - Order BK commits Parts and local transactions.
  - It can get complicated with multiple child DBs and nested transactions.

Nest Steps

- Sketch out a client-to-bookkeeper protocol
- Sketch out a bookkeeper-to-bookkeeper protocol
  - Within a consensus group
  - Between consensus groups
- Analyze
- Prototype!