Principle foundations of Radix.

13th November 2020
Dan Hughes CTO

RADIX

Where the journey started ...

Understand Bitcoin and it’s novel solution to the Byzantine Generals problem using Proof of Work and the strongest chain rule.

I’m an engineer and I like to break things.

What did I find?

- Scalability issues across compute, communication and storage
- Mining centralization overtime and general inefficiency
- Long term market volatility

Leaving Blockchain behind

Focus primarily on scalability issues, with stretch goal of improving efficiency of security model

- Blocktrees & DAG (2013) - Allowed multiple compute and consensus instances. Unable to transact outside of branch.
- CAST (2015) - First sharding model. Separation of data from state. Semi-stateless validators.
- Tempo (2017) - Improved sharding model. Asynchronous voting via logical clocks & state commitments. Achieved 1M transactions per second.
- Cerberus (2020) - Further improved sharding model. Synchronous voting via multi-decree BFT. Cross-shard atomicity.

RADIX

Cerberus

A culmination of all previous iterations and research over the past number of years, Cerberus exceeds the initial criteria that was set at the start of our journey.

It delivers highly scalable and responsive permissionless networks, without sacrificing security or decentralization and provides strong guarantees on liveness and safety.

Further desirable features of the technology are composability of complex actions which can be executed and committed atomically.

Technical content on Cerberus consensus theory can be found at https://arxiv.org/abs/2008.04450 and https://radixdlt.com for other content relating to topics in this presentation.

Data Model

Client actions are represented by primitives called Atoms.

Atoms contain one or more primitives called Particles that carry state and execution instructions.

Particles have a “spin” property representing it’s lifecycle.

UP/DOWN spin are akin to constructor and destructor of that state.

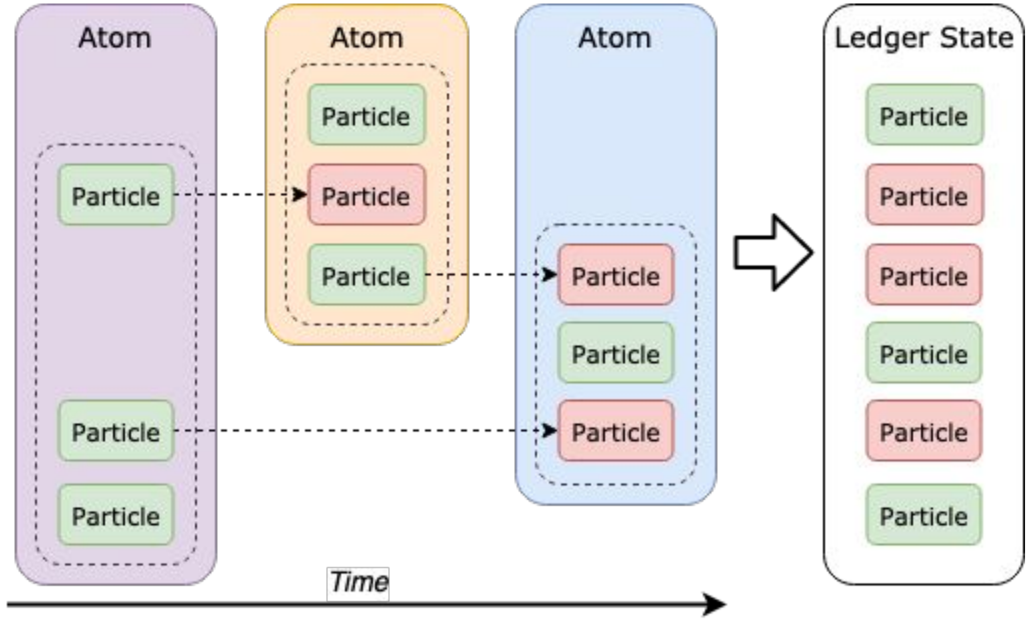
Non-existent particles implicity carry a NEUTRAL state.

The state model is heavily inspired by Bitcoin’s UTXO.

A UTXO style state model provides efficient sharding capability almost out of the box.

RADIX

Data Model


Atom
Particle
Particle
Particle
Atom
Particle
Particle
Particle
Atom
Particle
Particle
Particle
Ledger State
Particle
Particle
Particle
Particle
Particle
Particle
Time

Sharding Model

Fixed addressable shard space of 2^256 shards.

Shards contain consensus output and state information for it's particle.

Particles map to shards via their (atom||particle) hash.

Validators are assigned to validator sets who are responsible for a group of shards.

Validator sets are churned every epoch.

A 'root' shard stores all registered validator information.

Particle and state information lookups are efficient. The validator sets of the current epoch act as a map into shard space.

RADIX

Consensus Model

Two consensus domains, state and ledger.

State consensus is performed by **a** validator set upon a particular state.

Ledger consensus is performed by **all** validator sets for a particular atom.

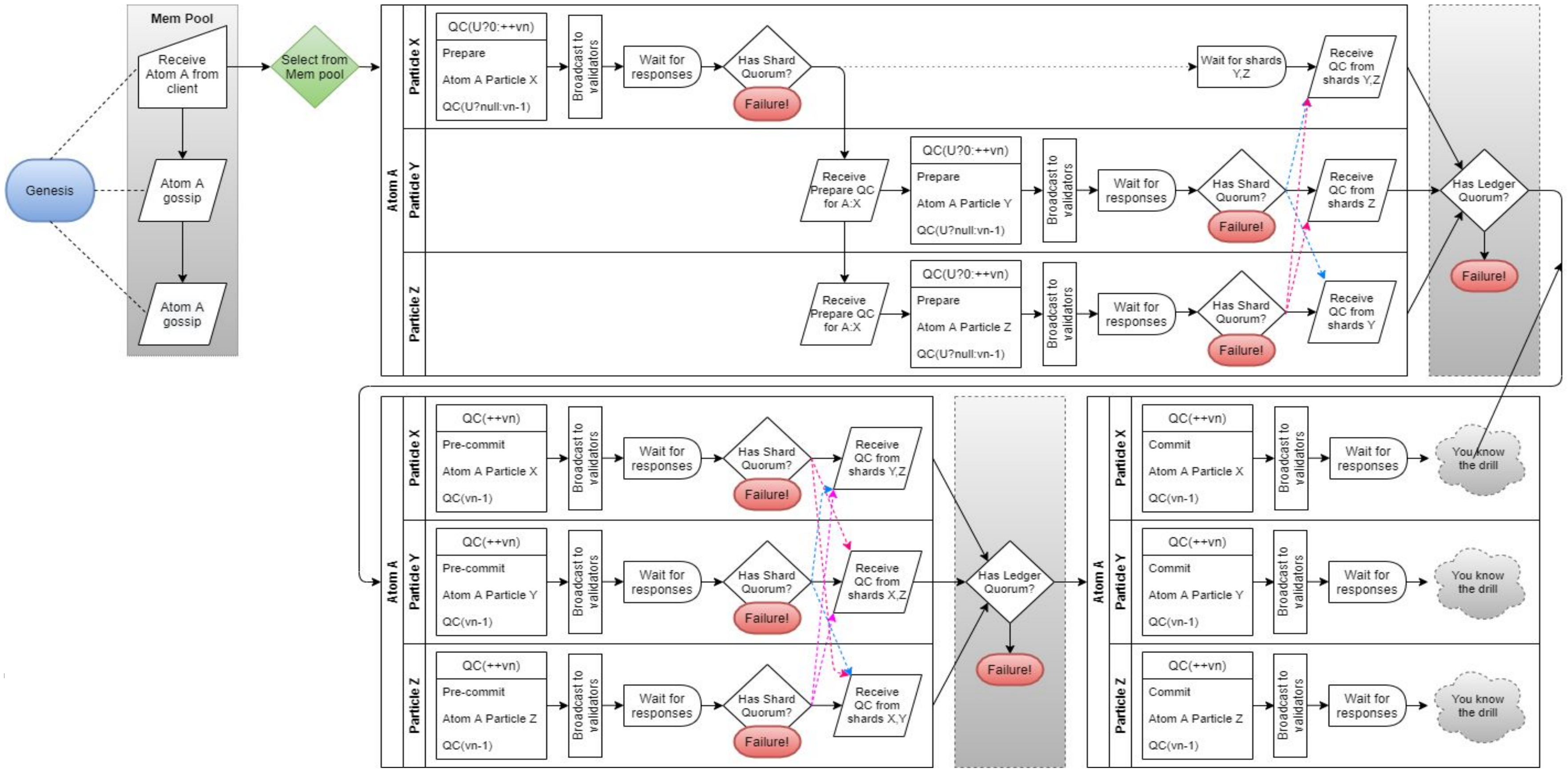
State domain consensus is largely agnostic but must be able to produce a quorum certificate.

We are using 3-phase Hotstuff as our state consensus.

All validator sets must agree with ***2f+1*** validators in all phases to commit.

Any validator set can disagree with ***2f+1*** validators in any phase to abort.

Consensus Model - Example protocol



Sybil Model

Proof of Stake based Sybil prevention.

A validator vote is weighted by the amount of stake controlled by it.

f represents quantity of faulty stake.

Acquisition of more stake requires buying from open market.

Large market buys will trend price upwards, making subsequent purchases more expensive.

Generation of validator sets is seeded by stake activity in the previous epoch.

Validator sets are “load balanced” to prevent single / colluding validators from controlling the set.

RADIX

De-Fi, UX and Atomicity

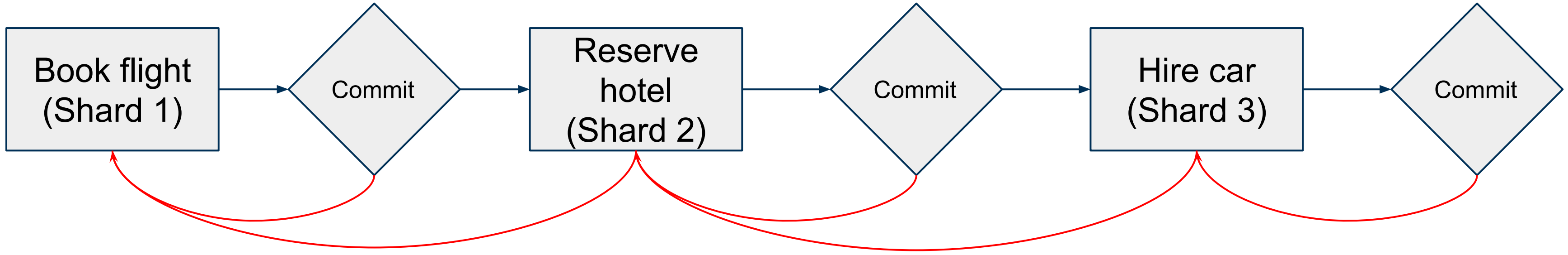
Blockchain and distributed ledger applications are becoming much more complicated, De-Fi especially.

There is a need to “touch” many points of information when executing an action, which in turn may also alter the state of those dependencies.

Atomic cross-shard commit is not just a nice feature, it is a **must** have feature.

- Better user experience
- Better developer experience
- DAPP code is simpler = more secure
- More efficient

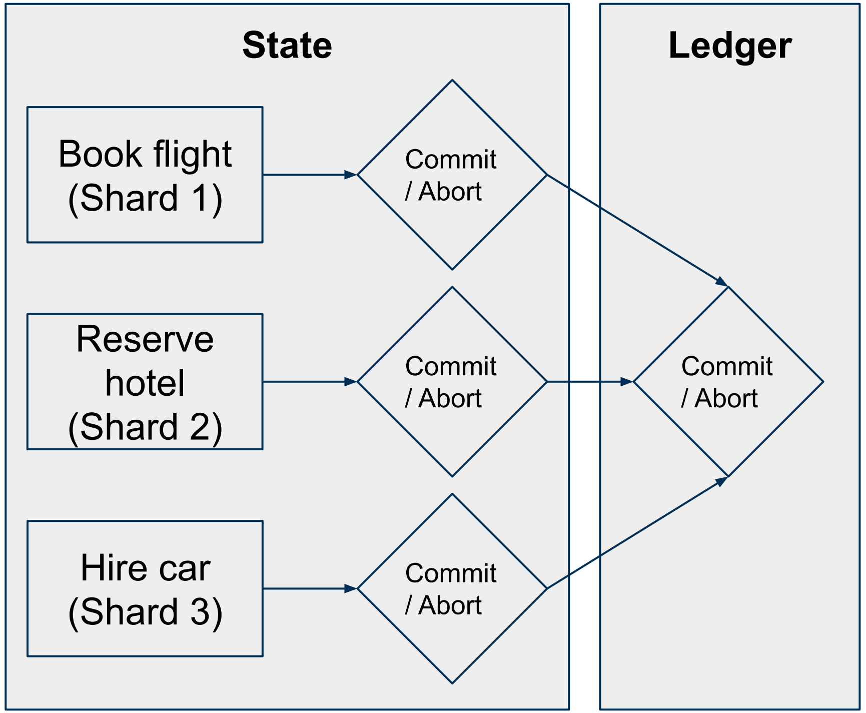
Non-atomic holiday



- Actions must be completed sequentially.
- Failure of an action requires the undoing of the previous
  - The undo commit may also fail
  - Some use cases may require a manual process
- Which action do I perform first?

RADIX

Atomic holiday



Actions undergo state consensus in their respective shards.

The output of the state consensus determine whether the set of actions is “committed” to the ledger.

- Actions are executed in parallel.
- Failure of any action aborts the commit.
  - There is nothing to undo.

RADIX

Thank you for listening.

Questions?

RADIX