Hyperledger Fabric

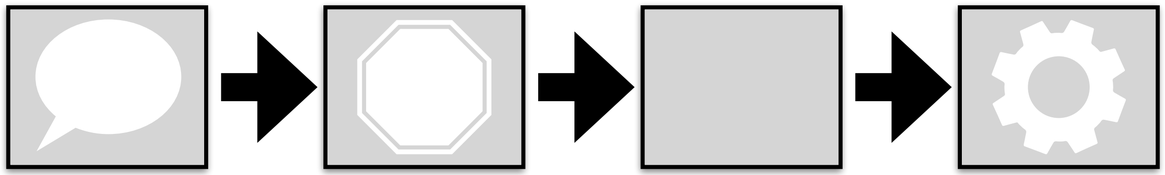
Execute-Order pipeline

Leander Jehl

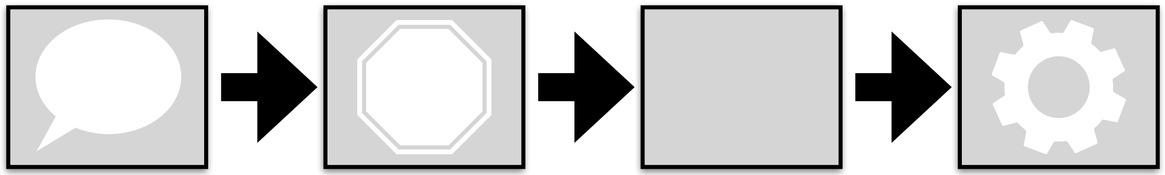
Hyperledger fabric

- a toolbox to run your own, permissioned blockchain

Transaction processing pipeline

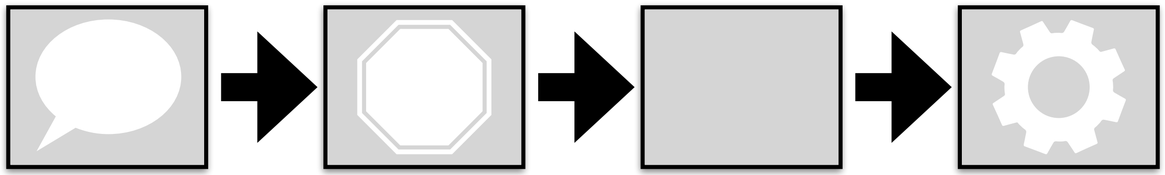


Transaction processing pipeline



- broadcast: send out transaction requires network resources
- validation: requires state
- ordering: requires coordination
- execution: requires state, must be deterministic.

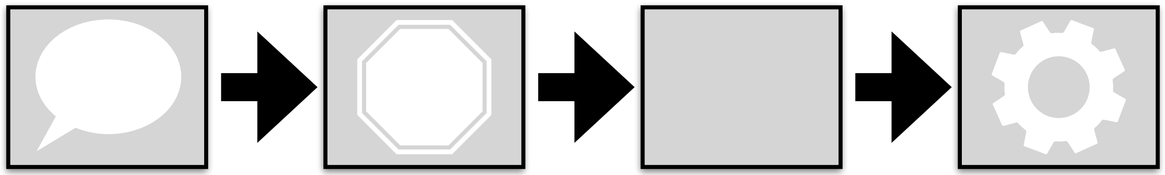
Transaction processing pipeline



- broadcast: send out transaction requires network resources
- validation: access rights
- ordering: requires coordination
- execution: requires state, must be deterministic.

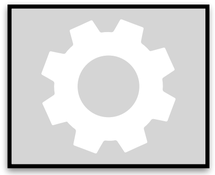
What is the bottleneck?

Transaction processing pipeline



- For complex workloads, and small scale BFT systems, execution is the bottleneck.
  - Single threaded execution to be deterministic
  - Can be complex workloads
  - Execution has privacy concerns (need access to data)

Transaction execution

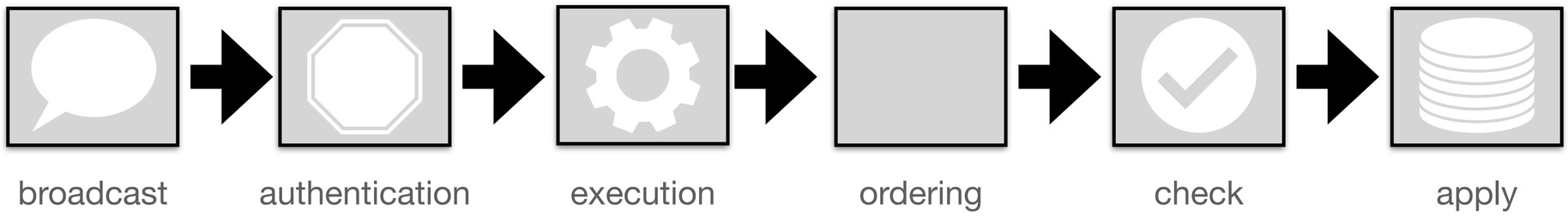


execution

Two approaches exist for crash fault tolerant systems:

- **Deterministic processing**: Each replica can process transaction and arrive at the same result.
- **Applying state change**: One replica executes transaction. Records state change Δ. All replicas apply Δ.

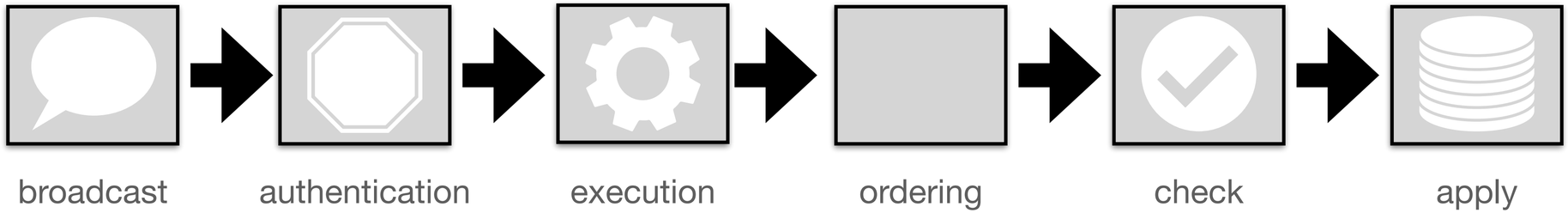
Transaction processing in Hyperledger fabric



- Execution happens before ordering.
- Execution policies, i.e. require *n* nodes to get the same result.
- Changes are submitted to ordering with signature from *n* nodes.
- During check, possibly inconsistent transactions are removed (aborted).

Transaction processing in Hyperledger fabric

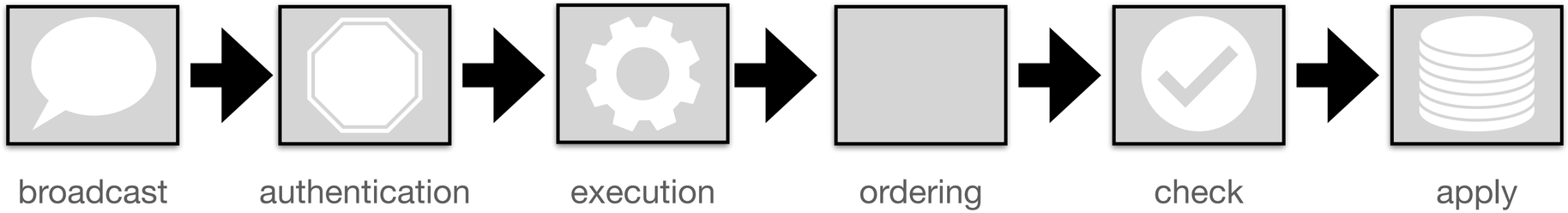
State



- State is organized as (key, value) pairs.
- Execution result records, new values for certain keys and which keys have been read.
- Based on read and write keys, *check* can remove inconsistent transactions

Transaction processing in Hyperledger fabric

State



- State is organized as (key, value) pairs.
- Execution result records, new values for certain keys and which keys have been read.
- Based on read and write keys, *check* can remove inconsistent transactions

For all stages but execution, values can be encrypted.

Use cases and problems

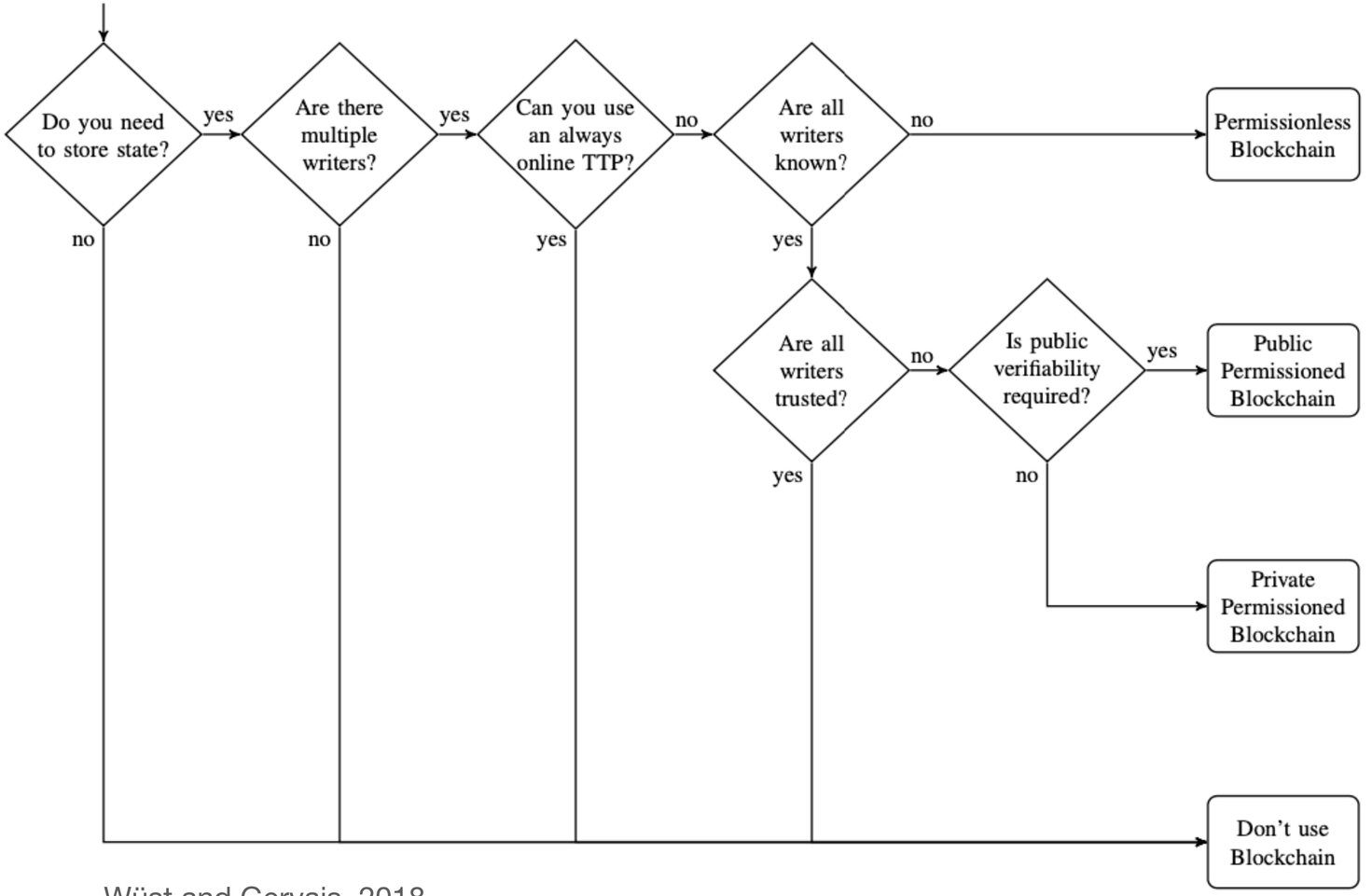
Use cases

Blockchain

- Financial
- Accountability
- Digital assets

Use cases

Do you need a blockchain?

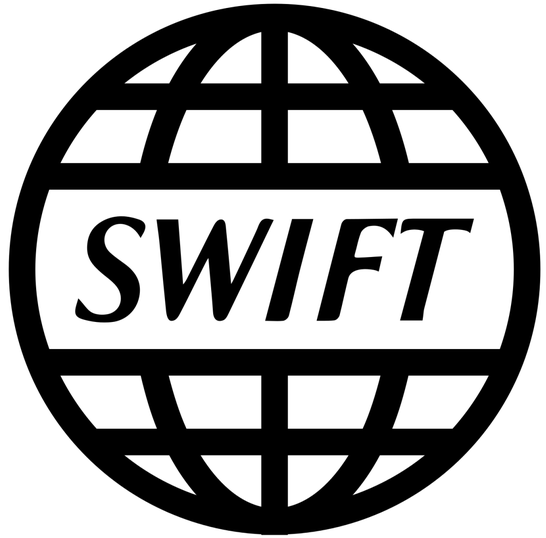


Wüst and Gervais, 2018

Use cases

Blockchain

- Financial
  - International bank payments
  - Banking the unbanked
  - *Central bank digital currency?*



Use cases

Blockchain

- Digital assets
  - Buy and sell digital goods
  - What goods?

Use cases

Blockchain

- Accountability
  - Blockchain datastructure is used as log.
  - E.g. record data access
  - E.g. record decisions (autonomous vehicles)

Use cases

Blockchain

- Supply chain
  - Idea: Record each step in manufactoring on the chain.
    - Can trace faulty components
    - Can prevent fraud
    - Verify manufacturing conditions
  - Problem: Is my wine the wine certified on blockchain?

Use cases

Blockchain

- Proof of intellectual property
- E-Voting
- IoT