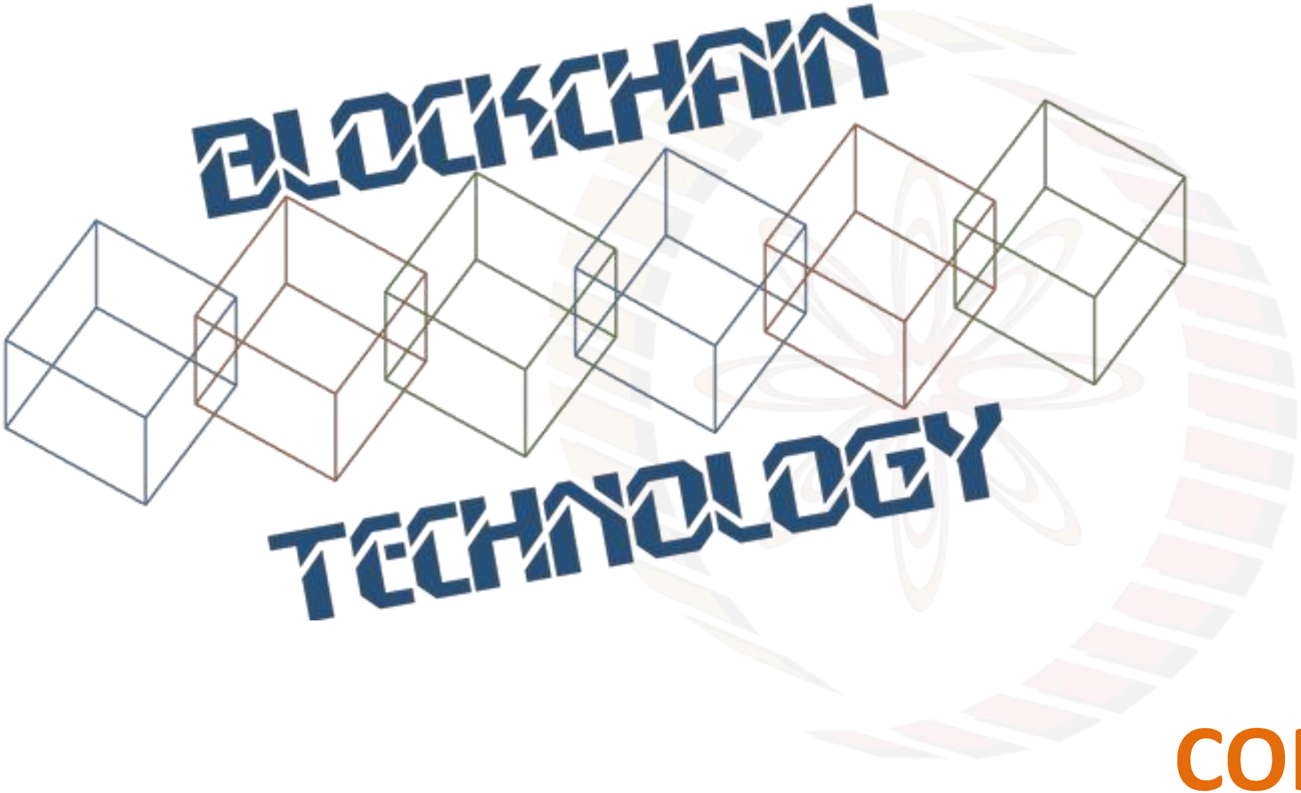
BLOCKCHAINS

ARCHITECTURE, DESIGN AND USE CASES

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

PRAVEEN JAYACHANDRAN
IBM RESEARCH,
INDIA

*Image courtesy: http://beetfusion.com/

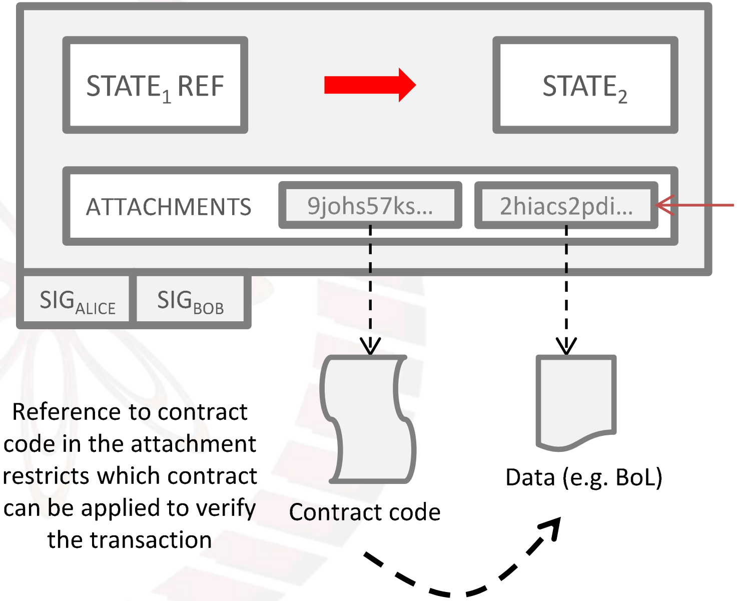


CORDA – PART 2

Transaction Attachments

- Attachments are arbitrary data in the form of ZIP/JAR files identified by a hash.
- Attachments are intended for data on the ledger that multiple peers may reuse over time.
- A transaction can reference on or more attachments (files not included in the transaction itself).
- Transactions may contain references to contract code or data files such as passport, BoL, etc.

Attachments (documents) are first-class citizens in Corda's design. Not hard to do with Fabric, but Corda makes the development experience simpler by providing APIs to easily manage attachments.



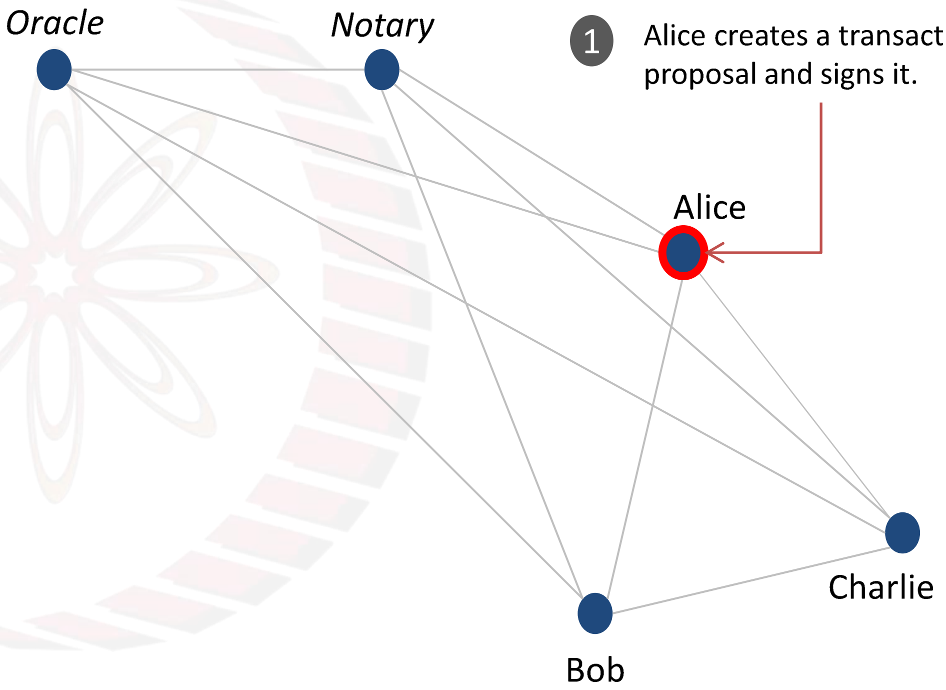
Attachments are added by tx creators and referenced by their hash

Reference to contract code in the attachment restricts which contract can be applied to verify the transaction

Data files can be accessed by the contract code. Code can check for authenticity of data by looking at signatures

Transaction Flow

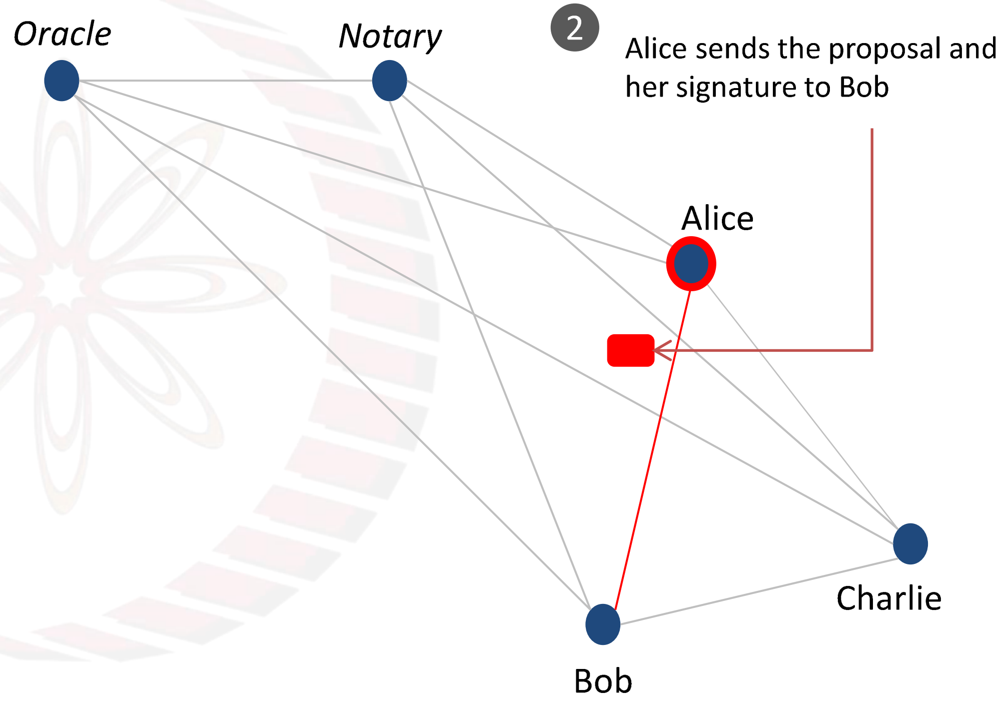
- In Corda, peers communicate point-to-point.
- Transactions specify a list of message recipients that are required sign and verify the transaction.
- The **Flow Framework** is designed to orchestrate this multistep coordination and verification of transaction messages.
- Thus to commit a transaction, there's a back-and-fourth communication flow between peers.



1 Alice creates a transaction proposal and signs it.

Transaction Flow

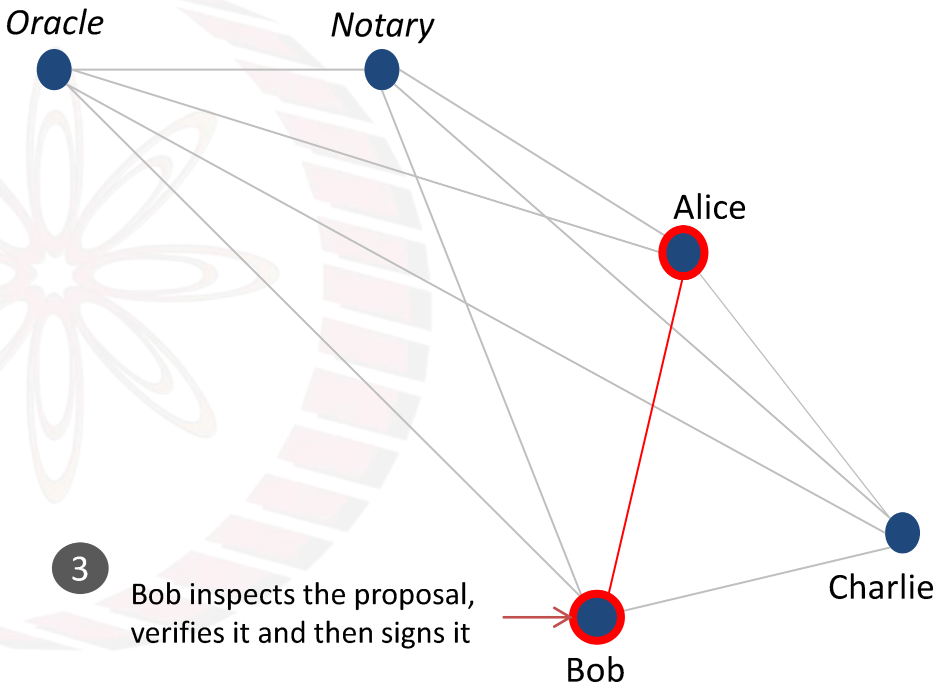
- In Corda, peers communicate point-to-point.
- Transactions specify a list of message recipients that are required sign and verify the transaction.
- The **Flow Framework** is designed to orchestrate this multistep coordination and verification of transaction messages.
- Thus to commit a transaction, there’s a back-and-fourth communication flow between peers.



Alice sends the proposal and her signature to Bob

Transaction Flow

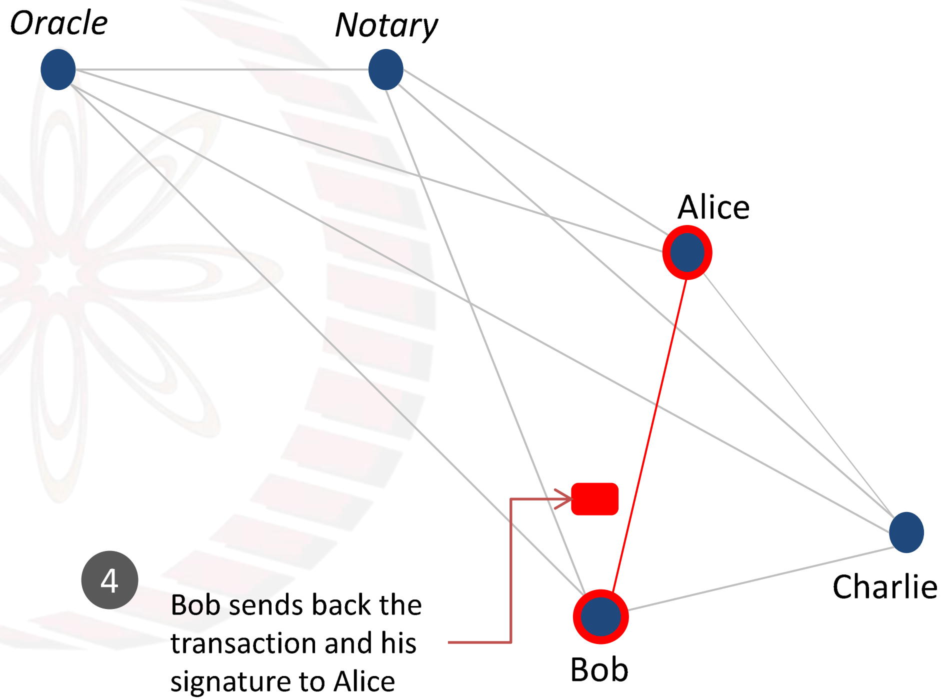
- In Corda, peers communicate point-to-point.
- Transactions specify a list of message recipients that are required sign and verify the transaction.
- The **Flow Framework** is designed to orchestrate this multistep coordination and verification of transaction messages.
- Thus to commit a transaction, there's a back-and-fourth communication flow between peers.



Bob inspects the proposal, verifies it and then signs it

Transaction Flow

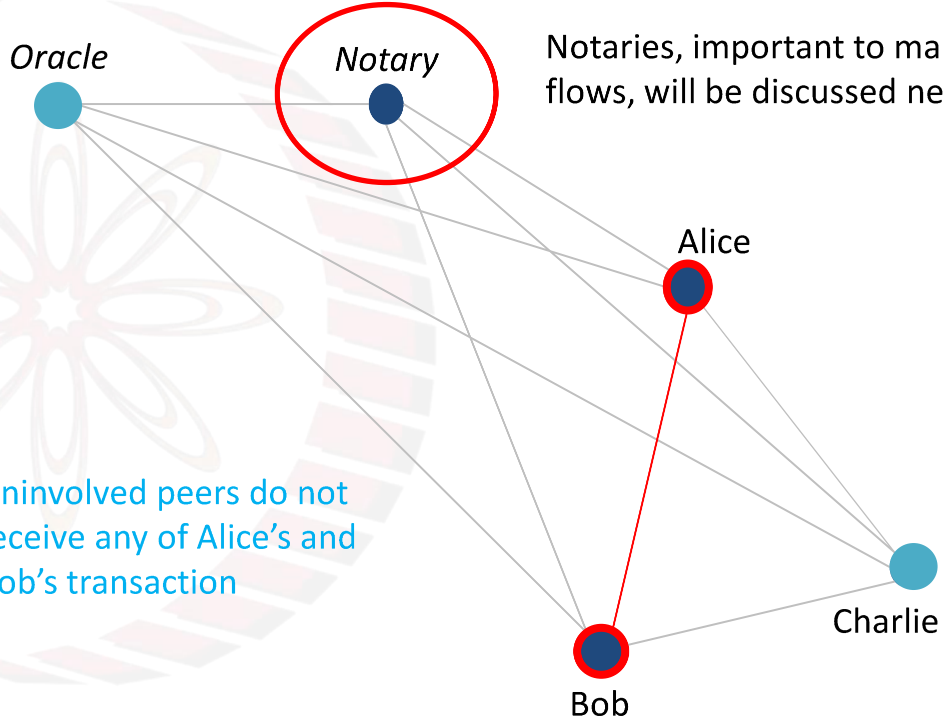
- In Corda, peers communicate point-to-point.
- Transactions specify a list of message recipients that are required sign and verify the transaction.
- The **Flow Framework** is designed to orchestrate this multistep coordination and verification of transaction messages.
- Thus to commit a transaction, there's a back-and-fourth communication flow between peers.



Bob sends back the transaction and his signature to Alice

Transaction Flow

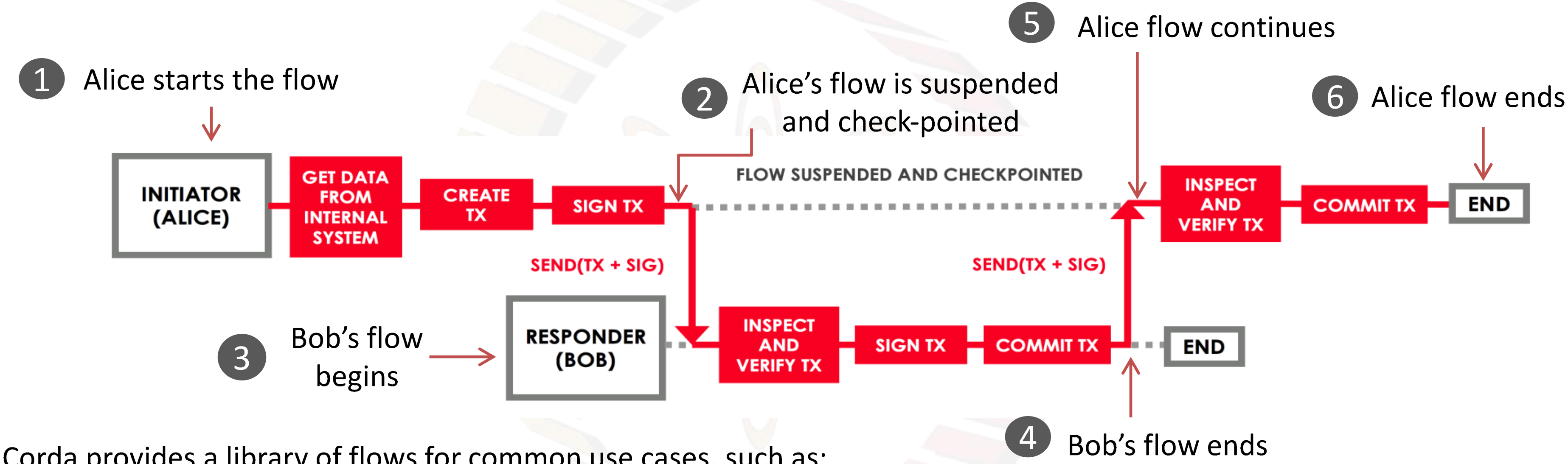
- In Corda, peers communicate point-to-point.
- Transactions specify a list of message recipients that are required sign and verify the transaction.
- The **Flow Framework** is designed to orchestrate this multistep coordination and verification of transaction messages.
- Thus to commit a transaction, there's a back-and-fourth communication flow between peers.



Notaries, important to many flows, will be discussed next.

Uninvolved peers do not receive any of Alice's and Bob's transaction

Flow Example



Corda provides a library of flows for common use cases, such as:

- Atomic asset swaps
- Broadcasting data to multiple peers
- Sending cash
- Agreeing multi-lateral deals

Consensus

Peers reach consensus in two ways

Validity Consensus

- Check that the transaction (and all dependencies) is signed by all required peers
- Check that the transaction satisfies the constraints define by the contracts

Uniqueness Consensus

- Check that the transaction output is unique in the network, i.e. there's no double-spend attempt

Problem: Since there's no global broadcast and transactions are P2P, what happens if an asset issued by Party A to Party B is transferred to Party C a month later? Party C can't see the provenance of the asset.

Validating Transaction Histories

Shared state between Charlie (Central Bank) and Dan

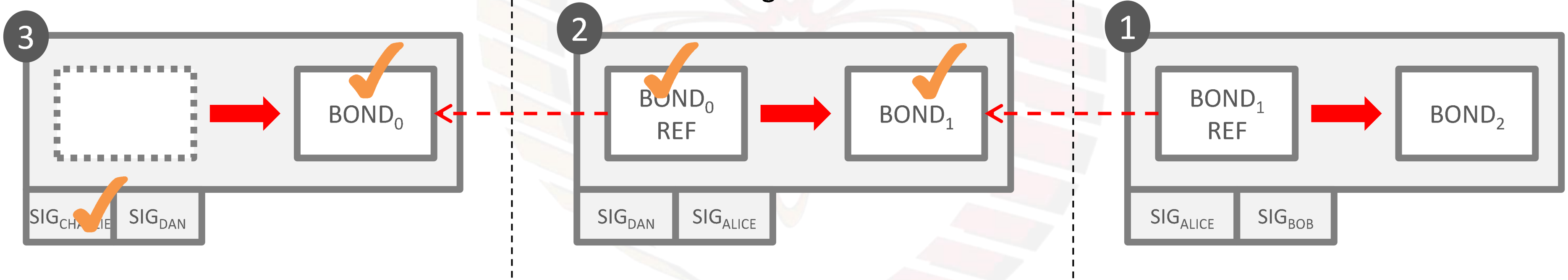
Charlie issues an asset to Dan

Shared state between Dan and Alice

Dan transfers asset to Alice in exchange for cash

Shared state between Alice and Bob

Alice transfers asset to Bob in exchange for cash



When a peer is presented with a transaction containing input state references, the prior transaction which created the reference output states needs to be verified – this recursive process is called *walking the chain*

Bob is uncertain on the origin of the asset signed by Alice

Validity Consensus

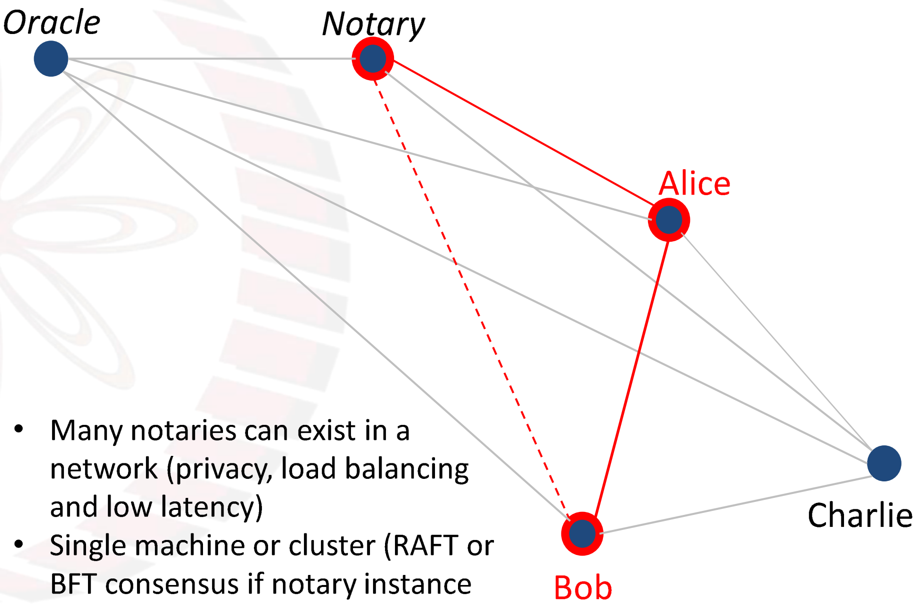
- There’s no global broadcast (unlike Fabric or Ethereum).
- Peers need to recursively retrieve the entire dependency graph of historic transactions from other peers to fully validate transactions.
  - Handled automatically by the Flow Framework
  - Neighboring peer usually has all history
  - Only required in cases where transactions are not confined to a set of participants (e.g. generic assets such as Cash that are transferable, where as a fixed trilateral agreement is OK).
- Can lead to long sequence of transactions being moved from peer to peer.
- Privacy concern as data can leak from issuing party to parties further down the chain that need provenance.

Solutions

- Problem only with transferable assets
- Using Merkle tree to hide transaction histories
- Signing key randomization to anonymize peers
- State re-issuance
- Intel SGX to verify transactions on the remote peer without transferring history

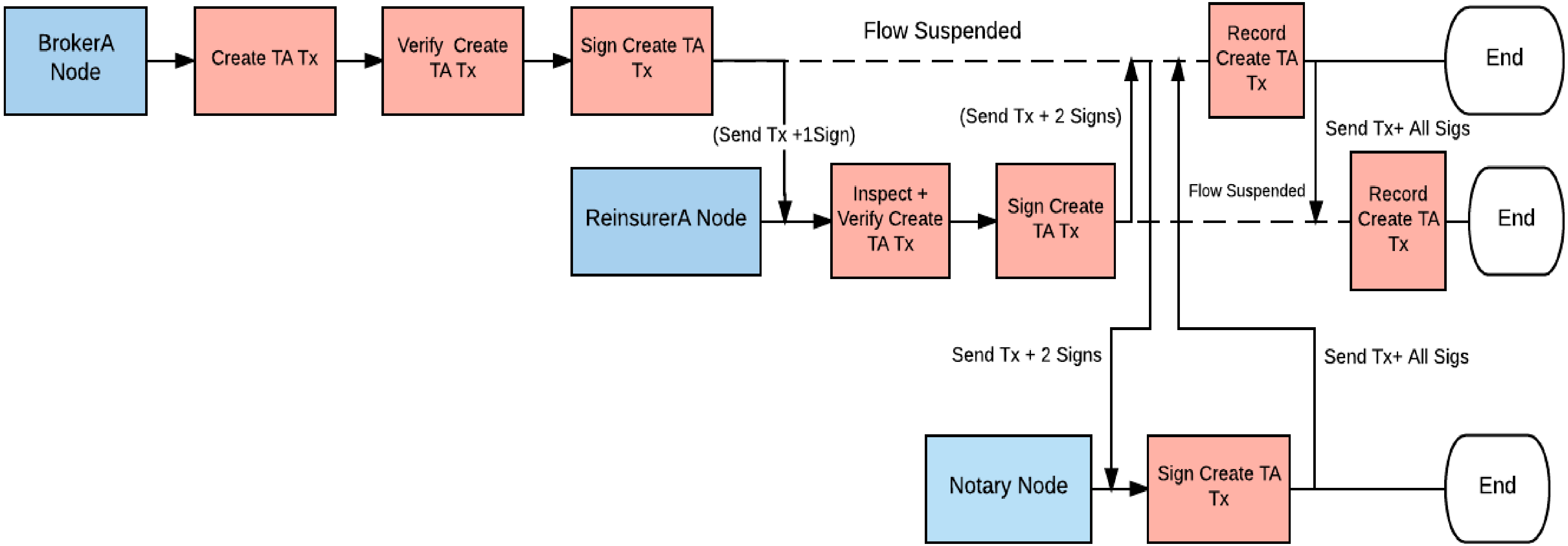
Notaries – Uniqueness Consensus

- Transaction finality is reached when the specified notary service signs the tx
  - A notary service is designated at the time a state is issued and remains fixed for the life-cycle of the state.
- In simple terms, the notary service maintains a map of transaction IDs used as input.
- If the notary service has not seen the input reference before, the input reference is added to the map, transaction is signed and is considered final.
  - This ensures the output is a unique successor of the input, and thus a double-spend did not occur.
- A special type of notary, called the Verifying Notary, will require the transaction and the entire dependency chain of histories for greater assurance.



- Many notaries can exist in a network (privacy, load balancing and low latency)
- Single machine or cluster (RAFT or BFT consensus if notary instance is run by multiple parties)

Example Flow Involving Notary

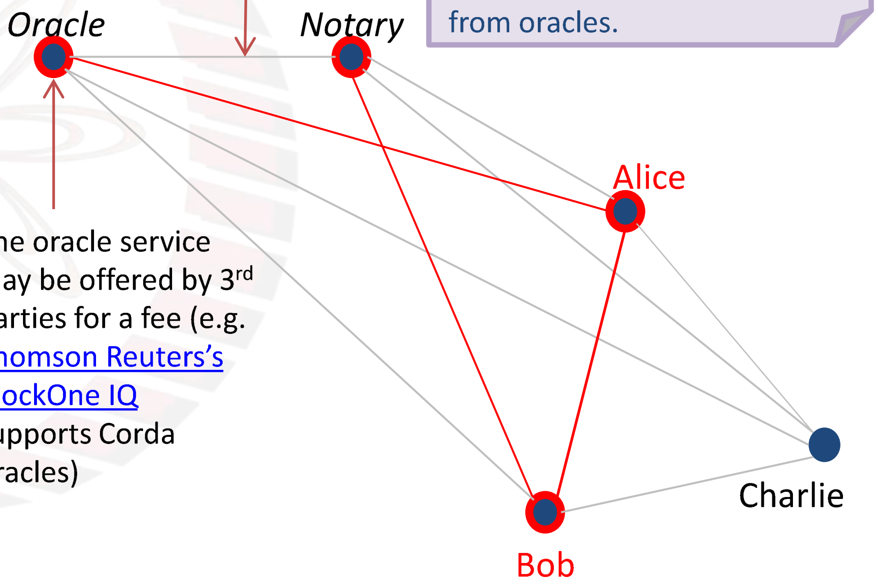


Notaries and Transaction Time-Windows

- If a transaction includes a time-window, it can only be committed during that window.
  - The notary is the timestamping authority, and refuses to sign transactions outside of that window.
  - Time-windows can have a start and end time, or be open at either end.
- The purpose of a time windows is to communicate a transaction's position in a time-line to contract code for the enforcement of contractual logic.
  - E.g. redemption of bond or late payment of debt.
- Time windows may be used for other purposes, such as regulatory reporting.

Oracles

- Oracles allow the validity of transactions, as determined by a contract, to be conditioned based on an off-ledger fact (e.g. price).
  - An oracle provides a digitally signed data structure that asserts an off-ledger fact.
- The oracle is a source which is accepted by multiple parties as **authoritative**, **binding** and **definitive** for a range of facts (or calculations).
  - The oracle can source data externally
  - Alternatively, the oracle can calculate results based on on-ledger states or attachments
- Oracles are available during the proposal and verification of a transaction.
- Oracles are implemented in a transaction using either Commands or Attachments.
  - The oracle becomes a required signer of the transaction.

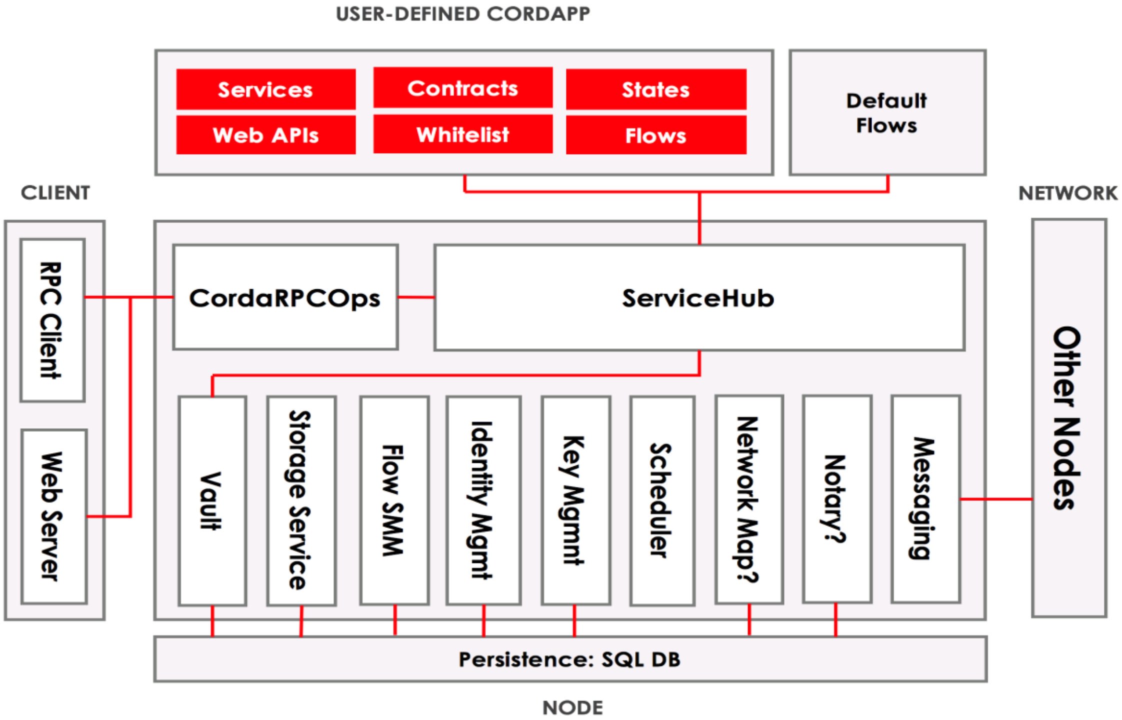


Flows are used to communicate with the oracle

The oracle service may be offered by 3rd parties for a fee (e.g. Thomson Reuters's BlockOne IQ supports Corda oracles)

Oracles are first-class citizens in Corda. Developer support makes it easy to write code that source off-ledger facts from oracles.

Corda Node Architecture



Fun Reading

- Short video on Corda Oracles (6 mins): https://vimeo.com/214157956 thank you!