The Hyperledger Suite of Frameworks and Tools

Overview

A blockchain is an immutable, verifiable and distributed ledger of transactions

Hyperledger is an umbrella of open-source blockchain projects

Hyperledger Frameworks are used to build blockchains for various types of applications

Hyperledger Tools simplify interactions with Hyperledger Frameworks

Quick Overview of Blockchain

Blockchain

An encrypted database of agreements

Blockchain

An encrypted database of agreements

Once a deal has been recorded neither party can go back and rewrite terms

Blockchain

An encrypted database of agreements

Recorded in the form of verified transactions

Blockchain

An encrypted database of agreements

Multiple transactions are stored in the form of blocks

Blockchain

An encrypted database of agreements

Serves as a bookkeeping platform or a ledger

Blockchain

An encrypted database of agreements

Incorruptible, enforces transparency and bypasses censorship

Blockchain

An encrypted database of agreements

Open for the world to view or can be restricted to specific users

Blockchain

An encrypted database of agreements

What Agreements?

Financial transactions

Real-estate sales

Supply chain management

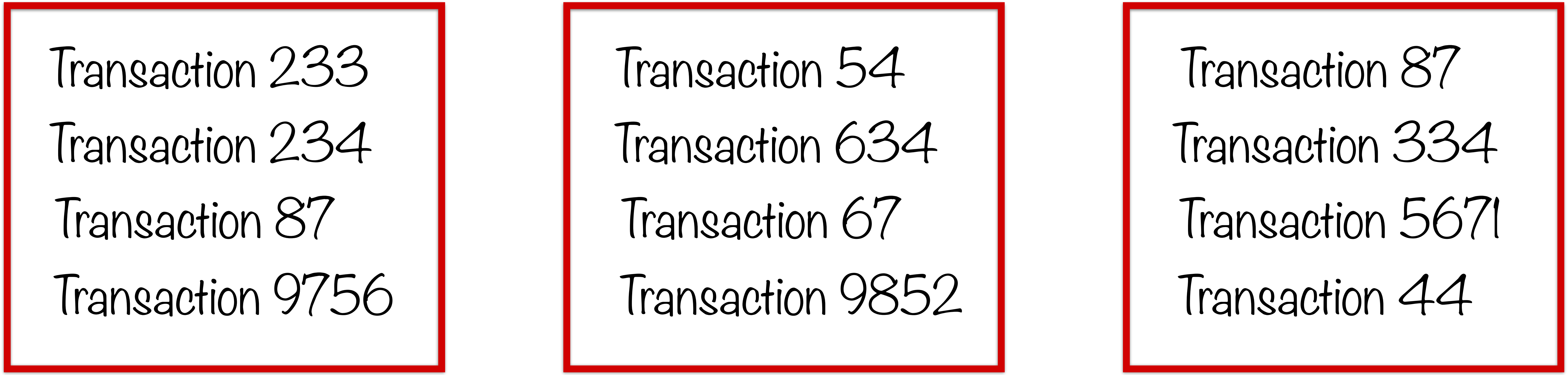
Voter records

Any contractual agreement

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

Block

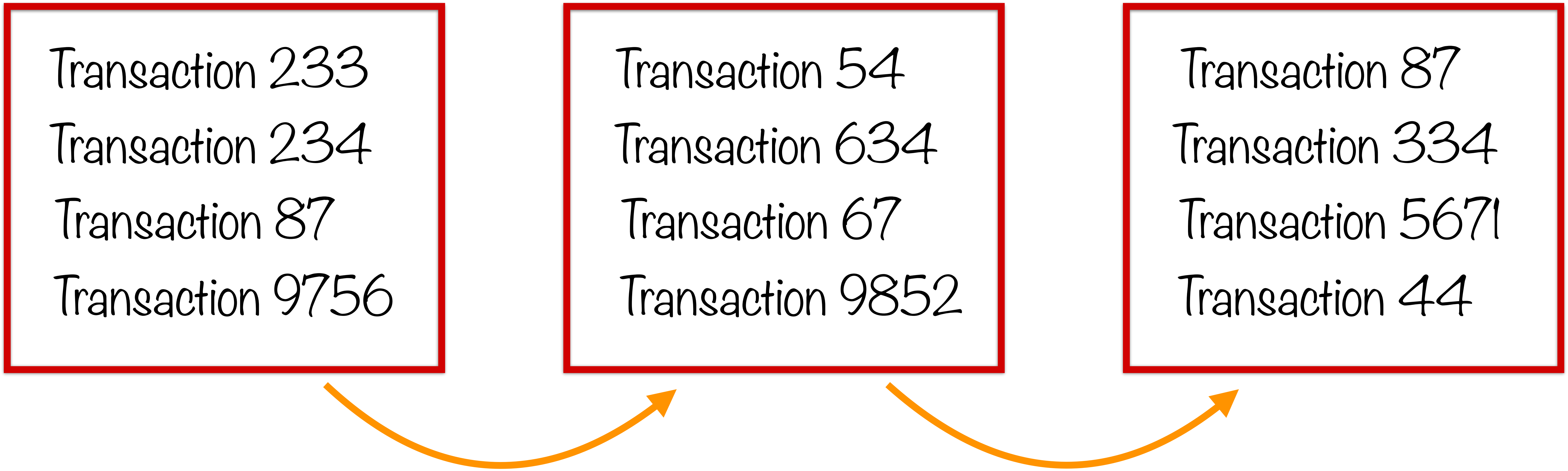


Transaction 233
Transaction 234
Transaction 87
Transaction 9756

Transaction 54
Transaction 634
Transaction 67
Transaction 9852

Transaction 87
Transaction 334
Transaction 5671
Transaction 44

Blocks Linked Together Cryptographically



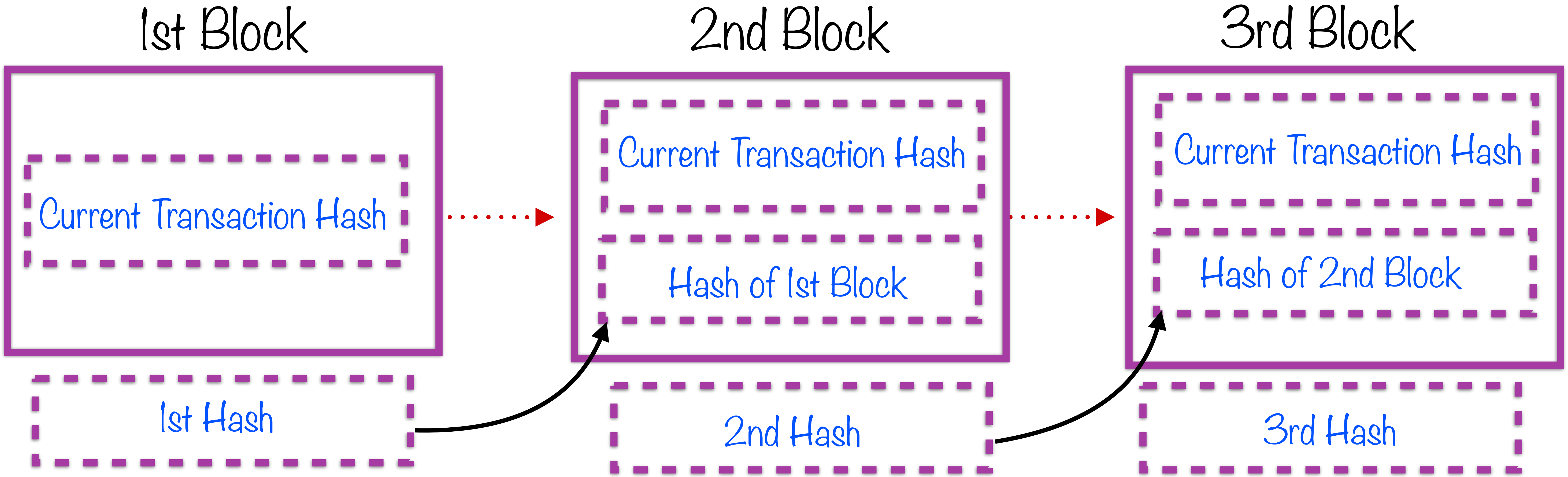
Transaction 233
Transaction 234
Transaction 87
Transaction 9756

Transaction 54
Transaction 634
Transaction 67
Transaction 9852

Transaction 87
Transaction 334
Transaction 5671
Transaction 44

Every block is linked to the next block containing the next set of transactions

Chain of Blocks



1st Block

Current Transaction Hash

1st Hash

2nd Block

Current Transaction Hash

Hash of 1st Block

2nd Hash

3rd Block

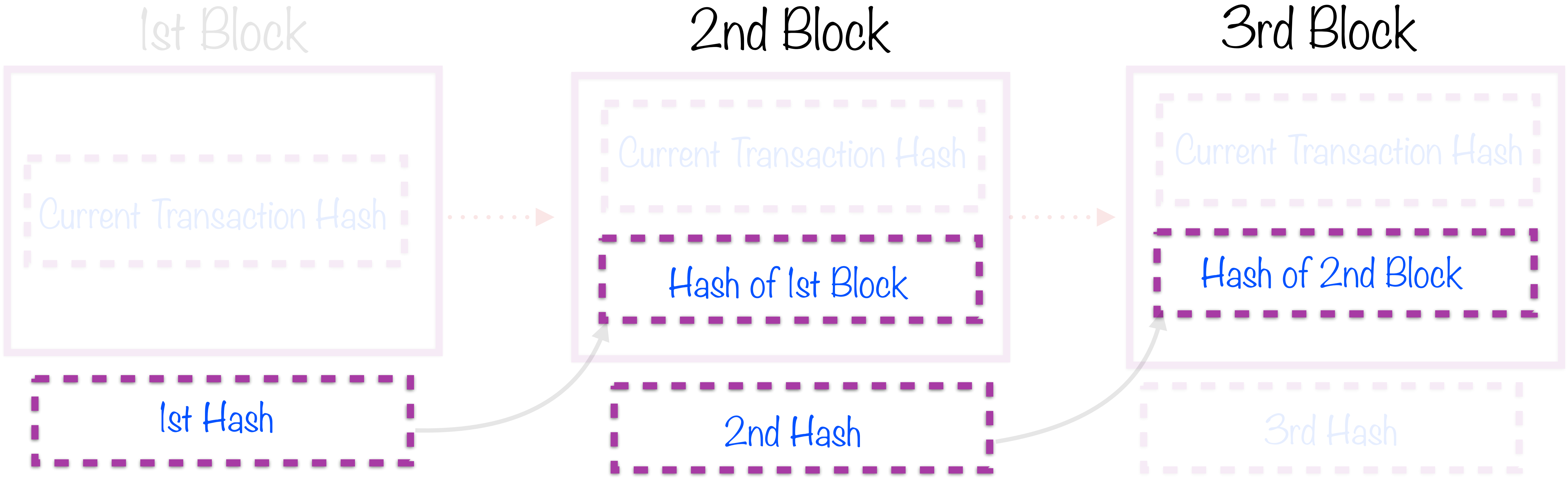
Current Transaction Hash

Hash of 2nd Block

3rd Hash

Each block contains hash of the preceding one in chain

Chain of Blocks



1st Block

Current Transaction Hash

1st Hash

2nd Block

Current Transaction Hash

Hash of 1st Block

2nd Hash

3rd Block

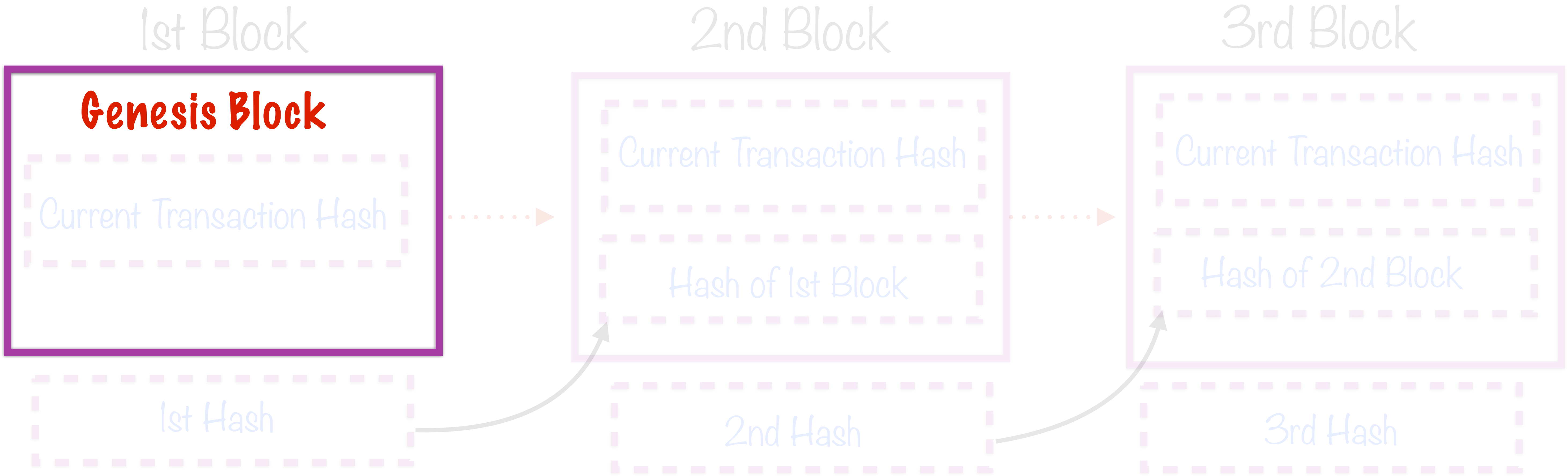
Current Transaction Hash

Hash of 2nd Block

3rd Hash

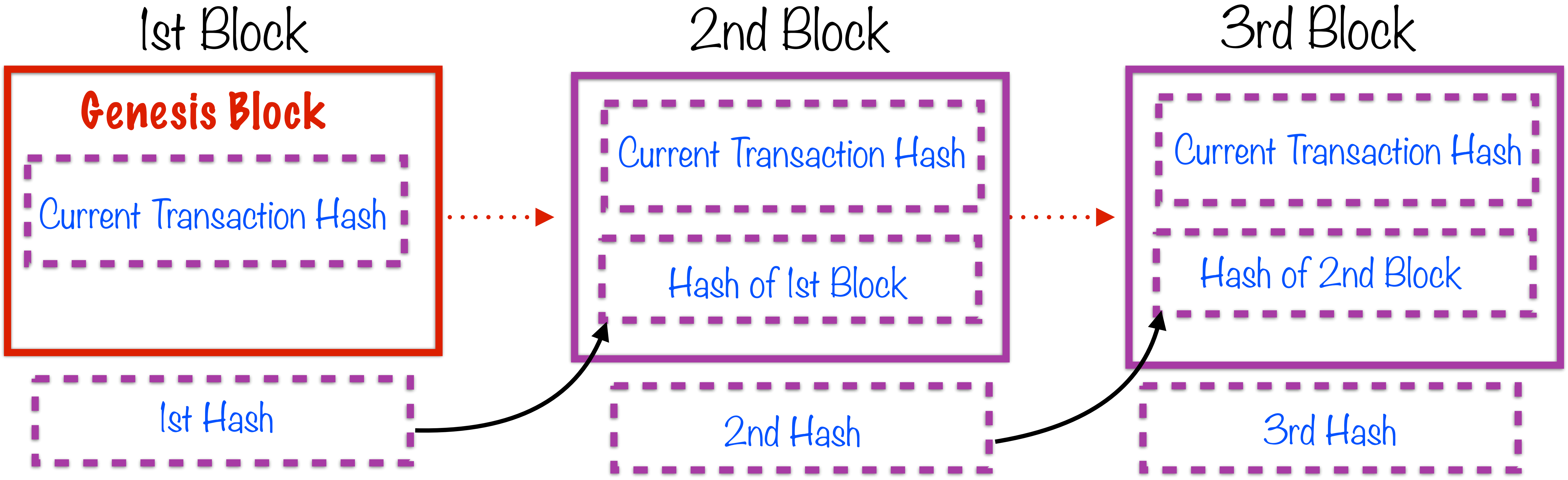
Each block contains hash of the preceding one in chain

Chain of Blocks



This is not true for the first block - called the Genesis Block

Chain of Blocks



Genesis block does not contain previous hash value

Cryptographic Link

Transaction data in that block can not be altered after-the-fact

- except by altering all subsequent blocks
- which is complicated and needs consensus

Ledger is distributed (not centralized)

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

The list is not stored in its entirety on any one node of that peer-to-peer network

Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

“Open, distributed ledger”

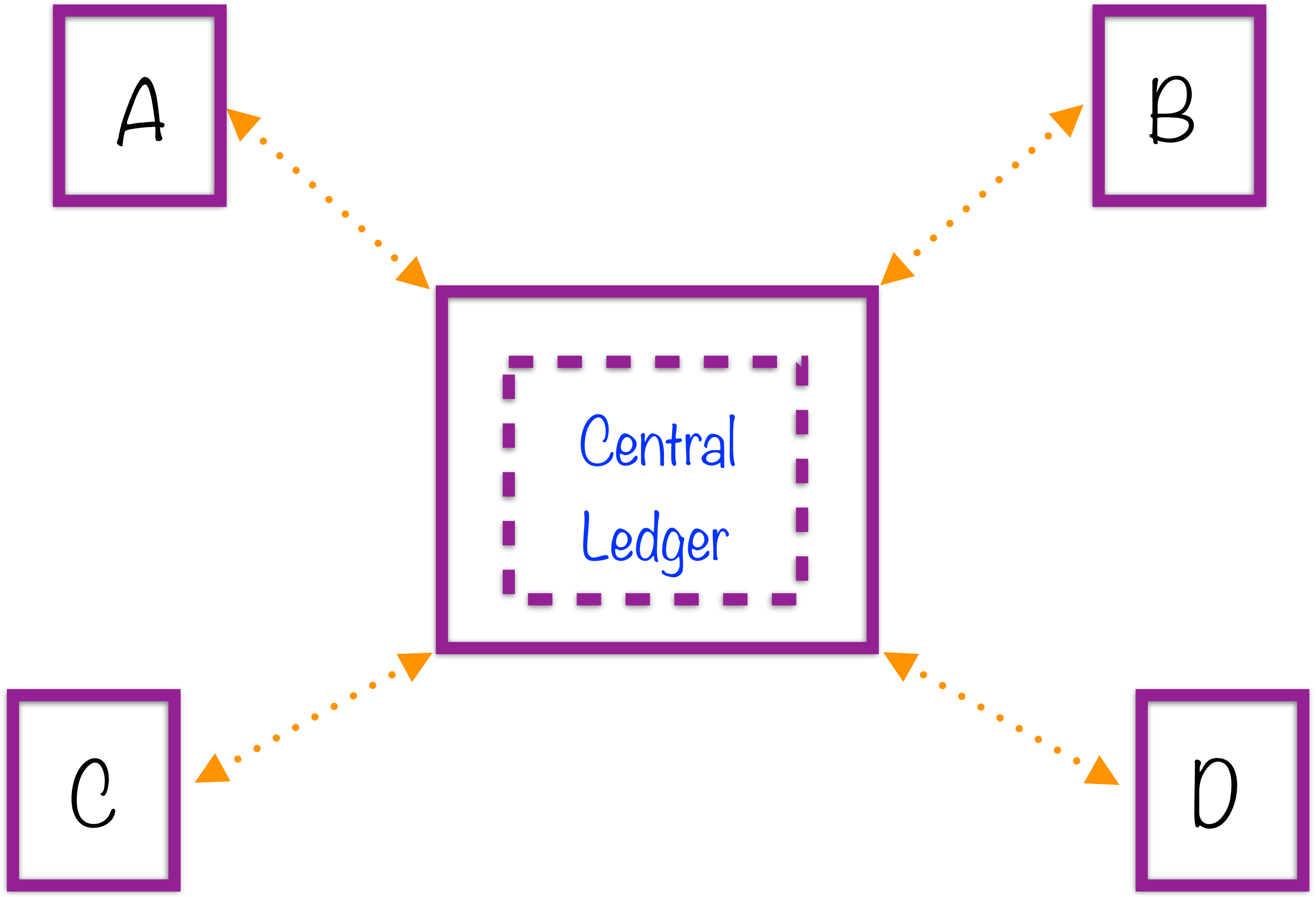
Blockchain

A blockchain is a growing list of records (called blocks) which are linked cryptographically

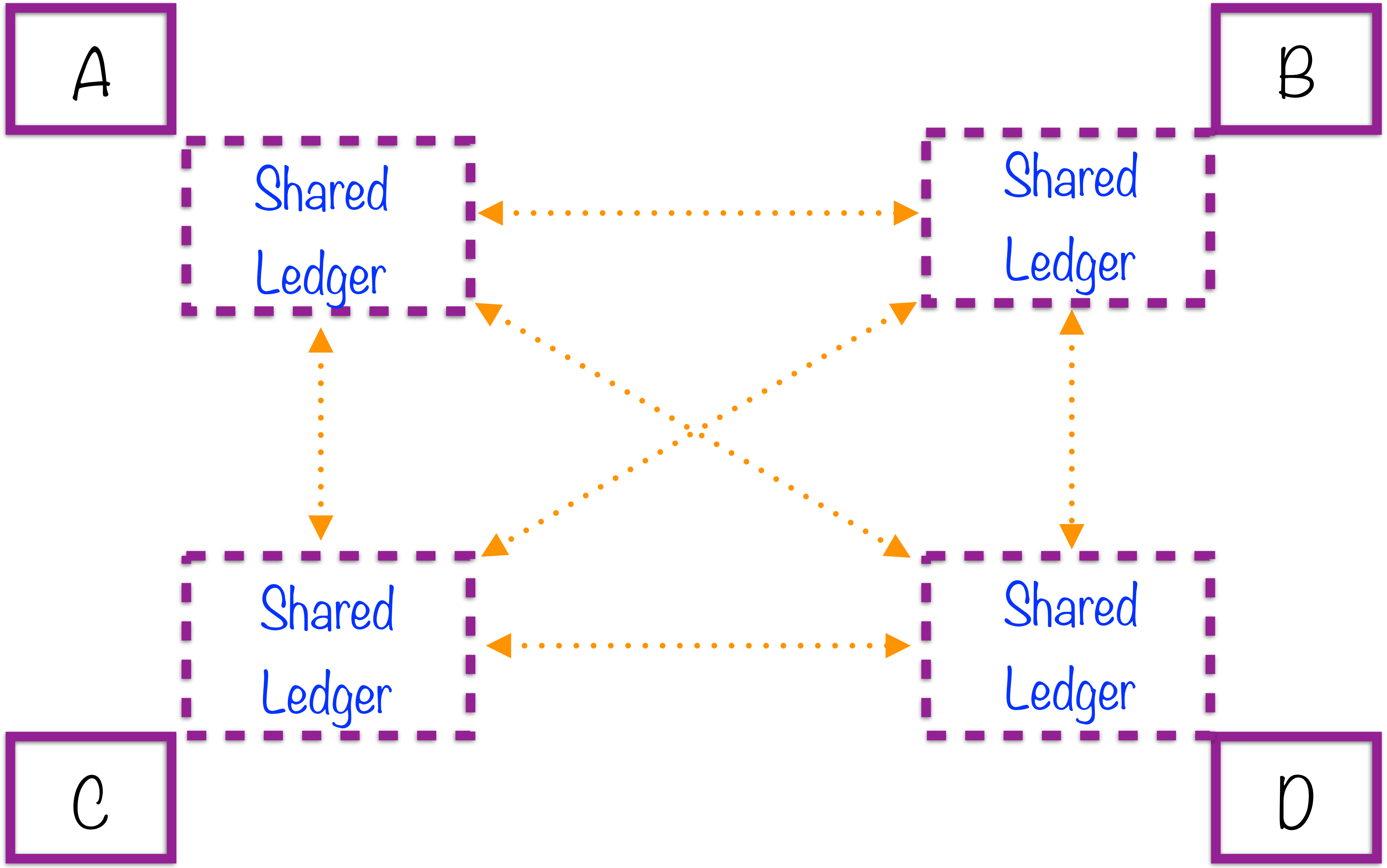
Blockchain

An open, distributed ledger that can record transactions in a verifiable and permanent manner

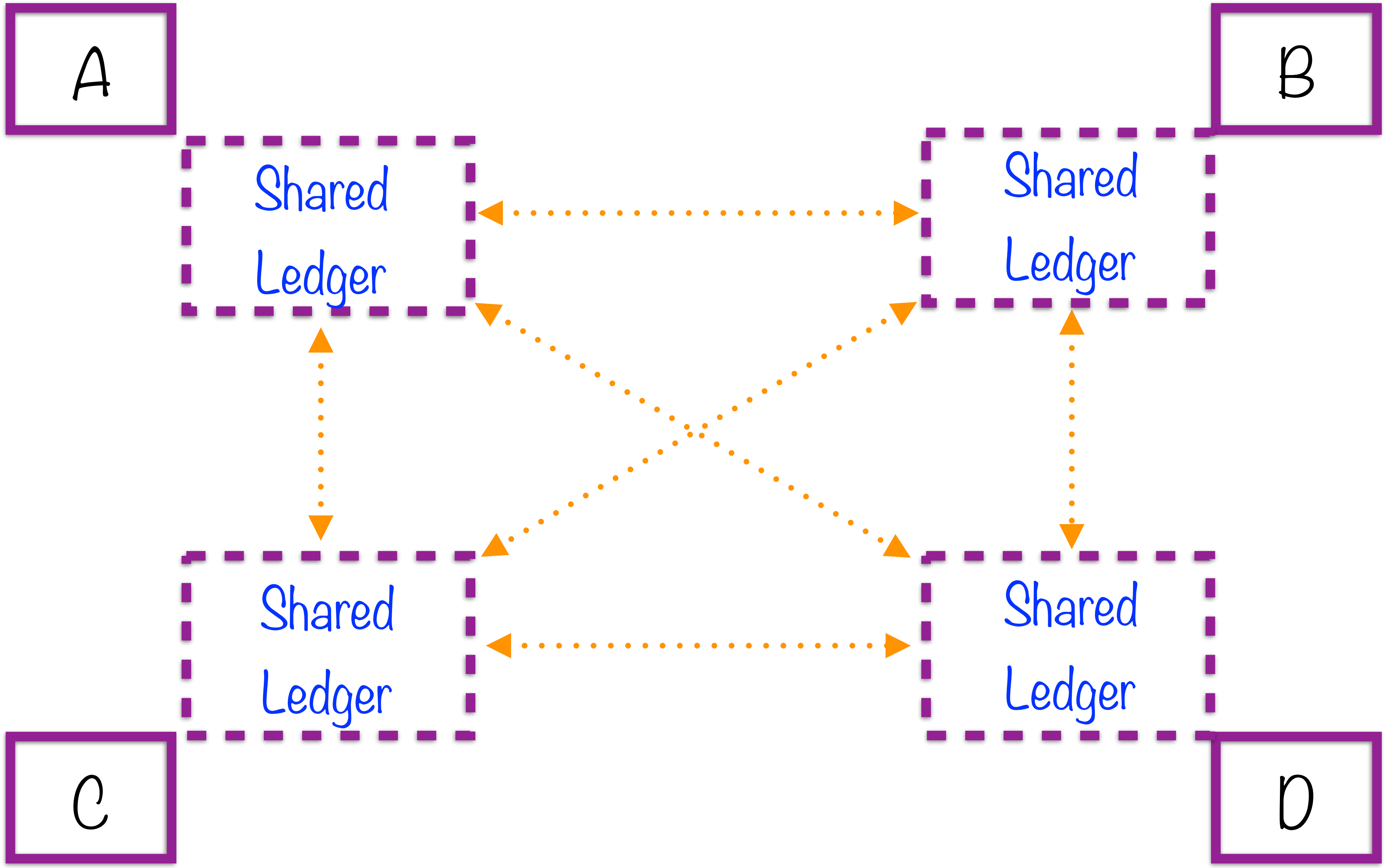
Centralized Ledger



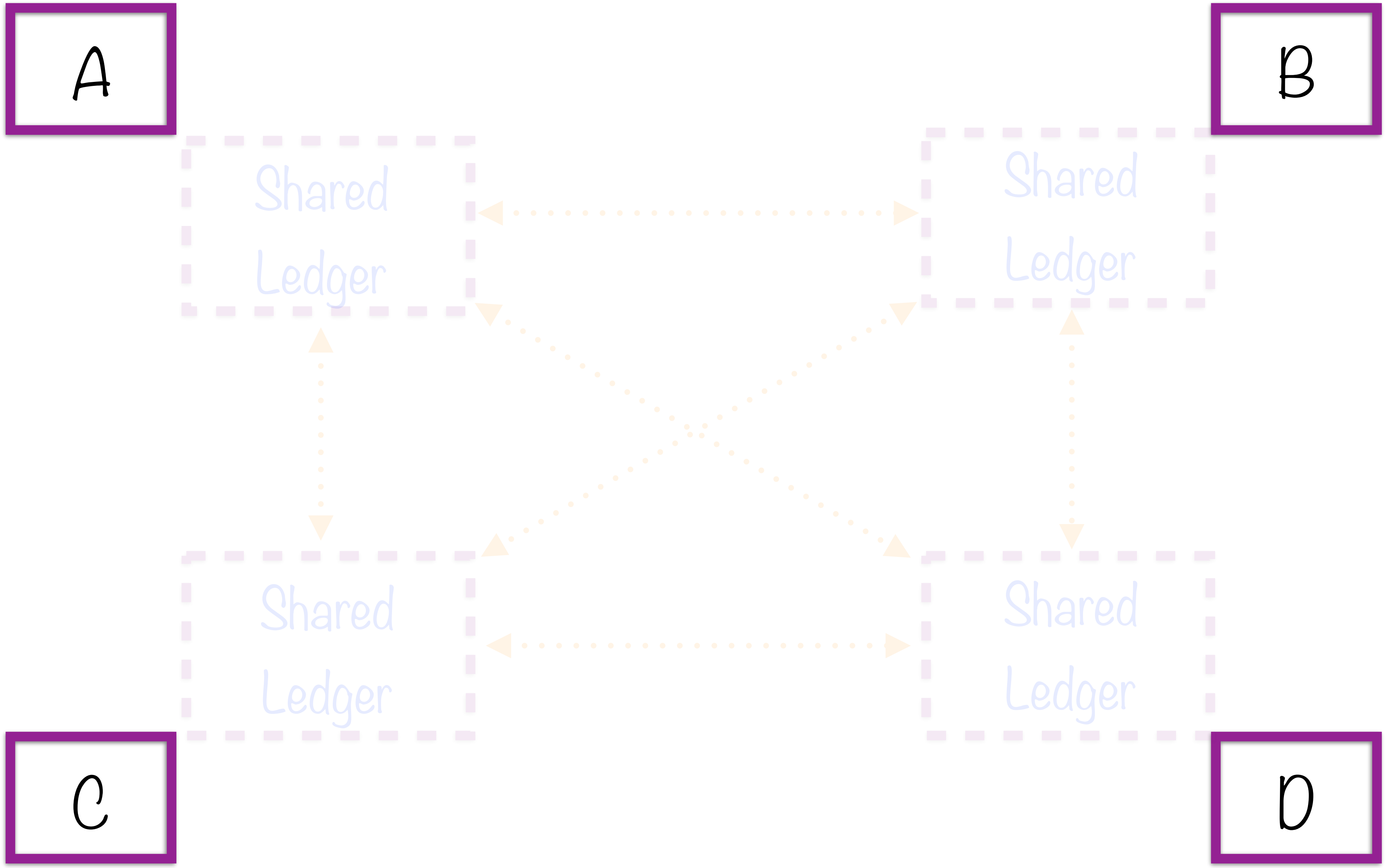
Distributed Ledger



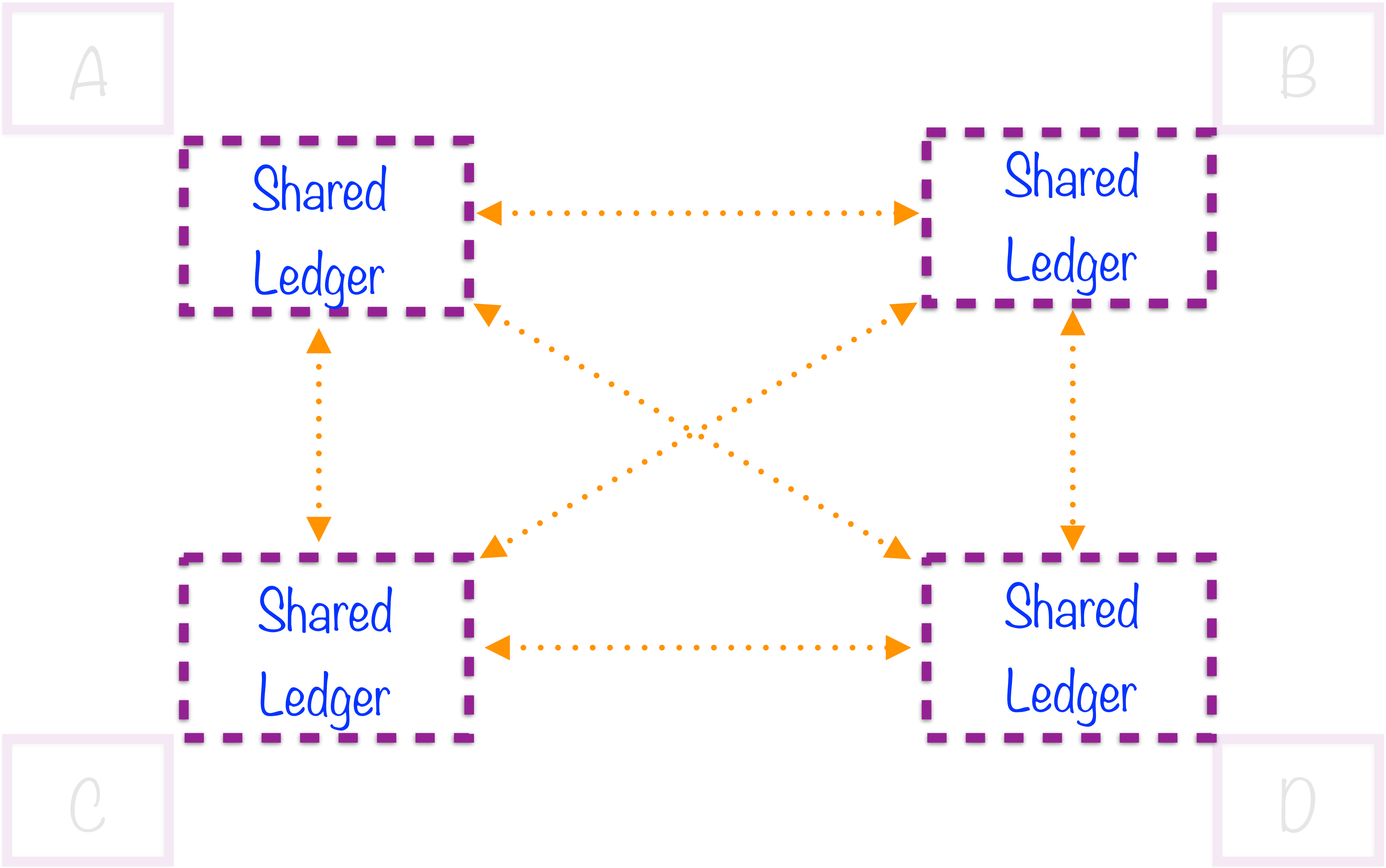
Blockchain Network



Nodes in a Blockchain Network



Blockchain


A
B
Shared Ledger
Shared Ledger
Shared Ledger
Shared Ledger
C
D

Public and Private Blockchain Networks

Public Network	Private Network
Truly open membership - anyone can join	Restricted entry - selection criteria
Lots of nodes	Fewer nodes
Slower to converge on consensus*	Faster convergence on consensus*
Higher probability of fraud	Lower probability of fraud

*Will be explained in just a bit

Hyperledger

Ethereum

Open-source, public blockchain-based platform that supports smart contracts

Smart Contract

Mechanism that allows common contractual clauses to be specified, verified or enforced even in the absence of trust between contracting parties and in the absence of a third party

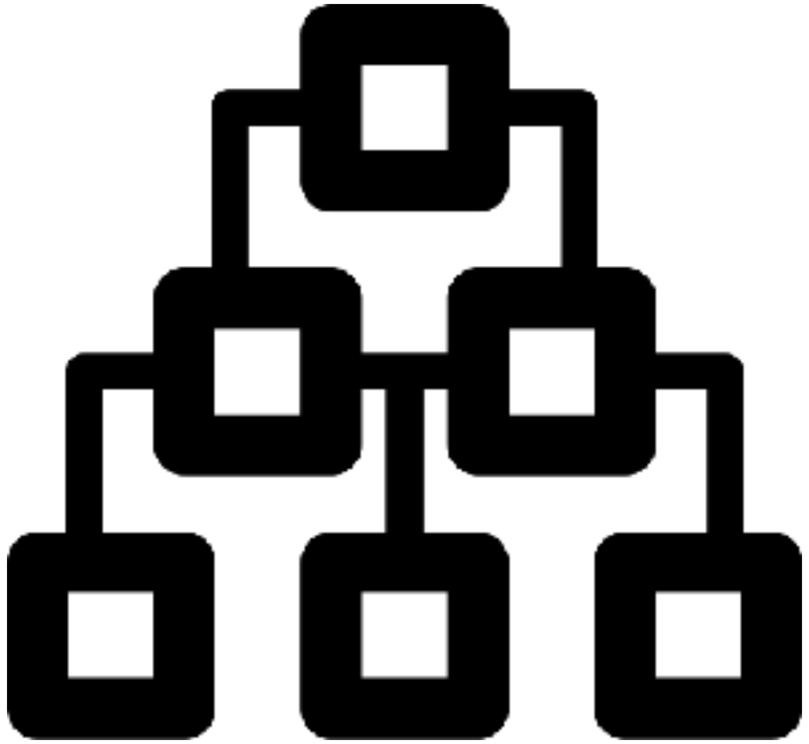
Ether

Cryptocurrency whose blockchain is generated by Ethereum and is used to compensate nodes on Ethereum for their participation

Ethereum Virtual Machine (EVM)

Execution environment that all Ethereum nodes run

Nodes



Each node runs the Ethereum Virtual Machine (EVM)

EVM is runtime environment responsible for

- Executing contract code
- Calculating transaction complexity (gas consumption)
- Verifying transactions

Concerns Around Ethereum

Scaling

Each peer node executes all transactions

Resource Usage

Proof-of-work algorithm incurs significant wasted effort

Confidentiality

Each transaction is broadcast to all peers in network

Hyperledger is designed to mitigate these concerns

Hyperledger

Umbrella project of open source blockchains; started in December 2015 by the Linux Foundation

Umbrella of Frameworks

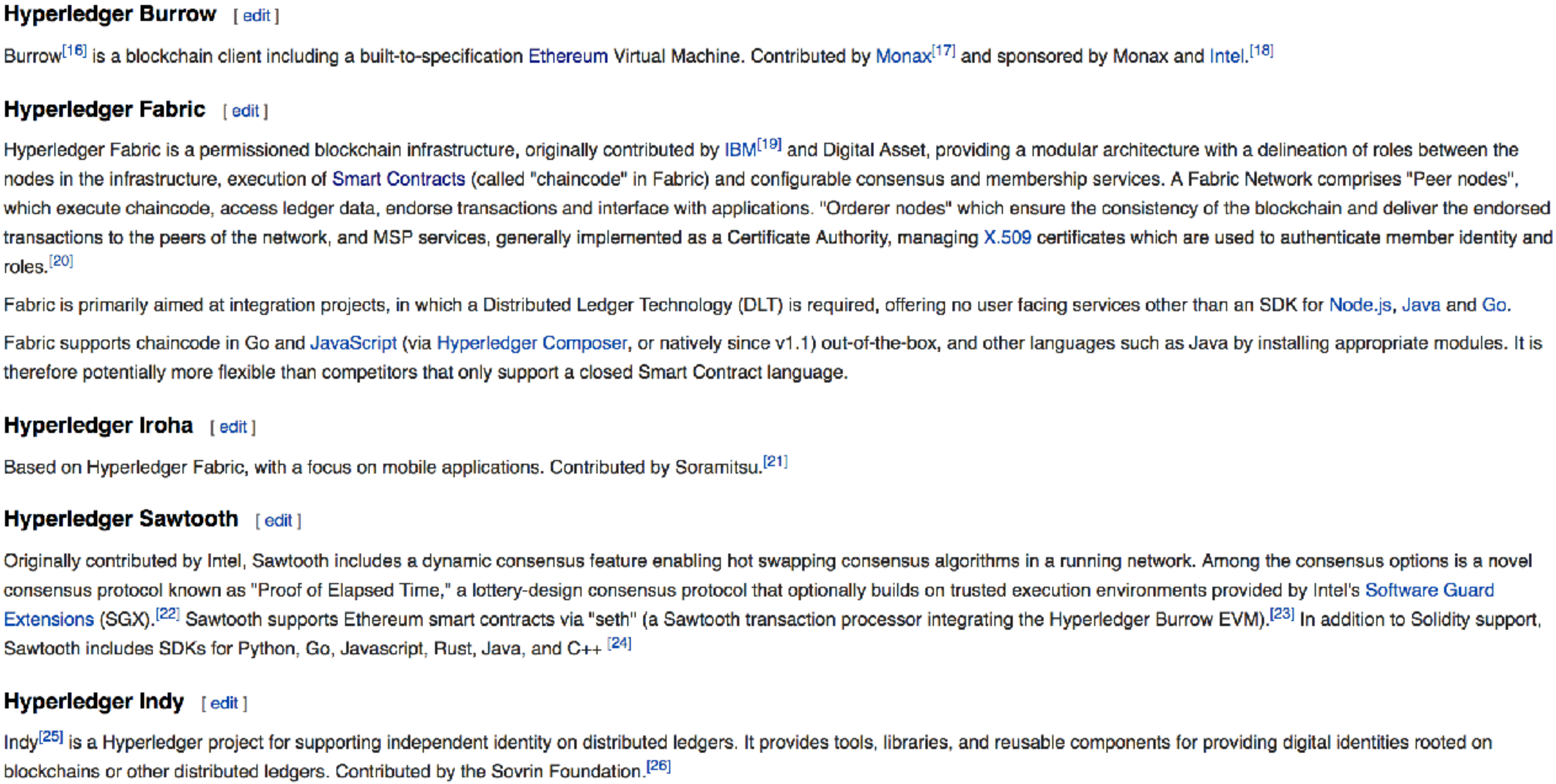
Hyperledger Fabric

Hyperledger Sawtooth

Hyperledger Iroha

Hyperledger Burro

Hyperledger Indy



Hyperledger Burrow [edit]

Burrow[16] is a blockchain client including a built-to-specification Ethereum Virtual Machine. Contributed by Monax[17] and sponsored by Monax and Intel.[18]

Hyperledger Fabric [edit]

Hyperledger Fabric is a permissioned blockchain infrastructure, originally contributed by IBM[19] and Digital Asset, providing a modular architecture with a delineation of roles between the nodes in the infrastructure, execution of Smart Contracts (called "chaincode" in Fabric) and configurable consensus and membership services. A Fabric Network comprises "Peer nodes", which execute chaincode, access ledger data, endorse transactions and interface with applications. "Orderer nodes" which ensure the consistency of the blockchain and deliver the endorsed transactions to the peers of the network, and MSP services, generally implemented as a Certificate Authority, managing X.509 certificates which are used to authenticate member identity and roles.[20]

Fabric is primarily aimed at integration projects, in which a Distributed Ledger Technology (DLT) is required, offering no user facing services other than an SDK for Node.js, Java and Go.

Fabric supports chaincode in Go and JavaScript (via Hyperledger Composer, or natively since v1.1) out-of-the-box, and other languages such as Java by installing appropriate modules. It is therefore potentially more flexible than competitors that only support a closed Smart Contract language.

Hyperledger Iroha [edit]

Based on Hyperledger Fabric, with a focus on mobile applications. Contributed by Soramitsu.[21]

Hyperledger Sawtooth [edit]

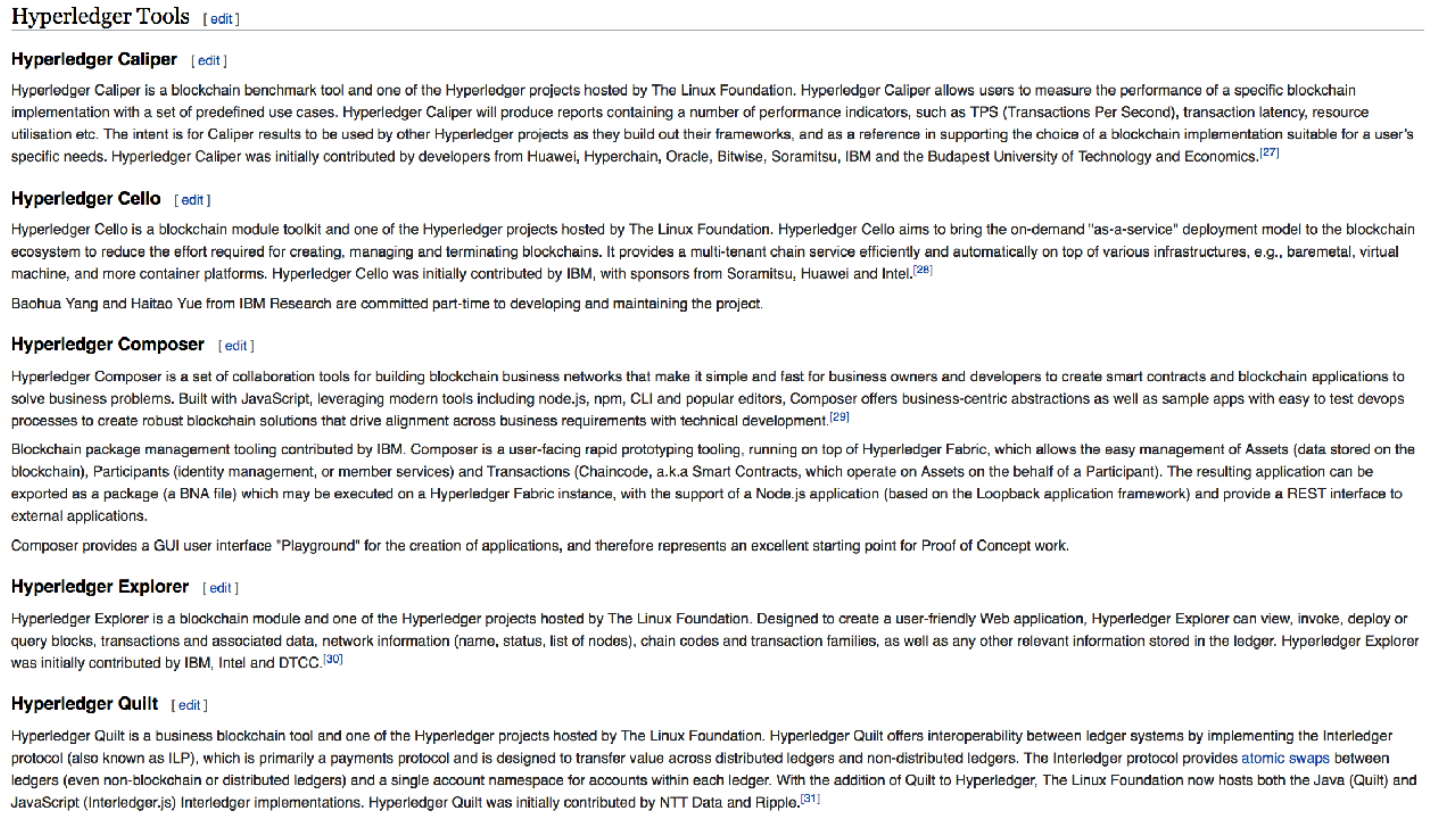
Originally contributed by Intel, Sawtooth includes a dynamic consensus feature enabling hot swapping consensus algorithms in a running network. Among the consensus options is a novel consensus protocol known as "Proof of Elapsed Time," a lottery-design consensus protocol that optionally builds on trusted execution environments provided by Intel's Software Guard Extensions (SGX).[22] Sawtooth supports Ethereum smart contracts via "seth" (a Sawtooth transaction processor integrating the Hyperledger Burrow EVM).[23] In addition to Solidity support, Sawtooth includes SDKs for Python, Go, Javascript, Rust, Java, and C++ [24]

Hyperledger Indy [edit]

Indy[25] is a Hyperledger project for supporting independent identity on distributed ledgers. It provides tools, libraries, and reusable components for providing digital identities rooted on blockchains or other distributed ledgers. Contributed by the Sovrin Foundation.[26]

Umbrella of Tools

- Hyperledger Cello
- Hyperledger Composer
- Hyperledger Caliper
- Hyperledger Explorer
- Hyperledger Quilt



Hyperledger Tools [edit]

Hyperledger Caliper [edit]

Hyperledger Caliper is a blockchain benchmark tool and one of the Hyperledger projects hosted by The Linux Foundation. Hyperledger Caliper allows users to measure the performance of a specific blockchain implementation with a set of predefined use cases. Hyperledger Caliper will produce reports containing a number of performance indicators, such as TPS (Transactions Per Second), transaction latency, resource utilisation etc. The intent is for Caliper results to be used by other Hyperledger projects as they build out their frameworks, and as a reference in supporting the choice of a blockchain implementation suitable for a user's specific needs. Hyperledger Caliper was initially contributed by developers from Huawei, Hyperchain, Oracle, Bitwise, Soramitsu, IBM and the Budapest University of Technology and Economics.[27]

Hyperledger Cello [edit]

Hyperledger Cello is a blockchain module toolkit and one of the Hyperledger projects hosted by The Linux Foundation. Hyperledger Cello aims to bring the on-demand "as-a-service" deployment model to the blockchain ecosystem to reduce the effort required for creating, managing and terminating blockchains. It provides a multi-tenant chain service efficiently and automatically on top of various infrastructures, e.g., baremetal, virtual machine, and more container platforms. Hyperledger Cello was initially contributed by IBM, with sponsors from Soramitsu, Huawei and Intel.[28]

Baohua Yang and Haitao Yue from IBM Research are committed part-time to developing and maintaining the project.

Hyperledger Composer [edit]

Hyperledger Composer is a set of collaboration tools for building blockchain business networks that make it simple and fast for business owners and developers to create smart contracts and blockchain applications to solve business problems. Built with JavaScript, leveraging modern tools including node.js, npm, CLI and popular editors, Composer offers business-centric abstractions as well as sample apps with easy to test devops processes to create robust blockchain solutions that drive alignment across business requirements with technical development.[29]

Blockchain package management tooling contributed by IBM. Composer is a user-facing rapid prototyping tooling, running on top of Hyperledger Fabric, which allows the easy management of Assets (data stored on the blockchain), Participants (identity management, or member services) and Transactions (Chaincode, a.k.a Smart Contracts, which operate on Assets on the behalf of a Participant). The resulting application can be exported as a package (a BNA file) which may be executed on a Hyperledger Fabric instance, with the support of a Node.js application (based on the Loopback application framework) and provide a REST interface to external applications.

Composer provides a GUI user interface "Playground" for the creation of applications, and therefore represents an excellent starting point for Proof of Concept work.

Hyperledger Explorer [edit]

Hyperledger Explorer is a blockchain module and one of the Hyperledger projects hosted by The Linux Foundation. Designed to create a user-friendly Web application, Hyperledger Explorer can view, invoke, deploy or query blocks, transactions and associated data, network information (name, status, list of nodes), chain codes and transaction families, as well as any other relevant information stored in the ledger. Hyperledger Explorer was initially contributed by IBM, Intel and DTCC.[30]

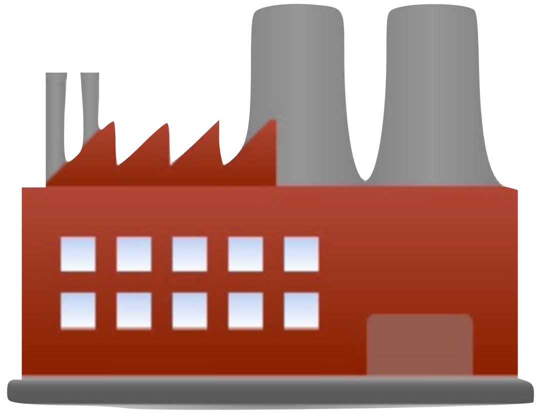
Hyperledger Quilt [edit]

Hyperledger Quilt is a business blockchain tool and one of the Hyperledger projects hosted by The Linux Foundation. Hyperledger Quilt offers interoperability between ledger systems by implementing the Interledger protocol (also known as ILP), which is primarily a payments protocol and is designed to transfer value across distributed ledgers and non-distributed ledgers. The Interledger protocol provides atomic swaps between ledgers (even non-blockchain or distributed ledgers) and a single account namespace for accounts within each ledger. With the addition of Quilt to Hyperledger, The Linux Foundation now hosts both the Java (Quilt) and JavaScript (Interledger.js) Interledger implementations. Hyperledger Quilt was initially contributed by NTT Data and Ripple.[31]

Collaborative Umbrella Project

Framework

Developers



Industries

Communities of software developer and companies meet and coordinate to build blockchain frameworks

Hyperledger Fabric

Permissioned blockchain

Restricted access

Great for private channels

Supports multiple

- consensus algorithms
- membership services

Hyperledger Fabric

No mining

No associated cryptocurrency

High throughput

Hyperledger Fabric

Written in Go

IBM origins

Most active, stable

Wide commercial usage

Hyperledger Sawtooth

Designed for broad, flat networks

Proof of Elapsed Time

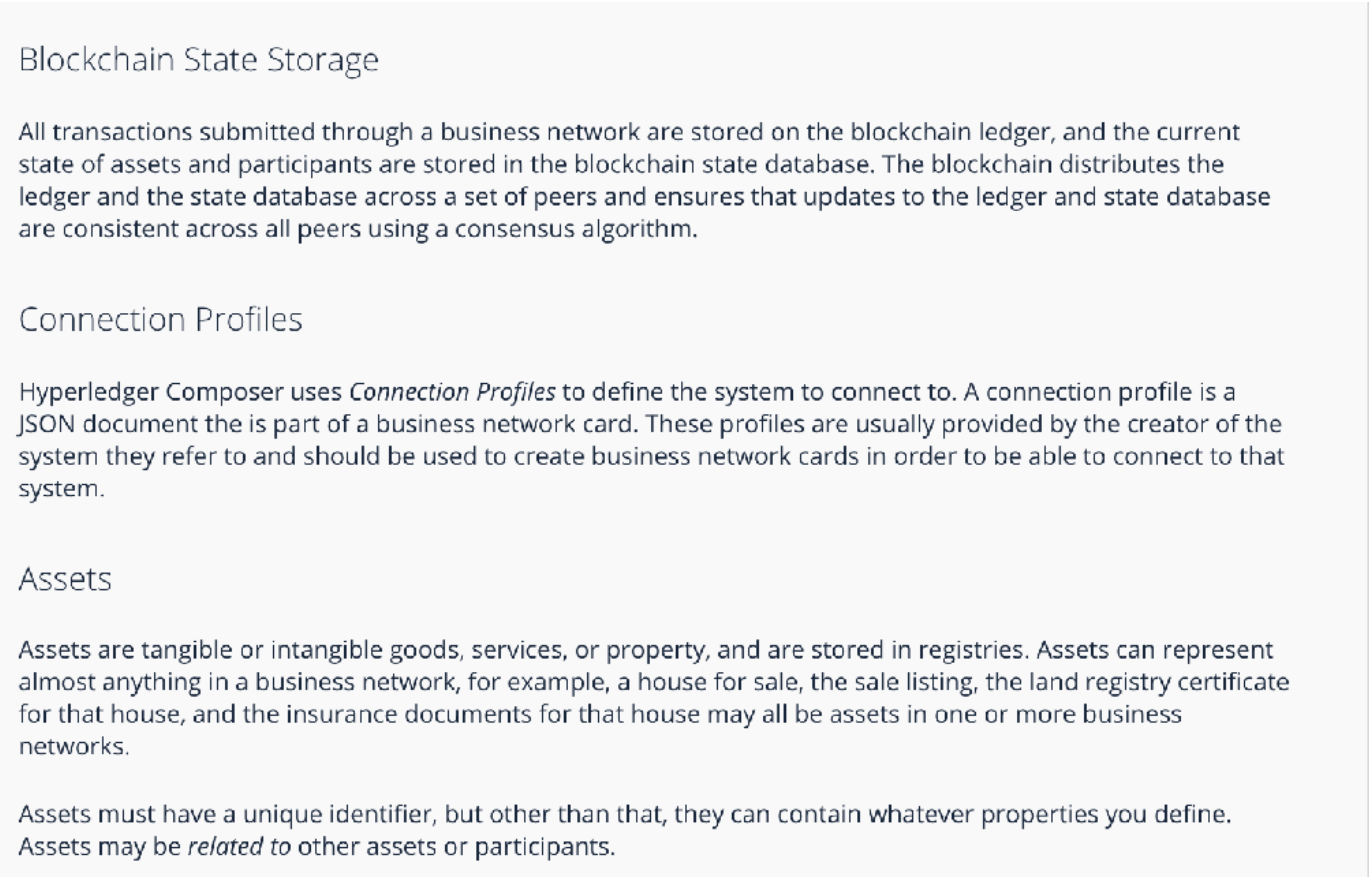
Originated at Intel

Hyperledger Composer

Browser-based UI to build apps

Allows easy specification of

- Assets
- Participants
- Transactions
- Blockchain state storage



Blockchain State Storage

All transactions submitted through a business network are stored on the blockchain ledger, and the current state of assets and participants are stored in the blockchain state database. The blockchain distributes the ledger and the state database across a set of peers and ensures that updates to the ledger and state database are consistent across all peers using a consensus algorithm.

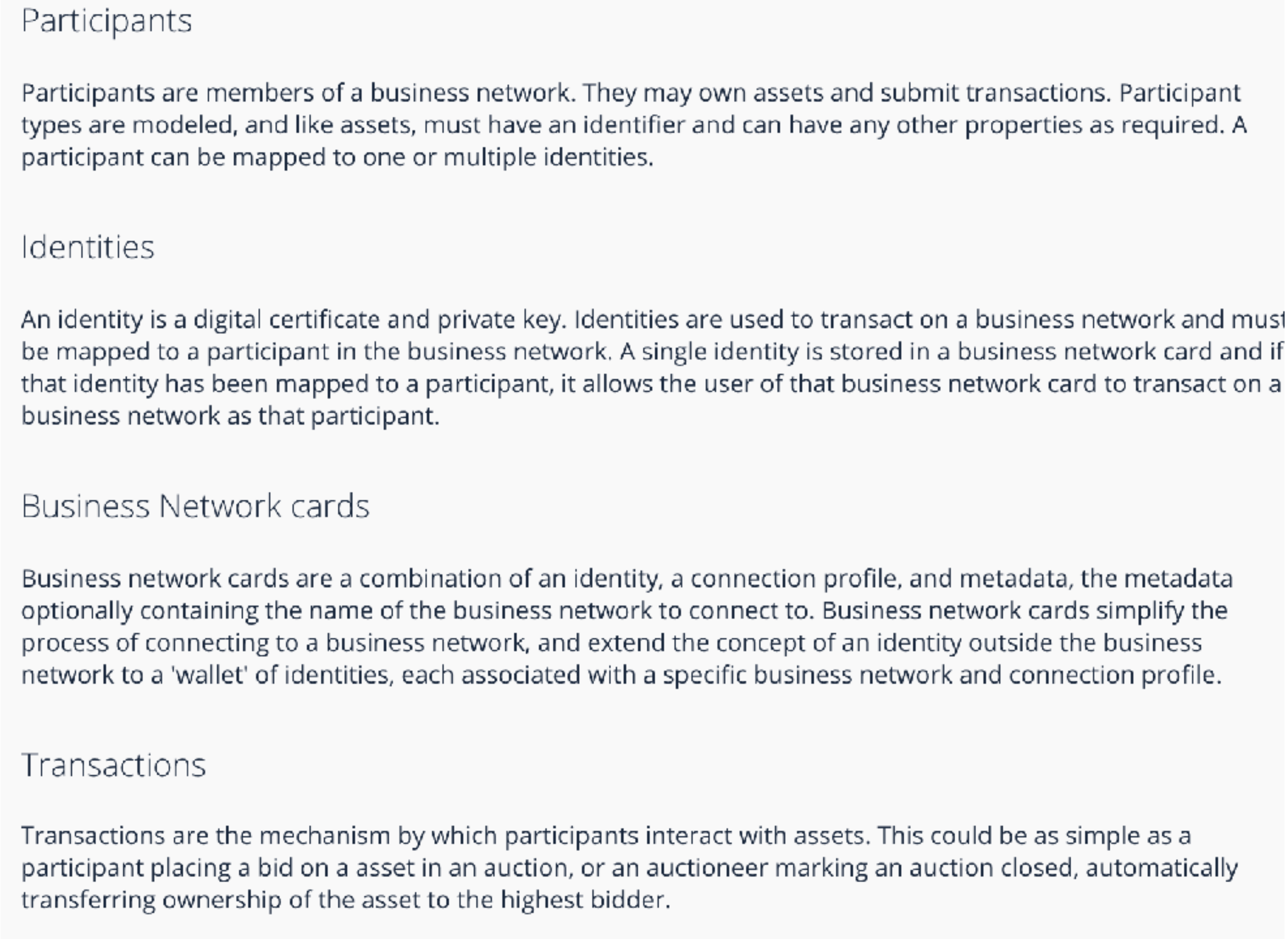
Connection Profiles

Hyperledger Composer uses *Connection Profiles* to define the system to connect to. A connection profile is a JSON document the is part of a business network card. These profiles are usually provided by the creator of the system they refer to and should be used to create business network cards in order to be able to connect to that system.

Assets

Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house, and the insurance documents for that house may all be assets in one or more business networks.

Assets must have a unique identifier, but other than that, they can contain whatever properties you define. Assets may be *related to* other assets or participants.



Participants

Participants are members of a business network. They may own assets and submit transactions. Participant types are modeled, and like assets, must have an identifier and can have any other properties as required. A participant can be mapped to one or multiple identities.

Identities

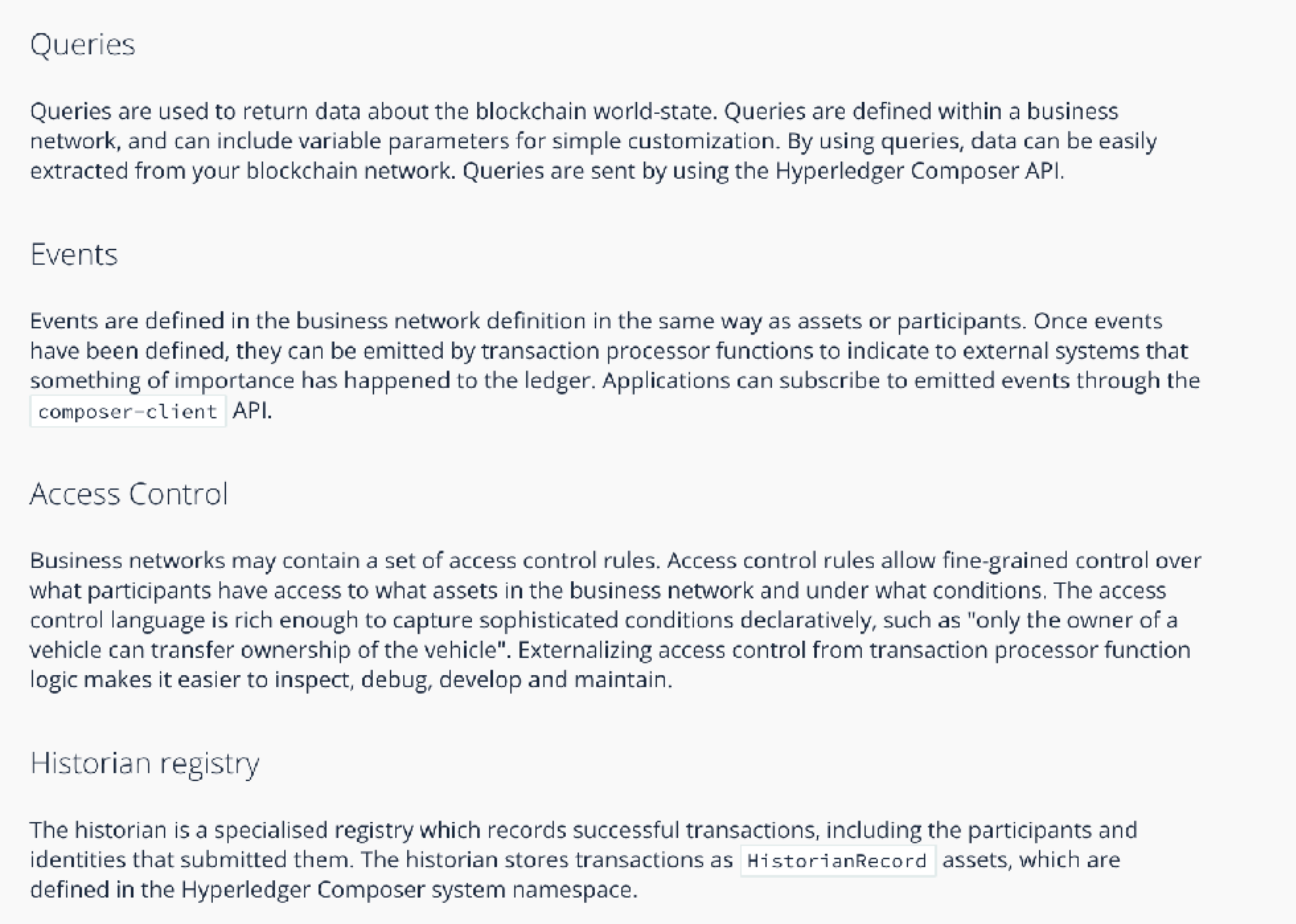
An identity is a digital certificate and private key. Identities are used to transact on a business network and must be mapped to a participant in the business network. A single identity is stored in a business network card and if that identity has been mapped to a participant, it allows the user of that business network card to transact on a business network as that participant.

Business Network cards

Business network cards are a combination of an identity, a connection profile, and metadata, the metadata optionally containing the name of the business network to connect to. Business network cards simplify the process of connecting to a business network, and extend the concept of an identity outside the business network to a 'wallet' of identities, each associated with a specific business network and connection profile.

Transactions

Transactions are the mechanism by which participants interact with assets. This could be as simple as a participant placing a bid on a asset in an auction, or an auctioneer marking an auction closed, automatically transferring ownership of the asset to the highest bidder.



Queries

Queries are used to return data about the blockchain world-state. Queries are defined within a business network, and can include variable parameters for simple customization. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.

Events

Events are defined in the business network definition in the same way as assets or participants. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger. Applications can subscribe to emitted events through the `composer-client` API.

Access Control

Business networks may contain a set of access control rules. Access control rules allow fine-grained control over what participants have access to what assets in the business network and under what conditions. The access control language is rich enough to capture sophisticated conditions declaratively, such as "only the owner of a vehicle can transfer ownership of the vehicle". Externalizing access control from transaction processor function logic makes it easier to inspect, debug, develop and maintain.

Historian registry

The historian is a specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as `HistorianRecord` assets, which are defined in the Hyperledger Composer system namespace.

Hyperledger Fabric vs. Ethereum

Hyperledger Fabric	Ethereum
Permissioned blockchain platform - not all users can validate transactions	Permissionless blockchain platform - all users can participate in validation
Private - entry by invitation only	Public - anyone can join

Hyperledger Fabric vs. Ethereum

Hyperledger Fabric	Ethereum
No associated cryptocurrency	Ether
Tokens exchanged via chaincode	Tokens exchanged via smart contracts

Hyperledger Fabric vs. Ethereum

Hyperledger Fabric

Transaction lifecycle: Execute-order-validate

Transactions can be kept private from some peers

Transactions can be executed in any order, even in parallel*

*Execution is followed by an explicit validation step

Ethereum

Transaction lifecycle: Order-execute (no separate validation phase)

Transactions are broadcast to all members of network

Transaction order must be agreed upon by all participants

No explicit validation step

Hyperledger Fabric vs. Ethereum

Hyperledger Fabric	Ethereum
Smart contracts without trusted third-party	Smart contracts without trusted third-party
Chaincode in general purpose languages (Go, Node.js, Java)	Smart contracts in domain-specific-language (DSL) e.g. Solidity

Hyperledger Fabric vs. Ethereum

Hyperledger Fabric	Ethereum
Broad, generic consensus mechanism	Proof-of-work (soon proof-of-scale)
Little wasted effort	Much wasted effort
Ease of scaling	Difficulty scaling

Concerns Around Ethereum

Scaling

Each peer node executes all transactions

Resource Usage

Proof-of-work algorithm incurs significant wasted effort

Confidentiality

Each transaction is broadcast to all peers in network

Hyperledger is designed to mitigate these concerns

Chaincode

Smart Contract

Mechanism that allows common contractual clauses to be specified, verified or enforced even in the absence of trust between contracting parties and in the absence of a third party

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

Implemented in Go,
Node.js or Java

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

Maintains its own private state

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

ChainCodeStubInterface which specifies Init and Invoke methods

Any peer on the Hyperledger Fabric network can invoke them

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

Chaincode

A self-contained program that runs on Hyperledger Fabric and satisfies a standard interface; typically used to implement smart contracts

Chaincode contains coded-up logic of the agreements between participants

Transaction Verification in Fabric

Consensus

Blockchain avoids use of trusted third parties

Verification needs to adhere to a carefully designed algorithm

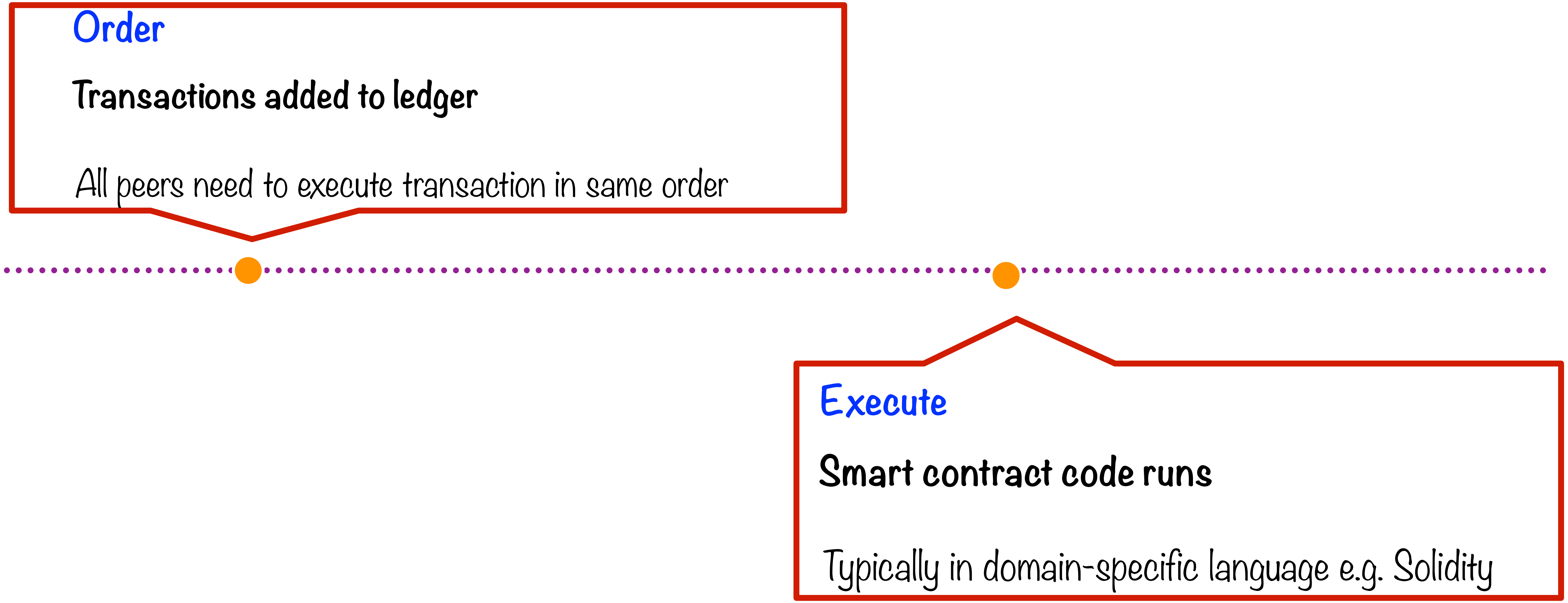
Ensure agreement among peers on network

“Consensus algorithms”

Consensus in Hyperledger Fabric: Execute-order-validate

Common Transaction Lifecycle

Most blockchain platforms operate in two steps



Order

Transactions added to ledger

All peers need to execute transaction in same order

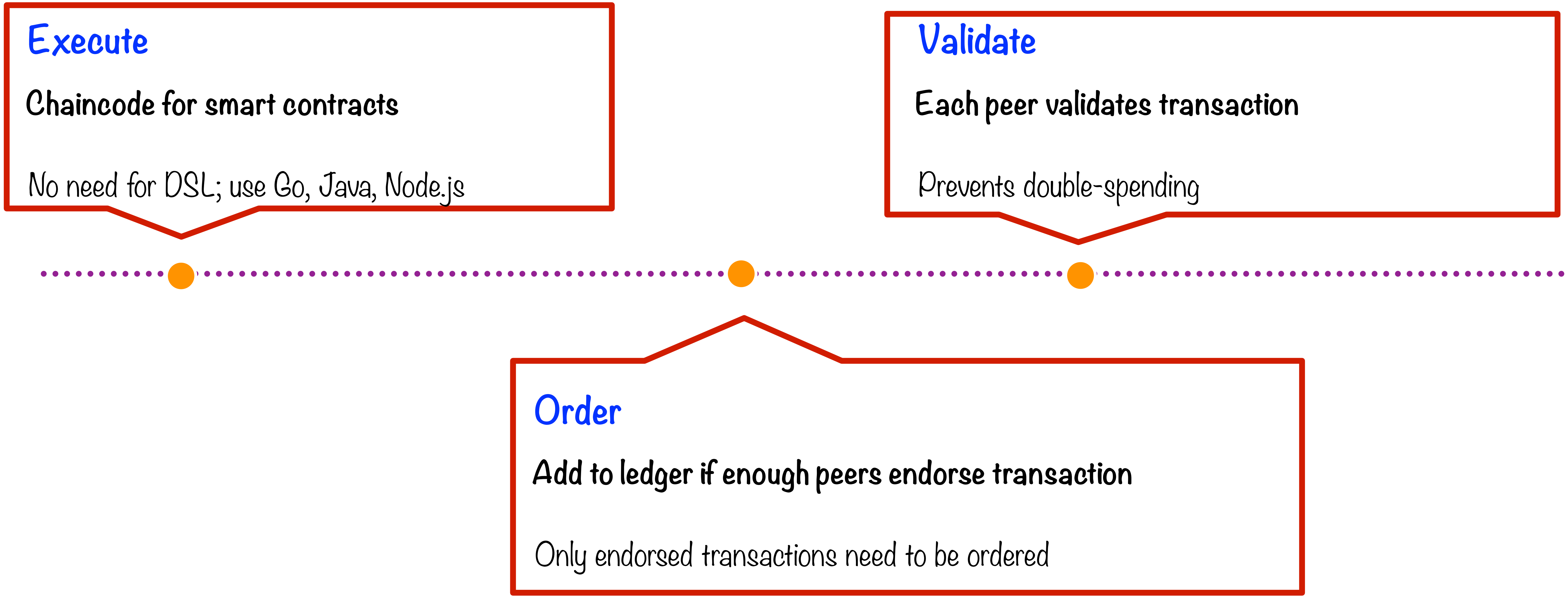
Execute

Smart contract code runs

Typically in domain-specific language e.g. Solidity

Common Transaction Lifecycle

Hyperledger follows a different, more elaborate process



Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Order

Add to ledger if enough peers endorse transaction

Only endorsed transactions need to be ordered

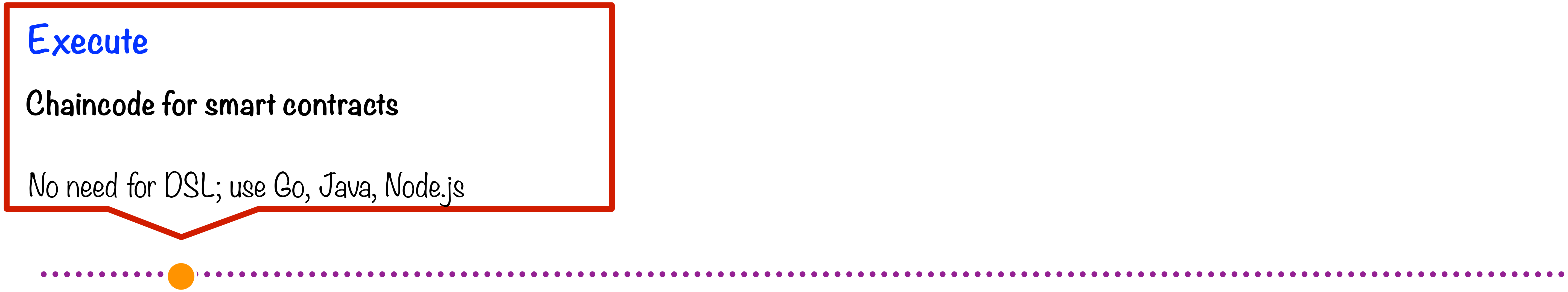
Validate

Each peer validates transaction

Prevents double-spending

Hyperledger Transaction Lifecycle

Hyperledger follows a different, more elaborate process

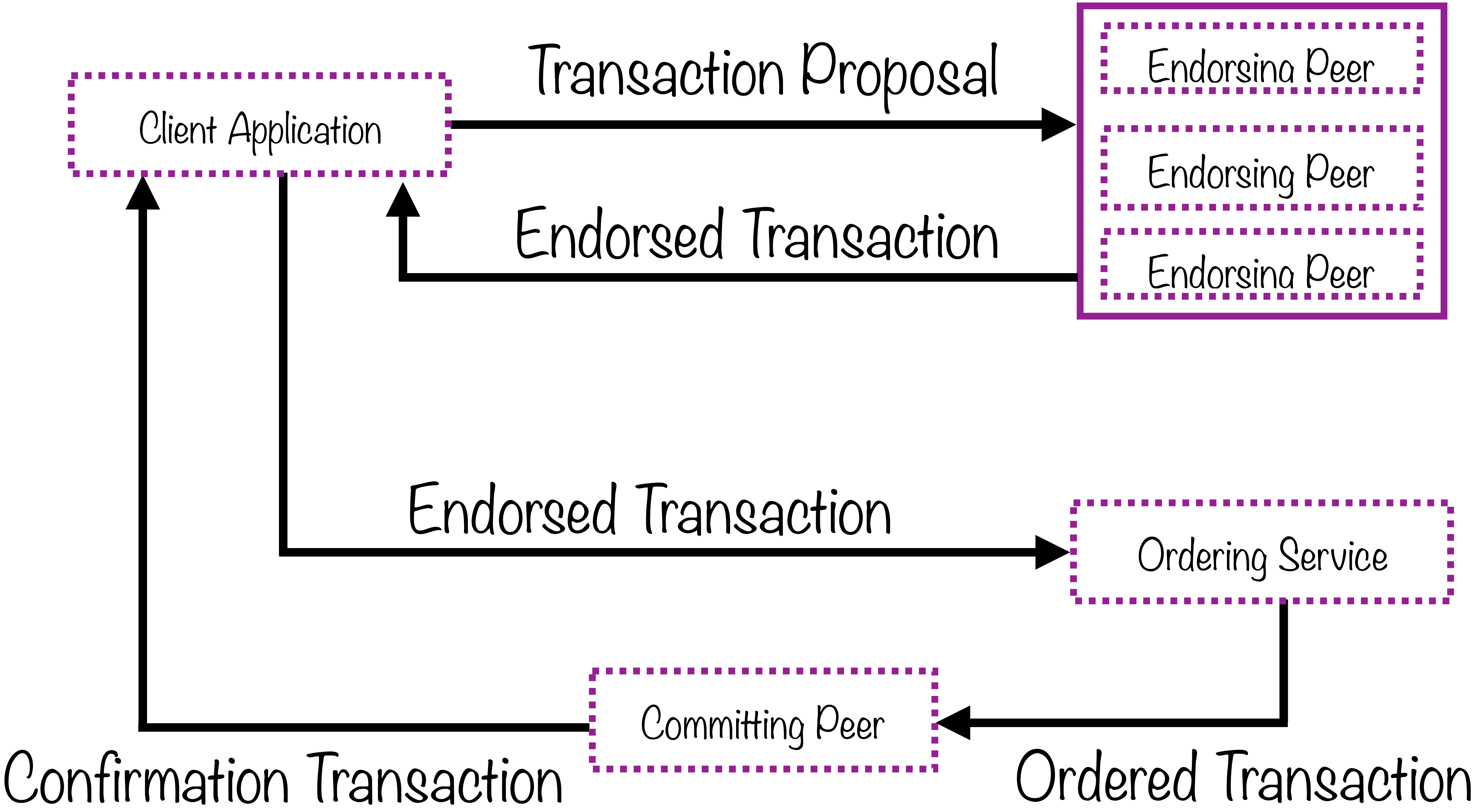


Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Transaction Flow in Fabric



Client Application

Transaction Proposal

Endorsing Peer

Endorsing Peer

Endorsing Peer

Endorsed Transaction

Endorsed Transaction

Ordering Service

Committing Peer

Confirmation Transaction

Ordered Transaction

Execution

Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Can happen in any order

Even in parallel

Relies on Endorsement Policy

Transaction Proposal

Execute

Chaincode for smart contracts

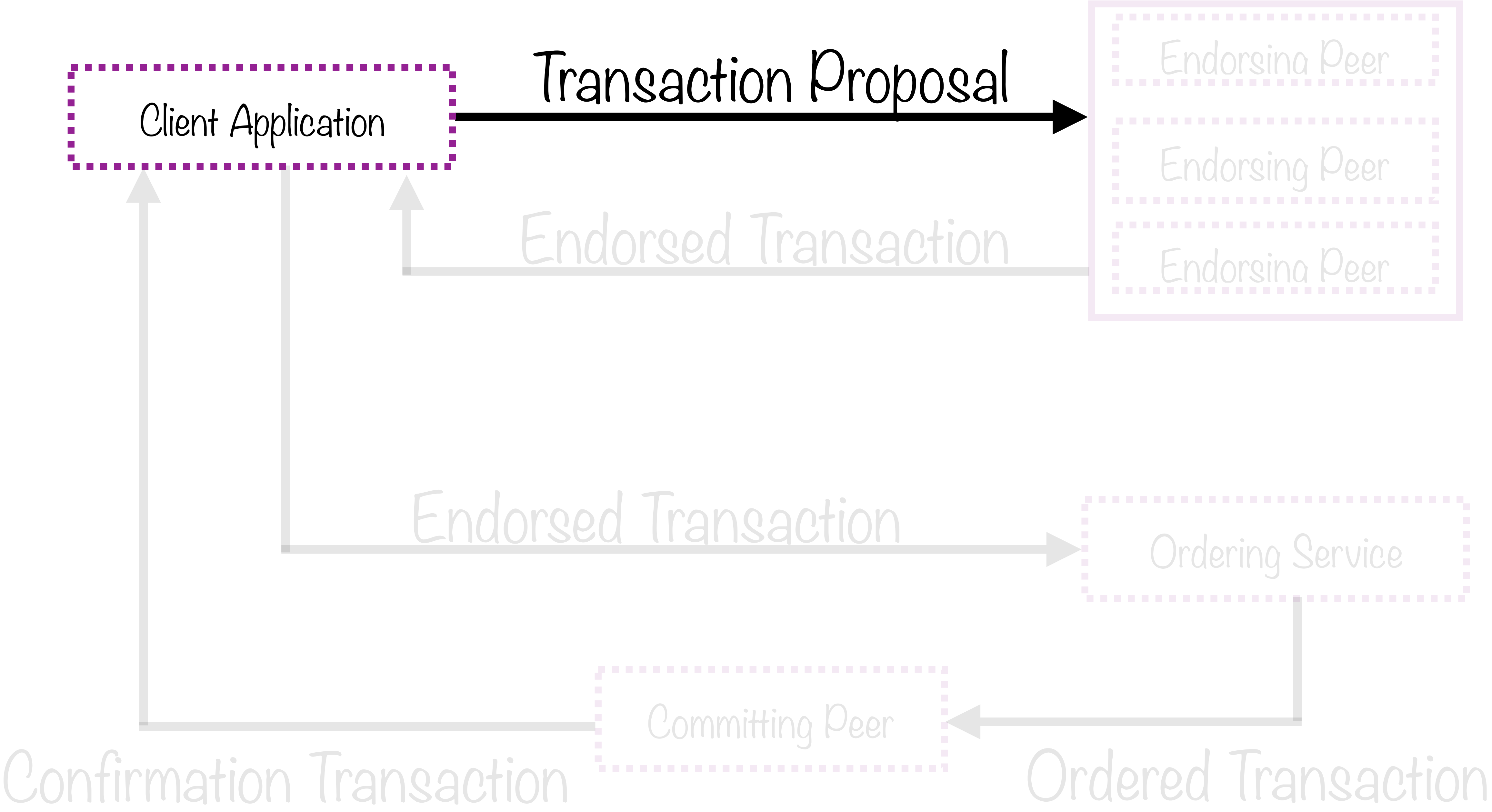
No need for DSL; use Go, Java, Node.js

Information needed to trigger chaincode

Does the proposal have support?

Check the endorsement policy

Transaction Flow in Fabric


Client Application

Transaction Proposal

Endorsing Peer

Endorsing Peer

Endorsing Peer

Endorsed Transaction

Endorsed Transaction

Ordering Service

Committing Peer

Confirmation Transaction

Ordered Transaction

Endorsement Policy

Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Specific policy to ensure some nodes agree on transaction results

Not all nodes need to be involved

Endorsement Policy

Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Only move to next step if transaction proposal is endorsed

Not every peer needs to be aware

Allows private transactions

Endorsement Policy

Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Examples

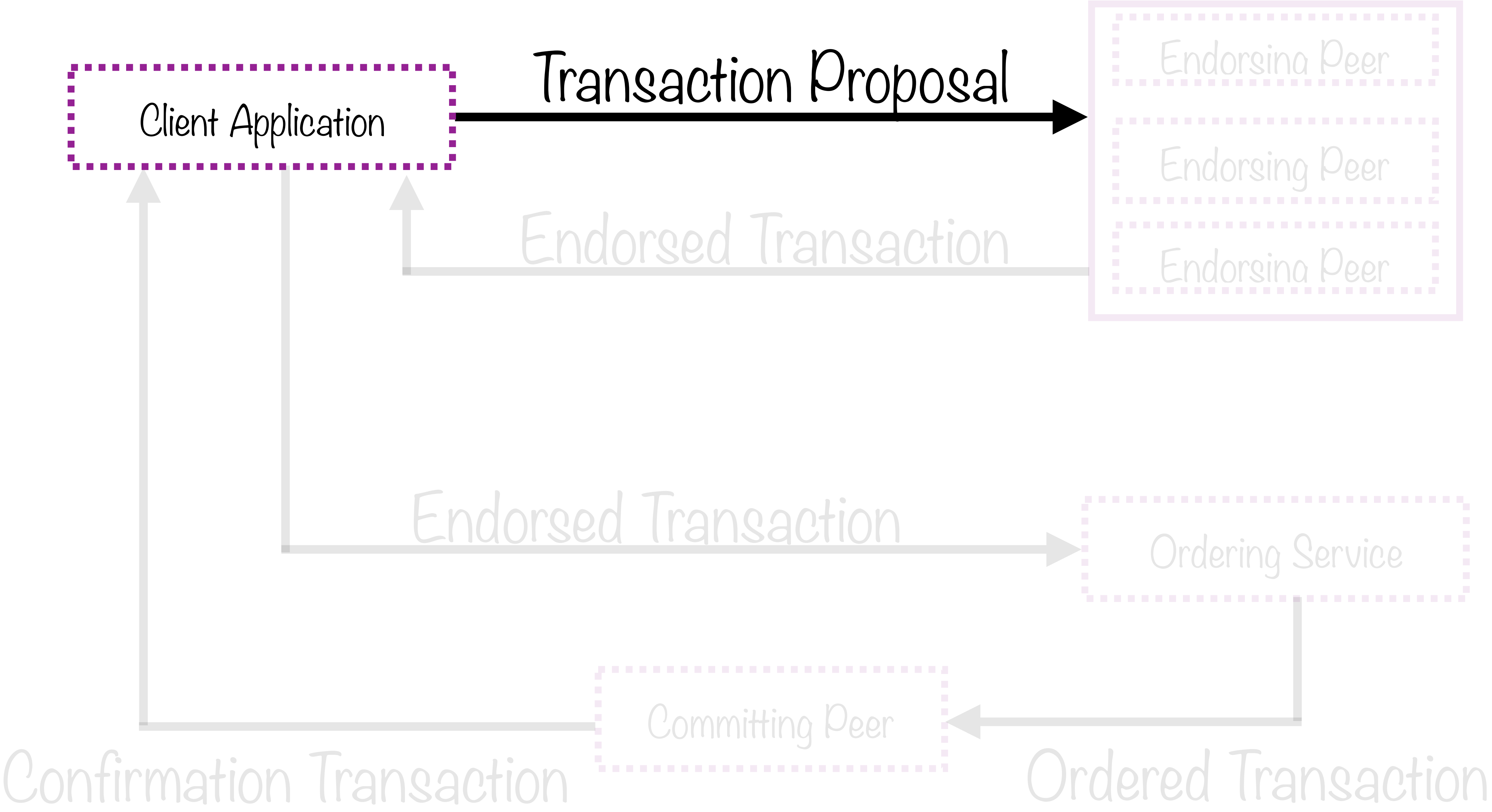
- “Specific list of peers must all endorse”
- “Majority of peers must endorse”
- “At least N peers in list must endorse”

Endorsement Policy

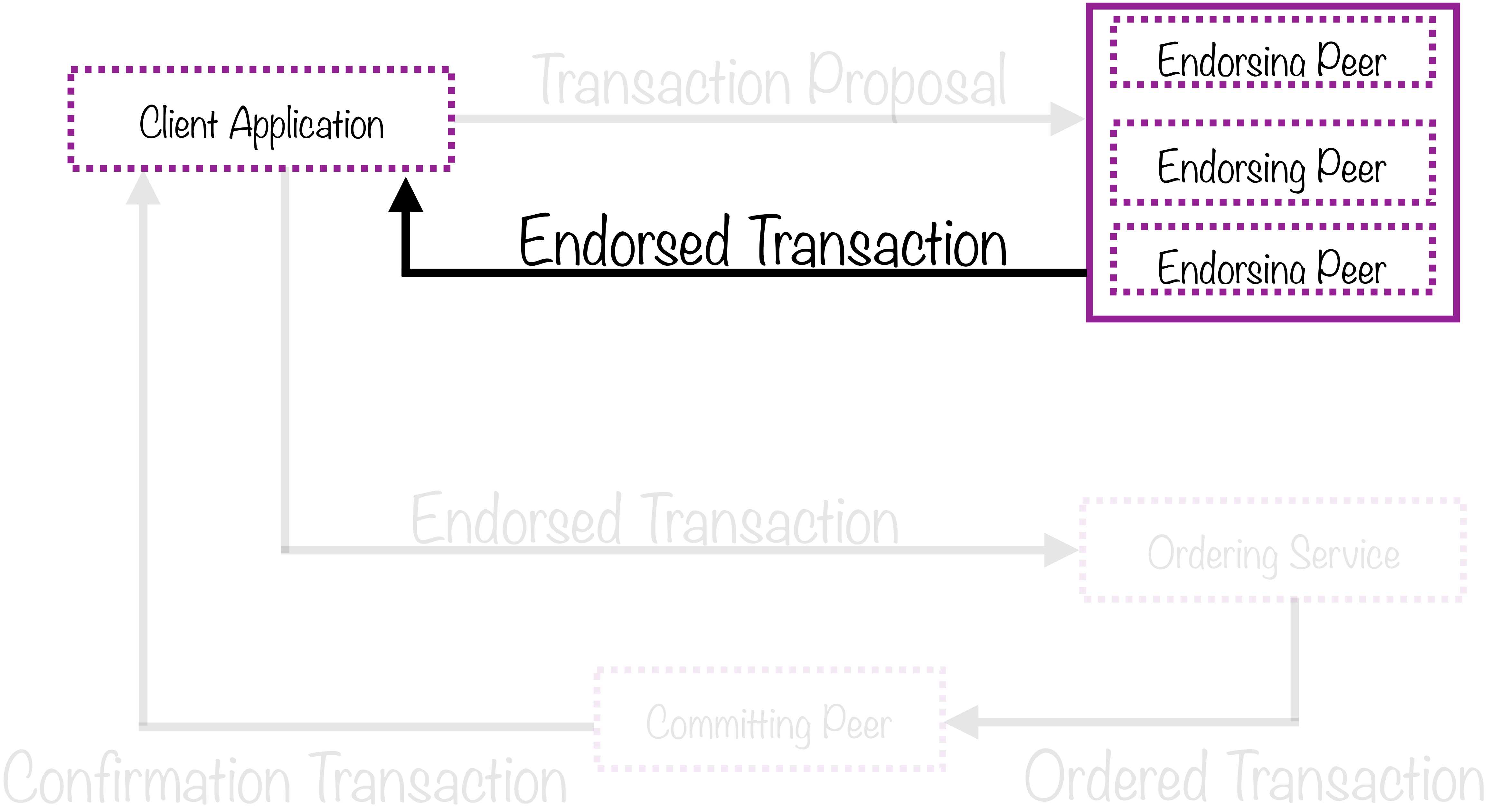
Need to actually execute smart contract

So, must have copy of chaincode

Transaction Flow in Fabric

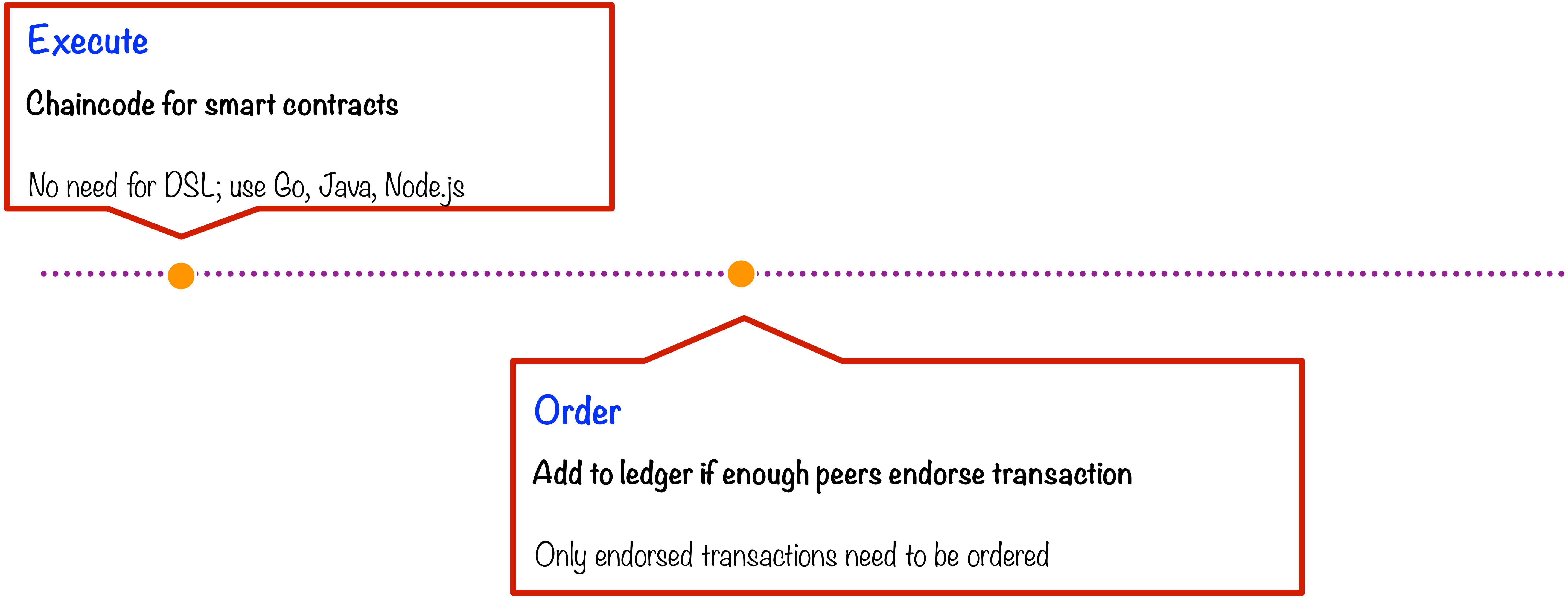


Transaction Flow in Fabric



Hyperledger Transaction Lifecycle

Hyperledger follows a different, more elaborate process



Execute

Chaincode for smart contracts

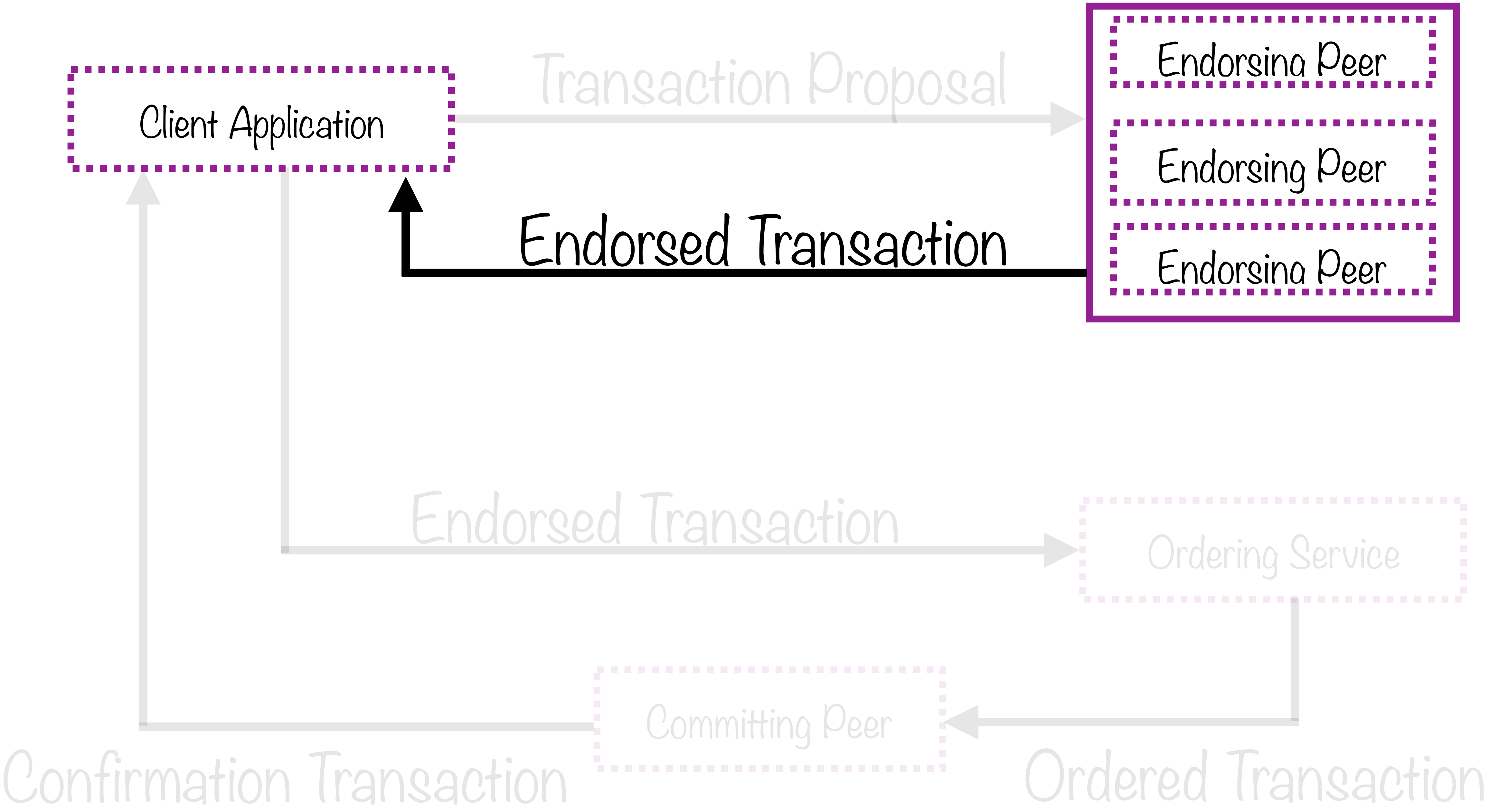
No need for DSL; use Go, Java, Node.js

Order

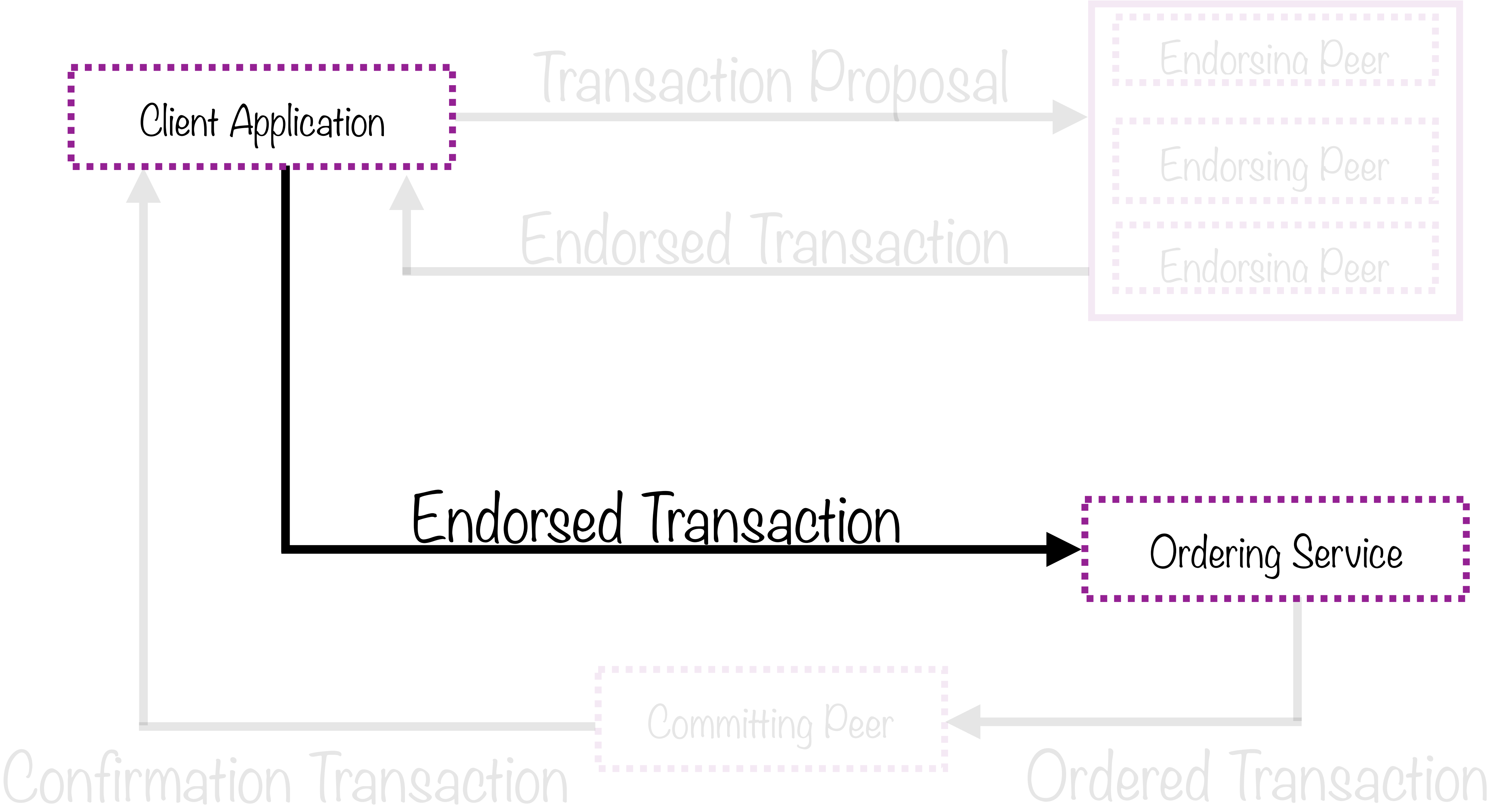
Add to ledger if enough peers endorse transaction

Only endorsed transactions need to be ordered

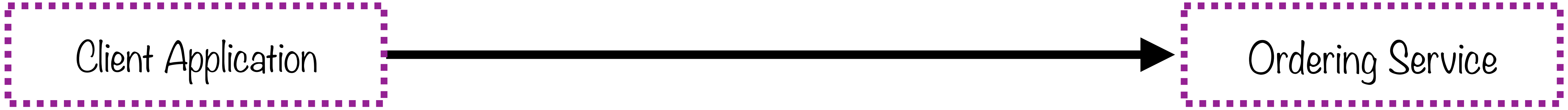
Transaction Flow in Fabric



Transaction Flow in Fabric

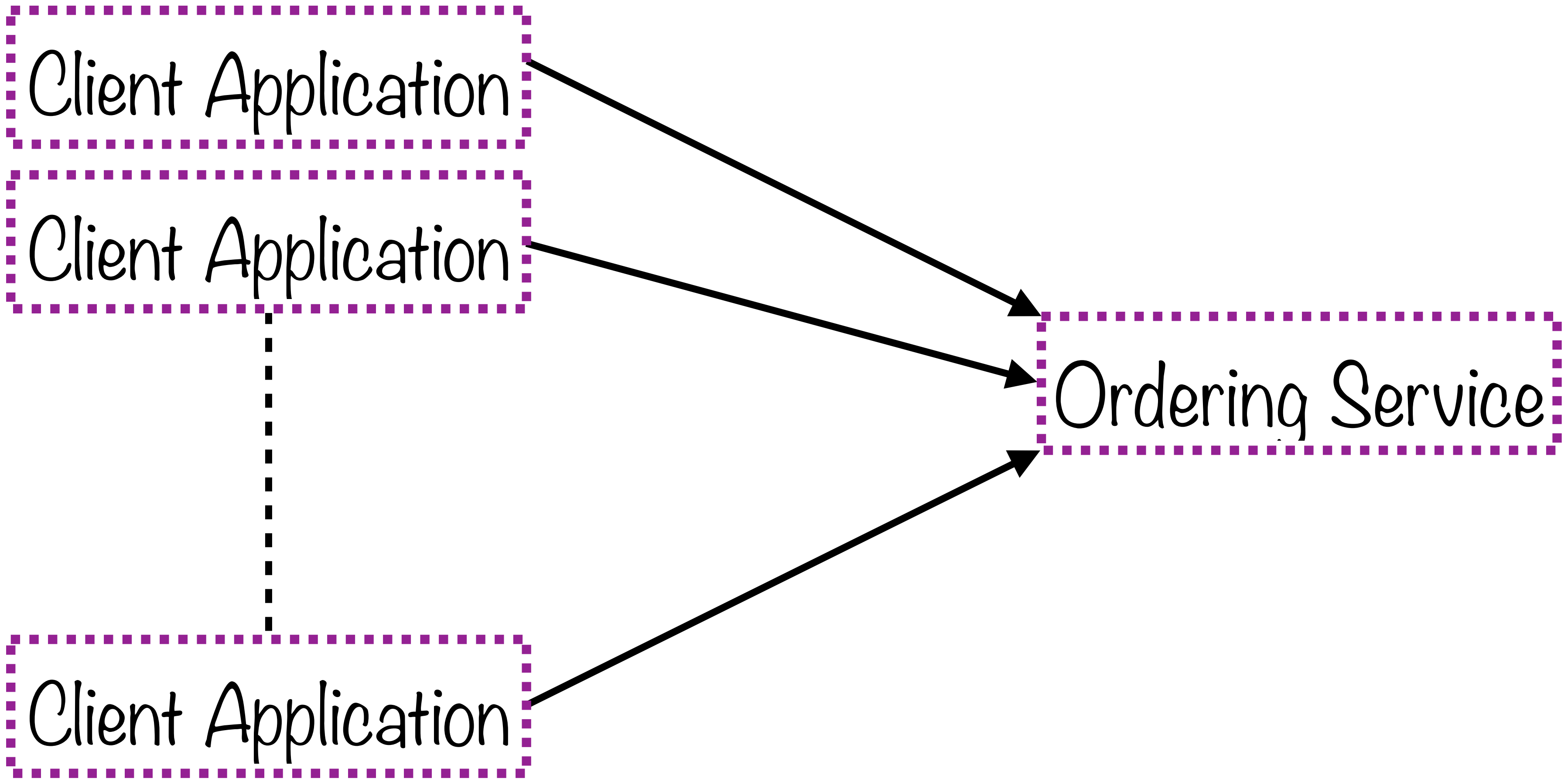


Transaction Ordering



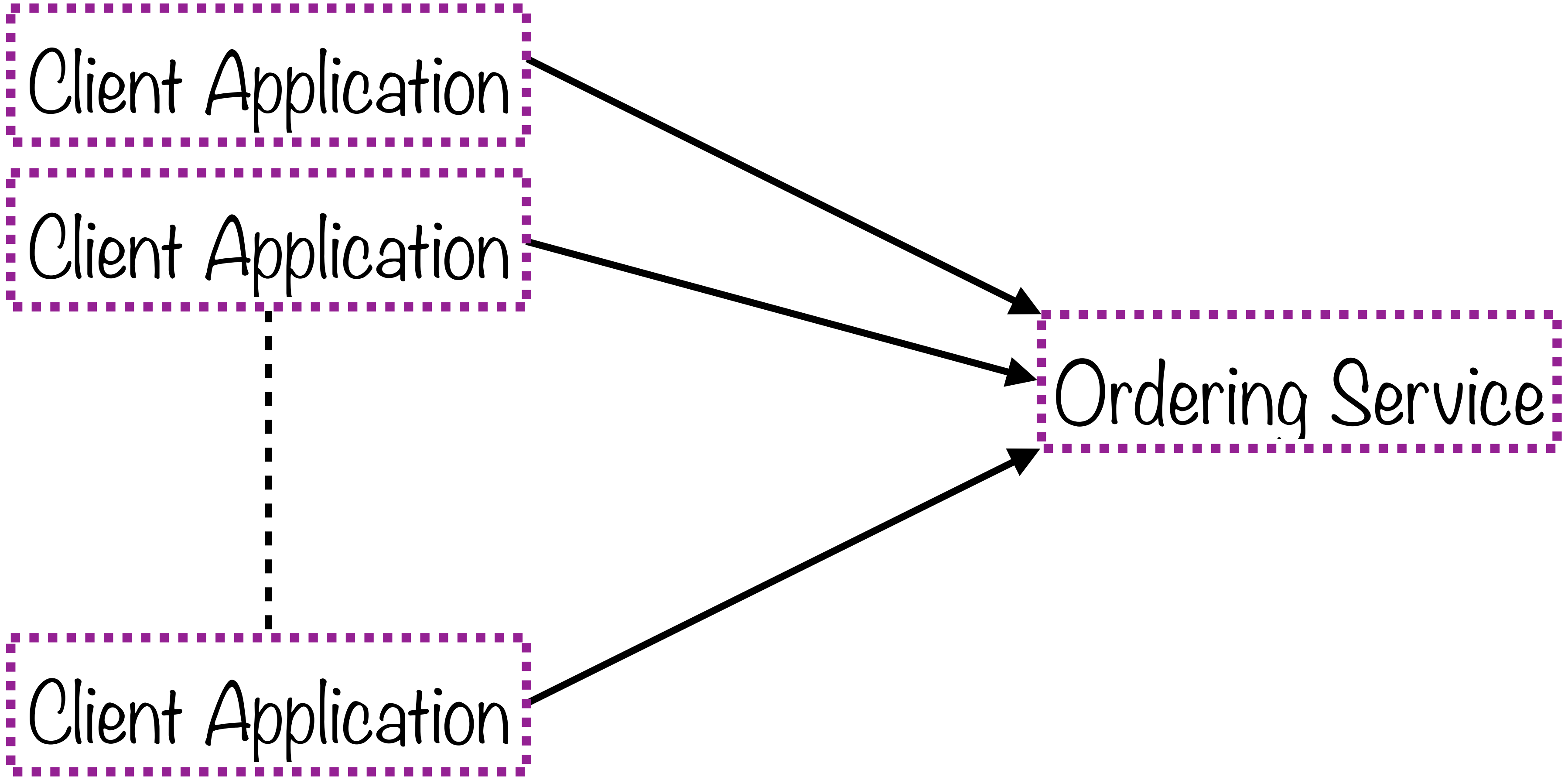
Client application submits the endorsed transaction to the Ordering service

Transaction Ordering



In reality multiple clients will be submitting endorsed transactions to the Ordering service

Transaction Ordering



Ordering service accepts the endorsed transactions and specifies the order in which those transactions will be committed to the ledger

Ordering Mechanisms in Hyperledger Fabric

SOLO

Kafka

SBFT

Agreed-upon ordering mechanisms

Ordering Mechanisms in Hyperledger Fabric

SOLO

Kafka

SBFT

Single ordering node - useful for experimental uses

Ordering Mechanisms in Hyperledger Fabric

SOLO

Kafka

SBFT

Widely used and trusted technology

Kafka

Default ordering service

Production use-cases

Distributed

Fault-tolerant

Unified, high-throughput, low-latency

Ordering Mechanisms in Hyperledger Fabric

SOLO

Kafka

SBFT

SBFT

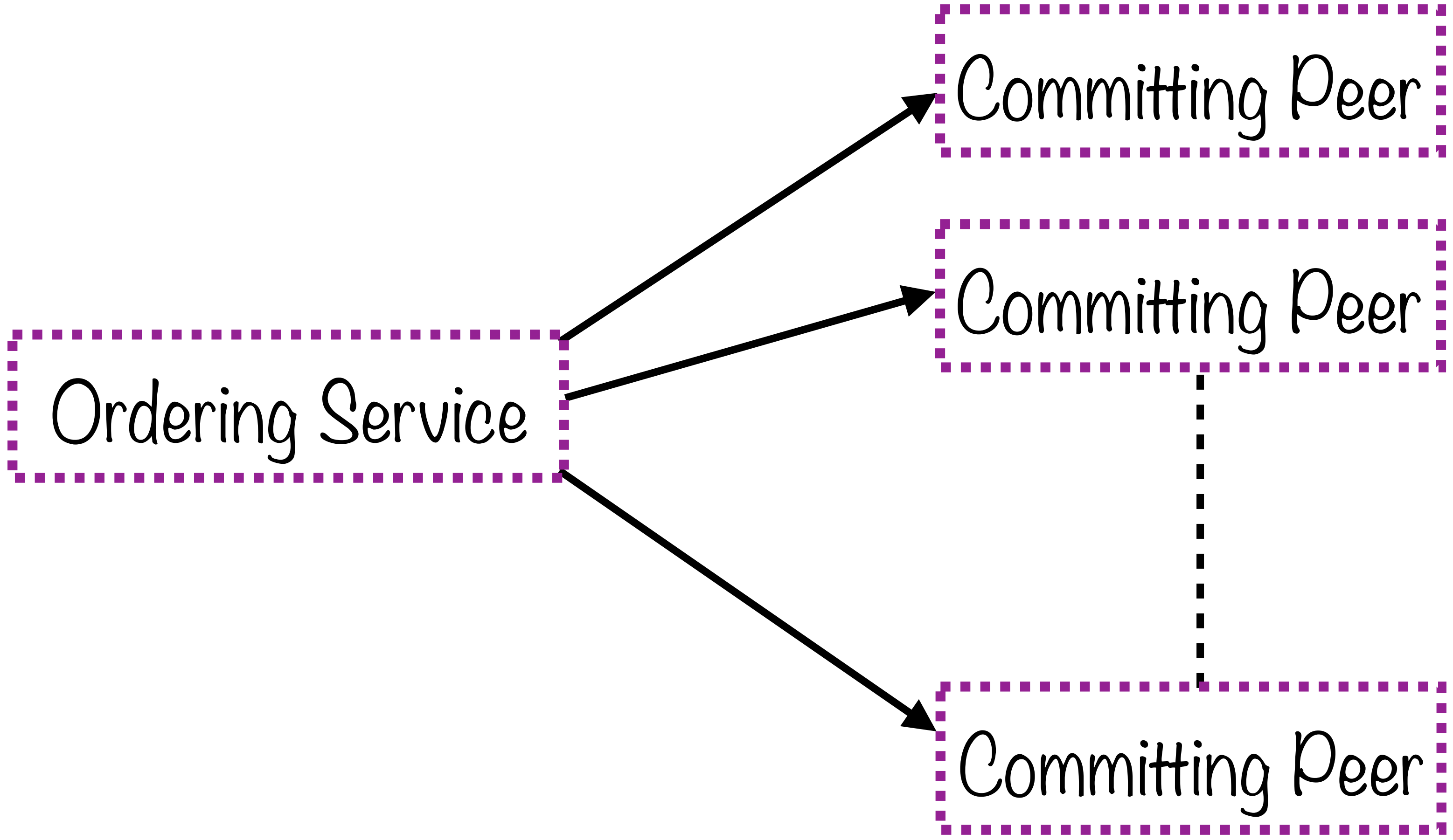
Simplified Byzantine Fault Tolerance

Works even in presence of malicious nodes

Two types of fault tolerance

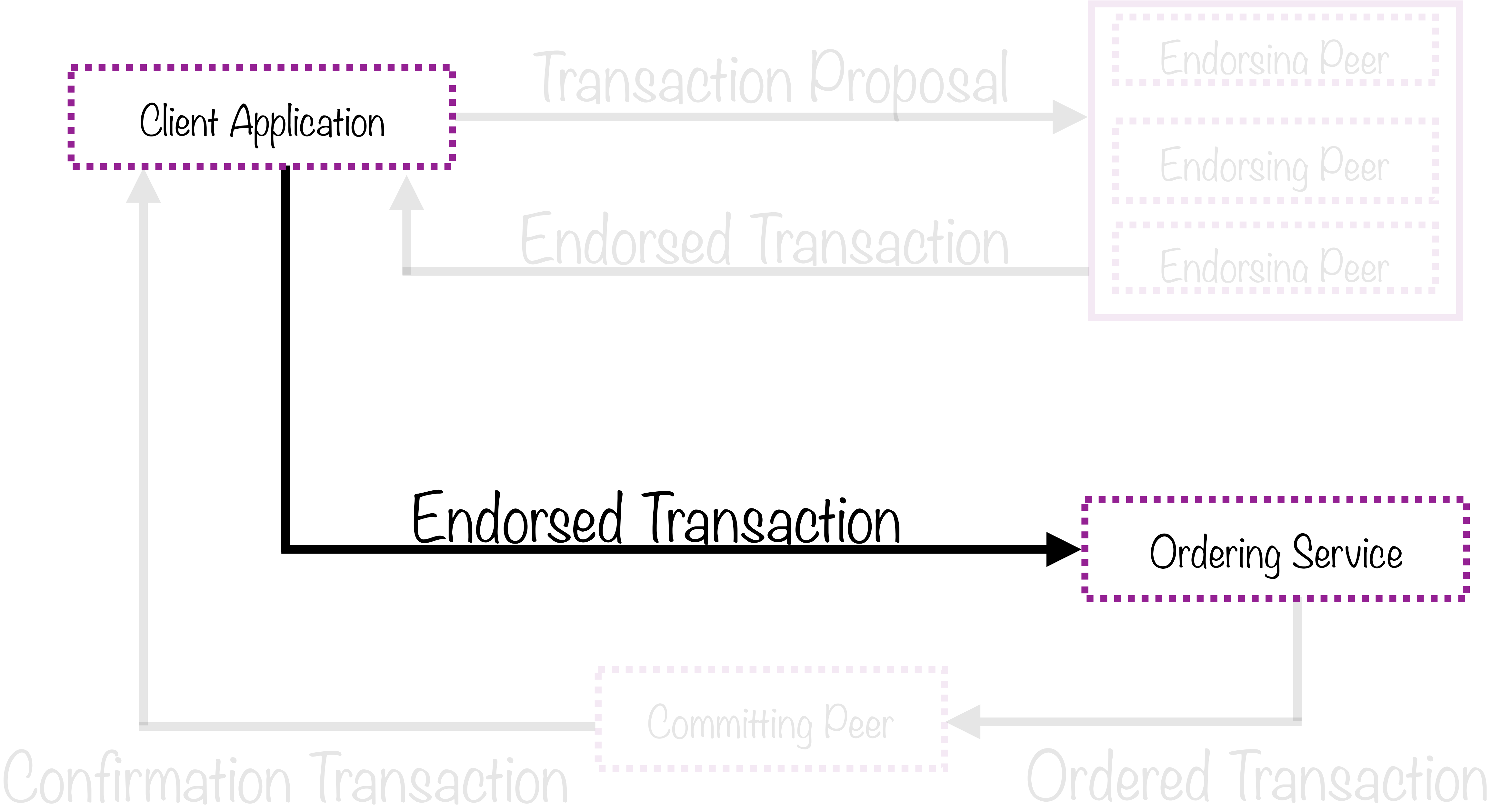
- crash fault tolerance
- byzantine fault tolerance

Ordering Mechanisms in Hyperledger Fabric

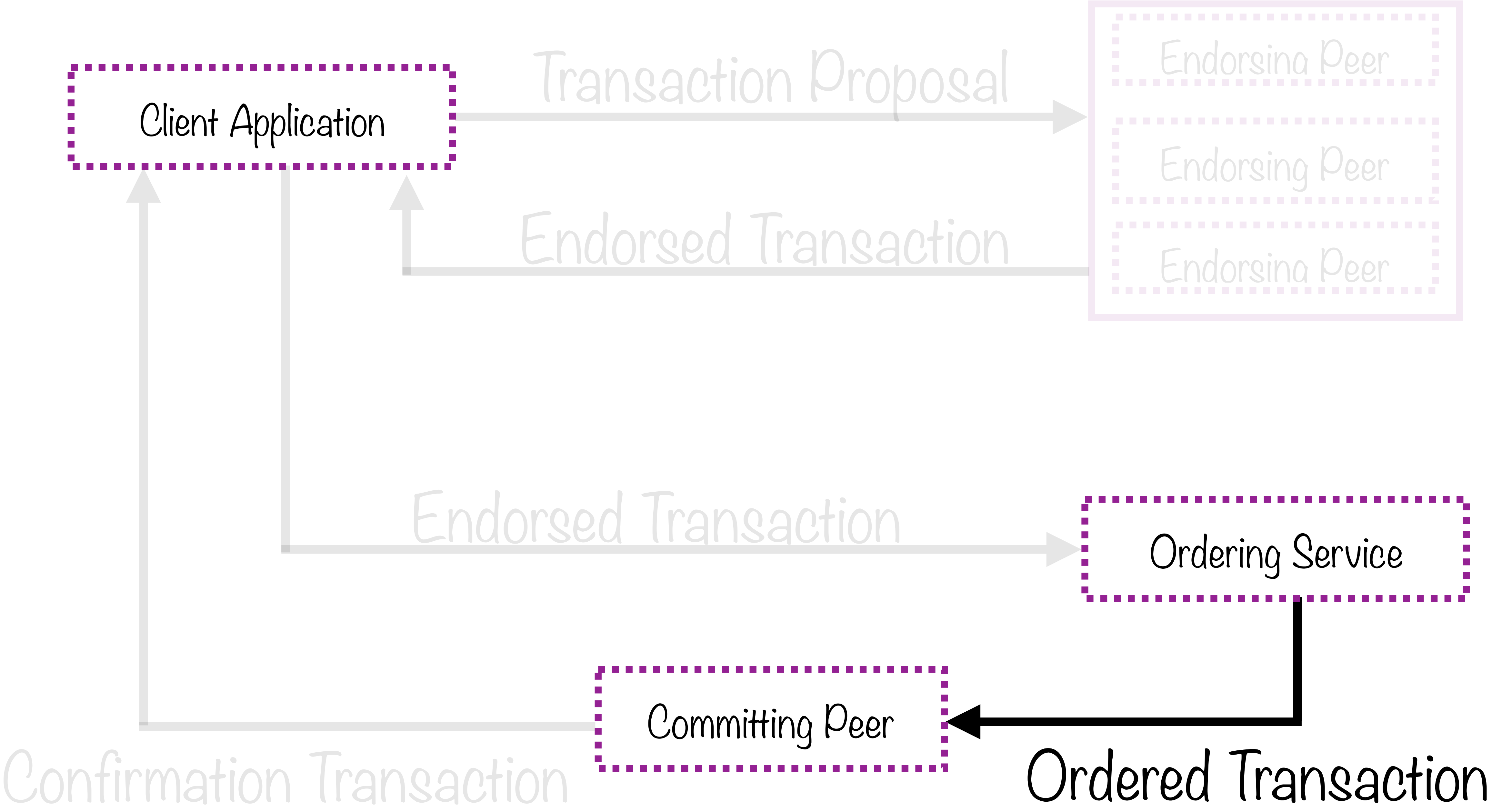


Ordering service orders endorsed transactions; these are then set into a block which is sent to each committing peer

Transaction Flow in Fabric

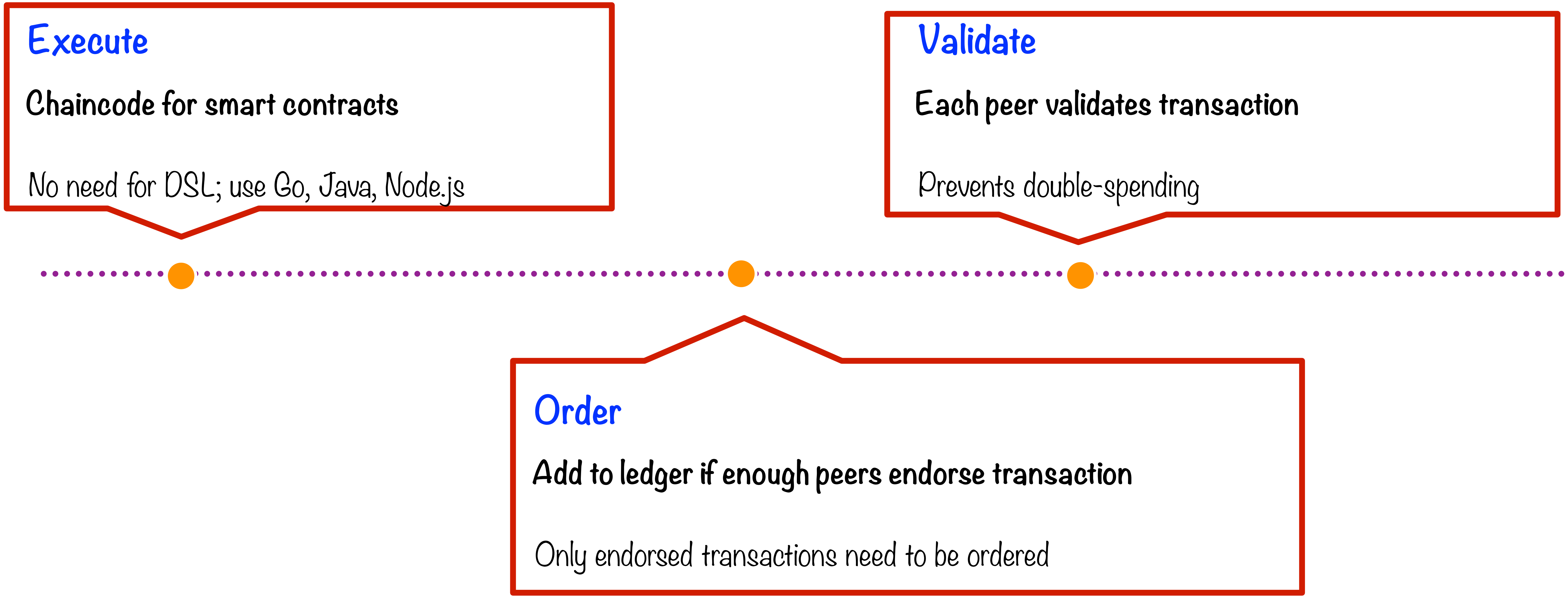


Transaction Flow in Fabric



Hyperledger Transaction Lifecycle

Hyperledger follows a different, more elaborate process



Execute

Chaincode for smart contracts

No need for DSL; use Go, Java, Node.js

Order

Add to ledger if enough peers endorse transaction

Only endorsed transactions need to be ordered

Validate

Each peer validates transaction

Prevents double-spending

Committing Peers

Need to validate (not execute) transaction

Need not have copy of chain code

Committing Peers

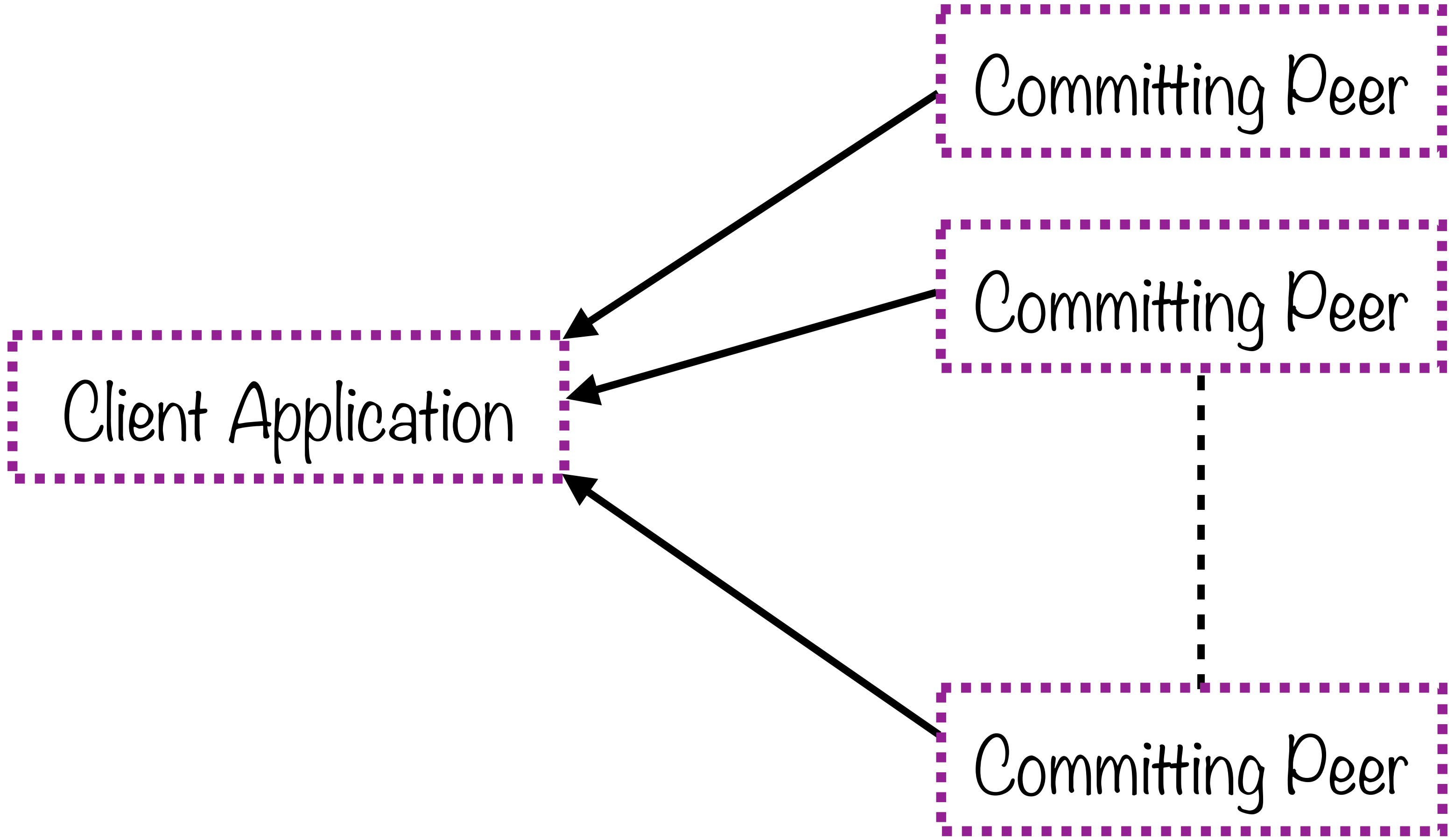
Must process transactions in order

Ensure that endorsement policy satisfied

Check if any previous transaction invalidated this transaction

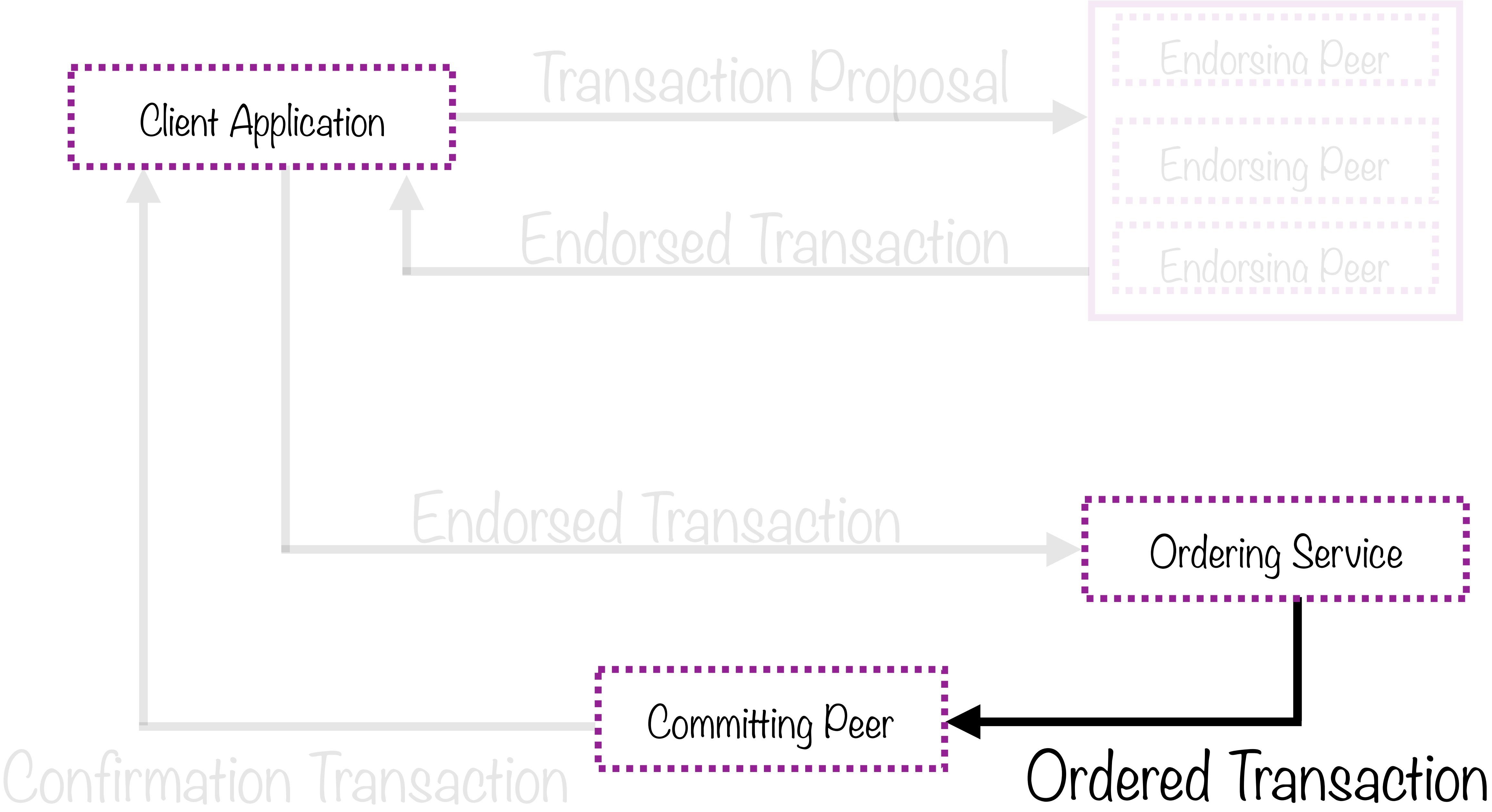
If not, write transaction to ledger

Validation and Commitment

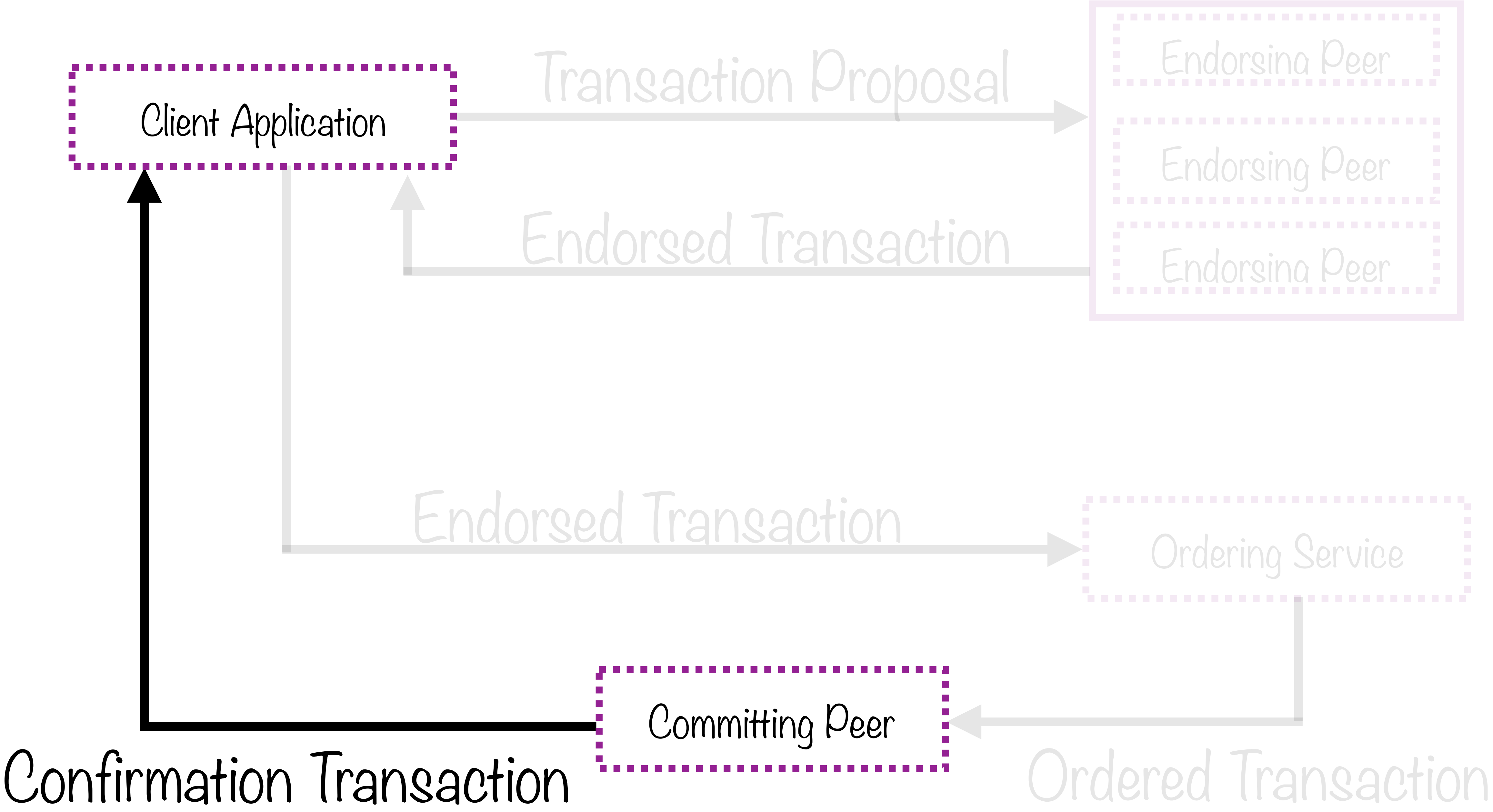


Each committing peer notifies client application of result

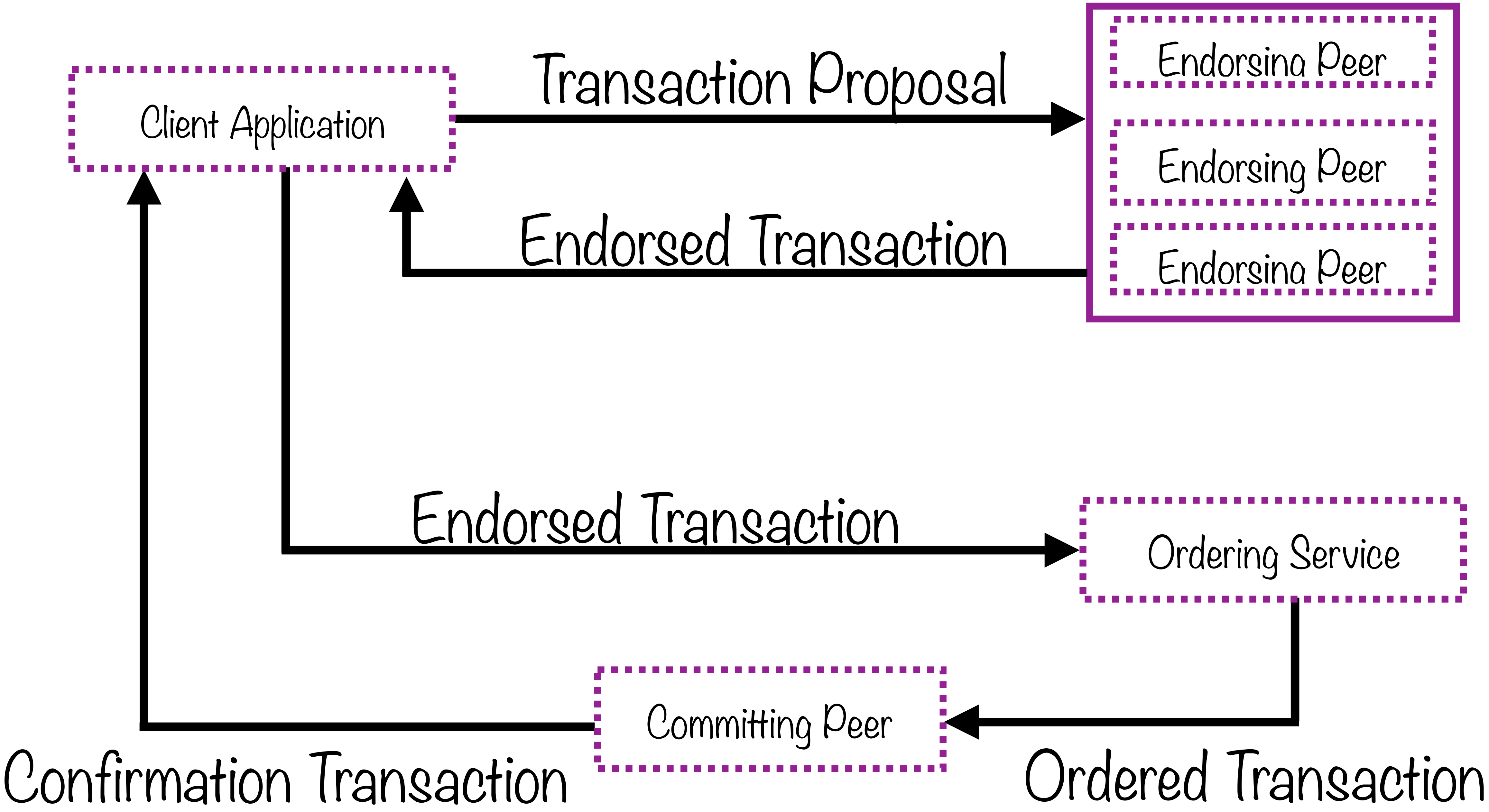
Transaction Flow in Fabric



Transaction Flow in Fabric



Transaction Flow in Fabric

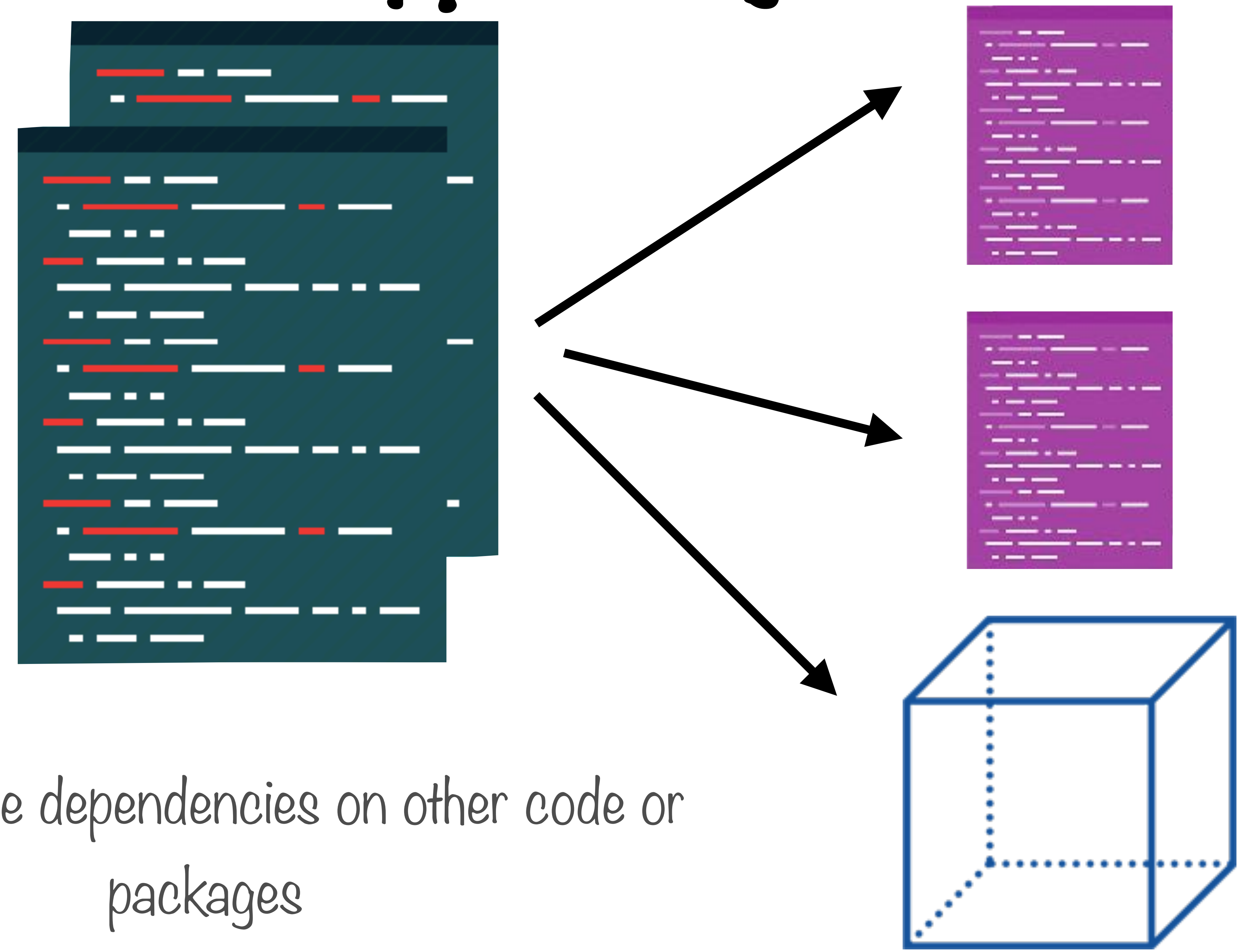


Docker for Hyperledger Fabric on AWS

Docker for Hyperledger Fabric on AWS

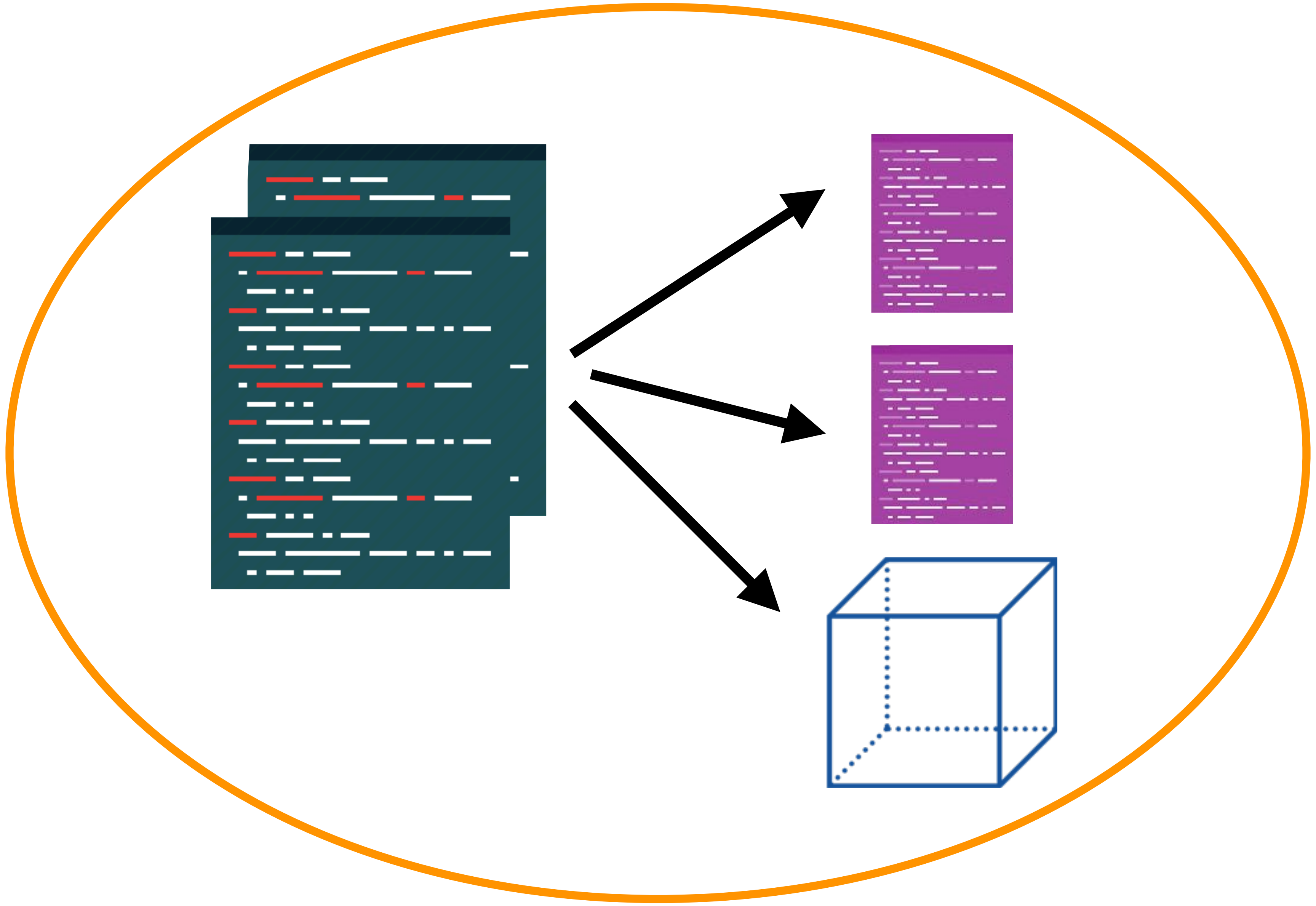
Application code

Docker for Hyperledger Fabric on AWS



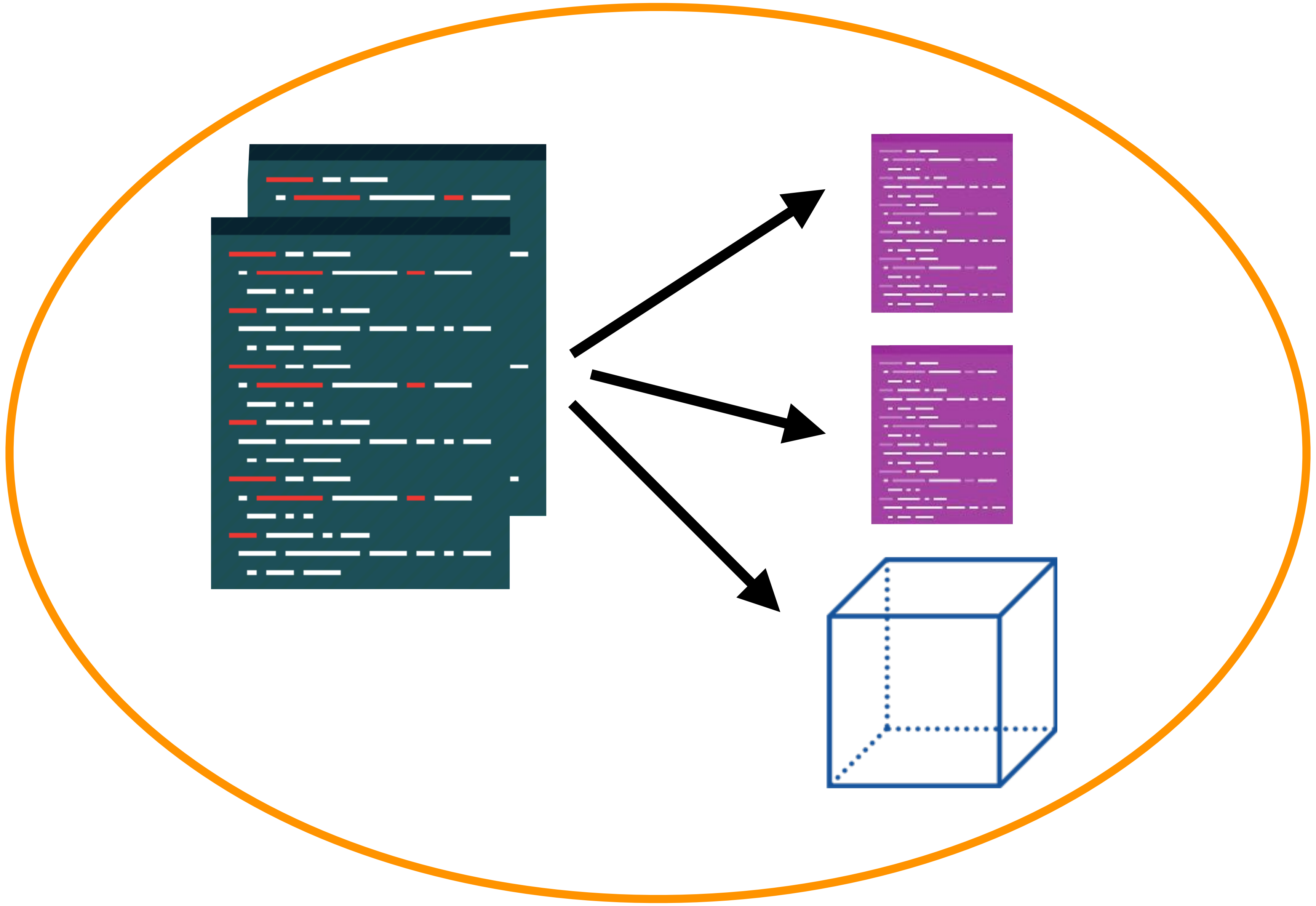
Code will have dependencies on other code or packages

🐳 Docker Containers



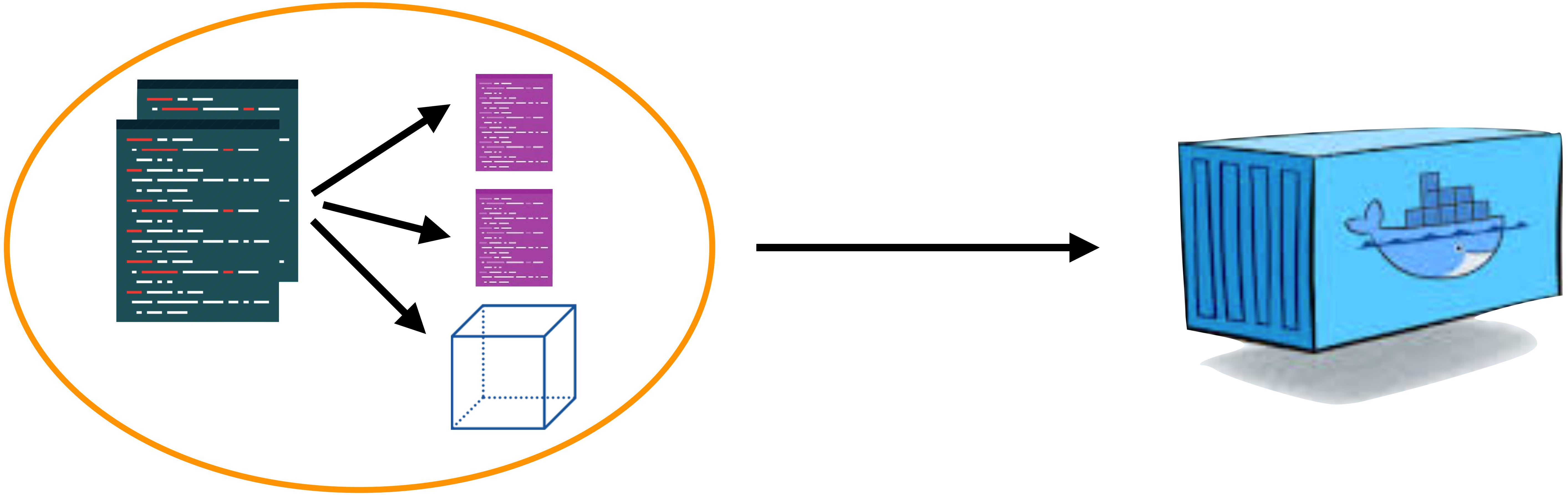
Docker packages all of this arbitrary code into an image

Docker Containers



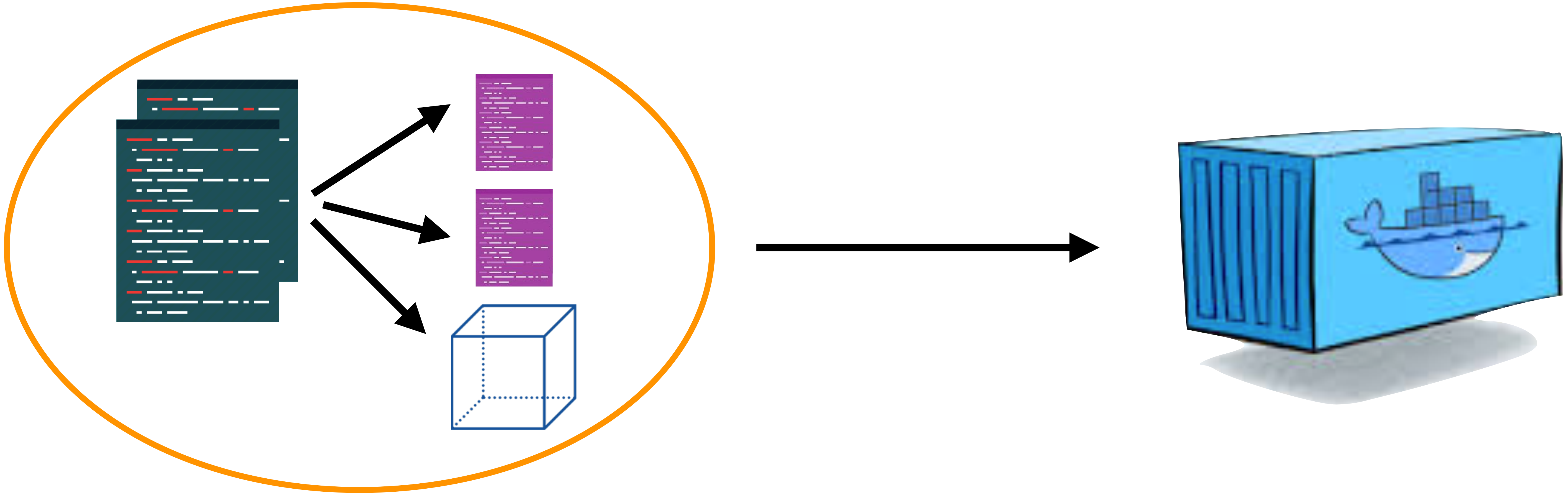
An image is completely self-sufficient

Docker Containers



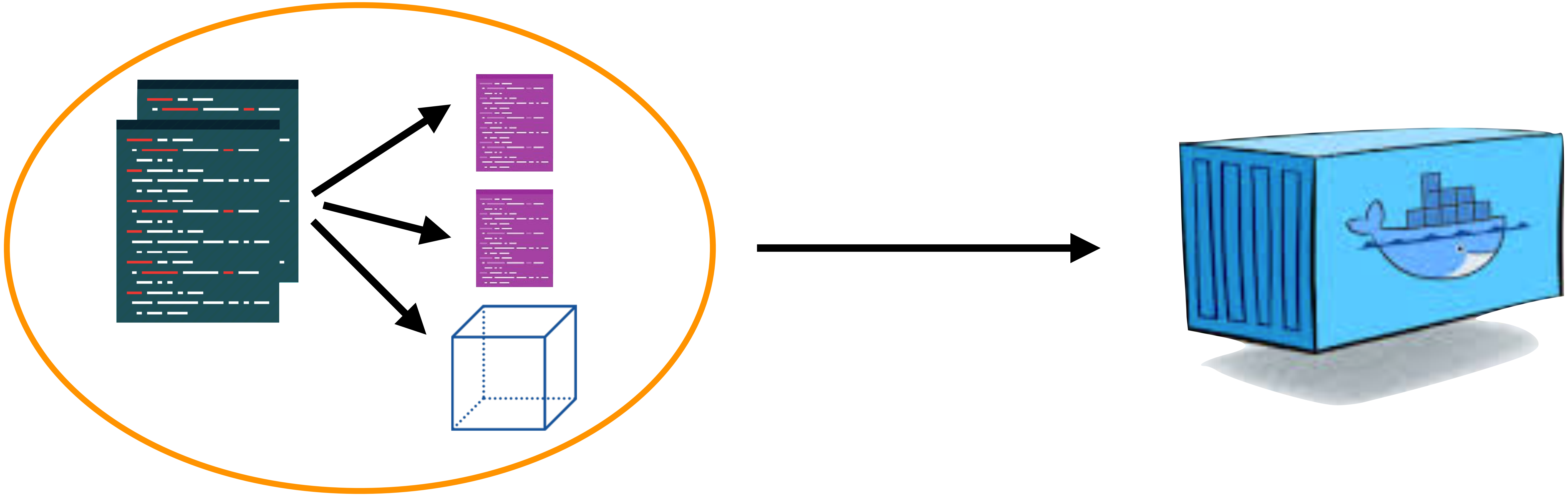
An image can be used to run a Docker container

Docker Containers



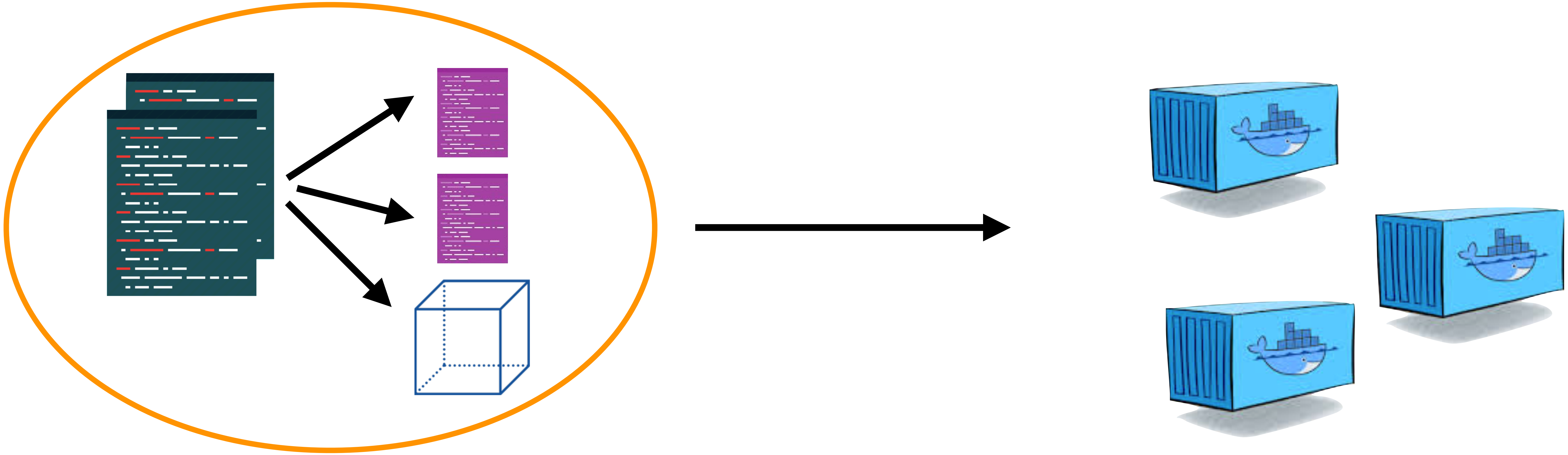
Containers are a fully self contained environment which executes code

Docker Containers

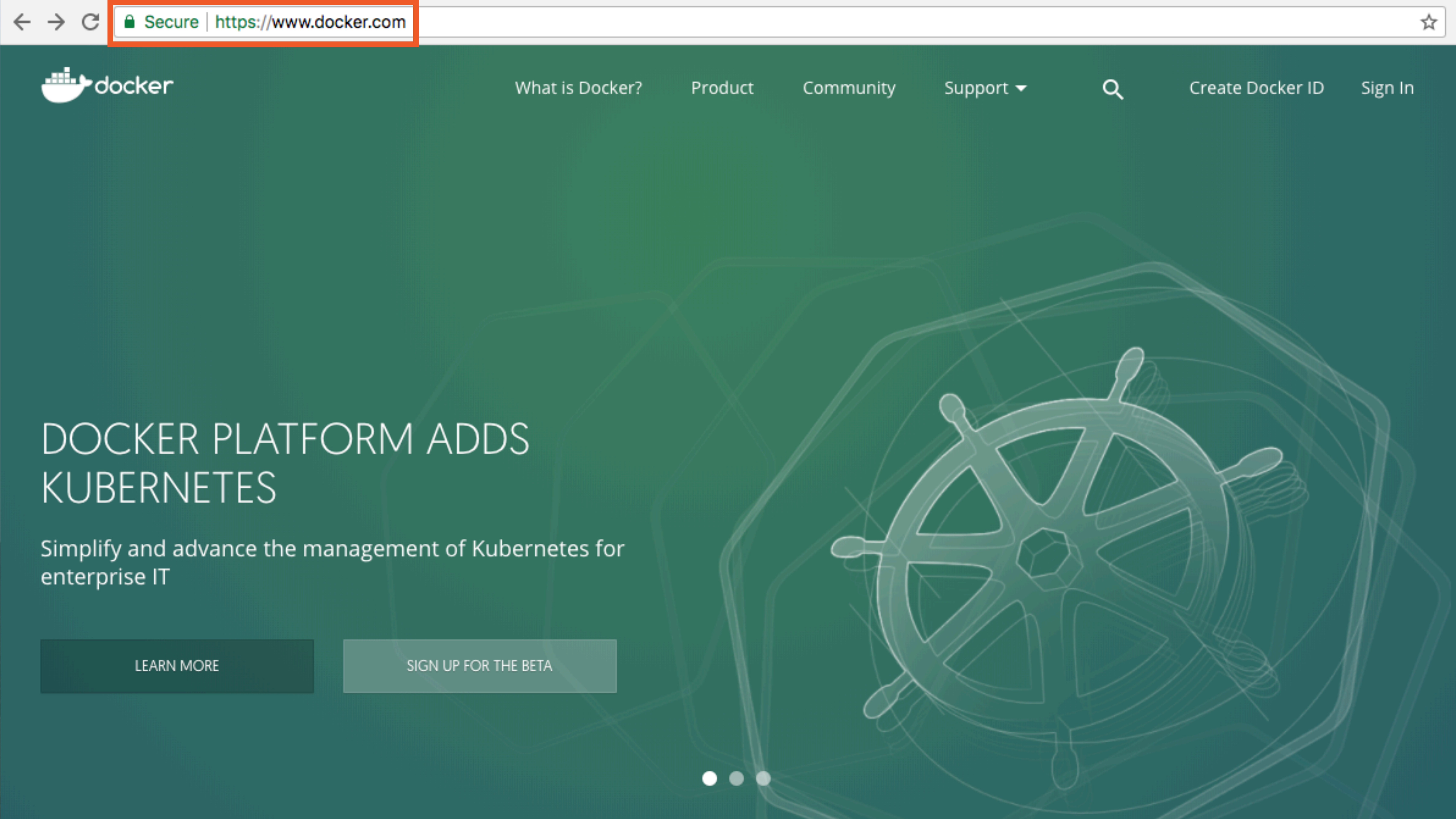


Containers do not contain the OS and your code is abstracted away from the machine

Docker Containers



You can create as many containers as you want from the same image

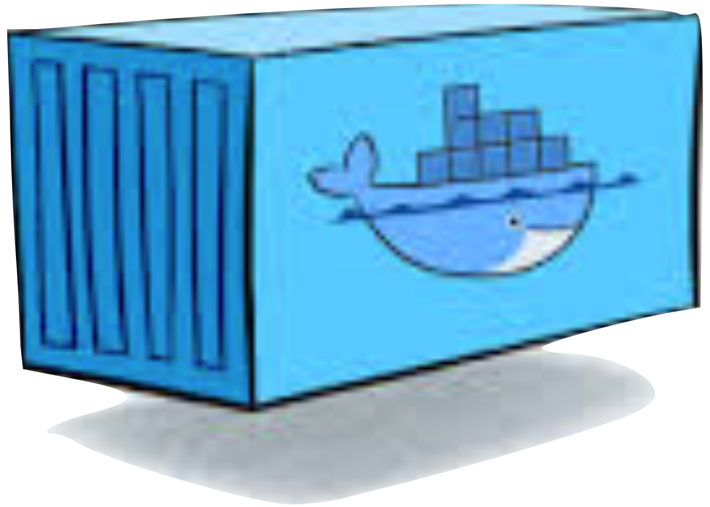


DOCKER PLATFORM ADDS KUBERNETES

Simplify and advance the management of Kubernetes for enterprise IT

Use Docker containers to use Hyperledger Fabric on AWS

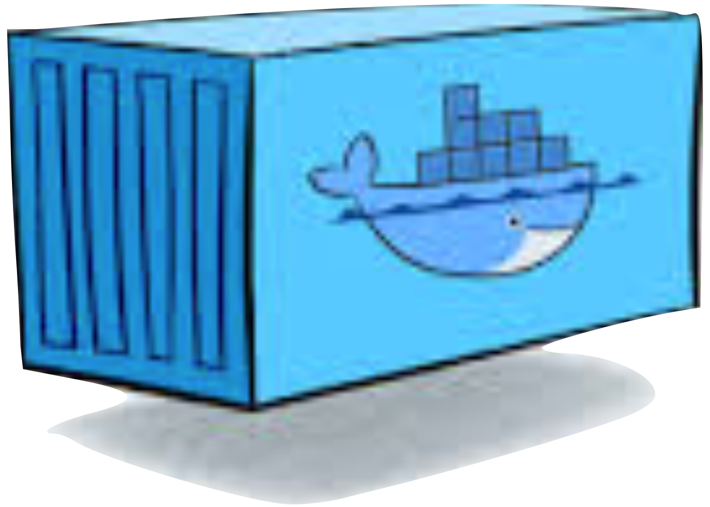
Docker



Create package with

- application code
- dependencies
- user space of Linux OS where app was developed

Docker Compose

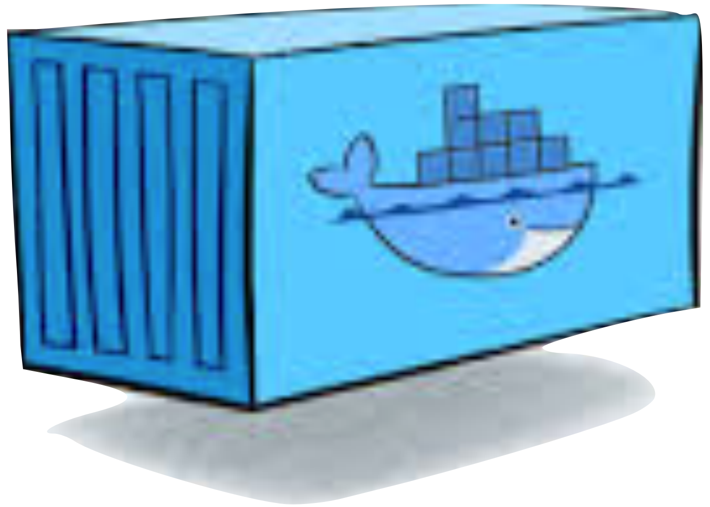


Create multi-container app

Specification in YAML file

- Yet Another Markup Language

Docker Compose for Hyperledger Fabric on AWS



AWS provides YAML file

Creates required containers

Demo

Working with Hyperledger Fabric

Building a Hyperledger Fabric Network

Hyperledger

Umbrella project of open source blockchains; started in December 2015 by the Linux Foundation

Umbrella of Frameworks

Hyperledger Fabric

Hyperledger Sawtooth

Hyperledger Iroha

Hyperledger Burro

Hyperledger Indy

Umbrella of Tools

Hyperledger Cello

Hyperledger Composer

Hyperledger Caliper

Hyperledger Explorer

Hyperledger Quilt

Hyperledger Cello

Blockchain-as-a-Service

Dashboard for ease-of-use

Users can create, search deploy, terminate

Hyperledger Cello

Initially contributed by IBM

Now also Intel, Huawei, Soramitsu

Hyperledger Cello

Cluster with master and workers

Compute resources on cloud or locally

Not fully stable/operational yet

Hyperledger Explorer

Blockchain visualization

Explore blocks

Query transaction information

Hyperledger Explorer

Query and explore

- Blocks
- Transactions and metadata
- Smart contracts
- Nodes and network information

Hyperledger Composer

Hyperledger Composer

Suite of tools for building blockchain networks and creating smart contracts from a web-based UI

Hyperledger Composer

Higher level abstractions

Built on top of Hyperledger Fabric

Templates for common use-cases

Hyperledger Composer

Modeling resources

Permissioning resources

Processing transactions

Hyperledger Composer

Modeling resources

- CTO: OO modeling language
- CTO: Name, not acronym
- includes primitive data types

Hyperledger Identities

Define assets, identities in CTO file

CTO file : Class definitions

Now can instantiate classes

I.e. can create objects

E.g. can create identity

Then can permission identities, resources

Hyperledger Identities

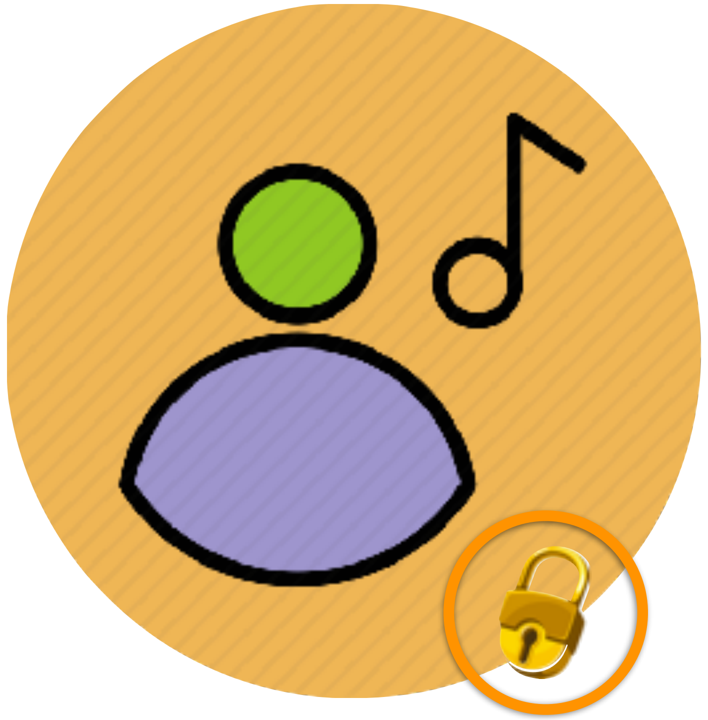
Participant instances

Participant registry

Identities must be issued to instances

Can do from Composer UI or API

Hyperledger Composer



Permissioning resources

- simple language for ACLs

Processing transactions

- Javascript for transaction logic

Demo

Working with Hyperledger Composer

Working with Hyperledger Iroha

Hyperledger

Umbrella project of open source blockchains; started in December 2015 by the Linux Foundation

Umbrella of Frameworks

Hyperledger Fabric

Hyperledger Sawtooth

Hyperledger Iroha

Hyperledger Burro

Hyperledger Indy

Umbrella of Frameworks

- Hyperledger Fabric
- Hyperledger Sawtooth
- Hyperledger Iroha
- Hyperledger Burro
- Hyperledger Indy

Hyperledger Iroha

Simple, fast permissioned blockchain framework with Byzantine fault-tolerant consensus

Hyperledger Iroha

Small set of fast commands

Blocks stored in files

Ledger state stored in PostgreSQL

No associated cryptocurrency

Hyperledger Iroha

Implemented in C++

Designed for ease of mobile development

Libraries available for

- Android (Java)
- iOS
- Javascript

Hyperledger Iroha

Assets

Accounts

Domains

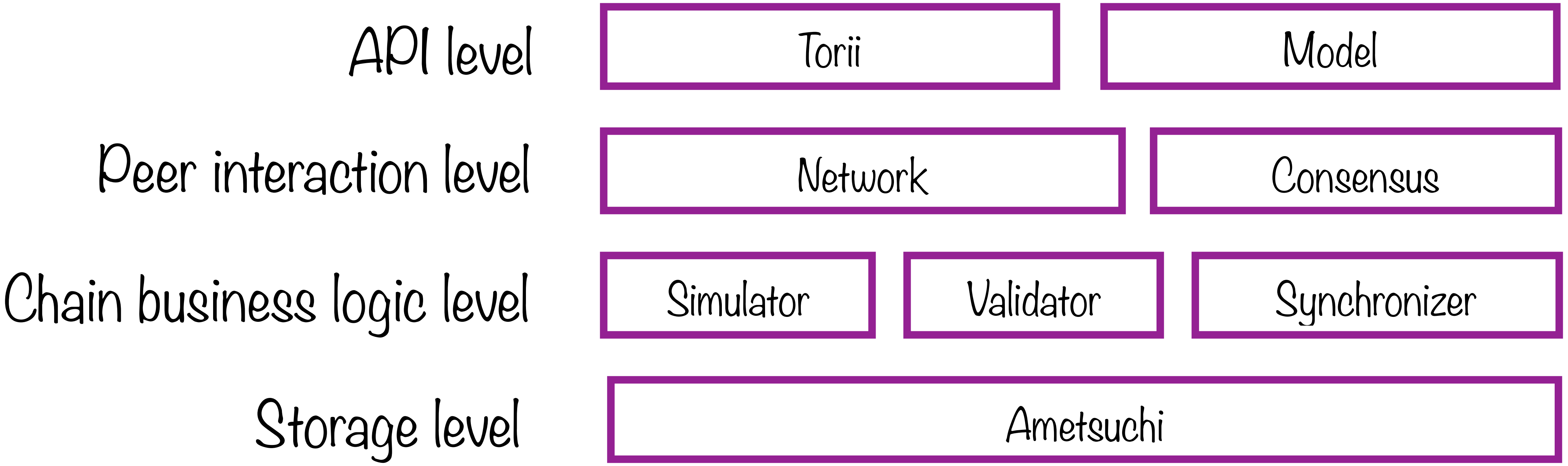
Hyperledger Iroha

Assets: Any countable commodity

Accounts: Iroha entities that can perform actions

Domains: Aggregations of assets and accounts

Architecture of Iroha



API level: Torii, Model

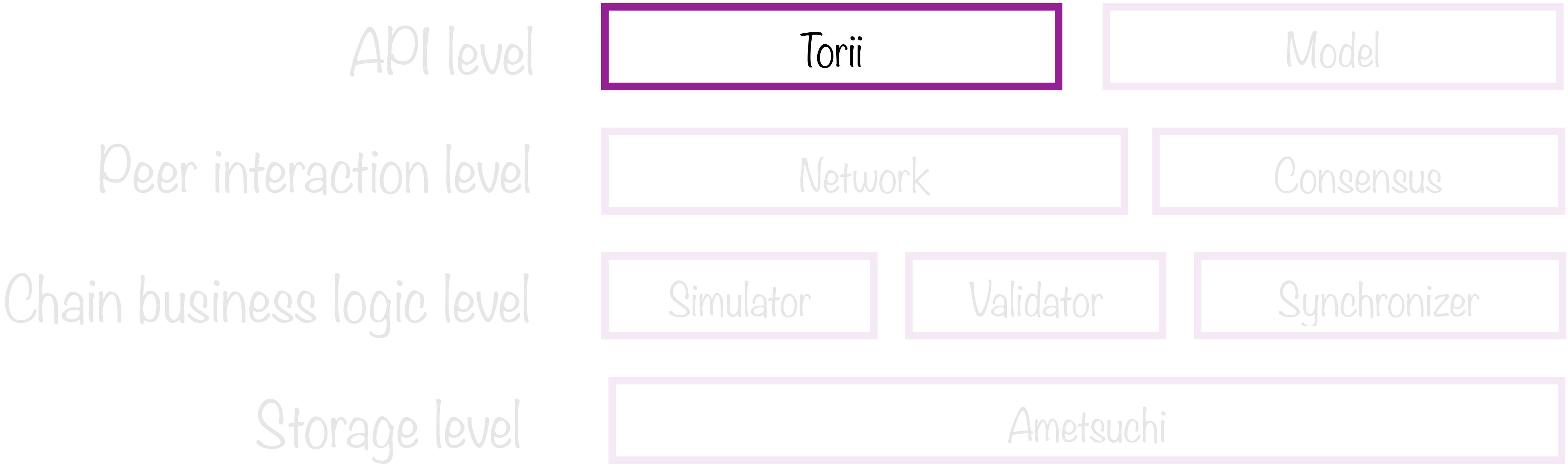
Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

Four-layered architecture - some noteworthy bits

Architecture of Iroha



Client interface

Torii

Client interface

Used by clients to connect with peers

Simple grpc server

Use for both

- transactions
- queries

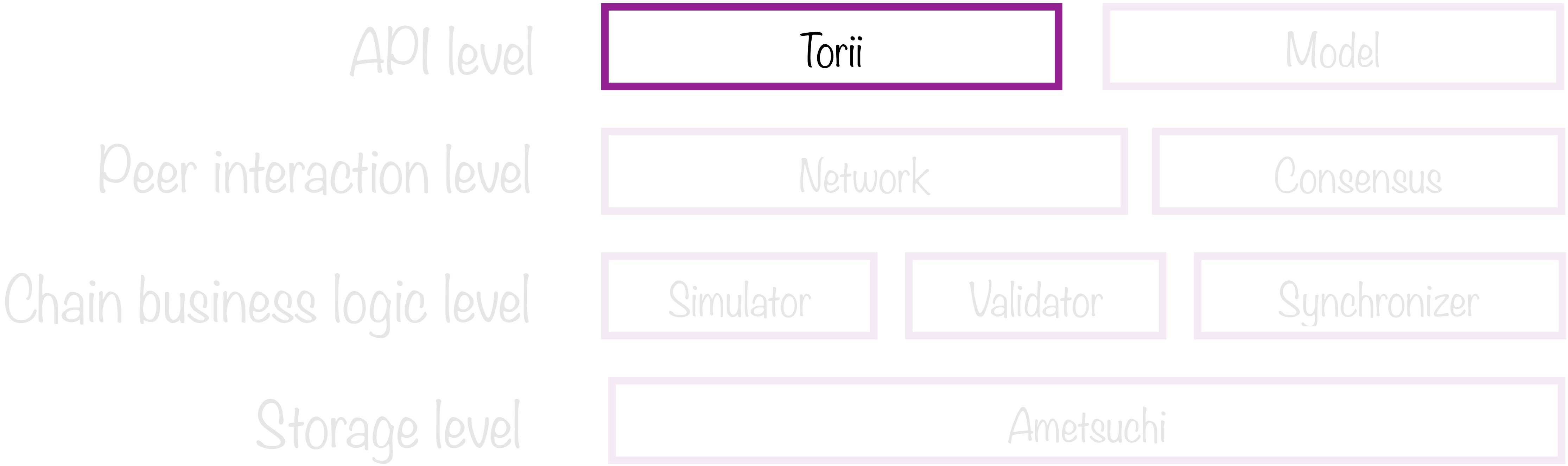
Clients

Clients have permission-based access to assets and accounts

Clients can query data (read)

Submit transactions to change system state (read-write)

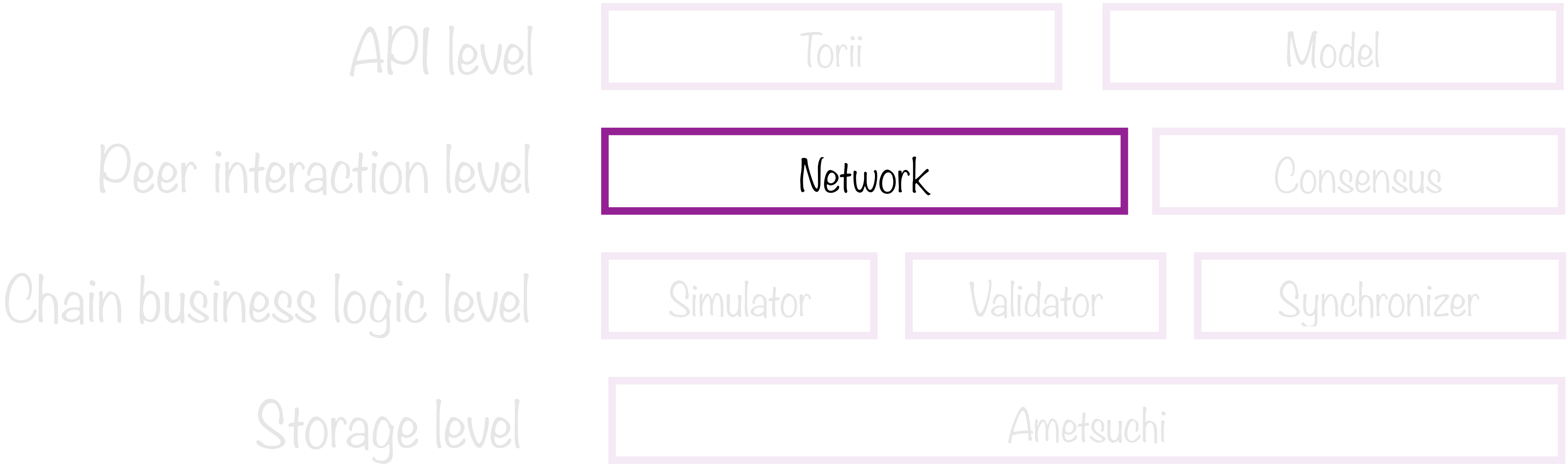
Architecture of Iroha



API level	Torii	Model	
Peer interaction level	Network	Consensus	
Chain business logic level	Simulator	Validator	Synchronizer
Storage level	Ametsuchi		

Four-layered architecture - some noteworthy bits

Architecture of Iroha



API level: Torii, Model

Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

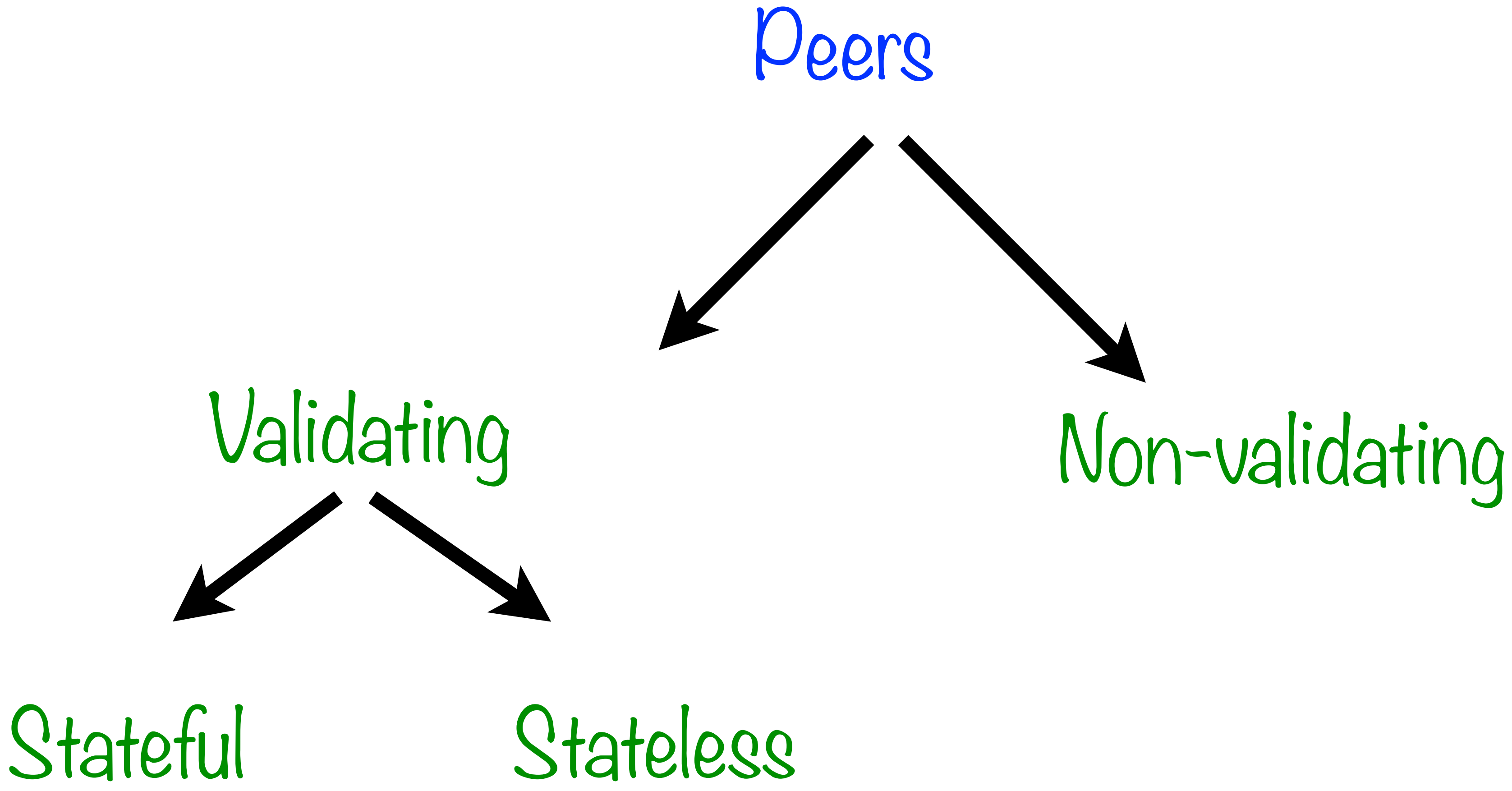
Interaction between peers on the network

Peers

Peer is a single entity in network

Possesses address, identity and trust (key pair)

Peers maintain their own copy of the shared ledger



Peers

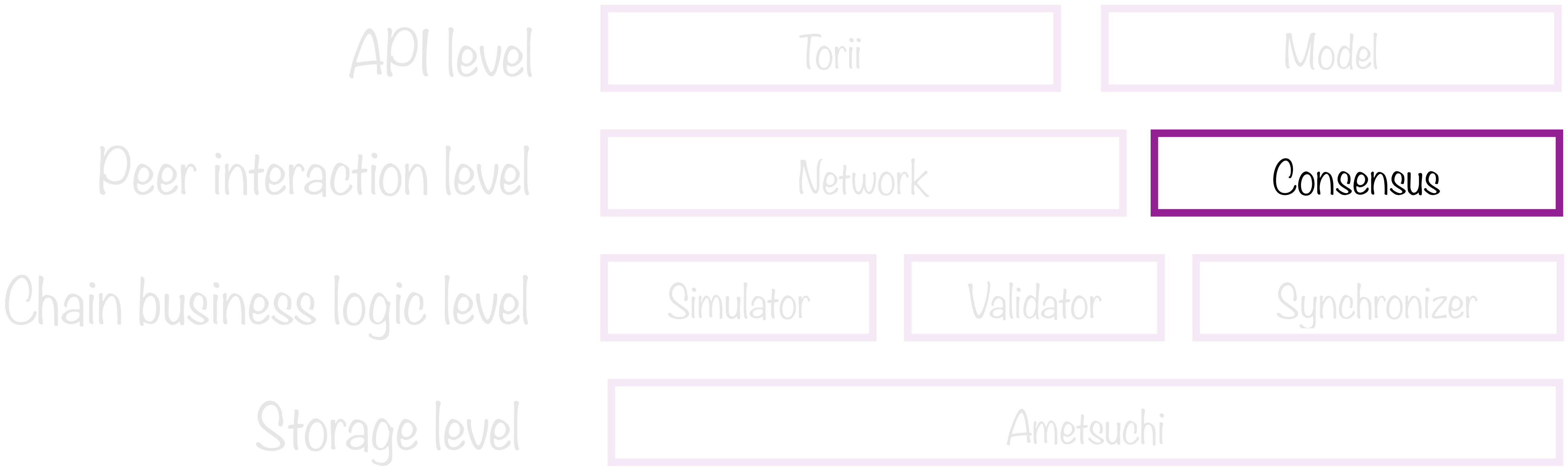
Validating

Non-validating

Stateful

Stateless

Architecture of Iroha



API level: Torii, Model

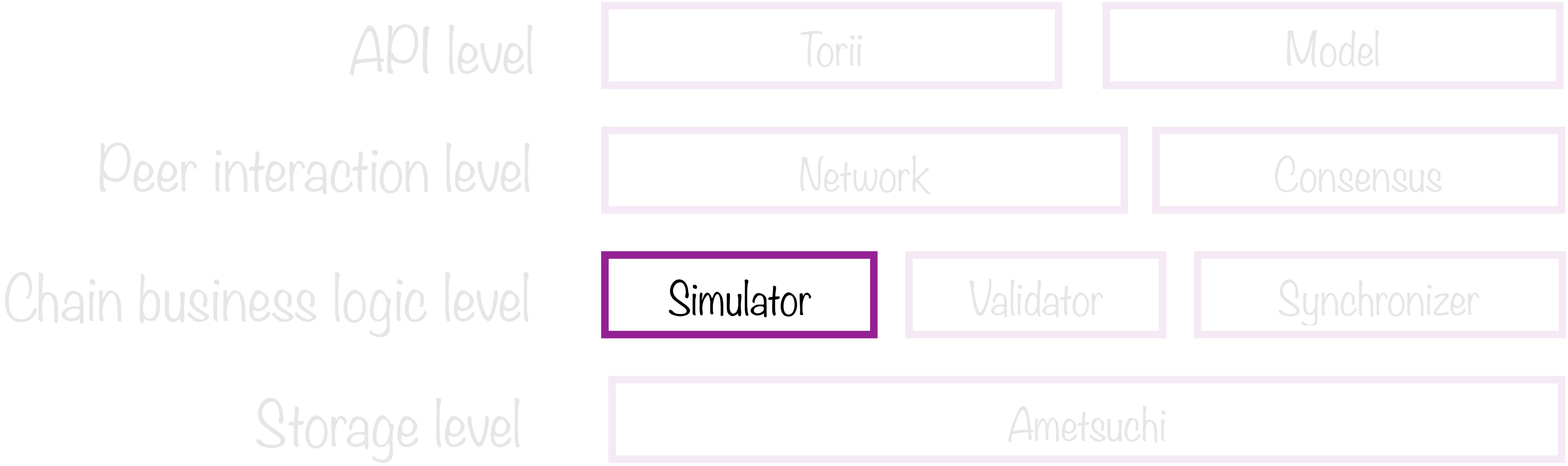
Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

Iroha relies on Byzantine Fault-tolerant consensus algorithm - also called Yet Another Consensus (YAC) algorithm

Architecture of Iroha



API level: Torii, Model

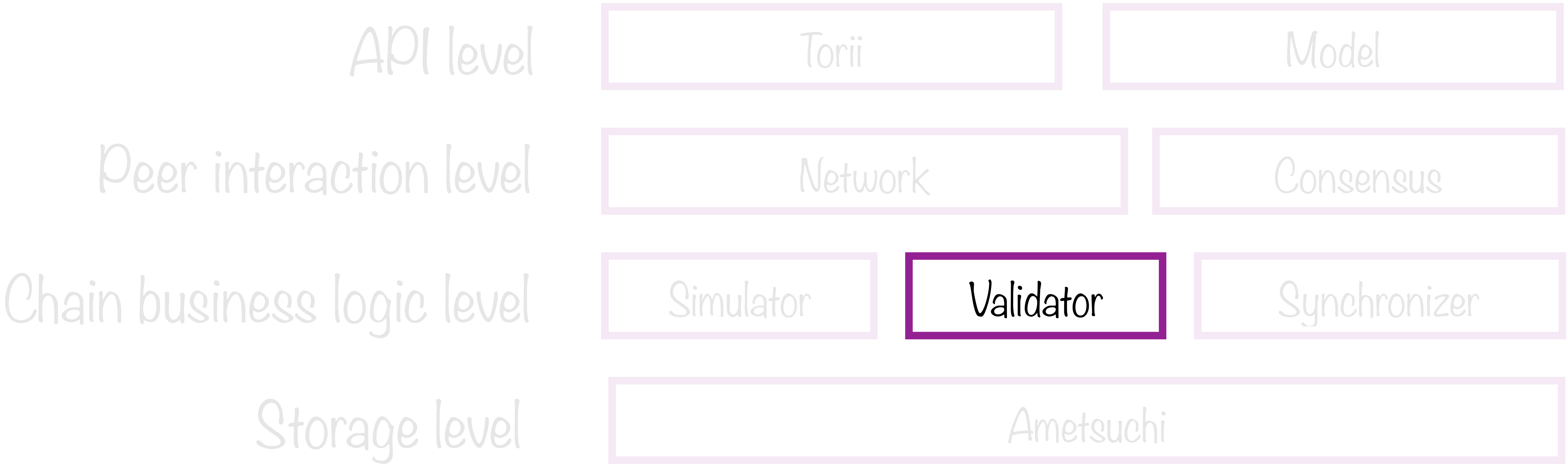
Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

During transaction validation, peers use the simulator to “play out” transactions to ensure the results are sound

Architecture of Iroha



API level: Torii, Model

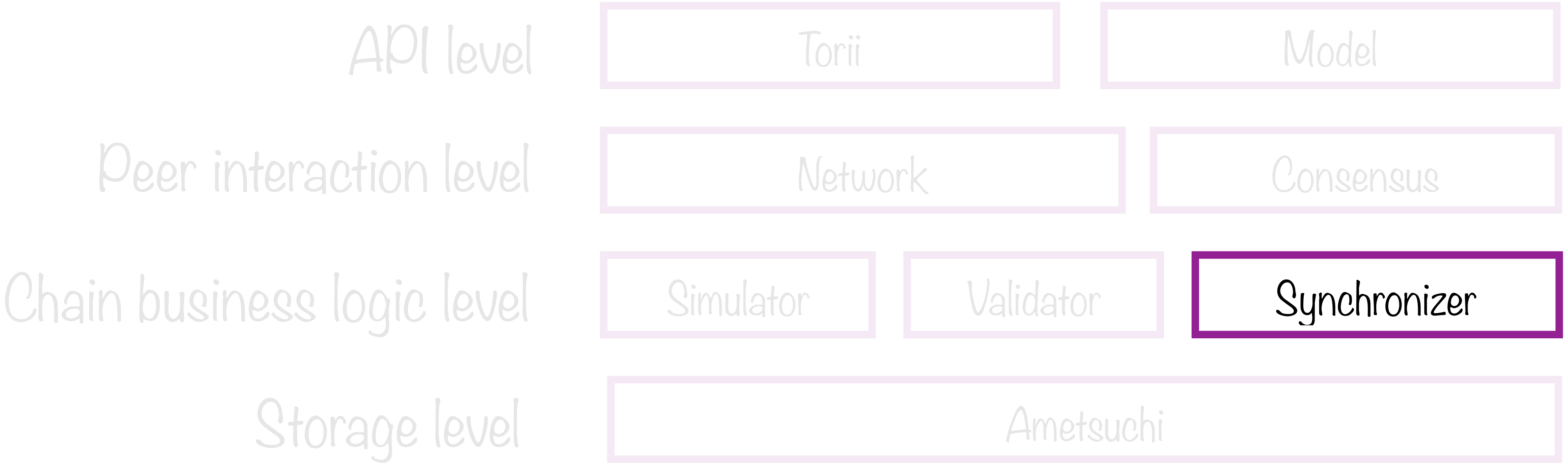
Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

Perform checks of transaction or query validity

Architecture of Iroha



API level: Torii, Model

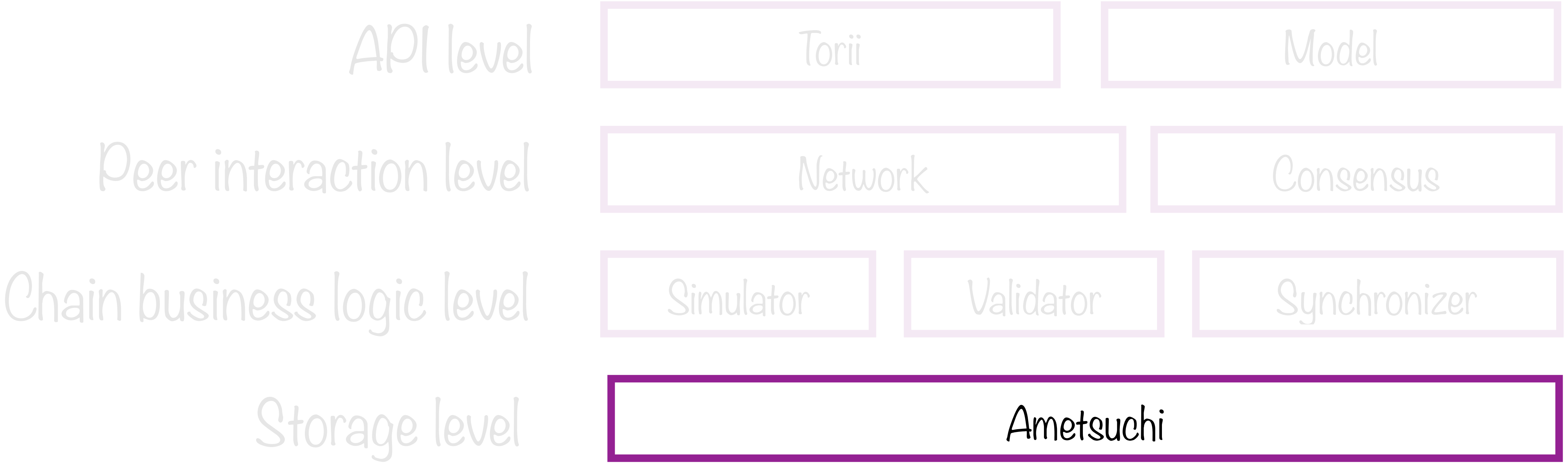
Peer interaction level: Network, Consensus

Chain business logic level: Simulator, Validator, Synchronizer

Storage level: Ametsuchi

Get newly added peers in-sync with system state

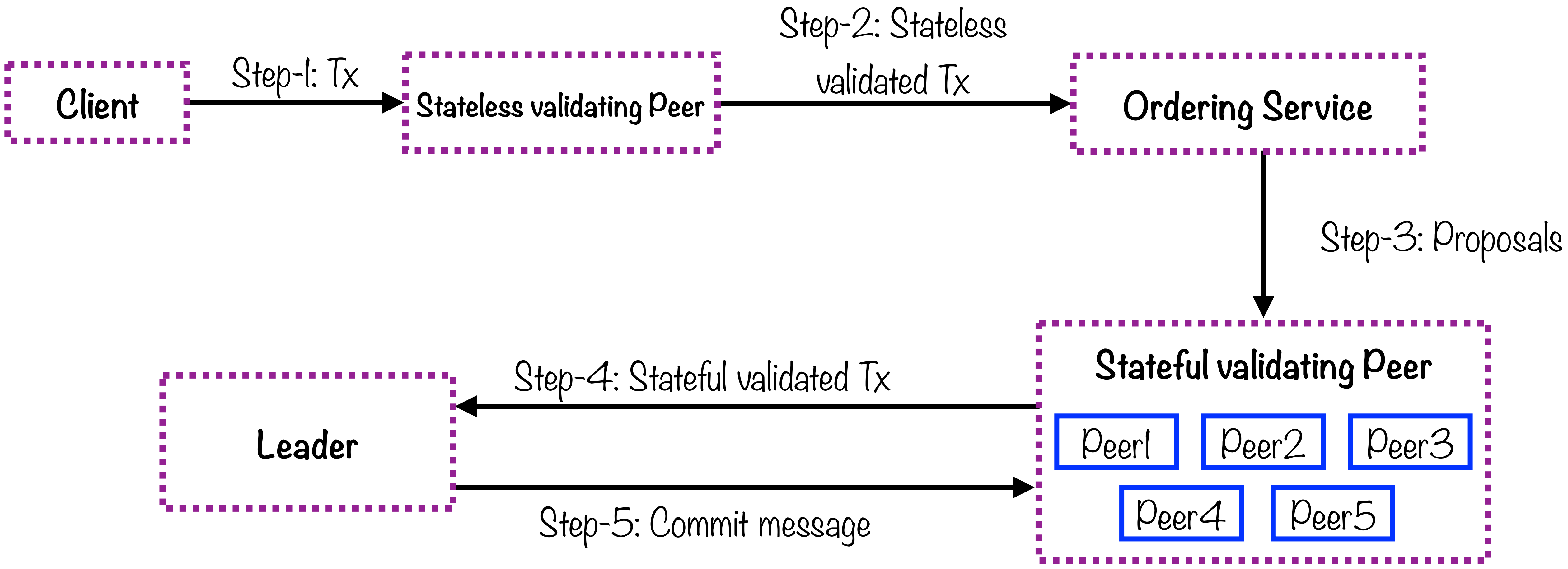
Architecture of Iroha



Block store and block index

Transaction Verification in Iroha

Transaction Flow in Hyperledger Iroha



Client

Step-1: Tx

Stateless validating Peer

Step-2: Stateless validated Tx

Ordering Service

Step-3: Proposals

Stateful validating Peer

Peer1

Peer2

Peer3

Peer4

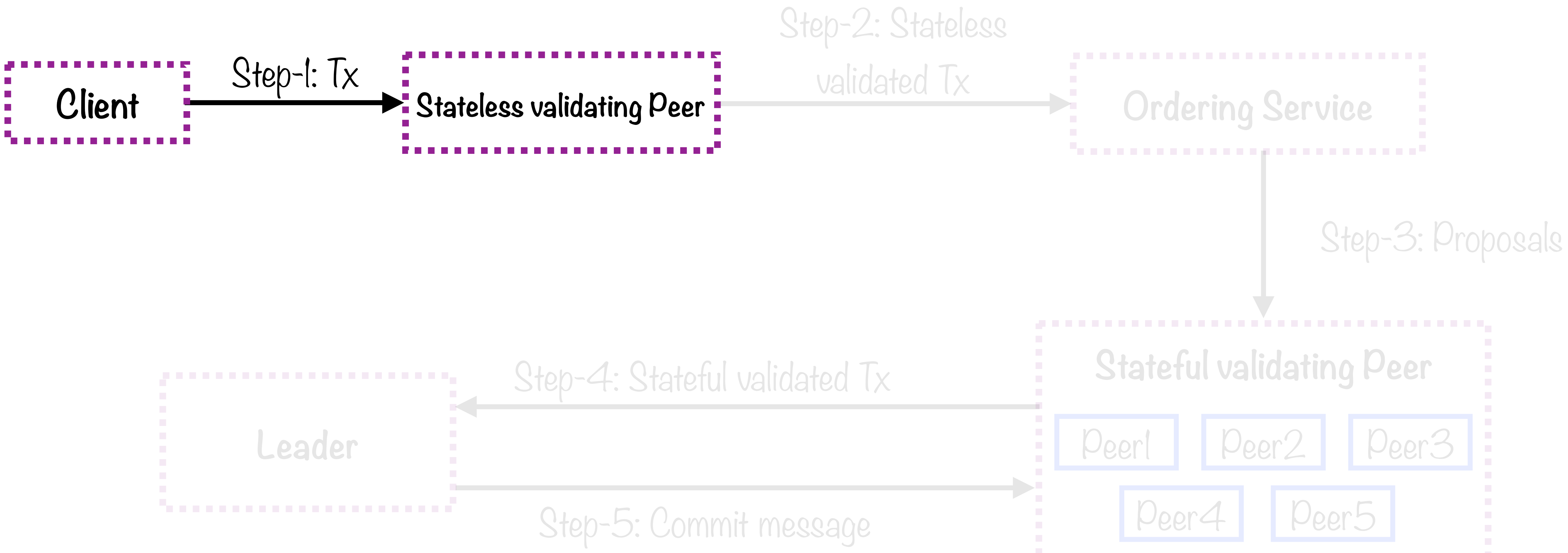
Peer5

Step-4: Stateful validated Tx

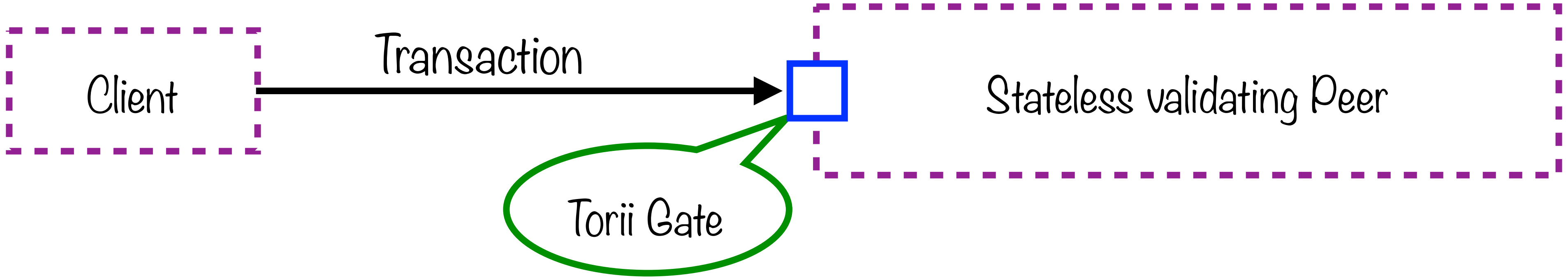
Leader

Step-5: Commit message

Transaction Flow in Hyperledger Iroha

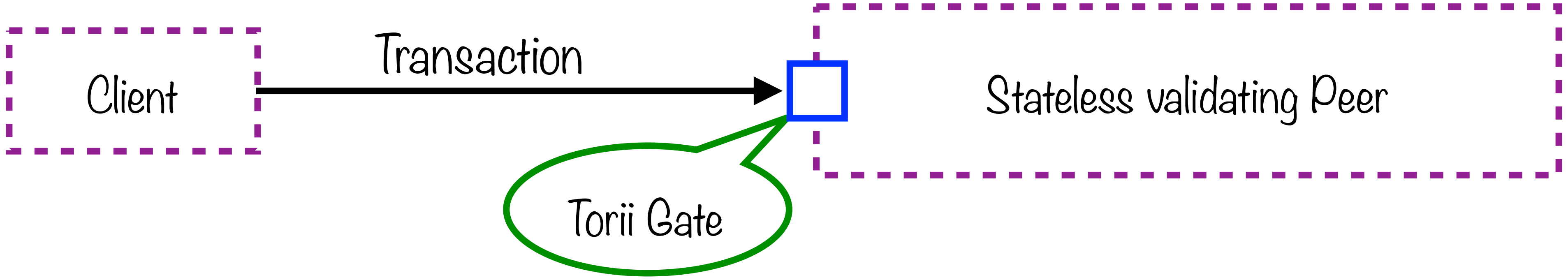


Transaction Flow in Hyperledger Iroha



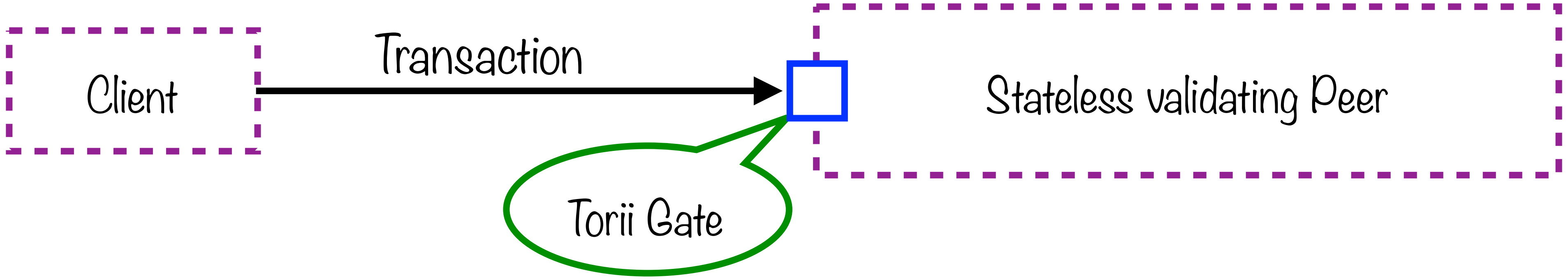
Client creates and sends a transaction to the Torii gate

Transaction Flow in Hyperledger Iroha



Torii gate routes the transaction to a stateless validating peer

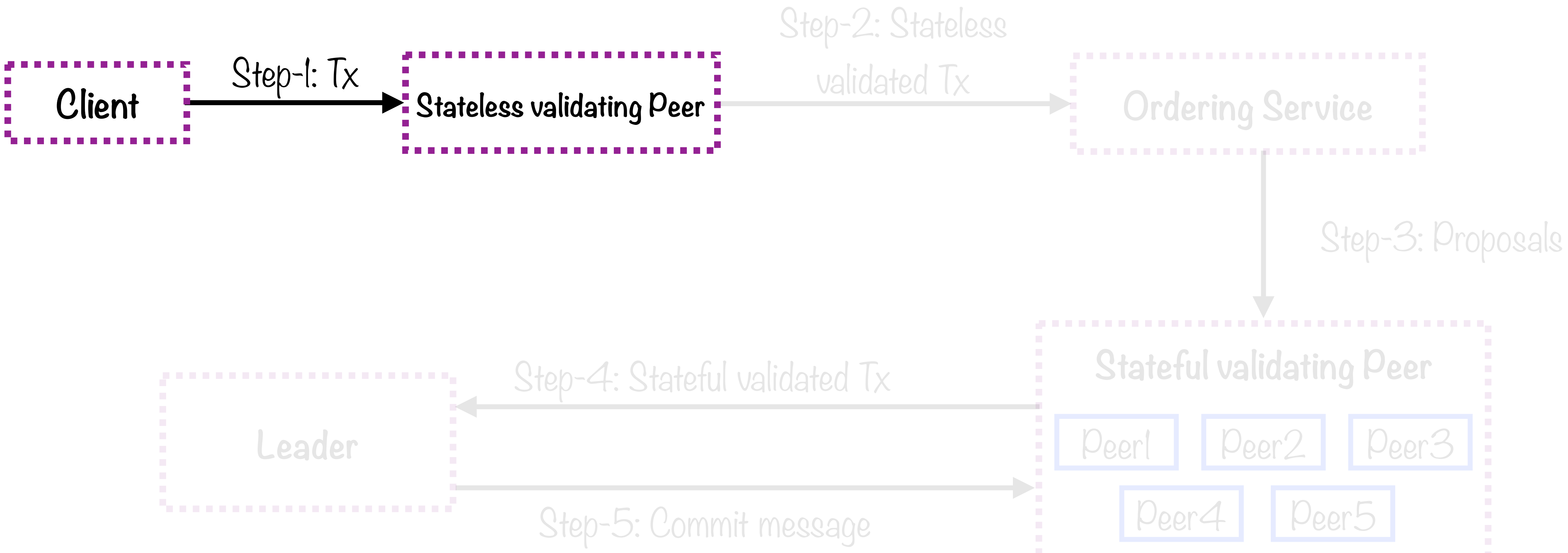
Transaction Flow in Hyperledger Iroha



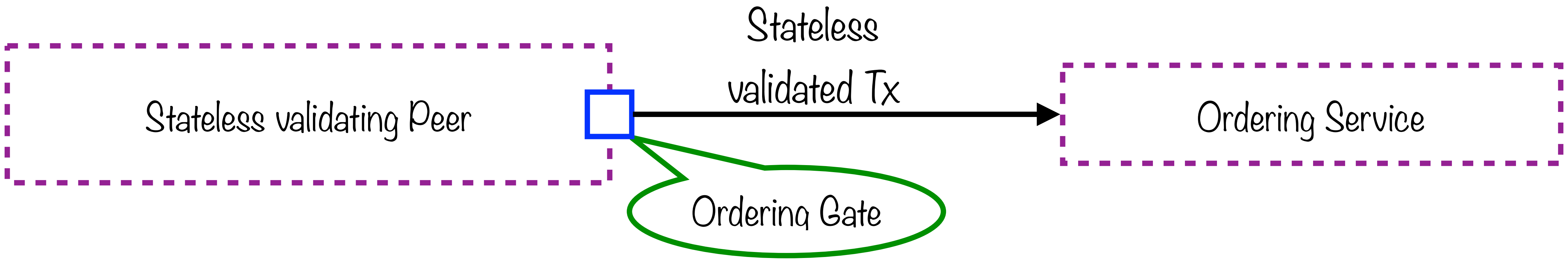
Stateless validation

- Checks the transaction schema
- Verifies the signature of the sender
- Is very quick

Transaction Flow in Hyperledger Iroha

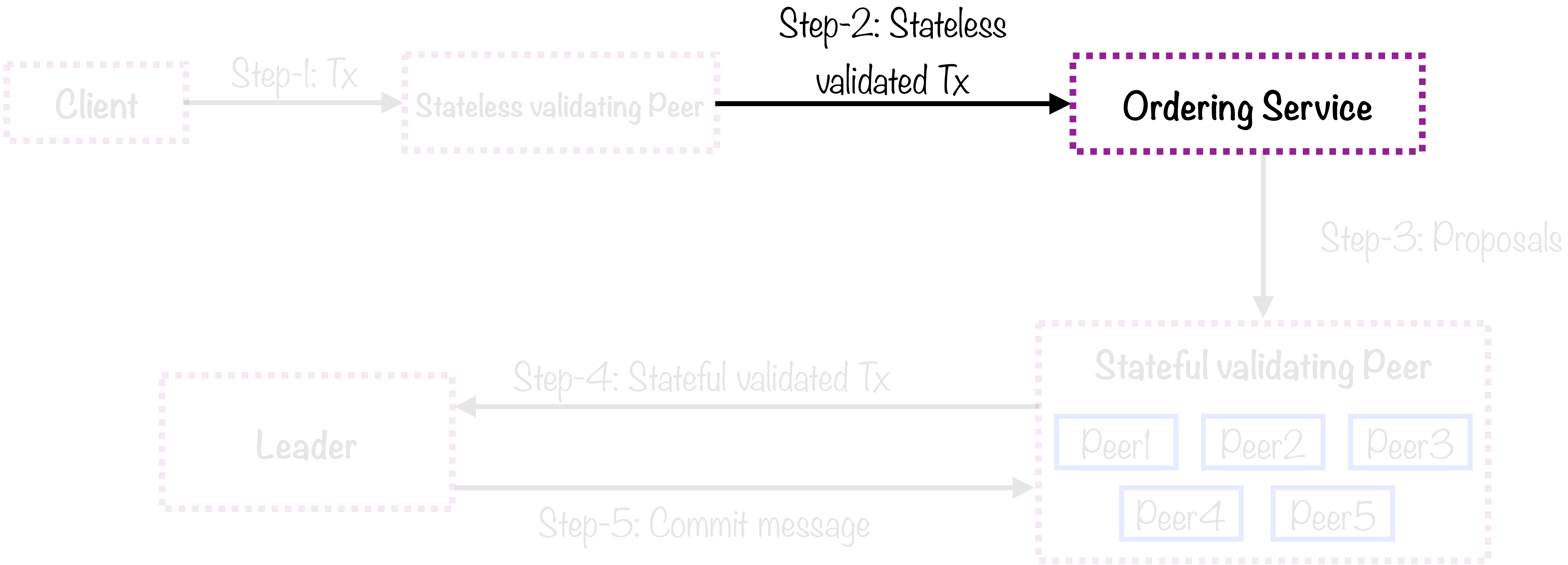


Transaction Flow in Hyperledger Iroha



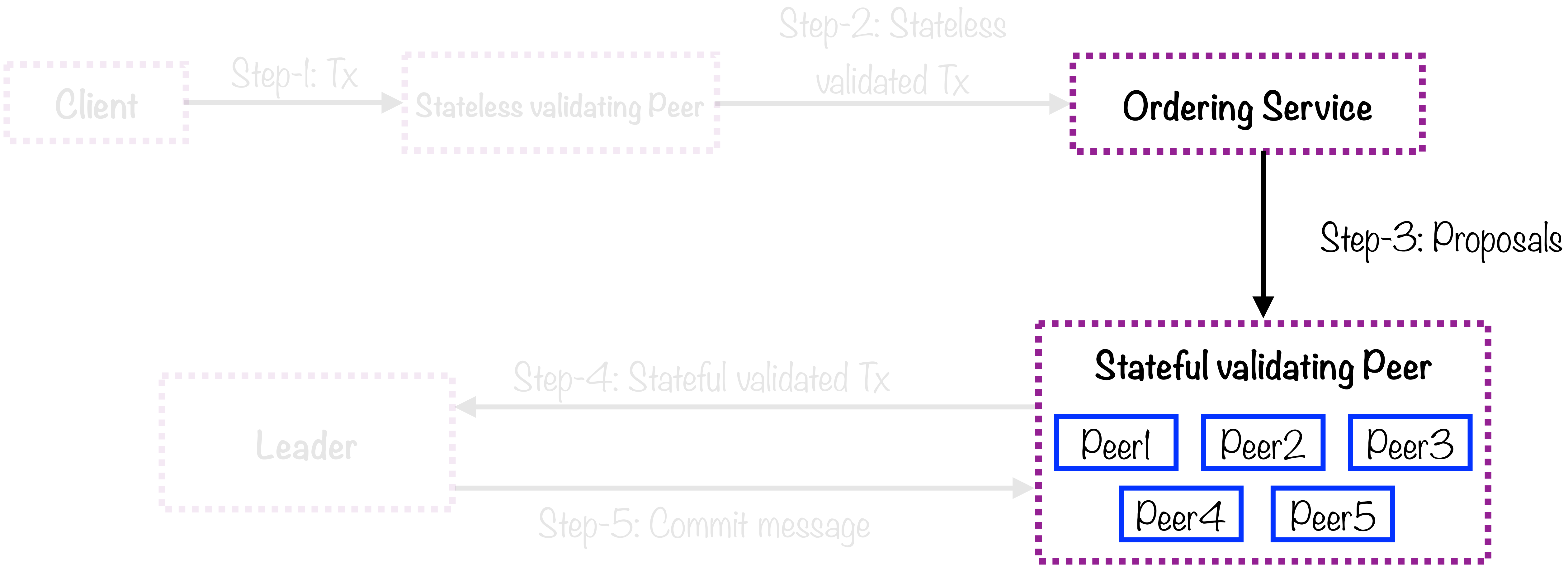
After stateless validation, the transaction is first sent to the ordering gate, which is responsible for choosing the right strategy of connection to the ordering service

Transaction Flow in Hyperledger Iroha

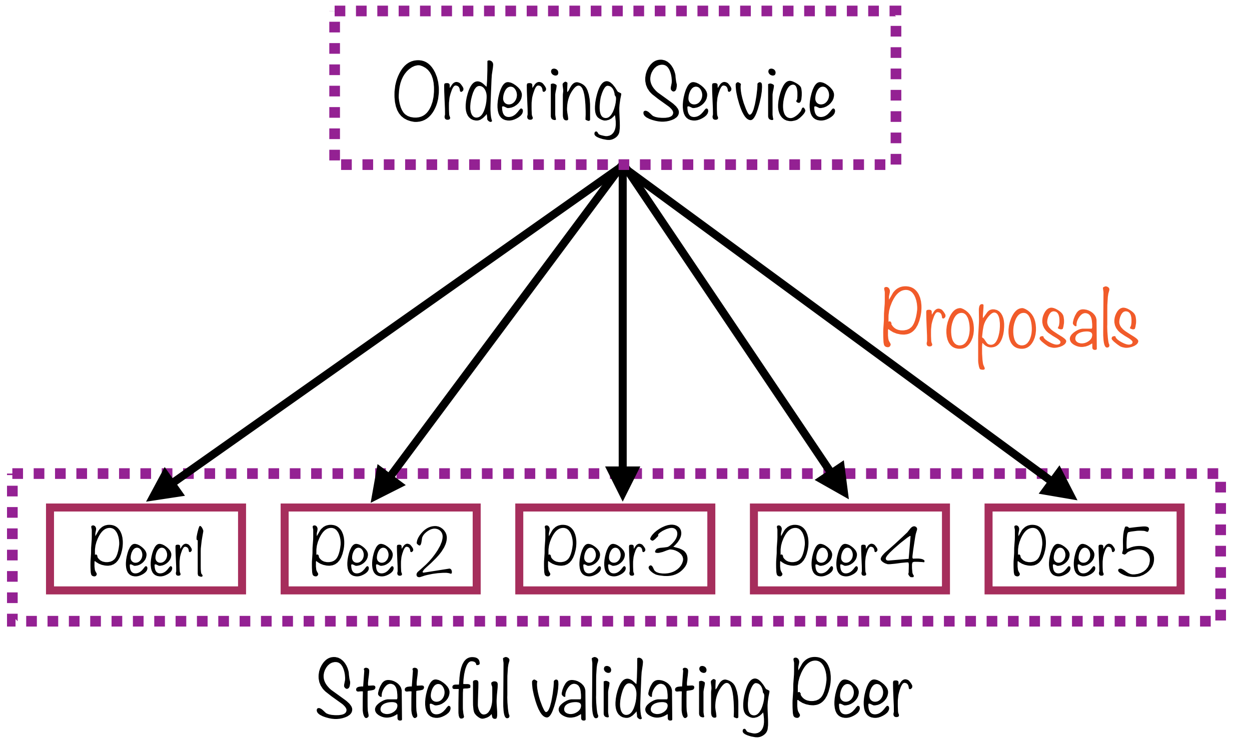


Kafka is used as ordering service (like in Fabric)

Transaction Flow in Hyperledger Iroha

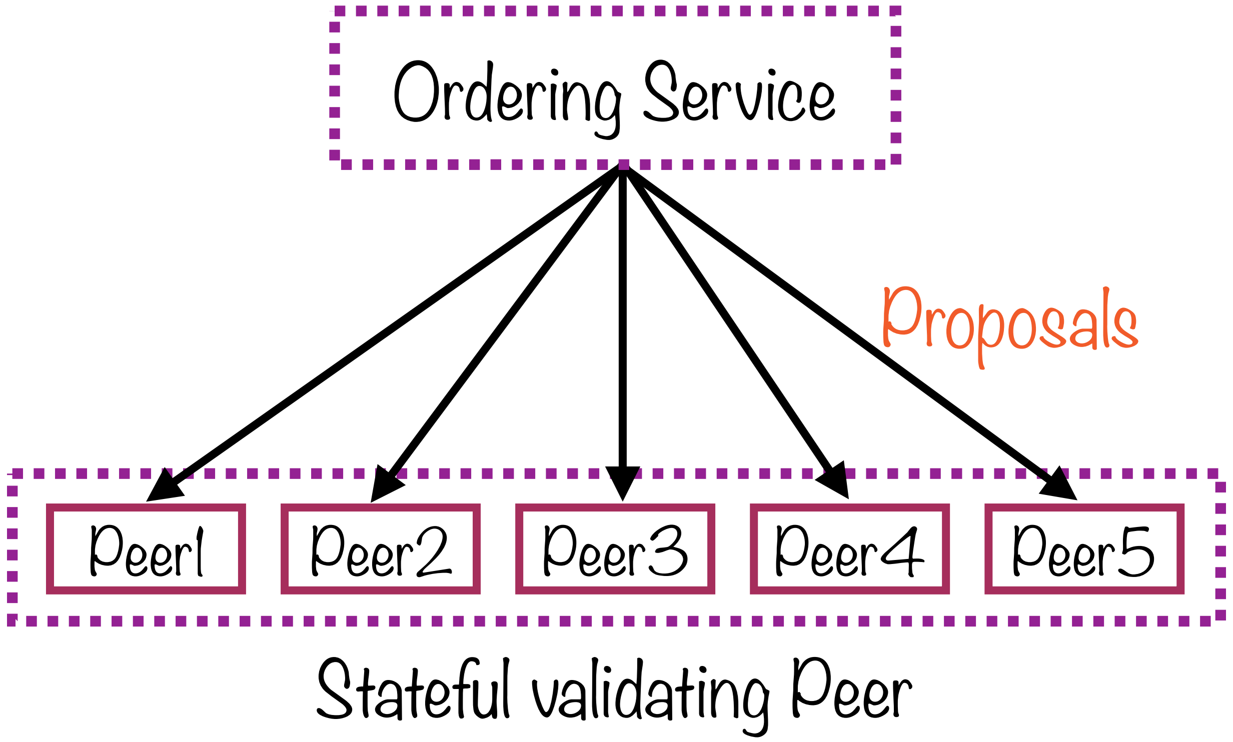


Transaction Flow in Hyperledger Iroha



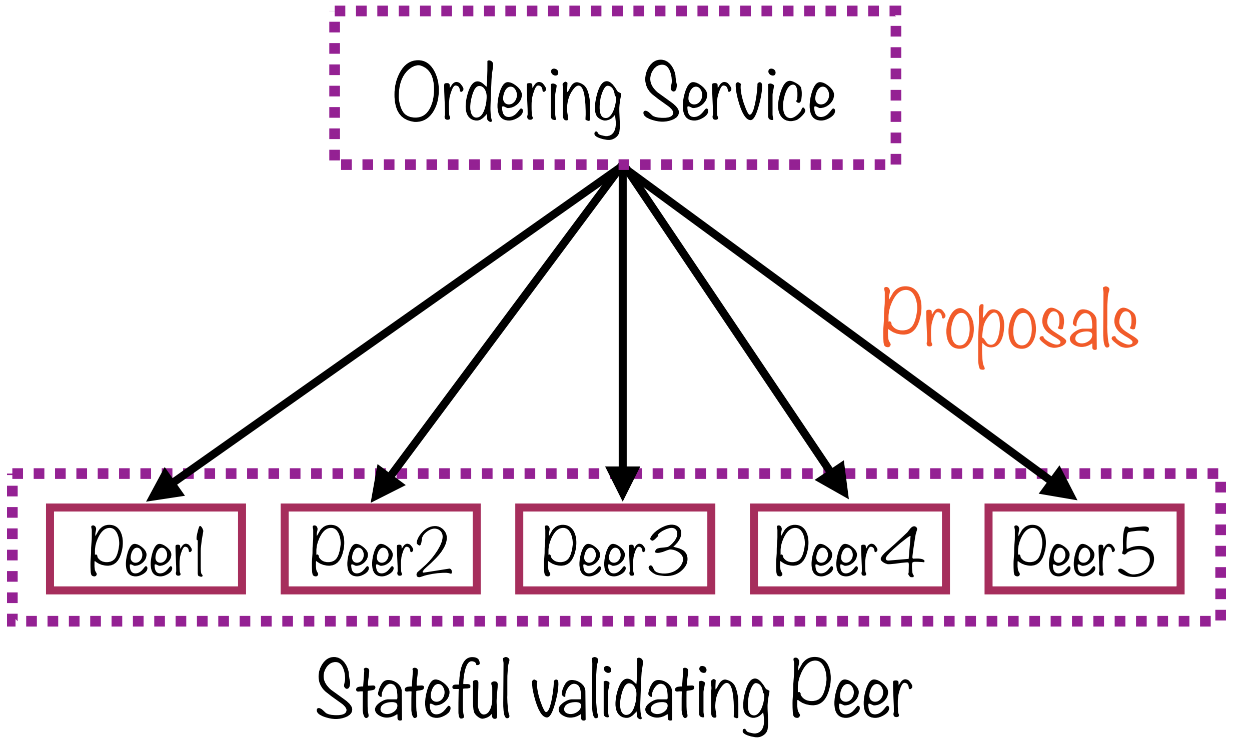
Ordering service forwards ordered (but still unverified) transactions to validating peers

Transaction Flow in Hyperledger Iroha



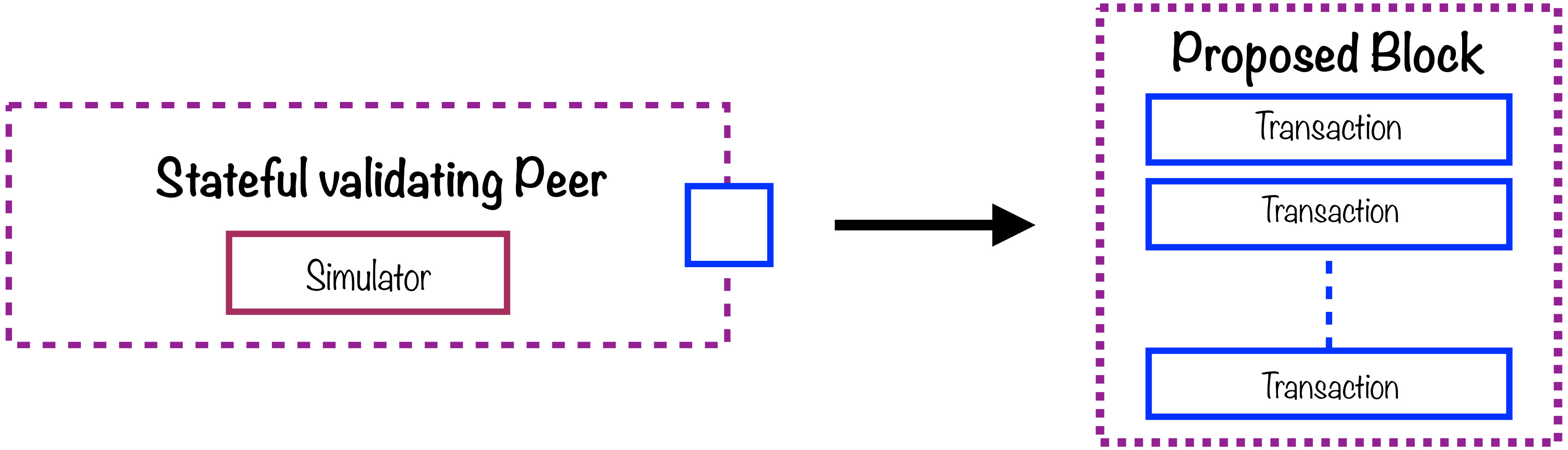
These ordered but unverified transactions are called proposals

Transaction Flow in Hyperledger Iroha



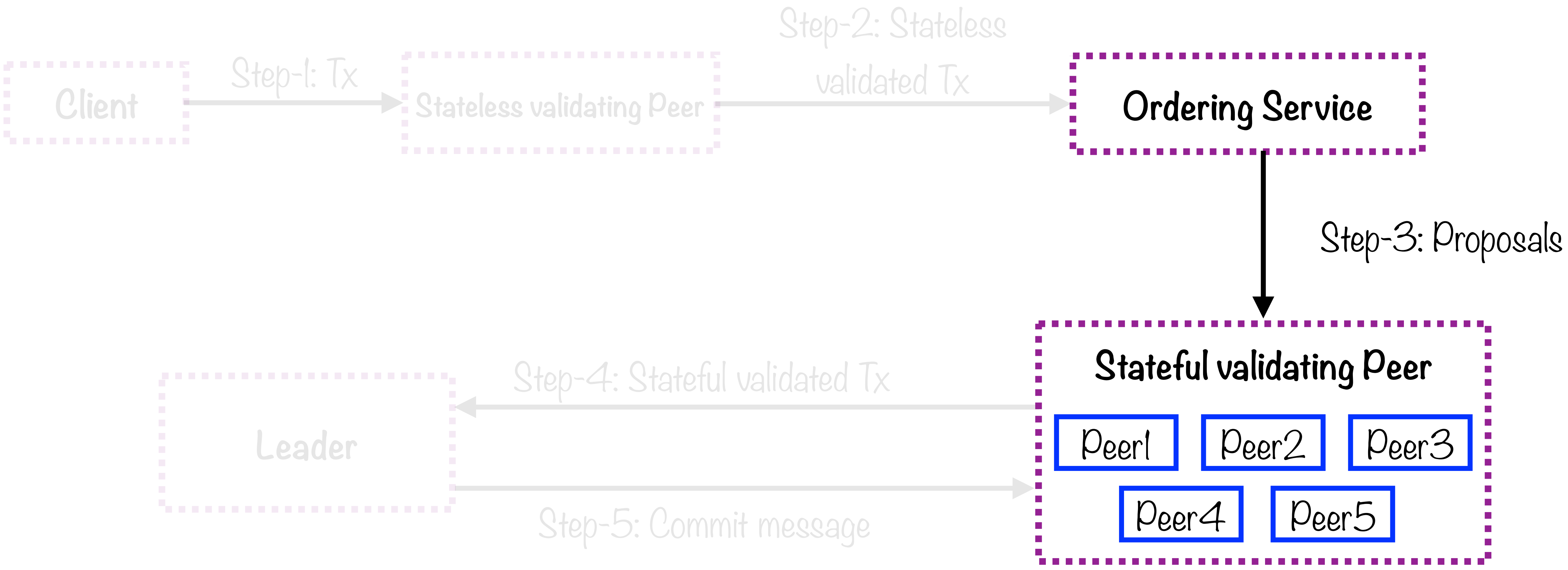
Proposals are forwarded only when enough transactions have been batched, or when a specified interval has elapsed

Transaction Flow in Hyperledger Iroha



Each peer verifies the contents of proposal in the Simulator and creates a block which consists only of verified transactions

Transaction Flow in Hyperledger Iroha

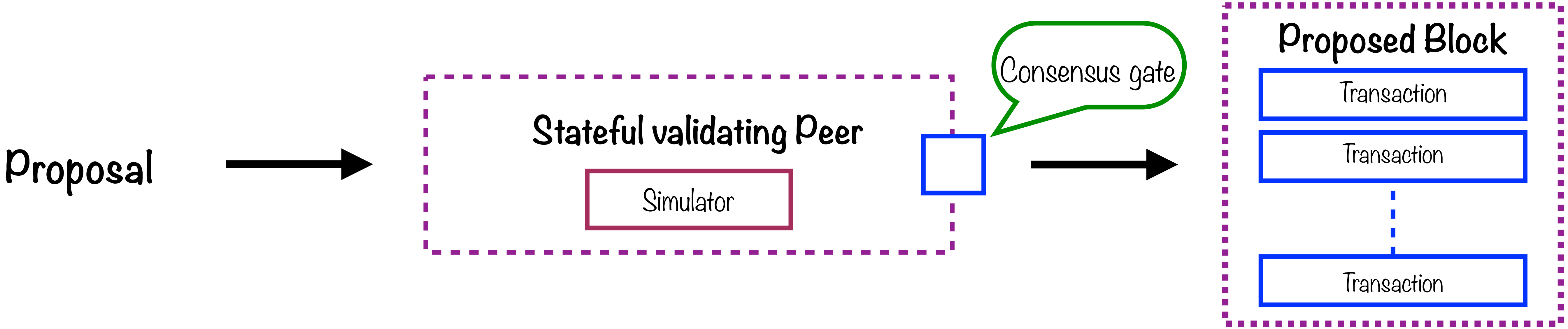


Stateful Validation

Stateful peers

- input: proposal
- output: proposed block
- action: discard all invalid transactions

Transaction Flow in Hyperledger Iroha



After that the block is sent to the Consensus gate which performs YAC (Yet Another Consensus) logic

Stateful Validation

Stateful peers access information

- account permissions
- assets in each account
- ownership of assets to be transferred
- quantity checks of assets to be transferred

Stateful Validation

Proposed block is ordered, verified transaction list

If 2/3 of peers agree on this, consensus is achieved

- proposed block is broadcast to all peers

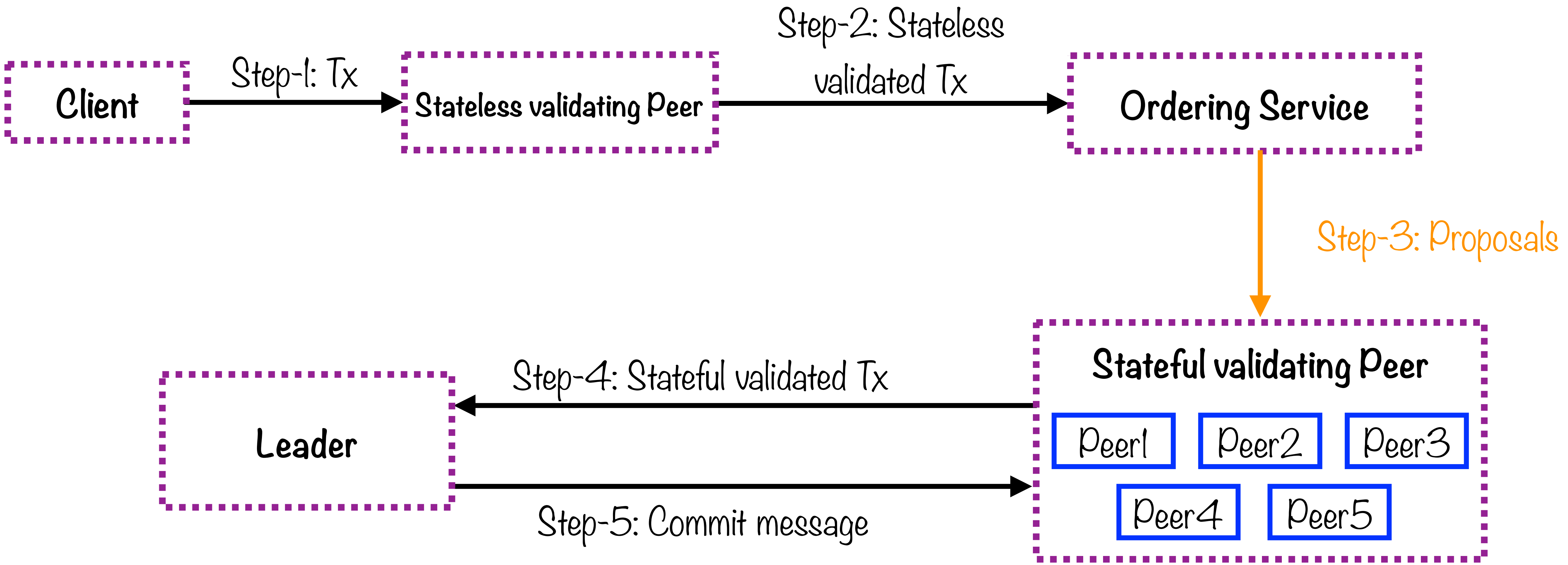
Stateful Validation

"Byzantine Fault Tolerant"

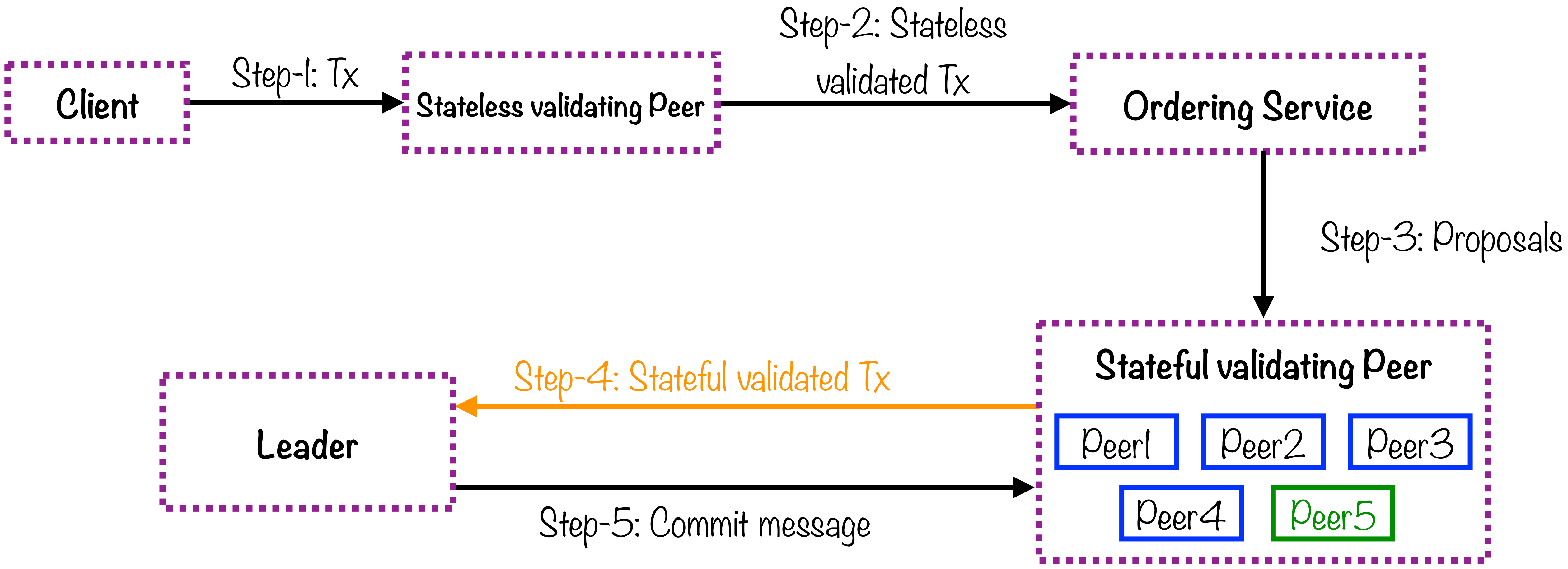
Can cope with malicious nodes

(Provided less than 1/3 nodes are rogue)

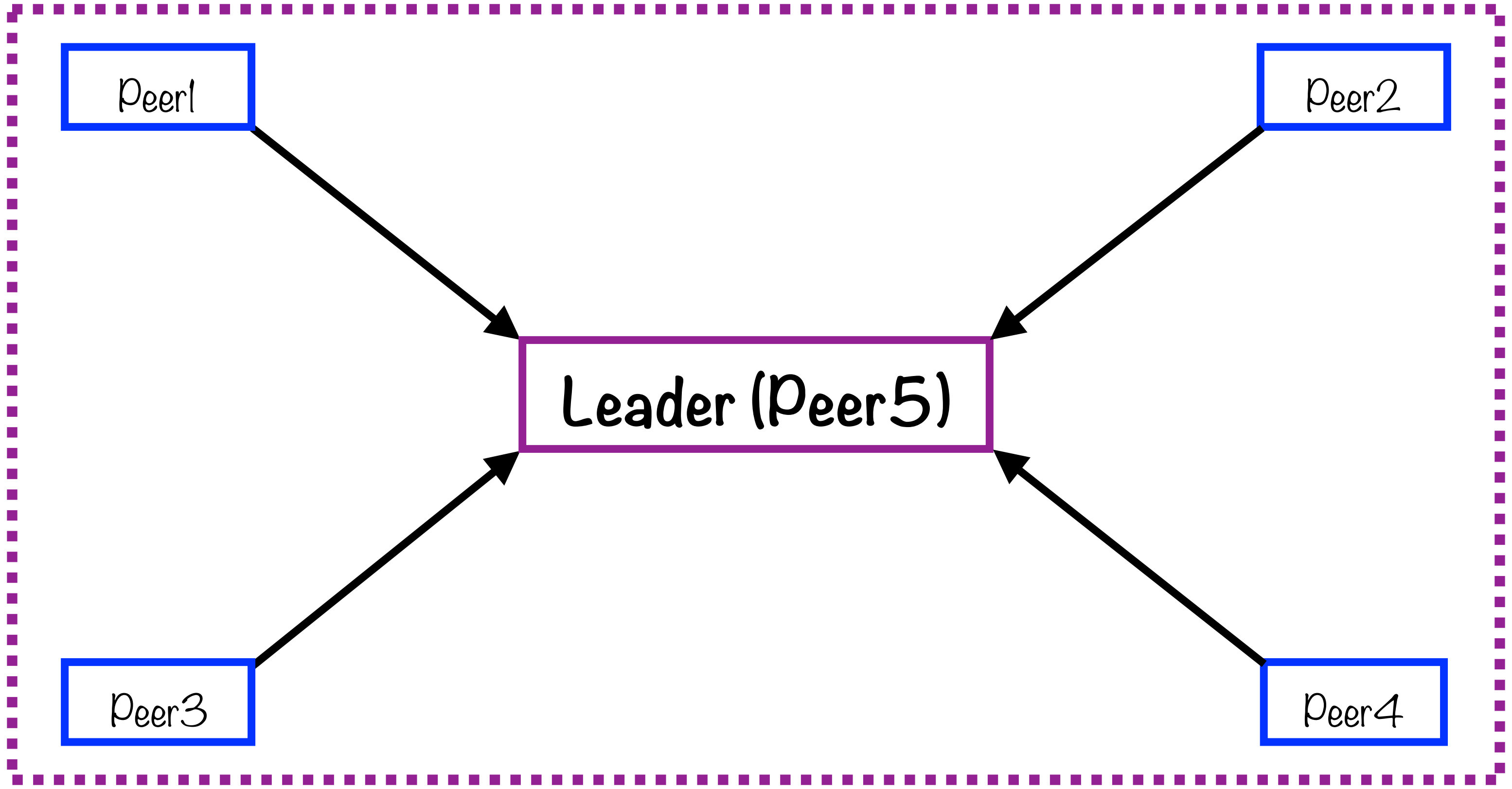
Transaction Flow in Hyperledger Iroha



Transaction Flow in Hyperledger Iroha

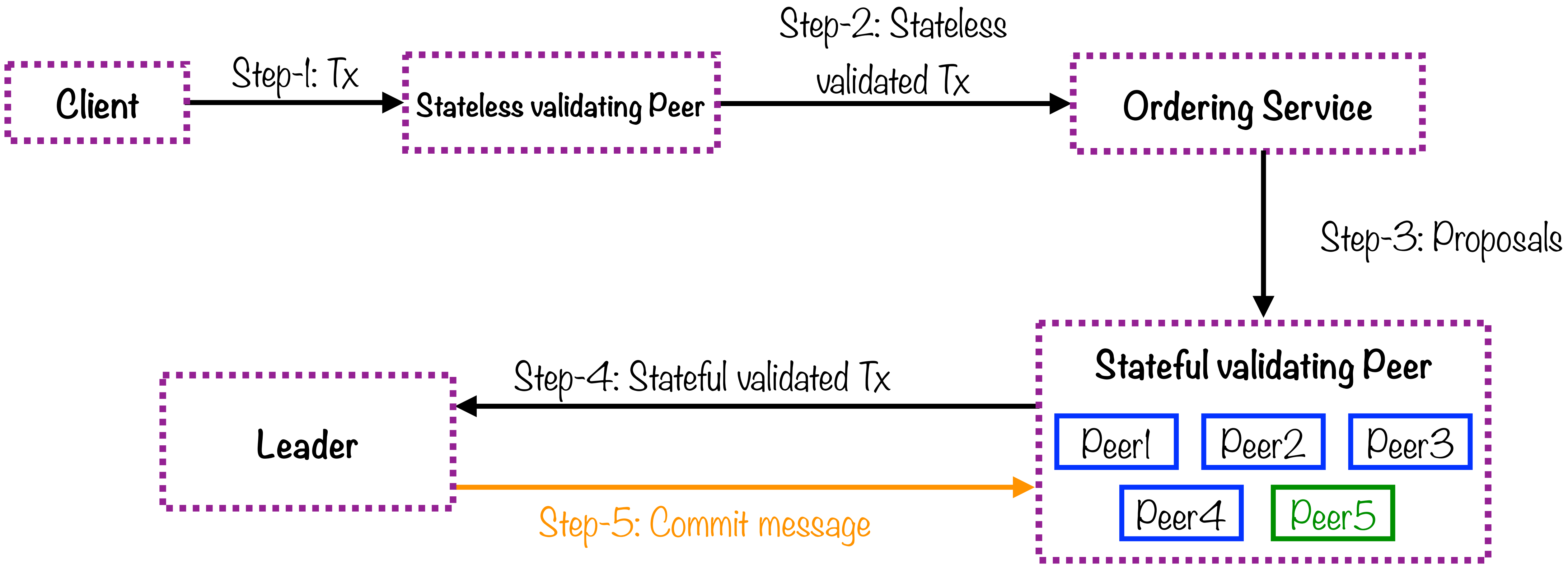


Transaction Flow in Hyperledger Iroha

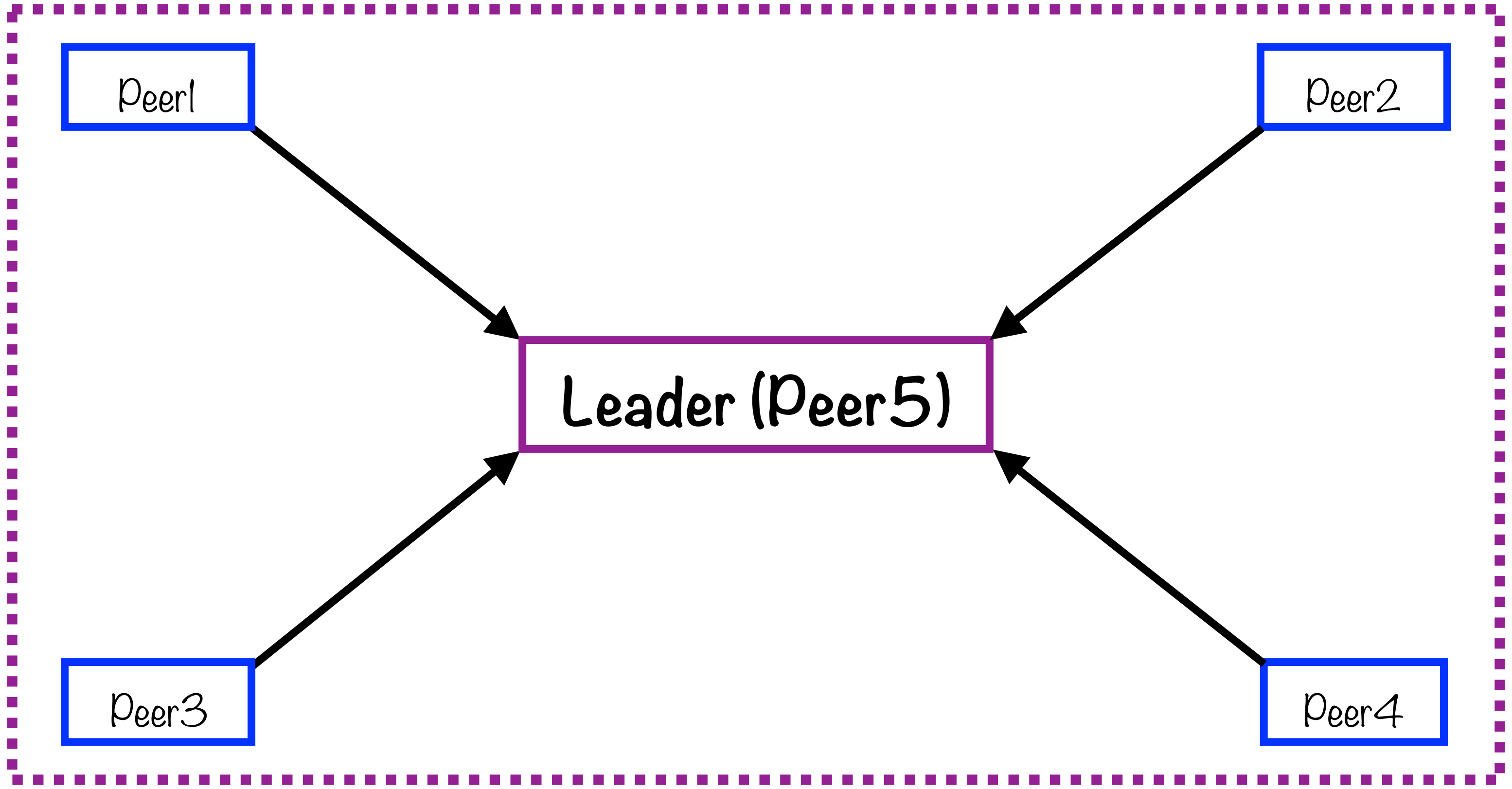


Each stateful validating peer signs the block and send their proposed block to the leader

Transaction Flow in Hyperledger Iroha



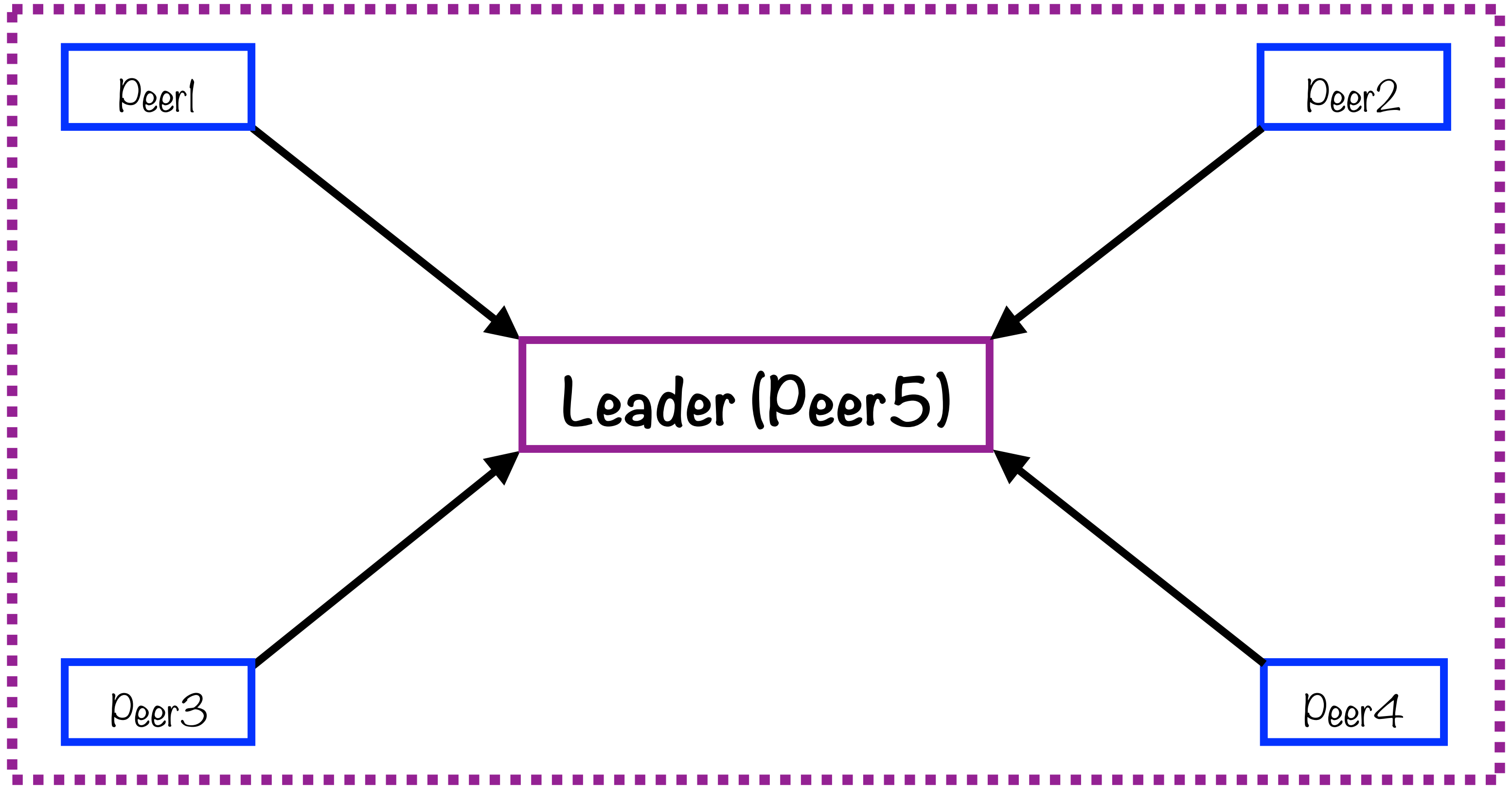
Transaction Flow in Hyperledger Iroha



Leader has 2/3 of signed proposed block

When the leader receives the same proposed block from 2/3 peers, then it sends a commit message to all peers

Transaction Flow in Hyperledger Iroha



Leader has 2/3 of signed proposed block

Once the commit message has been sent, the proposed block becomes the next block in the chain of every peer by using the synchronizer

Demo

Working with Hyperledger Iroha

Working with Hyperledger Sawtooth

Hyperledger

Umbrella project of open source blockchains; started in December 2015 by the Linux Foundation

Umbrella of Frameworks

Hyperledger Fabric

Hyperledger Sawtooth

Hyperledger Iroha

Hyperledger Burro

Hyperledger Indy

Umbrella of Frameworks

Hyperledger Fabric

Hyperledger Sawtooth

Hyperledger Iroha

Hyperledger Burro

Hyperledger Indy

Hyperledger Sawtooth

Distributed ledger contributed by Intel, designed for larger, enterprise-grade blockchain networks

Proof of Elapsed Time (PoET)

Consensus algorithm designed for large networks; relies on election of a leader using a lottery function

Key features of Sawtooth

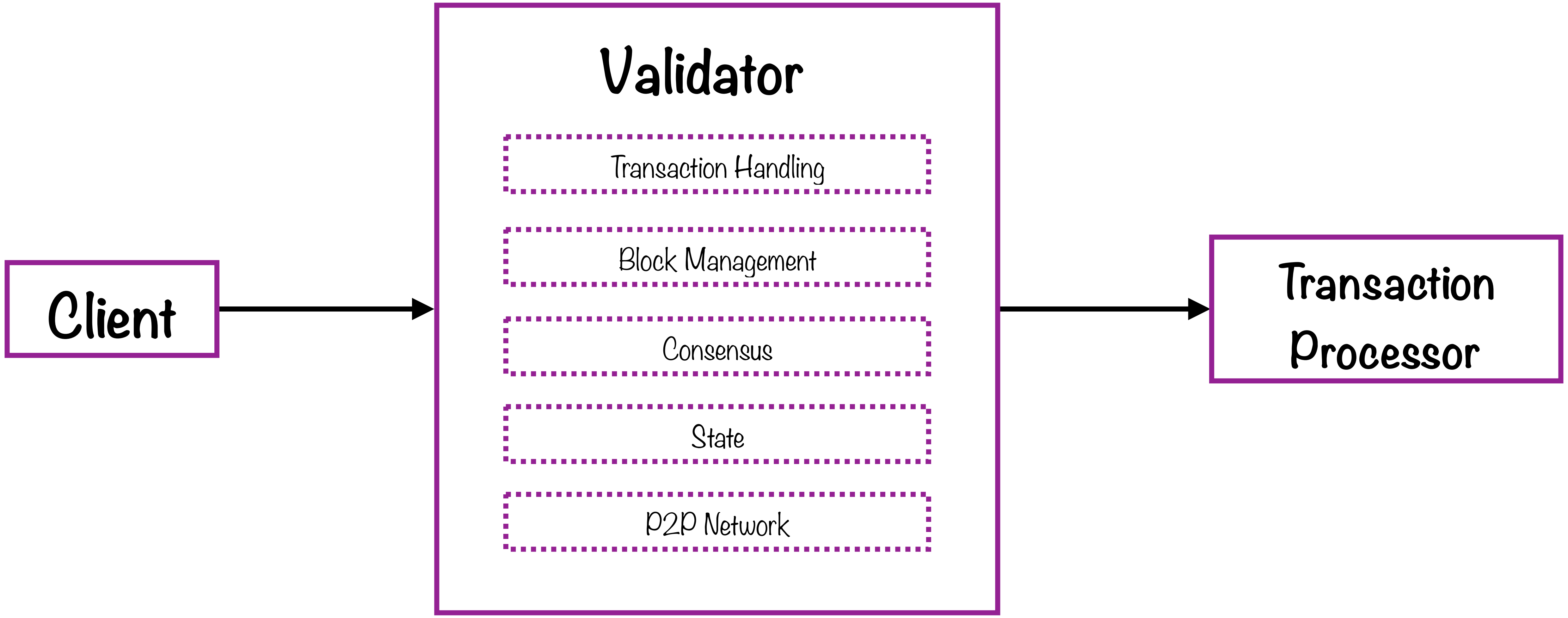
Scalability, which is good for larger blockchain networks

Unique support for permissioned and permissionless infrastructure

Allows confidential transactions, without passing information through a central authority

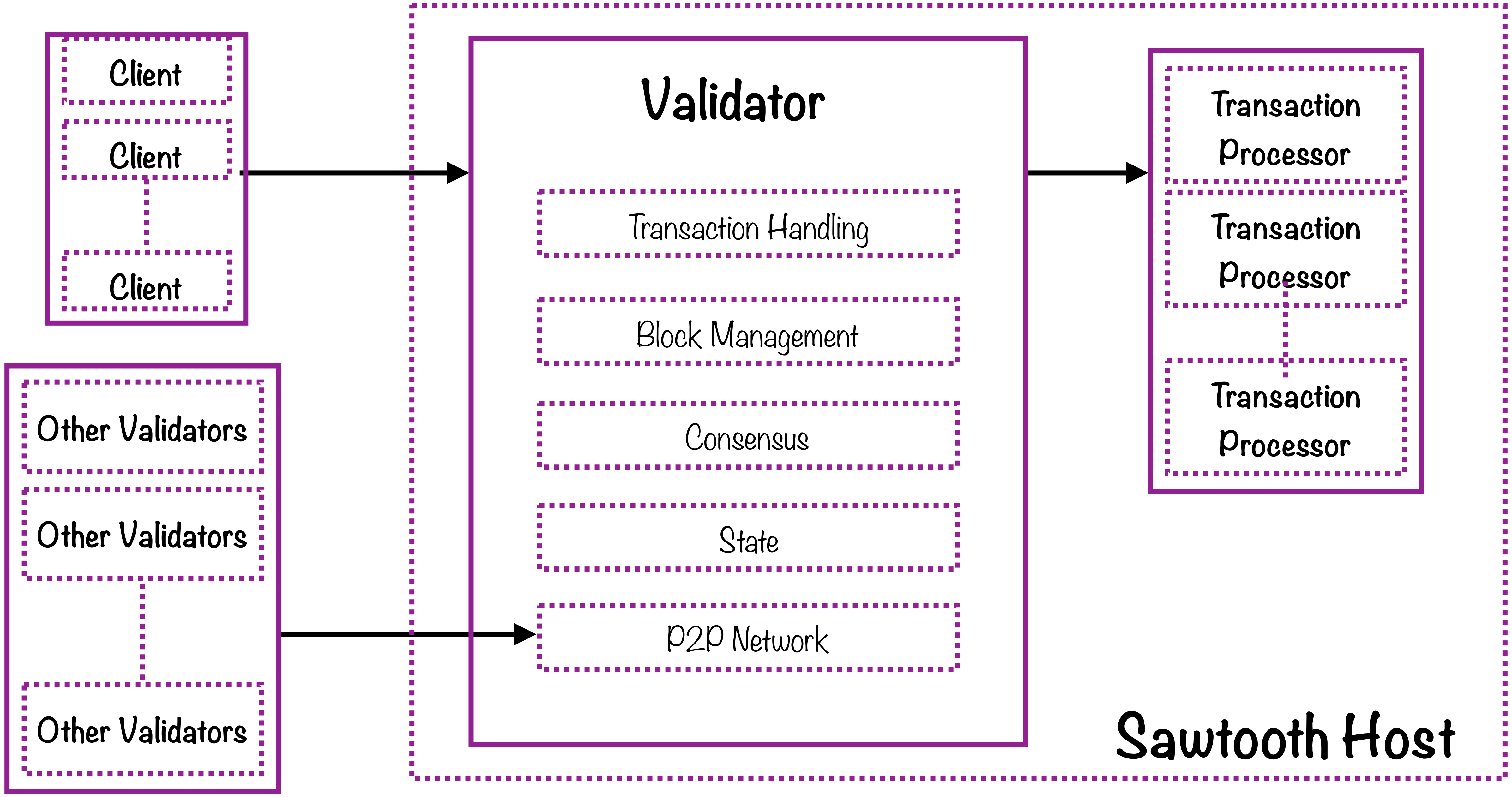
Finds its potential in IoT, manufacturing, finance, and enterprise

Hyperledger Sawtooth Architecture



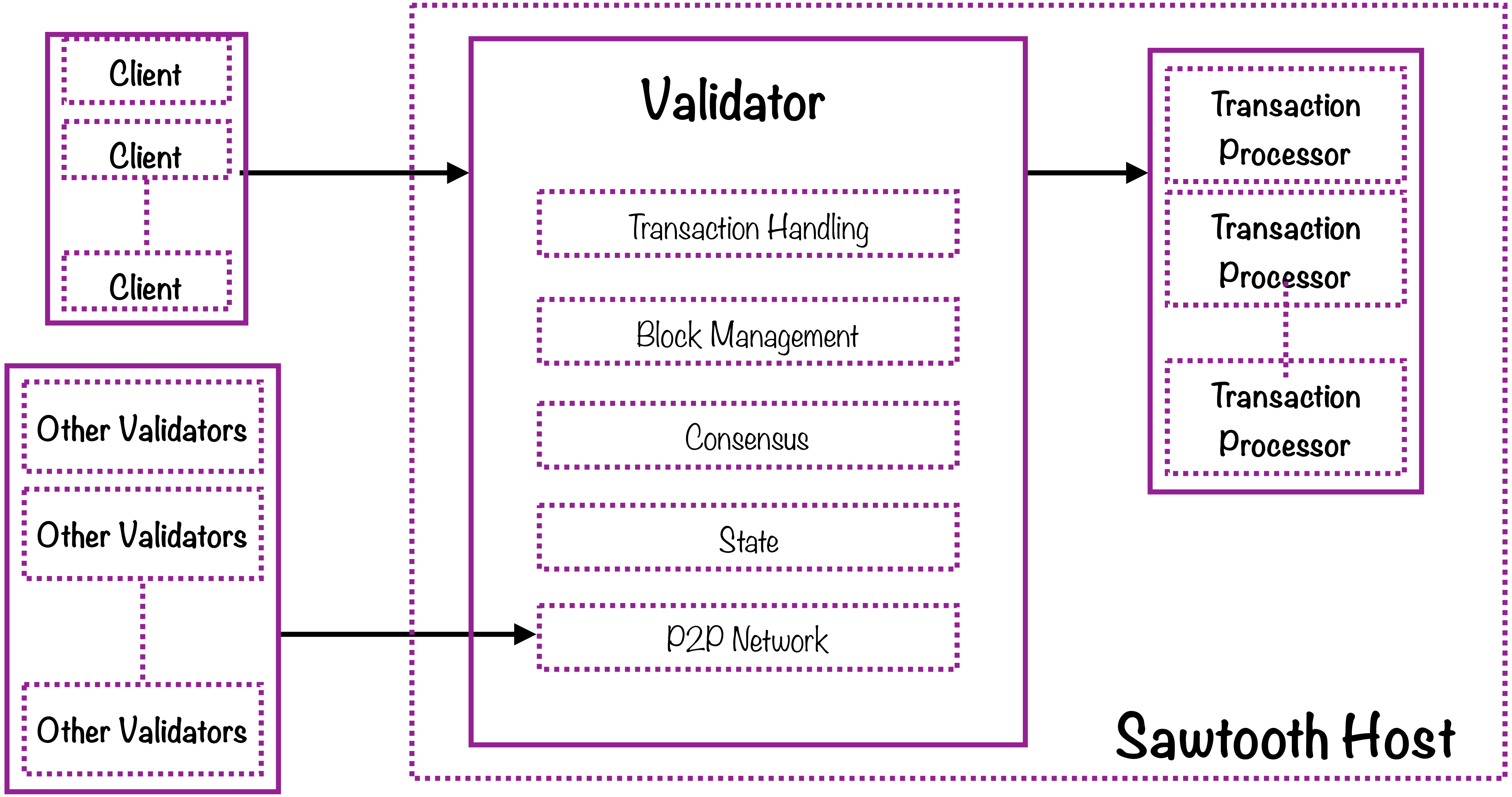
Basic architecture: Client, validator and transaction processor

Hyperledger Sawtooth Architecture



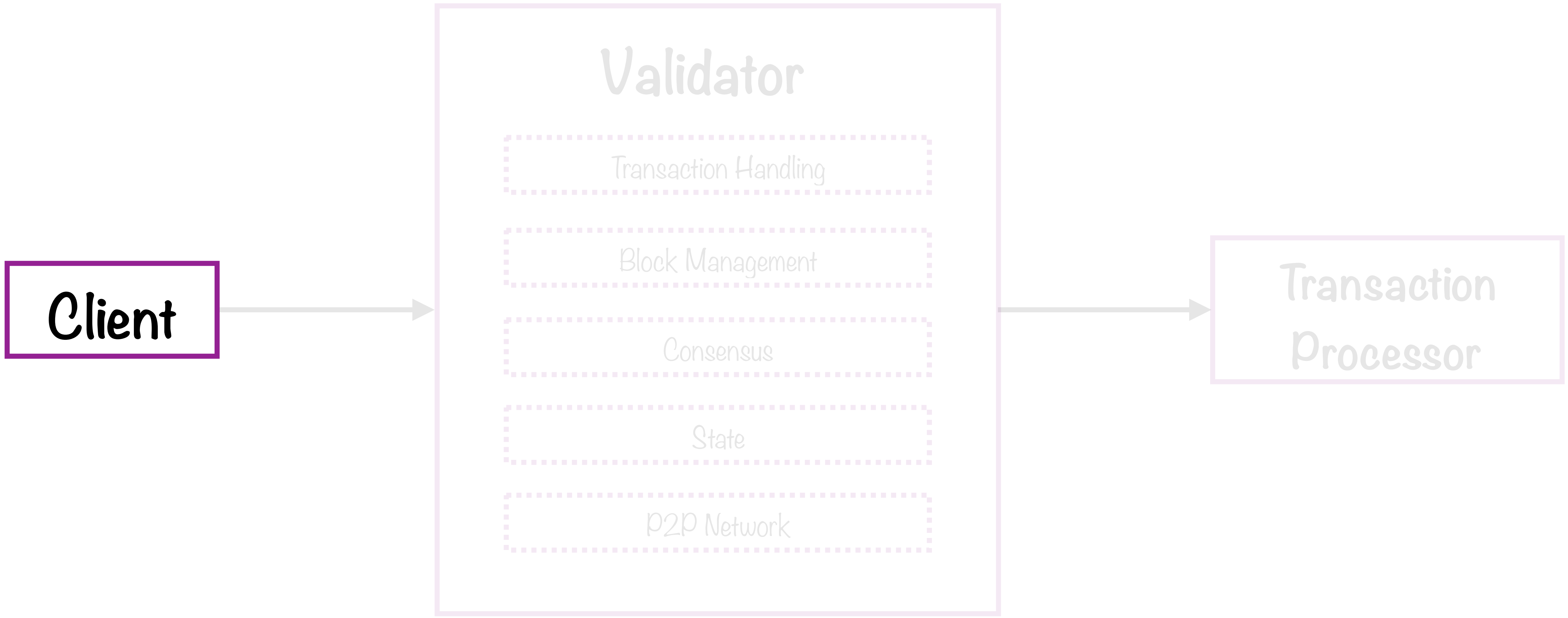
Real applications - multiple clients and processors

Hyperledger Sawtooth Architecture



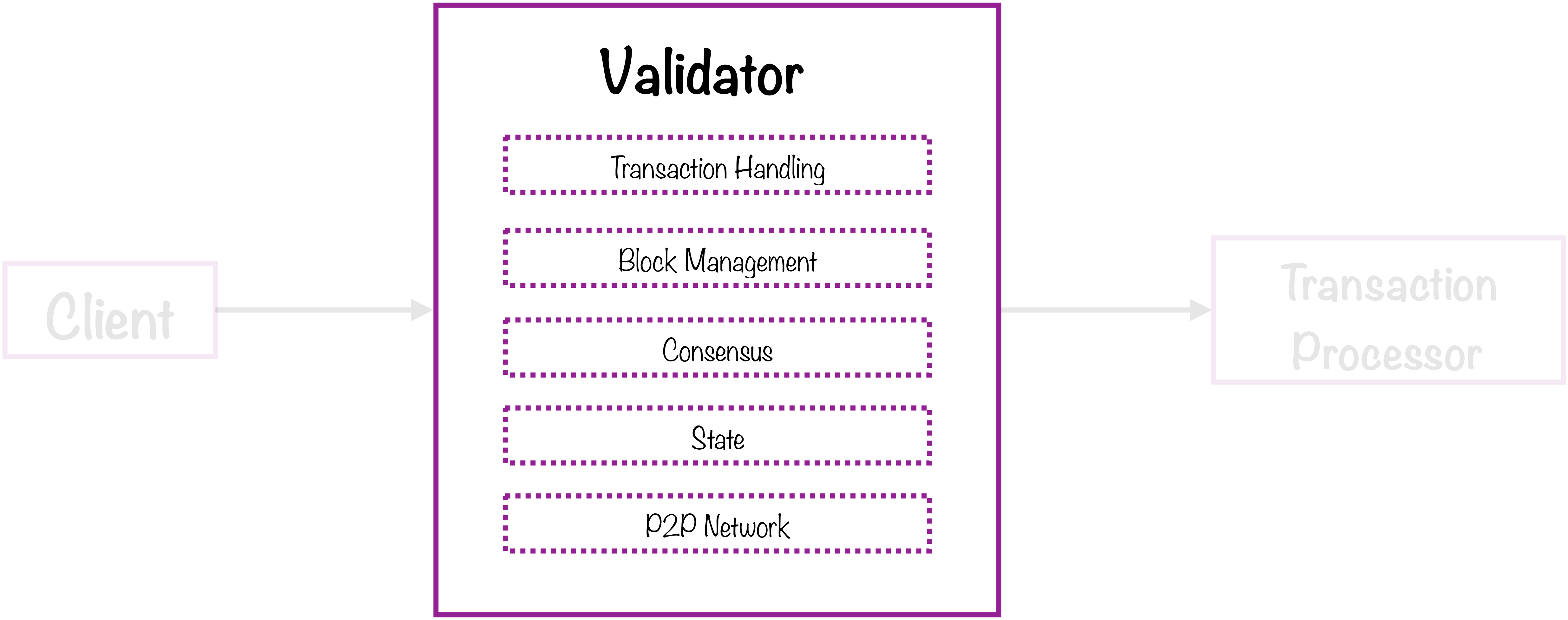
Multiple potential validators - one is selected by the Proof-of-Elapsed-TIme algorithm (more soon)

Hyperledger Sawtooth Architecture



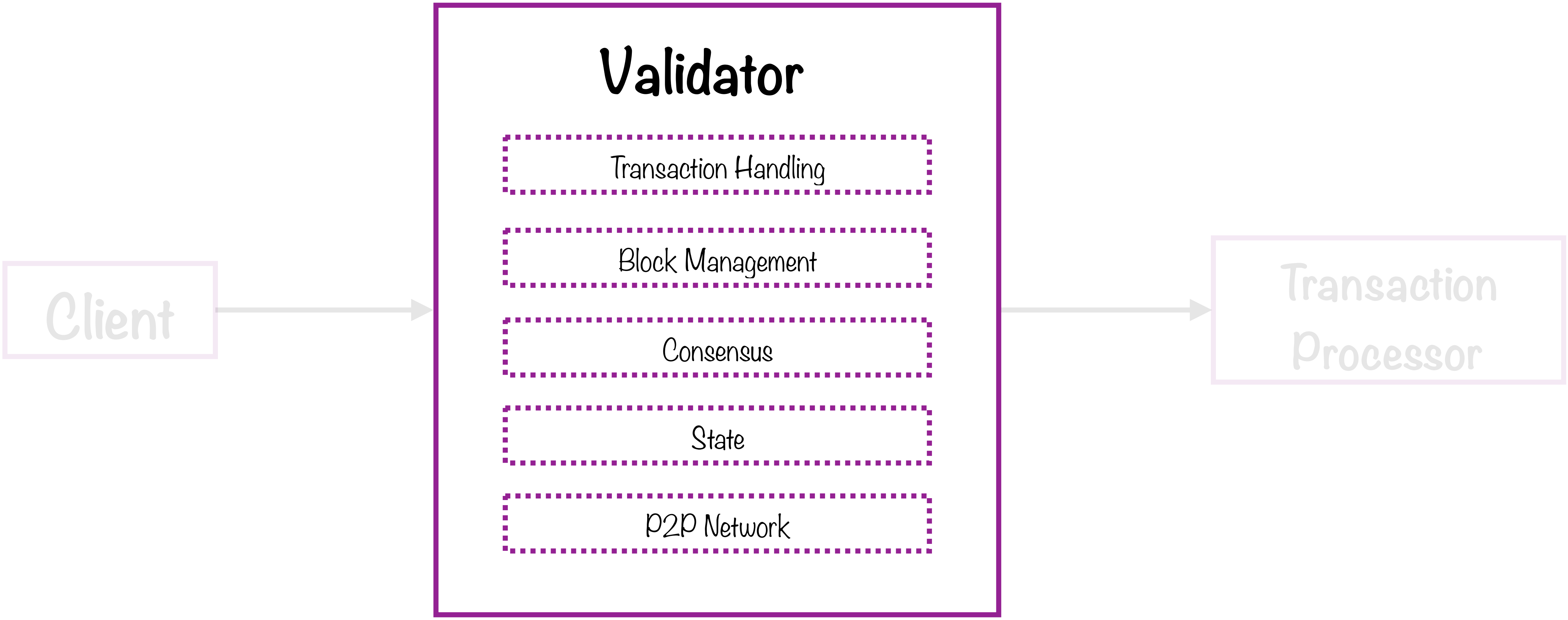
A user creates a batch containing one or more transactions, submits it to the validator, usually via a client

Hyperledger Sawtooth Architecture



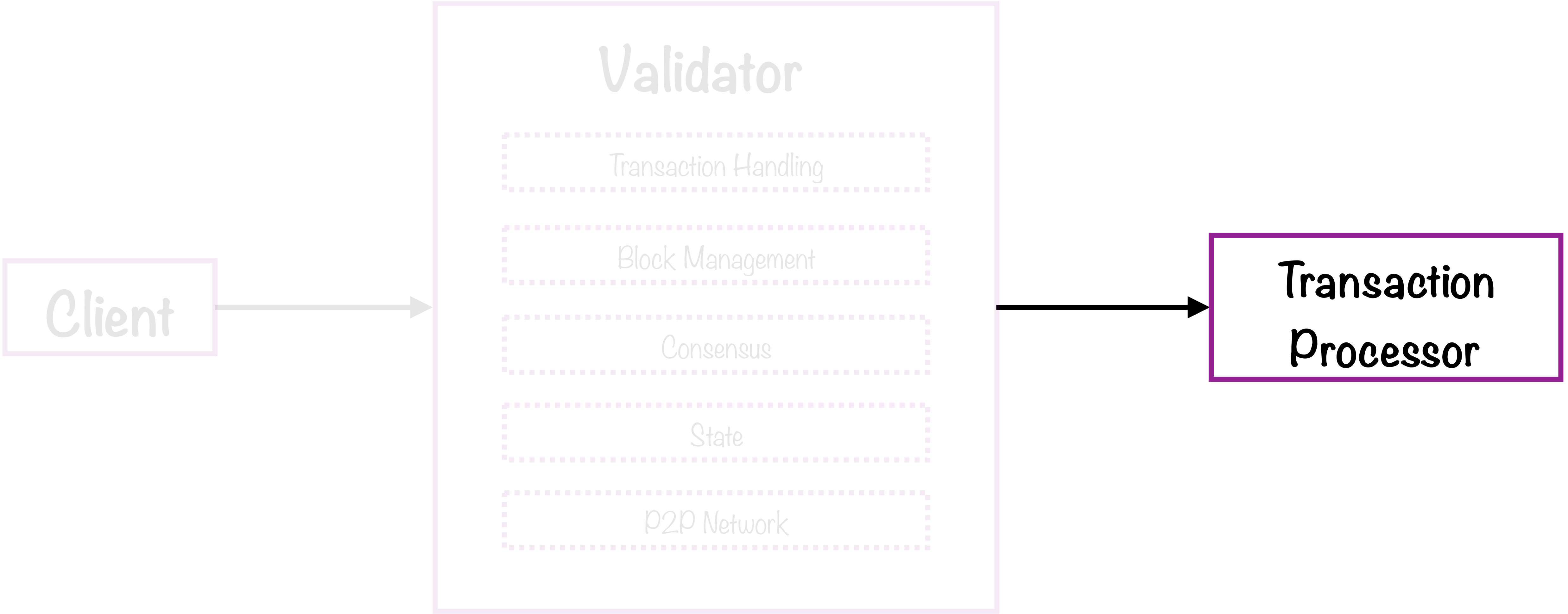
The validator then validates the transactions, if all the transactions are valid then state will be updated

Hyperledger Sawtooth Architecture



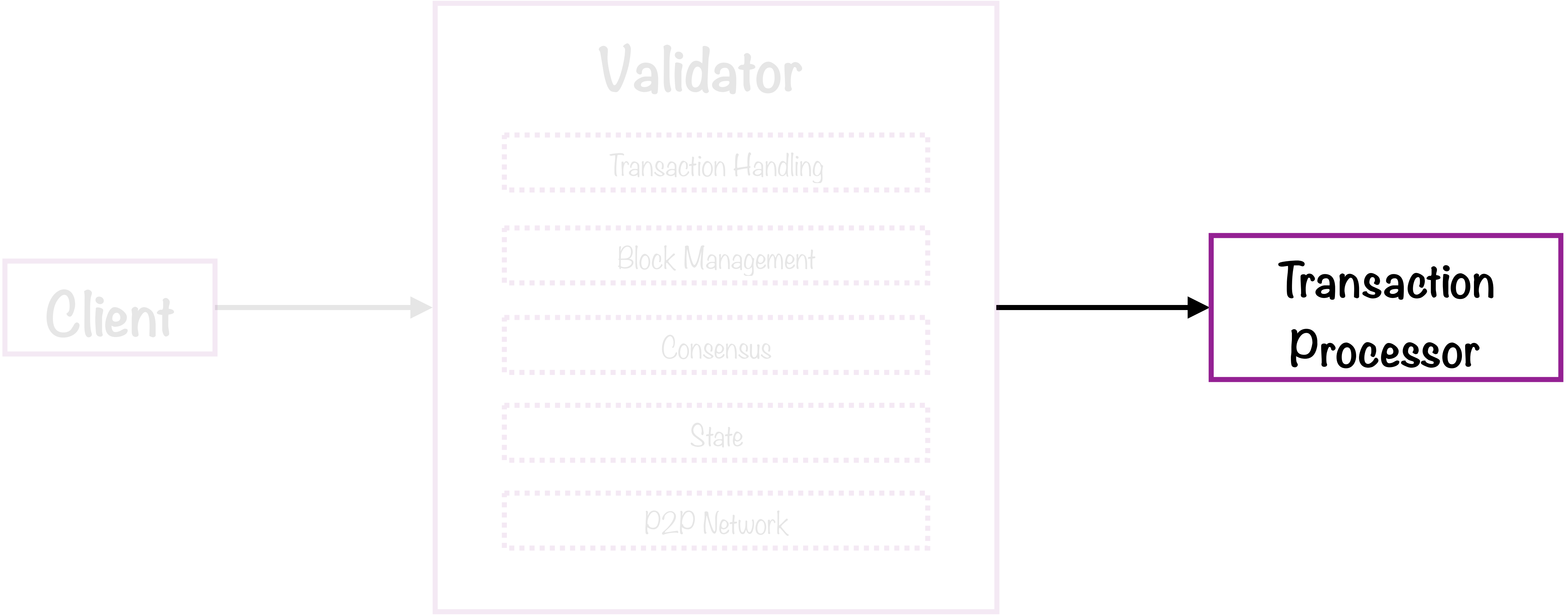
If any transaction is invalid then state will not be updated

Hyperledger Sawtooth Architecture



Transaction processors can be written in a variety of languages, including Javascript, Java, Rust, Python, and Go.

Hyperledger Sawtooth Architecture



Users can implement custom transaction processing logic

Proof of Elapsed Time (PoET)

Consensus algorithm designed for large networks; relies on election of a leader using a lottery function

Proof of Elapsed Time (PoET)

Each validator waits random length of time

The validator with the shortest wait is appointed leader

Proof of Elapsed Time (PoET)

Drawbacks:

- Vulnerable to rogue entity
- Does not currently support Byzantine Fault Tolerance

Advantages:

- Extremely scalable
- Energy efficient

Demo

Working with Hyperledger Sawtooth

Deploying Blockchains Using Hyperledger Cello

Demo

Working with Hyperledger Cello

Summary

A blockchain is a ledger of transactions which is distributed, immutable and verifiable

Hyperledger is an umbrella of open-source blockchain initiatives

Hyperledger addresses several flaws present in other blockchain implementations in addition to including new features

Different Hyperledger Frameworks for varying use cases

Hyperledger Tools provide a high-level abstraction for working with frameworks