Agenda

- Hyperledger
- Hyperledger Fabric
- Permissionless Vs Permissioned blockchains
- Fabric component architecture
- Fabric ordering service RAFT
- Distributed System Communicate
- BFT
- PBFT
- DLS
- BDLS
- Hyperledger Fabric Goal

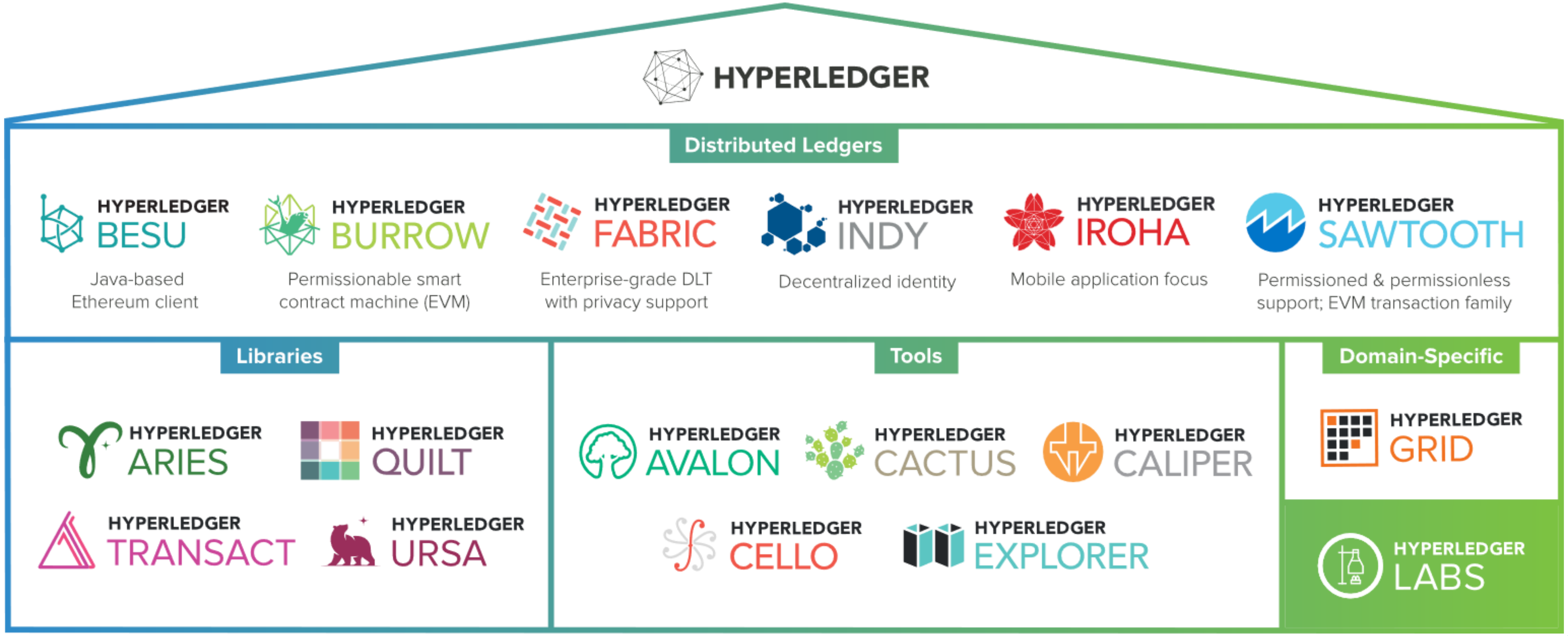
HYPERLEDGER

Hyperledger Project

Is an open-source collaborative effort created to advance cross-industry Blockchain technologies.

It is a project under the LINUX Foundation, Hyperledger is not a single project but a collection of projects under the Hyperledger umbrella

HYPERLEDGER



HYPERLEDGER

Distributed Ledgers

- HYPERLEDGER BESU: Java-based Ethereum client
- HYPERLEDGER BURROW: Permissionable smart contract machine (EVM)
- HYPERLEDGER FABRIC: Enterprise-grade DLT with privacy support
- HYPERLEDGER INDY: Decentralized identity
- HYPERLEDGER IROHA: Mobile application focus
- HYPERLEDGER SAWTOOTH: Permissioned & permissionless support; EVM transaction family

Libraries

- HYPERLEDGER ARIES
- HYPERLEDGER QUILT
- HYPERLEDGER TRANSACT
- HYPERLEDGER URSA

Tools

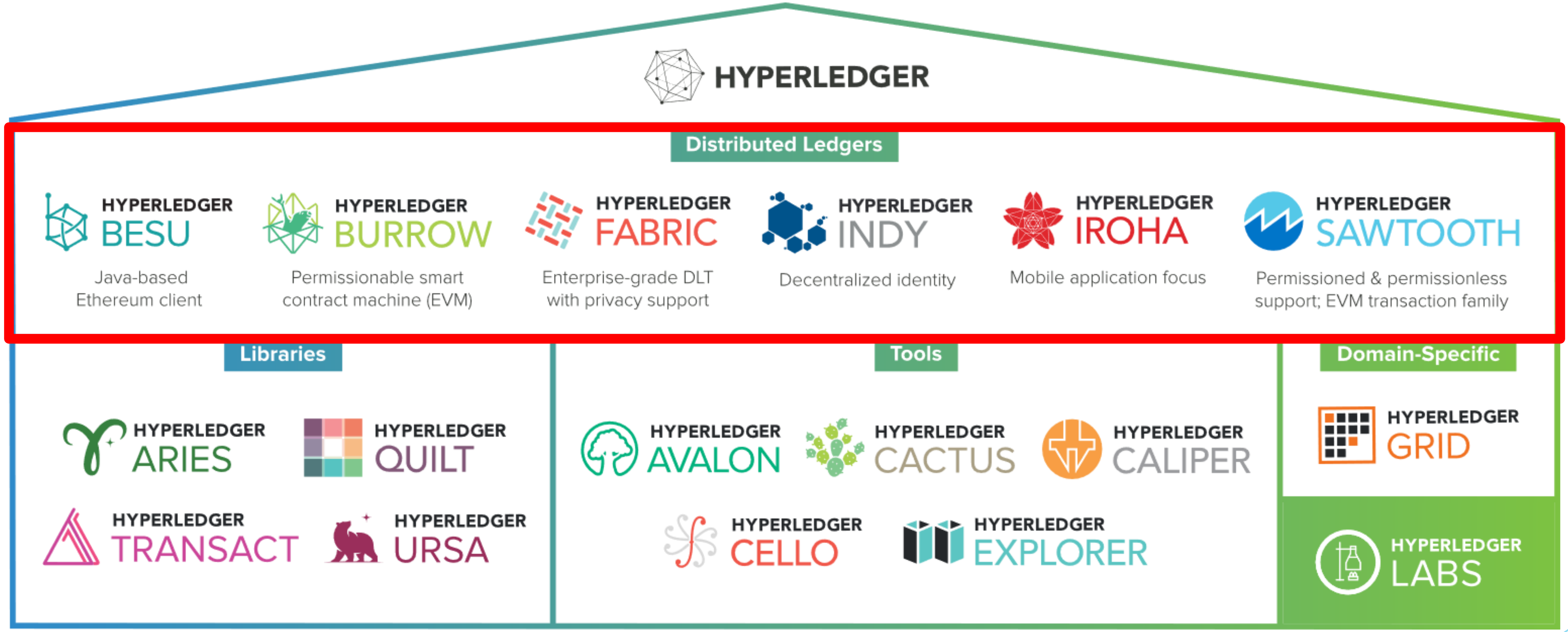
- HYPERLEDGER AVALON
- HYPERLEDGER CACTUS
- HYPERLEDGER CALIPER
- HYPERLEDGER CELLO
- HYPERLEDGER EXPLORER

Domain-Specific

- HYPERLEDGER GRID

HYPERLEDGER LABS

HYPERLEDGER



HYPERLEDGER

Distributed Ledgers

HYPERLEDGER BESU
Java-based Ethereum client

HYPERLEDGER BURROW
Permissionable smart contract machine (EVM)

HYPERLEDGER FABRIC
Enterprise-grade DLT with privacy support

HYPERLEDGER INDY
Decentralized identity

HYPERLEDGER IROHA
Mobile application focus

HYPERLEDGER SAWTOOTH
Permissioned & permissionless support; EVM transaction family

Libraries

HYPERLEDGER ARIES

HYPERLEDGER QUILT

HYPERLEDGER TRANSACT

HYPERLEDGER URSA

Tools

HYPERLEDGER AVALON

HYPERLEDGER CACTUS

HYPERLEDGER CALIPER

HYPERLEDGER CELLO

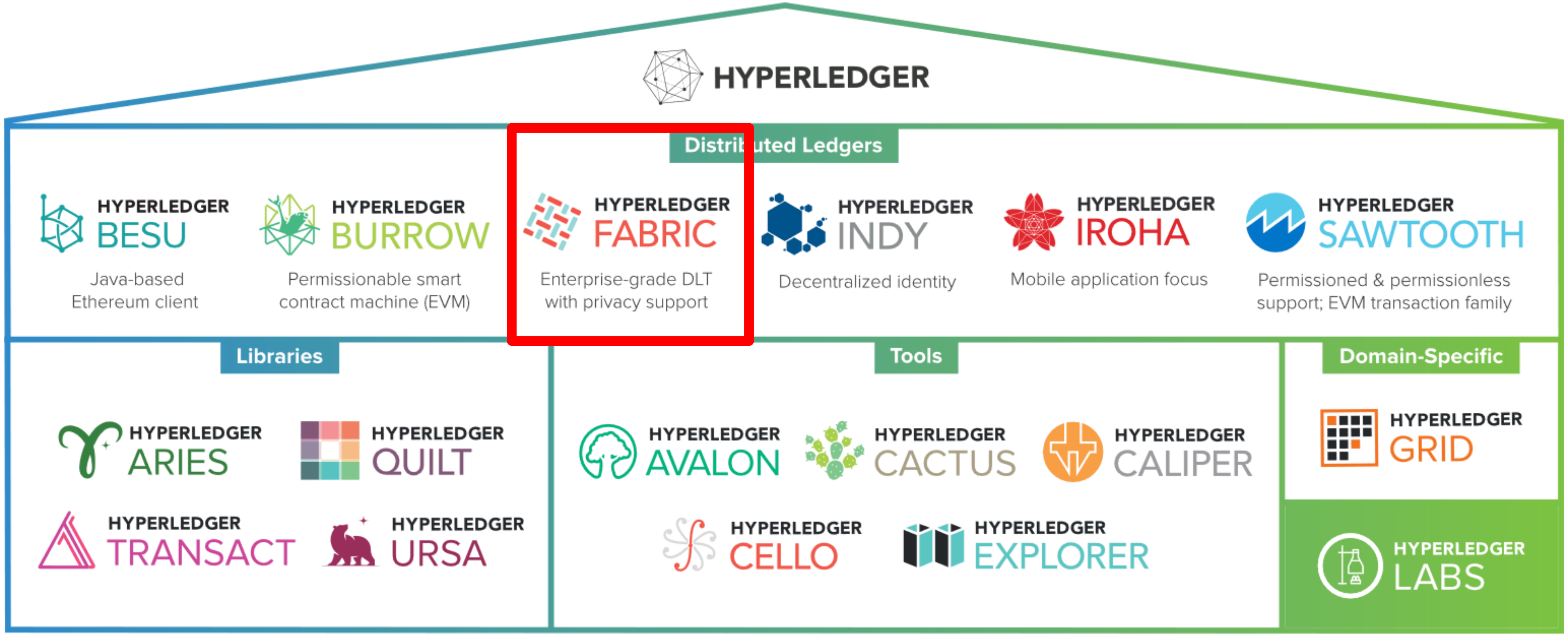
HYPERLEDGER EXPLORER

Domain-Specific

HYPERLEDGER GRID

HYPERLEDGER LABS

HYPERLEDGER



HYPERLEDGER

Distributed Ledgers

- HYPERLEDGER BESU: Java-based Ethereum client
- HYPERLEDGER BURROW: Permissionable smart contract machine (EVM)
- HYPERLEDGER FABRIC: Enterprise-grade DLT with privacy support
- HYPERLEDGER INDY: Decentralized identity
- HYPERLEDGER IROHA: Mobile application focus
- HYPERLEDGER SAWTOOTH: Permissioned & permissionless support; EVM transaction family

Libraries

- HYPERLEDGER ARIES
- HYPERLEDGER QUILT
- HYPERLEDGER TRANSACT
- HYPERLEDGER URSA

Tools

- HYPERLEDGER AVALON
- HYPERLEDGER CACTUS
- HYPERLEDGER CALIPER
- HYPERLEDGER CELLO
- HYPERLEDGER EXPLORER

Domain-Specific

- HYPERLEDGER GRID

HYPERLEDGER LABS

HYPERLEDGER FABRIC

Benefits of Hyperledger Fabric

Permissioned network
Establish decentralized trust in a network of known participants rather than an open network of anonymous participants.

Confidential transactions
Expose only the data you want to share to the parties you want to share it with.

Pluggable architecture
Tailor the blockchain to industry needs with a pluggable architecture rather than a one-size-fits-all approach.

Easy to get started
Program smart contracts in the languages your team works in today, instead of learning custom languages and architectures.

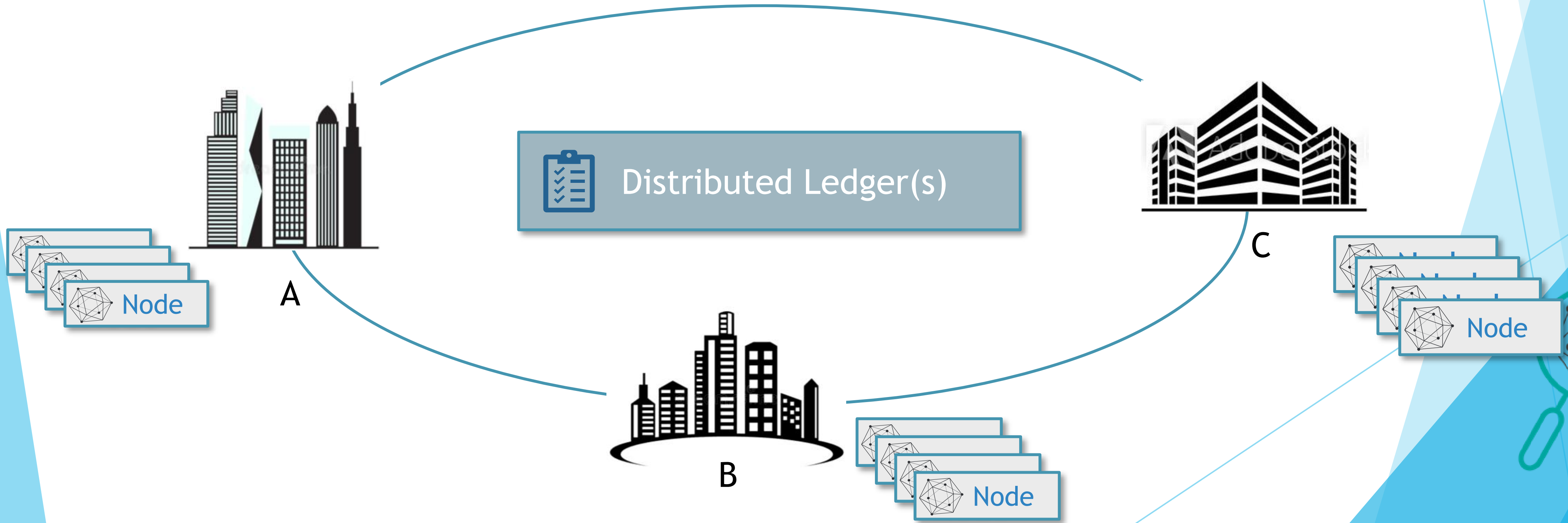
HYPERLEDGER
FABRIC

A Distributed Ledger Technology (DLT) framework for building Business Blockchain Applications.

HYPERLEDGER FABRIC

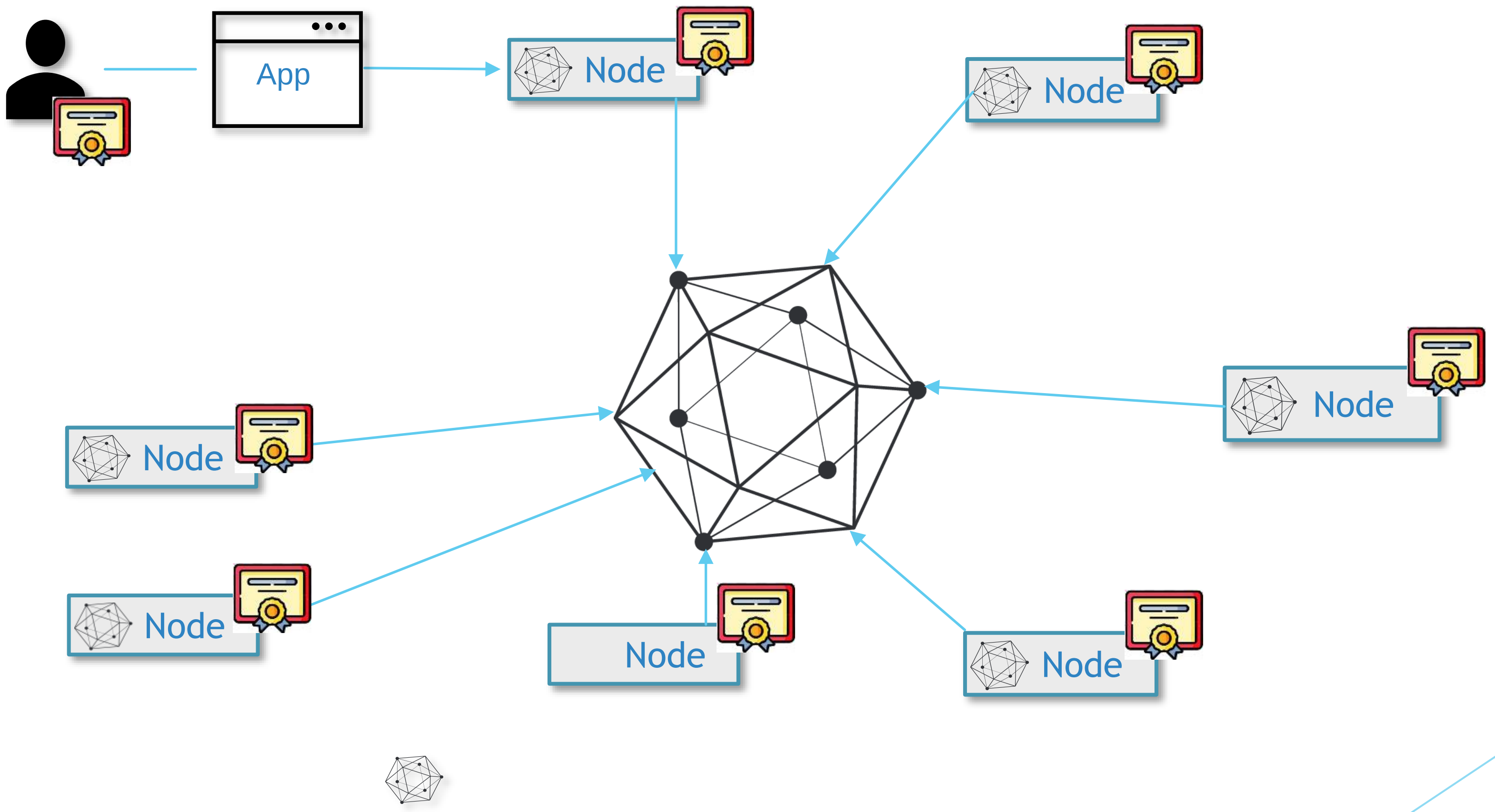
Hyperledger Architecture Component

Members = Legally separate entities



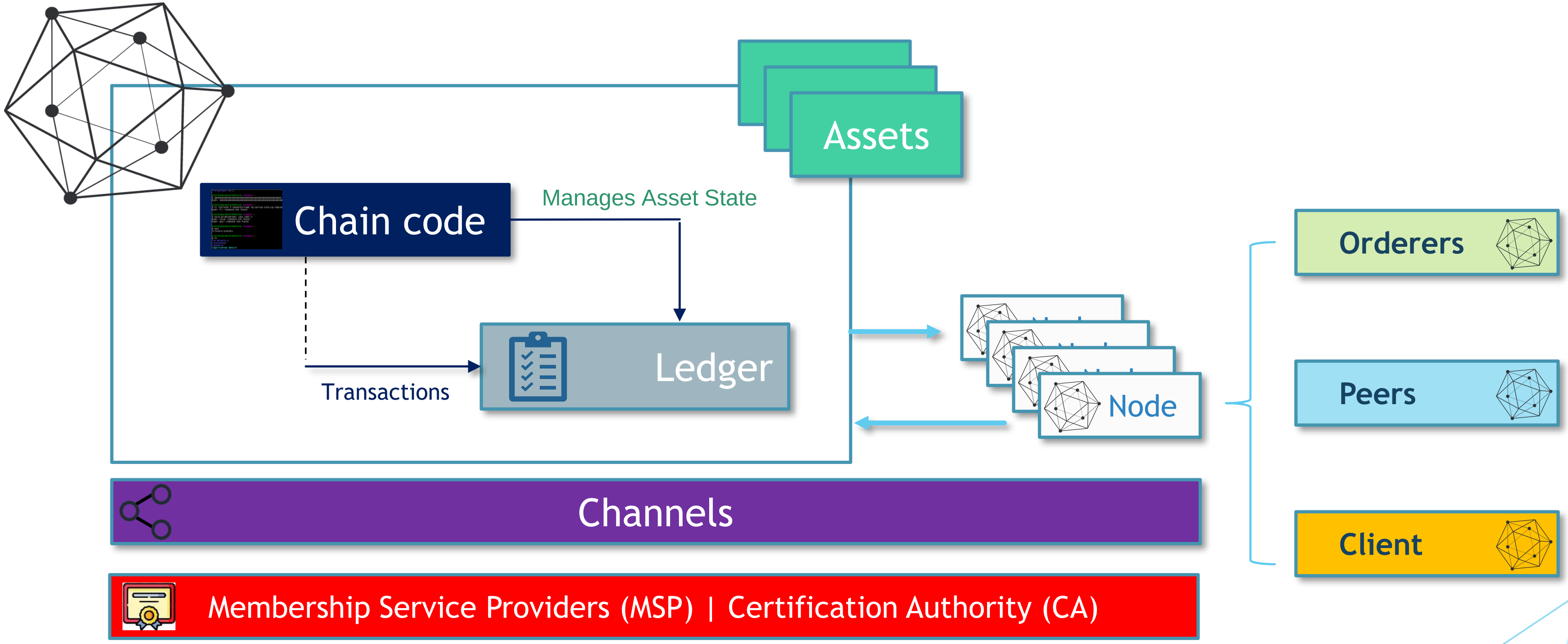
HYPERLEDGER FABRIC

Hyperledger Architecture Component



HYPERLEDGER FABRIC

Hyperledger Architecture Component



Assets

Chain code

Manages Asset State

Ledger

Transactions

Node

Orderers

Peers

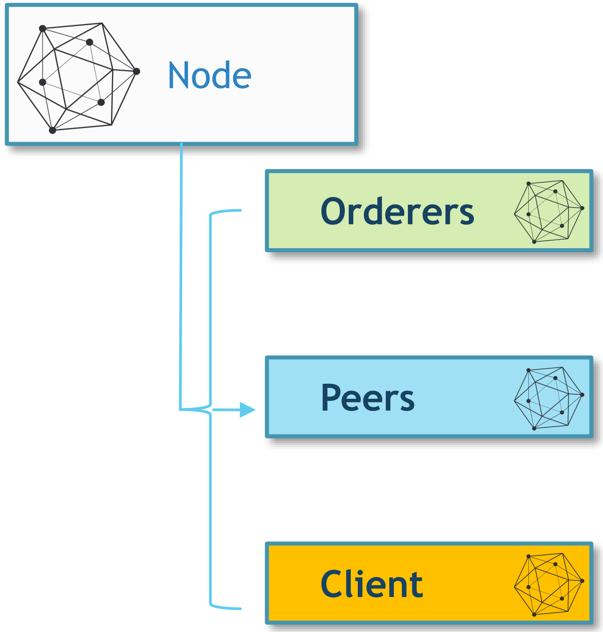
Client

Channels

Membership Service Providers (MSP) | Certification Authority (CA)

HYPERLEDGER
FABRIC

Hyperledger Architecture Component



- Leader peer
- Anchor peer

Communication channel of Fabric

→

Only nodes known outside the organization

Use SDK for the user's endpoints

HYPERLEDGER FABRIC

Hyperledger Architecture Component

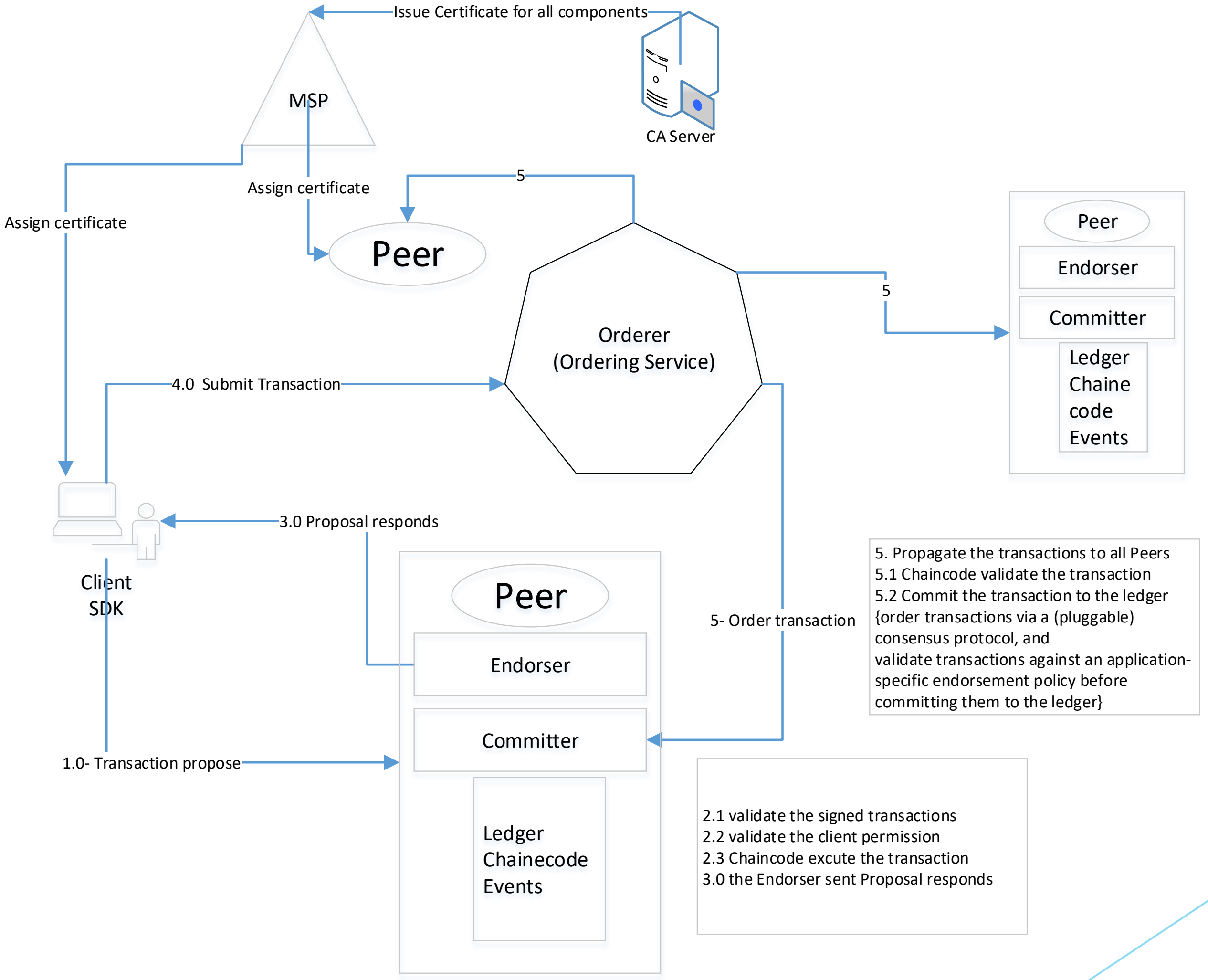
Orderers

Communication channel of Fabric

- Ordering Service
- Responsible for consistent ledger state across the network
  - Consensus mechanism
  - Ensures order of transactions
- Creates the blocks & guarantees atomic delivery

HYPERLEDGER
FABRIC

Hyperledger Architecture Component



5. Propagate the transactions to all Peers
5.1 Chaincode validate the transaction
5.2 Commit the transaction to the ledger
{order transactions via a (pluggable)
consensus protocol, and
validate transactions against an application-specific endorsement policy before
committing them to the ledger}

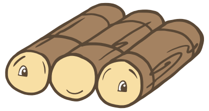
2.1 validate the signed transactions
2.2 validate the client permission
2.3 Chaincode excute the transaction
3.0 the Endorser sent Proposal responds

HYPERLEDGER
FABRIC

Hyperledger Architecture Component

Orderers

Implemented with Message Oriented Middleware



The Raft Consensus Algorithm

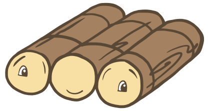
- Diego Ongaro and John Ousterhout, Stanford University (2014) “In Search of an Understandable Consensus Algorithm”
- Managing a replicated log.
- Crash fault-tolerance (CFT)
- Quorum = 1/2N
- It produces a result equivalent to (multi-)Paxos
- Raft is easier for students to learn than Paxos.
  - Leslie Lamport “The part-time parliament” (1989)
    “Paxos made simple” (2001)
- RAFT & Paxos are non-Byzantine model

HYPERLEDGER
FABRIC

Hyperledger Architecture Component

Orderers

Implemented with Message Oriented Middleware



The Raft Consensus Algorithm

- Diego Ongaro and John Ousterhout, Stanford University (2014)
  “In Search of an Understandable Consensus Algorithm”

- Demo: *http://thesecretlivesofdata.com/raft/*

- Demo the node rest on Raft website:

https://raft.github.io/

Byzantine Fault Tolerance

Byzantine Generals’ Problem

"The Byzantine generals problem“
Leslie Lamport, Robert Shostak, and Marshall Pease.(1982)

- System with x Byzantine nodes
  3x + 1 total nodes in order to reach consensus
- Potential traitor generals = Byzantine Nodes
- There is no solution in the present of 1/3 or grater percentage potential traitor generals.

The algorithms demonstrated in this paper are only designed to work in a synchronous environment.

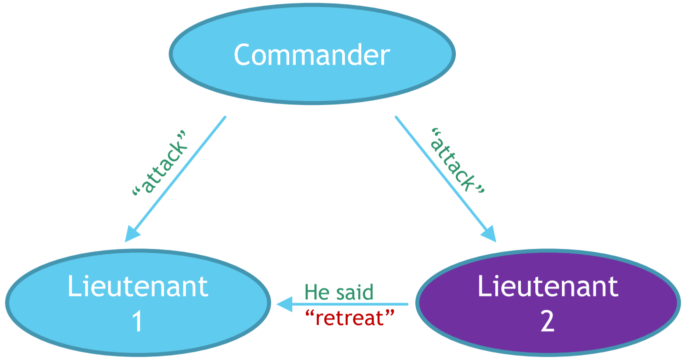


Fig.1 Lieutenant 2 a traitor

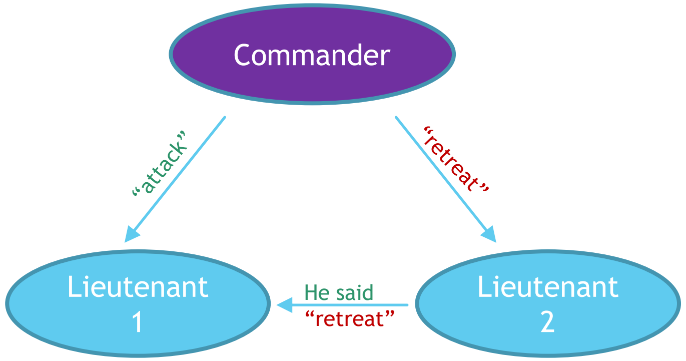


Fig.2 The Commander a traitor

Distributed System Communicate

Message passing

- By “message passing” between one or more other nodes
- Messaging protocol, HTTP, RPC, or a custom protocol.

Synchronous → messages will be delivered within some fixed time

Asynchronous → network may delay messages infinitely

(**DLS** and **PBFT**) That brought us closer than ever before to breaking the Byzantine + asynchronous barrier.

PBFT

Practical byzantine fault tolerance

"Practical byzantine fault tolerance." By Miguel Castro, and Barbara Liskov. (1999)

- Handle f Byzantine faults in a system with
  3f + 1 nodes
- Quorum over 2/3 voters.
- Main PBFT algorithm consists of three phases: pre-prepare, prepare, and commit.

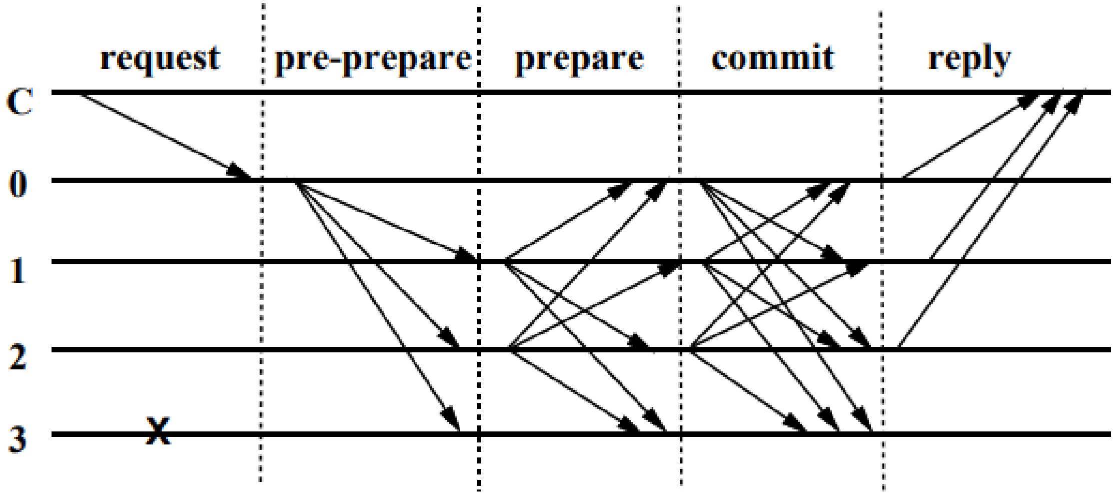


Figure 1: Normal Case Operation

DLS

The DLS Algorithm

“Consensus in the Presence of Partial Synchrony” by Dwork, Lynch, and Stockmeyer (1988)

- The first known asynchronous Byzantine Consensus solution
- Partial synchrony lies somewhere between
  Synchronous & Asynchronous.

Two versions of the partial synchrony assumption

1. Assume that fixed bounds exist for how long messages take to get delivered. But they are not known a priori.
2. Assume the upper bounds for message delivery are known, but they’re only guaranteed to hold starting at some unknown time (also called “*Global Stabilization Time*,” GST).

A series of rounds are divided into “trying” and “lock-release” phases.

Liveness is the property of the system continuing to work in case of failures.
Safety is the agreement of the network on a single state.

BDLS

BFT protocols in partial synchronous networks

"Byzantine Fault Tolerance in Partial Synchronous Networks." Wang, Yongge (2020).

- BDLS consensus based on DLS protocol algorithm.
- Able to achieve consensus with both reduced round complexity and reduced communication complexity.

PBFT
Mesh communication network

BDLS
Star networks

HotStuff using threshold cryptography. → Facebook's LibraBFT protocol

- Best existing linear communication/ authenticator complexity protocols require at least 7 steps to achieve agreement.

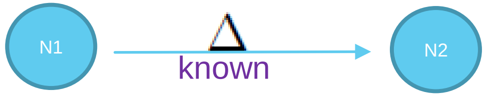
VS

- BDLS participants could reach agreement in 4 steps with linear communication/authenticator complexity to achieve agreement.

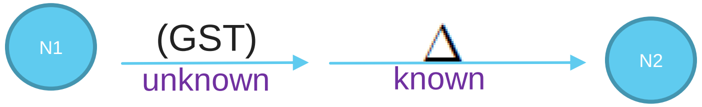
BDLS

BFT protocols in partial synchronous networks

Type I $\Delta < \infty$ is unknown.



Type II $\Delta < \infty$ holds eventually. participant knows the value of Δ
But this only holds after an unknown time slot Global Stabilization Time (GST).



BDLS is proved to be secure in Type II partial synchronous networks

Attacks against several widely deployed BFT protocols: SUCH AS:

- PBFT
- Tendermint BFT
- Casper FFG

Participants would reach a deadlock before GST and the deadlock could not be removed after GST.

BDLS

BFT protocols in partial synchronous networks

Steps	PBFT	Tendermint BFT	HotStuff BFT	BDLS	
1	📡	📡	📡	👂	
2	☁	☁	👂	📡	
3	☁	☁	📡	👂	
4			👂	📡	
5			📡		
6			👂		
7			📡		
message complexity	$2n^2 + n$	$2n^2 + n$	*7n*	*4n*	
authenticator complexity	O(n2)	O(n)	O(n)	O(n)	

📡 : Leader broadcasts

👂 : All participants send messages to the leader

☁ : All participants broadcast

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Hyperledger Goals

Create enterprise grade, open source, distributed ledger frameworks and code bases to support business transactions

Provide neutral, open, and community-driven infrastructure supported by technical and business governance

Build technical communities to develop blockchain and shared ledger POCs, use cases, field trails and deployments

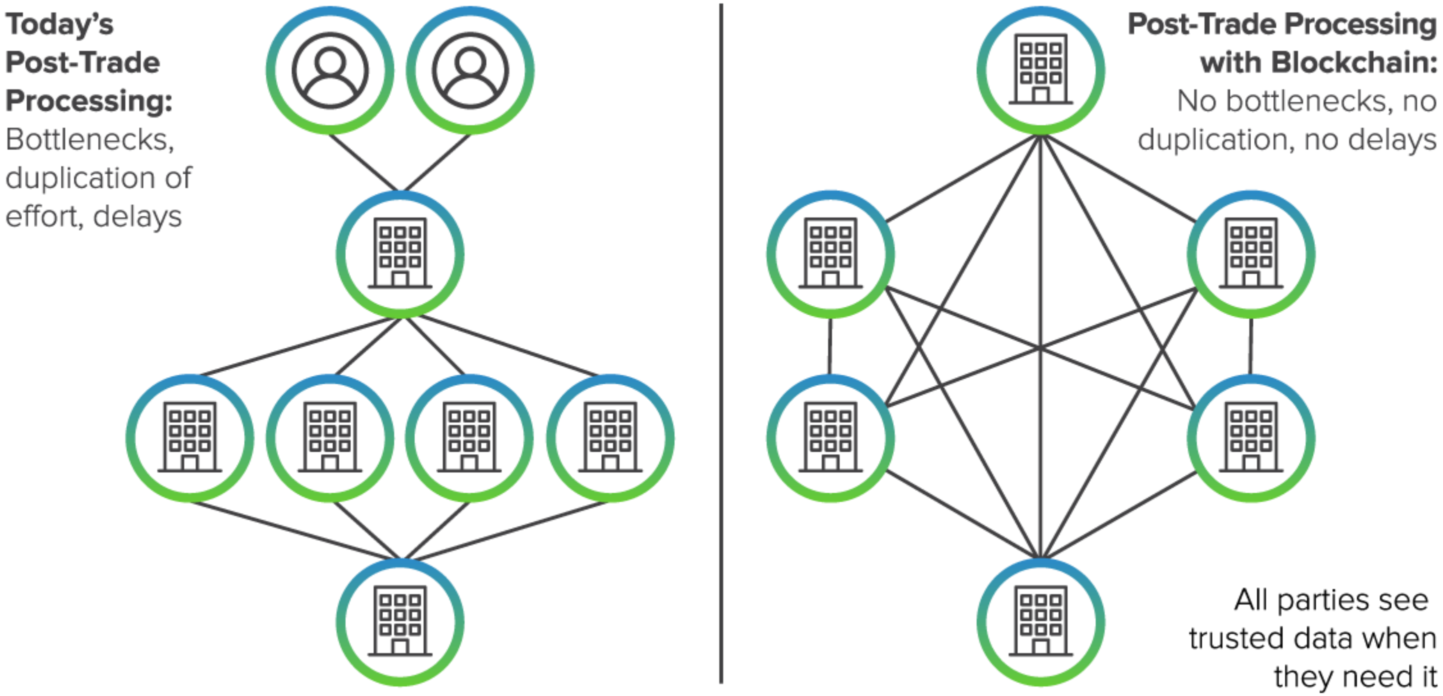
Educate the public about the market opportunity for blockchain technology

Promote our community of communities taking a toolkit approach with many platforms and frameworks

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Financial Services

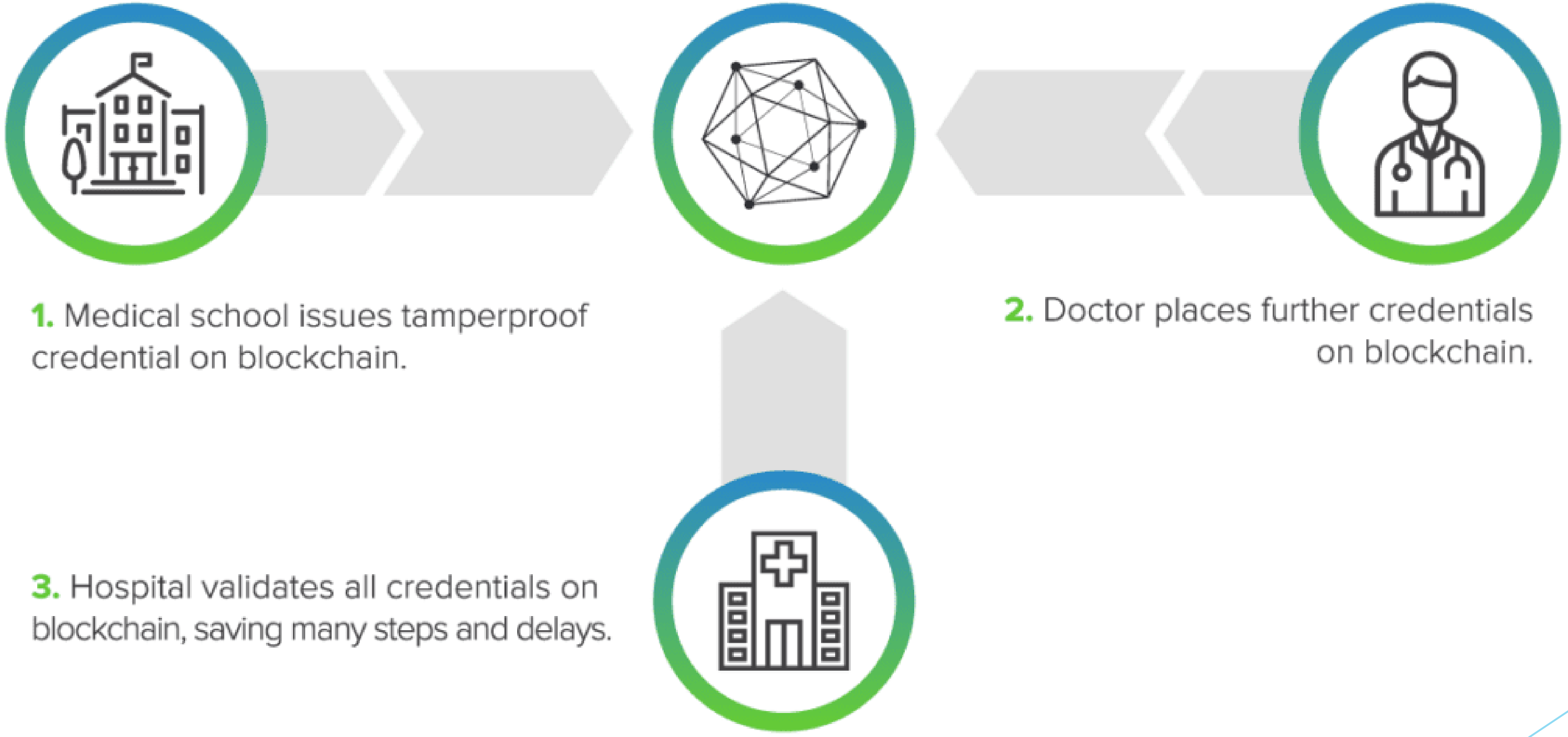


Today's Post-Trade Processing: Bottlenecks, duplication of effort, delays

Post-Trade Processing with Blockchain: No bottlenecks, no duplication, no delays

All parties see trusted data when they need it

Healthcare



1. Medical school issues tamperproof credential on blockchain.

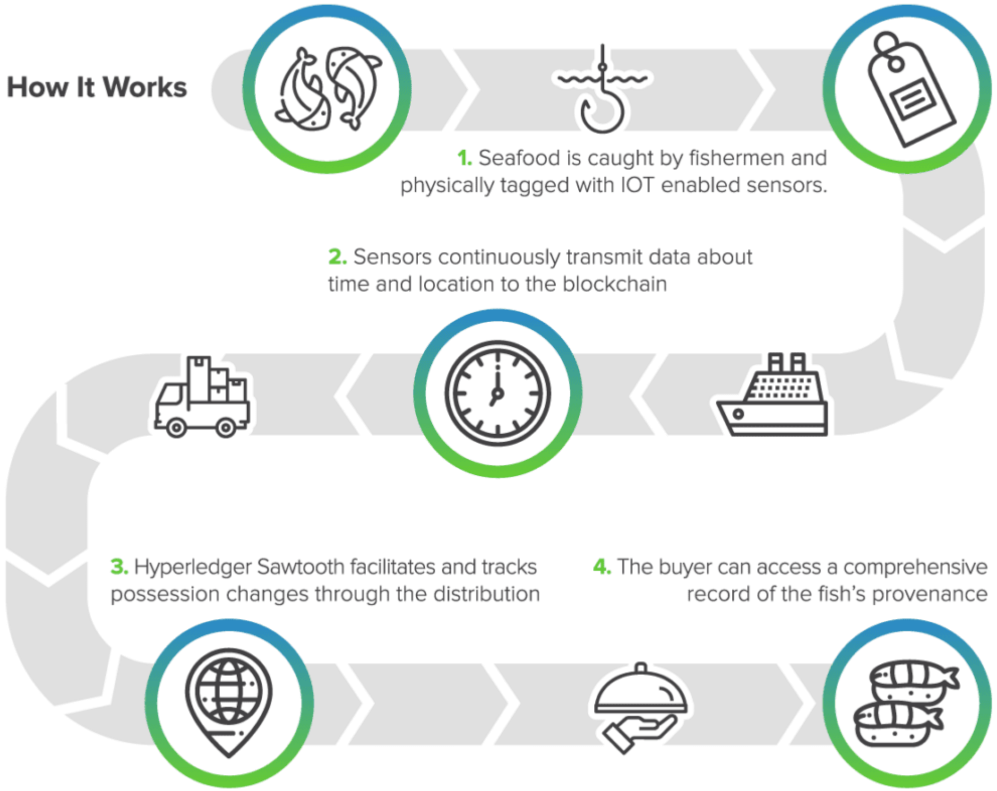
2. Doctor places further credentials on blockchain.

3. Hospital validates all credentials on blockchain, saving many steps and delays.

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Supply chain



How It Works

1. Seafood is caught by fishermen and physically tagged with IOT enabled sensors.

2. Sensors continuously transmit data about time and location to the blockchain

3. Hyperledger Sawtooth facilitates and tracks possession changes through the distribution

4. The buyer can access a comprehensive record of the fish's provenance

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Thanks

Q/A ?

HYPERLEDGER
FABRIC