Hyperledger – Introduction

Gallersdörfer, U., Holl, P., & Matthes, F. (2019). "Blockchain-based Systems Engineering“. Lecture Slides. TU Munich.

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

1. Introduction
   - Terminology
   - Motivation
   - Use Cases
   - Consortium Blockchain projects
   - Corda

2. Hyperledger
   - History
   - Mission
   - Projects

3. Fabric
   - Architecture
   - Membership service provider
   - Application
   - Ordering service
   - Peers
   - Chaincode

Terminology

Public Blockchain
The network is completely open for anyone who wants to participate. For reaching consensus on the current state of the ledger, the participants must agree up on a certain mechanism. Currently, a highly used consensus mechanism is Proof-of-Work (PoW). Network participants are usually financially incentivized by an implemented currency, e.g. Ether or Bitcoin.

Examples
Ethereum, Monero, Bitcoin, Litecoin

Private Blockchain
The access to the network is limited to a certain party. The operator of the network writes and verifies transactions. Therefore, private Blockchains usually achieve a much higher throughput rate compared to public chains. Private Blockchains are a very centralized approach to Blockchains and use cases have yet to be found were such solutions are superior to conventional systems.

Examples
Monax, MultiChain, C-Chain, Amazon Quantum Ledger

Source: Pilkington, Marc. "11 Blockchain technology: principles and applications." *Research handbook on digital transformations*(2016): 225.

IBM: The difference between public and private Blockchain

Terminology cont.

Consortium Blockchains

Consortium Blockchains are a hybrid of public and private Blockchains. The access to the network is limited to certain parties and participation usually requires an invitation by the starter of the network. Furthermore, reaching consensus is usually achieved differently as in public networks. The operators of the network write and verify transactions. Consortium Blockchains are used as a trustless data structure shared among different institutions.

Examples

Hyperledger, Corda

Difference to solely private chains according to Vitalik Buterin:

„So far there has been little emphasis on the distinction between consortium Blockchains and fully private Blockchains, although it is important: the former provides a hybrid between the 'low-trust' provided by public Blockchains and the 'single highly-trusted entity' model of private Blockchains, whereas the latter can be more accurately described as a traditional centralized system with a degree of cryptographic auditability attached" [1]

[1] Vitalik Buterin: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Terminology cont.

Sometimes there is also a distinction between **permissioned** and **permission-less**. **Private** and **consortium** Blockchains are **always permissioned** by definition. However, there are Blockchains that are considered **permissioned and public**, i.e., nodes can only enter the network if permitted by the network operators. On the other hand, the network can be used by anyone and the data of the ledger is also publicly accessible.

Example for a public and permissioned Blockchain:
Ripple

Motivation for consortium Blockchains

The problem
Multiple parties that do not fully trust each other need to share information in a tamper proof and secure way.

Challenges
Having a cost effective and secure data structure.

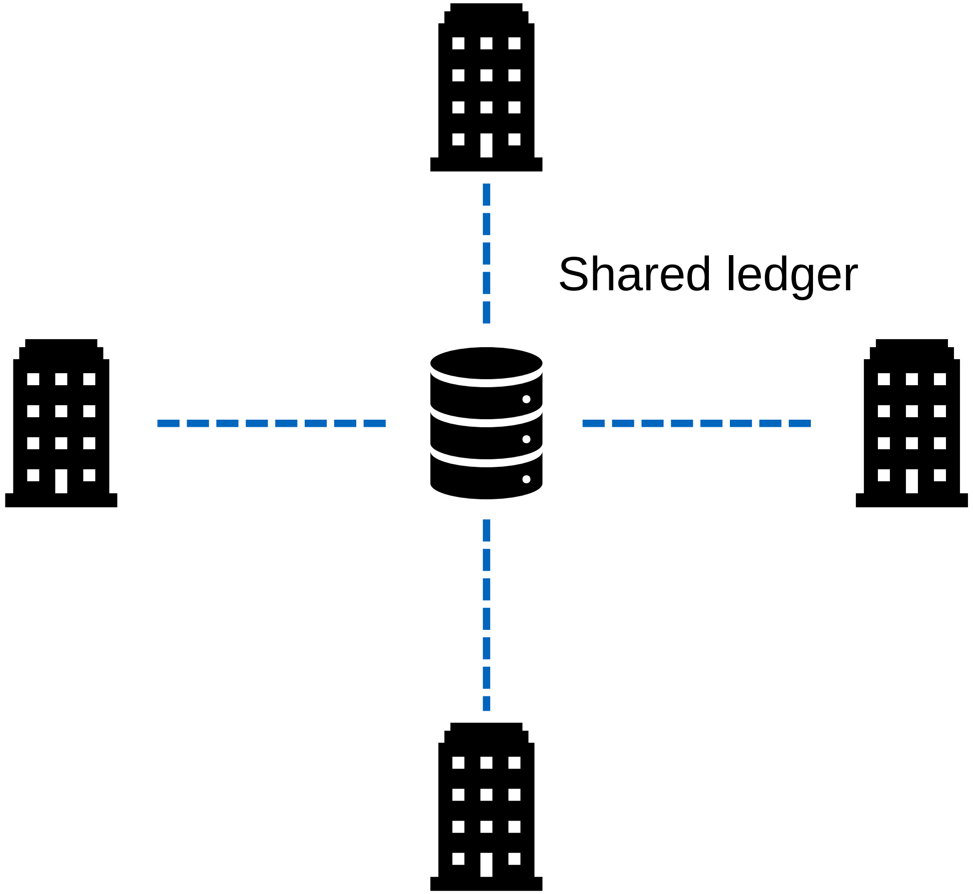
Who owns and operates the ledger?

Who pays for the shared data structure?

What are allowed operations on the shared data structure?

What is the process of adding new parties?

How is consensus achieved?



Business requirements for blockchain-based systems

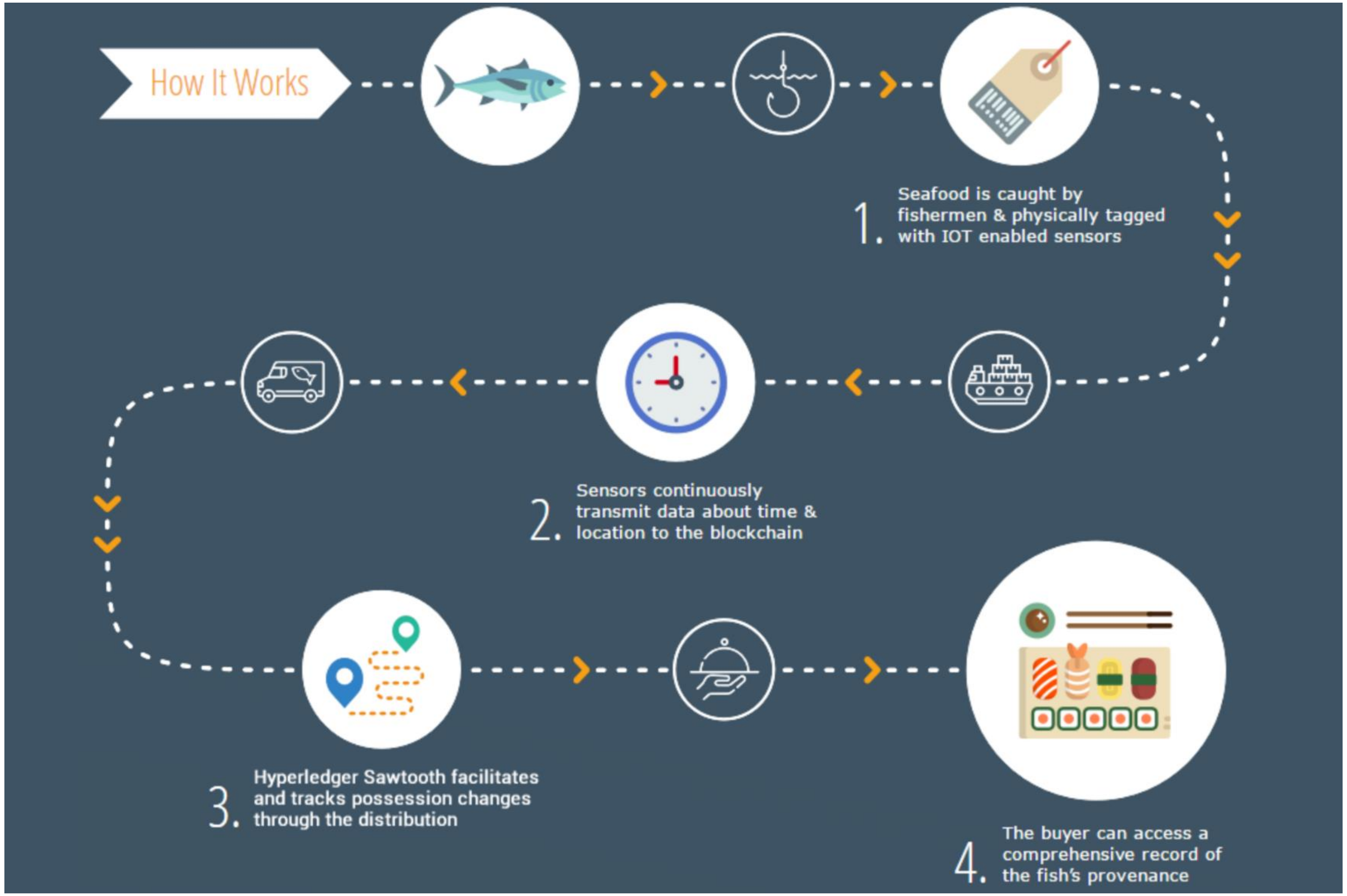
Privacy
The network must be secured via authentication and authorization mechanisms. Only involved parties should be able to issue transactions and participate in the network.

Trust
All transactions that are issued on the network must be verified and valid to a set of rules on which the participating parties agreed on.

Smart Contracts
The platform should support the execution of business logic on which the participating parties have agreed on.

Single source of truth
An immutable ledger that keeps track of all transactions. Participating parties must be accountable for transactions they issue on the network.

Example: Supply chain of a fish market


How It Works
1. Seafood is caught by fishermen & physically tagged with IOT enabled sensors
2. Sensors continuously transmit data about time & 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

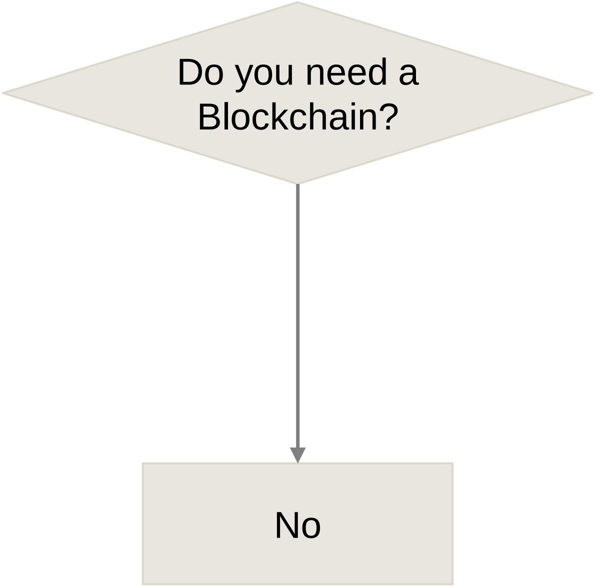
Parties must enter verified and valid information!

Source: https://www.hyperledger.org/projects/sawtooth/seafood-case-study

When to use a Blockchain

Blockchain is one of the most hyped technology trends nowadays and many enterprises are trying to leverage the technology. However, for finding a use case for a blockchain-based solution it is necessary to understand the technology and the use case well.

Therefore, several individuals and institutions came up with different models for making a decision whether a blockchain-based solution makes sense or not.

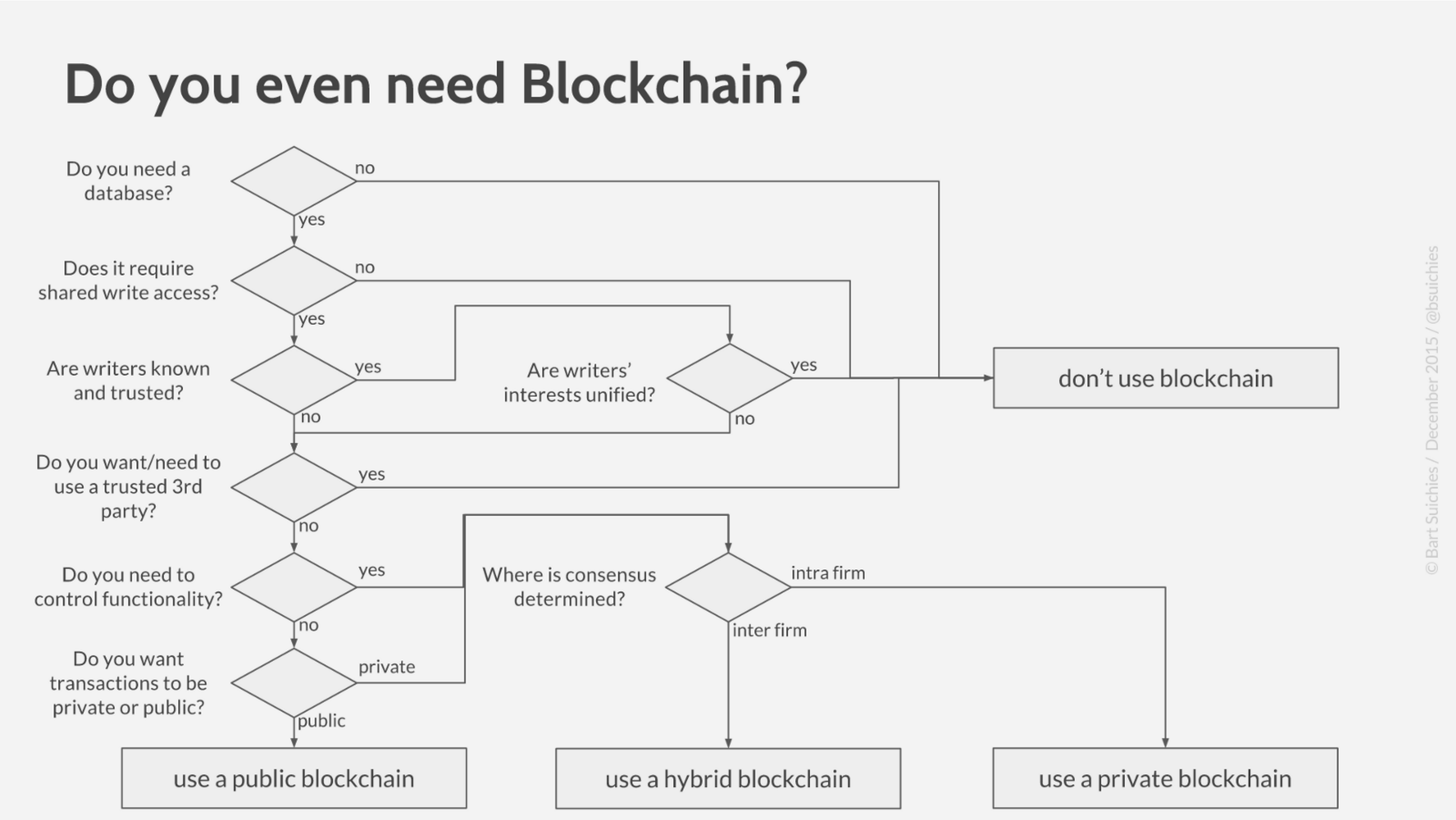


Do you need a Blockchain?

No

(Joke) Model by Dave Birch (https://twitter.com/dgwbirch?lang=de)

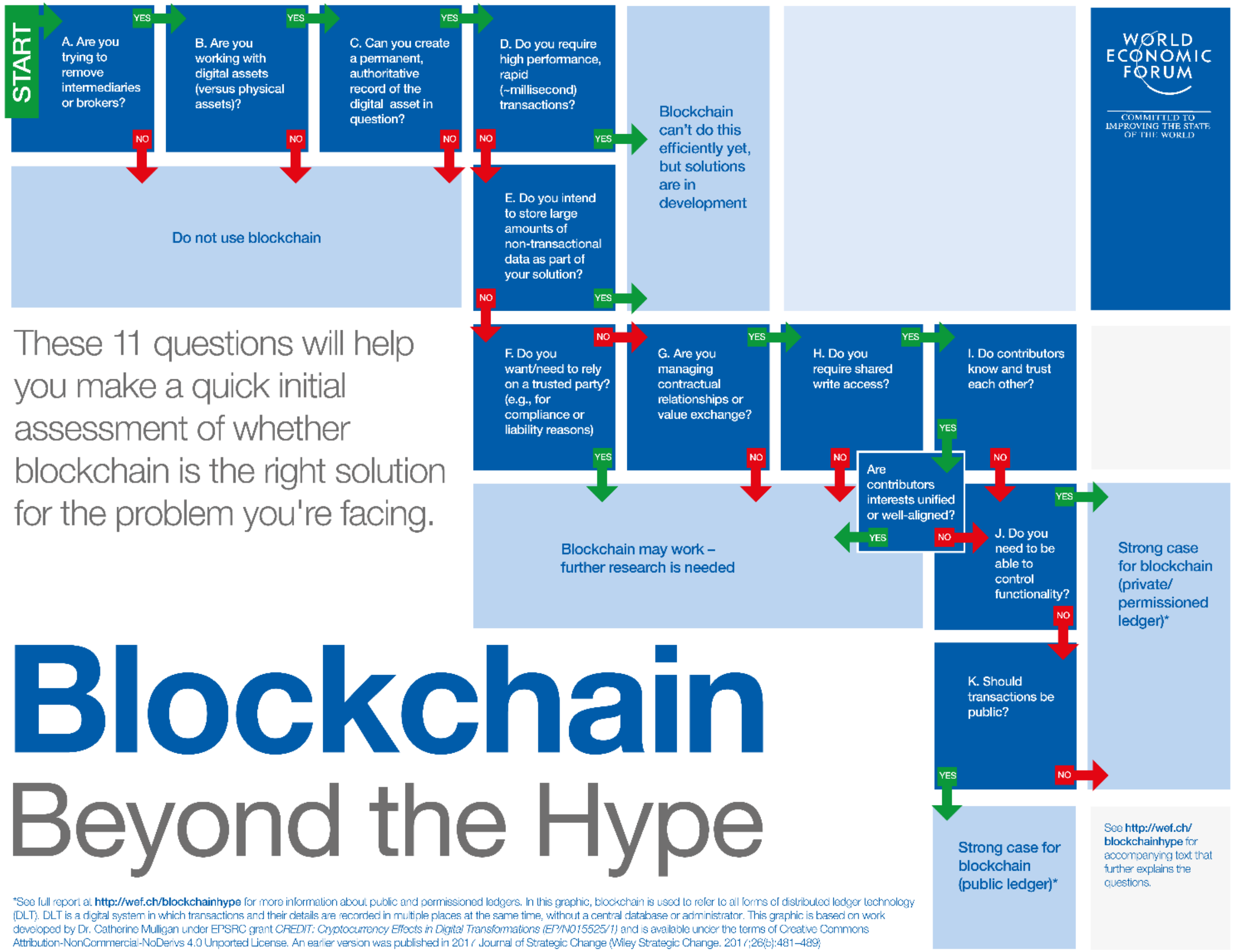
Bart Suichies model



Do you even need Blockchain?
Do you need a database?
no
yes
Does it require shared write access?
no
yes
Are writers known and trusted?
yes
no
Are writers' interests unified?
yes
no
don't use blockchain
Do you want/need to use a trusted 3rd party?
yes
no
Do you need to control functionality?
yes
no
Where is consensus determined?
intra firm
inter firm
Do you want transactions to be private or public?
private
public
use a public blockchain
use a hybrid blockchain
use a private blockchain
© Bart Suichies / December 2015 / @bsuichies

Source: https://medium.com/@bsuichies/why-blockchain-must-die-in-2016-e992774c03b4

Model by the World Economic Forum



A. Are you trying to remove intermediaries or brokers?

B. Are you working with digital assets (versus physical assets)?

C. Can you create a permanent, authoritative record of the digital asset in question?

D. Do you require high performance, rapid (~millisecond) transactions?

Blockchain can't do this efficiently yet, but solutions are in development

Do not use blockchain

E. Do you intend to store large amounts of non-transactional data as part of your solution?

These 11 questions will help you make a quick initial assessment of whether blockchain is the right solution for the problem you're facing.

F. Do you want/need to rely on a trusted party? (e.g., for compliance or liability reasons)

G. Are you managing contractual relationships or value exchange?

H. Do you require shared write access?

I. Do contributors know and trust each other?

Are contributors interests unified or well-aligned?

Blockchain may work – further research is needed

J. Do you need to be able to control functionality?

Strong case for blockchain (private/ permissioned ledger)*

K. Should transactions be public?

Blockchain

Beyond the Hype

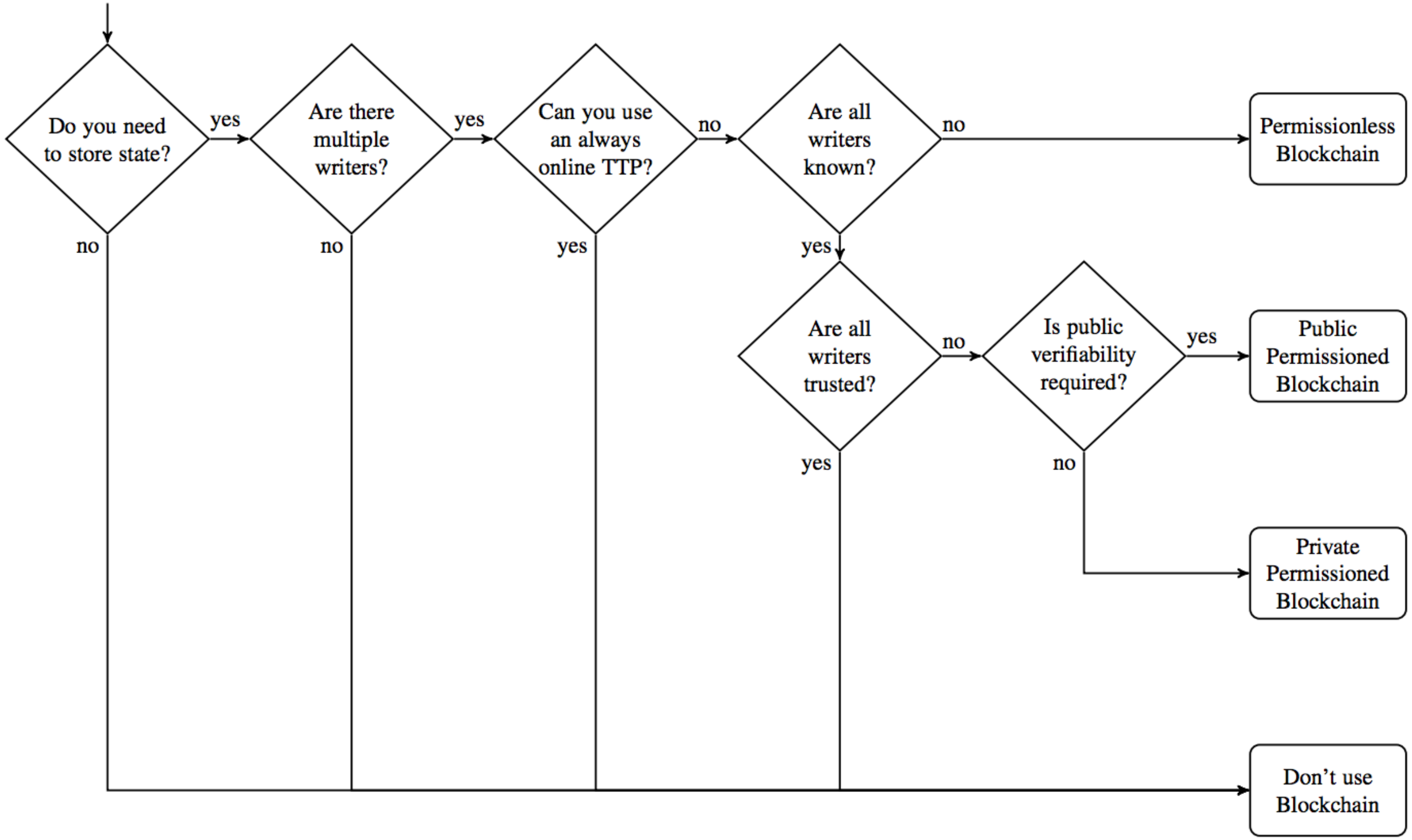
Strong case for blockchain (public ledger)*

See **http://wef.ch/blockchainhype** for accompanying text that further explains the questions.

*See full report at **http://wef.ch/blockchainhype** for more information about public and permissioned ledgers. In this graphic, blockchain is used to refer to all forms of distributed ledger technology (DLT). DLT is a digital system in which transactions and their details are recorded in multiple places at the same time, without a central database or administrator. This graphic is based on work developed by Dr. Catherine Mulligan under EPSRC grant *CREDIT: Cryptocurrency Effects in Digital Transfomations (EP/N015525/1)* and is available under the terms of Creative Commons Attribution-NonCommercial-NoDerivs 4.0 Unported License. An earlier version was published in 2017 Journal of Strategic Change (Wiley Strategic Change. 2017;26(5):481–489)

Source: https://www.weforum.org/agenda/2018/04/questions-blockchain-toolkit-right-for-business

Model by Karl Wüst & Arthur Gervais



Do you need to store state?
yes
no
Are there multiple writers?
yes
no
Can you use an always online TTP?
no
yes
Are all writers known?
no
yes
Permissionless Blockchain
Are all writers trusted?
no
yes
Is public verifiability required?
yes
no
Public Permissioned Blockchain
Private Permissioned Blockchain
Don’t use Blockchain

Wüst, Karl, and Arthur Gervais. "Do you need a Blockchain?." *IACR Cryptology ePrint Archive* 2017 (2017): 375.

Hyperledger & Corda: Consortium DLT projects

HYPERLEDGER

Hyperledger is an umbrella project hosted by the Linux foundation. The goal is to create open source software that enables institutions to set up a shared and trustless ledger database for collaboration. The most prominent project is Fabric which can be used to set up a permissioned Blockchain infrastructure with the capability of executing smart contracts. Fabric was originally initiated by IBM and Digital Asset.

corda

Corda is an open source project initiated by the R3 consortium. Similar to Fabric, Corda is used to set up the infrastructure for a permissioned network. Corda is developed in Kotlin and specifically designed for enterprise use cases. The system supports smart contracts that need to be written in Java or any other language that compiles to JVM bytecode.

Corda

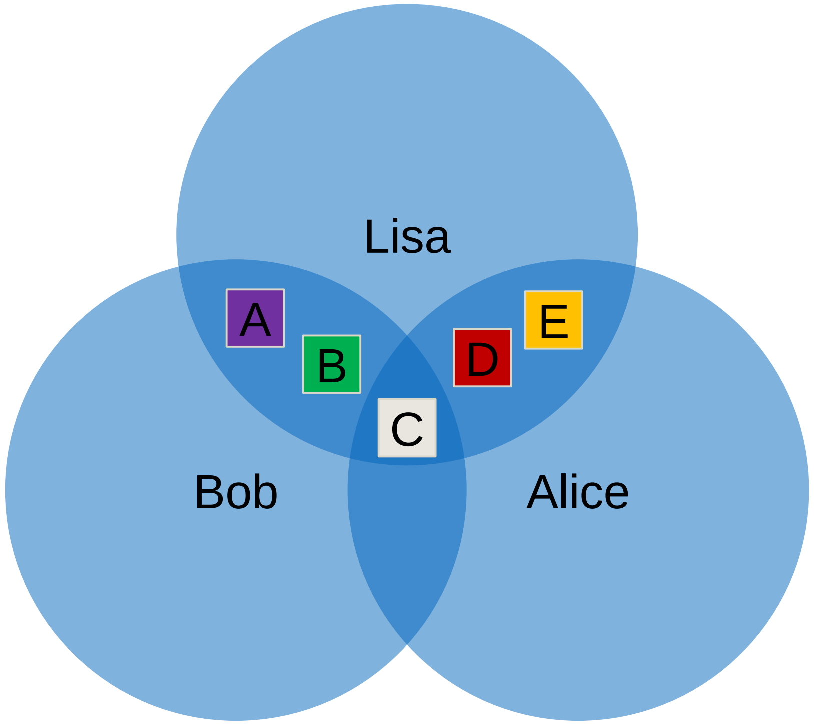
Corda is a distributed ledger platform but it is technically not a blockchain-based system. Corda is designed to be used in an isolated environment and therefore comes with several features that significantly differs from public Blockchain systems like Ethereum and Bitcoin.

Features

- **Not** based on **a Blockchain ➔ no mining**
- Has **no built-in coin** or token, i.e. cryptocurrency
- **No central ledger** that is shared by all participants
- Based on a **permissioned network** where only authorized peers can join
- **Transactions are not broadcasted** throughout the whole network
  - **Messages** are **directly sent** to a specified set of receivers
  - Peers in the network only see what they need to see / are allowed to see
- Is completely **JVM-based** and the core is written in Kotlin
- No “real” smart contracts. In Corda the term **smart contract** is used for a **pure function that verifies a transaction**.

Corda ledger

Corda uses individual “vaults” between peers instead of one central ledger like in usual blockchain-based systems like Ethereum or Bitcoin. A vault is an accessible subset of information in the network. Technically, the vault is a SQL database including a table of facts.



Ledger

All information in the network = A, B, C, D, E

Lisa has access to information: A, B, C, D, E
A,B,C is shared with Bob and C,D,E is shared with Alice

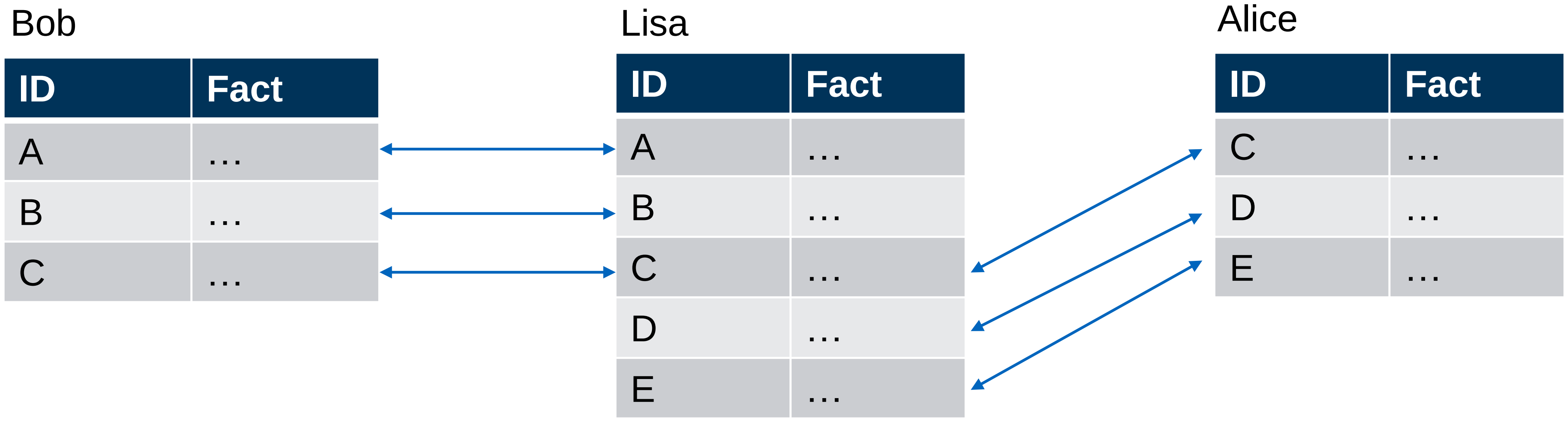
Bob has access to information : A, B, C
A,B,C is shared with Lisa and C is shared with Alice

Alice has access to information : C, D, E
C is shared with Bob and C, D, E is shared with Bob

Corda vaults

The vault is a set of facts and maintained by each peer on it’s own. There is no central database that contains all facts. The complete ledger, i.e. containing all facts, is the union of all vaults.

Technically, a vault is a SQL table that looks similar to the following:



Bob

ID	Fact
A	…
B	…
C	…

Lisa

ID	Fact
A	…
B	…
C	…
D	…
E	…

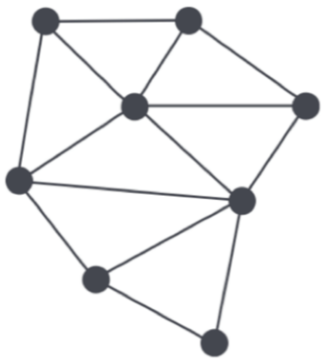
Alice

ID	Fact
C	…
D	…
E	…

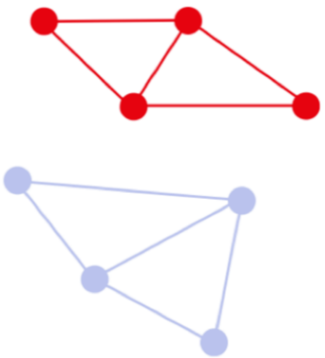
If the IDs are equal, the information must be the same on both tables

Corda Network – Vision: Seamless Interoperability

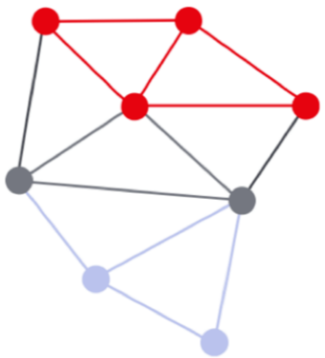
Corda aims a seamless interoperability through its “Global Corda network”: Assets gained from a particular service should be redeployed immediately without friction to pay a different trading partner utilizing a different application.



Public Blockchains



1st gen. Permissioned Blockchains



Corda Platform

1st gen. permissioned Blockchains: Interoperability between different Business Networks not possible.

Corda Platform:
Standardized set of interfaces for contracts are included to maximize interoperability from a diverse range of providers.

Source: The Corda Platform - https://www.corda.net/content/corda-platform-whitepaper.pdf

Outline

1. Introduction
   - Terminology
   - Motivation
   - Use Cases
   - Consortium Blockchain projects
   - Corda

2. Hyperledger
   - History
   - Mission
   - Projects

3. Fabric
   - Architecture
   - Membership service provider
   - Application
   - Ordering service
   - Peers
   - Chaincode

History

- In December 2015, the project was announced publicly for the first time by the Linux foundation.

- Hyperledger initially launched on February 2016 as a joint open source project with 30 founding members to advance in Blockchain technology.
  - The founding members included: ABN AMRO, Accenture, ANZ Bank, Blockchain, BNY Mellon, Calastone, Cisco, CLS, CME Group, ConsenSys, Credits, The Depository Trust & Clearing Corporation (DTCC), Deutsche Börse Group, Digital Asset Holdings, Fujitsu Limited, Guardtime, Hitachi, IBM, Intel, IntellectEU, J.P. Morgan, NEC, NTT DATA, R3, Red Hat, State Street, SWIFT, Symbiont, VMware and Wells Fargo.

- IBM and Digital Asset combined their work and built the first codebase for a Blockchain infrastructure project. The project was renamed to Hyperledger Fabric.

- On July 2017, the foundation released the first major version (1.0) of Fabric

- Currently, there are 6 frameworks released under the Hyperledger foundation to set up a Blockchain infrastructure:
  - Burrow, Fabric, Iroha, Sawtooth, Indy, and GRID

Mission

“[Building] a global, cross-industry community of communities advancing business Blockchain technologies.“ [1]

The vision behind Hyperledger is that there won’t and can’t be a one-fits-all Blockchain solution. This is because of the various different use cases and requirements on the network and consensus. Based on their vision statement, the Hyperledger foundations predicts a future where multiple different public and private Blockchain solutions co-exist. Each of them tailored to specific use cases.

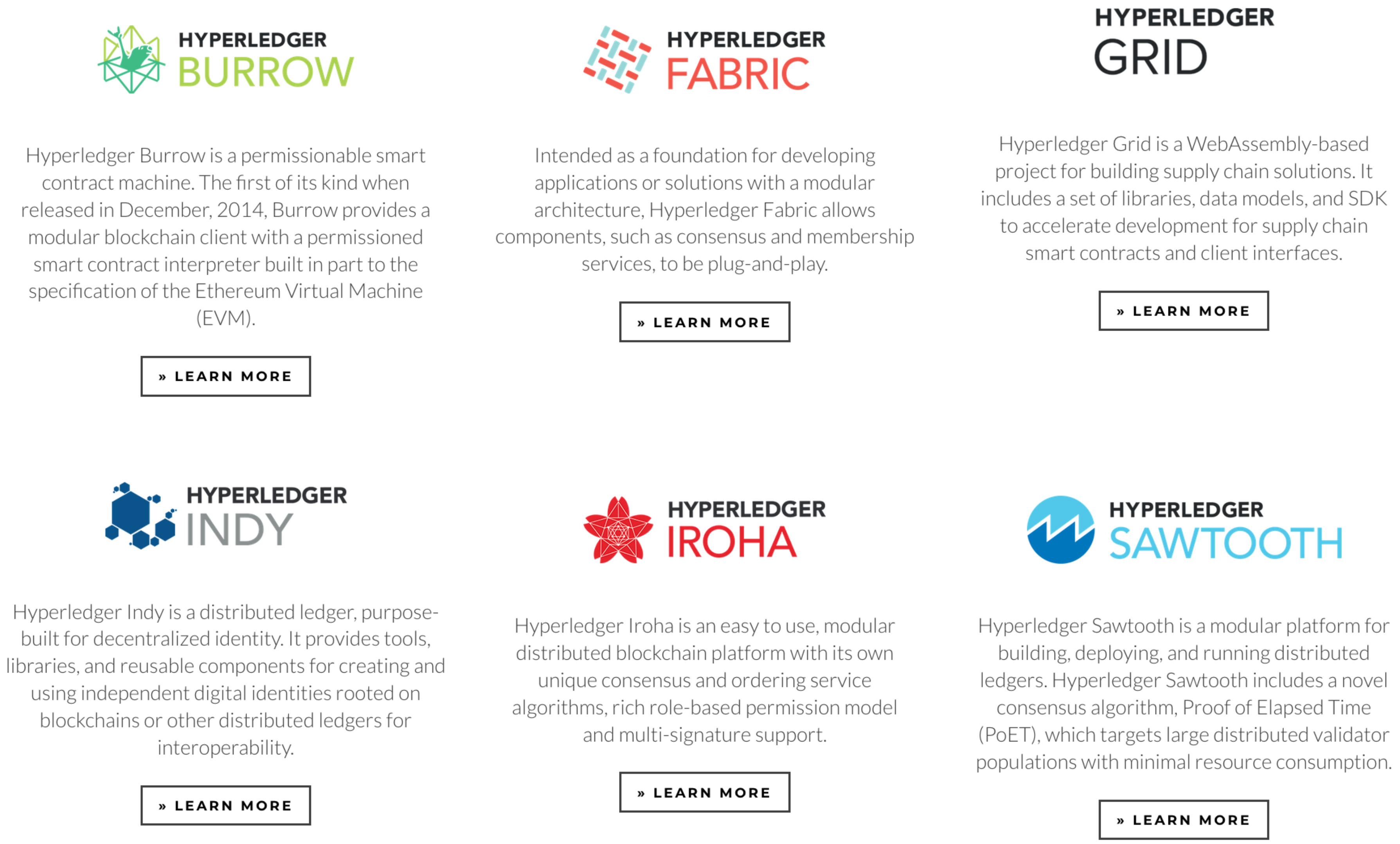
“In this environment, Hyperledger serves as a trusted source of innovative, quality-driven open source software development, creating modular components and platforms. Hyperledger is incubating and promoting enterprise-grade, open source business Blockchain technologies, on top of which anyone can set up apps to meet their business needs.” [1]

Currently, Hyperledger provides 6 frameworks to built Blockchain infrastructures and a set of tools on top of them.

[1] The Linux foundation: https://www.linuxfoundation.org/projects/case-studies/hyperledger/

Projects

Frameworks



HYPERLEDGER BURROW

Hyperledger Burrow is a permissionable smart contract machine. The first of its kind when released in December, 2014, Burrow provides a modular blockchain client with a permissioned smart contract interpreter built in part to the specification of the Ethereum Virtual Machine (EVM).

» LEARN MORE

HYPERLEDGER FABRIC

Intended as a foundation for developing applications or solutions with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play.

» LEARN MORE

HYPERLEDGER GRID

Hyperledger Grid is a WebAssembly-based project for building supply chain solutions. It includes a set of libraries, data models, and SDK to accelerate development for supply chain smart contracts and client interfaces.

» LEARN MORE

HYPERLEDGER INDY

Hyperledger Indy is a distributed ledger, purpose-built for decentralized identity. It provides tools, libraries, and reusable components for creating and using independent digital identities rooted on blockchains or other distributed ledgers for interoperability.

» LEARN MORE

HYPERLEDGER IROHA

Hyperledger Iroha is an easy to use, modular distributed blockchain platform with its own unique consensus and ordering service algorithms, rich role-based permission model and multi-signature support.

» LEARN MORE

HYPERLEDGER SAWTOOTH

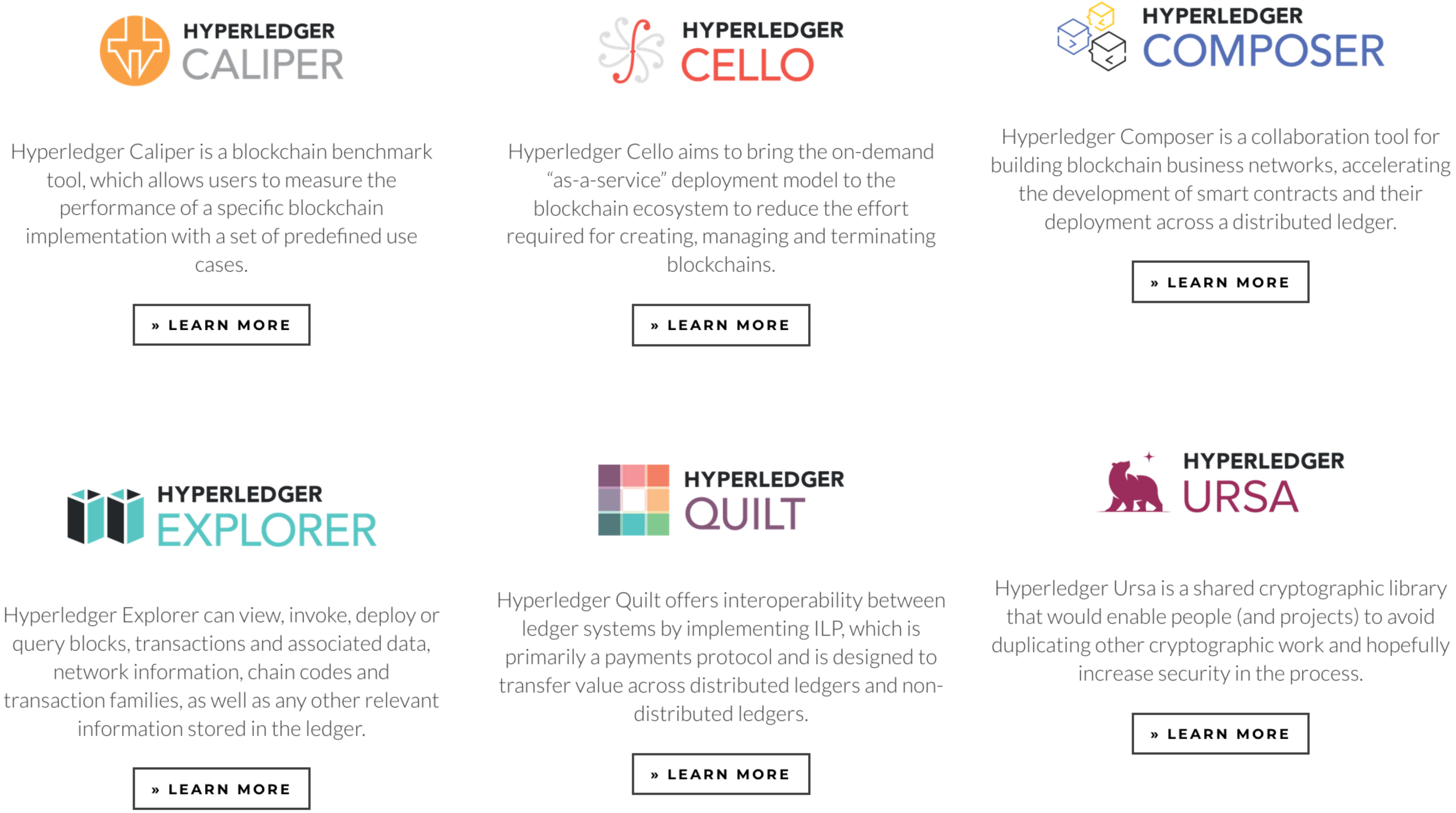
Hyperledger Sawtooth is a modular platform for building, deploying, and running distributed ledgers. Hyperledger Sawtooth includes a novel consensus algorithm, Proof of Elapsed Time (PoET), which targets large distributed validator populations with minimal resource consumption.

» LEARN MORE

Source: https://www.hyperledger.org/projects

Projects

Tools



HYPERLEDGER CALIPER

Hyperledger Caliper is a blockchain benchmark tool, which allows users to measure the performance of a specific blockchain implementation with a set of predefined use cases.

» LEARN MORE

HYPERLEDGER CELLO

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.

» LEARN MORE

HYPERLEDGER COMPOSER

Hyperledger Composer is a collaboration tool for building blockchain business networks, accelerating the development of smart contracts and their deployment across a distributed ledger.

» LEARN MORE

HYPERLEDGER EXPLORER

Hyperledger Explorer can view, invoke, deploy or query blocks, transactions and associated data, network information, chain codes and transaction families, as well as any other relevant information stored in the ledger.

» LEARN MORE

HYPERLEDGER QUILT

Hyperledger Quilt offers interoperability between ledger systems by implementing ILP, which is primarily a payments protocol and is designed to transfer value across distributed ledgers and non-distributed ledgers.

» LEARN MORE

HYPERLEDGER URSA

Hyperledger Ursa is a shared cryptographic library that would enable people (and projects) to avoid duplicating other cryptographic work and hopefully increase security in the process.

» LEARN MORE

Source: https://www.hyperledger.org/projects

Hyperledger Sawtooth

Sawtooth is a Blockchain project platform that was originally initiated by Intel and called Sawtooth Lake. The platform is specifically tailored for enterprise use cases with focus on security and transaction throughput.

Features

- **On-chain governance:** The participants of the network vote on the configuration settings of the network, e.g. to decide if a new peer should be given permission to join the network.
- **Parallel transaction execution**: Unlike Ethereum, where everything is strictly sequential, Sawtooth allows for parallel execution of transaction. Therefore, the systems implements a transaction scheduler that keeps track of all read and write operations for the set of transactions that is being put into the next block. Based on the read and write operations, the scheduler builds a dependency graph and executes the transactions that don’t interfere with each other in parallel.
- **Dynamic consensus algorithm:** Sawtooth isolates the consensus algorithm from transaction semantics. Therefore, the vision is to allow arbitrary consensus mechanisms based on the enterprise needs. However, currently there is only one consensus for production ready systems implemented (Proof of Elapsed Time PoET).

Hyperledger Iroha

Similar to Sawtooth, Iroha is platform for creating a permissioned general purpose Blockchain system. Iroha stems from the Japanese fintech company Soramitsu Co., Ltd. and is now open sourced under the Apache 2.0 license. Compared to other Hyperledger frameworks, Iroha specifically focuses on a high Byzantine fault tolerance.

Features

- **Identity management:** Iroha is built on a sophisticated account system including domains, roles and permissions. Domains are used to group certain accounts and assets together. A role is defined by a name and a set of permissions.

- **Asset management:** The system comes with built-in asset management. In comparison with Ethereum, where assets are represented as smart contracts (e.g. an ERC20 token), Iroha provides a special object to define any kind of asset, e.g. a currency unit, a document ownership etc.

- **Consensus**: Iroha implements it’s own consensus mechanism called Yet Another Consensus (YAC).

Hyperledger Burrow

Burrow is basically an Ethereum client that is extended with features for enterprise usage like a permissioned network access for peers. The project was initiated by Monax.io and sponsored by Intel. The system is fully compatible with the EVM specification and able to run compiled Solidity smart contracts.

Features

- **Consensus:** Burrow uses a consensus engine called “Tendermint”. Tendermint consensus is achieved on voting on the transactions by so-called validator nodes. The weight of the nodes with voting rights can by dynamically adjusted.

- **Permissioned EVM:** Ethereum uses the concept of gas to achieve finite contract executions. Since Burrow does not have a currency like Ether, an arbitrary amount of gas is added to every function execution. If a node with invalid permissions tries to execute a function, it will fail and no gas will be added.

- **API Gateway**: Burrow implements an extended API Gateway to query the Blockchain. It provides REST and JSON-RPC endpoints, as well as a socket interface for listening to Solidity events.

Hyperledger Indy

Indy is a distributed ledger that is specially tailored for managing user identities in a decentralized way. According to the official Hyperledger resources, the project is still in “Incubation” stage, i.e. not in productive use [1].

Features

- **Permissioned and public:** Compared to the other Hyperledger projects, Indy is designed to be a public DLT. However, transactions are validated only by permissioned nodes.

- **Transactions:** Data can only be shared with other entities in the network, if the owner of the data allows the transaction. The goal of this behavior is to give a user full control over it’s data.

- **Zero-knowledge proofs:** A core concept of Indy’s identity management are zero-knowledge proofs (ZKP). ZKPs enable the system to verify certain information without disclosing them, e.g. if the person is over a certain age without disclosing the actual age.

[1] Hyperledger wiki: https://wiki.hyperledger.org/start/

Hyperledger Grid

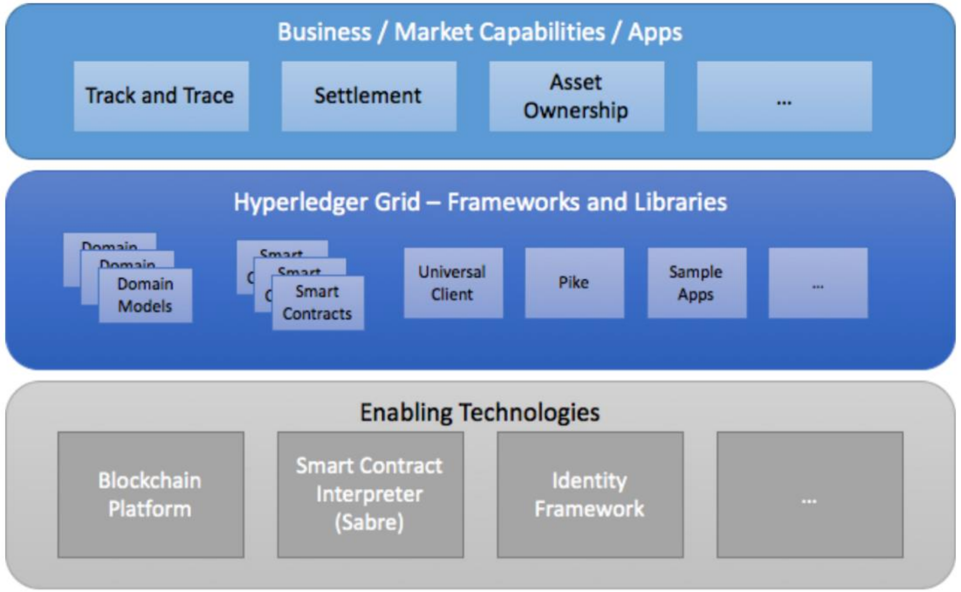
Hyperledger Grid is the most novel project, announced (22.01.2019). So far, it is in incubation phase. HL Grid focuses on supply chain distributed ledger use-cases, using shared, reusable tools. Hyperledger Grid seeks to assemble Hyperledgers shared capabilities in order to accelerate the development of ledger-based solutions for all types of cross-industry supply chain scenarios. Compared to Hyperledger Fabric, it aims to provide a more valueable and accessible functionality, compared to the very low level Hyperledger Fabric.

Features

- **Supply chain-centric:** Implementations of **supply chain-centric** data types, data models, and smart contract based business logic – all anchored on existing, open standards and industry best practices.

- **High level access:** Complexity Reduction and easy access to the underlying Blockchain

- Based on **Web Assembly** (WASM)



Business / Market Capabilities / Apps

Track and Trace

Settlement

Asset Ownership

...

Hyperledger Grid – Frameworks and Libraries

Domain Models

Smart Contracts

Universal Client

Pike

Sample Apps

...

Enabling Technologies

Blockchain Platform

Smart Contract Interpreter (Sabre)

Identity Framework

...

Source Hyperledger Grid Proposal: https://docs.google.com/document/d/1b6ES0bKUK30E2iZizy3vjVEhPn7IvsW5buDo7nFXBE0/edit/

Outline

1. Introduction
   - Terminology
   - Motivation
   - Use Cases
   - Consortium Blockchain projects
   - Corda

2. Hyperledger
   - History
   - Mission
   - Projects

3. Fabric
   - Architecture
   - Membership service provider
   - Application
   - Ordering service
   - Peers
   - Chaincode

Hyperledger Fabric

Fabric is currently considered as the most sophisticated and used framework in the Hyperledger ecosystem. The current version 1.4 is considered stable and ready for productive use. Fabric is highly modular and built with a focus on flexibility and scalability.

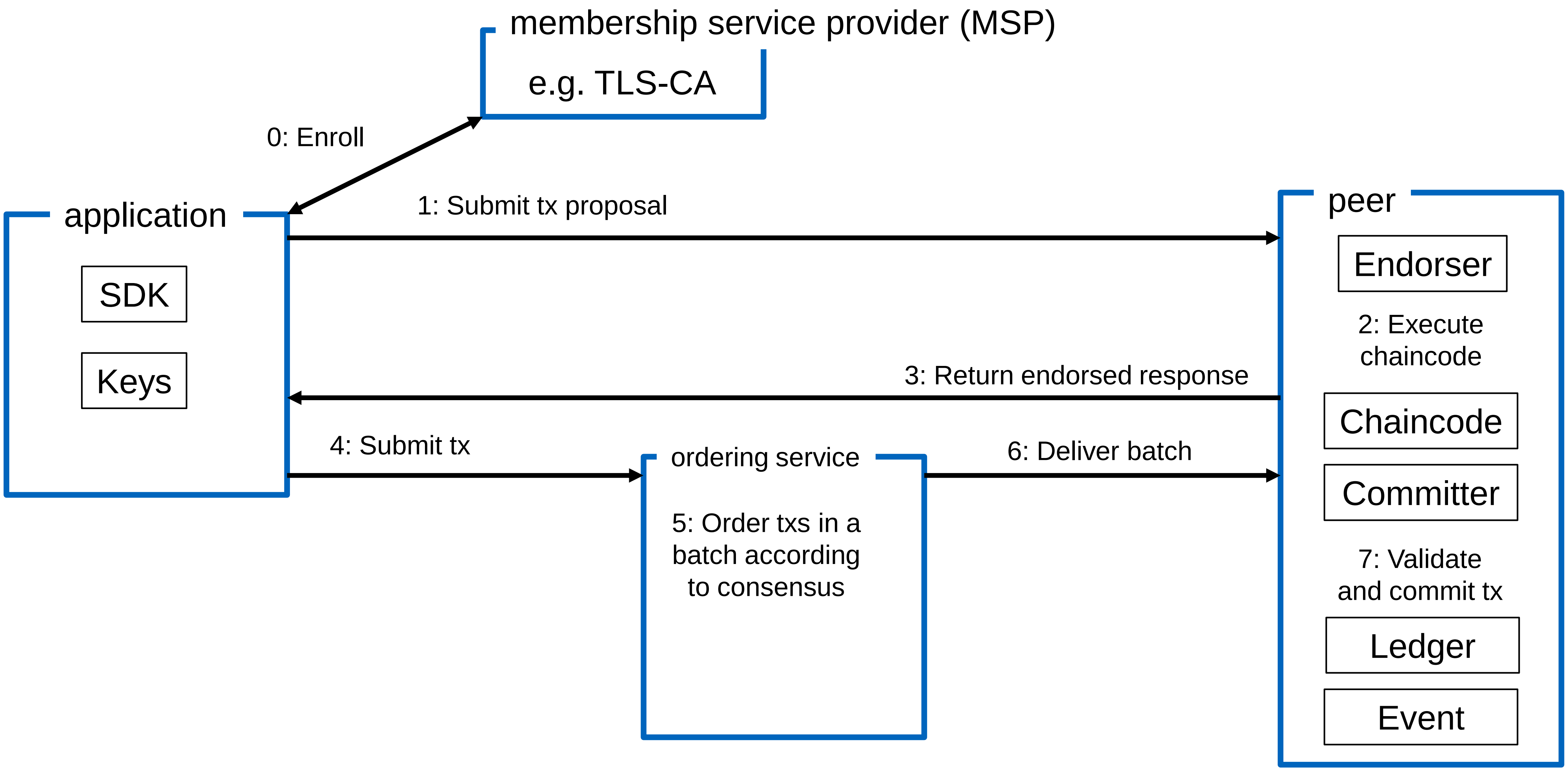
Features

- **Permissioned network:** Fabric is designed as platform to set up private and consortium Blockchain networks. The framework uses a “Membership Service Provider” (MSP) to enroll new nodes into the network.

- **Channels and multi-ledger:** A core feature of Fabric are so-called channels to establish connections between peers. Each member of a channel maintains a copy of the channel’s specific ledger. Like in other Blockchain networks, the ledger contains a sequenced list of all state transitions (transactions).

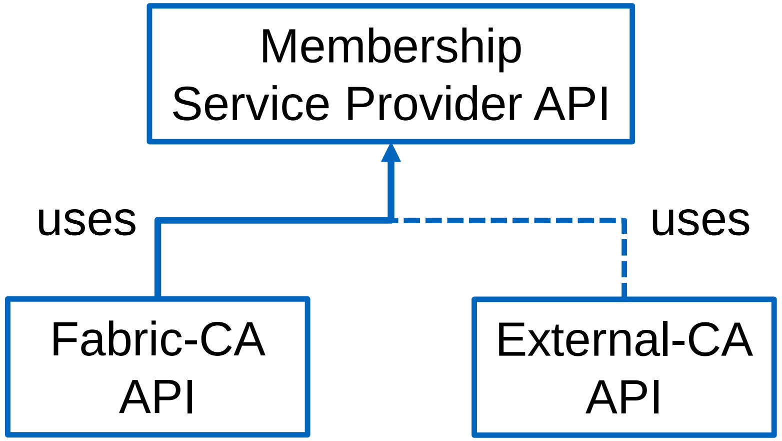
- **Chaincode:** Fabric allows network operators (system chaincode) and developers (application chaincode) to define certain business logic for a whole channel or for particular transactions.

Architecture



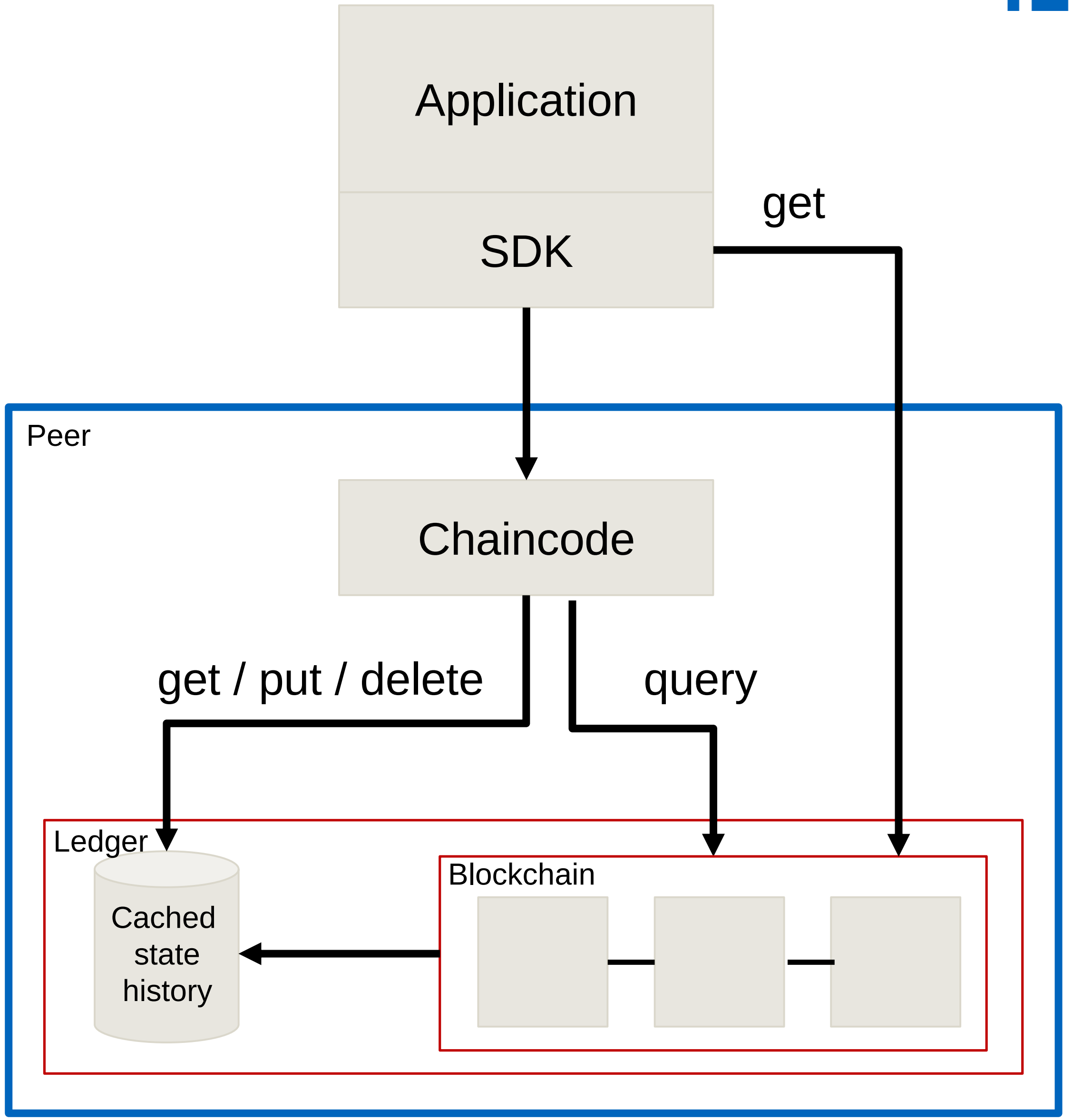
Membership Service Provider (MSP)

- The membership service provider has a pluggable interface that supports a range of credential architectures. A common external certificate authority (CA) is TLS-CA (https certificates).
- Fabric implements it’s own CA which is called Fabric-CA and used by default.
- The MSP is responsible for only letting authorized clients join the network. It governs the identity for applications, users, peers and the ordering service.
- Takes care of the the access control in the whole network. It ensures that only authorized clients can join certain channels and execute chaincode.



Applications

- Currently, applications are programs written in NodeJS or Go that use the Hyperledger Fabric Client (HFC) SDK to invoke calls on a smart contract.
- Applications are very similar to DApps in the Ethereum ecosystem. They usually handle the UI and enable an end user to use the Blockchain.
- Applications submit transactions to the network and invoke chaincode calls.
- Communicate directly with peers



Ordering service

- The ordering service consists of one or more ordering nodes. The nodes do not store any chaincode, nor do they hold the ledger.
- The ordering service enforces the network policy and takes care of the authentication and authorization of the peers.
- Applications must submit transactions to an ordering service node. The node then collects transactions and orders them sequentially. The transactions are then put into a new block and delivered to all peers of a specific channel.
- The blocks are then broadcasted to the peers that hold the chaincode and the ledger. Before the transactions are committed, the peers validate it.
- Consensus is reached when the endorsement of a transaction is accepted, the ordering was successful and the execution was valid and committed.

Peer

- A peer is comparable to full-node in Ethereum or Bitcoin. It maintains the state of the ledger and executes transactions.

- Multiple roles for peers exist:
  - **Committing peer**
    Maintains the state of the Blockchain and commits transactions. It verifies endorsements and validate the results of a transaction.

  - **Endorsing peer**
    An endorsing peer is always also a committing peer. The difference is that an endorsing peer additionally takes transaction proposals and executes them to create endorsements.

- The number of peers in a network is not limited and depends on the operating consortium.

- Chaincode supports the emission of events, e.g. when a certain transaction happened. Peers are responsible for handling and delivering such events to subscribers (applications), publish / subscribe architecture pattern.

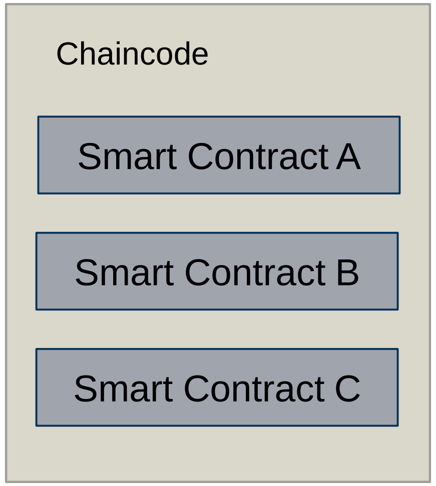
Chaincode and Smart Contracts

In Hyperledger Fabric, uses the terms ”Smart Contracts” and ”Chaincode”. While Smart Contracts refer to the transaction logic of an business object, whereas a chaincode is a technical container of a group of related smart contracts for installation and instantiation.

Basically, chaincode is a program written in Go, NodeJS or Java that is installed on a node. A chaincode program allows to write business logic that changes the state of the ledger.

There are two paradigms for developing Business Logic - Chaincode:

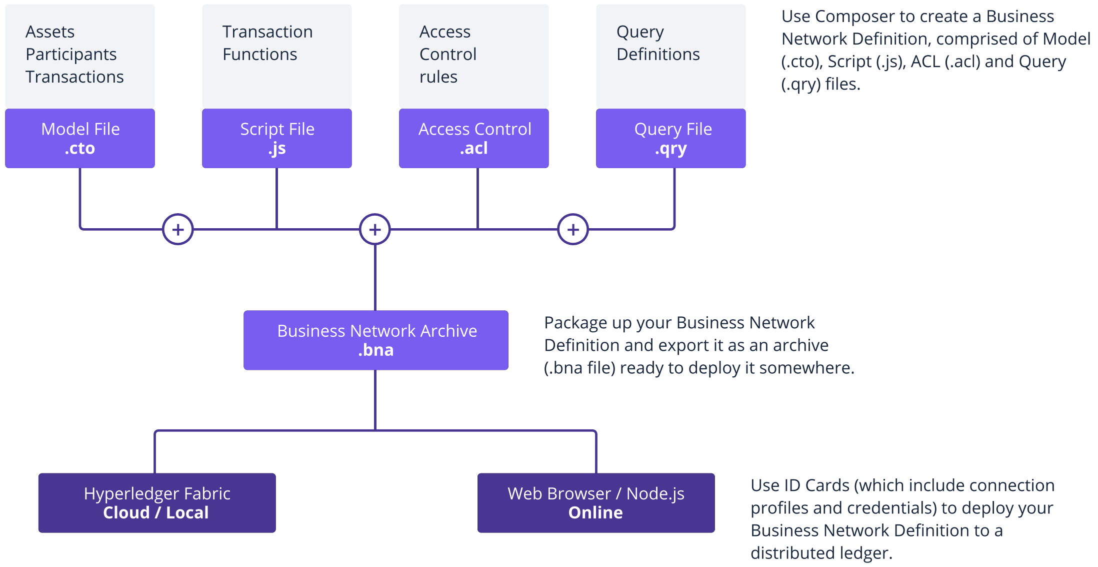
1. Native Fabric (written in Golang, for Production)
2. Hyperledger Composer (for Testing purposes and Proof of Concepts)



Paradigms for business logic development on Hyperledger Fabric

Composer

Hyperledger Composer **runs on HL-Fabric.** Composer is a set of abstractions, tools and APIs to model, build, integrate and deploy a Blockchain solution (a business network archive) consisting of 4 files:



Use Composer to create a Business Network Definition, comprised of Model (.cto), Script (.js), ACL (.acl) and Query (.qry) files.

Package up your Business Network Definition and export it as an archive (.bna file) ready to deploy it somewhere.

Use ID Cards (which include connection profiles and credentials) to deploy your Business Network Definition to a distributed ledger.

Native Fabric

Native chaincode is written in Go, NodeJS or Java that is installed on a node. Native Chaincode is way more complex compared to HL-Composer.

```
package main

import (
   "errors"
   "fmt"
   "strconv"

   "github.com/hyperledger/fabric/core/chaincode/shim"
)

var myLogger = logging.MustGetLogger("asset_mgm")

// HelloWorld example simple Chaincode implementation
type HelloWorld struct {
}

// Init is called during Deploy transaction after the container has been
func (t *HelloWorld) Init(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
   myLogger.Debug("HelloWorld Init is called!")
   return nil, nil
}

// Invoke is called for every Invoke transactions. The chaincode may change
// its state variables
func (t *HelloWorld) Invoke(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
   fmt.Printf("HelloWorld in Invoke with function %s!\n", function)
   return nil, nil
}
```

Comparison Composer vs. Native Fabric: Coding Approach

Composer

1. Data Model (.cto file)
2. Business Logic (.js, Transaction Processor functions)
3. Content Based Query
4. Defining Access Rules (.acl)
5. Emitting Events

Modular Business Logic comprises 4 files

90% (!) less code due to shorter logic and less boilerplate code

Fast but restricted deployment

Restricted Functionalities

Native Fabric

1. Data Model
2. Dispatch Incoming Calls
3. Validate Arguments
4. Lookup Asset
5. Deserialize Data
6. Update Data in Memory
7. Serialize Data and Persist
8. Content Based Query
9. Emitting Events

No Business Logic Seperation + granular fine tuning + general purpose language = complexity

Follow latest version of Fabric capabilities

Single language for both business logic and model

All Functionalities