The BlocWatch Blog

Top Five Differences between Ethereum and Hyperledger Fabric

Jan 29, 2019 / by BlocWatch

Both Ethereum and Hyperledger Fabric are Distributed Ledger Technology (DLT) frameworks built on the blockchain concept. They differ in several important ways, however. Ethereum was designed as a public blockchain that would enable distributed computing applications; Hyperledger Fabric is a permissioned private blockchain designed for use among large enterprises and business groups. Here are their five most important differences:

Participation and Confidentiality

Ethereum is a public and permissionless blockchain, meaning anyone can access and interact with it, and inspect all of its ledger data. Anyone can also build and deploy Decentralized Apps (DApps) on top of Ethereum, and utilize the thousands of nodes that participate in the Ethereum blockchain.

Hyperledger Fabric is the opposite, as it is private and permissioned. Users of a Fabric blockchain first must be granted both network and application access just to connect. Fabric networks can also be segmented into channels, which effectively allows the same nodes to participate in multiple blockchains at the same time. Also, Fabric has built-in support for private data collections, which allows private data to be shared within just one organization on a blockchain, instead of sharing it with all participants.

Smart Contracts and Chaincode

Ethereum smart contracts run in a specialized environment, known as the Ethereum Virtual Machine (EVM). This environment requires smart contracts to be written in special programming languages, of which the most

Posts by Topic

- Blockchain (9)
- Monitoring (7)
- Hyperledger (6)
- Conference (5)
- Hyperledger Fabric (5)

See all

Recent Posts

- BlocWatch Named a Cool Vendor in the May 2020 Gartner Cool Vendors in Blockchain Technology Report
- Monitoring and Securing Your Blockchain [Video]
- Use Cases from Hyperledger Global Forum to Keep an Eye On [Videos]
- BlocWatch and BlockApps Partner to Help Blockchain Users Optimize their Networks

popular is **Solidity**. While the Solidity community is growing fast, it can be hard to find programmers familiar with the Solidity language or its tools.

In contrast, Fabric smart contracts (also known as "chaincode") can be written in several different programming languages, including **Java**, **JavaScript**, and **Go**. These programming languages are used for a wide variety of applications by many programmers. While finding programmers who have experience working with blockchains can be difficult, finding programmers who can write code in Java or JavaScript is easy.

Consensus

"Consensus" in the blockchain world means agreeing to commit transactions to the ledger in a certain order. The order is important – if a user tries to send the same asset (ie spend the same cryptocurrency "coin") via two different transactions to two different recipients, whatever transaction is added first to the ledger first will succeed, and the second will fail (ie the first recipient will get the coin, and the second recipient will get nothing).

Ethereum requires a majority of participants to agree on committing transactions through races to solve computationally-intensive problems, known as Proof of Work (PoW). The winner of each race gets to add a small block of transactions to the ledger, and is given a small cryptocurrency award for their work. This is known as "mining a block".

Each Ethereum node keeps its own copy of the ledger. As Ethereum nodes race each other for mining rewards, their ledgers often get out of sync on the last few blocks. They correct their ledgers eventually, however (including erasing rewards they previously thought they'd won), because they can use their mining rewards only if their ledgers completely match everyone else's. But because the last few entries in the ledger of an Ethereum node may not contain the consensus entries, it's important to allow the ledger to "age" for a dozen or so blocks before trusting the entries.

Hyperledger Fabric takes a different approach to consensus. It separates the consensus process into three distinct steps, each executed by a separate set of nodes:

1. **Execute**: Transactions are executed and endorsed in any order by endorsing peers.
2. **Order**: Transactions are put in order by orderer nodes.
3. **Validate**: Transactions are validated by committing peers, and added to their own copy of the ledger only if valid.

- **Enterprise Blockchain Trends in 2020**

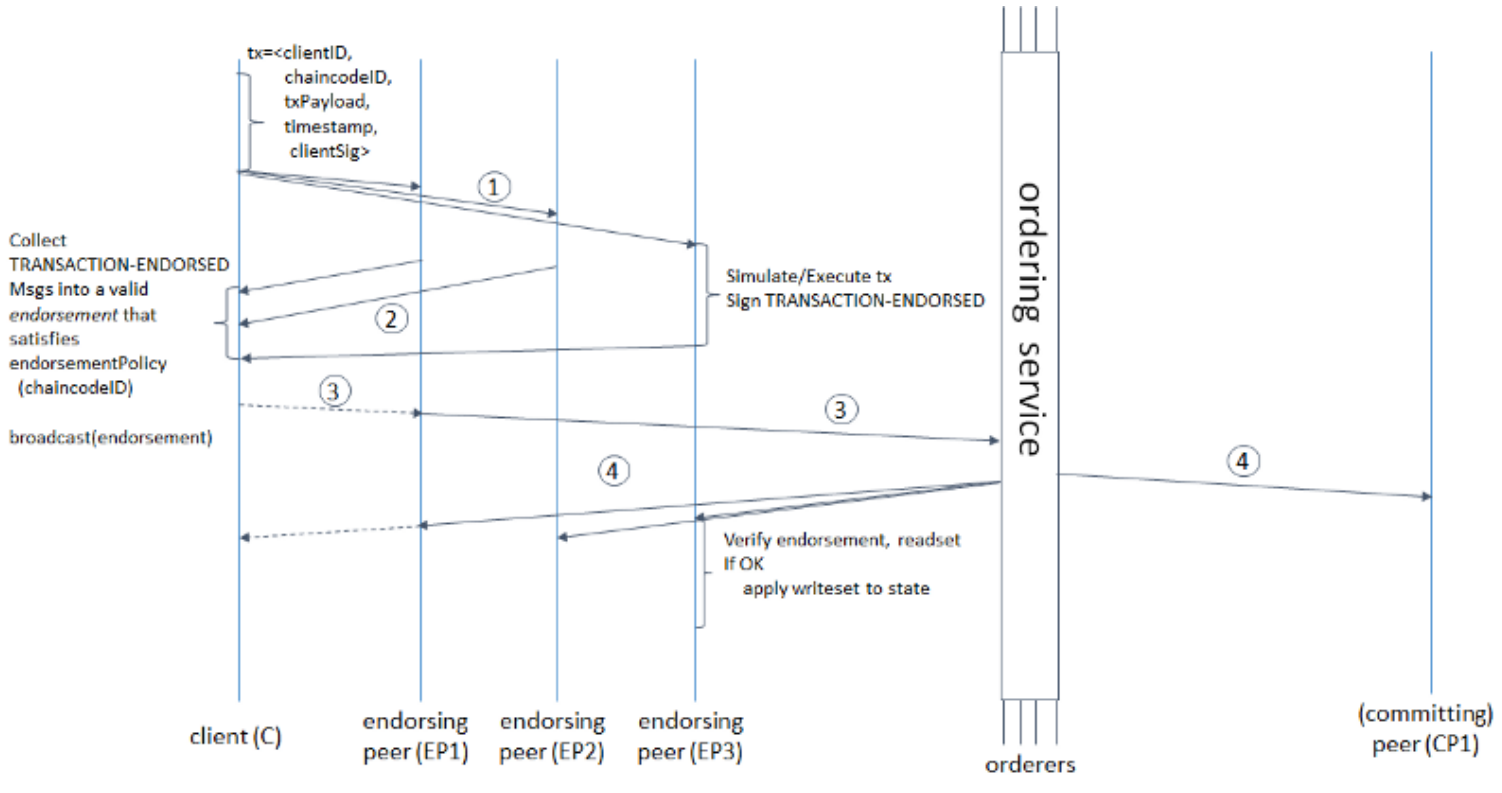

Illustration of one possible Hyperledger Fabric transaction flow (common-case path)

This means that transactions in the ledgers of Fabric nodes are always in the same order – they don't get out of sync. So any application reading from a Fabric ledger doesn't have to wait for blocks to age; they can be trusted immediately.

Also, different consensus algorithms can be plugged into Fabric's orderers. Fabric supports fault-tolerant consensus algorithms, including **Zab** (as implemented by Apache Zookeeper/Kafka) and **Raft** (implemented by Etcd).

Cryptocurrency

A cryptocurrency called "Ether" is built into the Etherum blockchain, and anyone who wants to add a transaction to the Etherum ledger must spend a small amount of Ether in order to do so. This Ether is used to fund mining rewards.

Ethereum also has a well-developed system of smart-contract interfaces that allows custom currencies and tokens (such as **ERC-20** tokens) to be developed on top of the Ethereum blockchain. These currencies/tokens are managed entirely by smart contracts, which can be coded with custom rules for minting new tokens or transferring tokens among accounts.

Fabric has no native currency. Tokens can be built on top of Fabric through chaincode, similar to Ethereum; however, Fabric doesn't yet have any widely-deployed token standards similar to ERC-20.

Accounts and Identity

Ethereum has the concept of "accounts" baked into it; every Ethereum transaction is sent from one account to another account. The sending account is always an **Externally Owned Account** (EOA, an account operated by a person), and the receiving account may be either an EOA or a **Contract Account** (an account run autonomously by a smart contract). To transfer Ether from one account to another, the owner of the first account would send a transaction to

the second account. To transfer a custom token from one account to another, the owner of the first account would send a transaction to the account of the smart contract that manages the token, and the smart contract would update the Ethereum ledger to indicate that the token is now owned by the second account.

Fabric has no direct analogy to Ethereum accounts. Transactions are submitted using identities provided by member organizations through **Membership Service Providers** (MSPs), and transactions have no designated recipients. To transfer an asset from one owner to another, a blockchain participant (usually the owner of the asset, but not necessarily, depending on the smart contract) would submit a transaction to execute a function of the smart contract that manages the asset, and the smart contract would update the ledger to indicate the asset now has a new owner.

In Summary

From those five key differences between Ethereum and Hyperledger Fabric, you can see why Ethereum is popular with people developing decentralized applications for public consumption, while Fabric is a favorite of businesses and organizations looking to exercise more fine-grained control for industry-specific vertical applications.

Tags: **Ethereum**, **Architecture**, **Hyperledger Fabric**

Written by **BlocWatch**