Solidity

Solidity is a high-level programming language, which is designed to work with the technology of the time -Blockchain. More specifically speaking, Solidity is designed to develop Smart Contracts in Ethereum Blockchain platform.

Smart Contracts

A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum Blockchain. In each Contract, we can define State Variables, Methods, and Events etc. This contract can manage transactions between blocks in Blockchain network. Each block has a particular address in the form of a cryptographic key that generated by the result of some functions including hashing of adjacent blocks. This creates a strong relationship between adjacent blocks. So that manipulation or any other form of hacking in nodes or blocks are not easy or not even possible. Solidity is one of the many languages that can be used to develop EVM (Ethereum Virtual Machine) understandable bytecode. There are many built-in classes and Libraries in Solidity which support hassle free smart contract development. You can use the IDEs like Remix, Visual Studio (With Solidity extension), Ether atom, IntelliJ IDEA plugin (both with Solidity extension) to develop.

Following are some of the features of solidity which are very similar to common high-level languages like Java and C++.

Statically typed Language

Though it is having a structure of JavaScript, unlike JavaScript it is a Statically Typed language. For example, you must declare the type of a variable like in C++ and Java before it is used. Otherwise, a compile-time error will be generated

Contract and Interfaces

'Contract' is a unique data structure of Solidity language, it helps to create and manage contracts easily. Contracts can be inherited by child Contracts and can create complex contract structures.

Function Modifier

It is similar to function override in other OOP languages. Suppose you want to execute a function in a different way when a condition is met. In this case, you can use function modifier feature to change the behavior of the function. Generally useful in inherited Contracts to override the parent function behavior.

Events

Events are used to write information from contract to Blockchain log. The event is similar to a function which takes the data as the argument and writes to Blockchain client's log.

Access Specifies

This is similar to the access specifies in other OOP languages like private and public. In Solidity the name and access rights have some change. 'Owned' and 'mortal' are two access specifies in Solidity. There are some more options available to extend security.

Explicit Type conversion

You can perform explicit conversion of different data types. Such conversions are generally checked at compile time, but there are exceptions also.

Memory Arrays

Dynamic arrays can be directly allocated to memory.

Libraries

You can access the vast built-in Contract Libraries, which can be used to develop your custom contract.

Import

With 'import' keyword you can import other source files to your contract.

These are some very basic features you can find in Solidity, refer more and understand the simplicity of Solidity.

A sample contract in solidity.

```
pragma solidity ^0.4.0;
contract SimpleStorage
{
        uint storedData; // State variable storedData declared as uint
(Unsigned integer)
        function set(unit x)
        {   storedData=x;
          }
        function get() constatnt returns (uint) {
          return storedData; }
}
```

This represents a contract in Blockchain.

Here, each block has the accessibility to 'set' and 'get' the values

If we need the restrictions like 'only the owner of the contract can set the value' then we can provide accessibility only for the owner of the contract or block, an example is below.

```
pragma solidity ^0.4.0;
contract SimpleStorage
{
        address owner; // Here 'Owner' holds the address of the current
block.
        function SimpleStorage() // The constructor  runs only once and set
the address of particular contract to the 'Owner'
        {
        owner=msg.sender
        }
        uint storedData;
        function set(unit x) {
        if(owner!=msg.sender) // when an address other than the 'Owner' try
to access the 'Set' function, the program deny access
        {
            return "Sorry you can't modify this data"
```

```
        }
        else
            storedData=x;
          }
        function get() constatnt returns (uint) {
        return storedData;
        }
}
```

Like this, we can modify the permissions of different blocks in a Blockchain

Hyperledger Fabric

Hyperledger Fabric is a blockchain framework implementation initially developed by Digital Asset and IBM and now hosted by Linux Foundation under the hyperledger project. Fabric joined the hyperledger project for incubation in the early 2016 and after 1 year of incubation, it became the first project get into the 'active' state. On July 11, 2017, the hyperledger Technical Steering Committee announced their first production-ready distributed ledger codebase, Hyperledger Fabric V1.0.

The Fabric platform is intended as a foundation for developing blockchain applications, products or solutions. The fabric is a Private and Permissioned system which delivers a high degree of confidentiality, resilience, flexibility, and scalability. It adopted a modular architecture and supports pluggable implementations of different components like consensus, membership services etc. Like other blockchain technologies, Fabric has a ledger and smart contracts. **The smart contract in the fabric is known as chaincode** and it is in the chaincode the business logic is embedded. The following features impart a high degree of security and privacy for the fabric framework.

Channels: The private blockchains built on fabric protocol. Each channel consists of only the authorized partners

Visibility settings: It is possible to restrict who can see the input data using visibility settings

Data Encryption: The data can be Hashed or Encrypted before calling the chaincode

Data Access Restriction: By embedding access controls into the chaincode logic, it is possible to restrict data access to certain roles in the organization.

File encryption: Ledger data at rest can be encrypted using file system encryption, and data-in-transit can be encrypted using TLS protocol.

Fabric v/s Ethereum

Both the Fabric and Ethereum are two blockchain technologies available today. But there are several differences in Structure, Mode of Operation and many other aspects of both technologies. It is worth to understand the difference between these two.

Governance:

As discussed earlier, Fabric is governed by the Linux Foundation and Ethereum is hosted and governed by Ethereum Developers community.

Platform:

Ethereum is a generic blockchain platform for any kind of application whereas Fabric is a modular blockchain platform which is customized mainly for supporting different business domain

Targeted Crowd:

Ethereum is targeted towards applications that are distributed in nature, on the other hand, the fabric is meant for business domains including banking, health, goods delivery etc.

Network Structure:

In Ethereum, network partitioning is not possible. Therefore, the transactions are visible to everyone in the network including the competitors. Obviously, it is not suitable for the business environment. Fabric offers a provision to create private networks, called channels. Each channel consists of only the authorized

partners. In addition to that, there is 'Endorsing Node' which are designated to approve a transaction

Mode of operation:

Ethereum is a permission-less system, which means any node is allowed to participate in the network. On the other hand, Fabric being a private and permissioned system, the members who are enrolled through MSP (Membership Service Provider) can only participate in a network

Consensus:

In Ethereum consensus is based on PoW (Proof of Work) scheme. All participants have to agree upon a common ledger and all participants have the access to all entries ever recorded. This may affect the transaction processing as the network grows. In Fabric, the consensus can be achieved in different ways. The process of endorsing is based on an endorsing policy. Nodes can take a certain role called Endorsing Peers (endorser), who are responsible for endorsing transactions based on the policy. All this results in a better performance and security.

Smart Contracts:

In Ethereum, business logic is included in smart contracts written in Solidity language. In fabric, smart contract is implemented using chaincode and it can be written in languages like Go, Python etc.

Currency:

In Ethereum, there is a built-in cryptocurrency called Ether. Digital tokens for custom use-case can be also be built on top of ether. In fabric, since the consensus is not reached through mining, there is no need of built-in cryptocurrency. However, it is possible to develop a currency or a digital token for specific use-case.

Even though a full-fledged application development process has not yet taken place in Fabric, the initial results and the features it provides indicate that the Fabric is a promising technology for future enterprise level application.

Hyperledger Iroha

Hyperledger project has attracted the attention of many MNCs to co-develop blockchain frameworks suitable for different fields. Under the hyperledger project, 5 frameworks have been unveiled so far and Iroha is one of them. It is one of the blockchain platform implementation inspired by hyperledger fabric. It was initially developed by Japanese startup Soramitsu, Hitachi, NTT Data and Israeli startup Colu and now it is hosted by Linux Foundation. Iroha joined the hyperledger project in October 2016 and became the 3rd project under the hyperledger umbrella. On May 2017, the hyperledger Technical Steering Committee changed the state of Iroha from 'incubation' to 'active'.

Iroha provides a development environment for C++, web, and mobile application developers so that the developers can contribute not only to Iroha but to the whole Hyperledger Project. It allows the developers to create reusable components in C++ and that can also be called from other languages like Go. Iroha includes some common use cases like deploying new currencies, sending messages, etc. in its core framework.

The consensus methodology adopted in Iroha is Sumeragi which is a new chain-based Byzantine Fault Tolerant consensus algorithm. Sumeragi is found to be one of the fast executing consensus algorithms. The robust libraries of reusable components provided by Iroha are highly useful in custom project development in it. Some of them are.

- Sumeragi consensus library
- Ed25519 digital signature library
- SHA-3 hashing library
- Iroha transaction serialization library
- P2P broadcast library
- API server library
- iOS library
- Android library
- JavaScript library
- Blockchain explorer/data visualization suite

Components of Iroha

- Iroha is composed of the following:
- Iroha core
- Iroha Native iOS Library
- Iroha JavaScript Library
- Iroha Native Android Library

Iroha core includes the distributed ledger infrastructure, data membership services, consensus algorithm, peer-to-peer network transmission, data validation, chaincode infrastructure etc.

Features of Iroha

- Simple construction
- Modern, domain-driven C++ design
- Emphasis on mobile application development
  It includes libraries for android, iOS and JavaScript
- Creation and management of custom complex assets
- User account management

Hyperledger Sawtooth

Hyperledger Sawtooth is blockchain development platform initially developed by Intel and now hosted by Linux foundation under its hyperledger project. Similar to other hyperledger projects sawtooth is also a highly modular open source platform for blockchain development. **The sawtooth is mainly proposed for the insurance claim processing, IoT assisted supply chain management, and international remittance.** Sawtooth already developed some versatile DApps in the above-mentioned areas.

Components of sawtooth

Every distributed ledger must contain the following basic components:

- A data model:-To represent the current state of the ledger.
- The language of transactions : To change ledger state.
- Protocol:-To makes consensus among participants.

The versatility, security, and simplicity of a hyperledger depend upon the features of these three components. The sawtooth has a well-defined structure for these components**. In sawtooth, the data model and language of transactions are combined together as a 'transaction family'.** So each transaction family will have a specific data model for defining the ledger state and a language of the transaction for changing the state. **The user can create their own transaction family according to their ledger requirements.**

Some transaction family models are

Validator registry

- Contains the methodology for registering ledger services

Integer Key

- Used for testing deployed ledgers

Settings

- For storing on-chain configuration settings

Identity

- Handles on chain permissioning for participants and validators. i.e.; for managing identities for the list of public keys.

The user can create the transaction families based on any of the above models. There are some readily available transaction families for some specific areas.

Smallbank

Used for performance analysis and benchmarking

Seth

Used for creation and execution of smart contracts

Blockinfo

Used for storing information about the configurable number of historic blocks.

XO

To play tic-tac-toe

Supply chain

For tracking objects

track & trade

For tracking ownership, custodianship

Consensus in Sawtooth

Sawtooth follows some special protocols which will ensure a universal agreement on transaction validations. This vigorous consensus method differentiates sawtooth from other platforms. In sawtooth blockchain, there will be clients and validators. **The validators are selected randomly for each transaction. It employs an algorithm to choose the validators, then the selected validators will engage in transaction validation process.** Sawtooth provides two

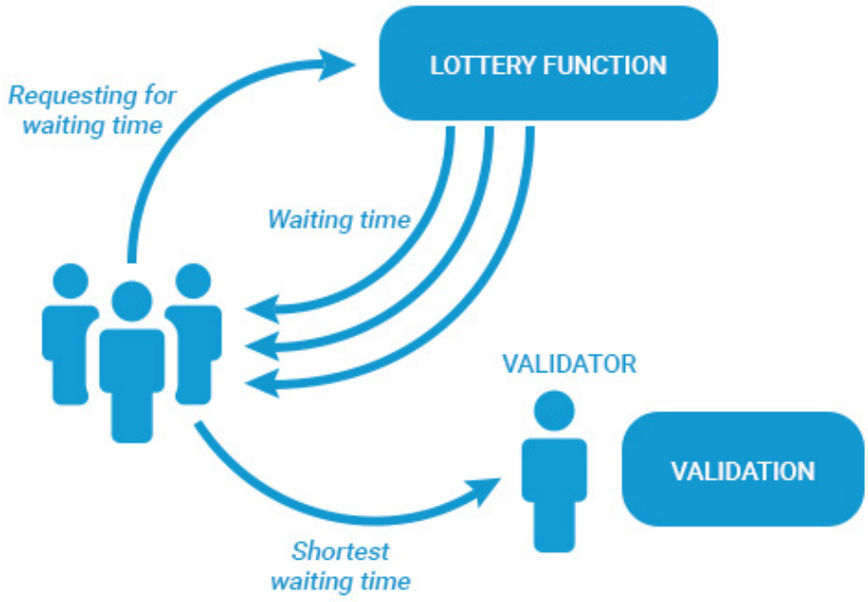
consensus implementation method they are **Dev**-mode and **PoET.**

Dev-Mode

It is a simplified random leader algorithm used for consensus implementation. This is useful for developers and testers of sawtooth blockchain apps.

PoET (Proof of Elapsed time):

Which is a production-grade protocol for large networks, were consensus-making is a bit heavy task. In this method, the network uses a lottery functions for implementing consensus. A lottery algorithm is used for finding the leaders from a set of participants in a blockchain. In the PoET algorithm, all the validators will request for a wait time to an enclave function. Once they got the wait time, the validator with the shortest wait time will consider as the leader for validating that transaction. The function is broadcasted to all eligible participants in the blockchain to maintain fairness in the selection process; so that the leadership will be distributed randomly among these validators.



DApps in sawtooth

The hyperledger sawtooth already implemented some DApps. The Main apps are

1) Seafood supply chain traceability.

2) Bond asset settlement.
3) Marketplace digital asset management.

Seafood supply chain traceability

This application will provide the traceability and accountability of seafood from the harvest to the end customer. Basically, it works on IoT, the physical objects are traced through the blockchain network. The sawtooth app will trace the entire journey of a seafood from the ocean to the dining table with the help of IoT sensors. It can trace the ownership, possession and other parameters like location, temperature, humidity, motion, shock, and tilt etc.

The customers will also get the opportunity to trace the 'journey' of their food. They can verify the quality themselves through the blockchain network.

Bond Asset Settlement

This will provide a secure and trustworthy solution for the bond settlement. Sawtooth provides bond asset settlement application with highly secure smart contract facility. The app provides a user interface to create, buy, sell and settle the bonds.

Marketplace Digital Asset Exchange

This application provides an infrastructure for transferring digital assets. The user can easily create participants and asset, and can easily manage all the asset transactions.

Maybe it is too early to make a definite conclusion on future of sawtooth. However, by considering the features it provides and the developments happened so far, we can definitely say that the sawtooth has the potential to make game-changing applications.

Cello

Perhaps, the complexities, as well as the benefits of a blockchain is clear to most of the aspiring learners. And most of the minds are stuck up with its complexity more than the enormous possibilities. But many tools are being created daily to transform the blockchain management process as simple as any other software handling. Hyperledger cello is one of the tools in this category which makes blockchain management functions easier for normal users. The Cello acts as a middle layer in between the Blockchain and the blockchain infrastructures.

The Cello toolkit is essentially helpful for Implementing Blockchain as a Service (BaaS), as it eases the effort required for blockchain management such as create, manage, and terminate etc.

How it works?

Cello creates a layer in between the Blockchain and the blockchain infrastructure and provides automatic multi-tenant chain services. Operators can create, manage and perform other blockchain operations in simple steps by the dashboard provided by the cello.

Features of Cello

- Cello can manage the different stages of blockchain such as create, delete, terminate etc.
- Can create customized blockchain networks
- Supports various kinds of virtual machines and containers
- Supports heterogeneous system architecture
- Automatic monitoring and screening of blockchain are possible.

Cello operator dashboard

The Hyperledger cello helps blockchain developers in various aspects. **The main attraction of Hyperledger cello is the quick creation of Private Blockchain The developers can quickly create blockchain and provide BaaS from scratch without doing the complex programs.** The customization feature allows the user to

decide the basic features like size of the chain, consensus mechanism, hosts etc. Through the dashboard the operator can manage a number of blockchains on top of virtual clouds, container clusters, bare metals etc. (e.g., Swarm, Docker, Kubernetes,). The tool support functionalities like adjusting the chain numbers, check system status, scale resources and so on.

The hyperledger cello can bring great flexibility and scalability in Blockchain network management. Currently, most of the cello services are available only for hyperledger fabric. It will expand its services to other hyperledger frameworks like Sawtooth, Embark etc.-

Comparison of Bitcoin, Ethereum and Hyperledger

So far we have discussed the popular blockchains like Bitcoin and Ethereum and the Blockchain development framework Hyperledger. We also peeked into some of the enterprise level blockchain frameworks under Hyperledger project. To get the complete picture at a glance let's have a look at the comparison chart. Remember again that Bitcoin and Ethereum are two blockchains while Hyperledger includes a lot of Blockchain development frameworks. So it will be a general comparison.

	Bitcoin	Ethereum	Hyperledger Frameworks
Cryptocurrency Based	Yes (Bitcoin)	Yes (Ether)	No
Permissioned	No	No	Yes (In general)
Auditable	Yes	Yes	Yes
Immutable Ledger	Yes	Yes	Yes
Modularity	No	No	Yes
Smart Contract	No	Yes	Yes
Consensus Protocol	PoW	PoW	Various
Major application domains	Cryptocurrency	DApps including cryptocurrency and more.	Enterprise blockchain development