Ethereum

Part 2– Transaction Lifecycle, Consensus, Tokens, Dapps

Prof. Marco Comuzzi
Department of Industrial Engineering
Ulsan National Institute of Science and Technology (UNIST)
mcomuzzi@unist.ac.kr

1.
Ethereum transaction and blocks life cycle

A lot is just like Bitcoin

Transactions are issued, validated by the nodes, stored locally, and gossiped

Valid transactions are assembled into a new block
(mining/minting, see later)

Block is gossiped and validated by each node

When a new block is received, a node executes all the transactions in it to reach the new "Ethereum state" agreed by all the nodes

How?

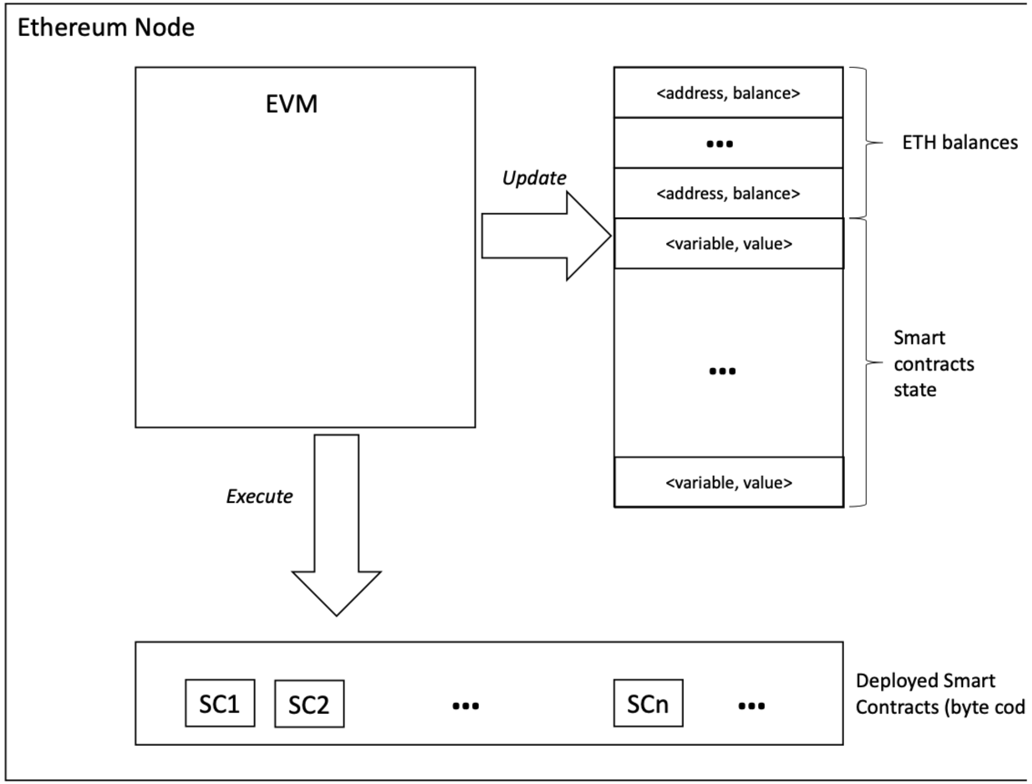
EVM – Ethereum Virtual Machine

EVM is the "software" core of an Ethereum node

It executes transactions

Modifies the Ethereum state

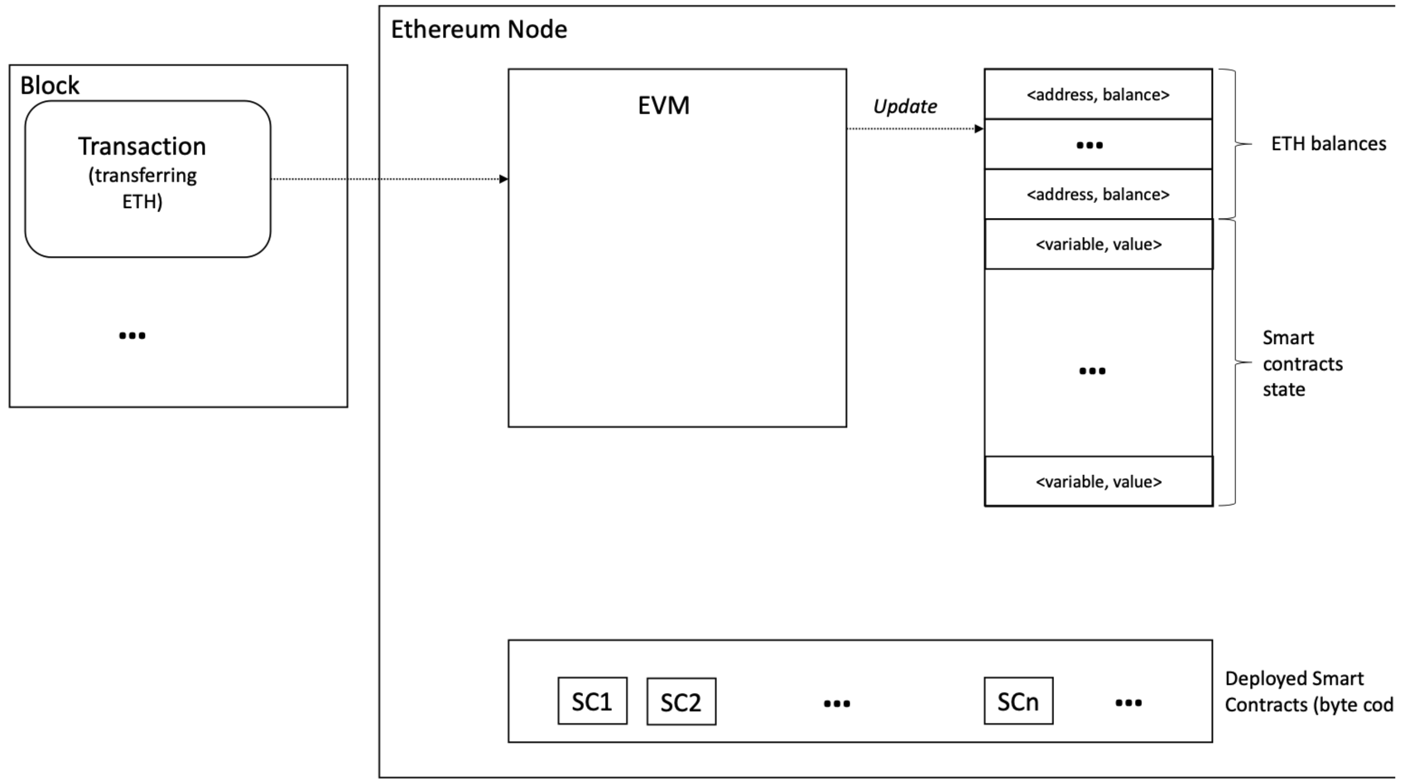
Executes smart contracts



Type 1: ETH transfers

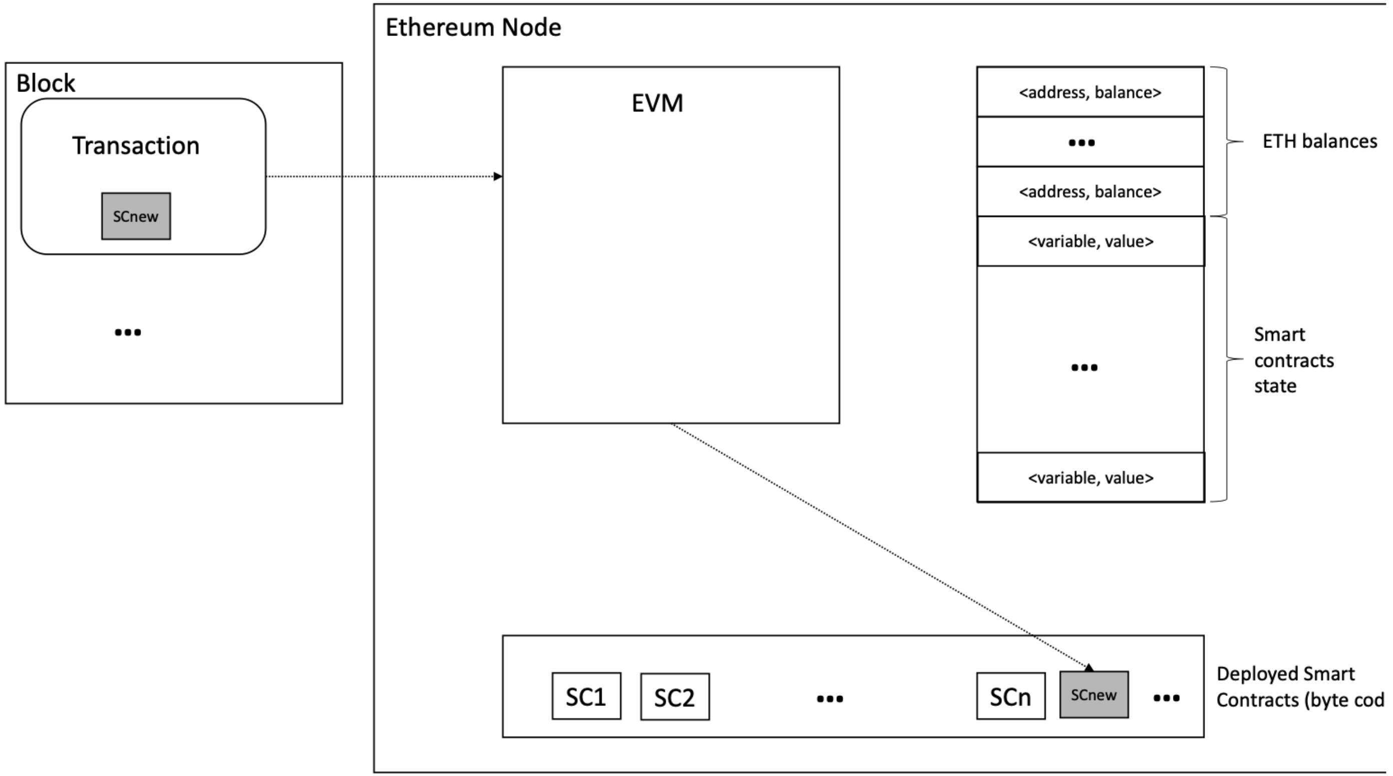
Update the state

(ETH balances of sender and receiver accounts)



Type 2: Deploy Smart Contracts

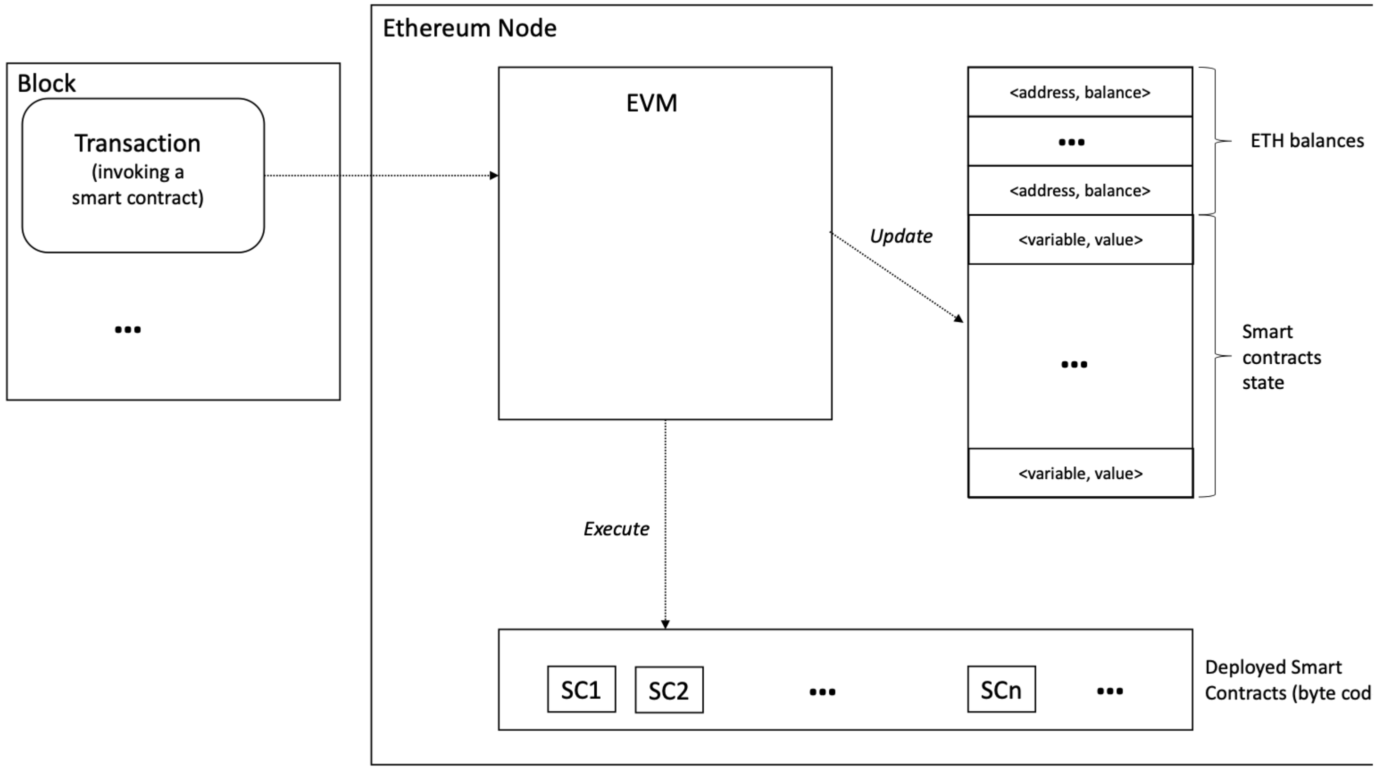
Install locally the byte code of the new smart contract



Type 3: Invoking Smart Contracts

Execute (run) the smart contract

Execution may update the Ethereum state



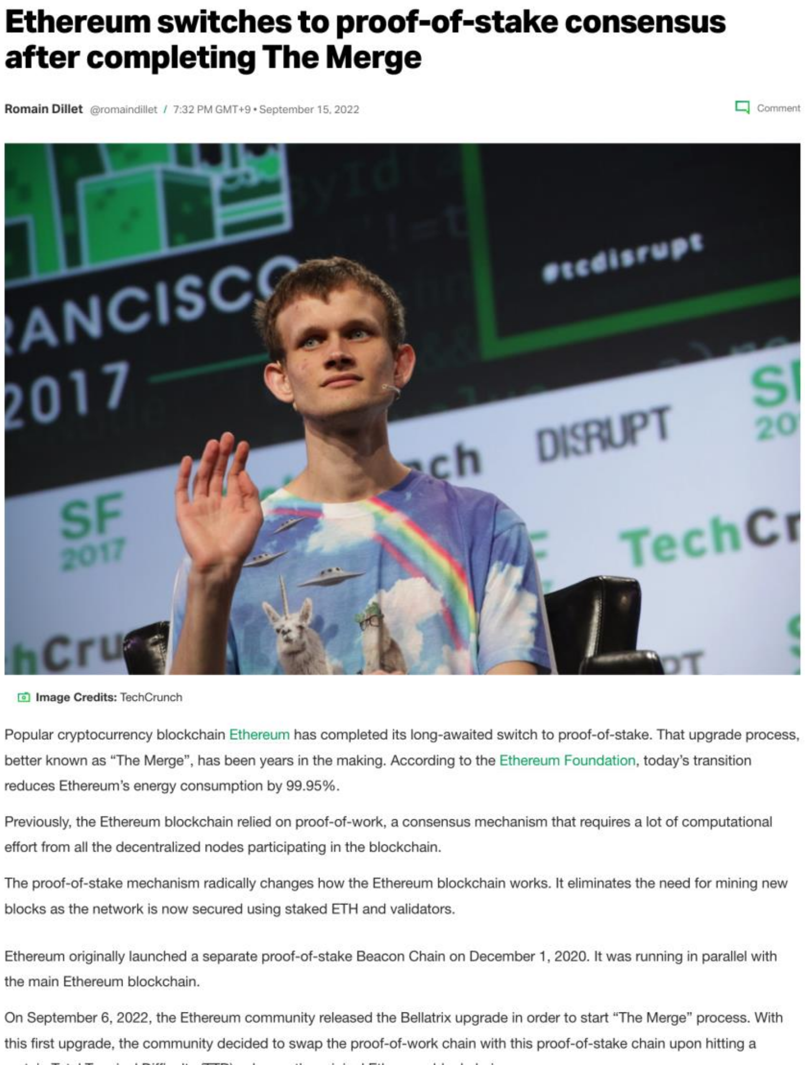
2.
Ethereum consensus mechanism

How are new block assembled?

The same problem as Bitcoin: assembling and distributing blocks…

Until September 15th, 2022: Ethash, a PoW-based consensus mechanism

Since September 15th, 2022: A new mechanism based on "Proof-of-Stake"



Ethereum switches to proof-of-stake consensus after completing The Merge

Romain Dillet @romaindillet / 7:32 PM GMT+9 • September 15, 2022

Comment

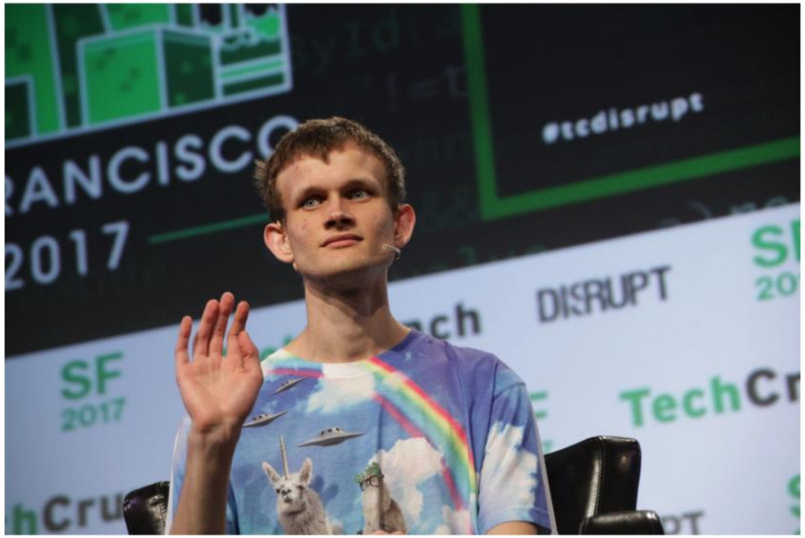


Image Credits: TechCrunch

Popular cryptocurrency blockchain Ethereum has completed its long-awaited switch to proof-of-stake. That upgrade process, better known as "The Merge", has been years in the making. According to the Ethereum Foundation, today's transition reduces Ethereum's energy consumption by 99.95%.

Previously, the Ethereum blockchain relied on proof-of-work, a consensus mechanism that requires a lot of computational effort from all the decentralized nodes participating in the blockchain.

The proof-of-stake mechanism radically changes how the Ethereum blockchain works. It eliminates the need for mining new blocks as the network is now secured using staked ETH and validators.

Ethereum originally launched a separate proof-of-stake Beacon Chain on December 1, 2020. It was running in parallel with the main Ethereum blockchain.

On September 6, 2022, the Ethereum community released the Bellatrix upgrade in order to start "The Merge" process. With this first upgrade, the community decided to swap the proof-of-work chain with this proof-of-stake chain upon hitting a

Ethash

The last step is hashing, with the objective of obtaining a hash value below a certain target

The difficulty of the algorithm is adjusted to have on average a new block mined every 12 seconds

When a new node is mined, a miner node receives

- A fixed reward of 3 ETH
- All the gas used in the block
  (= all the gas actually spent to run transactions in the new block)

Ethash: How did it work?

It was based on a large-sized DAG (Directed Acyclic Graph)

Size increases to increase complexity

DAG started at 1GB, became a few GBs

Fetch some random data from the DAG, combine it with transaction data and return the hash of the result

“Memory hard”

Execution time depends mainly on the time required to read and load the DAG in the memory of the miner node and NOT on the computation done on the DAG

Ethash: Why a “memory hard” consensus?

Bitcoin PoW is not “memory hard”

The only data to load in memory is the block

PoW is complex because of a lot of hashing

Why a “memory hard” PoW algorithm?

- Avoid ASIC miners and explosion of computing power required for successful mining
- ASIC (and, generally, hardware components) are good at executing many simple computations very quickly, but they can only have very limited memory

From PoW to PoS (Proof-of-Stake)

In PoW, the miners must prove to have consumed an off-chain, scarce resource (electricity) to be rewarded when mining a new block

What if instead miners could "bet" an on-chain resource (e.g., coins that they own) to become miners of new block?

- If they propose a valid new block, they keep their coins and get a (small) reward
- If they propose an invalid block, they loose (at least part of) their coins

Move the "race" from improving hardware to stacking coins!

Ethereum PoS: How does it work

No miners, but validators. Validators propose block and validate blocks proposed by others

To become a validator, a node must stake at least 32 ETH (~40,000 USD nowadays).

ETH are staked by sending them to a smart contract, which locks them

"Pools" of validators are chosen randomly at each "epoch" (~6.5 minutes)

Within a pool, the proposer of new blocks is also randomly chosen

Ethereum PoS: Incentives

Rewards:

- Interest on the ETH staked
- Given the basic reward B, 1/8 B goes to proposer of new valid blocks , 7/8 split among the validators of a block
- B can be adjusted dynamically by the protocol

Validators loose ETH if they do not validate a node that they should (e.g., they are offline)

Validators loose all their staked ETH if they propose an invalid block

Ethereum PoS: Implications

PoS consumes 99.5% electricity less than PoW (Ethash)

(Probably) more secure than PoW: to hack PoW one miner needs 51% of mining power, to hack PoS one validator must stake 51% of the ETH in the world.

Allows much more flexibility

E.g. “Sharding” (parallel forks of the ledger) should improve Ethereum’ throughput from 15 to 100,000 transactions per second

3.
Ethereum Tokens

Ethereum tokens

"Tokens" in Ethereum are smart contracts that define the behaviour of classes of digital assets in the Ethereum blockchain

A token smart contract defines:

What users can do with with tokens

How the state of tokens can be modified (e.g., ownership)

What can be a token?

Class	Description/examples	Type
Currency	Private cryptocurrencies, like a cryptocurrency that can be used only on a university campus, or one that users can spend only to pay for the services provided by a network of private hospitals	Fungible
Computing Resources	Storage capacity of CPU time provided by a cloud service	Fungible
Asset	Digital equivalent of physical assets, such as gold, paintings, cars, real estate, or diamonds	Depends of the physical asset (e.g., fungible for gold, non-fungible for paintings or cars)
Access	Access rights to physical property, such as a hotel room or an exclusive Web site	Non-fungible
Equity	Shareholder equity of an organization	Fungible
Voting	Right to vote in a digital or legal system	Fungible
Identity	A digital or legal identity	Non-fungible

Fungible v. Non-Fungible Tokens

A token is "fungible" when its units are fundamentally interchangeable, "non fungible" otherwise.

Currency, gold, equity >> fungible

Paintings, cars, digital art >> non-fungible

Fungible v. Non-Fungible: Ownership

Fungible Tokens
<Alice, 25 tokens>
<Bob, 15 tokens>
…

State specified by how many token units owned by nodes

Non-Fungible Tokens
<Token id: “45fc…”, Bob>
<Token id: “rt44…” , Alice>
…

State specified by which node owns each token

Counter-party risk in Non-Fungible Tokens

Non-fungible tokens represent assets in the real world (physical or digital)

Identification and **integrity** can be a problem

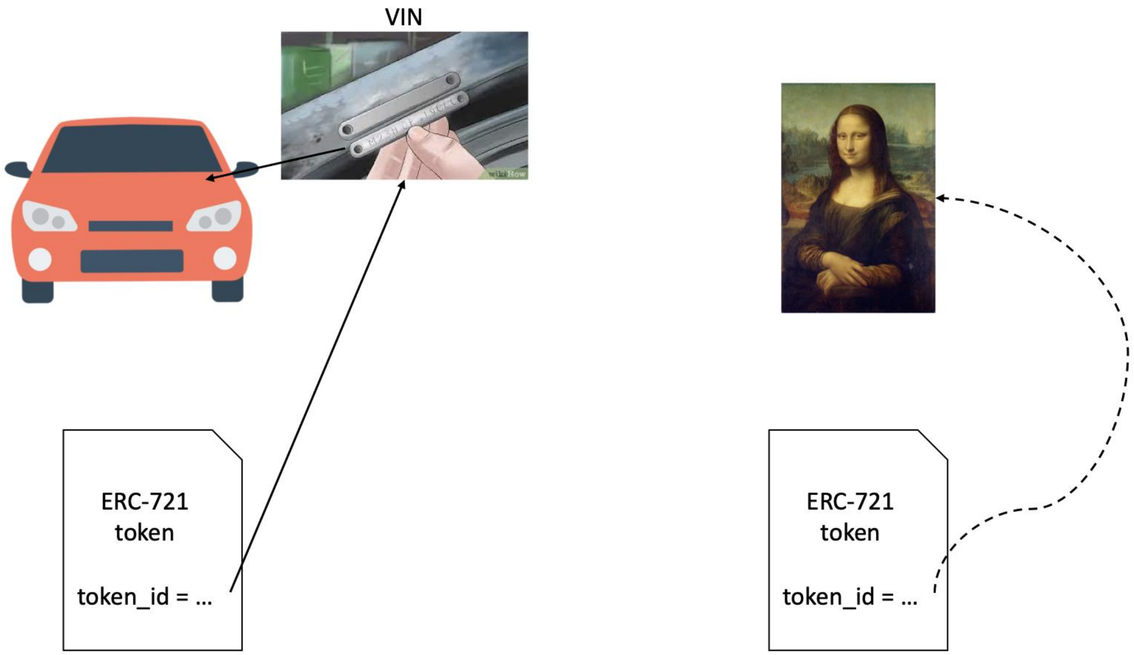
How is this link established?

How can this link be maintained forever?

Identification and Non-Fungible Tokens

Vehicle Identification Number can be used to identify a motor vehicle worldwide

How to we identify a painting?



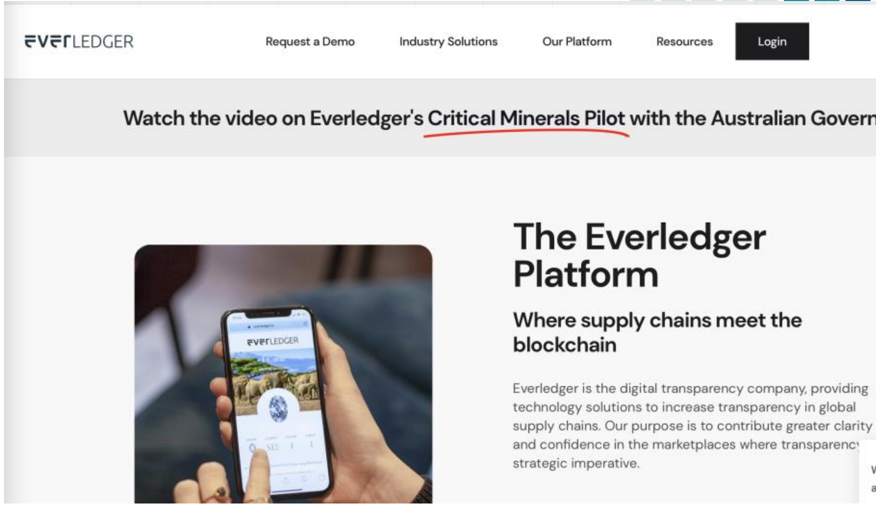
Marco Comuzzi mcomuzzi@unistac.kr

Case Study: Everledger

Solutions to improve collecting, verifying and managing information about assets in markets where transparency is a principal requirement

Use a combination of AI, IoT and, obviously, blockchain

First major use case: the diamond trade



Watch the video on Everledger's Critical Minerals Pilot with the Australian Govern

The Everledger Platform

Where supply chains meet the blockchain

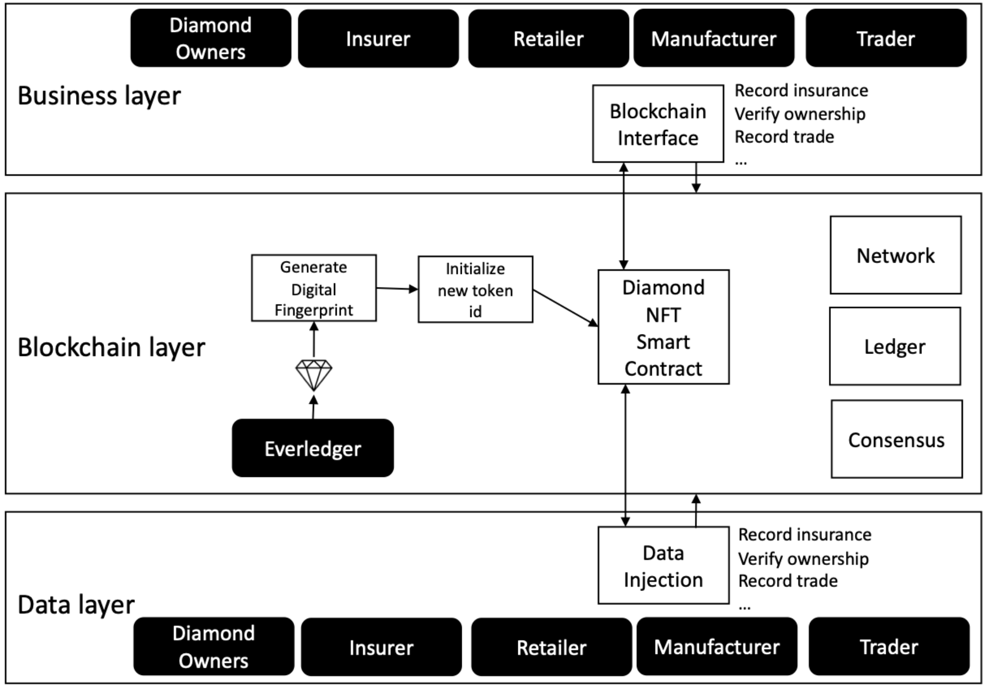
Everledger is the digital transparency company, providing technology solutions to increase transparency in global supply chains. Our purpose is to contribute greater clarity and confidence in the marketplaces where transparency strategic imperative.

Everledger in the diamond trade

Generate a unique id for each diamond

Initialize a non-fungible token using the unique id

Owners, vendors, traders etc. record changes in the smart contract



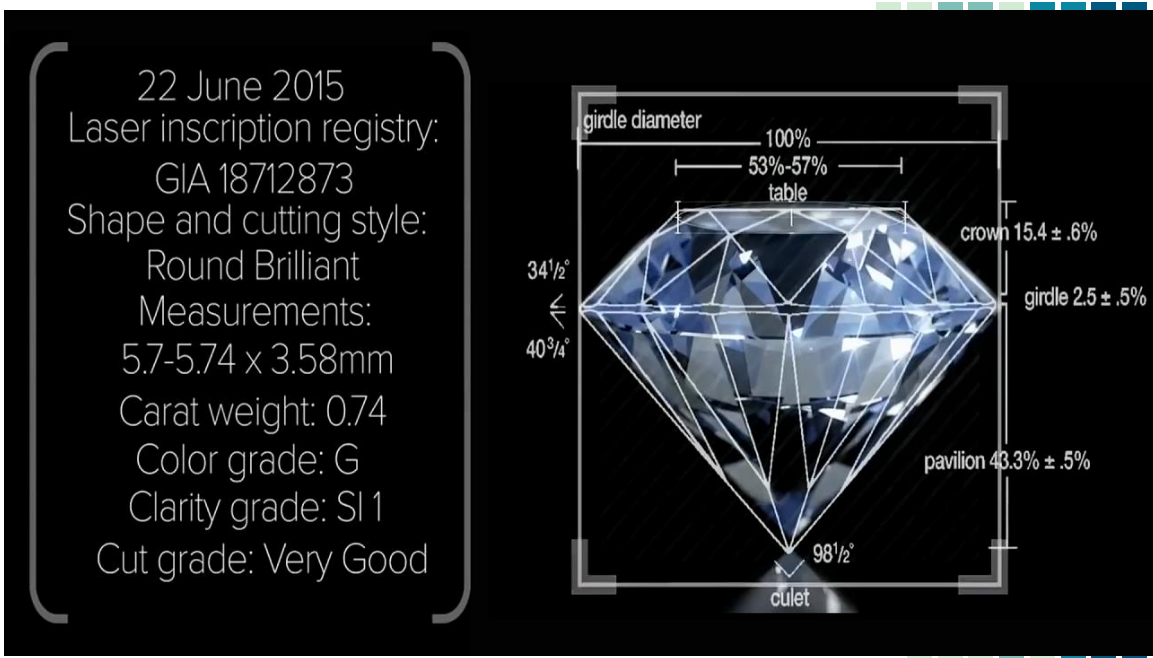
Everledger and counter-party risk

Unique id is a hash of unique measurement properties of each diamond

Easily measurable

Hard to forge

Maintained by a diamond forever



22 June 2015
Laser inscription registry:
GIA 18712873
Shape and cutting style:
Round Brilliant
Measurements:
5.7-5.74 x 3.58mm
Carat weight: 0.74
Color grade: G
Clarity grade: SI 1
Cut grade: Very Good

Counter-party risk for digital assets?

Fully "digital" assets suffer much less from counter-party risk

- Digital paintings
- Digital collectible stickers and sports cards
- …

These assets are "bits"

A "hash" of these assets can be used to uniquely identify them

Non-Fungible Tokens … or NFTs

The market of so-called NFTs has skyrocketed in 2021 and 2022

What are NTFs?

Ethereum Non-Fungible Tokens

A good explainer: https://youtu.be/H3TABd_nBJU

Implementation of Tokens

Ethereum defines a set of standard interfaces (ERC) for tokens

“Interface” as in Object-Oriented Programming

Define the signature of the “functions” that a token must implement

ERC-20: standard for fungible tokens

ERC-721: standard for non-fungible tokens

The ERC-20 standard

Function	Normal expected behavior
name()	Returns the name of this token, e.g. the 'Comuzzi-Grefen-Meroni Token'
symbol()	Returns a short name of this token. This is normally used in asset exchange markets to identify the token, e.g. CGM
totalSupply()	Returns the number of units of this token created from its creation.
balanceOf(address_owner)	Returns the number of units of this token owned by the node address-owner
transfer(address_to, value)	Allows the caller (i.e., the node issuing a transaction invoking this function of the token smart contract) to transfer a certain amount of tokens that they own (value) to a recipient node (address_to)

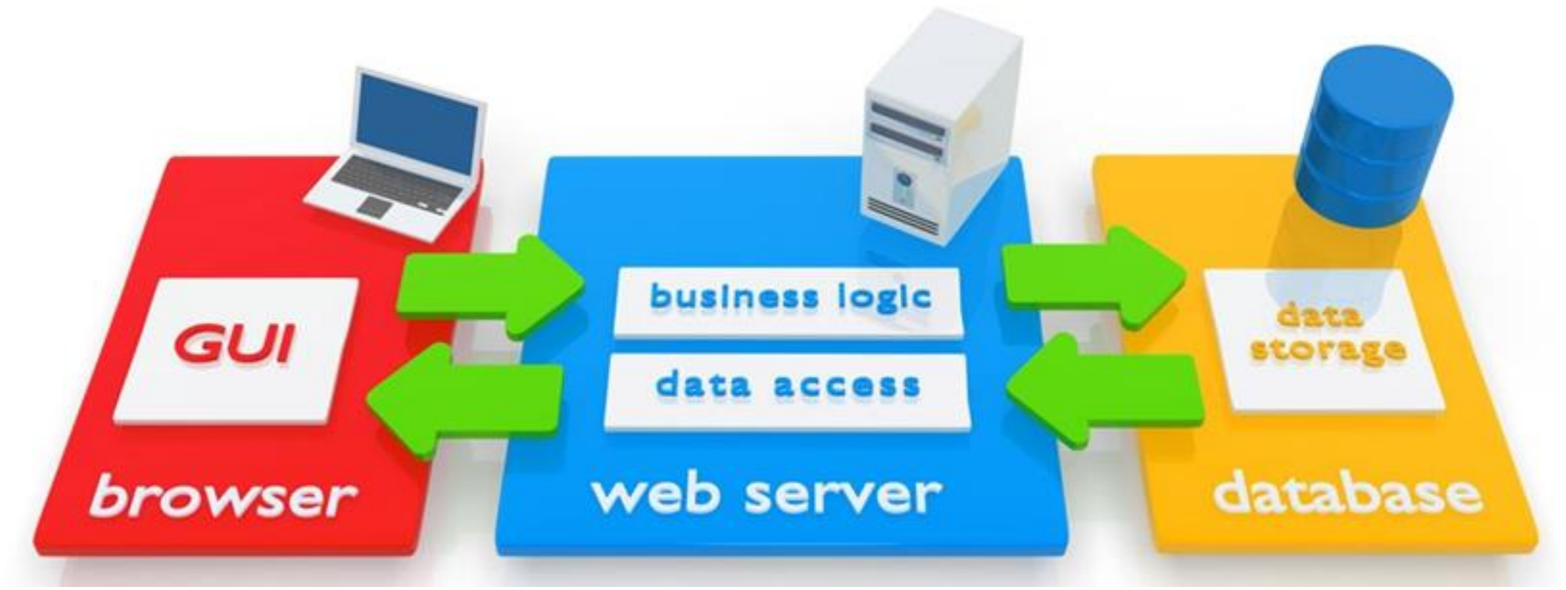
The ERC-721 standard

Function	Normal expected behavior
name()	Returns the name of this token, e.g. the ‘Comuzzi-Grefen-Meroni Token’
symbol()	Returns a short name of this token. This is normally used in asset exchange markets to identify the token, e.g. CGM
totalSupply()	Returns the number of units of this token created from its creation.
balanceOf(address_owner)	Returns the number of units of this token owned by the node address-owner
transfer(address_to, value)	Allows the caller (i.e., the node issuing a transaction invoking this function of the token smart contract) to transfer a certain amount of tokens that they own (value) to a recipient node (address_to)

Let’s make our own ERC-20 Token

UBST (UNIST Blockchain Systems Token)

4.
Ethereum Dapps



FRONT-END BACK-END DATA STORAGE

Dapp = Decentralised application

A software application that is mostly or entirely decentralised to run on a blockchain

- Application logic
- Frontend software
- Data storage

Most common case is distributed application logic, executed by a (set of) smart contract(s)

Back-end

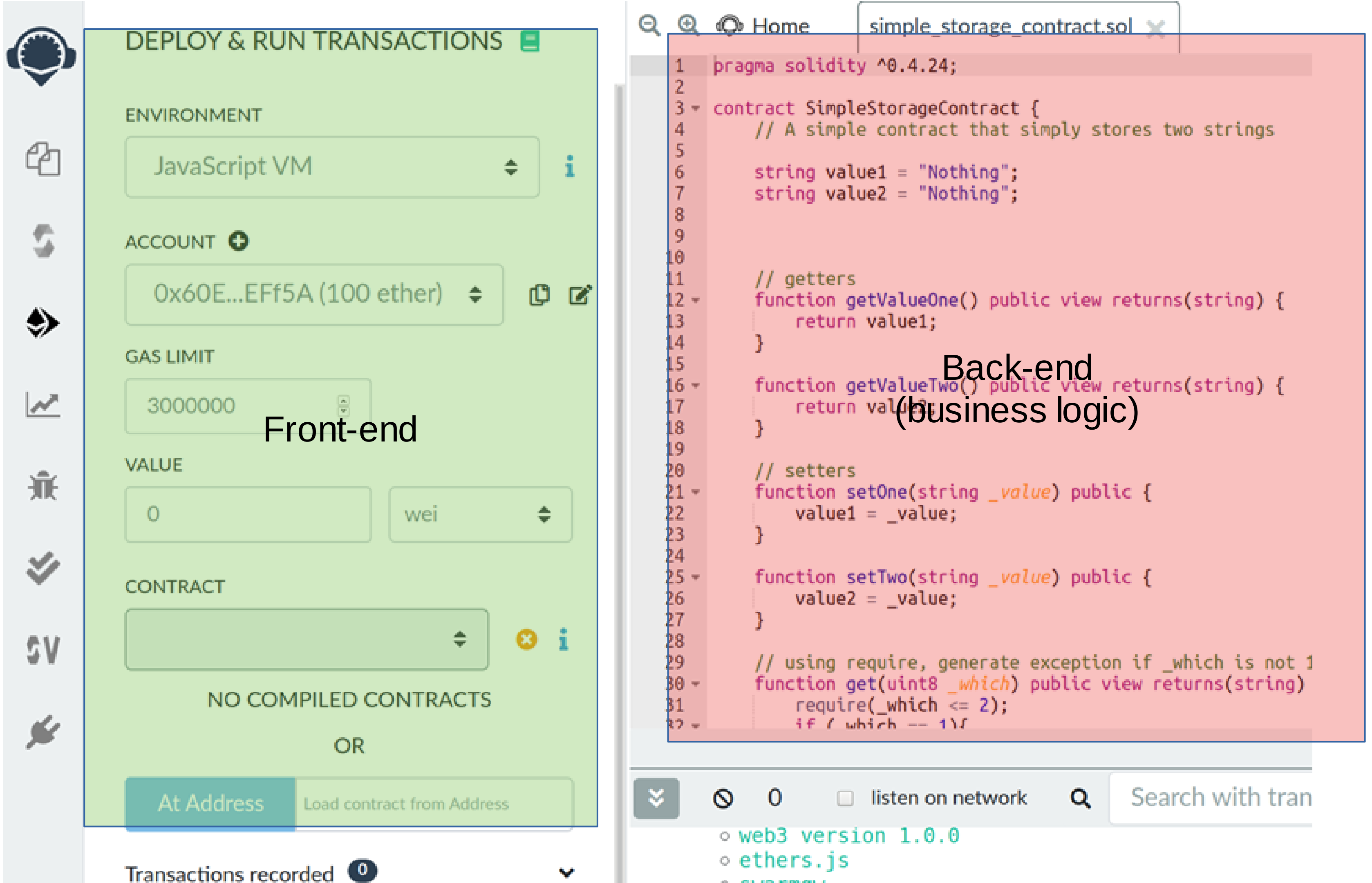
- Encoded into smart contracts

Front-end

- Developed as traditional Web app
- Example: web3.js Javascript library for connecting a browser to an Ethereum EOA wallet

Data storage

- Utilise off-chain data storage to avoid high costs
- IPFS, Swarm (distributed file systems)



```
pragma solidity ^0.4.24;

contract SimpleStorageContract {
    // A simple contract that simply stores two strings

    string value1 = "Nothing";
    string value2 = "Nothing";



    // getters
    function getValueOne() public view returns(string) {
        return value1;
    }

    function getValueTwo() public view returns(string) {
        return value2;
    }

    // setters
    function setOne(string _value) public {
        value1 = _value;
    }

    function setTwo(string _value) public {
        value2 = _value;
    }

    // using require, generate exception if _which is not 1
    function get(uint8 _which) public view returns(string)
        require(_which <= 2);
        if (_which == 1){
```

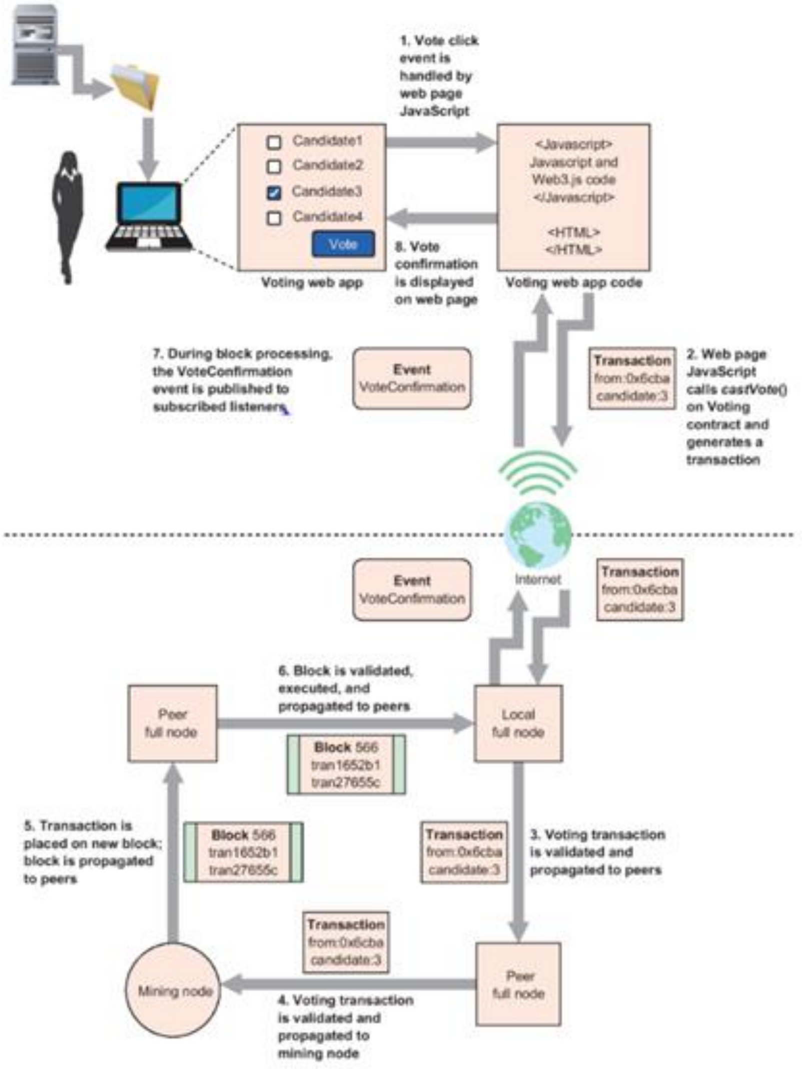
```
o web3 version 1.0.0
o ethers.js
```

NO off-chain data!

Voting app

Web app interacts with Ethereum networks by submitting transactions

- "Data" of transactions are managed by the Dapp business logic
- Transactions are validated and mined as any other Ethereum transaction


1. Vote click event is handled by web page JavaScript
Candidate1
Candidate2
Candidate3
Candidate4
Vote
Voting web app
<Javascript> Javascript and Web3.js code </Javascript> <HTML> </HTML>
Voting web app code
8. Vote confirmation is displayed on web page
7. During block processing, the VoteConfirmation event is published to subscribed listeners
Event VoteConfirmation
Transaction from:0x6cba candidate:3
2. Web page JavaScript calls castVote() on Voting contract and generates a transaction
Internet
Event VoteConfirmation
Transaction from:0x6cba candidate:3
6. Block is validated, executed, and propagated to peers
Peer full node
Local full node
Block 566 tran1652b1 tran27655c
5. Transaction is placed on new block; block is propagated to peers
Block 566 tran1652b1 tran27655c
Transaction from:0x6cba candidate:3
3. Voting transaction is validated and propagated to peers
Transaction from:0x6cba candidate:3
Mining node
Peer full node
4. Voting transaction is validated and propagated to mining node

Dapps: Pros and Cons

Pros	
Deterministic and verifiable business logic execution	A smart contract code is received and can be inspected by every node. Therefore, the application logic of a Dapp is verifiable by any user. Also, smart contracts only allow the execution of deterministic logic (no random numbers allowed!), which means that by inspecting the code users know exactly how the Dapp is going to behave.
Zero downtime (decentralization)	The execution of the application logic is decentralized across the Ethereum network. Users do not have to wait for a server somewhere (which may become unavailable) to execute their input. Practically, this means that a Dapp has no downtime: a transaction invoking a Dapp's smart contract is sooner or later picked up by a miner to be included in a block.
Resistance to censorship	Because it relies on the decentralized paradigm of the Ethereum blockchain, a Dapp is highly resistant to censorship. While a Dapp hosted on a (cloud) server may be taken down by a hostile government, it is virtually impossible that a single institution could take down the whole Ethereum network. An Ethereum Dapp stays alive as long as there is still at least one node somewhere in the world running the Ethereum protocol and connected to the Internet.
Immutability of data	All the interactions between the users and a Dapp (i.e., transactions) and the data that they generate (i.e., changes of the Ethereum state), are immutably recorded on the Ethereum blockchain. As such, they remain available forever to be consulted, for instance for auditing reasons or to resolve disputes or, simply, to verify their existence.
Cons	
Cost	Issuing transactions on the Ethereum network is not free of charge. Every time users invoke a function of a Dapp, they have to pay a fee in ETH. The price of ETH in fiat currencies is extremely volatile and, depending on the current exchange rates, the fees may not be negligible in many business settings. To solve the cost issues, Dapps often choose to store only the most important data on the Ethereum blockchain. Other data can be stored off-chain on distributed file system solutions, such as IFPS.
Performance	Every interaction of the users with a Dapp cannot be trusted until it is stored in a block deep enough in the Ethereum main chain. Even though Ethereum is quicker than Bitcoin from this point of view, the time required to complete the invocation of a Dapp's function may exceed the limits considered acceptable in many business settings. Moreover, the performance of the Ethereum blockchain is not particularly reliable. The time it takes for a transaction to be picked up by a miner may fluctuate unpredictably depending on the network load condition and the location or time zone of the nodes issuing the transactions.
Maintenance and regression to centralization	From a technical standpoint, the tools provided by the Ethereum protocol to manage the evolution of smart contracts are very limited. Basically, once a smart contract is deployed, it cannot be modified. A new version of it, where even only one small instruction is updated, must be deployed as a new smart contract. These may create enormous maintenance problems for Dapps. To overcome this issue, often a Dapp delegates to a smart contract only the core part of its application logic, which is not likely to change in the near future. The remaining application logic can be delegated to other (non-Dapp) applications, such as other Web applications. This, however, means moving away from the blockchain paradigm: any other traditional application, in fact, may be seen as a centralized intermediary.

THANKS!

https://sites.google.com/site/marcocomuzziphd

http://iel.unist.ac.kr/

You can find me at:

@dr_bsad

mcomuzzi@unist.ac.kr