Blockchain Mechanism and Platforms (1 of 2)

KHU

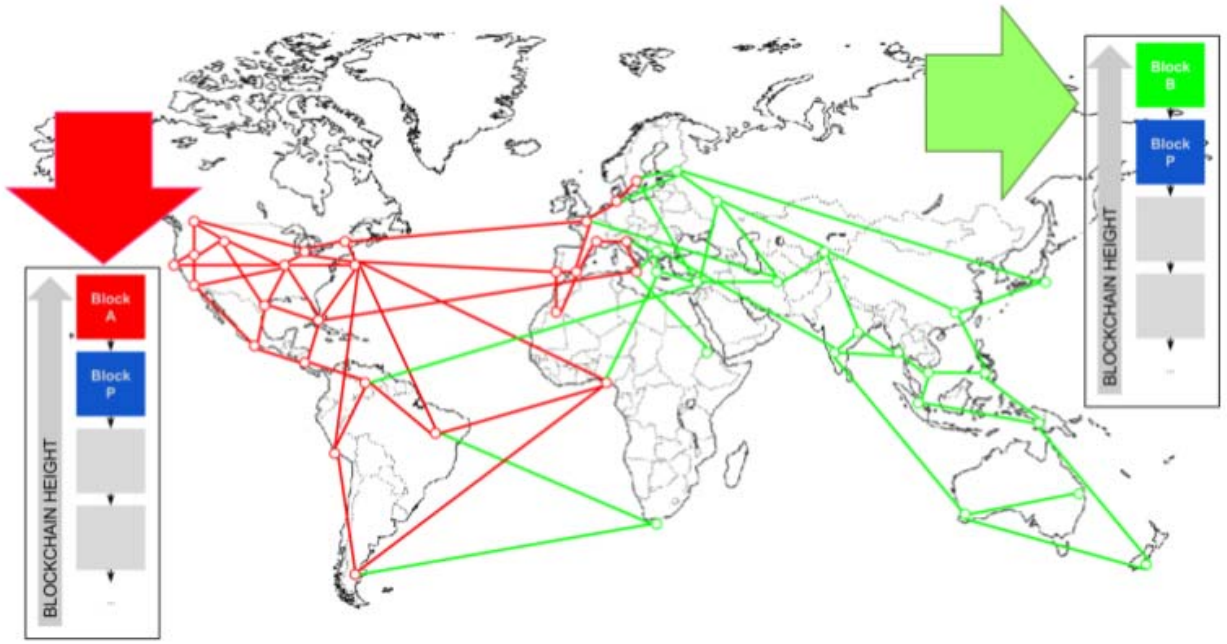
경희대학교
KYUNG HEE UNIVERSITY

Contents

Blockchain Mechanism

1. Blockchain Mechanism: solve collisions of blockchain

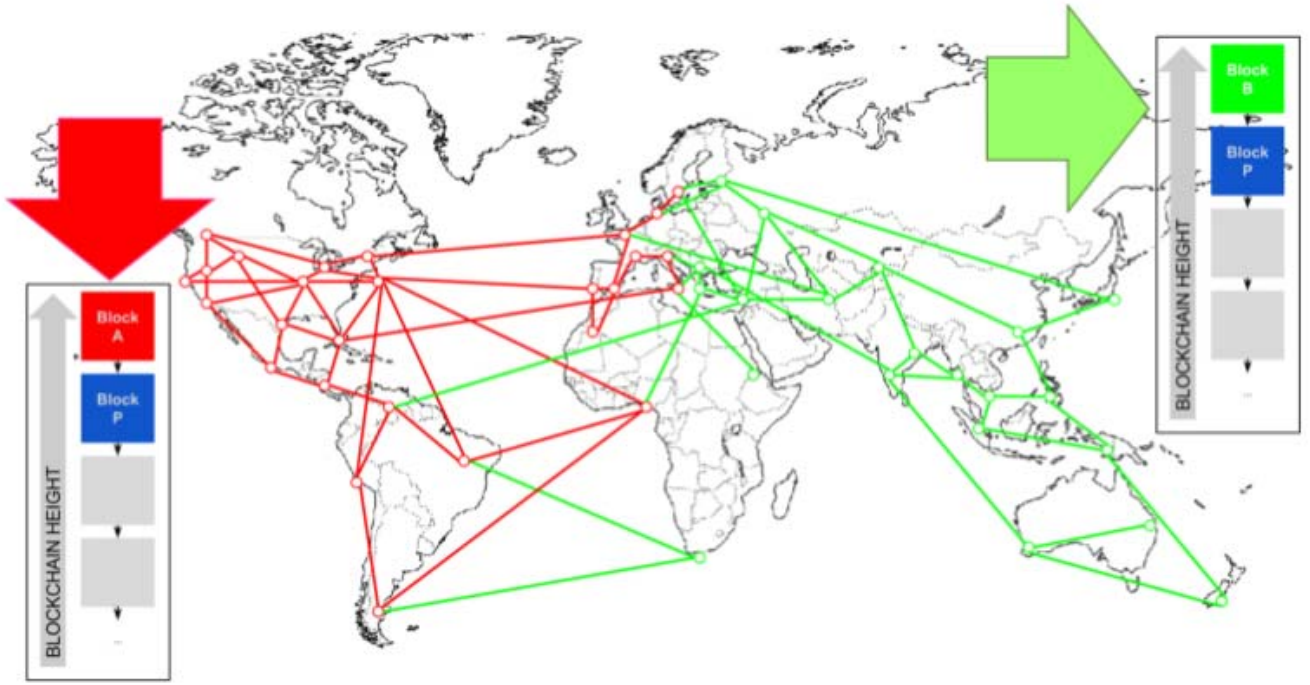
- Far distant nodes find the nonce value at almost the same time, add each new block to the last block P, propagate to neighbor nodes



source : https://homoefficio.github.io

1. Blockchain Mechanism: solve collisions of blockchain

- Nodes that received the green block first ignore the red block propagated, and vice versa



Source : https://homoefficio.github.io

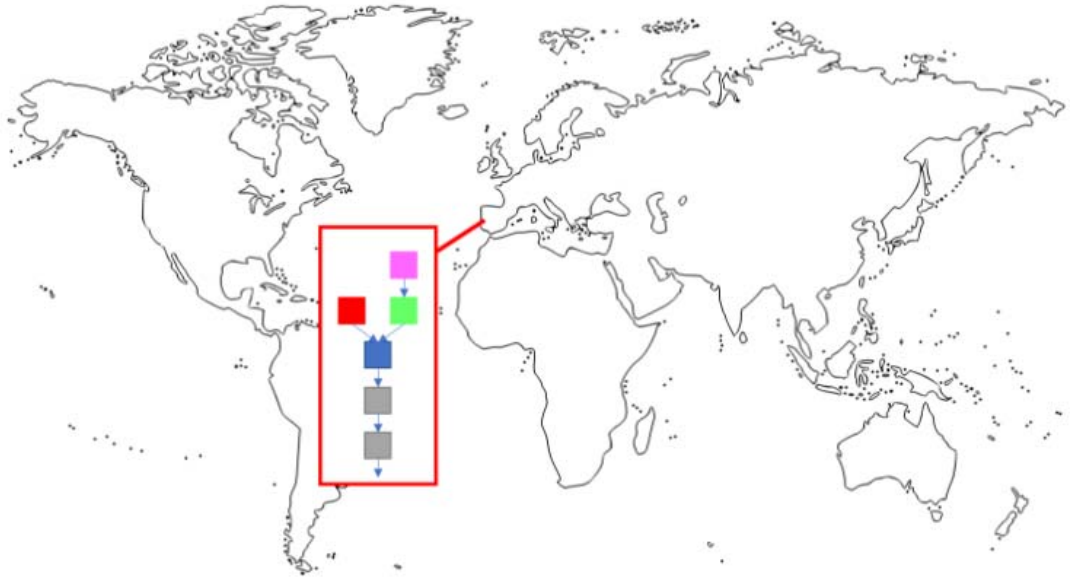
1. Blockchain Mechanism: solve collisions of blockchain

- The nonce value is obtained from the node which inherits the green block of the intermediate point and propagates the purple block

Source : https://homoefficio.github.io

1. Blockchain Mechanism: solve collisions of blockchain

- If a branch occurs at a node that has formed a red block, it is replaced with a longer green-purple block chain



Source : https://homoefficio.github.io

1. Blockchain Mechanism: Consensus Algorithm

- **Proof-of-Work (PoW)** : Bitcoin, Ethereum, Litecoin
  - **Pros:** Very secure
  - **Cons:** Slow throughput, expensive computations
- **Proof-of-Stake (PoS)** : Dash, Stratis, NAV Coin, Peercoin, Decred, Nxt, Nova Coin
  - **Pros:** Attacks more expensive, energy efficient
  - **Cons:** Prone to centralisation
- **Delegated Proof-of-Stake (DPoS)** : Steemit, BitShares, EOS, Lisk, Ark, BitShares, Ethereum Casper, Tendermint, Slasher
  - **Pros:** Cheap transactions, scalable, energy efficient
  - **Cons:** Partially centralized
- **Proof-of-Authority (PoA)** : POA Network, Ethereum Kovan/Rinkeby testnet
  - **Pros:** Simple, Cost efficient, High throughput, scalable
  - **Cons:** Centralized
- **Byzantine Fault Tolerance (BFT)** : Hyperledger, NEO, Stellar, Ripple, Dispatch
  - **Pros:** High throughput, Transaction finality, Cost efficient, scalable
  - **Cons:** Centralized, Semi-trusted
  - Variantes: Practical Byzantine Fault Tolerance (PBFT) : Hyperledger, Federated Byzantine Agreement (FBA) : Stellar, Ripple, Delegated Byzantine Fault Tolerance (dBFT)

https://hackernoon.com/a-hitchhikers-guide-to-consensus-algorithms-d81aae3eb0e3

1. Blockchain Mechanism: Consensus Algorithms

- In the case of bitcoin, proof-of-work algorithm is used. In the case of Etherium, proof-of-work(PoW) and proof of stake(PoS) algorithm are tested in hybrid form from August, 2017. Going forward, the Etherium aims to convert to POS
- Proof of Work (PoW)
  - In the blockchain, the most commonly used consensus algorithm is to compute the nonce value by hashing the hash value of the specific difficulty using computational power and verify the nonce value.
- Proof of Stake (PoS)
  - Algorithm developed to solve the problem of waste of computing power of PoW is to distribute authority based on assets possessed by nodes and to obtain agreement and distribute compensation
- There are various algorithms such as Proof of Elapsed Time (PoET)

1. Blockchain Mechanism: Consensus Algorithms

- Private Blockchain uses PBFT and PAXOS algorithms in general, and Quorum, which is a representative project of Enterprise Ethereum, adopts Raft algorithm.

- PAXOS
  - o Selecting Leaders with the most common consensus algorithm and agreeing with majority consent

- Practical Byzantine Fault Tolerance (PBFT)
  - It is widely used in the private block chain because of the consensus derived from the three-step protocol that adopted the voting mechanism as a consensus algorithm designed to solve the Byzantine general problem

- Raft
  - Complemented PAXOS, which simplifies the process by electing the leader through voting and random timeout

1. Blockchain Mechanism: Uncertainty of PoW Finality

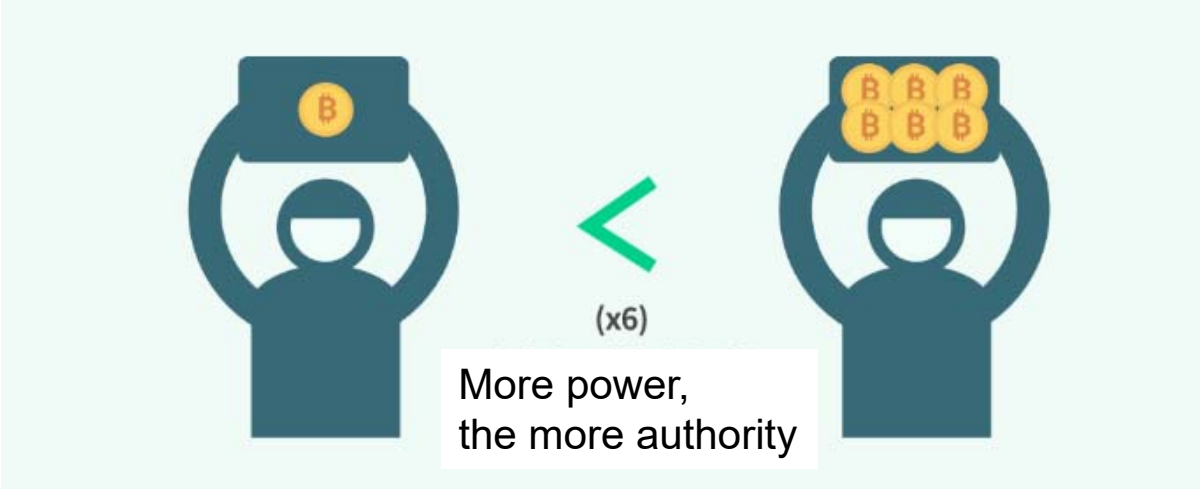
- PoW judges that the long chain is correct when the blockchain branches
- If a short chain is discarded, it can happen that there is no transaction.
- In order to prevent such a phenomenon in the case of bitcoin, it is restricted such that it waits for 6 blocks even if the transaction is confirmed.

1. Blockchain Mechanism: Limitation of PoW Performance

- It is not possible to eliminate the spreading time to the blockchain network sharing a single information in a P2P network
- It requires time to set the consensus because it guarantees the reliability of information through consensus among several nodes.
- Therefore, it is difficult to raise performance (response time and throughput)
- It takes about 10 minutes to generate the block, so real-time property is not guaranteed

1. Blockchain Mechanism: PoS (Proof of Stake)

- Unlike the previous PoW, it is called the proof of stake. It generates blocks based on the participant's coin share, not the computer's hash power.
- If participants‘ stake is larger, the share of the coin will be more.



Source : https://brunch.co.kr/@banksalad/313

1. Blockchain Mechanism: Delegated Proof of Stake(DPoS)

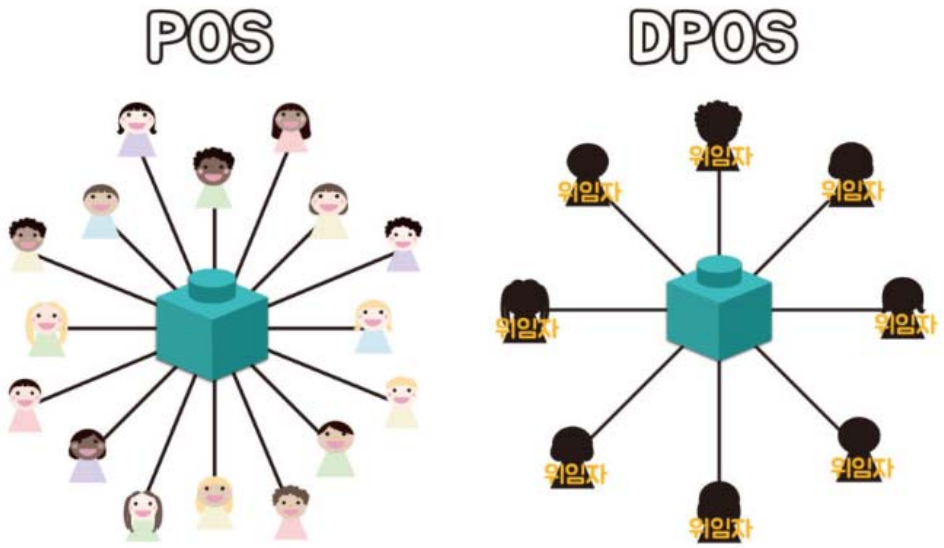
- Delegation It may be called proof of stake and delegating authority to allow PoS only for specific persons (entities)

- Becoming a representative by delegating authority to an elected parent node as a result of voting on nodes in the network

- Allocates revenue with delegated representative

1. Blockchain Mechanism: DPoS

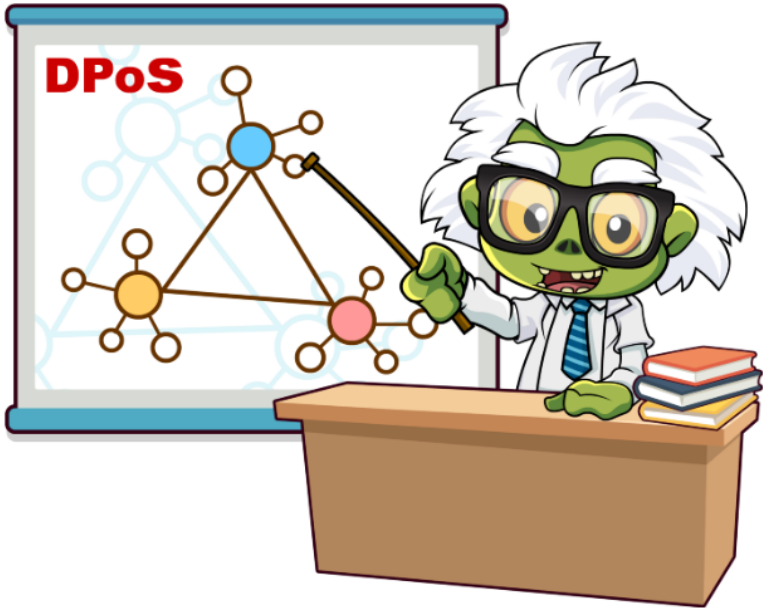
- In the case of PoS, it takes a long time for all nodes that have a certain stake to be granted block generation rights
- In the case of DPoS, the time and cost of consensus are reduced due to the relatively small number of nodes designated as the voting result



Source : https://brunch.co.kr/@mobiinside/1163

1. Blockchain Mechanism: DPoS Block Representives

- EOS : 21
- BitShares : 101
- Steemit : 21
- Lisk : 101
- Ark : 51

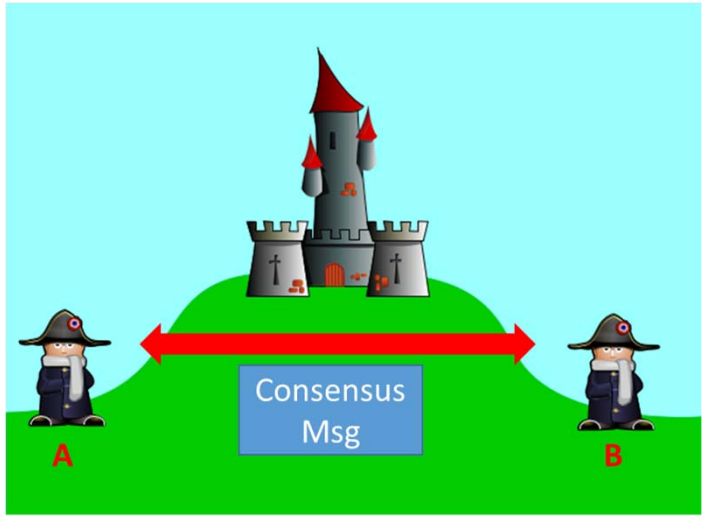


https://medium.com/loom-network/understanding-blockchain-fundamentals-part-3-delegated-proof-of-stake-b385a6b92ef

PBFT
PBFT(Practical Byzantine Fault Tolerance)

Two Generals' Problem

- Both generals A and B must attack the enemy at the same time, and the enemy can be captured.
  - Both generals have enemy units in between, and consensus messages can only arrive if they pass an enemy
- Case that A sends an attack time agreement message to B
  - Also, it is not possible to confirm whether the message of A was modulated by the enemy
- Case that B responds to A
  - It is not possible to agree between A and B
- It is not possible to agree between A and B



Byzantine Empire

Src: https://www.slideshare.net/YongRaeJo/pbft-86070872

BGP (Byzantine Generals Problem)

- Paper
  - "The Byzantine Generals Problem “, Lamport, L.; Shostak, R.; Pease, M. (1982). ACM Transactions on Programming Languages and Systems

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

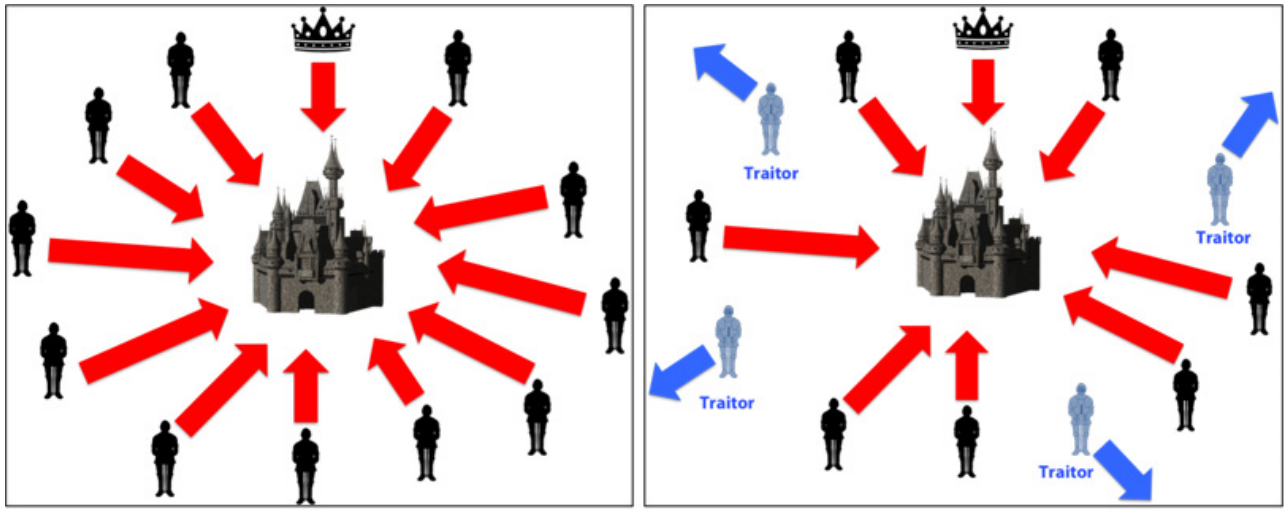
Categories and Subject Descriptors: C.2.4. **[Computer-Communication Networks]**: Distributed Systems—*network operating systems*; D.4.4 **[Operating Systems]**: Communications Management—*network communication*; D.4.5 **[Operating Systems]**: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

BGP (Byzantine Generals Problem)

Study how to agree when there is a malicious general (Byzantine) among the generals

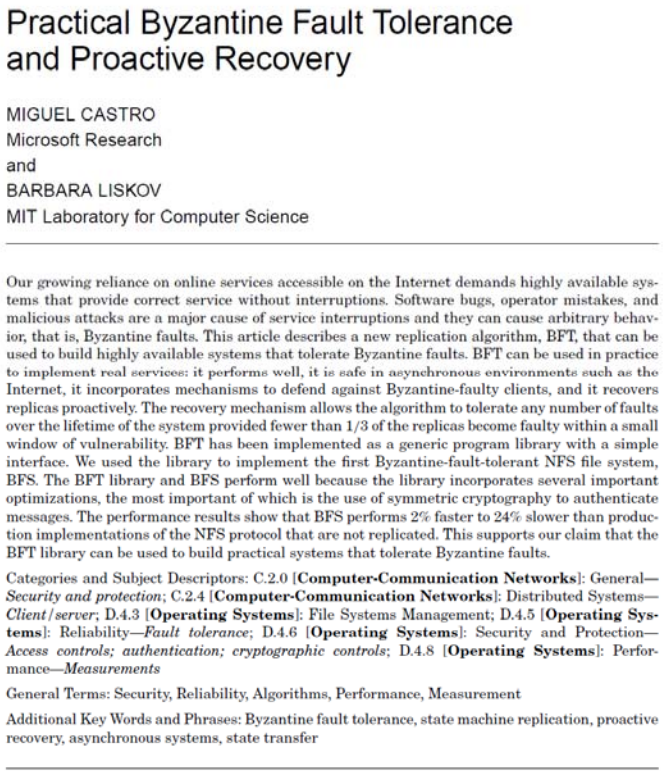


Coordinated Attack Leading to Victory

Uncoordinated Attack Leading to Defeat

PBFT (Practical Byzantine Fault Tolerance)

- **Paper**
  - **"Practical Byzantine Fault Tolerance and Proactive Recovery “, Castro, M.; Liskov, B. (2002). ACM Transactions on Computer Systems.**
- **N = 3f + 1**
  - **N = Number of All Network nodes**
  - **f = Number of Byzantine nodes**



Practical Byzantine Fault Tolerance and Proactive Recovery

MIGUEL CASTRO
Microsoft Research
and
BARBARA LISKOV
MIT Laboratory for Computer Science

Our growing reliance on online services accessible on the Internet demands highly available systems that provide correct service without interruptions. Software bugs, operator mistakes, and malicious attacks are a major cause of service interruptions and they can cause arbitrary behavior, that is, Byzantine faults. This article describes a new replication algorithm, BFT, that can be used to build highly available systems that tolerate Byzantine faults. BFT can be used in practice to implement real services: it performs well, it is safe in asynchronous environments such as the Internet, it incorporates mechanisms to defend against Byzantine-faulty clients, and it recovers replicas proactively. The recovery mechanism allows the algorithm to tolerate any number of faults over the lifetime of the system provided fewer than 1/3 of the replicas become faulty within a small window of vulnerability. BFT has been implemented as a generic program library with a simple interface. We used the library to implement the first Byzantine-fault-tolerant NFS file system, BFS. The BFT library and BFS perform well because the library incorporates several important optimizations, the most important of which is the use of symmetric cryptography to authenticate messages. The performance results show that BFS performs 2% faster to 24% slower than production implementations of the NFS protocol that are not replicated. This supports our claim that the BFT library can be used to build practical systems that tolerate Byzantine faults.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Client/server*; D.4.3 [**Operating Systems**]: File Systems Management; D.4.5 [**Operating Systems**]: Reliability—*Fault tolerance*; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls; authentication; cryptographic controls*; D.4.8 [**Operating Systems**]: Performance—*Measurements*

General Terms: Security, Reliability, Algorithms, Performance, Measurement

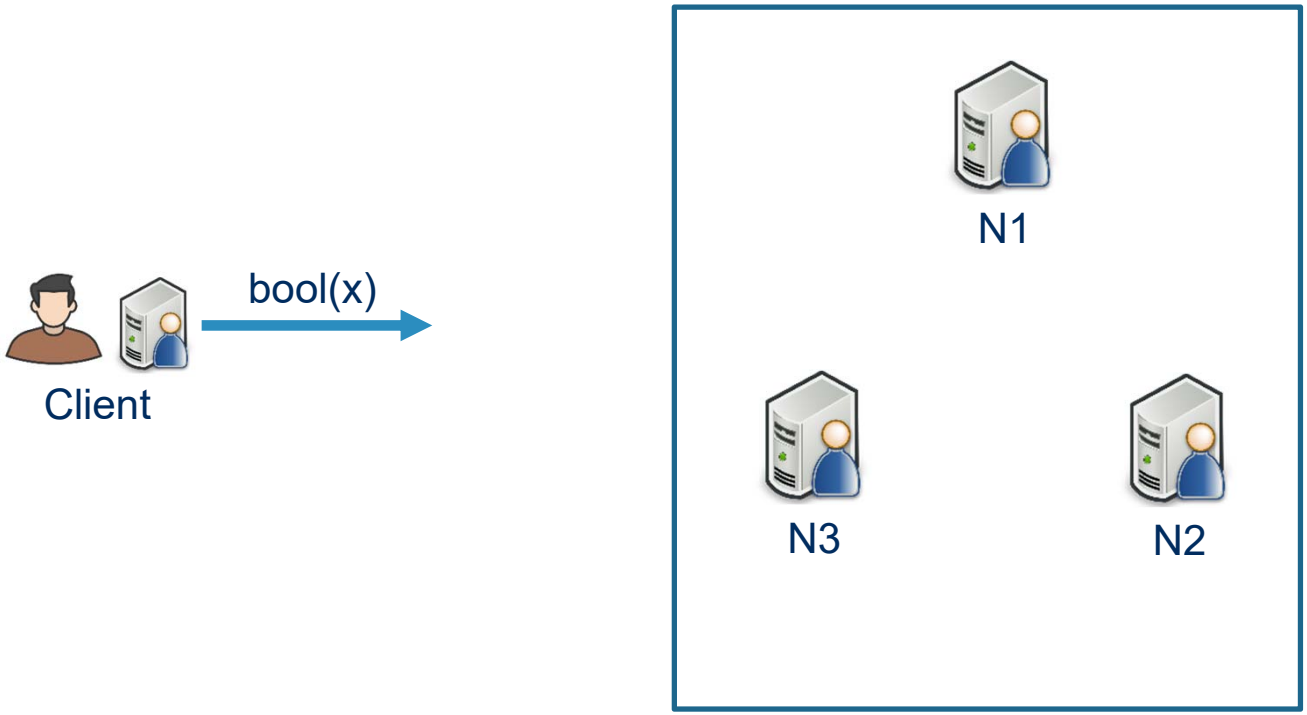
Additional Key Words and Phrases: Byzantine fault tolerance, state machine replication, proactive recovery, asynchronous systems, state transfer

BFT (Byzantine Fault Tolerance)

- **Terminology**
  - **Byzantine Fault:** System failures, or malfunctions caused by malicious attacks
  - **Byzantine Failure:** the network service interrupted by Byzantine Fault
  - **BFT Nodes:** The number of Byzantine nodes that a distributed network can operate normally

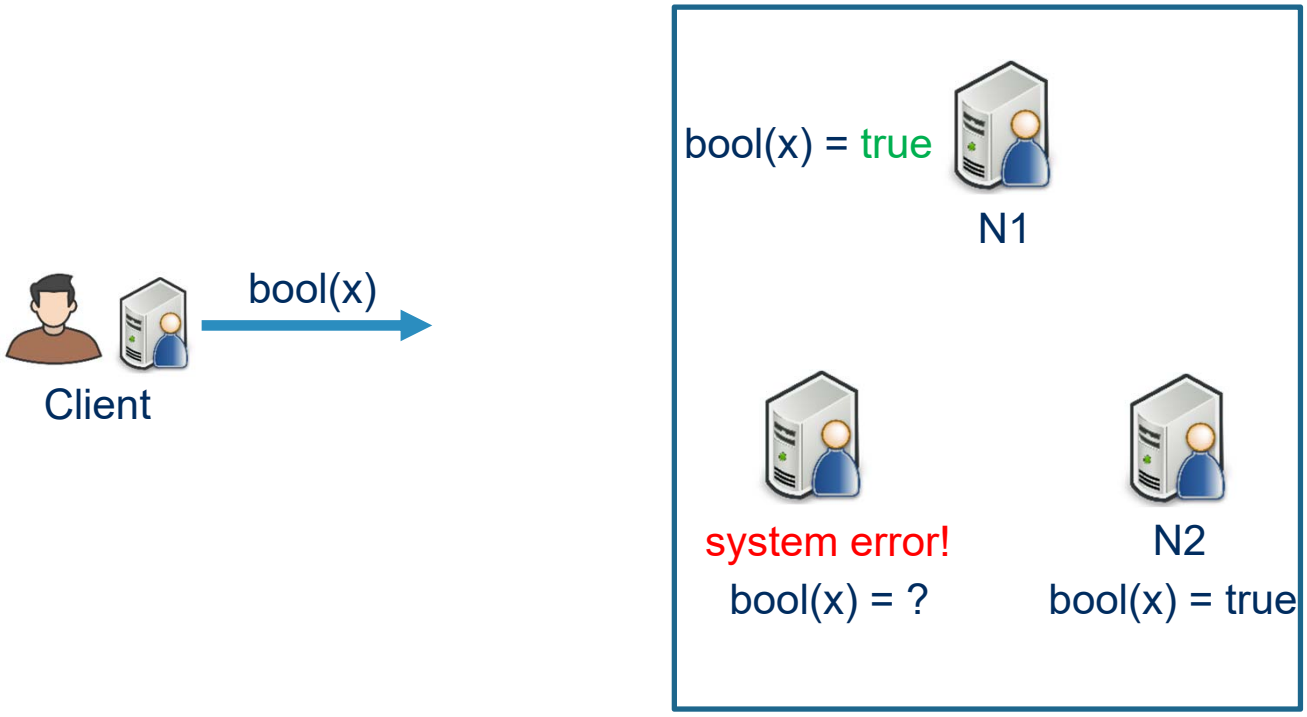
Why N = 3f + 1?

- **N = 2f + 1, f = 1 ?**



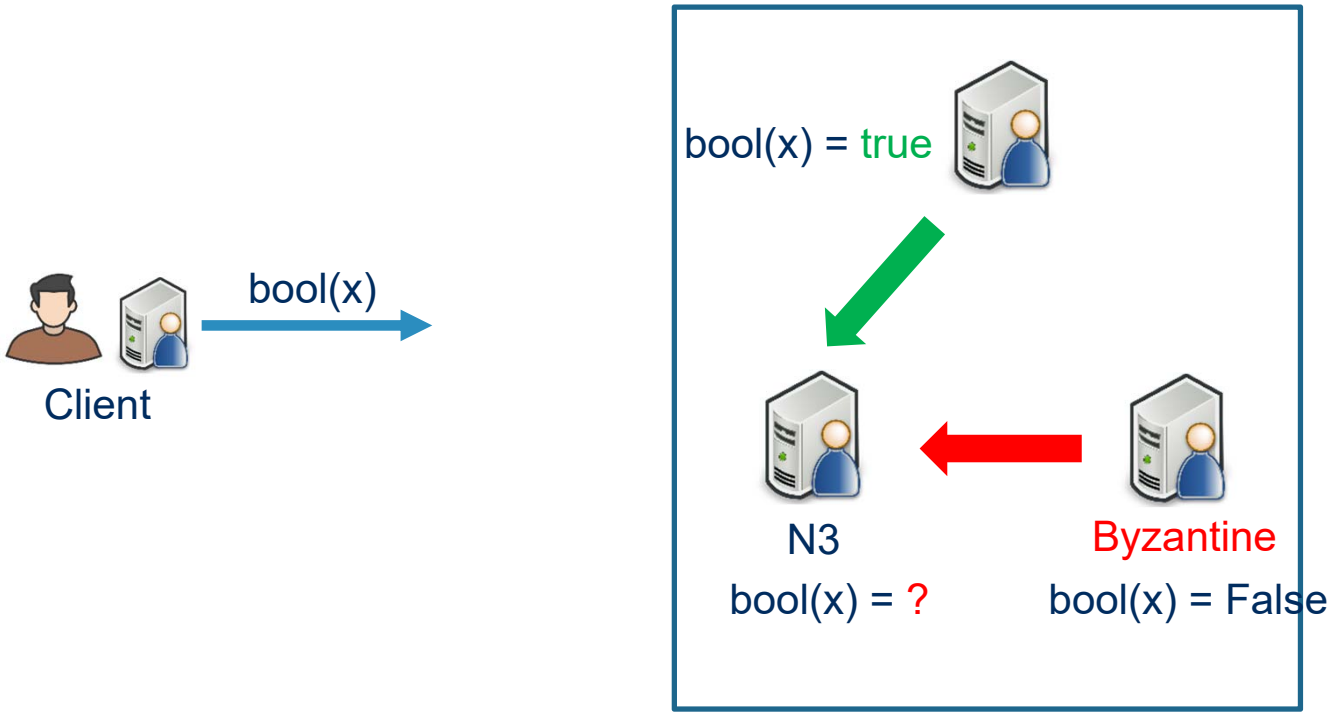
Why N = 3f + 1?

- **N = 2f + 1, f = 1 ?**



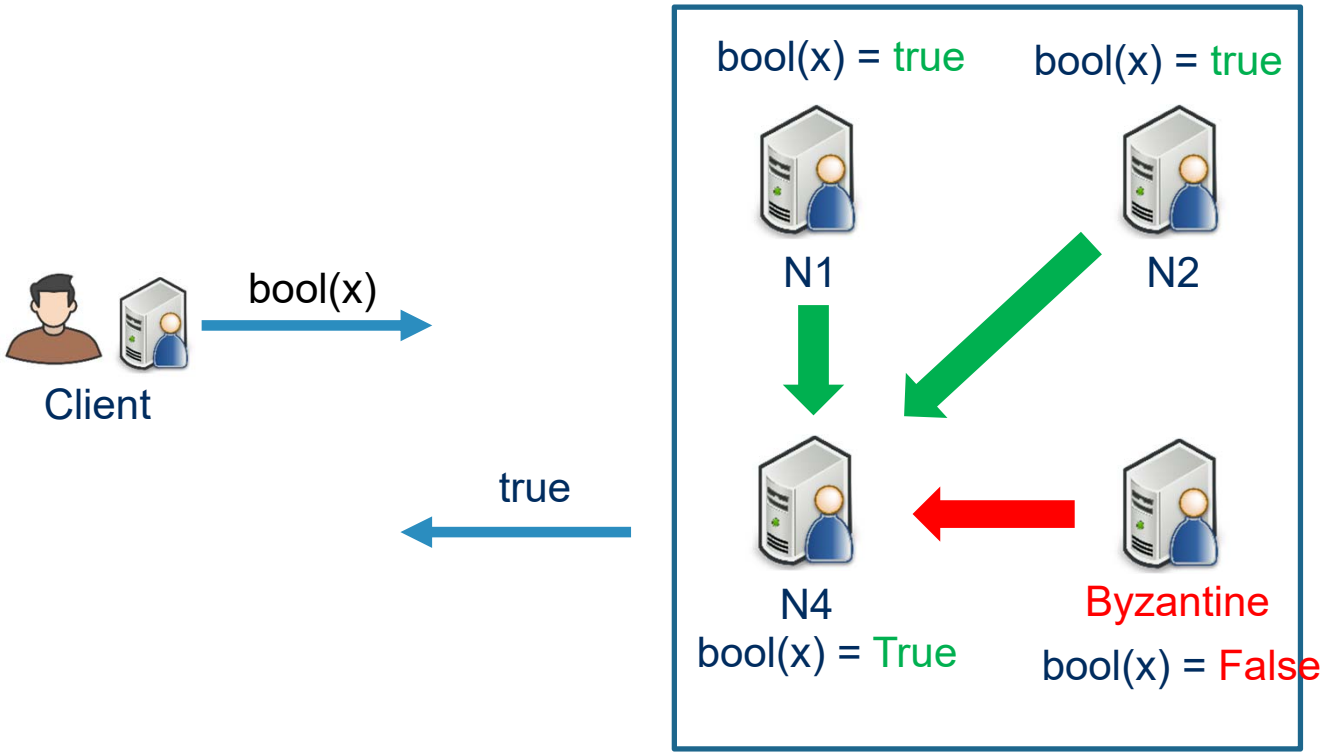
Why N = 3f + 1?

- **N = 2f + 1, f = 1 ?**



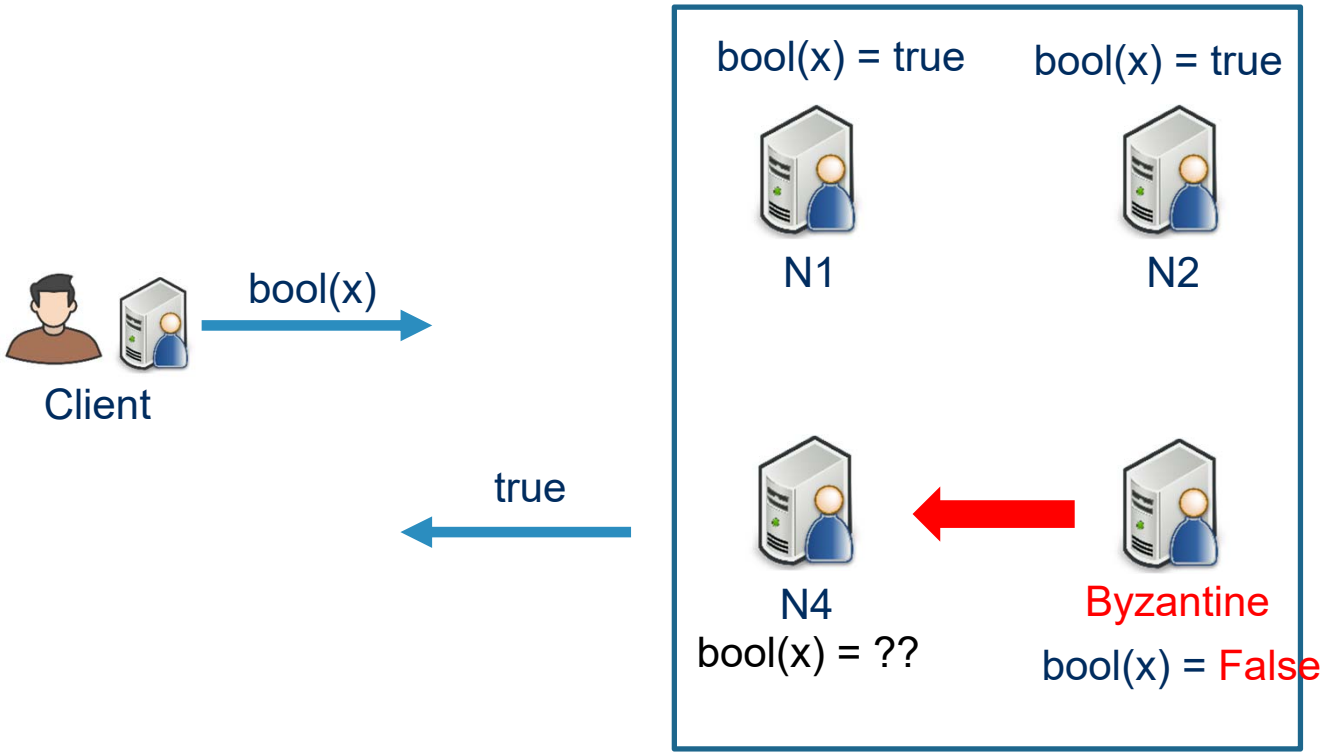
Why N = 3f + 1?

- **N = 3f + 1, f = 1 ?**



Why N = 3f + 1?

- **N = 3f + 1, f = 1 ?**



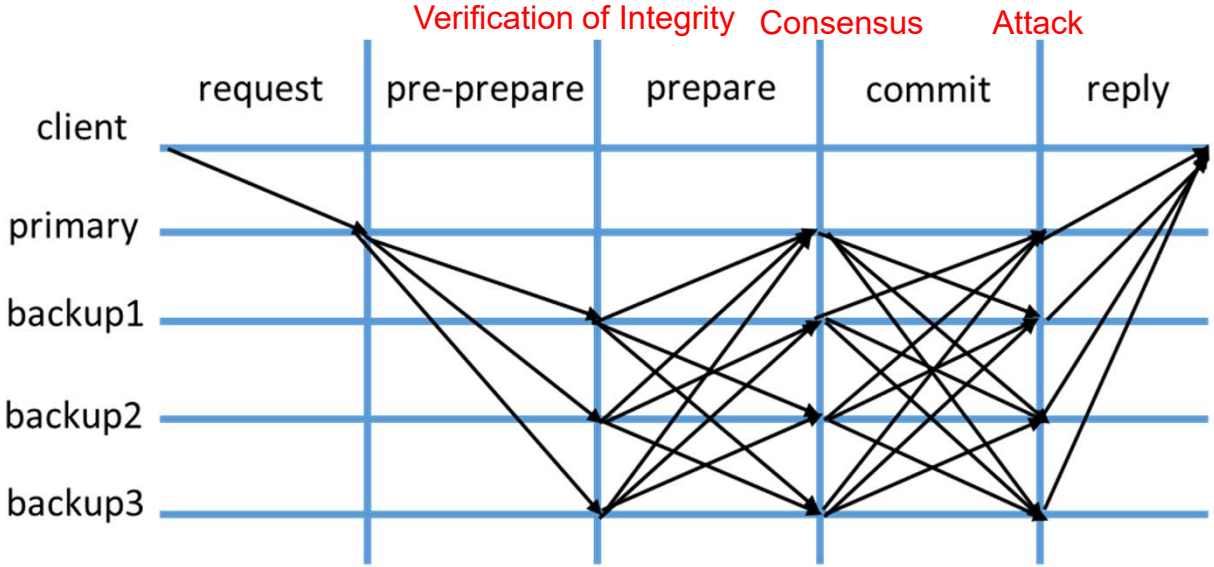
PBFT(Practical Byzantine Fault Tolerance)

- A Practical Protocol to Solve the Byzantine General Problem
  - The Byzantine General Problem: Assuming the general can do malicious acts
  - A protocol that allows the entire system to operate reliably in spite of simple malfunctioning nodes as well as malicious nodes
- The most representative protocol that can be used practically among BFT series protocols
- There is a primary node in the replica that serves as a decision-making leader, and commands are executed sequentially under the control of the primary node.
- If the primary node is malfunctioning or behaves maliciously, change the primary node through a process called 'view change'

PBFT(Practical Byzantine Fault Tolerance)

PBFT in rough

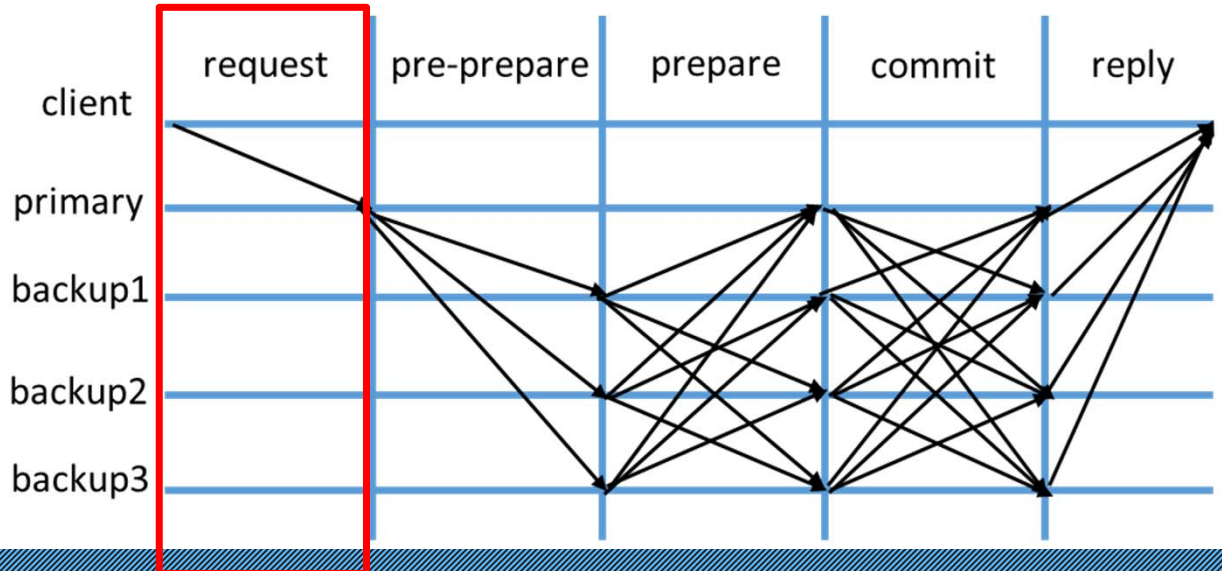
- The client sends a request to the primary node
- The primary node propagates the request to the backup nodes and performs the consensus process.
- When the consensus process is completed, the primary node and the backup nodes send a completion message to the client
- If the client receives f + 1 or more identical replies from the backup node, then the client is sure that the request is reflected correctly



PBFT(Practical Byzantine Fault Tolerance)

Request

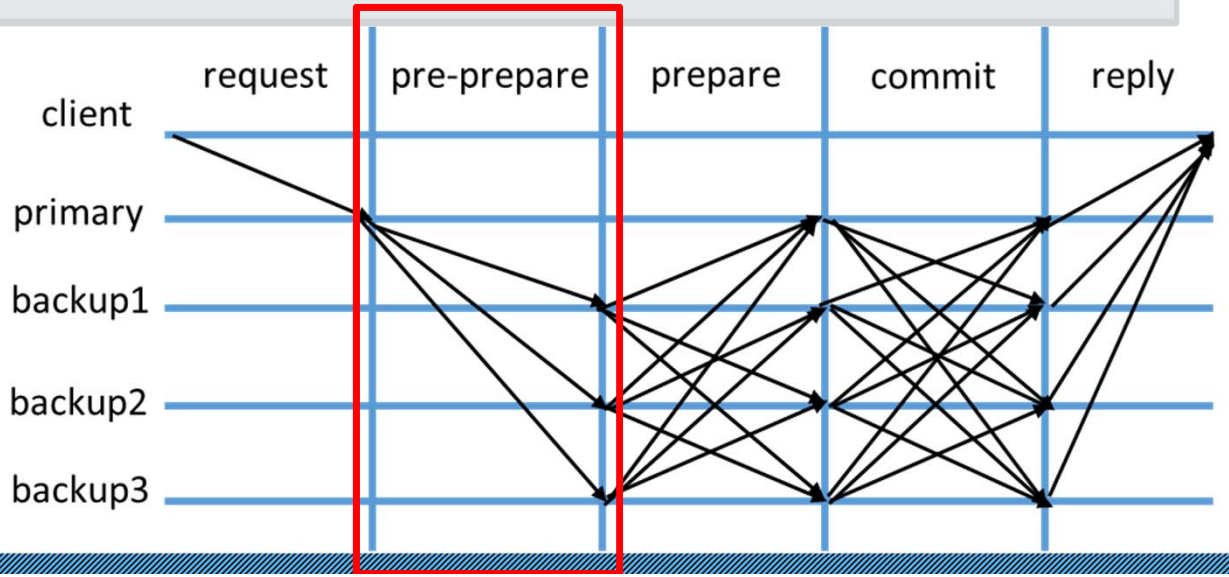
- The Client requests the Primary node to work
- **(REQUEST, o, t, c)s_c**
  - **REQUEST: Request Phase**
  - **t: Timestamp**
  - **c: Client ID**
  - **s_c: Client Signature**



PBFT(Practical Byzantine Fault Tolerance)

Pre-prepare

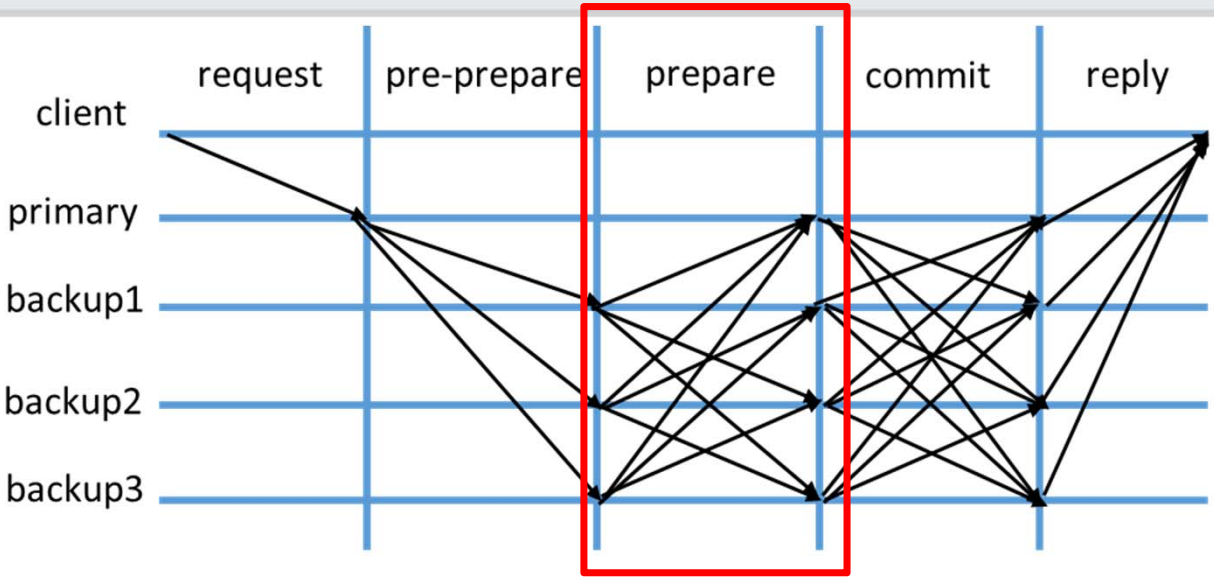
- When Primary receives a valid request, it generates the following message and sends it to the backup node
  - Backup node: All nodes except Primary and Clien
  - **((PRE-PREPARE,v,n,d)s_p, m)**
    - ✓ **PRE-PREPARE: Vrification-of-Integrity Phase**
    - ✓ **v: view number** (you can see who is currently the primary node)
    - ✓ **n: sequence number**
    - ✓ **d: m의 message digest**
    - ✓ **s_p: Primary node signature**
    - ✓ **M: message sent by client**



PBFT(Practical Byzantine Fault Tolerance)

Prepare

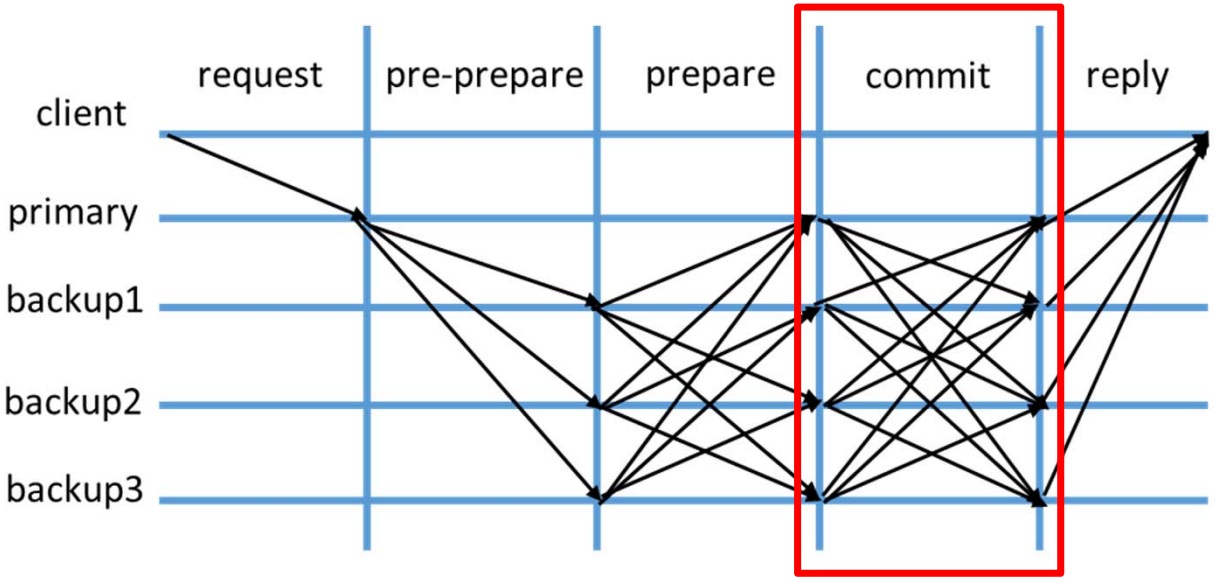
- Backup node generates the following message and sends it to all nodes
  - **((PREPARE,v,n,d, i)s_i)**
    - ✓ **PREPARE: Consensus Phase**
    - ✓ **i: Message sending node i**
    - ✓ **s_i: signature of node i**
  - **The node received 2f+1(including itself) pre-prepared messages became prepared(m,v,n,i) state**



PBFT(Practical Byzantine Fault Tolerance)

Commit

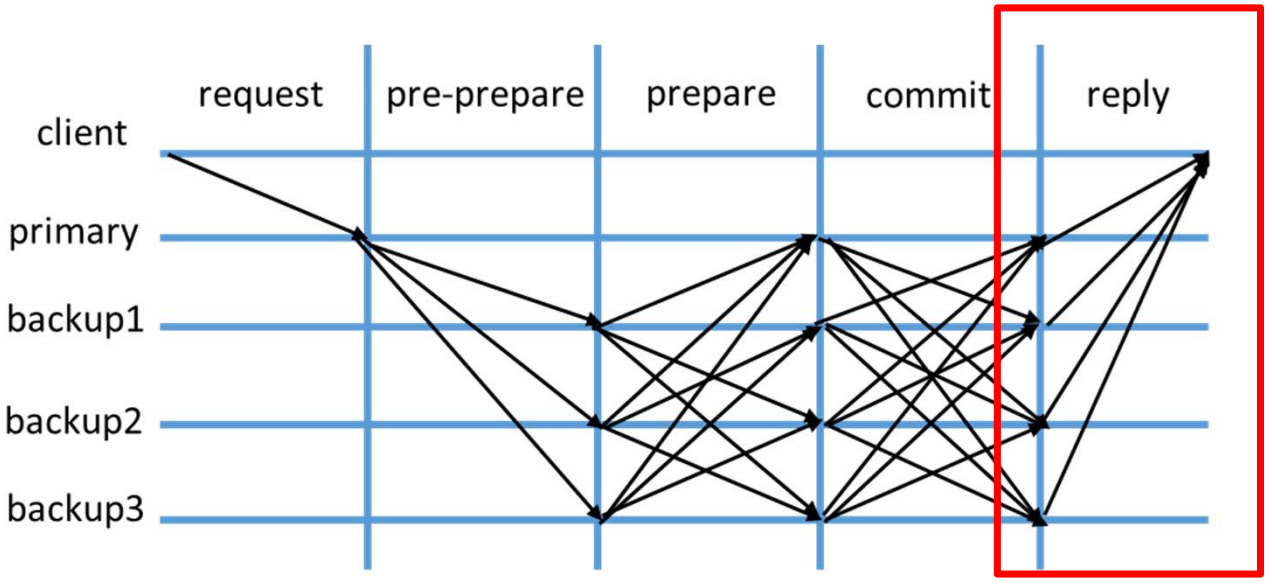
- When the backup node confirms that the node in the prepared (m, v, n, i) phase is f + 1 (non-faulty nodes), the state became Committed (m, v, n) State
  - **((COMMIT,v,n,d, i)s_i)**



- ✓ **v: view number(current Primary node can be known)**
- ✓ **n: sequence number**
- ✓ **m: message the cliend sent**
- ✓ **d: message digest**
- ✓ **i: Message sending node I**
- ✓ **s_i: Signature of node i**

PBFT: Reply

- When the client receives f + 1 or more identical responses from backup and primary, it verifies that the request is complete



PBFT : Conclusion

- Provides an algorithm for voting between nodes
  - High latency through each phase
- Significant Traffic Generation
- Conclusion: an algorithm suitable for a network composed of small nodes

Blockchain Mechanism and Platform (2 of 2)

KHU

경희대학교 KYUNG HEE UNIVERSITY

Blockchain Platform

2. Blockchain Platform : Characteristics in each Generation

- 1st Generation (2009~2014) - Bitcoin
  - o Virtual Currency
  - o Asset Transaction

- 2nd Generation (2015~Current) – Ethereum, Hyperledger
  - o Smart Contract (Business Automation)
  - o Decentralized Application

- 3rd Generation (on going) – Various Platforms
  - o Scalability
  - o Interoperability
  - o IoT support

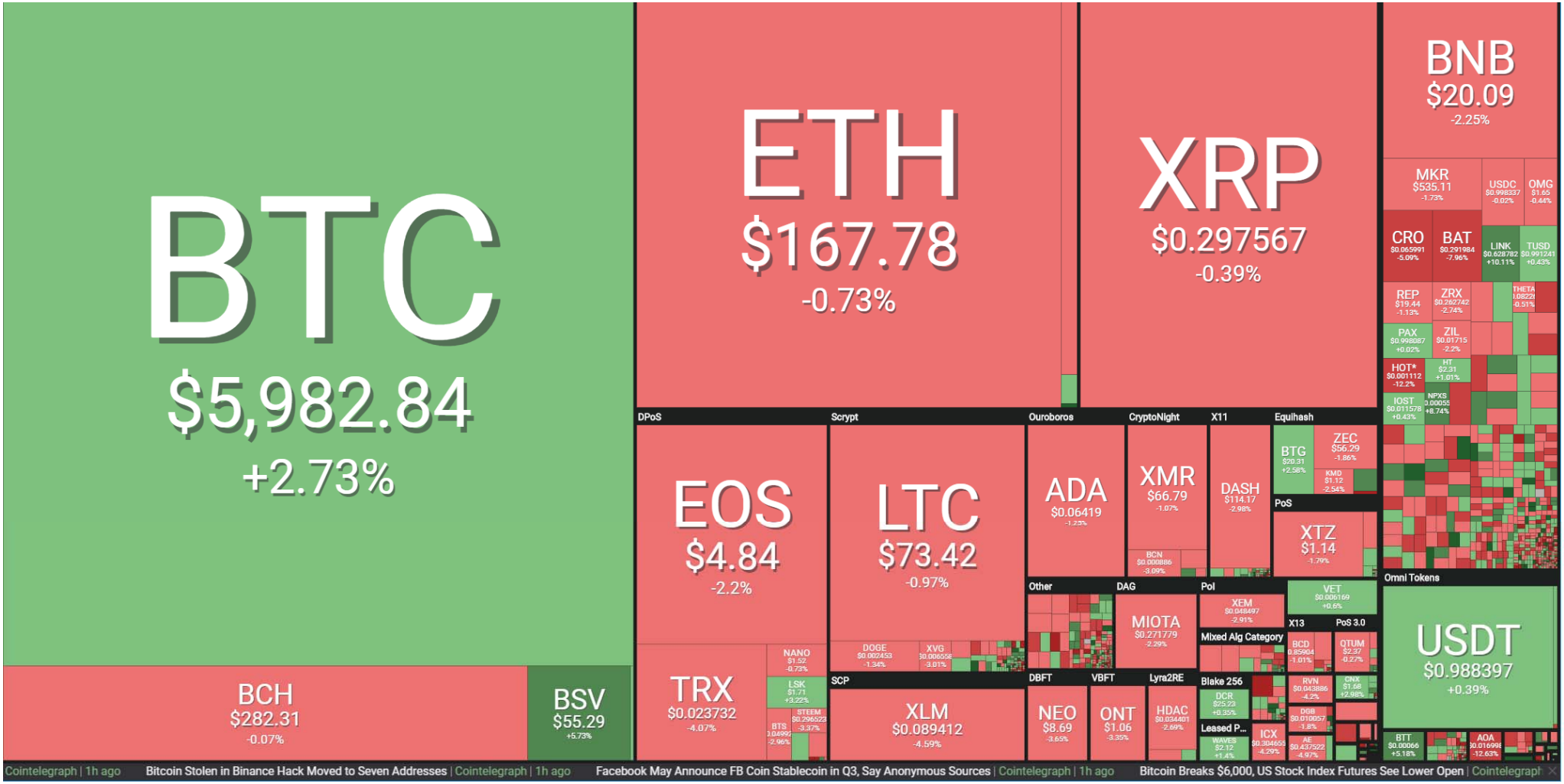
2. Blockchain : CoinMarketCap (Top 10 coins)

암호화폐 ▾ 거래소 ▾ 관심 목록 KRW ▾ 다음 100 → 모두 보기

#	이름	시가총액	가격	거래량(24시간)	유통 공급량	변경(24시간)	가격 그래프(7일)
1	Bitcoin	₩126,985,748,766,723	₩7,178,026	₩19,181,139,410,835	17,690,900 BTC	2.54%	
2	Ethereum	₩21,259,251,114,288	₩200,567	₩7,873,319,653,976	105,995,979 ETH	-0.64%	
3	XRP	₩14,899,416,143,472	₩353.63	₩1,057,470,053,747	42,133,310,721 XRP *	-0.70%	
4	Bitcoin Cash	₩5,982,310,127,995	₩336,604	₩1,651,747,257,267	17,772,550 BCH	-0.39%	
5	Litecoin	₩5,379,863,056,298	₩87,200.73	₩3,143,165,479,970	61,695,157 LTC	-0.74%	
6	EOS	₩5,213,384,427,934	₩5,723.41	₩1,951,131,590,579	910,887,960 EOS *	-1.22%	
7	Binance Coin	₩3,384,144,883,156	₩23,971.19	₩206,884,369,240	141,175,490 BNB *	-1.76%	
8	Tether	₩3,294,477,822,005	₩1,186.52	₩15,916,018,375,270	2,776,595,295 USDT *	0.56%	
9	Stellar	₩2,058,815,510,601	₩107.44	₩247,161,446,283	19,162,820,780 XLM *	-3.78%	
10	Cardano	₩1,914,630,736,218	₩73.85	₩53,144,179,785	25,927,070,538 ADA	-3.53%	

https://coinmarketcap.com/@May 9, 2019

2. Blockchain: Coin360 Classification



2. Blockchain Platform: CoinMarketCap (all)

2151	TOKOK	TOK	$?	$0.004427	? *	$0	-0.47%	-0.35%	-6.59%	•••
2152	Gamblica	GMBC	$?	$0.000972	? *	$0	0.00%	2.83%	11.04%	•••
2153	COZ	COZ	$?	$0.124183	? *	$0	0.00%	0.00%	0.00%	•••
2154	UTEMIS	UTS	$?	$0.000272	? *	$0	0.00%	-1.09%	4.01%	•••
2155	RabbitCoin	RBBT	$?	$0.000003	?	$?	0.00%	0.00%	20.76%	•••
2156	Bubble	BUB	$?	$0.003023	? *	$?	0.00%	0.00%	6.35%	•••
2157	Axiom	AXIOM	$?	$0.004442	?	$?	0.00%	0.00%	0.00%	•••
2158	ClubCoin	CLUB	$?	$0.207019	? *	$?	0.00%	0.00%	1.76%	•••
2159	AvatarCoin	AV	$?	$0.141351	? *	$?	0.00%	0.00%	75.85%	•••
2160	Francs	FRN	$?	$0.003325	?	$?	0.00%	0.00%	0.00%	•••
2161	Aces	ACES	$?	$0.000052	? *	$?	0.00%	0.00%	0.00%	•••
2162	Wink	WINK	$?	$0.000106	? *	$?	0.00%	0.00%	0.00%	•••
2163	Ethereum Lite	ELITE	$?	$0.079566	? *	$?	0.00%	0.00%	0.00%	•••
2164	BTCMoon	BTCM	$?	$0.001719	? *	$?	0.00%	0.00%	25.27%	•••

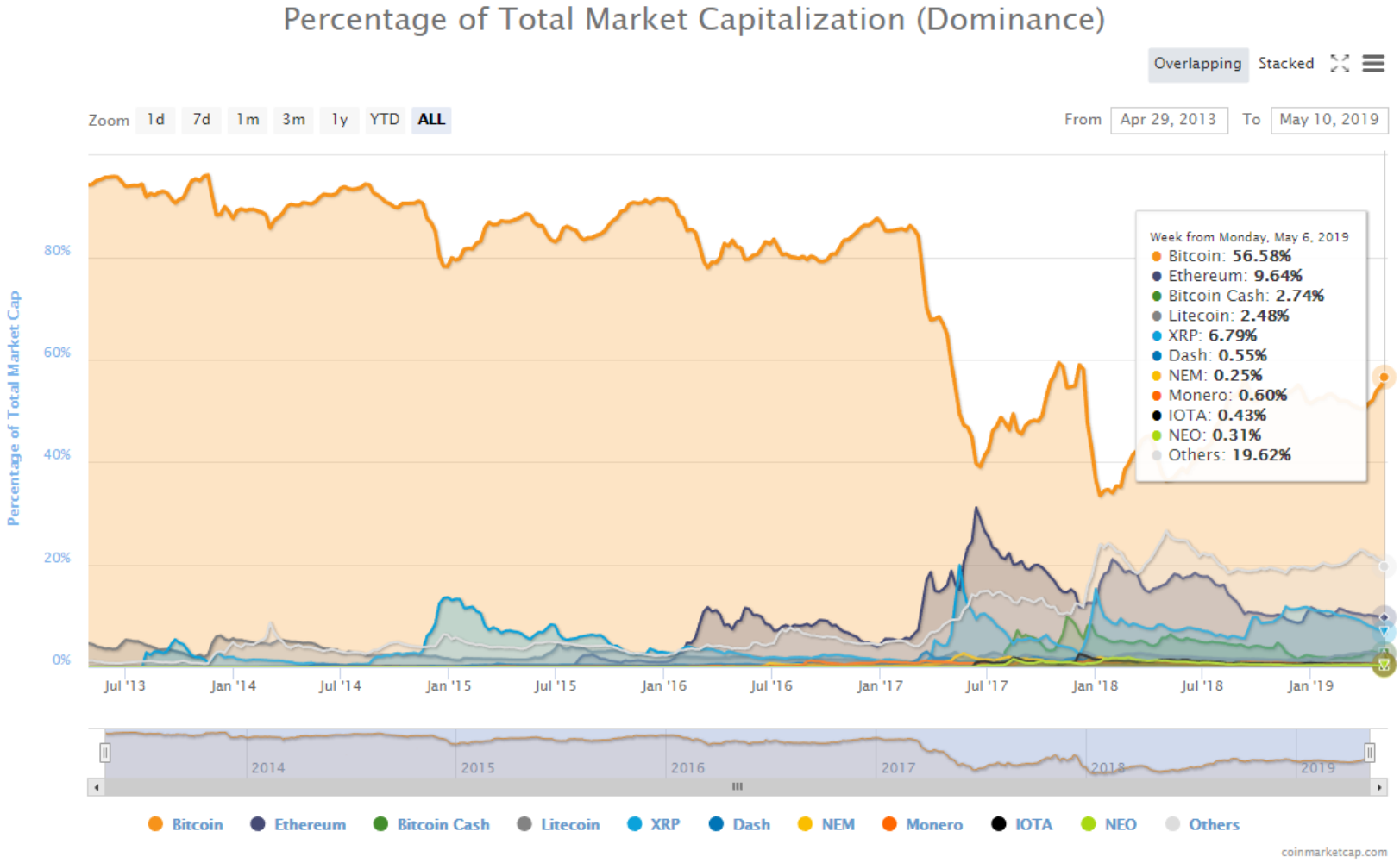
** Not Mineable*

← Back to Top 100

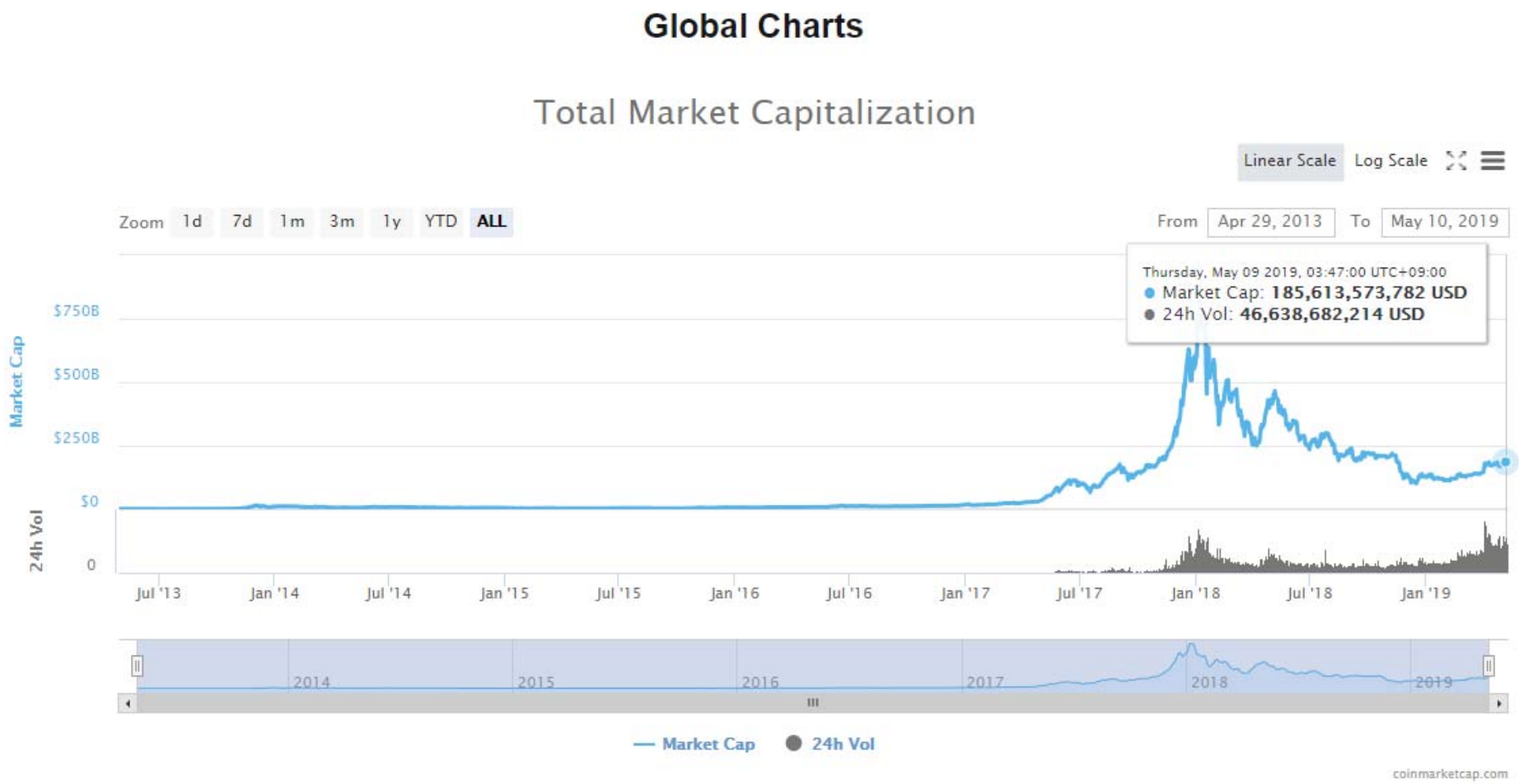
Total Market Cap: $187,142,041,881

Last updated: May 09, 2019 12:50 PM UTC

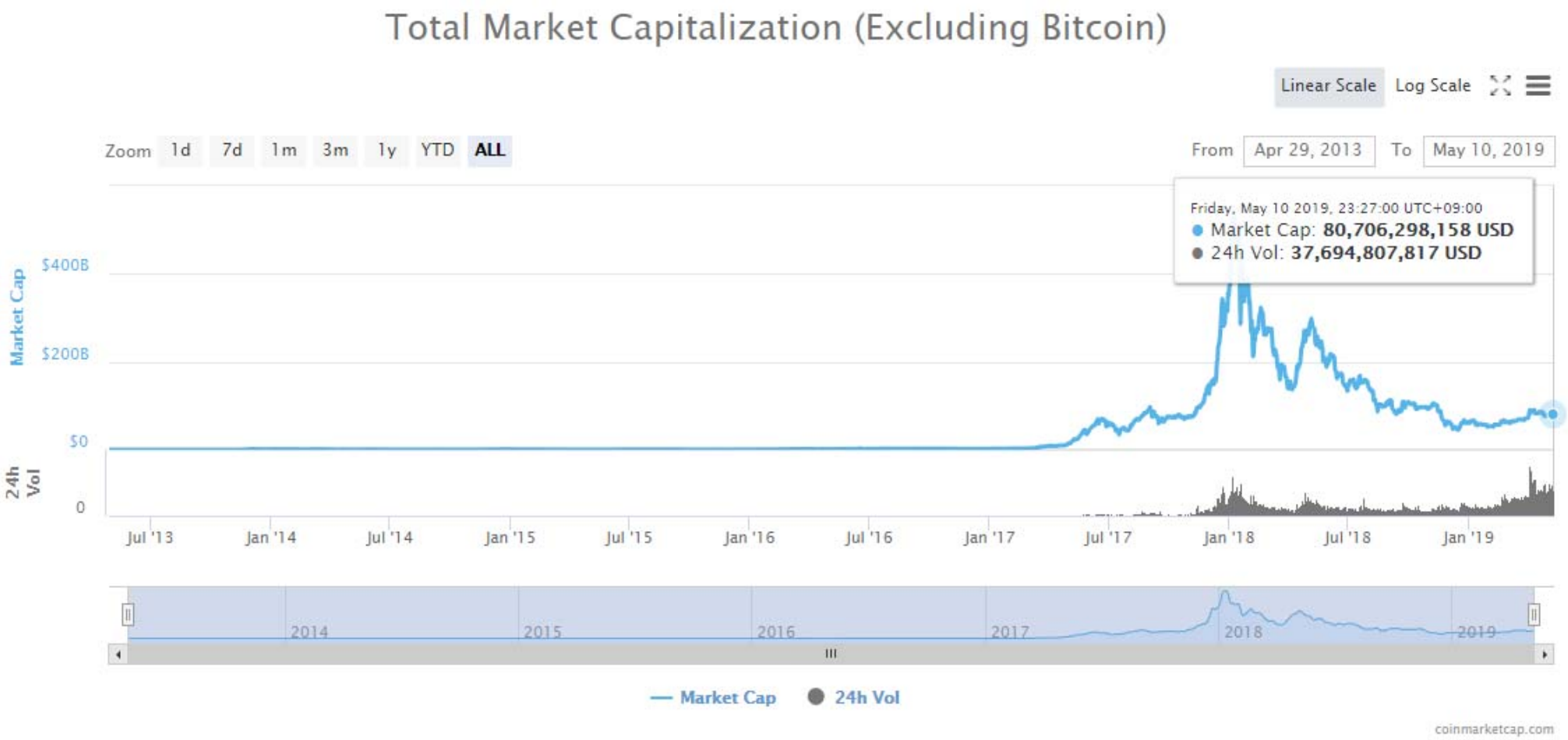
2. Blockchain Platform : Coin Market



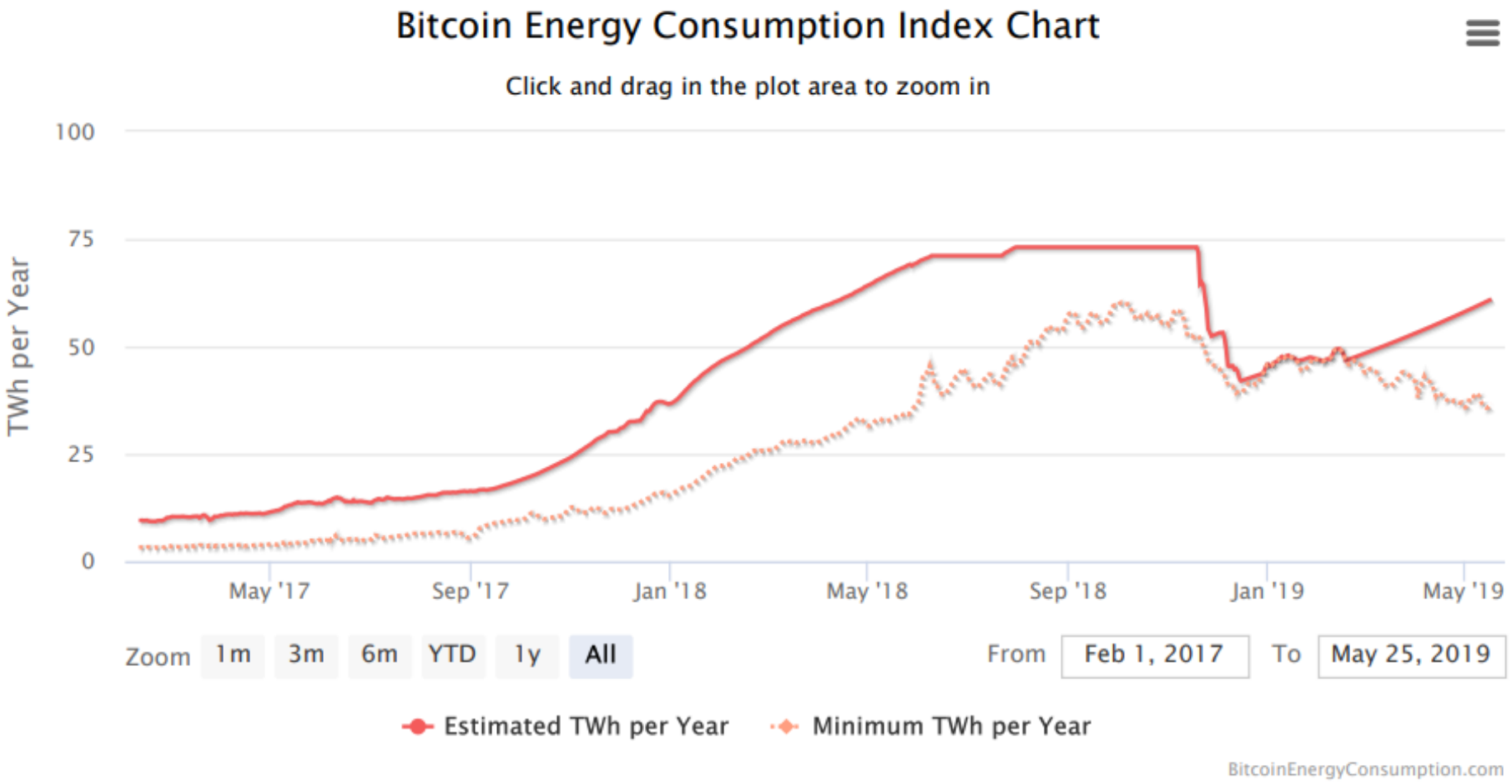
2. Blockchain Platform : Value of Total Coint Market



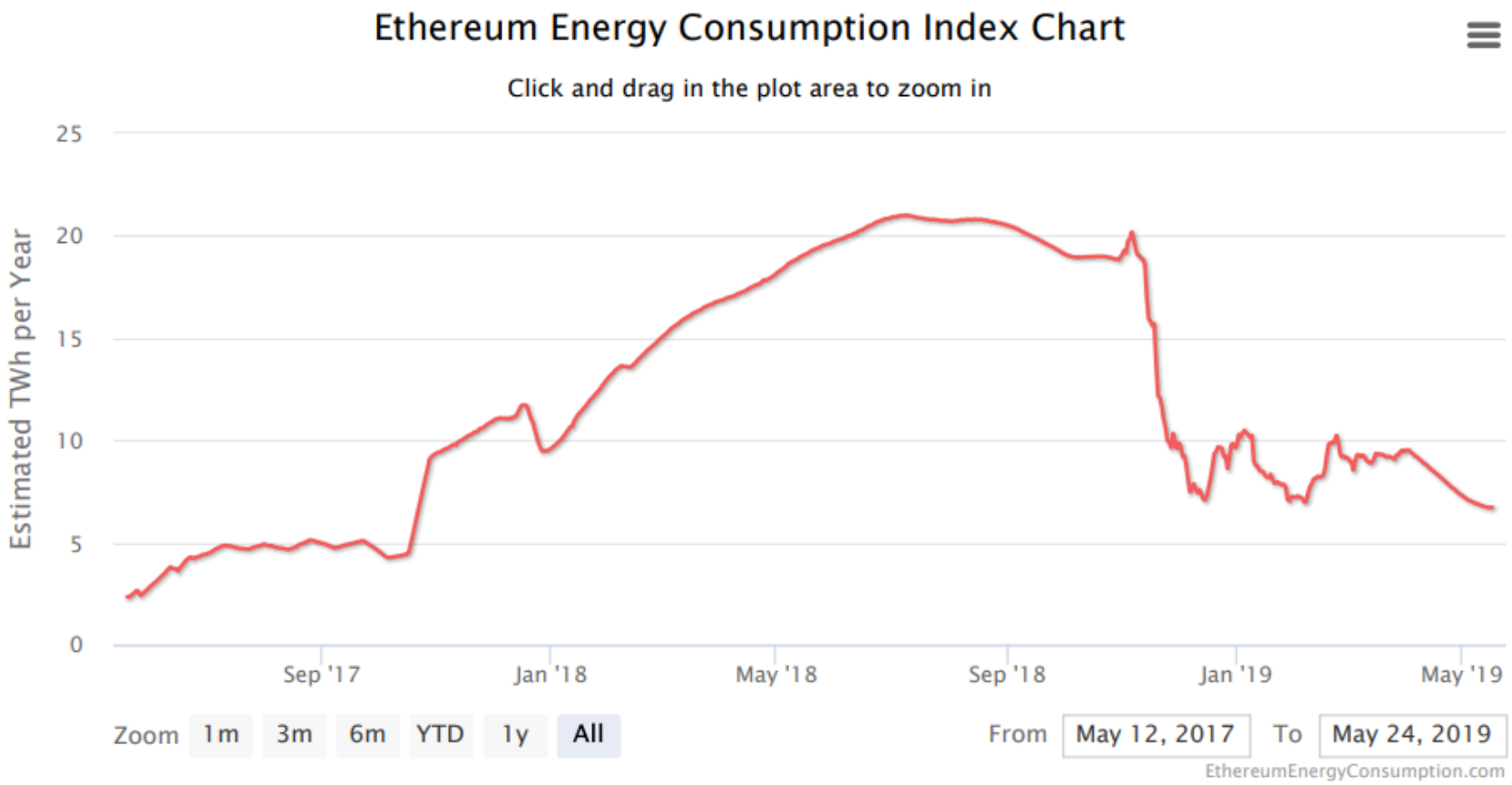
2. Blockchain Platform : Value of Total Coint Market(except bitcoin)



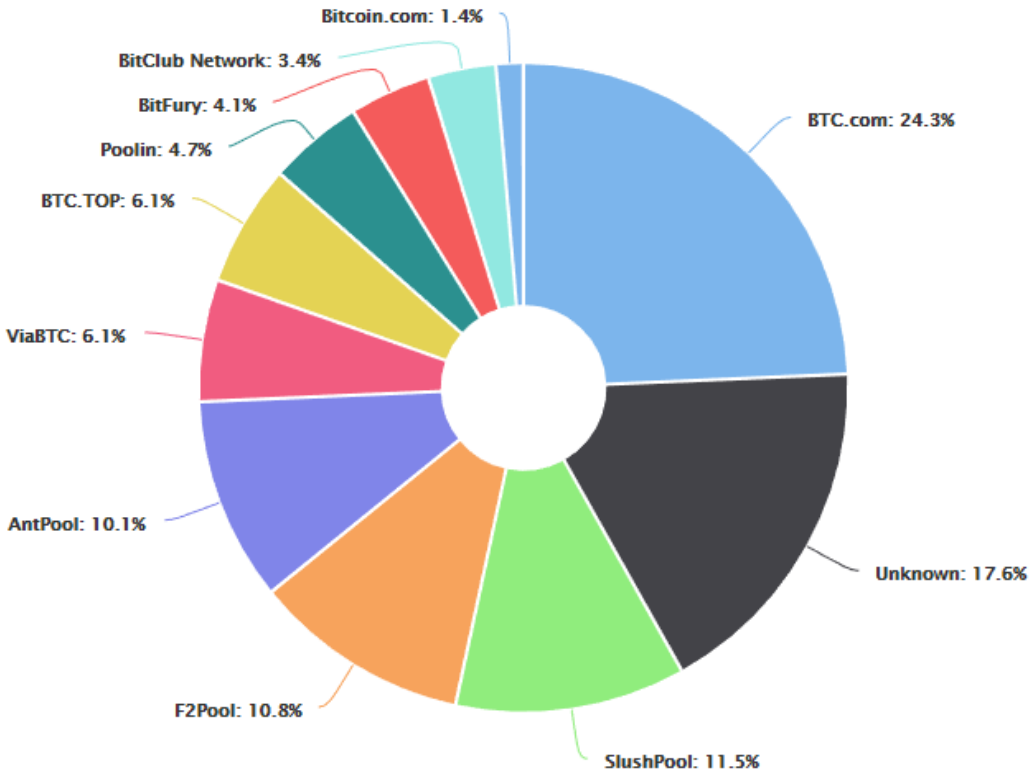
2. Blockchain Platform: Energy Comsumption of PoW



2. Blockchain Platform: Energy Comsumption of PoW



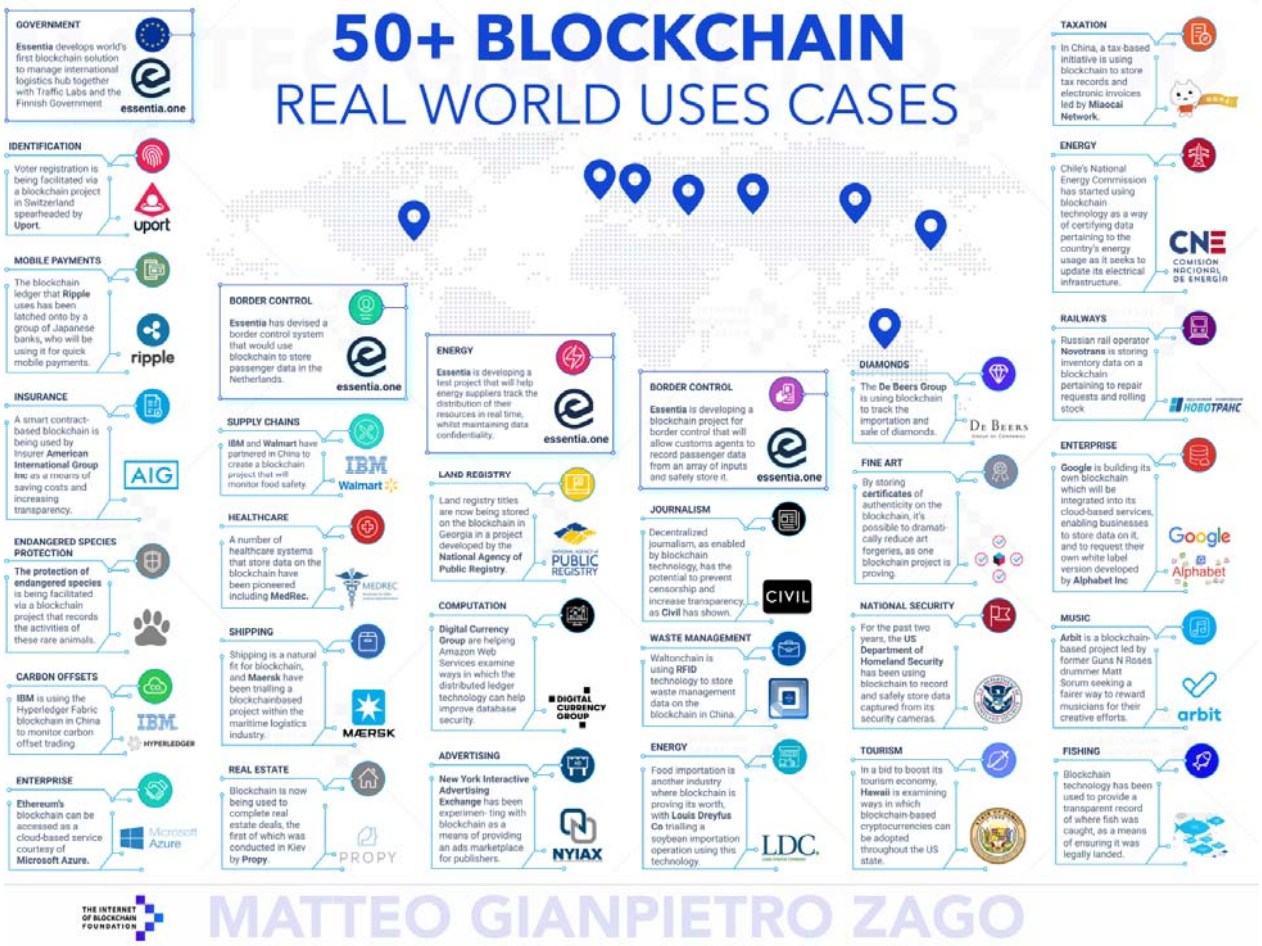
2. Blockchain Platform : BTC Hashrate Distribution



Known Blocks.

Relayed By	count
BTC.com	36
Unknown	26
SlushPool	17
F2Pool	16
AntPool	15
ViaBTC	9
BTC.TOP	9
Poolin	7
BitFury	6
BitClub Network	5
Bitcoin.com	2

2. Blockchain Platform: : 50+ Blockchain



Government
Waste Management
Indentification
Healthcare
Enterpise
Medical
Music
Carbon Offsets
Supply Chains
Diamonds
Real Estate
Fishing Industry
Fine Art
Public Utilities
Tourism
National Securit
Taxation
LGBT Rights
Mobile Payments
Land Registry
Gaming
Energy Distribution
Railways
Oil Industry
Smart Cities
Journalism
Advertising
Endangered
Species Protection
Insurance
Computation

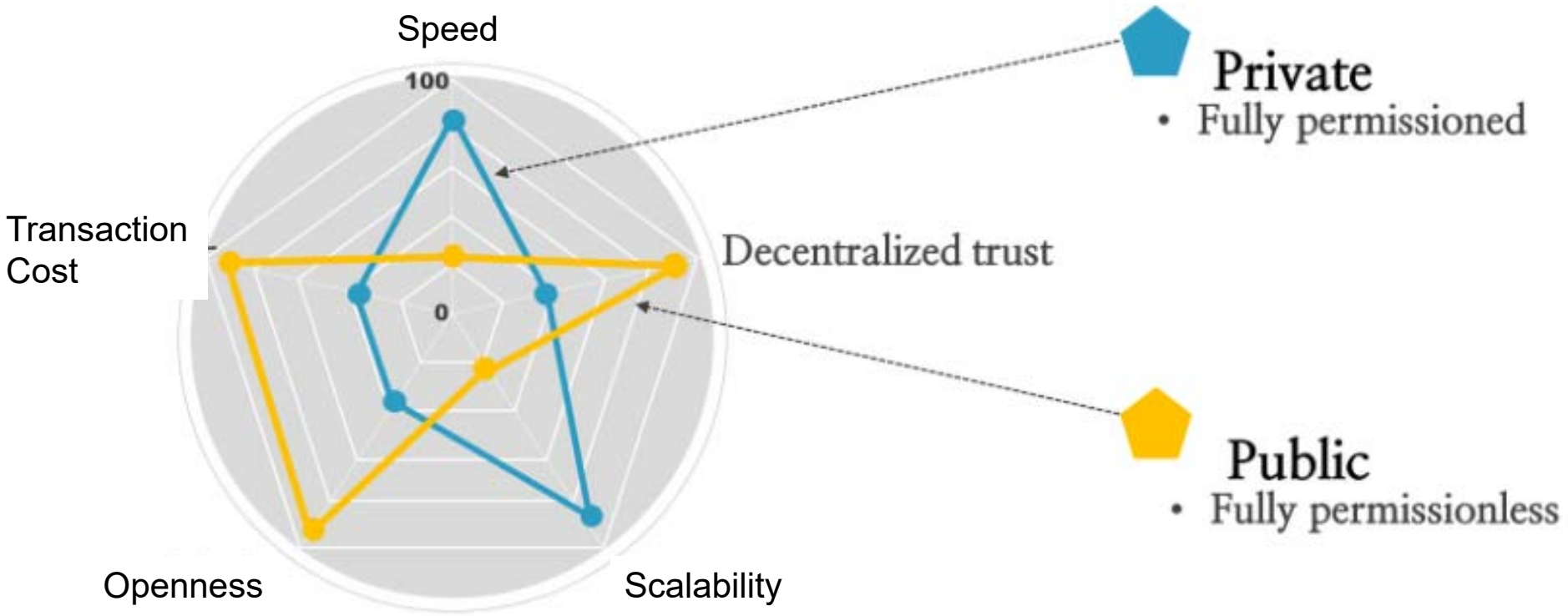
https://medium.com/@matteozago/50-examples-of-how-blockchains-are-taking-over-the-world-4276bf488a4b

Public and Private Blockchain

Public vs Private Blockchain

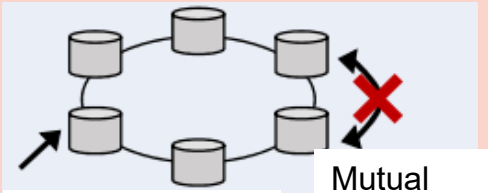
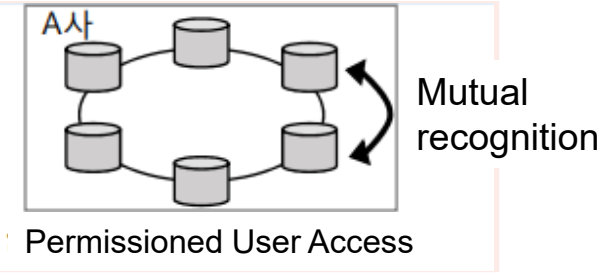
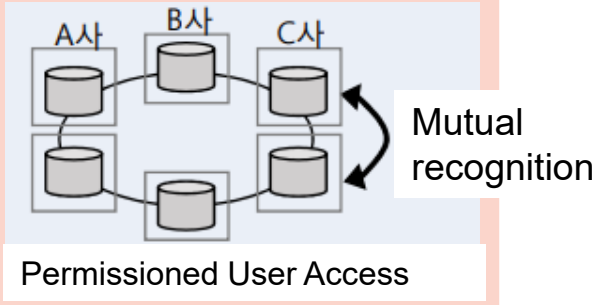
	Public blockchain	Private blockchain
Accesibility	Any	Permissioned
Speed	Slow(7~20 TPS)	Fast(1000 TPS +)
Identity	Anonymous node	Known node
Fee	Necessity	Not need, a little
Hard Fork	possible	Impossible
Upgrade	difficult	easy
Decentralization	high	low
Consensus Algorithm	PoW, PoS, DPoS 등	BFT 계열
Cryptocurrency	Bitcoin,Ethereum, etc	Ripple, Fabric, ICON

Public vs Private Blockchain



Source : http://s-springnote.tistory.com/4

Public vs Private Blockchain

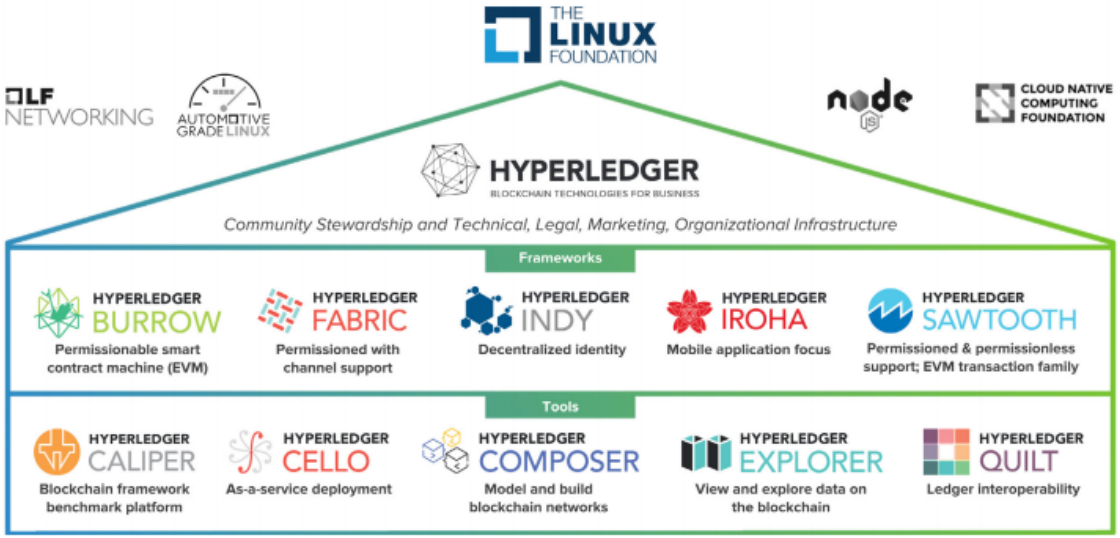
Type Classification	Characteristics	Architecture
Public Blockchain	• First Block Chain Use Case • Disclosed and operated to all over the Internet • Anyone can participate in notarization through computing power • Difficult network expansion and slow transaction	
Private Blockchain	• Private type blockchain • One entity manages the internal network as a blockchain • Platform service for the development of the chain	
Consortium Blockchain	• Semi-central type block chain • Only a few pre-selected subjects (N) can participate • Notarized participation through agreed rules between the parties • Easy network expansion and fast transaction speed	

비트코인, 이더리움, 하이퍼레저 비교

	bitcoin	Ethereum	Hyperledger
Type	Public Blockchain	Public Blockchain	Private Blockchain
Participation as a node	Anyone	Anyone	Can only participate in authorized nodes through membership service, issue certificate based on PKI
Consensus Algorithm	PoW	PoW -> PoS	PBFT/RAFT
Payment Completeness	None	None	Exist
Performance	Blockchain block generation every 10minutes	Generate blocks approximately every 12 seconds	Ensures superior performance because the agreement is finalized upon renewal
Transaction stagnation	Transaction information is public	Release of Transaction Information	Release of Transaction Information/Encryption is selectable
Smart Contract	Almost no. Limited use	EVM, Ethereum Virtual Machine/ Develop member – Development of Solidity Lang.	Smart contracts can be implemented through chain codes. Developed with Go and Java체인

Hyperledger

- A project that was initiated by Linux foundation in December 2015 to advance blockchain technology
- A collaborative effort by its members to build an open source distributed ledger framework that can be used to develop and implement cross-industry blockchain applications and systems
- The key focus is to build and run platforms that support global business transactions.
- Projects under the Hyperledger umbrella:



Hyperledger Projects

Infrastructure

Technical, Legal, Marketing, Organizational

Ecosystems that accelerate open development and commercial adoption

THE LINUX FOUNDATION

Cloud Foundry | Node.js | **Hyperledger** | Open Container Initiative

Frameworks

Meaningfully differentiated approaches to business blockchain frameworks developed by a growing community of communities

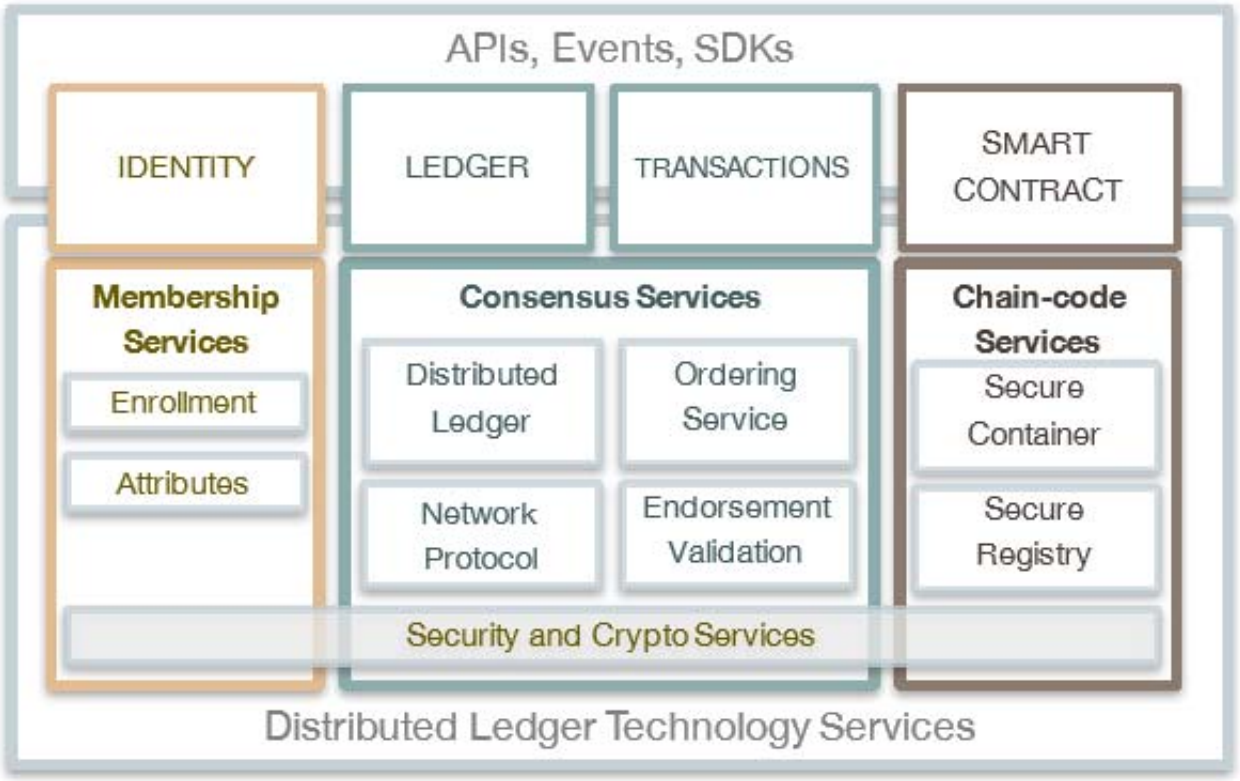
Hyperledger **Indy** | Hyperledger **Fabric** | Hyperledger **Iroha** | Hyperledger **Sawtooth** | Hyperledger **Burrow**

Tools

Typically built for one framework, and through common license and community of communities approach, ported to other frameworks

Hyperledger **Composer** | Hyperledger **Explorer** | Hyperledger **Cello**

Hyperledger Reference Architecture



IDENTITY
Pluggable, Membership, Privacy and Auditability of transactions.

LEDGER | TRANSACTIONS
Distributed transactional ledger whose state is updated by consensus of stakeholders

SMART-CONTRACT
"Programmable Ledger", provide ability to run business logic against the blockchain (aka smart contract)

APIs, Events, SDKs
Multi-language native SDKs allow developers to write DLT apps

- A chaincode typically handles business logic agreed to by members of the network, so it may be considered as a "smart contract". State created by a chaincode is scoped exclusively to that chaincode and can't be accessed directly by another chaincode..

Some Applications for the Future Networks of Blockchain

- Blockcahin-Based Energy Trading -

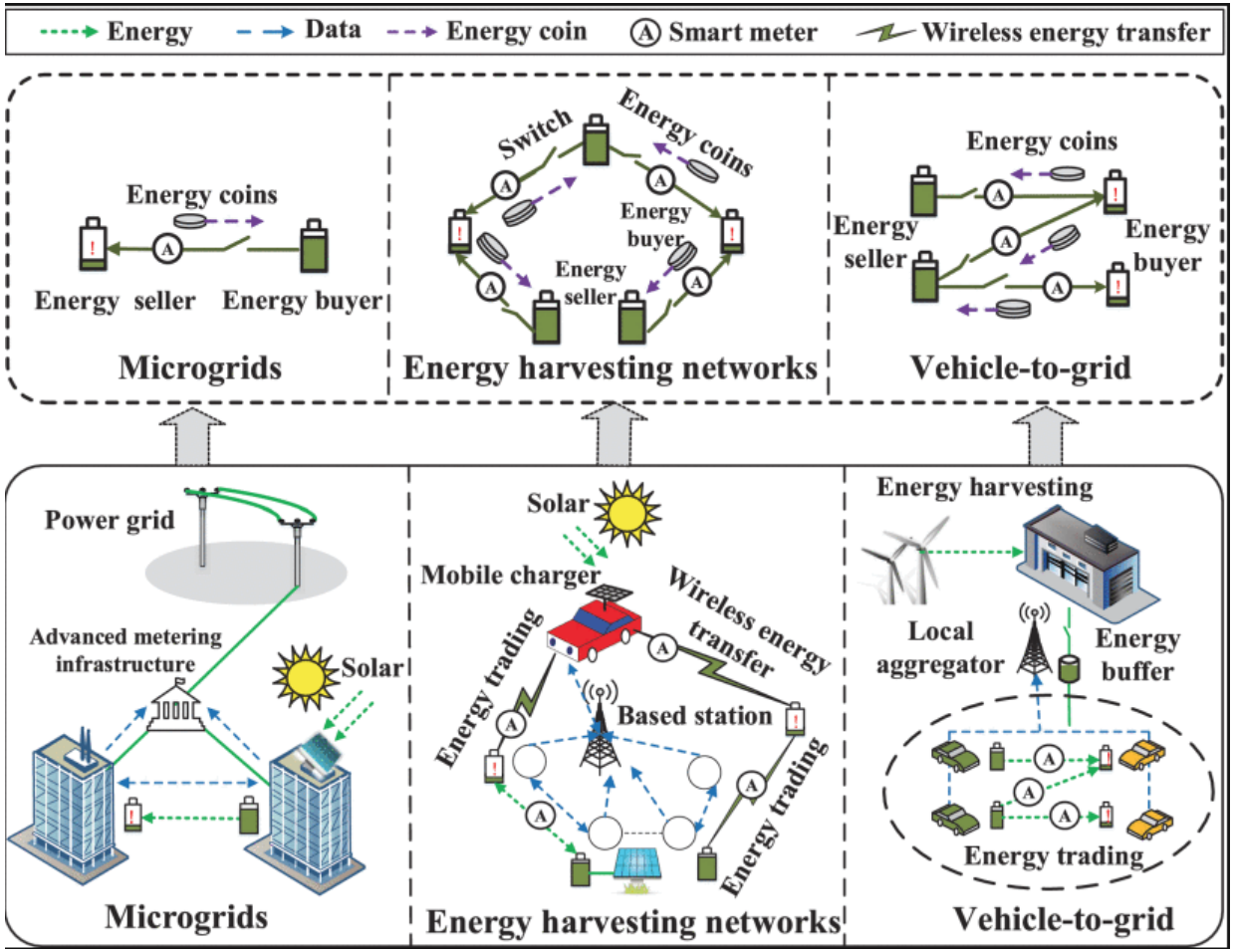
경희대학교
KYUNG HEE UNIVERSITY

Blockcahin-Based Energy Trading

- Many emerging technologies have been introduced into green industrial systems, e.g.,
  - Energy harvesting,
  - Wireless power transfer, and
  - Vehicle-to-grid
- Combined with these technologies, industrial systems develop various efficient and sustainable P2P energy trading scenarios
- There are three typical P2P energy trading scenarios for IIoT as following.
  - Microgrids
  - Energy harvesting networks
  - Vehicle-to-grid networks

Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3690-3700, Aug. 2018. doi: 10.1109/TII.2017.2786307

Blockchain-Enabled Energy Trading for IIoT



Microgrids: Smart buildings with solar panels or wind generators can form microgrids, in which the buildings harvest ambient energy and trade energy with each other by a P2P manner among the microgrids.

Energy harvesting networks: Industrial nodes with energy harvesting ability can obtain energy from renewable energy, also charge themselves through a mobile charger using wireless power transfer by P2P energy trading.

Vehicle-to-grid networks: Electric vehicles acted as energy storage devices perform charging operations at load valley, and feed their energy back into the power grid to reduce load peaks. Vehicles can also sell their energy to neighboring charging vehicles in a P2P manner with the help of local aggregators .

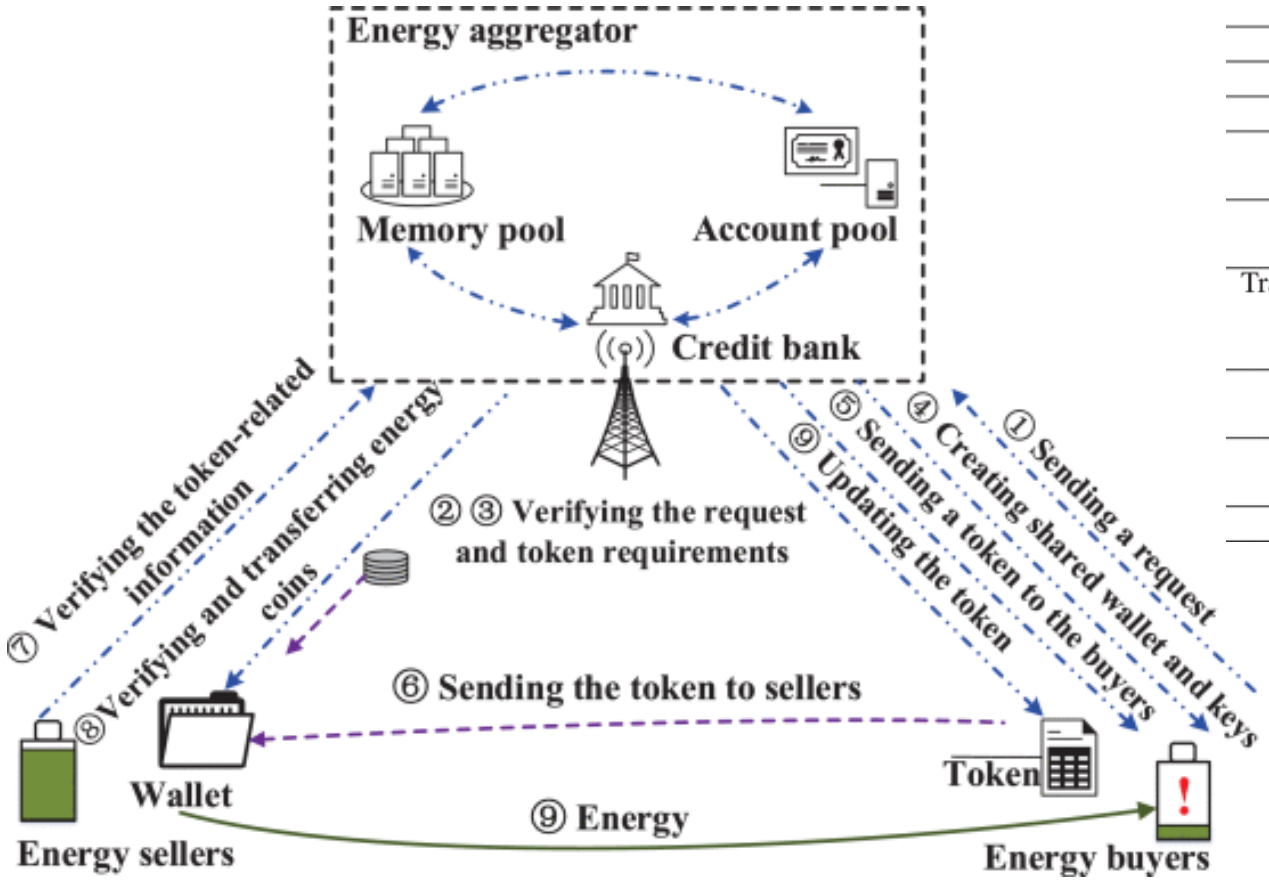
Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3690-3700, Aug. 2018. doi: 10.1109/TII.2017.2786307

Challenges of P2P energy trading

- Although P2P energy trading plays a vital role in IIoT, there are common security and privacy challenges for general P2P energy trading scenarios.
  - It is insecure for IIoT nodes to carry out large-scale decentralized energy trading in untrusted and nontransparent energy markets.
  - IIoT nodes with surplus energy may be not willing to participate as energy suppliers due to their concerns about privacy. In this case, energy supply and demand are unbalanced among IIoT nodes.
  - In P2P energy trading, there is an intermediary to audit and verify transaction record among IIoT nodes.

Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3690-3700, Aug. 2018. doi: 10.1109/TII.2017.2786307

Overview process



Terms	Description
Energy nodes	The IIoT nodes in the energy blockchain.
Energy sellers	The energy nodes with surplus energy to sell.
Energy buyers	The energy nodes with energy demand.
Borrowers	Energy buyers that borrow energy coins from credit banks.
Credit banks	The entities that provide energy coins to borrowers based on their credit values.
Transaction servers	The entities that collect and count energy requests, and thus match transaction pairs of energy trading.
Account pool	The entity that record wallets, wallet address and energy coin accounts in an EAG.
Memory pool	The entity that stores all transaction records of local energy nodes.
Wallet	The entity that store energy coins.

Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3690-3700, Aug. 2018. doi: 10.1109/TII.2017.2786307

Q & A