WHITEPAPER

PERSISTENT

Understanding Blockchain Consensus Models

Dr. Arati Baliga
Corporate CTO Office

April 2017

Outline

Executive Summary

Blockchain technology enables participants to read from and update to a common shared ledger (or blockchain) whose state is collectively maintained by the network in a decentralized fashion. The blockchain is updated via the consensus protocol that ensures a common, unambiguous ordering of transactions and blocks and guarantees the integrity and consistency of the blockchain across geographically distributed nodes. The Bitcoin cryptocurrency, which first popularized the concept of blockchain, also introduced Proof-of-Work (PoW) based consensus, which scaled to thousands of completely untrusted participants (nodes). PoW based consensus requires nodes to solve a hard cryptographic puzzle by brute-forcing and produce a winning value before being able to add a block to the blockchain. It however has a few drawbacks such as high latencies, low transaction rate and significant energy expenditure, which makes it a less than perfect fit for many applications. As blockchain independently emerged as a powerful technology, decoupled from the cryptocurrency, its consensus mechanisms also evolved independently dictated by the blockchain platform and application requirements. Since the consensus model maintains the sanctity of data recorded on the blockchain, it is important to ensure that it functions correctly in normal as well as adversarial conditions. This whitepaper provides an overview of consensus models as adopted by popular blockchain platforms and analyzes their merits and demerits.

Introduction

A blockchain-based system is a classical distributed system with shared state (i.e. the blockchain) where all participants are geographically distributed and connected via different kinds of networks. Blockchain platforms can be classified into two main types – permissionless and permissioned. Open-ended systems such as Bitcoin and Ethereum are permissionless. They are publicly available for use. Any node can conduct transactions as well as take part in the consensus process to advance the blockchain. Permissioned platforms such as Hyperledger Fabric and Multichain are aimed at consortiums where participation is close-ended. While clients are allowed to submit transactions, advancing the blockchain is restricted to a fixed set of peering nodes that are run by consortium members.

In a permissionless setup, the number of nodes is expected to be large, and these nodes are anonymous and untrusted since any node is allowed to join the network. Consensus mechanisms for such a setup have to account for maliciousness; particularly Sybil attacks. Sybil attacks on a blockchain network can allow a single user to generate several online identities to influence and manipulate the consensus process. Bitcoin solves this problem by designing the consensus round to be computationally hard. Nodes have to prove that they have expended significant amount of energy as Proof-of-Work (PoW) towards solving a hard cryptographic puzzle. This approach, though wasteful in terms of energy expense, is required to ensure the safety of the consensus process. Early blockchain platforms that were designed to be permissionless directly adopted PoW consensus mechanisms from Bitcoin or designed some variant of it, for example, NameCoin, LiteCoin, DogeCoin and Monero.

Permissioned platforms have semi-trusted members where only known participating nodes that are part of a consortium, are verified and registered. These groups are expected to be small in number and therefore can employ alternative consensus mechanisms. Achieving consensus in a distributed system has known solutions in the research literature, e.g. Paxos, RAFT, and various Byzantine Fault Tolerance algorithms. Permissioned blockchain platforms have largely adopted these consensus algorithms.

As blockchain platforms are being challenged to meet rigorous real-world application requirements, such as low latencies, immediate transaction finality, high performance and good scalability, limitations of existing consensus models are being recognized. On one hand, while PoW models support open-ended participation, they are a poor match for applications that need immediate transaction finality and have high transaction rates. They also waste a lot of power. One study found that the electricity wasted in Bitcoin mining is comparable to the average electricity consumption of Ireland [1]. On the other hand, consensus models designed for closed systems do not scale well beyond about twenty peering nodes and cannot have any open-ended participation. To address these limitations, new consensus models have been designed and newer ones are still emerging.

“A blockchain based system is as secure and robust as its consensus model. ”

The security of the consensus model is perhaps the most crucial aspect that requires close attention when choosing a blockchain platform. The consensus mechanism maintains the sanctity of the data recorded on the blockchain. The blockchain system will safeguard the transaction and block order thereby safeguarding all the key properties of blockchain, such as immutability and auditability only when the underlying assumptions are correct and the consensus model can uphold the state of the blockchain under failure and adversarial conditions.

Poor choice of a consensus mechanism can render the blockchain platform useless thereby compromising the data recorded on the blockchain. Below are some of the issues that can result when the consensus mechanism fails.

Blockchain Fork – A blockchain fork can result in different nodes in the system converging on different blocks as being part of the blockchain. In Bitcoin, though temporary forks may exist due to network latencies, the protocol is designed such that all nodes will eventually converge on a single chain. A blockchain fork can wreak havoc on applications leading to completely inconsistent view of data recorded on the blockchain thereby forcing applications to behave in an unpredictable manner. The Stellar network, which originally forked code from Ripple, experienced a fork in the Stellar blockchain due to a misconfiguration [2].

Consensus Failure – Certain consensus algorithms may not guarantee the ability to reach consensus. For e.g. if the consensus algorithm requires a super-majority vote from a certain percentage of nodes, failing to reach this number because of node or network failures, non-compliant nodes or as a result of valid honest nodes not being able to make a decision due to inconsistent messages received from other nodes, may result in consensus failure.

Dominance – Consensus round outcomes can be manipulated by a single or group of entities if it is not designed to be resilient against Sybil attacks, where one or handful of nodes can generate millions of identities that they control. Having such dominance allows the dominating group to confirm the transactions and blocks as per their rules, even include transactions that can double-spend the cryptocurrency. Dominance can also be achieved by other means, such as controlling 51% of mining power in a PoW network [3].

Cheating – Validating nodes either individually or in collusion can independently maintain parallel forks in the blockchain of fraudulent transactions or altered reality that can been provided as proof to the auditor or external third party. The consensus and blockchain reading mechanism has to ensure that such attacks cannot be carried out on the blockchain platform.

Poor Performance – Based on the design of the consensus algorithm, it may require more time under certain conditions for consensus to converge. These conditions could be dynamic where other nodes have turned malicious or a network partition may delay messages that are exchanged between nodes, etc. This may manifest as inconsistently high latencies in applications.

Consensus Background

Consensus mechanisms allow secure updating of a distributed shared state and have been a topic of active research in the past three decades. Common technique used for achieving fault tolerance in a distributed system is to distribute the shared state across multiple replicas in the network. Updating the replicated shared state happens according to pre-defined state transition rules defined by the state machine that is executed on all the replicas. This technique is known as state machine replication. Replication of state ensures that the state is not lost if one or more nodes crash. The state machine rules ensure that all nodes executing them with identical inputs, will eventually produce the same outputs. This results in eventual agreement on the change of state via the consensus protocol. The replicas also communicate with each other to build consensus and agree upon the finality of the state after a state change is executed. With a blockchain based system, the shared state is the blockchain and the state transition rules are the rules of the blockchain protocol.

Achieving consensus in a distributed system is challenging. Consensus algorithms have to be resilient to failures of nodes, partitioning of the network, message delays, messages reaching out-of-order and corrupted messages. They also have to deal with selfish and deliberately malicious nodes. Several algorithms are proposed in the research literature to solve this, with each algorithm making the required set of assumptions in terms of synchrony, message broadcasts, failures, malicious nodes, performance and security of the messages exchanged. For a blockchain network, achieving consensus ensures that all nodes in the network agree upon a consistent global state of the blockchain.

A consensus protocol has three key properties based upon which its applicability and efficacy can be determined.

i. **Safety** – A consensus protocol is determined to be safe if all nodes produce the same output and the outputs produced by the nodes are valid according to the rules of the protocol. This is also referred to as consistency of the shared state.

ii. **Liveness** - A consensus protocol guarantees liveness if all non-faulty nodes participating in consensus eventually produce a value.

iii. **Fault Tolerance** – A consensus protocol provides fault tolerance if it can recover from failure of a node participating in consensus.

While all the above three properties are crucial, a famous result by Fischer, Lynch and Paterson [4] known as the FLP Impossibility Result, states that no deterministic consensus protocol can guarantee safety, liveness and fault tolerance in an asynchronous system. While fault tolerance is crucial for globally distributed networks to operate, distributed systems tend to choose between safety and liveness depending on their system requirements and assumptions.

Fault tolerance refers to two types of faults in distributed systems. Fail-stop faults deal with node failures that cause nodes to stop participating in the consensus protocol. These are benign faults caused by hardware or software crashes. When a fail-stop fault occurs, the node just stops responding. The second category of faults are Byzantine faults, which cause nodes to behave erratically. This category of faults was identified and characterized by Leslie Lamport as the Byzantine General's Problem [5], summarized in the side note. Byzantine faults or failures can occur because of software bugs or as a result of the node being compromised. A Byzantine node can lie, can provide ambivalent responses or completely mislead other nodes involved in the consensus protocol. The consensus protocol has to be able to operate correctly and reach consensus in the presence of

The Byzantine Generals Problem

A group of generals, each commandeering a part of the Byzantine army has surrounded an enemy city. To attack the city, all generals have to agree on a battle plan. Generals can communicate via messengers only. The messengers might be captured by the enemy and the message might never reach the other general. The difficulty in the agreement is that one or more generals might be traitors and are interested in sabotaging the battle plan. To this end, they might send false messages, distort messages or not send any messages at all. All loyal generals will act according to the plan. A small number of traitors should not cause the loyal generals to adopt a bad plan.

Byzantine nodes as long as the number of Byzantine nodes within a distributed system are limited.

Traditional consensus approaches in distributed systems have focused on building fault tolerance in the face of unreliable systems provisioning mainly for fail-stop faults. Paxos [6], Raft [7] and variants, view-stamped replication [8] can be used for ordering transactions in distributed databases or to order client generated requests and respective state change in distributed applications using replicated state machines. Paxos was built to achieve fault tolerance and consistency in the face of failing nodes, which might either permanently fail or fail for some time and then recover, or in the face of an unreliable network, where messages are not reliably delivered. In the face of such failures, these algorithms guarantee progress and consistency in the data structures that were replicated across physical machines. The number of nodes needed in such networks are “2f+1” to be able to tolerate “f” fail-stop failures. Tolerating Byzantine faults, increases the complexity of the consensus protocol by adding several extra layers of messaging into the system. Practical Byzantine Fault Tolerance (PBFT) [9], which was first proposed by Miguel Castro and Barbara Liskov, was the first practical approach that allowed for Byzantine fault tolerant applications with low-overhead. Tolerating Byzantine faults needs “3f+ 1” replicas to be able to tolerate “f” faults in the system. PBFT uses the concept of primary and secondary replicas, where the secondary replicas automatically check the sanity and liveness of decisions taken by the primary and can collectively switch to a new primary if the primary is found to be compromised.

Bitcoin’s Proof-of-Work Invention

The Bitcoin system facilitates transfer of the cryptocurrency (Bitcoins) from one individual to another in a completely decentralized fashion, and no central entity controls either the production of Bitcoins or is involved in their transfer. The Bitcoin blockchain is replicated on multiple nodes and the nodes order the transactions based on a Proof-of-Work (PoW) consensus mechanism.

To add blocks to the blockchain, each node has to show that it has performed some amount of work, also known as Proof-of-Work (PoW). In Bitcoin, the node has to find a hash value that is less than a certain number, also referred to as the difficulty level set by the network. The difficulty level is dynamically tuned by the Bitcoin protocol, which currently ensures that one block is produced every ten minutes. The process of solving the PoW puzzle to find a winning hash value is known as mining. The first node to find a winning hash gets to add its proposed block to the blockchain and also claim the mining reward. Due to the distributed, concurrent nature of this process, sometimes more than one node is able to find a winning hash at the same time. Each winning node adds its own proposed block to the blockchain and broadcasts this over the peer-to-peer network. In such cases, there is a temporary fork in the blockchain, where some nodes are adding blocks to one branch, while other nodes are adding blocks to other branches, based on which winning node is closest to them. However, as more blocks are added to these forks, the protocol will ensure that the branch with the maximum PoW (i.e. the longest branch) will get included in the blockchain and others will be discarded. This leads to an eventual consistency among all nodes regarding the state of the blockchain.

The Bitcoin PoW consensus algorithm works well in an open environment where any number of nodes can participate in the network and start mining. No knowledge or authentication is needed of any participants, thereby making this kind of consensus model extremely scalable in terms of supporting thousands of nodes. The Bitcoin PoW consensus is however vulnerable to “51%” attacks, where a mining pool that is able to control 51% of the mining power (i.e. hashrate), can write its own blocks into the blockchain or fork it to create an independent branch that converges at a later point with the main blockchain. The advantage for the attacker in launching such an attack, is that he can double spend his own funds and selectively reject transactions that he does not want included on the blockchain.

Another research [10] proves that a new type of attack can be carried out with an approach known as selfish mining, where the normally honest miners are incentivized to support the attacker and join in carrying out a 51% attack. In this attack, the attacker performs erratic mining, at the cost of his short term revenue by maintaining a separate private blockchain in parallel to the Bitcoin blockchain. He selectively publishes many blocks all at once, forcing rest of the network to discard their blocks and ultimately losing revenue. This incentivizes honest miners to join the attackers’ coalition to increase their revenue, which eventually can get to the size of 51% of the network's mining power, enabling the attacker to carry out a 51% attack.

Bitcoin uses the PoW model, which guarantees eventual consistency in the Bitcoin blockchain despite temporary forks. This approach results in longer transaction confirmation times, to ensure reasonable consensus finality, resulting in slower transaction confirmation rate, which at the time of writing is approximately 7 transactions/second. This is considered very slow in the world of payments, particularly when contrasted with MasterCard or VISA's 10,000 transactions/sec. Bitcoin-PoW also wastes a lot of energy in computation of hashes during the mining process. It however has excellent scalability in terms of nodes participating in the network and operates completely in a decentralized fashion.

Blockchain Consensus Models

Blockchain platforms use a range of consensus models which are PoW and PBFT in their original form or variations of it providing certain advantages desired over the original model. Bold new models are also proposed, such as Proof-of-Stake (PoS) and Proof-of-elapsed-Time (PoeT) and variations of PBFT appear as viable alternatives. In this section, we review the broad categories of consensus models used by the popular blockchain platforms.

Proof of Work – Ethereum (Homestead)

Ethereum is a general purpose blockchain platform that supports smart contracts, a Turing complete programming language, Solidity, for writing smart contracts and a virtual machine called Ethereum Virtual Machine (EVM) for executing smart contract code on Ethereum nodes. Like Bitcoin, Ethereum network is open and permissionless; any user can download the Ethereum client to create an account and join the Ethereum network. It uses an internal cryptocurrency called Ether, which is used to pay for the network resources as well serve as an anti-spamming/DDoS defense measure. Ethereum is designed to be general purpose and can be used by any kind of application requiring blockchain support. All transactions are recorded on the Ethereum blockchain and can be verified by any entity using the Ethereum network.

Ethereum (in the current version called Homestead) uses its own PoW consensus model, called EthHash that provides fast confirmation times and builds ASIC resistance to counter 51% attacks that Bitcoin is susceptible to. Ethhash was designed to counter mining centralization. Mining centralization was a weakness in Bitcoin where a large number of ASICs were cheaply produced to perform hashing operations at very high rates, thereby outnumbering and outperforming the general purpose computer hardware by a very large margin. This allowed select powerful entities, such as large corporations to create mining pools, with a very high hashing rate and that allowed them to control a large portion of the computing power of the Bitcoin network.

Ethhash uses two techniques for combating mining centralization. First technique uses a property called memory hardness. Memory hardness refers to the ability of the computer to move data around in memory (rather than perform calculations), a property that general purpose computer hardware is already designed to perform significantly well but cannot be achieved efficiently on ASICs. By making the algorithm ASIC resistant, it prevents large powerful companies from seizing control of the mining power in the Ethereum network. A second technique referred to as GHOST, includes the headers of the recently orphaned blocks known as uncles. Orphaned blocks are blocks that were included on the temporary forks created off the main blockchain. The node producing the uncle block and the node including it in the blockchain are given a reduced reward to encourage them to continue with the latest block in the Ethereum blockchain.

Ethhash also (similar to Bitcoin) uses the concept of finding a correct nonce input that can generate a hash value below a certain difficulty level. In all PoW algorithms, this is a time consuming process where the node simply needs to cycle through the nonce values and run the algorithm each time to generate a result. The algorithm works by choosing subsets to a fixed resource dependent on the nonce and the block header. The fixed resource is a directed acyclic graph (DAG), which is few gigabytes in size, which each Ethereum client has to generate. The DAG changes and is totally different for each epoch in the Ethereum system. An epoch is identified as a time period taken to generate 30000 blocks. The DAG only depends on the block height and the clients can pre-generate and cache the DAG. If the pre-generation is not done, the client can experience massive delays at every epoch transition as mining cannot begin without having a DAG for that epoch. DAG is needed for mining only and not for verification. Verification is a light weight process that can be computed in a fixed amount of time with low power CPU and small amount of memory. The Ethereum network adjusts the difficulty level to produce a block every 15 seconds.

Ethereum shares the same concerns as Bitcoin regarding the 51% attack. If an attacker can control 51% of the mining power, a fork can be generated in the Ethereum blockchain. However, the ASIC resistant design builds in significant resistance towards carrying out a 51% attack in the Ethereum network. Ethereum is scheduled to move to a Proof-of-Stake algorithm in its future release (Serenity).

Proof-of-Stake Model (PoS) – Ethereum (Serenity)

Proof-of-Stake algorithms are designed to overcome the disadvantages of PoW algorithms in terms of the high electricity consumption involved in mining operations. PoS completely replaces the mining operation with an alternative approach involving a user’s stake or ownership of virtual currency in the blockchain system. Putting it another way, instead of a user spending say $2000 buying mining equipment to engage in PoW algorithm and winning a mining reward, with PoS she can buy $2000 worth of cryptocurrency and use it as stake to buy proportionate block creation chances in the blockchain system by becoming a validator. The PoS algorithm pseudo-randomly selects validators for block creation, thereby ensuring that no validator can predict its turn in advance. Naïve PoS algorithms suffer from a problem called Nothing-at-Stake. These implementations do not provide incentives for nodes to vote on the correct block. Therefore nodes can vote on multiple blocks supporting multiple forks to maximize their chances of winning a reward as they do not “expend” anything in doing so as opposed to in PoW, where the node would be splitting up its resources to vote on multiple forks. This is the Nothing-at-Stake problem which needs to be tackled for a correct and efficient implementation of PoS.

Ethereum’s PoS algorithm, called Casper, is perhaps the most advanced PoS algorithm. Though multiple rounds of PoCs are released, it is still in testing and is expected to be released in the Serenity version of the Ethereum platform. Casper uses the concept of security deposits and bets to achieve consensus.

Nodes are allowed to be bonded with the Ethereum system by making significant security deposits set by the protocol. These nodes are bonded validators and show commitment and interest in advancing the Ethereum blockchain via staking their security deposits. The initial list of bonded validators is tracked by a special contract known as the Casper contract. From there on, the bonded validator list can evolve based on newer nodes joining in and older ones leaving the system. Each validator is pseudo-randomly selected to produce a block from the active validator set, with the probability of selection linearly weighted by each validator’s deposit. If a validator is offline, a different validator is selected and this process repeats until an online validator is found that creates a block. If a validator produces a block that gets included in the chain, they receive a block reward equal to the total ether in the active validator set. If the validator produces a block that does not get included in the chain, the protocol works such that the validator loses the security deposit equal to the block reward. This mechanism proposes to solve the Nothing-at-Stake problem where it stop nodes from producing blocks that won’t get included in the main chain. Proof-of-Stake was first designed and a naïve version of it used by PeerCoin. Different variations of PoS are also used by BitShares, NXT and Tendermint.

Proof of Elapsed Time – Intel SawtoothLake

IntelLedger or Intel SawtoothLake is a blockchain platform developed by Intel and subsequently open sourced for use by the community. The project is officially now under Linux Foundation’s HyperLedger project as a proposal for further development. IntelLedger uses a consensus algorithm, designed by Intel, called Proof of Elapsed Time (PoET) intended to run in a Trusted Execution Environment (TEE), such as Intel’s Software Guard Extensions (SGX).

PoET uses a random leader election model or a lottery based election model based on SGX, where the protocol randomly selects the next leader to finalize the block. The random leader election algorithm uses this model to deal with untrusted nodes and open–ended participation of nodes in the consensus algorithm. For the consensus to work correctly, it has to randomly distribute the leader election among all available participating nodes and it needs a secure way for other nodes to verify that a given leader was correctly selected without any scope for manipulation. This is achieved using the TEE to guarantee the safety and randomness of electing a leader.

Leader election works as follows. All validating or mining nodes have to run the TEE using Intel SGX. Each validator requests a wait time from the code running inside the TEE. The validator with the shortest wait time wins the lottery and can become the leader. The functions within the TEE are designed such that their execution cannot be tampered with by external software.

When a validating node claims to be a leader and mines a block, it can also produce proof generated within the TEE that other nodes can easily verify. It has to prove that it had the shortest wait time and it waited for a protocol designated amount of time before it is allowed to start mining the next block.

The randomness in generating wait times ensures that the leader role is randomly distributed among all validating nodes. The only drawback of this algorithm is the reliance on specialized hardware.

Byzantine Fault Tolerance and variants – HyperLedger Fabric

Hyperledger Fabric, which is the most popular permissioned blockchain platform being developed by the Linux Foundation provides a flexible architecture with a pluggable consensus model. Fabric is designed for consortiums where the group of participants in the consortiums is not only known, but their identities are registered and verified with a central registry service running within the system. It also supports smart contracts on the blockchain, also known as chaincode. Hyperledger currently supports two consensus models – the popular Practical Byzantine Fault Tolerance algorithm (PBFT) and its variation SIEVE that is able to handle non-deterministic chaincode execution. Current proposals are considering Crash Fault Tolerance (XFT) [11], which is a variation of Paxos with Byzantine Fault Tolerance built-in, as an alternative consensus algorithm for future versions.

PBFT

The Practical Byzantine Fault Tolerance algorithm proposed by Miguel Castro and Barbara Liskov was the first practical solution to the achieving consensus in the face of Byzantine failures. It uses the concept of replicated state machine and voting by replicas for state changes. It also provides several important optimizations, such as signing and encryption of messages exchanged between replicas and clients, reducing the size and number of messages exchanged, for the system to be practical in the face of Byzantine faults. This algorithm requires “3f+1” replicas to be able to tolerate “f” failing nodes. This approach imposes a low overhead on the performance of the replicated service. The authors report a 3% overhead for a replicated network file system (NFS) service that they conducted their experiments on. PBFT however has only been scaled and studied to 20 replicas. It’s messaging overhead increases significantly as the number of replicas increase.

SIEVE

SIEVE consensus protocol is designed to handle non-determinism in chaincode execution. When non-determinism is present within the chaincode, it can produce different output when executed by different replicas in a distributed network. SIEVE handles transactions that are usually deterministic, but which may occasionally yield different outputs. The SIEVE protocol treats the chaincode itself like a bloack box. It initially executes all operations speculatively and then compares the outputs across replicas. If the protocol detects a minor divergence among a small number of replicas, the diverging values are sieved out. If the divergence occurs across several processes, then the offending operation itself is sieved out.

Cross-Fault Tolerance (XFT)

The XFT protocol is a new protocol that simplifies the attack model and make Byzantine Fault tolerance feasible and efficient for practical scenarios. BFT protocols assume a powerful adversary where the adversary is able to control the compromised nodes as well as the message delivery of the entire network. Being able to tackle such a powerful adversary brings in lot of complexity in BFT protocols and therefore makes them less efficient. XFT relaxes the assumption of the powerful adversary and solves the state machine replication problem by simplifying it and providing an efficient solution that can tolerate Byzantine faults.

XFT assumes that the adversary cannot control majority of the nodes and generate network partitions at will at the same time. XFT is particularly interesting to blockchain based systems. In such systems, nodes might have financial incentives to act maliciously, yet lack the means and capabilities to compromise communications between nodes or create arbitrary network partitions. In such geo-replicated systems, often there are multiple communication paths between peers and therefore communication is harder to break.

XFT is designed to to provide correct service as long as majority of the replicas are correct and can communicate with each other synchronously. It uses the same number of resources as protocols that can tolerate fail-stop failures and can yet tolerate Byzantine faults.

Federated Byzantine Agreement – Ripple & Stellar

Ripple and Stellar are two blockchain based platforms and payment protocols that use variations of the Byzantine Fault Tolerance consensus models by making them open-ended with respect to node participation. These blockchain platforms target financial use cases and the payments domain in particular. They provide payment protocols, which can settle cross-border transactions in a matter of seconds as opposed to today's infrastructure that takes days for the same.

The participants in such systems are end users, financial institutions that act as gateways and market makers that can be either users or financial institutions. End users generate payment transactions using client software and must trust some gateways to hold their payments. Gateways are like banks that people use in the real world to hold their money. Gateways hold user funds issued to the gateway in fiat currencies and create equivalent issuances in the Ripple/Stellar network, which reflect as an account balance in the global blockchain. A payment transaction can be verified by all nodes by referring to account balances in the global blockchain. Transaction settles by adjusting balances in the blockchain. Market-makers provide the required liquidity in this network. Market-makers maintain accounts with multiple gateways and in multiple currencies. They can trade in multiple currencies and their liquidity is used to settle trades and payment transactions. Since most transactions deal with payments in fiat and other cryptocurrencies, it is of utmost importance that the protocol orders transactions consistently to prevent double spend attacks. Ripple and Stellar use their own consensus models that are a derived form of Byzantine Fault tolerance modified to include open-ended participation from users, gateways and market-makers.

Ripple Consensus Protocol Algorithm - Ripple protocol requires each node to define a Unique Node List (UNL). The UNL comprises of other Ripple nodes that are trusted by the given node not to collude against it. Consensus in the Ripple network is achieved by each node by consulting other nodes in its UNL. Each UNL has to have a 40% overlap with other nodes in the Ripple network [12]. Consensus happens in multiple rounds where each node collects transactions in a data structure called "candidate set" and broadcasts its candidate sets to other nodes in its UNL. Nodes validate the transactions, vote on them and broadcast the votes. Based on the accumulated votes, each node refines its candidate set and transactions receiving the largest number of votes are passed to the next round. When a candidate set receives a super-majority of 80% of votes from all nodes in the UNL, the candidate set becomes a valid block or in Ripple terms a "ledger". This ledger is finalized and considered the "Last Closed Ledger (LCL)" and added to the Ripple blockchain by each node. Next round of consensus is started with newer transactions and pending transactions that did not make it into the last round of consensus. Consensus in the entire network is reached when each individual sub-network reaches consensus.

Stellar Consensus Protocol - Stellar Consensus protocol algorithm [13] uses the concept of quorums and quorum slices. Quorum is a set of nodes sufficient to reach agreement. A quorum slice is a subset of a quorum that can convince one particular node about agreement. An individual node can appear on multiple quorum slices. Stellar introduces quorum slices to allow each individual node to choose a set of nodes within its slice thereby allowing open participation. These quorum slices and quorums are based on real life business relationships between various entities thereby leveraging trust that already exists in business models. To reach global consensus in the entire systems, quorums have to intersect. Overall consensus is reached globally from decisions made by individual nodes.

The consensus protocol works as follows. Each node first performs initial voting on transactions, also generically considered as statements. This is the first step of the federated voting process. Each node performs its selection of statements and will never vote for another statement contradicting its selection. It can however accept a different statement if its quorum slice has accepted a different one. Second step is the acceptance step. A node accepts a statement if it has never accepted a statement contradicting the current statement and each node in its v-blocking set has accepted that statement. A v-blocking set is a set of nodes one each from a quorum slice to which the current node belongs to. Quorum slices influence one another leading to quorums that agree on a certain statements. This step is known as ratification when all members of a quorum agree on a statement. Confirmation is the final step of the voting process and signifies system level agreement. This step ensures that nodes send each other confirmation messages so that all agree upon the final value of the state in the system.

Comparative Analysis

Below, we present a comparison of the consensus model categories that we discussed so far in this document. Table 1 below summarizes our findings, which are further elaborated in this section.

	PoW	PoS	PoET	BFT and variants	Federated BFT
Blockchain type	Permissionless	Both	Both	Permissioned	Permissionless
Transaction finality	Probabilistic	Probabilistic	Probabilistic	Immediate	Immediate
Transaction rate	Low	High	Medium	High	High
Token needed?	Yes	Yes	No	No	No
Cost of participation	Yes	Yes	No	No	No
Scalability of peer network	High	High	High	Low	High
Trust model	Untrusted	Untrusted	Untrusted	Semi-trusted	Semi-trusted
Adversary Tolerance	<=25%	Depends on specific algorithm used	Unknown	<=33%	<=33%

Table 1: A comparison of popular blockchain consensus mechanisms

Blockchain type - Blockchain type indicates the type of blockchain platform – permissioned or permissionless, in which the consensus model can be used. This is mainly governed by the type of membership allowed by the consensus model. While PoW and Federated BFT models are built exclusively for permissionless platforms with open-ended participation, they can technically be used with permissioned platforms but won't be ideal in that setting.

Transaction finality - Transaction finality indicates whether the transaction once added to a block in the blockchain is considered final. PoW and PoET based consensus models carry the risk of multiple blocks being mined at the same time due to their model of leader election in combination with network latencies. Since this generates temporary forks in the blockchain and there is eventual chain that becomes the main chain, transactions that were previously confirmed and ended up is losing blocks will get rejected. This leads to a probabilistic transaction finality model where clients will have to wait much longer for transactions to be confirmed and finalized. With PoS, temporary forks can co-exist for short times if validators vote in parallel on multiple chains to maximize their rewards. However, a good PoS algorithm like Casper can impose penalties for voting on multiple chains causing them to lose security deposits. This will disincentivize validators from exhibiting greedy behavior. In such systems, the chain with maximum stake behind it is the final chain. In the other models with immediate finality, once the transaction is included in the block, it is confirmed and will not be rolled back.

Transaction rate - Transaction rate is higher with platforms that can confirm transactions immediately and reach consensus fast. PoW approaches are probabilistic and have to spend significant amount of time solving the cryptographic puzzle. Therefore these models have high transaction latencies and therefore a low transaction rate. PoET might be able to support much higher transaction rate because of a faster mechanism for leader election compared to PoW. Therefore, it supports medium transaction rate. BFT based approaches, PBFT and PoS can confirm transactions fast and are expected to support high transaction rates.

Token needed? - A cryptographic token is inherently required for PoW and PoS models as their design itself is based on existence of the token. Other three models do not require a token for consensus to function. It might however be used in certain platforms using these models mainly as an anti-spam anti-DDoS measure.

Cost of participation - PoW and PoS have an inherent cost associated for participation in consensus. PoW requires expending energy, which is a resource that is external to the consensus protocol, while PoS requires nodes to buy some initial cryptocurrency to generate a security deposit for declaring interest and bonding with the platform.

Scalability of peer network - Scalability of the consensus models is its ability to reach consensus when number of peering nodes are constantly increasing. All models summarized above, except BFT and variants, have high scalability. For BFT and variants it is recommended to keep the number of peers in consensus network to less than 20. Increasing the number of peers beyond 20 causes an increase in the number of messages sent between them resulting in a huge amount of overhead.

Trust model - Trust model determines if the nodes participating in the consensus have to be known or trusted. In PoW, PoS and PoET, nodes can be untrusted as the mechanism to reach consensus is based on other means such as computational work or security deposits. As long as more than 25-50% of the network is not adversarial, consensus decisions will not be affected. With blockchains using BFT, peering nodes have to be known and registered with the system to be involved in consensus decisions. The nodes can get compromised or contain bugs in code but as long as more than 33% of the nodes are not compromised, consensus process will be intact. With Federated Byzantine agreement, each node has to ensure that it include its trustworthy nodes in its trusted list.

Adversary tolerance - The fraction of the network that can be compromised without the consensus being affected. Each consensus model has a certain threshold to adversary tolerance.

Conclusions

Consensus models used by popular blockchain platforms today are largely driven by the type of applications the platform expects to cater to and the threats it envisages to the integrity of the chain. Typically the permissionless platforms are achieving robust consensus among very high number of untrusted peers using computational or memory complexity while sacrificing transaction finality and throughput. On the other hand, the permissioned, consortium blockchains are opting for a less scalable but much higher throughput model that ensures faster transaction finality. When looking at Blockchain to solve a business problem, it is imperative to look at the scale of the intended network, the relationships between participants, and both functional and non-functional aspects (such as performance and confidentiality) before determining the right platform and the right consensus model to use. We hope that this whitepaper sheds light on the background and current landscape of the consensus models and helps in that decision making.

References

1. Bitcoin mining and its Energy Footprint, Karl J. O'Dwyer and David Malone, (pdf)
2. Stellar switches to Centralized System after node issue causes accidental fork (link).
3. Are 51% attacks a real threat to Bitcoin ? (link)
4. Impossibility of Distributed Consensus with One Faulty Process, Michael J. Fischer, Nancy A. Lynch and Michael S. Paterson (pdf).
5. The Byzantine Generals Problem, Leslie Lampot, Robert Shostak and Marshall Pease, (pdf).
6. Paxos made Simple, Leslie Lamport, (pdf).
7. In Search of an Understandable Consensus Algorithm, Diego Ongaro and John Ousterhout (pdf).
8. Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems, Brian M. Oki and Barbara H. Liskov (pdf).
9. Practical Byzantine Fault Tolerance, Miguel Castro and Barbara Liskov (pdf).
10. Majority is not Enogh: Bitcoin Mining is Vulnerable, Ittay Eyal and Emin Gun Sirer (pdf).
11. Practical Fault Tolerance Beyond Crashes, Shengyun Liu, Paolo Viotti, Cristian Cachin, Vivian Quema, Marko Vukolic (pdf).
12. The Ripple Protocol Consensus Algorithm, David Schwartz, Noah Youngs and Arthur Britto (pdf).
13. Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, David Mazieres (pdf).

PERSISTENT

About Persistent Systems

Persistent Systems (BSE & NSE: PERSISTENT) builds software that drives our customers' business; enterprises and software product companies with software at the core of their digital transformation. For more information, please visit: www.persistent.com

India
Persistent Systems Limited
Bhageerath, 402,
Senapati Bapat Road
Pune 411016.
Tel: +91 (20) 6703 0000
Fax: +91 (20) 6703 0009

USA
Persistent Systems, Inc.
2055 Laurelwood Road, Suite 210
Santa Clara, CA 95054
Tel: +1 (408) 216 7010
Fax: +1 (408) 451 9177
Email: info@persistent.com

DISCLAIMER: "The trademarks or trade names mentioned in this paper are property of their respective owners and are included for reference only and do not imply a connection or relationship between Persistent Systems and these companies."