Choosing the right blockchain will make or break your dApp

Roy Shadmon
University of California, Santa Cruz
rshadmon@ucsc.edu

December 22, 2019

Abstract

This is a survey paper that analyzes and compares the different consensus protocols and features employed in the following blockchain networks: Ethereum, Tron, EOS, Burst, Cosmos Network, Neo, IOST, and VeChain.

1 Introduction

Due to the blockchain's coveted guarantees: integrity, authenticity, and availability, developing applications on the blockchain is quickly becoming more prevalent. As more blockchain applications are being developed, the foundations running each blockchain need to compete with one another to get developers to host their decentralized applications (dApps) on their respective blockchain. There are many reasons why blockchain foundations seek and incentivize dApp engineers to use their blockchain–three being: it's an easy way to advertise their blockchain (applications expose user interest), it subsequently increases the number of unique users using the blockchain (since they're using the application), and it gives users a reason to purchase the underlying token running on top of the blockchain–all of which are correlated to the value of the respective cryptocurrency. More users is directly correlated to more unique users purchasing the respective token, which is directly correlated to the token price increasing.

Since each foundation is made up of a group of people with a large stake in the respective cryptocurrency (since many are early adopters), it is rather common (and an obvious strategy) for the foundation to incentive dApp developers to use their blockchain through financial and technical support. Although those incentives may be tempting to developers, it is extremely important for those developers to pick the right blockchain with the most applicable features for their dApp (consensus protocol, block production time, transaction fees, documentation, support, etc.). Specifically, dApp developers need to answer the following questions prior to picking which blockchain is best suited to host their dApp:

1. What do we need the blockchain for?
2. Do we need to handle a lot of transactions in a short period of time?
3. How long does it take for a transaction to be confirmed?
4. What is the reputation of the blockchain and of its foundation?
5. Will we need documentation?
6. What are the security guarantees?
7. Can the data be stored privately?
8. Do they offer developer support?

In this paper, we will discuss eight of the most popular blockchains that support Smart Contracts (Ethereum, Tron, EOS, Burst, Cosmos Network, Neo, IOST, and VeChain)[1]. The paper assumes that the reader has some background of what a blockchain is since explaining what a blockchain is could be a paper in itself. However, if that is not the case, we encourage the reader to read the paper that started the blockchain revolution [1] prior to reading this.

Our paper is structured as such: We will first introduce what a dApp is. We will highlight the consensus protocol and how the protocol mitigates attacks. We will then list possible answers to the eight questions listed above (including a performance metric comparison), and report which blockchain is best suited to the respective proposed answers. We will then present different types of dApps. We will then finish the paper with a conclusion and acknowledgements. At the end of the paper is an appendix section that contains research we did but were not able to incorporate into the main part of our paper.

2 dApps

A dApp is a decentralized application run by many nodes on a decentralized network with trustless protocols (such as a Tor [2], BitTorrent [3], or a blockchain). When we reference dApp in this paper (unless specified), we will be addressing dApps on the blockchain. Specifically, a dApp usually has some sort of graphical user interface (GUI) as the front-end and a smart contract processing the backend. The code running the dApps are made open sourced and any user can verify the integrity of the code[2].

Now some may wonder why dApps, what is the point of decentralization, why should I care? The answer to these questions, in our eyes, is quite simple: we shouldn't be forced to trust applications to run honestly and fairly–we should be able to verify what the applications are doing with our data (or at least give the opportunity to someone outside of the organization). Decentralization allows for more uptime (and consequently less downtime) and cheaper costs as project owners only need to pay for the resources their dApps use. Moreover, dApps running on the blockchain are nearly hack-proof (given the code is bug-free and the blockchain network remains honest[3]). As a developer or project owner, you should care because there isn't any single point of failure to the processes running the application (given the network remains up-and-running), there is no data loss (every event is logged on the blockchain), and any interaction to the dApp is cryptographically secure (unlike other applications that just require a simple password for ultimate access). As a user, you should also care for decentralization as no one can spoof your identity and you will benefit from the application having no downtime.

2.1 Smart Contract

A smart contract is code that facilitates specified interactions between untrusting parties on the blockchain as it can behave as an automated trusted third-party. Smart contracts can be trusted to enforce specified terms and run code in a deterministic way. Smart contract code is public and immutable–once it is published to the network, any party can read and verify the terms of the contract prior to engagement and be assured that the terms will not change (unless specified in the contract).

Once published to the blockchain, the smart contract is given an address that is capable of storing the blockchain's underlying token. Smart contracts are processed on a virtual machine running on the blockchain and are able to store and query data. Users are able to engage with the contract through sending transactions that trigger specific functions in the contract. In addition, smart contracts can be programmed to include an access control list, as well as time-sensitive parameters. For example, a parent can easily set up a contract that gives their child a fixed allowance exactly once a week. Once published to the network, their child can trigger a function once a week (no more) that releases the set allowance to their wallet. Moreover, smart contracts can be used in applications

[1] It is impossible to survey all blockchains, so part of our selection process was based on the uniqueness of the the blockchain's consensus protocol. For example, we did not want to pick eight blockchains all of which use Proof-of-Stake as that would hinder the diverseness of our paper.

[2] If a dApp does not open source their code be wary as that is not the standard.

[3] An honest blockchain means that the majority of the nodes forging blocks and and forming consensus on those blocks are behaving with integrity.

running on the internet to enforce a service-level agreement (SLA) (given that the SLA covers some programmatic event). In addition, smart contracts are often used as the backend of a dApp since developers can create a myriad of data structures to store any type of data. In short, a smart contract is a trusted third-party that enables trustless transactions made and is supported by the following blockchains we will discuss [4, 5, 6, 7, 8, 9, 10].

3 Ethereum

Ethereum is one of the largest, if not the largest, blockchain at the time of writing. It currently is going through heavy improvements as the current consensus protocol (Proof-of-Work) isn't able to handle the transaction throughput (the number of transactions per second the protocol is able to process) currently being made. In addition, Ethereum aims to have a transaction throughput compared to that of the Visa (45,000 tx/s)[4].

We did research on Ethereum's proposed Casper protocol [11] which aimed at being a stepping stone to improving Ethereum's transaction throughput, however, after finishing our research the Ethereum foundation cancelled Casper and stuck with Proof-of-Work for the time being. Since we already did the research, we'll explain Casper in the Appendix of this paper as we find it rather interesting. Some features of Casper can also be possibly used to improve the consensus protocols used in Neo and Cosmos Network, which we will discuss in those sections.

3.1 Proof-of-Work

Ethash is Ethereum's implementation of the Proof-of-Work (PoW) consensus protocol that is energy heavy compared to other staking protocols [4, 12]. It is slightly different to the PoW used in Bitcoin (discussed in the Appendix) as the Ethereum's Ethash protocol is ASIC proof [13].

In Proof-of-Work miners compete with each other to be the first to produce a valid block. Miners validate a set of pending transactions and if the transactions are valid they group them into a block. After creating a block, the miners then hash their block's metadata with a number (also known as a nonce), where their goal is to find the resultant hash to be that of the target hash provided by the network. The first miner to compute a hash that matches the target hash propagates their block to the network. The network will validate the block and if they deem the block to be valid (the hash is that of the target hash), they will restart the mining process (with a new set of transactions) and try try to build a succeeding block to the just approved block. Otherwise, they will continue mining their block until they or another miner proposes another block.

Ethereum's Ethash requires miners to scan through all the previous block headers to compute a seed for each block they're mining. From the seed, the miners compute and store a pseudorandom cache. From the cache, the miners generate a dataset, where each item in the dataset depends only on a small number of items within the cache. In addition, the dataset grows linearly with time. The mining process thus involves hashing random slices of the dataset with the block metadata. The large dataset is only updated once every 30,000 blocks, which makes the majority of the miner's effort reading the dataset and not making changes to it. The reading addition aspect is what makes Ethereum PoW mining ASIC proof as there is currently no ASIC capable of doing this task.

3.2 Attack Mitigation

Common attacks to PoW protocols are 51% attacks, long range revision attacks, selfish mining, double spending, and replay attacks.

3.2.1 51% Attack

A 51% attack is when more than 50% of the network's compute power collude to behave maliciously (processing invalid transactions). The attack consists of node(s) participating in the attack to start a fork from a previously validated block with the goal to have their malicious chain overcome in length the current chain. If they were to succeed, the rest of the network would be forced to build off of the new longest (yet malicious) chain so they can compete in block production to earn block rewards. The malicious miners would then have successfully changed

[4]Transactions per second (tx/s)

the history of the ledger–specifically, any block that was validated after the fork on the honest chain. This is a major threat to Ethereum as it allows the colluding actors to rewrite the ledger's history. Although this attack is rather expensive to achieve (> 50% of the compute power requires a lot of compute resources), if done there is no defense against it besides the set of honest nodes trying increase the amount of resources they use to mine in hopes of decreasing the malicious nodes > 50% compute power.

3.2.2 Double Spending

Double Spending is where an attacker tries to spend the same money twice–possibly through a replay attack (described in section 3.2.4). Ethereum defends against this attack by requiring a nonce (different from the mining nonce) to be associated to each transaction from a single address. For example, a miner won't process transaction 6 from address A if it hasn't processed transaction 5 from address A. An attack is able to double spend, however, if they can successfully complete a 51% attack.

3.2.3 Selfish Mining

Selfish mining is when a miner (or a pool of miners) do not propagate a successfully forged block, continues mining a successive block after producing their secret block, and only propagating their secret block once another miner propagates a competing block. Competing blocks are two valid blocks with the same block height. Once both blocks are propagated to the network, the network must choose which block to continue building on top of: the selfish chain or the honest chain (causing a fork)[5]. Since the selfish chain already started mining a new block, they have an advantage and can statistically propagate a successive block faster than the honest chain. If the selfish chain is able to propagate a successive block faster than the honest chain, then the selfish chain will be the longest chain, which will force the entire network to begin mining new blocks off of the selfish chain. As a result the attacking group produced two consecutive blocks without giving a fair chance to the rest of the network (which allows them to control which transactions are getting validated). Unfortunately, selfish mining is extremely difficult to detect, and the network hopes miners do not selfishly mine.

3.2.4 Replay Attack

A replay attack is when an attack tries to propagate the same transaction twice (possibly to double spend tokens or interact with a smart contract in an invalid way). To prevent this type of attack, every transaction from a single address must contain an ordered unique nonce as described in the double spend attack.

4 Tron

Tron is a popular independent blockchain that was originally a fork off of the Ethereum blockchain. Tron aims to decentralize the web, where through its consensus protocol is able to support a high transaction throughput.

4.1 Delegated Proof-of-Stake Protocol

Delegated Proof-of-Stake (DPoS) is a widely used consensus protocol that democratizes who has the right to forge the next block on the longest chain. DPoS allows for any participant of the network to stake tokens (cryptocurrency) and vote on a delegate (block producer) that will participate in producing blocks onto the longest chain. Staking tokens requires participants to freeze part or all of their token balance, but what differentiates it from PoS (described in section 12.1.1) is that any person can influence which block validator will protect their interest in keeping the blockchain secure. The main difference in PoS compared to DPoS is that in PoS any person that owns a specified minimum amount of tokens may produce blocks, where as in DPoS validators' right to produce blocks isn't directly correlated to the amount of tokens they can stake, but of the number of votes they receive from the network. In addition, anyone who owns some tokens and chooses to stake may vote for a delegate and is

[5]Since both blocks are valid, the network will naturally split into two equally competing forks

rewarded the proportion of their vote of the total amount of votes used to elect their delegate when their delegates produces a block. Moreover, a common misconception between DPoS and PoS is that DPoS is more decentralized than PoS, however, that is not true. In PoS, there isn't a limit on how many nodes may validate blocks–the only requirement is that they hold and stake a significant amount of tokens, which increases the barrier of entry to participate in the protocol (we must trust the rich to validate transactions). On the other hand, in DPoS, there is a finite amount of block validators, which makes the protocol (possibly) less decentralized, however, significantly more democratized (the rich and/or the poor can validate transactions).

In Tron's DPoS, a fixed number of 27 delegates, also known as Super Representatives (SRs) take turns validating pending transactions from a queue and producing blocks at a fixed rate. Since no work needs to be done (like in PoW), the time it takes to process transactions is significantly reduced and less energy expensive. The order of which delegates produce blocks is randomized in a deterministic manner such that all delegates are able to agree on the order of who produces the next block.

DPoS works because the delegates confirming and verifying transactions are supported and trusted by peers who have a stake in the network, which as a result of malicious behavior would affect the way peers' vote for their delegate. Therefore, if a block producer becomes corrupt or fails to produce a block during their assigned turn, then the stake holders would re-stake their tokens (re-vote) for another block producer. Since there's a finite number of allowed block producers (all of which receive financial rewards when they produce blocks), block producers are incentivized to produce valid blocks at the risk of being removed as a delegate (and losing their right to earn block rewards).

Producing blocks in DPoS is rather simple. When a user submits a transaction, they must provide their assumption of the state of the blockchain that is based upon the recent blocks they witnessed. If consensus of the longest chain changes, then it would invalidate the user's assumptions when they first had signed (consented) to their transaction. Specifically, users must include a hash of a recent block in their transaction, and if a block producer cannot validate the block hash specified in the transaction is part of the longest chain, then the validator rejects the transaction. This protocol is called Transactions as Proof-of-Stake (TaPoS), and it ensures that transactions made from an orphaned block or minority chain will not be validated on the majority chain and is widely used in PoS or DPoS protocols.

4.2 Attack Mitigation

Although the following attacks in Tron are possible, the main preventative measure used to ensure delegates are honestly producing blocks is through the voting incentive.

4.2.1 Delegate produces an invalid block

There is no measure that prevents a delegate from validating an invalid transaction or forging an invalid block besides the obvious disincentive where voters will remove their vote for the malicious delegate who will lose their right to forging blocks. In addition, given the following delegate is honest, they will ignore the block say at height x the malicious delegate produced and produce a competing block at block height x. This process will cause a fork, and the next delegate will decide which fork is valid.

4.2.2 Double Spend and Replay Attacks

An attacker cannot cause a double spending attack because of TaPoS.

4.2.3 51% Attack

In Tron, a 51% attack requires the majority of producers to collude and become corrupt. Though this is possible, the corrupted delegates are disincentivized from doing this as they would be voted out by the network. If the attackers were able to control > 50% of the total stake, they would forever be able to control the majority of the delegates. This is also disincentivized as a corrupted network would diminish the value of the underlying token, which would ultimately affect the malicious nodes the most since they own a large stake in the network.

4.2.4 Long Range Revision Attack

Long range revision attack is similar to a 51% attack, however, rather than starting the fork 2 or 3 blocks in history, the attack would start 100s/1000s of blocks in time (or even at the genesis block[6]. This attack is unlikely to occur for the same reasons as the 51% attack in the previous section 4.2.3.

5 EOS

EOS expresses itself as a next generation, open source blockchain protocol with flexible, performant utility for businesses across the world. Using DPoS as its consensus protocol, EOS is able to separate itself through free transactions and a high transaction immutability speed.

5.1 BFT Delegated Proof-of-Stake

Byzantine Fault Tolerant Delegated Proof-of-Stake (BFT-DPoS) is a consensus protocol that is exactly like DPoS (described in section 4.1), however, the 21 delegates work together to validate blocks. Specifically, the main difference of how EOS forms consensus compared to Tron is how they determine which chain is valid during a fork. Tron determines the valid chain to be the longest chain, where EOS determines the valid chain to be whichever chain $\frac{15}{21}$ (super majority) of the current delegates support (prior work such as PBFT [14] and Paxos [15] have proven BFT mathematically). If a chain doesn't have the support of a super majority, then the default valid chain is the longest chain. Since EOS delegates are guaranteed to be delegates for one round containing 126 blocks, this mechanism allows for a new set of delegates to override blocks in the case of malicious behavior occurring in a round.

5.2 Attack Mitigation

Most of the attacks in EOS are similar to that of Tron. We will add a possible attack specified in the EOS' white paper [7] that are a result of integrating a BFT protocol in the consensus mechanism.

5.2.1 Delegate produces competing blocks

If a delegate were to produce competing blocks (two blocks with the same height) if they weren't sure which chain was valid, then the network would remove the delegate from the protocol as this action is cryptographically verifiable. Delegates are responsible for verifying all transactions and prior blocks and not guessing which chain is valid.

6 Burst

Burst is the first blockchain to support smart contracts, which they called at the time Automated Transactions (AT) that was intended to compete with the likes of Bitcoin. Although they were first blockchain in the smart contract space, they never took off. Recently, they started heavily improving Burst in an attempt to compete with Ethereum. Since they use a rather interesting consensus protocol that is $\frac{1}{500}$ more energy efficient than Bitcoin's PoW, we decided to present this blockchain and hopefully make it more known among the blockchain community.

6.1 Proof-of-Capacity

Proof-of-Capacity (PoC) is an energy efficient consensus protocol (compared to PoW in section 3.1) that consists of two processes: plotting the hard drive with data (preparation stage) and mining plotted data to forge blocks [9]. The data (plots) are files stored on hard drives that are the results of repetitively hashing data 8,912 times using the Shabal-256 [16] cryptographic hashing algorithm. Shabal hashes are relatively hard to compute compared to

[6]The genesis block is the first block on the chain.

SHA-256 hash functions [17, 18], however, easy enough to do smaller live verifications which has obvious benefits. Specifically, the plots contain all the computations necessary to process transactions and forge blocks such as address' balances. Plots are also bound to the validator's Burst account ID, which makes it impossible for two competing validators that have different Burst account IDs to have (or produce) identical plots (forcing all validators to do the work once). Moreover, when the validators generate the plot files, they generate nonces that are used to calculate Deadlines, which represents the number of seconds that must elapse since the last block was forged before the validator is able to forge a block. Each nonce has its own identifier in the range of 0 to $2^{64} - 1$ and is organized into 4,096 different places of data called scoops, where each scoop stores two of the 8,192 hashes. After the validators process as much data in respect to their storage capacity (producing plot files with different nonces), the validators are able to start the mining process by computing Deadlines.

Once the validator proposes the lowest Deadline they're able to compute, they submit the Deadline to the wallet (a node connected to the network), which will verify the validity of the Deadline. The wallet will wait until the deadline period has passed before announcing the forged block to other wallets. If, on the other hand, the wallet receives a forged block prior to its block's Deadline passing, then it will verify the validity of the block and accept it if it's valid. Otherwise, it will ignore the block and wait for its Deadline to pass to announce its block and claim the block reward if it then has the smallest deadline. Moreover, to keep the average block production time to four minutes, the base target (the target Deadline time) dynamically changes and is calculated from the time in between each of the last 24 blocks. The lower the base target, the more difficult it is for a miner to find a shorter Deadline. What makes Burst more energy efficient than PoW systems is that the plots only need to be calculated once (they are reused for each round) rather reproduced continuously for every block.

6.2 Burst Optimizations

When the Burst white paper was first released [9], the throughput of the Burstcoin blockchain was 1 tx[7]/second. Since then, the Burstcoin community has been able to increase the throughput to 80 tx/second using a new Multi-Out and Multi-Out Same transactions technique explained [19], which also reduces the cost of transaction fees as multiple transactions can be coupled into one transaction. In the future, another major improvement will infinitely scale the number of tx/second the network can handle through Dymaxion [9], which will use an Atomic Cross-Tangle Transaction feature explained [9] and out of the scope of this paper.

6.3 Attack Mitigation

Since we will currently not recommend using Burst as a blockchain to develop dApps ontop of, we will not go into too much detail about attack mitigation. One interesting attack is the 51% attack, and since Burst is designed to compete with Bitcoin, we will only discuss how it prevents the 51% attack compared to PoW protocols.

6.3.1 51% Attack

In PoW, whichever node forges a successful block (their block is included in the longest chain) reaps the reward of producing the block. In Burst, however, rather than block winners receiving the block reward, they earn votes. Those votes must then be submitted to a future block, where the miner forging the future block must accept their vote (approving their block) for the original validator to earn the block reward. This process ensures that blocks are being re-verified. As a result, the more votes accepted on blocks on a certain chain, the stronger that chain is.

7 Cosmos Network

Cosmos Network is powered by independent and parallel running blockchains (also known as shards) each running Tendermint as their consensus protocol. The shards help increase the transaction throughput by acting

[7] tx stands for transaction

as a multithreaded blockchain, where each shard is responsible for a certain set of addresses to process transactions of. Each shard communicates with other shards through an inter-blockchain communication (IBC) protocol explained in [10].

7.1 Tendermint

Tendermint, initially created [20] and improved in [21], is a partially synchronous leader-based BFT consensus protocol used in the Cosmos Network [10], where at every new round validators attempt to form consensus together on the next block (consensus is done one block at a time). In addition, the protocol assumes up to f byzantine nodes and the total number of nodes n to be $n > 3f$ (the protocol will remain secure given that the total byzantine validators will have less than $\frac{1}{3}$ of the total voting power), as well as the validator set remaining fixed. Nodes also send messages to each other through the gossip protocol and all nodes can verify messages from each other using public-key cryptography. Every round is also locked, meaning the next proposing validator cannot propose the next block until consensus on the current block is made. In addition, validators must stake tokens like in PoS or DPoS protocols.

Each round consists of four steps: *PROPOSAL*, *PREVOTE*, *PRECOMMIT*, and *COMMIT*. In the *PROPOSAL* step, the dedicated node multicasts a potential signed block to all the other nodes who can verify the node is the dedicated block proposer[8]. Moreover, there is a time period for the proposer to propose a block, which prevents the protocol from waiting and guaranteeing protocol timeliness (there is also a time deadline to vote for the *PREVOTE* and *PRECOMMIT* stages). *PREVOTE* and *PRECOMMIT* are votes to accept or reject the proposed block. They can also be thought of "This is my opinion" and "This is what I see the consensus will be based on the *PREVOTE* messages I received", respectively. Specifically, upon a validator receiving the *PROPOSAL* message and $2f+1$ *PREVOTE* messages of the same response (on the respective *PROPOSAL* message within the respective timeout period, the validator will then send its *PRECOMMIT* message. This *PRECOMMIT* message will either be to approve or reject the block.

7.2 Attack Mitigation

7.2.1 Validator sends accept vote for invalid transaction

If a validator were to send a vote in support of an invalid block, then they risk their stake getting slashed.

7.2.2 Validators Crash

If $> f$ validators crash or become byzantine through a network partition or failure, then the network would come to halt. Tendermint does not have a protective mechanism to prevent the network from halting, which is a significant problem. Casper has a mechanism where unresponsive nodes lose part of their stake for ever round of votes that they miss. Eventually, the byzantine nodes' stake would deplete enough such that the active nodes' stake would be able to form a super majority. This obviously cause a decrease in f and since Tendermint doesn't support new nodes joining/leaving the protocol, it seems like a lot of work to implement. Casper on the other hand does support these features.

8 Neo

Neo is a blockchain that aims to be an open network for the smart economy. Through their unique consensus protocol provides a strong safety guarantee, it comes with the risk of the network halting (its liveness).

8.1 Delegated Byzantine Fault Tolerance

Delegated Byzantine Fault Tolerance (dBFT) is the consensus protocol used to process blocks on the NEO blockchain [22] that combines the delegation aspect of Tron and the consensus process in Tendermint (described

[8]The proposers are deterministically picked in a weighted round-robin fashion based on their voting power.

in section 4.1 and 7.1, respectively) [23]. dBFT enables all members of the network[9] to determine which nodes should be block validators. Tendermint consensus is based off of other BFT consensus protocols such as PBFT [14] and Paxos [15] and also described in section 7.1. Thus, we will not take the time to explain it again.

8.2 Attack Mitigation

The attacks on Neo are the same as of Cosmos Network. There is an additional attack to Neo that isn't possible in Cosmos Network because of a feature in the block structure called a spam attack.

8.2.1 Spam Attack

Neo blocks currently support up to 20 free transactions (transactions that don't include a transaction fee), which are selected based on the order in which they are sent to the network (there is a FIFO queue). Thus, a spam attack can occur in Neo when malicious users spam the network with a bunch of empty transactions (since nothing is at stake), which would require honest nodes not wanting to pay transaction fees to wait for an indefinite period of time before their transaction will be validated. There is currently no mechanism to prevent the spam attack, and Neo recommends to at least include a small transaction fee with all transactions to avoid a transaction from getting stuck in the queue (remember that transactions are final–once a valid transaction is propagated, there is no way to prevent it from being included in a block in an indefinite period of time in the future).

9 IOST

IOST is a blockchain that aims to unleash the power of the blockchain. Through a rather interesting and trusting protocol, they are able to support a high transaction throughput.

9.1 Proof-of-Believability

Proof-of-Believability (PoB) is an intra-shard Believable-First approach consensus protocol used on the IOST blockchain. PoB is similar to DPoS (used in EOS and Tron), however, the selected block producers are eventually forced out for a δ-period of time to ensure that a robust unique number of block producers have the opportunity to produce blocks throughout each day. Specifically, the protocol divides all validators into two groups: believable and normal. Believable validators are responsible for processing transactions quickly (phase 1) and normal validators verify the validity of the processed transactions by the believable nodes to provide finality (phase 2). Only after the normal nodes (a larger group of nodes) validate the the processed transactions in phase 1 are the transactions considered finite.

Specifically, nodes are awarded Servi, which is a non-tradable, self-destructive, and self-issuance token that measures a node's contribution to the community and encourages members to contribute to the development of IOST. Servi is automatically deposited to a node's account after certain contributions to the network, such as providing community services, evaluating services provided by other entities, and/or validating believable nodes' processed transactions. In a sense, Servi aims to represent the believability score of a respective node [5]. After each round, the Servi balance of each committee member (believable node) is reduced by the balance of the 17^{th} believable node. Given that the size of the normal node list is positively large, every new round will have a new set of believable nodes since at least one believable node will have a Servi value of 0 and the Servi of all normal nodes in the queue only increases. In comparison to DPoS, there is no mechanism that forces a change of the elected block producers, which PoB has. Ultimately, the way in which the PoB protocol determines the nodes who produce blocks makes the nodes validating blocks more robust [24].

In addition, believable nodes are incentivized to process only valid transactions as the punishment of validating an invalid transaction is specifically harsh. A believable node who signs a block with an invalid transaction loses all their tokens and reputation (Servi), and all defrauded users are compensated for any loss they may

[9] People who own Neo tokens.

have experienced. As a result, PoB allows for high transaction throughput because transactions are processed as believable-first and normal nodes secondly validate the transactions are valid [5].

9.2 Attack Mitigation

9.2.1 Believable node validates invalid transaction

If believable nodes validate invalid transactions, then the normal nodes will report them to the network. The punishment is very severe, where the malicious node will lost all its reputation, staked tokens, and all previous transactions will be checked. Therefore, there is no incentive for this attack.

10 VeChain

VeChain is a public blockchain that is based off of Ethereum. We selected VeChain because of its interesting consensus protocol.

10.1 Proof-of-Authority

Proof-of-Authority (PoA) is a consensus protocol similar to PoS or DPoS that stakes a block producer's public reputation/credibility rather than their financial stake. To stake reputation, block producers must publicize their identity and be approved by the the foundation running the blockchain who verifies their identity. Because their identity (reputation) is staked and they must remain a significant token balance above a certain threshold they are incentivized to act honestly as they would also be harmed by the network losing trust. Through the mentioned requirements, block producers are incentivized to ensure the network remains secure. Any mistakes or malicious acts would immediately cause for removal as one of the block producers.

In PoA systems, there are a set number of authority masternodes (AMs) responsible for producing blocks. Blocks are produced in a consistent set interval of time, and every available AM has an equal opportunity to be selected to produce the next block. To prevent an attacker from attempting a DoS attack on an AM producing the next block, the protocol uses a deterministic pseudo-random process (DPRP) based on the AMs' status are "active" (able to produce a block) or "inactive" (unable to produce a block) to determine the particular order of which AM will produce block $B(n,t)$, where n is the block number (or block height) and t is the timestamp. Moreover, unlike in PoS where block producers are punished for failing to propose a block during their allotted time, no punishment is bestowed if an AM fails to produce a block. Missing a block is deemed an acceptable event as it would cause the order of which block producers create the next block. In a sense, this would also make it significantly more difficult for an attacker to predetermine which block producer will be responsible for producing a number of consecutive blocks at a relatively far time from now. It is also important to state that AMs are not required to always be online.

10.1.1 Advantages

There are many advantages to PoA such as there's no minimum requirement for the total number of available AMs to produce blocks, unlike in BFT-style protocols (since BFT requires inter-node communication to reach consensus). One obvious concern would cause the question could there ever be a situation where all block producers are unavailable (since there is no punishment for being offline)? The answer to this question is no. Unlike PoW, where block producers must solve a complicated cryptographic puzzle or BFT, where nodes must multicast messages to reach consensus, PoA has a very low requirement for computational power (no work is needed to prove the block is valid) and the AMs don't need to communicate to reach consensus (AMs are trusted). Because there is no requirement for inter-node communication or computational work to be done, the PoA protocol is able to support near-instant transactions.

10.2 Attack Mitigation

10.2.1 Catastrophic Crash

A catastrophic crash is when majority of the AMs stop producing blocks. VeChain is guaranteed to make progress if at least one AM is active.

10.2.2 AM misses their assigned block

If an AM fails to produce a block during their assigned time, then the network latency becomes 2δ, where δ is the time it takes to forge one block. The foundation is OK with AMs missing blocks because the deterministic pseudo-random process (DPRP) that determines which AM will produce the next block takes into account if a node fails to forge a block–causing the entire order to change.

10.2.3 DoS

Since the AMs are publicly known, an attacker can calculate which AM is responsible to produce the next block. This attack is prevented by the DPRP algorithm explained above as if one AM misses their assigned slot, the attacker would have to recalculate which AM to try a DoS attack–a very difficult and improbable task.

10.2.4 51% Attack

A commonly discussed attack on the blockchain is a 51% attack, which is when the majority of the network collude to produce invalid blocks (in PoW or PoS it's when > 50% of the total network compute power or total stake are colluding, respectively). In PoA, however, a 51% attack occurs when more than half of the currently available AMs are maliciously colluding. This attack is mitigated since the colluding nodes risk the foundation removing them from consensus. In VeChain's history, only one AM has been removed.

10.2.5 Long Range Attack

Long range attack in PoA is when an attacker proposes a block from an old block and tries to broadcast it to the network in an attempt to override the existing chain (causing a fork). In PoA, this kind of attack isn't possible because there must be a δ-second interval between each block, which would make it impossible for the fork to ever overcome the longest chain as it would always be behind the current chain. In addition, this attack would only succeed in the presence of a 51% attack since the valid chain is whichever chain the majority of the AMs recognize it to be.

11 Evaluation

Please view the performance metrics in Figures 1, 2, and 3. The rest of this section will be used to answer the originally proposed questions, as well as, some sample dApps.

11.1 What do I need the blockchain for?

The blockchain is very good at supporting applications where there is something measurable at stake. For example, any application that must guarantee data integrity, secure cryptographic authentication, efficient transferring of money, or any SLA that can be expressed pragmatically is a good candidate to be a blockchain dApp. The blockchain inherently handles data integrity as only those with specified permissions can modify the state of data. The blockchain also inherently provides authentication through public key cryptography–given that users are able to keep their keys secure (which is an assumption of the blockchain). Typically, if there's nothing at stake in the application, then the blockchain might not be the best fit for the application as the dApp doesn't need to ensure the provided guarantees. One issue to consider when developing smart contracts for dApps is the expensive cost given

it is a new field and a relatively small proportion of engineers are advanced smart contract developers (supply and demand).

11.2 Do I need to handle a lot of transactions in a short period of time?

Based on Figure 1 using a blockchain such as IOST, EOS, Cosmos Network, or Tron might best suit your needs if you need to support a high transaction throughput. State channels [25] in Ethereum can process transaction throughput equal to the latency of the network connecting the sender and receiver as not all transactions need to be processed on-chain. State channels allow for user(s) to deposit money into a smart contract (both of which agreeing to the SLA of the contract), and sending signed messages (containing off-chain transactions) to each other that if a dispute were to occur, each party can send the signed messages to the smart contract which will determine what action to take. If transaction throughput isn't of importance to your application, use Ethereum. In addition, Ethereum is currently developing the Serenity protocol (also known as Ethereum 2.0) [26], which will significantly increase the transaction throughput as the protocol uses PoS. Once Serenity protocol is released, we recommend to use Ethereum as it's the most advanced blockchain.

11.3 How long do I want to wait for immutability?

Based on Figure 3, if you need quick immutability, the best blockchain to use is EOS. Otherwise, you should use Ethereum.

11.4 What's the reputation of the blockchain?

The blockchain with the best reputation is Ethereum and EOS. The blockchain with the worst is Tron as it has a lot of negative talk surrounding it, however, we see Tron as an up-and-coming and viable solution.

11.5 Will we need good documentation?

You will always need good documentation. Ethereum by far has the best documentation, followed by VeChain, Tron, and EOS.

11.6 Are there any security problems I need to be concerned of?

We're not a huge fan of BFT style consensus as it has the potential for the network to halt. We recommend using any blockchain that can prevent the network from halting as that would hinder the success of any dApp.

11.7 Do I need my data to be stored privately?

Currently, no blockchain is able to store data privately. All data is publicly viewable. The only way for data to be stored privately is if the inputted data is encrypted prior to being stored. Enigma is a blockchain company that is designing a secure computation protocol where "secret nodes" in the network perform computations over encrypted data. Simply put, each node in the protocol executes a different part of the computation to ensure privacy. The protocol uses a technique called secure multi-party computation that is out of the scope of this paper and details can be found here [27].

11.8 Do I need developer support?

While writing this paper, We made successful contact with developers from Burst, IOST, and VeChain–all of whom were very nice. Blockchains such as Ethereum and Tron are so large that it's hard to get in contact with someone capable of answering technical questions. It seems to us that any blockchain that isn't too popular will provide a lot of developer support as they are incentivized to attract dApp developers, where Ethereum and Tron are so well known they don't need to do as much outreach.

11.9 Sample dApps

Some applications that would benefit being on the blockchain include all gambling apps, apps that require pragmatic SLA enforcement, supply-chain, and more. It is crucial to understand the blockchain only provides tools to ensure integrity, authenticity, and availability guarantees. It is up to the developers to develop secure code that can take advantage of the blockchain, as well as ensuring their code remains up-to-date with the newest security features.

11.9.1 Gambling dApps

Blockchain is particularly useful to enforce the rules and game play of gambling dApps. Specifically, users should be assured the casino isn't cheating, and the casino should be assured that the user is gambling money that belongs to them (though the puclic-key guarantee). In addition, the steps of which casino games are played can also be pragmatically designed to ensure fairness. Moreover, both parties want to be assured that they'll receive their winnings and the blockchain is perfect to enforce that.

11.9.2 SLA Enforcement

Applications for IoT, data services, cloud services and more would benefit from the blockchain as it would help them enforce and be more transparent with their users if outages occurred. For example, if some company guarantees that they'll return any query in 5 seconds, then that is easily enforced in a smart contract. This would require we trust the code that measures the latency of the query, however, the measuring code should be open sourced and verifiable for both sides. In addition, don't forget that it's the users choice to engage with a smart contract SLA, so while you write one for your dApp, make sure it's fair or for your users' requirements.

11.9.3 Supply Chain

Supply chain is another huge industry that could benefit from using dApps. The history of changes to the supply chain are immutable and verifiable, which allows for more efficient transparency between the application and the users. Rather than users needing to call and ask for updates, they can seamlessly query the blockchain and be assured that the data is most up-to-date (given the company built their system in a transparent way).

12 Conclusion

Overall, we think it's fairly clear that Ethereum is the best blockchain to develop dApps ontop of–especially once the Serenity protocol is released. Although there are many other choices, it's hard to recommend users to develop on other blockchains simply because there isn't comparable man power working on other blockchains. Because of all the development going on with Ethereum, we see Ethereum as the future of blockchain dApps and other blockchains as successful, yet smaller competitors.

Acknowledgements

We'd like to thank the active members of Burst, IOST, and VeChain for answering our questions on Discord, Telegram, and Slack.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] T. Project.

[3] BitTorrent. https://www.bittorrent.com.

[4] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," *https://github.com/ethereum/wiki/wiki/White-Paper*, 2015.

[5] I. Foundation, "Internet of services: The next-generation, secure, highly scalable ecosystem for online services." https://github.com/iost-official/Documents/blob/master/Technical_White_Paper/EN/Tech_white_paper_EN.md.

[6] VeChain, "Vechain thor." https://cdn.vechain.com/vechainthor_development_plan_and_whitepaper_en_v1.0.pdf.

[7] EOSIO, "Eos.io technical white paper v2." https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[8] T. Foundation, "Tron: Advanceed decentralized blockchain platform." https://tron.network/static/doc/white_paper_v_2_0.pdf

[9] S. Gauld, F. von Ancoina, and R. Stadler, "The burst dymaxion." https://www.burst-coin.org/wp-content/uploads/2017/07/The-Burst-Dymaxion-1.00.pdf.

[10] J. Kwon and E. Buchman, "Cosmos whitepaper." https://cosmos.network/resources/whitepaper.

[11] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[12] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[13] Ethereum, "Ethash." https://github.com/ethereum/wiki/wiki/Ethash.

[14] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[15] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[16] E. Bresson, A. Canteaut, B. Chevallier-Mames, C. Clavier, T. Fuhr, A. Gouget, T. Icart, J.-F. Misarsky, M. Naya-Plasencia, P. Paillier, *et al.*, "Shabal, a submission to nist's cryptographic hash algorithm competition," *Submission to NIST*, 2008.

[17] P. Gallagher and A. Director, "Secure hash standard (shs)," *FIPS PUB*, vol. 180, p. 183, 1995.

[18] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," in *International workshop on selected areas in cryptography*, pp. 175–193, Springer, 2003.

[19] T. Créance, "1st hard fork explained: changes in transaction dynamics." https://www.burstcoin.ist/2018/05/04/1st-hard-fork-explained-changes-in-transaction-dynamics/.

[20] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, 2014.

[21] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on bft consensus," *arXiv preprint arXiv:1807.04938*, 2018.

[22] NEO, "Neo white paper: A distributed network for the smart economy." https://docs.neo.org/en-us/whitepaper.html.

[23] E. Zhang, "A byzantine fault tolerance algorithm for blockchain." https://docs.neo.org/en-us/basic/consensus/whitepaper.html.

[24] T. Wang, "PoB – IOST'S Consensus Algorithm: How to Achieve Decentralized Consensus." https://iost.io/pob-iosts-consensus-algorithm-how-to-achieve-decentralized-consensus/.

[25] EthHub.io, "State channels." https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/state-channels/.

[26] EthHub.io, "Ethereum 2.0 (serenity) phases." https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/eth-2.0-phases/.

[27] Enigma, "We're securing the decentralized web." https://enigma.co.

[28] Ethereum, "What is proof of stake." https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs.

[29] H. Partz, "Samsung produces asic chips for new halong bitcoin miner." https://cointelegraph.com/news/samsung-produces-asic-chips-for-new-halong-bitcoin-miner.

[30] N. De, "Intel wins patent for energy-efficient bitcoin mining." https://www.coindesk.com/intel-just-won-a-patent-for-an-energy-efficient-bitcoin-miner.

[31] M. J. S. Smith, *Application-specific integrated circuits*, vol. 7. Addison-Wesley Reading, MA, 1997.

[32] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on bft consensus," *arXiv preprint arXiv:1807.04938*, 2018.

[33] T. Créance, "The burst mining process explained." https://www.burstcoin.ist/2017/10/16/the-burst-mining-process-explained/.

[34] burst wiki, "Burst wiki." https://burstwiki.org/en/.

[35] I. Foundation, "Servi node guide: Requirements and on-boarding." https://medium.com/iost/servi-node-guide-requirements-and-on-boarding-abcae30f18e1.

[36] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 583–598, IEEE, 2018.

[37] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017.

Appendix

12.1 Casper

Casper is a hybrid protocol that uses both Proof-of-Work and a heavily modified Proof-of-Stake algorithm that is rather complicated. Thus, we will first present what normal Proof-of-Work and Proof-of-Stake is, and then present Casper.

12.1.1 Proof-of-Stake

Proof-of-Stake (PoS) is a consensus protocol that grants the right for a set of block producers to propose the next block on the longest chain. Any stake holder (a person owning the native token of the blockchain) in the network is able to be deposit some or all of their tokens to be provided the right to produce blocks, and their overall influence is directly proportional to the number of tokens they stake compared to the entire stake of the network. There are two types of PoS protocols: chain-based and Byzantine Fault Tolerant (BFT-based). Since most PoS protocol do not use chain-based PoS, we will only highlight chain-based PoS, and focus the rest of this subsection on the BFT-based PoS protocol. We will then introduce Casper, Ethereum's hybrid PoS and PoW protocol, BFT-style protocol.

12.1.2 Chain-based PoS

Chain-based PoS uses a pseudo-random algorithm that allocates block producers a time slot where they are given the right to append the next (a single) block to a previous block (which would typically be the most recent block of the longest chain). The only problem with chain-based PoS is that it doesn't protect against the "nothing at stake" problem: that behaving maliciously is not expensive. As a result, chain-based PoS prioritizes more on availability (the next block can always be built) over consistency (the validity of the block) since the next producer producing block n could connect it to the $n-2$ block causing a fork.

12.1.3 BFT-based PoS

BFT-based PoS consensus algorithm is built off of proven mathematical properties, such as if at most $(\leq)\frac{1}{3}$ of the weighted protocol participants are malicious (the set of malicious block producers whose combined stake is proportional to $\frac{1}{3}$ of the entire network stake), then the protocol guarantees that no two conflicting blocks (regardless of network latency) will finalize. In addition, BFT PoS guarantees are based off of early BFT research such as PBFT [14].

Moreover, BFT-style PoS works by randomly assigning validators to a timeslot where they are given the right to propose the next block. Agreeing on a block, however, is done through a multi-round process, where each validator sends votes for a specific block during each round. Only at the end of each round do the validators permanently agree on which block is part of the chain. One major problem with BFT PoS is that it's not obvious how to hold validators accountable (for producing or voting for invalid blocks), how to verifiably detect unresponsive validators, and how to update the validator set.

12.2 Casper Protocol

Casper is a consensus protocol that repeats 99 PoW blocks (described in section 3.1) and one modified BFT-style PoS block[10] that provides accountability, a way to update the validator set, and defenses against long range revision attacks and if more than $\frac{1}{3}$ of validators are byzantine [11]. For efficiency and security purposes, Casper forms a checkpoint tree rather than dealing with the full blockchain (the genesis block to present), which is used to finalize all child blocks of the checkpoint block (the parent block) and permanently invalidate all child blocks not connected to the checkpoint block. Moreover, the genesis block is the first checkpoint and every block number that is an exact multiple of 100 is also a checkpoint.

[10] At the time of writing, Casper is going through some modifications and we will update the paper with any future changes.

Consensus on checkpoints is formed through vote messages sent by validators. Each vote message must contain: two checkpoints s and t with height $h(s)$ and $h(t)$ where checkpoint s must be an ancestor of checkpoint t (or else the vote is invalid). In addition, the validator set is public and finite for the respective vote and if neither the public key of the validator nor the validator's signature are valid, the vote is not considered. In addition, there are two commandments required for all validators to follow or else the violator's stake is slashed: a validator shall not publish two unique votes for the same checkpoint height and a validator shall not vote for an earlier checkpoint after voting for a more recent checkpoint. The two commandments also address the "nothing at stake" problem.

Another major difference of Ethereum's Casper protocol is the fork choice rule. In most blockchains[11] other than Ethereum, the standard mechanism is to "always build atop the longest chain". However, there are scenarios where Casper gets stuck if a checkpoint isn't able to achieve a super majority such as if some blocks built atop the longest chain cannot be finalized. In this case, it would take for altruistic validators to validate the checkpoint at the cost of losing their stake. To avert this problem, the fork choice rule is to follow the chain containing the justified checkpoint of the greatest height.

Although typical BFT consensus protocols, such as [14, 15], do not support nodes joining/leaving the validator set, Casper does by defining the dynasty of block b, which is the number of finalized checkpoints in the chain from the genesis block to the parent block b. Thus, if a node were to join/leave the protocol, they would have to make a deposit (of 32 Ether) or withdraw transaction, respectively. Given the transaction is included in block d, they would join or leave the protocol in the $d+2$ block.

12.2.1 Attack Mitigation

Long range revision attacks in Casper are when a coalition of validators who at one point had more than $\frac{2}{3}$ of the stake during a checkpoint in the past try to use their historical super majority to finalize a conflicting checkpoint without the risk of their stake being slashed (it's already withdrawn). Simply, this attack is infeasible because of the fork choice rule and because validators have already seen a finalized checkpoint block at the same height and will refuse to revert it.

Another possible threat to Casper is in the case of a catastrophic crash where more than $\frac{1}{3}$ of validators crash-fail in the same moment, which could happen due to a network partition, computer failures, or a malicious coalition. If this were to occur, no super majority would be able to form–causing no future checkpoints to be finalized. To prevent this event from occurring, Casper instituted an "inactivity leak" where any validator who doesn't vote for checkpoints has their deposit slowly drained. Eventually, the byzantine validators' deposits will be burned enough such that the current voting validator set will control enough stake to form a super majority.

12.3 Bitcoin Proof-of-Work

Bitcoin [1] PoW requires each miner to place as many pending transactions as possible into a block (more included transactions results in a larger reward) and repeatedly hash the block's unique header metadata with a constantly changing the nonce value. The miner would then compare the result with the current target hash. The first miner to find a hash that matches the current target hash (this process is a competition) will broadcast the block throughout the network, where each other miner will validate and add the block to their copy of the ledger given that the block is valid. If the miners deem the proposed block to be invalid, then they will keep mining their current block until they or another miner broadcasts a new proposed block. Otherwise, whichever miner won the block round (by proposing the accepted block) will win the block's reward, and a new round of mining a block will begin where the process above restarts. The process of repetitive hashing has been so profitable that companies such as Samsung [29] and Intel [30] (and more) have built hardware called ASICS to do this single specific task [31] to mine Bitcoin.

12.4 Nothing at Stake Attack

Nothing at stake is a problem that all consensus protocols must prevent in order to work. Nothing at stake is a problem where there's no disincentive for a block producer to produce an invalid block. In normal BFT consensus

[11] We are unable to claim "for all" blockchains due to the impossibility of researching 2000+ blockchains/cryptocurrencies

protocols such as PBFT [14] and Paxos [15], validating nodes can send byzantine messages to halt the protocol from making progress. All the protocols we present in this paper solve this problem either through being forced to use computational resources (electricity), risking tokens (money), or risking votes/reputation.

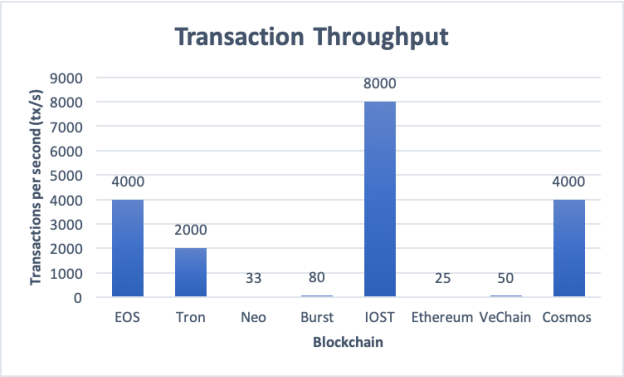

Figure 1: This figure shows the amount of transactions per second each blockchain processes. Note that some blockchains claim a higher throughput, however, we decided to graph the all time high throughput of each blockchain.

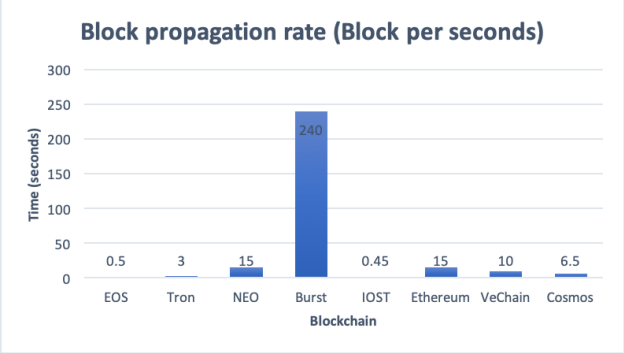

Figure 2: This figure shows how often blocks are generated.

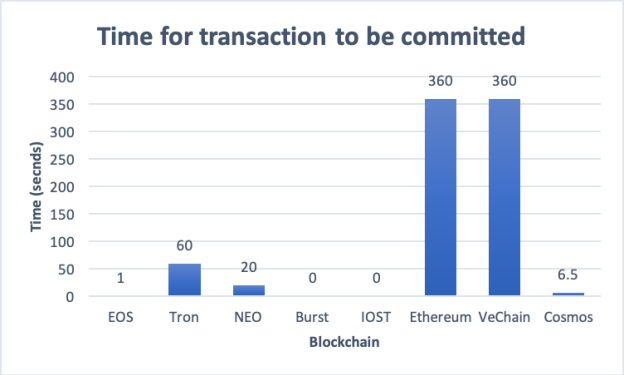

Figure 3: This figure shows how long it takes for a transaction to be deemed immutable. Please note that some values are estimates. We also couldn't find any data on Burstcoin or IOST, and thus left their time to be 0.