IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A graph model based blockchain implementation for increasing performance and security in decentralized ledger systems

Konstantinos Tsoulias[1], Georgios Palaiokrassas[1], Georgios Fragkos[2], Antonios Litke[1] and Theodora Varvarigou[1]
[1]National Technical University of Athens, Zografou Campus, 15773 Athens, Greece
[2]Dept. of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 87131, USA

Corresponding author: Georgios Palaiokrassas (e-mail: geopal@mail.ntua.gr).

The research leading to these results has received funding from the European Commission under the H2020 Programme's project M-Sec (grant agreement nr. 814917).

ABSTRACT Blockchains are being recently used as a supporting technology framework for decentralized applications requiring functionalities such as exchange of value through tokens, cryptocurrency and smart contracts. In this paper, we have developed a decentralized application model in Python, where blockchain data are stored in a Neo4j graph database. Following the basic principles of Ethereum blockchain network, we implemented a Casper-like consensus mechanism and tested its effectiveness in achieving finality. For block proposing, we employed both Proof of Work and Proof of Stake protocols and examined how participants' incentives and consensus criteria differ according to each one. A major part of this work is to incorporate the graph model in the functionality of the blockchain and its components, while also exploiting its benefits in data analysis by finding relationships between data and extracting their true value. Through this approach, we were able to monitor and visualize changes in blockchain data in various use case scenarios. Lastly, we ran a series of simulated experiments to test the efficiency of the implemented technologies and mechanisms in preventing the most common blockchain attacks such as the 51% Attack, Catastrophic Crashes and Attack from dynamic validator sets. We show how the modelling of the blockchain data as a distributed graph can assist protocols operations, enhance their security, and facilitate the application of analytical methods to the stored information through path-dependent queries.

INDEX TERMS Blockchains, Proof of Work, Proof of Stake, consensus mechanisms, graph databases, blockchain security, Casper, Neo4j.

I. INTRODUCTION

Blockchains are regarded as both public and private ledgers containing transactional data within their decentralized data structures, which form a series of tightly connected, timestamped blocks [1]. Their unique architecture makes blockchain systems immutable in the sense that transactions cannot be tampered once they are officially validated and registered in a block of the chain[1]. Based on cryptographic proof, blockchain technology abolishes the need for a trusted third party, enabling for reliable and robust decentralized applications, implemented on open and trustless networks of peers [2]. Blockchains have been used as the underlying technology for many cryptocoins and tokens [3], setting the ground for disrupting the future Internet [4] as well as the traditional business model by providing new means for exchanging value. Thus, the research on various features of blockchains has become very important in order to be able to enhance the technological framework with characteristics that can pave the way for a wider adoption of blockchain technology. One such feature is finality, that needs to be achieved through consensus protocols [5], assuring cryptocurrency transactions cannot be changed, reversed, or canceled after being published in the blockchain. Finality in Bitcoin's [6]

[1] A series of connected blocks that starts on the Genesis Block.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

blockchain is achieved with the Proof of Work (PoW) protocol that requires users' CPU power to link new blocks of transactions to the existing blockchain, and thereby forming a continuous record that cannot be altered without redoing all the work. In the case of a fork, this process, also known as mining [7], encourages users to always mine on top of the longest chain, since it came from the largest pool of CPU power and so it is the most difficult to reproduce.

The inherent characteristics of blockchain architecture, like transparency, verifiability, privacy, and anonymity, have encouraged since then, various industries and operational domains to further explore its numerous benefits and applications [8]. Blockchain technology has also its drawbacks, with scalability [9], security, and energy consumption [10] problems being the most significant. Nonetheless, new protocols and solutions are continually being developed [11]-[14] to address these problems and to consolidate the blockchain technology and the decentralized model, potentially transforming the way people choose to transact globally [15]. One such example is the Proof of Stake (PoS) protocol [16] that attempts to restrict PoW's wastefulness, by using tokens instead of computational work, as a scarce and well-distributed resource to prevent cheap attacks to the blockchain.

However, PoS stakeholders' incentives [17] differ from those of PoW miners' in a way that may compromise network's security. Virtually the most profitable tactic for a stakeholder is to vote on every branch of the blockchain tree[2], thus making it harder to identify the most reliable chain and reach a clear consensus. To tackle the so-called Nothing-at-Stake problem [18], Ethereum [19] developers created a partial consensus mechanism, called Casper [20], that combines the PoS research and Byzantine Fault Tolerance (BFT) [21] consensus theory. Casper overlays an existing blockchain and offers the appropriate tools and regulations to readjust participants' incentives [22], so that they always consent to the most secure chain. This technology is so recent that it has yet to be tested in a real cryptocurrency, leaving some problems associated with still open.

Along with the troubleshooting, efforts are also being made to involve new tools and test new approaches in blockchain technology [23]-[25], expanding its capabilities and applications. In this context, and because of the high interconnection of blockchain data, the representation of blockchain as a distributed graph database is far from absurd. Relationships between its data, keep blockchain coherent, and may bear information of great analytical value. Only a database that natively embraces relationships is able to store, process, and query those connections efficiently. While other databases compute relationships at query time through expensive JOIN operations, a graph database stores connections alongside the data in the model, allowing millions of connections per second to be traversed.

In this paper, we have developed a decentralized application model in Python that is connected to a Neo4j database [26], where blockchain data are stored. Following the basic principles of Buterin and Griffith's original paper [20], we practiced a Casper-like consensus mechanism to function alongside the most popular block proposal mechanisms: the PoW and the PoS protocols. A major part of our work was to incorporate Neo4j in the functionality of the above mechanisms and ultimately improving their performance. For that reason, we developed a versatile Graph Model for our blockchain database that allows for a multilevel viewing of the stored data and, by extension, numerous ways of accessing them. From the Neo4j Desktop [27] application, we were able to monitor and visualize changes in the deployed graph database in various use case scenarios. Another advantage of the blockchain graph database is the ease in applying analytical methods to the stored data and to the relationships between them with graph analysis tools. This innovation could solidify the blockchain analytics field by facilitating the evaluation of blockchain's components and the behavior of the network's nodes. For this reason, we ran a series of simulated experiments and by utilizing the annotated graph model, we tested the efficiency of the implemented technologies and mechanisms in preventing the most common blockchain attacks; namely the 51% Attack, Catastrophic Crashes, and the Attack from dynamic validator sets.

The rest of the paper is structured as follows: In Section II we present the theoretical background for the tools and mechanisms developed in this paper. In Section III we discuss briefly about the published work that technically relates to blockchain and the ideas proposed in our paper. In Section IV we describe the architecture of the decentralized application, while details about the implementation and the functionality of its components are given in Section V. In Section VI we run our application and test the performance of the employed blockchain data model and the security of the implemented protocols against the most common blockchain attacks. Section VII is the conclusion of this paper, where we summarize our findings and suggest possible applications for the mechanisms we developed.

II. BACKGROUND

A. CASPER CONSENSUS MECHANISM

Casper is a partial consensus mechanism combining Proof of Stake algorithm research and Byzantine fault-tolerant consensus theory. Casper's operations are backed by a group of particular nodes, the validators [28], who are responsible for voting on checkpoints and finalizing transactions. A checkpoint is only a regular block, whose height in the blockchain tree is an exact multiple of a number. In Ethereum, for instance, this number is set to 100, so through the resultant checkpoint tree, validators can finalize every 100 blocks at once, rather than voting on every single block.

Every node can become a validator by depositing at least the predetermined minimum amount of tokens. The number

[2] The forking of chains in the ledger results in tree like structure rooted at the Genesis Block.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

of tokens deposited also represents the stake of the validator, which rises and falls with rewards and penalties. A node's voting power is determined by his share of the number of tokens deposited by all validators. Hence, when we say "2/3 of validators", we are referring to the deposit-weighted fraction. To exit the validator sets and collect his share a node must publish a withdraw message. After exiting, the node is forever forbidden to re-enter the sets. Validators can broadcast a vote message containing four pieces of information: two checkpoints of the same subchain[3] s and t, together with their respective heights h(s) and h(t). Therefore a vote can be represented with a link from a source to a target checkpoint.

If at least 2/3 of the validators (by deposit) have published the same vote with source s and target t, then s → t is called a *supermajority link*.

A checkpoint c is called *justified* if (1) it is the root, or (2) there exists a supermajority link c' → c where checkpoint c' is justified.

A checkpoint c is called *finalized* if (1) it is the root or (2) it is justified, and there is a supermajority link c → c' where c' is a direct child of c.

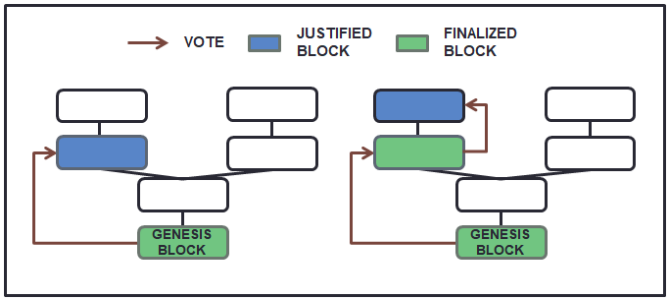


FIGURE 1. Example of justifying and finalizing checkpoints in the checkpoint tree

Casper's proper function precludes two checkpoints of different subchains from being both finalized (Figure 1). To achieve this, all validators must comply with the following rules:

An individual validator must not publish two different votes $\{ s_1, t_1, h(s_1), h(t_1) \}$ and $\{ s_2, t_2, h(s_2), h(t_2) \}$ such that either:

I. $h(t_1) = h(t_2)$.

Equivalently, a validator must not publish two distinct votes for the same target height.

or

II. $h(s_1) < h(s_2) < h(t_2) < h(t_1)$.

Equivalently, a validator must not vote within the span of his other votes.

Breach of any of the above rules results in the slashing of the offending validators (Figure 2); the permanent withdrawal from the validator sets and the deletion of their deposits. In case of a rule violation, Casper guarantees that all relevant evidence can be found, and the offenders can be identified.

For the mathematical proof of the above proposition we will be working on the checkpoint tree. Given two finalized checkpoints x_m and y_n on two conflicting subchains, there are two distinct chains of supermajority links from a common starting checkpoint s (whether that is the Genesis Block or not) to x_m and y_n respectively:

$s \rightarrow y_0 \rightarrow y_1 \rightarrow \ldots \rightarrow y_n \rightarrow y_{n+1}$

and

$s \rightarrow x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_m \rightarrow x_{m+1}$

Where, x_{m+1} and y_{n+1} are the children of x_m and y_n respectively, since x_m and y_n are finalized (finalization rule). The heights of all checkpoints x_j , y_i in the above chains should be different, otherwise rule I is violated. Without loss of generality we assume that $h(x_m) > h(y_n)$, hence that $h(x_m) > h(y_{n+1})$, since $h(x_j) \neq h(y_i)$. Let k be the lowest integer such that $h(x_k) > h(y_{n+1})$; then $h(x_{k-1}) < h(y_n)$ (or $h(x_{k-1}) = h(y_n)$, which again violates rule I). This implies the existence of a supermajority link $x_{k-1} \rightarrow x_k$, where $h(x_{k-1}) < h(y_n) < h(y_{n+1}) h(x_k)$, thus violating rule II. If two conflicting supermajority links l_1 and l_2 exist, we can conclude that at least 1/3 of the validators violated the slashing conditions, since at least 2/3 of the validators have published l_1 and at least 2/3 of the validators have published l_2.

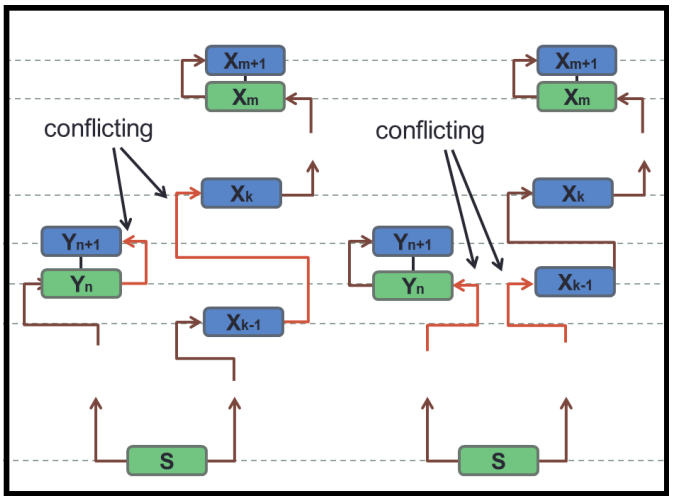


FIGURE 2. Proof of the effectiveness of Casper's slashing conditions.

In the case of a fork, miners/stakeholders are incentivized to always build on the branch that contains the highest justified checkpoint. This correct-by-construction fork choice rule [29], besides being the optimal strategy for nodes, also prevents pathological scenarios to occur; by following the longest chain fork choice rule, Casper can get "stuck" where any blocks built atop the longest chain cannot be finalized without some validators getting slashed. So, this rule is to be followed by every miner/stakeholder since it ensures the liveness of the consensus protocol

[3] A series of connected blocks that starts on a fork of a chain.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

B. NEO4J GRAPH DATABASE

A graph database (GDB) [30] is a database designed to treat the relationships between data as equally important to the data themselves. Graph databases are part of the NoSQL databases created to address the limitations of the existing relational databases. This is achieved by using graph structures for semantic queries with nodes, edges, and properties to represent and store data. A graph database is intended to hold data without constricting them to a predefined model by storing connections alongside the data in the model, while other databases compute relationships at query time through expensive JOIN operations. Hence, accessing nodes and relationships in a native graph database is an efficient, constant-time operation.

Neo4j [26] is an open-source, NoSQL, highly scalable native graph database that provides an ACID-compliant transactional backend for developing applications. This means that it efficiently implements the property graph model down to the storage level while using pointers to navigate and traverse the graph. Performance-wise Neo4j delivers consistent, real-time efficiency for multi-hop queries on large, interconnected datasets. Moreover, it offers a versatile property graph model that allows for fluidly evolving solutions to meet user's requirements. Cypher [31], a declarative query language similar to SQL, but optimized for graphs, is now used by other databases like SAP HANA [32] Graph and Redis graph [33] via the openCypher project [34].

The property graph model of Neo4j organizes data as nodes, relationships and properties. Nodes are the entities in the graph that can hold any number of attributes (key-value pairs), called properties. They can also be categorized into labels, that each represents a specific role for the nodes tagged with it. Two semantically-relevant nodes can be linked with a directed relationship. Relationships are characterized by their type, and like nodes, they can too hold properties. Additionally, due to the efficient way in which they are stored, any number or type of relationships can be shared by two nodes without sacrificing performance.

Neo4j also offers a growing, open library of graph algorithms [35] that are optimized for fast results. With little to no coding required, these algorithms reveal the hidden patterns and structures in the stored connecting data around pathfinding, centrality and community detection. Lastly, Neo4j Browser, a graphical user interface (GUI) that can be run through a web browser, allows for querying, visualization, and data interaction. All these capabilities make Neo4j the ideal tool to employ in representing, visualizing and analyzing the cumbersome and highly connected blockchain data.

III. RELATED WORK

A consensus protocol is a fault-tolerant set of rules that ensures all nodes agreement on the order in which entries are appended to the blockchain, despite the malicious or ambiguous acting of individual nodes amongst them. The CPU voting consensus that Nakamoto suggested with Proof of Work (PoW) [6] encouraged a multitude of new mechanisms based on proof of concepts [36] to try and tackle PoW's problems while maintaining a similar level of security. Proof of Stake (PoS), the most popular and energy-saving alternative to PoW [10] [37], requires participants to prove the ownership of the amount of currency, expecting a strong correlation between a node's wealth and its fidelity. Our application incorporates the above protocols as block proposal schemes and highlights the adjustments needed for them to work appropriately under a Casper-like consensus mechanism.

Other protocols such as Proof of Activity (PoA) [38] combine useful elements from both PoW and PoS. Operating PoA requires building blocks from miners via PoW, which are controlled and signed by active network stakeholders. The hash of any new block header solved by a miner is mapped to one of the satoshis in the network. Then a procedure is followed to track its owner, who then is responsible for signing the new block header. This process is repeated N times, for the new block is published. As it understood, like in a PoS scheme, the more tokens a node possesses, the more chances he has to be elected. The protocol is called Proof of Activity because it also requires the N stakeholder to be active; otherwise, another block header (with different N stakeholders) will be the first to sign.

Casper [20], the partial consensus mechanism is perhaps the most advanced PoS algorithm; it's innovation is so new that it has yet to be thoroughly tested in a large scale environment. Recently, Ethereum's developers announced the first release [39] of Casper Friendly Finality Gadget (FFG) and the code was made available to researchers, auditors and client developers, to start testing the software. Essentially, Casper FFG is a simplified version of a Byzantine fault tolerant protocol [21], with "votes" for checkpoints taking the place of prepares and commits. Shortly, Ethereum 2.0 [40] is expected to be launched, which will include Casper CBC [41]; an upgraded version of Casper FFG that will complete the transition from PoW to PoS consensus. However, researchers have already been examining the effectiveness of Casper's principles, the incentives involved and mechanisms through individual decentralized application models. Such an example is presented in the work of Moindrot *et al.* [42] where a simulation of a blockchain application in Python was developed to familiarize the reader with Casper's basic operations, as well as to examine the impact of latency and disconnected nodes in the protocol's finality. However, this simulation is far from a working DApp since it does not involve active users, and some key blockchain and Casper components, like consensus protocols and dynamic validators sets were not implemented.

Furthermore, the Ethereum project has adopted the GHOST protocol, that suggests an alternative to the longest-chain rule of common PoW protocols; that is,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

selecting the heaviest sub tree rooted at each fork. By doing so, developers aim to avoid scenarios where an attacker's chain can grow longer without him having the majority of network's computing power. An example of that is when larger blocks are created that take longer to propagate through the network, thus resulting in more forks to occur. In that case, the Greedy Heaviest-Observed Sub-Tree rule proposes that those off the main chain blocks can still contribute to it's validity. That idea of using graphs as a way to optimize performance and security of distributed ledgers was further examined in the cryptocurrency space. A well-known blockchain protocol that introduces a unique graph based model in blockchain is IOTA's tangle. This protocol is entirely based on a DAG, which is used for storing and verifying transactions by connecting them to others, already confirmed. Nevertheless, this implementation differs in many ways from the typical blockchain structures since it doesn't use blocks to store transactions, it combines the roles of transaction issuers and transaction approvers and unlike most protocols it doesn't include monetary rewards. Acknowledging the tremendous benefits that graph solutions can provide in distributed ledgers, we propose a model, that stores and connects blockchain's digital entities in a Neo4j database.

Modeling blockchain as a graph database is not a novel idea [43]. Several studies [44]-[46] have highlighted the analytical value within the blockchain data and the relationships between them that can be optimally exploited through a high-fidelity blockchain graph model. In [44] specifically, this was done, by parsing and deserializing the Bitcoin raw binary data files into a suitable format for importing into Neo4j. Then, they ran the annotated graphthrough a graph-analysis framework that uses path-dependent Cypher queries to extract and summarize useful statistics. This implementation paves the way for a blockchain analytics field that focuses on identifying and even predicting behaviors in both the nodes and their published messages. In our paper we extend the idea of a graph blockchain database by also incorporating the graph model into the core functions of blockchain and its mechanics. Furthermore, we suggest a both flexible and lean blockchain model that negates the need for a locking-unlocking graph mechanism by being stored alongside the traditional blockchain for a low memory overhead. This implantation intends to access data used by consensus protocols and block-proposal schemes at much greater speeds than the traditional way, ultimately resulting in higher performance decentralized systems.

IV. SYSTEM OVERVIEW

A. P2P NETWORK

Every decentralized application is supported by a P2P network [47] where members can interact with one another without the need for a trusted authority. In our model, we simulate such a network by utilizing the Python Flask Microframework. In particular, every node is implemented as a separate Web application of the same structure, to ensure equality. For the purposes of this paper, we deployed the P2P network on a single machine by having each node run on a different port of Python's local development server. This implementation allowed us to uniquely identify each node by its port number so that it functions as the node's address.

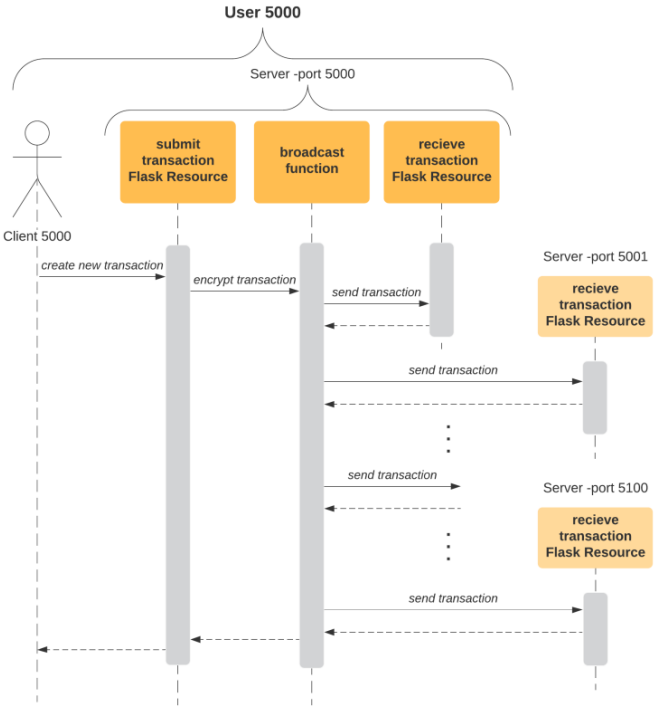


FIGURE 3. Transaction broadcasting sequence diagram.

Communication between nodes is enabled through the Flask-RESTful extension; each node stores its peers' ports-addresses and can transmit messages to them by merely invoking the suitable API Resources with a supported HTTPS method. Newcomer nodes query one or more IP addresses hardcoded into their scripts that act like DNS seeds, by storing and transmitting peers' IP addresses. The procedure followed when a node broadcasts a transaction to the network can be visualized in the sequence diagram of Figure 3.

Lastly, every peer initializes and utilizes a Neo4j distributed graph database, in which blockchain data are stored and dynamically accessed.

B. BLOCKCHAIN GRAPH MODEL

Representing common blockchain data in a Graph Database can be arranged in a forthright manner; a node can be labeled either as Block or as Transaction, with dedicated attributes in each case. Two consecutive blocks are linked with a "CHILD_OF" relationship, while transactions are connected to the Blocks they belong to, with an "INCLUDED_IN" relationship. Following this model, we can depict any data broadcasted in the network, that is stored in a Block and has attributes, as a separate node or label in the Neo4j database.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

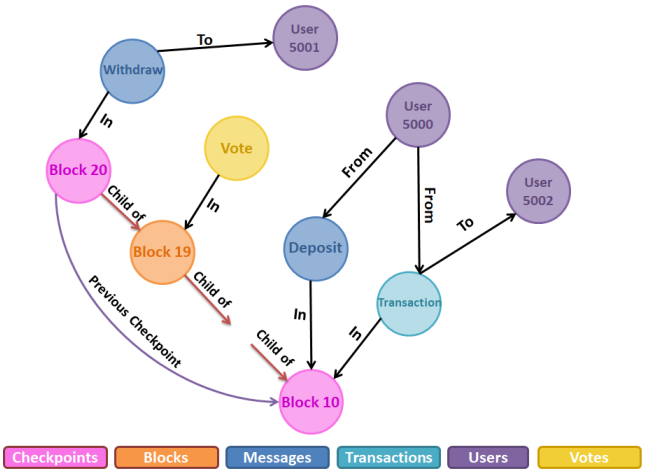


FIGURE 4. Blockchain database Graph Model.

However, the benefits of this implementation are not limited to presenting data in an organized and comprehensible way. Neo4j offers the ability to quickly access stored information by utilizing graph theory algorithms as well as to apply analytics [48] on blockchain data, by executing graph algorithms through complex Cypher queries. Nevertheless, not all of the information is worth storing in the Neo4j graph database. The separation criteria are related to the usability of the stored data in blockchain functions and their analytical value. The efficiency of Neo4j is further optimized when the graph model expedites the retrieval of inaccessible native dataresulting in a higher performance system.

Hence, the graph database design is not absolute but rather is to be considered as a versatile tool, completely dedicated to its blockchain, containing the information of value and connecting them according to the needs of its mechanisms and protocols. One such example could be calculating a user's balance, which would require finding all transactions that he participates in by crawling each block in the blockchain tree. A blockchain that values such metric should store and connect users' transactions in an optimal way. Also, another practice might involve the handling of smart contracts [49][50] perhaps in an e-shop application. In that case a useful indicator could be the credibility score of a user, calculated by retrieving the contracts they were involved in and by considering the credibility scores of the other parties as well as the method of the contract's resolution [51] (agreement, dispute, use of mediator etc.). Hence, the graph model of such an application should include smart contract nodes, link them with the users that participate in them and store a resolution method property.

Bearing in mind the above principles, we have designed a graph model that facilitates the most common blockchain's operations as well as those of Casper's consensus mechanism, while allowing for path-dependent queries to be applied and information to be retrieved in a resourceful manner. The design presented in Figure 4 takes advantage of blockchain's highly interconnected data and offers a simplified architecture that can be stored alongside the original blockchain with an additional charge of 33% (for additional details please check the Running the System subsection) in terms of space complexity, to assist the operation and the evaluation of its components. Our model classifies blockchain data into six distinct node labels. Users are linked to the Transactions they participate in and to the messages they publish. Messages are associated with validator activities and are categorized into deposit, withdraw, and slashing messages. Transactions, messages and votes are linked to the blocks in which they are added. Each Block is connected with its parent-block, while Checkpoints Blocks are double-labeled and additionally linked to the previous Checkpoint in the checkpoint tree.

V. IMPLEMENTATION DETAILS

A. BLOCK PROPOSAL MECHANISMS

1) PROOF OF WORK

In Proof-of-Work blockchains, nodes compete with each other to solve a cryptographic puzzle, like producing hashes with specific patterns. This procedure, known as mining, is implemented in our application and uses three main components: a hash function, a random number generator, and a winner verification method.

Every prospective miner first initiates a subprocess for mining the next block. The procedure begins by constructing the new block for the miner's selected chain as an object of class Block. For this block to be published, the miner must first solve it by appending random numbers to its header and hashing the resulted string. When an SHA-256 hashcode [52] with a specific amount of zeros is produced, the block is considered solved and is broadcasted in the network. The number of zeros required is defined by the difficulty parameter, with which we can adjust the average block time.

After receiving and verifying the new block, the other miners terminate any ongoing mining processes and update their local data. Python multiprocessing library allows for the mining process to be executed in parallel, without impeding the rest of the Flask Server operations.

2) PROOF OF STAKE

The Proof of Stake block proposal system, simulates the mining process by using instead of computational effort, as proof of the block constructors' reliability. Here, users who possess a larger amount of tokens have a higher chance of being selected, as they have stronger incentives to protect the network and the value of the cryptocurrency.

In our implementation, a node can apply to become a stakeholder by broadcasting an HTTPS message to the network. Nodes of the same chain will store the received application with the applicant's public key and stake. The election process is triggered by sending a "start_pos" HTTPS request to one of its nodes and is essentially a stake-weighted random choice between applicants. If the "start_pos" requests are sent to randomly selected nodes,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

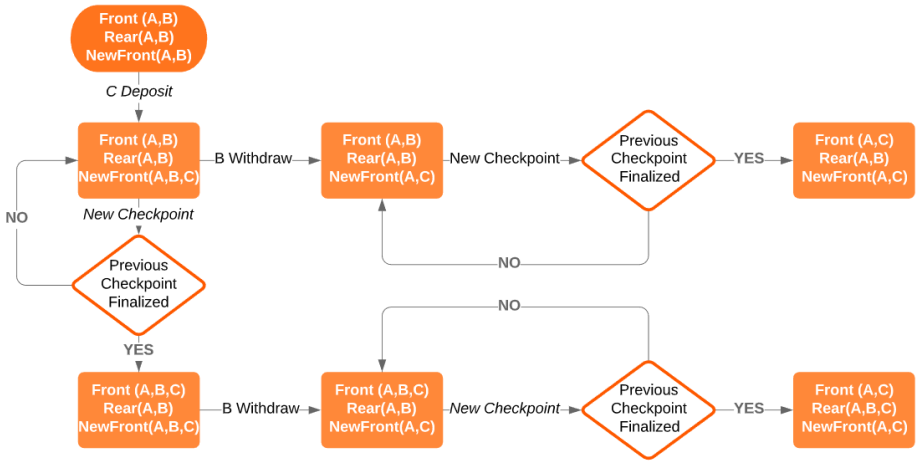


FIGURE 5. **New block's validator sets' configuration example.**

eventually, the chain with the most online nodes will grow the longest. However, this affects neither the security nor the finality of the consensus protocol, since here, unlike PoW, chain length is not a measure of difficulty.

Instead, we use the staking of each block to determine which chain was the hardest to create and thus the hardest to be reproduced. The problem that emerges here is that stakeholder incentives run counter to this security measure, since the optimal tactic for them is to bet on the blocks of every branch. Thus staking-wise, the weaker blocks are made indistinguishable from the stronger ones, and the Staking measure becomes unreliable. Casper aims to alter these incentives by assigning consensus to the validators.

B. CASPER-LIKE CONSENSUS PROTOCOL

1) DYNAMIC VALIDATOR SETS

To achieve finality in our Proof of Stake protocol, we implemented a Casper-like consensus protocol, that can work over any block proposal scheme as well. This mechanism should also allow the switching of validators without compromising blockchain's security. Our design incorporates the key principles [53] laid down by Casper's creators into one straightforward simple implementation. Specifically, we suggest the use of two dynamic validator sets responsible for voting on checkpoints; the "current_front" and the "current_rear" validator sets. In addition to these two, we utilize the "new_front" structure to store the next state of the current_front validator set, thus greatly simplifying the process. Every block stores and handles these structures, recognizing in that way its expected voters and the future state of all three validator sets. A node can become or quit being a validator by broadcasting a deposit or a withdraw HTTPS message, respectively, to the rest of the network. Every block inherits its parent's structures, and subsequently, depending on the deposits and withdrawals it includes, it adds and removes nodes from its new_front set. Changes in the voting validator sets take effect after the finalization of a checkpoint. So, if checkpoint C gets finalized, the next block B created in the same subchain, that is also a child of block A will have its validator sets modified as shown in Figure 5.

In this way, current validators authorize the next by finalizing specific blocks. After a validator's withdrawal gets finalized, he will remain in the "current_rear" set until another checkpoint gets finalized. Further, by having both front and rear validators agree, for a checkpoint to get finalized, we ensure that all new validators vote in line with their predecessors, achieving a continuous line of consensus throughout the blockchain. The final measure that we need to adopt is preventing the out of order checkpoint finalization. This step is crucial for Casper's safety since otherwise, there can occur scenarios where conflicting justification and finalization votes have been sent by disjointed validator sets,
and therefore it is impossible to trace and punish the offenders.

2) VOTING

Given the changes in the operation of the validator sets and the new security risks arising from them, we redefine a supermajority link, a justified and a finalized checkpoint as follows:

An ordered pair of blocks (s, t), has a *supermajority link*, if both at least 2/3 of checkpoint's t "current_front" validator set have published votes s → t and at least 2/3 of checkpoint's t "current_rear" validator set of t have published votes s → t.

Given an ordered pair of checkpoints (s, t), t is considered *justified*, if there is a supermajority link s → t.

Given a checkpoint s' and its direct child t', s' is considered *finalized*, if s' is justified, there is a supermajority link s' → t' and all votes justifying and finalizing checkpoint s' are included in the subchain before the creation of the next checkpoint

As with the implementation of the validator sets, we use three auxiliary data structures that are stored in each block and facilitate the voting process. In the front_votes and rear_votes structures, we store the sent links with the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

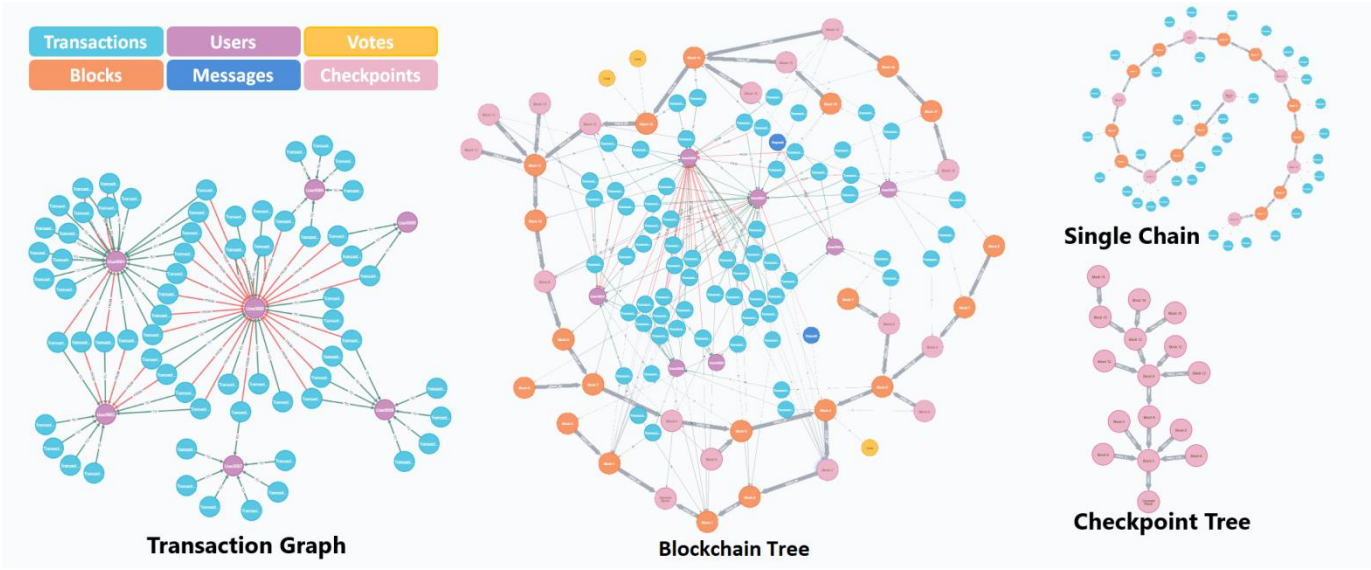


FIGURE 6. Versatility of the Blockchain Graph Model.

majority rates received, while in the "justified" list. we store the justified checkpoints.

A node broadcasts a vote s → t, via a "submit_vote" HTTPS message. For this vote to be valid, it must contain the required information; "source_hash" , "source_height", "target_hash", "target_height", "public_key". When importing such a vote into a block b, a series of tasks are performed by the miner/stakeholder:

i) checking that blocks b and t belong to the same subchain; ii) calculating the sender's stake participation in each of the validator sets of block t and confirms that at least one of them is greater than 0%; iii) adding the calculated percentages to the total vote rates that the s → t link has already received in the "front_votes" and "rear_votes" sets; iv) including t in the "justified" list in the case of s → t being a justification link, that has achieved a supermajority of at least 66%.

These data structures depict the votes included in the current block's ancestors and not in the whole blockchain tree. Hence, each block entry can be uniquely described only by the block's height.

Finally, when creating a new checkpoint, the miner will check its stored data in order to determine whether the previous checkpoint in the chain got finalized. If the finalization supermajority link exists and the previous checkpoint also appears justified, we understand that all votes required have arrived in time and are stored in blocks of the same subchain. In this case, the miner notifies peers that the previous checkpoint is to be considered finalized

VI. EVALUATION

A. RUNNING THE SYSTEM

While blockchain owes many of its advantages in the way that it organizes its data, there is scope for improvement on accessing them. A key point of our research was to suggest a graph model, that abolishes the need for crawling each block in the blockchain and allows for the otherwise cumbersome information to be optimally retrieved. In this section, we run the decentralized application and examine how the employed graph model enhances the performance of the blockchain and its components. The Neo4j Desktop application we are using, runs the Neo4j Browser version 4.0.1 and the Neo4j Server version 3.4.10, providing an environment to visualize dataand work on our local Neo4j databases. Figure 6 shows the versatility of our graph model, that can be traversed in multiple ways depending on the type of information requested. We present two instances in which we recorded significant performance augmentation by utilizing the Neo4j Graph Database; calculating balances, tracing offending validators.

The calculation of a user's balance in the blockchain tree can be performed rapidly, by pinpointing his corresponding Neo4j node in the "Users" label and parsing his "To" and "From" relationships. For the same calculation to be done along only one branch, we execute a directional paths finder algorithm from the final block b to the user node and keep all "From" and "To" relationships included in those paths.Our design allows each block to connect only to its previous block through a backwards "CHILD_OF" relationship. Since each block can have only one parent block, we can isolate a single chain from block b to the Genesis Block with a path consisting of “CHILD_OF” relationships.

The following Cypher query returns the total number of tokens sent to "ADDRESS" with transactions that are included in the chain extending from block of hash “HASH” to the Genesis block:

```
MATCH (b:Blocks{hash: HASH }) - [:CHILD_OF*0..]
 -> () - [k] - (t:Transactions) - [:TO] ->
(u:Users{hex:'ADDRESS'})
RETURN SUM(t.amount)
```

Unlike the classic blockchain structure, the graphical model here allows for bidirectional traversing of entities in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

the requested path. We can evaluate and analyze any query through the Neo4j by looking at its execution plan: The PROFILE command runs the given statement while keeping track of how many rows pass through each operator, and how much each operator needs to interact with the storage layer to retrieve the necessary data. Profiling the above query reveals how the Neo4j Planner takes advantage of blockchain's data model and optimizes the match by taking the node-degree into account when checking for the connection, starting on the smaller side while caching internally. Specifically, it avoids crawling each block and examining the hundreds of transactions contained in them; instead it follows only the transactions that the requested node participates in.

Casper bases its effectiveness on its ability to identify and punish malicious validators. In our Casper-like protocol, we incentivize the network's nodes to track and report those offenders by offering them financial rewards in the case of a successful slashing. Furthermore, all evidence for a rule violation can be discovered and recovered by any node, as all the sent votes are stored publicly in the blockchain.

The incorporation of Neo4j into our application and the complex Cypher queries it allows, further facilitates the detection of such offending votes in the blockchain. Specifically, we can request all distinct pairs of conflicting votes sent by the same validator with two simple Cypher queries:

```
MATCH (v1:Vote), (v2:Vote)
WHERE v1.r_from = v2.r_from
AND v1.target_height = v2.target_height
AND ID(v1) < ID(v2)
RETURN v1.r_from, v1, v2
```

Returns all distinct pairs of votes; v1, v2, sent by the same validator with targets at the same height.

```
MATCH (v1:Vote), (v2:Vote)
WHERE v1.r_from = v2.r_from
AND v1.target_height > v2.target_height
AND v1.source_height < v2.source_height
AND NOT ID(v1) = ID(v2)
RETURN v1.r_from, v1, v2
```

Returns all distinct pairs of votes; v1, v2, sent by the same validator, in which one vote is within the span of the other.

The above queries apply to all published votes in the blockchain tree and not in a specific branch. The process of tracking offenders is greatly simplified since sent votes are indexed with the "Votes" label and serial block access is no longer required.,

If measured by traditional DB criteria, traditional blockchain, seems poor: throughput is only a few transactions per second, capacity is a few GB and most importantly it has essentially no querying abilities, thus making it unsuitable for applying statistics on its data. Several efforts [54] have been made to improve the traditional blockchain database in terms of performance, scalability and queryability. However, these implementations follow a NoSQL approach meaning that they store sets of disconnected aggregates, that makes it difficult to use them for connected data. The most common strategy for adding relationships to such stores is to embed an aggregate's identifier inside the field belonging to another aggregate, effectively introducing foreign keys. But, this requires joining aggregates at the application level, which given blockchain's high interconnectivity quickly becomes prohibitively expensive. We find that graph databases optimally exploit the benefits of blockchain's unique architecture and create versatile data structures that can be traversed in real time.

The performance evaluation of our graph tool compared to that of the document-oriented approach is in agreement with the results of several studies [55][56] that suggest the overall superiority of graph databases regarding querying time of the connected information. Specifically, in Figure 7 we present our findings regarding the average query execution time for both our private blockchain application that follows a document-oriented database architecture and our Neo4j blockchain database in logarithmic scale. The query used in this case was a simple balance calculation for a specific user. To have a fair comparison, memory consumption in Neo4j should not exceed 13GB of RAM, which is what an Ethereum full-node uses. To achieve we set both heap and page cache to 4GB each, assuring that when combined with the extra memory that JVM needs to function correctly, Neo4j's process memory consumption will not grow beyond the desired levels.

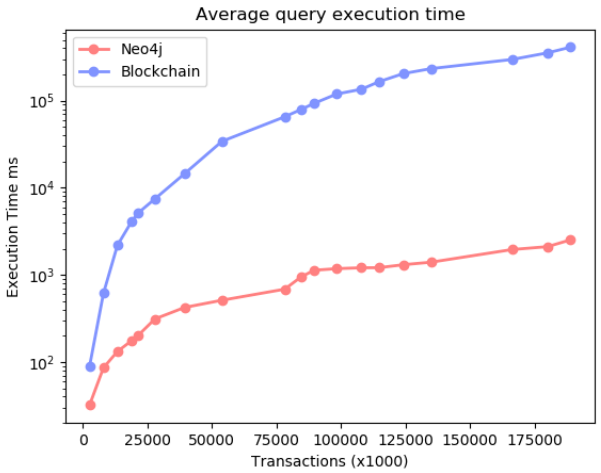


FIGURE 7. Average query execution time for Neo4j and Blockchain databases.

While that the graph model and the appropriate Cypher queries can simplify the procedures performed by the protocols and mechanisms that function in the blockchain, the space complexity of our implementation, should also be considered. We can calculate the Neo4j stored records' sizes as follows: 15 B for Nodes, 34 B for Relationships, 41 B for Properties for nodes and relationships. In our graph model, we suggest that a Transaction consists of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

following contents: the Transaction Node, 3 Relationships (FROM, TO, IN) and 2 Properties (amount, hash). Hence a published Transaction occupies $1 \times 15 + 3 \times 34 + 2 \times 41 = 199$ B of disk space. Comparing this number to the average Transaction size in Bitcoin which is 600 B, we understand that storing and utilizing our graph model alongside the traditional blockchain data would mean a 33% increase of the total blockchain size.

For the computation, memory configuration depends on how much virtual memory will the JVM for Neo4j use at runtime. Thus, the more the memory allocated matches the size of the database, the less swapping of data will occur at runtime, ultimately resulting in higher performance. Storing the 258 GB of Bitcoin's data [57] in our graph model would require almost $258 \times 0.33 = 86$ GB of memory. However, the memory used by the Neo4j instance is the collection of data requested by the client. For path-dependent queries, a precise calculation of that can be complicated since it may involve duplicate nodes and relationships. On the contrary, we can make a reasonable estimation of the memory required for the two simpler queries used to spot offending validators. In that case, we request all N pairs of votes that consist of one node and four properties. Hence, the memory required for this query to be optimally executed is $N \times N \times (15 + 4 \times 41) = 179N^2$ B. To further reduce the space complexity of our implementation and improve its effectiveness we are exploring additional graph models and tools that will entirely base their operation on them.

B. PREVENTING ATTACKS

In this section we test the robustness of the mechanisms developed against the most renowned blockchain attacks. In the case of Casper, security against several types of attacks is provided by its nature. Casper can tolerate 1/3 of the validators being malicious in achieving finality; any percent larger than that can stop the network from finalizing any new checkpoints. On the contrary, its security is only compromised when the dishonest validators achieve supermajority in both validator sets; thus being able to fully control the finalization of new checkpoints. Sybil attacks are prevented as Casper operates in a Proof of Stake manner; the size of a validator's deposit determines his voting power. To further reduce the impact of multiple-address users, Casper requires a large number of tokens being deposited, to become a validator.

While we saw how the Neo4j can assist in tracking Casper's offending validators, safety under other types of attacks is to be examined through a series of simulated experiments on our application. The vulnerability of PoS Casper systems in 51% attacks, and the optimization of parameters for a consensus protocol resistant in catastrophic crashes will be the main focal points of this section.

The simulation process is enabled through a batch script that initializes nodes and performs the basic functions of miners and validators. The above process also provides for the existence of side-chains that can be created in pseudo-random manner; that is an adjustable parameter that determines the possibility of a fork to occur on each block. The simulated results are gathered with the aid of a python script that executes Cypher queries to the Neo4j database, where the native blockchain data are stored. Thus, we once more showcase the benefit of the blockchain graph model in quickly pinpointing and extracting high-value analytics regarding, in this case, the evaluation the blockchain's mechanisms as well as the behavior of its participants.

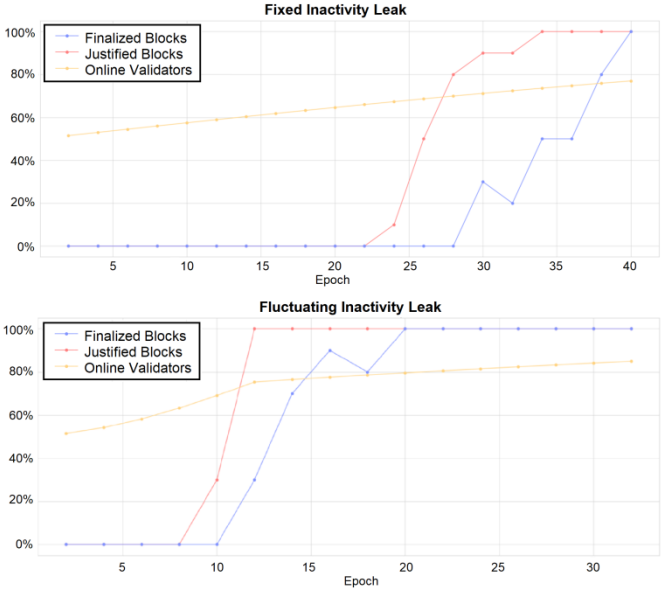


FIGURE 7. Consensus with Fixed and Fluctuating Inactivity Leak.

1) CATASTROPHIC CRASHES

If more than 1/3 of the validators are disconnected from the network due to computer failures, network partitioning, or risky behavior, it is virtually impossible to finalize new blocks. In this case, the Inactivity Leak can help the blockchain recover, by gradually decreasing the stake of the offline validators and thus weakening their voting power. Inactivity Leak's value can either be fixed or fluctuating and the money deducted from the inactive validators can either be erased or returned to them sometime after they get back online. In our study, we are focusing on optimizing the role of the Inactivity Leak in Casper's security. In other terms, penalties should be adjusted, so that the network can effectively and quickly overcome a catastrophic crash, while voting remains ultimately profitable for validators with short absences.

We examine the efficacy of a fixed and a fluctuating Inactivity Leak in achieving consensus, after a Catastrophic Crash occurs, at which 50% of current validators disconnect. To simulate this, we initialized 1000 validators of which 500 were only online and able to cast a vote. The consensus rates on justification and finalization votes can be stored in the checkpoint nodes as a separate property, while not significantly affecting the overall space complexity of our Neo4j implementation, since checkpoints are sparsely distributed throughout the blockchain tree. In both cases of Inactivity Leaks, penalties are initially set to -1% and take place per 10 checkpoints. Should consensus not be reached during that period, non-fixed Inactivity Leak decreases by 5% until at least one block gets finalized. On

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

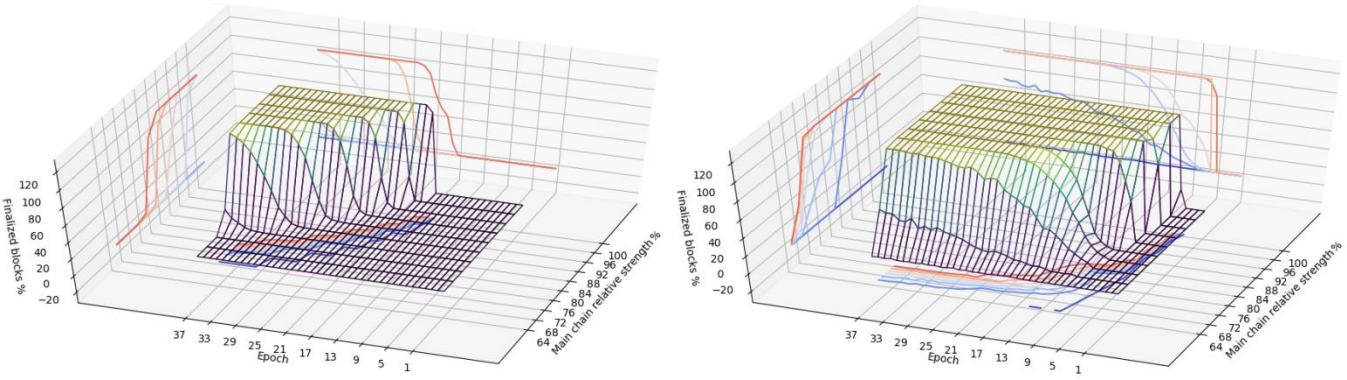


FIGURE 8. Consensus with Fixed and Fluctuating Inactivity Leaks for subchains of varying strengths

Figure 7 , we can observe the change in the percentage of active validators, justified and finalized checkpoints for a stable and a fluctuating (Figure 7) Inactivity Leak.

If we now consider the existence of additional candidate main chains in the network, we can assume that the probability of a checkpoint being voted is proportional to its subchain's relative strength. The relative strength of a checkpoint or a subchain derives from the criteria that validators use when voting. Thus, in a PoS blockchain with honest validators, the relative strength of X could be interpreted as the total amount of tokens staked on X compared with that on checkpoints at the same height. Now, we simulate the previous voting process, while taking validators, as it shrinks the time window within which validators can recover a crash without suffering extensive losses on their deposits.

2) 51% ATTACKS

The 51% attack refers to a blockchain attack performed by a group of miners that control more than 50% of the network's mining or computing power and could potentially control new transactions' confirmation to double-spend coins. Here we examine whether such an attack could be feasible in our Proof of Stake-Casper model and whether the inherent features of these mechanisms favor the Voting Power centralization amongst stakeholders and validators, respectively.

Regarding PoS, Voting Power centralization can be checked using the PoS scheme we have implemented in our network. According to this, stakeholders have a chance of being selected proportionally to their share of capital. The winning node will be rewarded with a fixed amount of cryptocurrencies. In our simulation we initialized 1000 nodes with the voting power distribution being similar to that of real PoS networks. Then, we initiate the stakeholder election process for 1,000,000 simulated blocks and check again for potential wealth accumulation. To calculate the total stake of each stakeholder in each case, we take advantage of the Cypher queries presented in Section A, which greatly simplify the process. The initial distribution of hashrate as well as the distribution after the election process is presented in Figure 9 (a) in a detailed and simplified form where only PoS stakeholders that possess into account different probabilities of the main chain being voted.

The results displayed in Figure 8 for both types of Inactivity Leak suggest that the fluctuating inactivity leak diminishes the influence that the relative strengths and the number of subchains have on the speed of reaching consensus, since in every case the first finalized block appear around epoch 9. In the instance of the fixed inactivity penalty consensus is highly dependent on the strength of the candidate blocks, delaying the first checkpoint finalization as long as 30 epochs in some cases. However, the acceleration of consensus that a Fluctuating Leak offers associates additional risk with the role of more than 3% of the total tokens are shown. The two distributions appear almost identical after 1,000,000 blocks, which means that the model followed maintains any financial differences between the nodes of the network without expanding them percentage-wise.

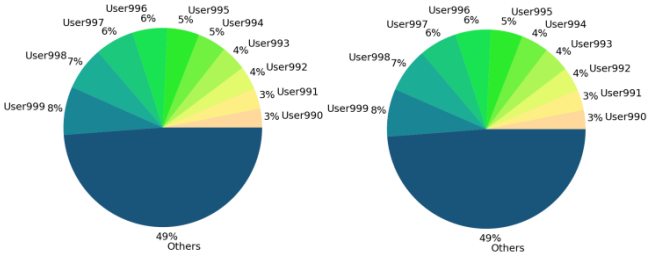
Like many other BFT protocols, Casper uses 1/3 as the maximum number of faults it can tolerate. Given n total nodes, of which there are f byzantine nodes, we need at least t nodes to agree to reach consensus. Assuming that the n-f nodes are split into two equally sized groups of (n - f) /2, we want to make sure that the influence of the byzantine nodes that may act arbitrarily isn't enough to achieve consensus. Hence $t > (n - f) / 2 + f$, ensuring that the two groups cannot decide different things and result in a safety failure. For liveness, we make sure that the n - f nodes can come to a consensus, without the cooperation of the f byzantine nodes. Thus $(n - f) \geq t$. By combining the two constraints, we get $n/3 > f$ as the fault tolerance threshold of Casper.

In this way, to adequately control Casper's finalization process, the attackers would have to acquire at least 67% of the total deposits in both validator sets. Still, the 34% in just one validator set would be enough to block the network from finalizing any new checkpoints. Another fundamental parameter is that Casper's structure should be such, that it does not amplify economic differences between validators, jeopardizing the sets' decentralized character. Other than being the backbone of Casper's operation the adopted reward-punishment system also dictates the motives in consensus groups and hence the possibility of centralization

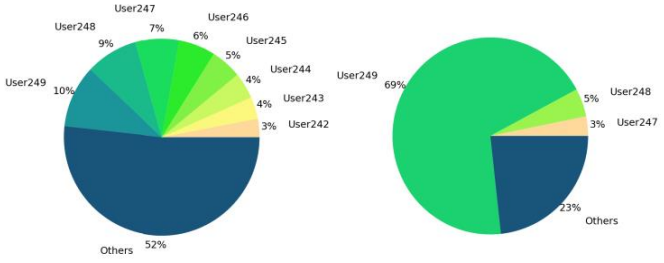
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3006383, IEEE Access

of power in them. To underline the significance of rewards, we tested two different approaches in our Casper-like protocol: a system that encourages consensus and another that encourages participation.

The results in each case were gathered with the aid of Cypher queries, for quickly pinpointing each validator's votes through the "Vote" label and retrieving their voting percentages in each set from the node's properties. While the initial distribution of PoS voting power depicted previously can be set similar to that of the real PoS cryptocurrencies, the novelty of Casper's ideas imposes several restrictions when selecting appropriate data inputs. However, the fact that Ethereum requests 1500 ETH as the minimum validator's deposit, a demand that only 5000 addresses can fulfill at the moment, in combination with the risk associated with being a validator allows for a good estimation of the front and rear sets sizes. By extension, the distribution of voting power can be directed by that of the stake distribution of the top 5000 Ethereum addresses.



(a) Example of decentralized voting power distribution in PoS before and after 1,000,000 blocks



(b) Example of centralization of voting power in a validator set before and after 2,000 checkpoints

FIGURE 9. Examining distribution of power in consensus groups.

To recover from a dead-end situation where consensus was not reached for several checkpoints, the protocol can favor groups of validators with consensus rates that exceed a regulated percentage, through bonus rewards. This scheme suggests the gradual tradeoff between system's liveness and security where the strongest consensus group will eventually prevail over the rest of the set. However, rewarding consensus underlies a voting power centralization danger. By implementing this scheme for 250 validators and running it for 2000 checkpoints, as demonstrated in Figure 9 (b), we saw, that even with all validators being honest, those with larger deposit shares are more likely to receive this bonus, and further increase their power. Thus, economic differences are amplified exponentially. Moreover, those powerful nodes would have a stronger influence on the rest of the validators who would be incentivized to follow the majority to reach consensus and earn the bonus.

The danger of voting power centralization, lurking in consensus-enforcing reward systems leads us to the participation valued approach of the original Casper. This can be achieved with a deposit-equivalent reward given to those who have cast at least N votes during a predetermined period with targets of height > h, where h is the height of the highest justified checkpoint, so that the broadcasted votes are relevant with the checkpoint finalization process. By doing so, powerful validators have no advantages over the rest of the set as long as everyone participates in the voting process. This rewarding system is deceptively more similar to that of the PoS than the previous one as long as Casper is responsible for the finalization, the only difference being that in PoS non-participating nodes face 0 projected profits instead of Casper's negative penalties. Finally, to resolve dead-ends, validator's minimum deposit is made expensive, while gradually lowering participation rewards when no consensus is reached for several periods. With this amendment, a consensus-blocking attack would be costly and less profitable for the attackers than participating honestly in the voting process.

VII. CONCLUSION AND FUTURE WORK

The establishment of decentralized applications and the widespread adoption of blockchains in mainstream financial technology applications require the refinement of the current consensus mechanisms and approaches involved, thus overcoming blockchain's safety, efficiency, and scaling barriers. In our work, we developed a fully customizable blockchain application that enabled the integration of new technologies and the evaluation of up-to-date mechanisms in the blockchain. We have shown how the modeling of the burdensome blockchain data as a distributed graph can assist protocols operations, enhance their security, and facilitate the application of analytical methods to the stored information through path-dependent queries. Besides, through the tangible representation of the data provided by graph databases such as Neo4j, we were able to monitor the fundamental processes of the consensus protocols and block proposal schemes developed. Furthermore, we adapted this implementation so that it serves the most up-to-date blockchain consensus mechanisms. Focusing on Casper's BFT consensus protocol, we showcased how the annotated model enhances network's security in deterring attacks from dynamic validator sets by quickly pinpointing conflicting votes and punishing the offenders. Finally, we ran a series of simulations that tested our approach's resilience to the most widespread blockchain attacks. In particular, we have examined through Cypher queries how the configuration of the Inactivity Leak is affecting finality's recovery from a catastrophic crash, whether a 51% attack on Casper - PoS blockchains is possible, and how adjusting Casper's

penalties and rewards can prevent hashrate centralization scenarios

Our simplified model allows for a lean and versitile blockchain implementation that exploits the benefits of graph databases over the SQL approaches in both storing and accessing the blockchain interconnected data. At the same time blockchain data analysis is enabled through graph analytics and social network analysis numerous graph representations of the stored data to accurately evaluate the operation of the involved mechanisms as well as the behaviours of network's agents. The promising results of our research provide a clear direction for studying and developing other memory efficient graph models that optimally exploit the benefits of this technology at both operating and analytical level. However, it is left as future research to implement and deploy them in real blockchain applications, which is always the final measure of evaluation.

Concurrently, other innovations are continually being developed in the blockchain field, which may become the solutions to blockchain's most critical challenges. Most notable is the "Lightning Network" [23] payment protocol, which overlays an existing blockchain and tackles cryptocurrencies scaling problems. The incorporation of a graphic model into the operation of such a micropayment system that will monitor fraudulent transactions amongst all channels may eliminate the need for outsourcing trust to 'watchtower' nodes and, thus, expand its limitations.

It is left for future research as well to examine how the proposed reference implementation can be applied in upper level transaction consolidation frameworks such as the Lightning Network and explore whether it can enhance the security features and the analytical methods for path-dependent queries and relationships analytics of transactions in the blocks.

REFERENCES

[1] S. Haber and W. Stornetta, "How to time-stamp a digital document," Journal of Cryptology, vol. 3, no. 2, 1991.
[2] K. F. Buford, H. H. Yu, and E. K. Lua, P2P networking and applications. Amsterdam: Elsevier/Morgan Kaufmann, 2009.
[3] G. Hileman and M. Rauchs, "2017 Global Cryptocurrency Benchmarking Study," SSRN Electronic Journal, 2017.
[4] J. Hendler "Web 3.0 Emerging" Computer vol. 42 no. 1 pp. 111-113 2009.
[5] N. Chowdhury, "Consensus Mechanisms of Blockchain," Inside Blockchain, Bitcoin, and Cryptocurrencies, pp. 49–60, 2019.
[6] S. Nakamoto, ''Bitcoin: A peer-to-peer electronic cash system,'' Tech. Rep., 2008.
[7] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wangm, Y. Wen and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," IEEE Access vol. 7 pp. 22328-22370 2018.
[8] D. Tapscott and A. Tapscott, Blockchain revolution: how the technology behind Bitcoin is changing money, business and the world. UK: Portfolio Penguin, 2018.
[9] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On Scaling Decentralized Blockchains," Financial Cryptography and Data Security Lecture Notes in Computer Science, pp. 106–125, 2016
[10] "Bitcoin Energy Consumption Index," Digiconomist. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption.
[11] S. Kim, Y. Kwon, and S. Cho, ''A survey of scalability solutions on blockchain,'' in Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC), Oct. 2018, pp. 1204–1207.
[12] J. Kwon, ''Tendermint: Consensus without mining,'' Tech. Rep., May 2014.
[13] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, ''Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,'' in Proc. IEEE Symp. Secur. Privacy (SP), May 2016, pp. 839–858.
[14] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, ''Bitcoin-NG: A scalable blockchain protocol,'' in Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2016, pp. 45–59.
[15] M. Swan, Blockchain: Blueprint for a New Economy. Newton, MA, USA: O'Reilly Media, 2015.
[16] S. King and S. Nadal, ''Ppcoin: Peer-to-peer crypto-currency with proof of-stake,'' Tech. Rep., Aug. 2012.
[17] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in P2P systems," Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003), Oct. 2003.
[18] V. Buterin, "On Stake," Ethereum Blog, Jul-2014. [Online]. Available: https://blog.ethereum.org/2014/07/05/stake/?source=post_page.
[19] G. Wood, ''Ethereum: A secure decentralised generalised transaction ledger,'' Ethereum Project Yellow Paper, vol. 151, pp. 1–32, Apr. 2014.
[20] V. Buterin and V. Griffith "Casper the Friendly Finality Gadget," 2017, *arXiv:1710.09437*. [Online]. Available: https://arxiv.org/abs/1710.09437
[21] M. Castro and B. Liskov, ''Practical Byzantine fault tolerance,'' in Proc. OSDI, vol. 99, 1999, pp. 173–186.
[22] V. Buterin, D. Reijsbergen, S. Leonardos, and G. Piliouras, "Incentives in Ethereum's Hybrid Casper Protocol," 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2019
[23] J. Poon and T. Dryja, "The Bitcoin Lightning Network." [Online]. Available: http://lightning.network/lightning-network-paper.pdf.
[24] "IBM Research: Behind the Architecture of Hyperledger Fabric," IBM Research Blog, 08-Feb-2019. [Online]. Available: https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/.
[25] M. Samaniego R. Deters "Blockchain as a service for IoT" 2016 IEEE International Conference on Internet of Things (iThings) pp. 433-436 2016.
[26] "Neo4j Database," Neo4j Graph Database Platform. [Online]. Available: https://neo4j.com/neo4j-graph-database/?ref=home-banner/.
[27] "Neo4j Desktop User Interface Guide," Neo4j Graph Database Platform. [Online]. Available: https://neo4j.com/developer/neo4j-desktop/.
[28] NKB Group, "Ethereum releases Casper v0.1: A short description for validators," Medium, 15-May-2018. [Online]. Available: https://medium.com/@theNKBGroup/ethereum-releases-casper-v0-1-a-short-description-for-validators-3e0a7676d286
[29] V. Buterin, "Immediate message-driven GHOST as FFG fork choice rule," Ethereum Research, 14-Jul-2018. [Online]. Available: https://ethresear.ch/t/immediate-message-driven-ghost-as-ffg-branch-choice-rule/2561.
[30] I. Robinson, J. Webber, and E. Eifrem, Graph Databases: New Opportunities for Connected Data . Sebastopol, CA: OReilly & Associates, 2015.
[31] O. Panzarino, Learning Cypher. Birmingham, United Kingdom: Packt Publishing, 2014.
[32] "What is SAP HANA? An unrivaled data platform for the digital age," *SAP*. [Online]. Available: https://www.sap.com/products/hana.html?infl=32095c59-c617-45d7-a13d-8af08c419145.
[33] "RedisGraph," *Redis Labs*. [Online]. Available: https://redislabs.com/redis-enterprise/redis-graph/.
[34] Neo4j, "openCypher," *openCypher · openCypher*. [Online]. Available: http://www.opencypher.org/.

[35] M. Needham and A. E. Hodler, Graph algorithms: practical examples in Apache Spark and Neo4j. Sebastopol, CA: OReilly Media, 2019.
[36] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," arXiv.org, 07-Jul-2017. [Online]. Available: https://arxiv.org/abs/1707.01873
[37] T. Swanson, "How much electricity is consumed by Bitcoin, Bitcoin Cash, Ethereum, Litecoin, and Monero?," 2018. [Online] Available: https://www.ofnumbers.com/2018/08/26/how-much-electricity-is-consumed-by-bitcoin-bitcoin-cash-ethereum-litecoin-and-monero/
[38] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of Activity," ACM SIGMETRICS Performance Evaluation Review, vol. 42, no. 3, pp. 34–37, Aug. 2014
[39] Ethereum, "First release" GitHub. [Online]. Available: https://github.com/ethereum/casper/releases/tag/v0.1.0
[40] V. Buterin, "Ethereum 2.0 Mauve Paper." [Online]. Available: https://www.win.tue.nl/~mholende/seminar/references/ethereum_mauve.pdf.
[41] V. Buterin, "A CBC Casper Tutorial," Dec-2018. [Online]. Available: https://vitalik.ca/general/2018/12/05/cbc_casper.html.
[42] O. Moindrot and C. Bournhonesque, "Proof of Stake Made Simple with Casper," Stanford University, 2017. [Online]. Available: http://www.scs.stanford.edu/17au-cs244b/labs/projects/moindrot_bournhonesque.pdf.
[43] "Bitcoin to Neo4j," GitHub. [Online]. Available: https://github.com/in3rsha/bitcoin-to-neo4j.
[44] D. Mcginn, D. Mcilwraith, and Y. Guo, "Towards open data blockchain analytics: a Bitcoin perspective," Royal Society Open Science, vol. 5, no. 8, p. 180298, 2018.
[45] M. Bartoletti, S. Lande, L. Pompianu, and A. Bracciali, "A general framework for blockchain analytics," Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers - SERIAL 17, 2017.
[46] H. Kalodner, S. Goldfeder, A. Steven, M. Möser, and A. Narayanan, "BlockSci: Design and applications of a blockchain analysis platform," arXiv.org, 08-Sep-2017. [Online]. Available: https://arxiv.org/abs/1709.02489.
[47] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," Proceedings First International Conference on Peer-to-Peer Computing, Sep. 2001
[48] A B M Moniruzzaman and S. A. Hossain, "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison," arXiv:1307.0191, 30-Jun-2013. [Online]. Available: https://arxiv.org/abs/1307.0191.
[49] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform," github. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Accessed: 14-May-2020].
[50] M. Bartoletti, T. Cimoli, and R. Zunino, "Fun with Bitcoin Smart Contracts," Lecture Notes in Computer Science Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice, pp. 432–449, 2018.
[51] M. Bartoletti and R. Zunino, "BitML," Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018.
[52] "US Secure Hash Algorithms (SHA and HMAC-SHA)," *IETF Tools*. [Online]. Available: https://tools.ietf.org/html/rfc4634. [Accessed: 15-May-2020].
[53] V. Buterin, "Safety Under Dynamic Validator Sets," Medium, 11-Jun-2017. [Online]. Available: https://medium.com/@VitalikButerin/safety-under-dynamic-validator-sets-ef0c3bbdf9f6.
[54] McConaghy, Marques, Muller, De Jonghe, McConaghy, McMullen, Henderson, Bellemare, and Granzotto, "BigchainDB: A Scalable Blockchain Database." [Online]. Available: https://mycourses.aalto.fi/pluginfile.php/378362/mod_resource/content/1/bigchaindb-whitepaper.pdf. [Accessed: 14-May-2020].
[55] R. Hecht and S. Jablonski, "NoSQL evaluation: A use case oriented survey," 2011 International Conference on Cloud and Service Computing, 2011.
[56] C. Weinberger, "Benchmark: MongoDB, PostgreSQL, OrientDB, Neo4j and ArangoDB," ArangoDB, 10-Mar-2020. [Online]. Available: https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/. [Accessed: 14-May-2020].
[57] "Blockchain Size," Blockchain.com. [Online]. Available: https://www.blockchain.com/el/charts/blocks-size. [Accessed: 22-Jan-2020].

KONSTANTINOS TSOULIAS received his diploma from the School of Electrical and Computer Engineering of the National Technical University of Athens (NTUA) in 2019, after completing his thesis on "Developing a consensus mechanism in blockchain trees". His research interests include distributed systems, machine learning, social network analysis and blockchain.

Dr. GEORGIOS PALAIOKRASSAS received his diploma from the Dept. of Electrical and Computer Engineering of the National Technical University of Athens (NTUA) in 2013 and his PhD in 2019 from the same department, where he is currently a postdoctoral researcher and senior Research Associate. He has participated in numerous EU-funded projects and his research interests include social networks, blockchain, machine learning, and IoT.

GEORGIOS FRAGKOS is a PhD candidate and research assistant in the Department of Electrical and Computer Engineering, University of New Mexico. He received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 2018. His main research interests include deep reinforcement learning, game theory, optimization, contract theory, and blockchain.

Dr. ANTONIOS LITKE has more than 18 years of experience as a professional ICT engineer. He has participated in R&D teams of over 15 research projects (EC and nationally funded) and has led technical teams for highly demanding commercial IT projects. Dr. Litke received the diploma from the Dept. of Computer Engineering and Informatics of the University of Patras, Greece in 1999, and the PhD from Electrical and Computer Engineering Department of National Technical University of Athens in 2006 (his thesis had been awarded by Thomaidis Foundation). He is the author of more than 30 scientific articles with over 500 citations and a reviewer of several international journals and conferences. His research interests include parallel and distributed computing, service oriented architectures, blockchains and cybersecurity.

Theodora A. Varvarigou is a professor of computer science at the National Technical University of Athens. Her research interests include parallel algorithms and architectures, fault-tolerant computation, optimization algorithms, and content management. Professor Varvarigou received a PhD in computer science from Stanford University.