DéMars: A Peer-to-Peer Mobile Money System

Abstract

Cryptocurrencies like Bitcoin and Ethereum have enabled decentralized validation of payment transactions without the need for a centralized trusted authority. Bitcoin uses a proof-of-work protocol which results in wastage of electricty and computational power. Proof-of-stake models concentrate the power in few high-value nodes. This has made these systems decentralized rather than truly distributed. Also, being part of these systems often requires a high bandwidth connection and a device with sufficient computational power or stake. To solve these problems, we propose DéMars, a novel blockchain protocol that leverages a hypercube DHT to optimize the amount of data which needs to be stored and transferred. This enables the dynamic creation of *zones*, which are random subsets of the network, but maintain network locality at the same time. The zone for every block is determined dynamically and the consensus group is elected using sortition in the zone. These nodes validate the blockchain using the PBFT algorithm.

1. Introduction

Cryptocurrencies have become popular since Bitcoin [1] was introduced in 2009. They maintain a distributed ledger of transactions which are validated in a decentralized manner without requiring a trusted arbitrator. This is achieved using various consensus mechanisms.

The original proof-of-work mechanism proposed in Bitcoin relies on solving a cryptographic puzzle and leads to high electricity costs and the concentration of mining power in places with cheaper electricity. Alternatives include proof-of-stake algorithms in which the voting power is decided based on the amount staked by an user. These users then reach consensus using algorithms like PBFT [2] or XFT [3]. However, though this reduces the cost of expensive computations, the voting power is concentrated in the hands of users with more money. Moreover, high storage and bandwidth requirements make these systems difficult to realize in areas with poor internet services and expensive hardware. The users in such areas often only have access to mobile phones. With mobile devices, storage is limited and though newer devices are computationally capable, they are limited by their batteries.

To address these challenges, DéMars proposes a blockchain that can run trustlessly and securely with low computing and low data transfer requirements. Nodes are fully autonomous and do not depend on third parties or other nodes for full validation. It runs on a structured peer-to-peer network based on hypercube geometry [4]. Hypercube geometry is based on a concept of measurable node proximity by using Hamming distances [5] between nodes and provides an efficient navigation algorithm to communicate with defined neighbourhoods. This provides DéMars with unbiased network zones at different points in time which are responsible for electing proposers and validators which carry out consensus messaging within the boundaries of the zones.

DéMars deviates majorly from most blockchain solutions by the supposition that all blocks, which maintain all the transactions since the genesis block which in the case of Bitcoin involves a download of over 150GB of data (at the time of writing), are not necessary to validate a blockchain. Several properties of the DéMars blockchain together allow the full validation of transactions even though nodes hold only a partial blockchain:

- Unspent account balance commitment (UABC) snapshots and embedded at checkpoint blocks in the blockchain.
- The PBFT consensus algorithm makes it practically infeasible to fork the blockchain.
- Jump or skip blocks effectively allow a logarithmic forward traversal of the blockchain and the possibility of a linear backwards traversal, made possible by verifiable validator sets.
- Transactions are stored inside buckets in a block. This allows the retrieval of a subset of account balances relevant to current operations. Bucketing is based on the address of the sender and receiver.
- Blocks and buckets are distributed in different parts in the network, reducing the storage overheads.

Details of each of these properties and how the combination of these properties achieve low data transfer rates, low computing requirements while not compromising trust, are covered in the next sections.

The low computing and data transfer requirements of running DéMars means that mobile phones can participate in the network and no tiered node structure is required. A mobile node does not compromise its trust model and operates in the same capacity and independence as any other node on the network unlike most other blockchains where mobile clients have to either depend on 3rd party centralized API wallets or trust full nodes to reduce the data transfer load (SPV wallets). With Bitcoin, less than 10,000 full nodes validate transactions, with the node distribution skewed to a handful of countries. In DéMars, all user nodes can participate in node validation.

2. Background and Motivation

2.1. Cryptocurrencies and Blockchain

Cryptocurrencies are digital assets and can be used as a medium of exchange. The transactions for creation and transfer of assets are secured using cryptography and decentralized control. The decentralized control is realized using a distributed ledger like a blockchain. The first truly decentralized cryptocurrency was introduced in 2009 in the form of Bitcoin [1]. Transactions are validated a group of distrustful participants using different mechanisms like proof-of-work, proof-of-stake, Byzantine algorithms [6], etc.

Blockchain is a chain of blocks wherein each block is cryptographically signed and verified in a decentralized manner. They are tamper-proof and a block cannot be modified without modifying the entire chain.

2.1.1. Bookkeeping in Blockchain

UTXO (unspent transaction output) is a method of bookkeeping used in transactional blockchain like Bitcoin, where every transaction spends output from prior transactions. This transaction, in turn, generates newer outputs for future transactions to process. In order to compute and validate the latest unspent transaction output, all blocks from the genesis block need to be retrieved, and this has high network and storage overheads. The most popular solutions to circumvent this issue are third party wallets and SPV wallets [1]. For third party wallets, the third party runs the full blockchain validation and the client trusts that third party, which compromises the decentralized nature of the blockchain. SPV wallets require clients to trust full nodes behaving correctly and therefore, offer a weak form of validation.

DéMars uses a different form of bookkeeping which is more efficient at optimizing the network and storage requirements, while also providing strong validation, thereby enabling it to run on mobile nodes.

2.1.2. Limitations of Proof-of-Work (PoW) Blockchains

PoW blockchains like Bitcoin and Ethereum [7] reach consensus using the process of mining wherein multiple nodes compete to a solve cryptographic puzzle. This competition results in heavy wastage of computational power and also incentivizes the miners to join mining pools. Most of the hashing power in Bitcoin[1] and Ethereum[2] are concentrated in few mining pools which have more than 50% of the mining power. PoW also requires the compute puzzle to increase in difficulty over time so that it takes significant amount of time. This caps the rate at which blocks can be produced in the blockchain. There is no mining in DéMars.

2.1.3. Limitations of Proof-of-Stake (PoS) Blockchains

There are two typical categories of PoS blockchains: chain-based PoS and BFT-style PoS. Chain-based PoS randomly elects a node to propose a block based on its stake. In BFT-style PoS, a Byzantine fault tolerant [8, 9, 2] algorithm is used to reach consensus on the block to be added. The proposers and validators are chosen based on their stakes in the system. Misbehaviour is usually punished by slashing stakes of the nodes.

In chain-based PoS, we have the so-called nothing-at-stake issue where the proposer can submit multiple blocks to different nodes which results in a weak subjectivity on knowing the correct chain. Ethereum's upcoming PoS blockchain Casper says that a user can know the correct chain by *"[...] asking their friends, block explorers, businesses that they interact with, etc. for a recent block hash in the chain that they see as the canonical one."*

In both the above types of PoS systems, the centralization of stake can compromise the consensus mechanism. For example, in case of Dash[3], a node needs to stake 1000 coins (currently valued at $\approx$ 180 USD each) to become a masternode which can vote on quicker transactions.

[1] `https://www.blockchain.com/pools` Retrieved: Sep 2018

[2] `https://etherscan.io/stat/miner?range=7&blocktype=blocks` Retrieved: Sep 2018

[3] `https://www.dash.org/masternodes/` Retrieved: Sep 2018

In the DéMars blockchain, a malicious user with less than 1/3 of the total number of coins cannot fork the chain.

2.1.4. Algorand

Algorand [10] overcomes the limitations of PoW and PoS listed above by leveraging a sortition function which allows nodes to randomly elect themselves as proposers and validators. It also does not suffer from the centralization of stake as long as a malicious user does not control more than 1/3 of the stake. However, the issue with Algorand is the high bandwidth required because it uses a gossip protocol where all nodes need to receive and relay all messages. The authors calculate the bandwidth usage to be 10 Mbps for 1 MB blocks and 50,000 users. The majority of nodes need to keep synced with the network.

DéMars distinguishes itself from Algorand by optimizing on the high bandwidth requirements by introducing several novel concepts: segmentation of the blockchain network, bucketing of the blockchain, two-way blockchain traversal, fast syncing using skip blocks and UABC blocks. We discuss how all these combine to reduce the amount of data needed for the blockchain to progress in subsequent sections. We also explain how our solution prevents malicious users with less than 1/3 of the total coins in the network to influence voting.

2.1.5. Sharding in Blockchains

A proven approach for scaling databases is *sharding* [11], or partitioning the state which are independently handled in parallel by different subsets of nodes. Multiples approaches to blockchain sharding have been proposed which obtain different levels of trade-offs between performance, security, decentralization and execution overheads [12, 13, 14, 15]. All of the existing techniques, however, forfeit execution overheads in favour of security, performance and/or decentralization. OmniLedger [12] is one of the state-of-the-art techniques which achieves all features except low execution overheads, by relying on secure, multi-party computation protocols for random election of validators. It prevents Sybil attacks [16], but does not safeguard the shards against DDoS attacks. Sharded blockchains achieve high throughput by parallel validation of transactions, but suffer from a proportionate increase in resource utilization. For example, the authors of Algorand [10] calculate the bandwidth usage during consensus to be 10 Mbps for 1 MB blocks and

50,000 users. By parallelizing the consensus among n shards to achieve n times throughput, the bandwidth required increases to $10n$ Mbps.

DéMars aims to achieve low storage and network overheads while maintaining decentralization and security. Hence, we localize the block election to small portions of the network called *zones*. At a given time, only one zone is present which is responsible for electing the current block. By serializing the election process (and not parallelizing it), we are able to achieve the throughput of un-sharded blockchains but at a lower bandwidth. DéMars is orthogonal to all existing approaches and focus on the previously ignored problem of reducing network bandwidth.

2.2. Distributed Hash Table (DHT)

A DHT is a decentralized system with a lookup service like a hash table. The responsibility of maintaining the keys and values is distributed among the nodes in a way that reduces the disruption during churn (addition or removal of nodes). They can scale to large number of nodes and can handle a variety of failures. Such systems were first utilizes by P2P content delivery systems like Napster. Researchers have designed multiple systems with specific features, e.g., CAN [4], Chord [17], Pastry [18], Tapestry [19], etc. These systems emphasize on three major properties: decentralization, scalability and fault tolerance.

Kademlia's XOR distance metric [20] is highly suitable for selecting shards or zones in the network. The network is structured as a binary tree and the first n bits of an address can be used to determine which of the 2^n zones it belongs to. However, this makes the zones fixed and unchangeable. An attacker can take over the majority of nodes in a specific zone and wait for it to be chosen as the validator zone. The problem arises because XOR metric gives more importance to the more significant bits and taking over a few of the bits is easy for an attacker. On the other hand, if the zones were defined based on random bits, the advantage of the DHT topology is lost.

In order to give equal importance to all the bits, we calculate the Hamming weight of the XOR distance and use it as the metric for placing and locating nodes. This essentially structures the network in the form of a hypercube [21, 4]. A simple hypercube of order 4 is shown in Figure 2. What this implies is that nodes in the network with similar bits in their addresses are in the same peer neighbourhood thereby enabling zones to communicate easily. It is possible to find a path to nodes in different neighbourhoods by querying neighbours for nodes closer to the destination segment area.

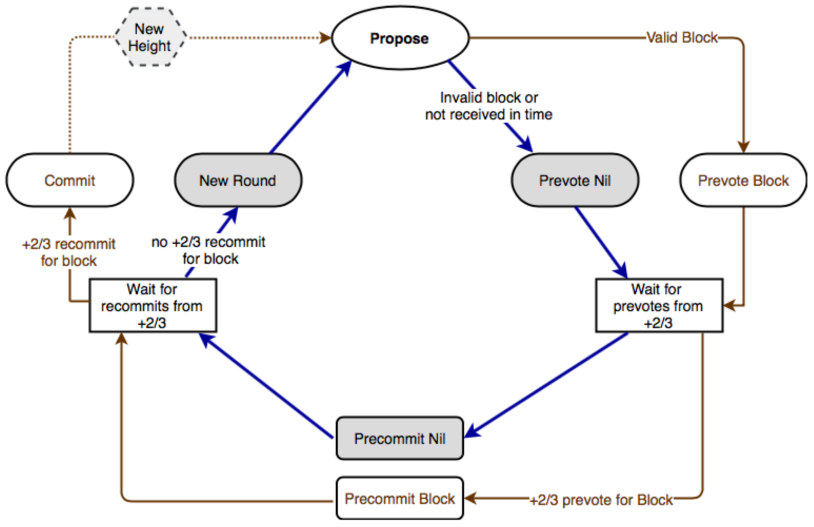

Figure 1: PBFT Consensus Algorithm

2.3. Byzantine Fault Tolerance

The Byzantine Generals' Problem [9, 8] is an agreement problem in distributed systems in the presence of malicious nodes. Pease et al. [9] proved that at least $3f + 1$ nodes are required f faults. In other words, at least 2/3 of the nodes need to be non-malicious.

2.3.1. Practical Byzantine Fault Tolerance (PBFT)

Castro and Liskov [2] presented the first practical solution for weakly synchronous environments. In a system with $3f+1$ nodes, PBFT guarantees safety and liveness if the number of faults is less than f. This was followed by multiple efforts towards optimizing Byzantine replication algorithms [22, 23, 24, 25]. PBFT consists of three steps (illustrated in Figure 1):

1. *Pre-prepare.* The proposer or leader proposes the next record or system.
2. *Prepare.* Every node validates the proposal and sends a prepare message.
3. *Commit.* On observing $2f + 1$ prepare messages, each node sends a commit message. Once $2f + 1$ commit messages are received, the block is committed.

PBFT has two major limitations in the context of blockchain systems. First, it assumes that a fixed set of nodes take part in the consensus process. Hence, these nodes become easy targets of attacks. Second, due to high message overheads, the protocol does not scale beyond few dozens of nodes. This is usually solved by restricting the validation process to a subset of the nodes. The challenge lies in selecting this subset in a fair and safe manner.

3. Network Structure

DéMars is a completely decentralized peer-to-peer network with little distinction between nodes. Mobile (low power, low data usage) nodes are possible and will comprise the bulk of the nodes of this network and guarantee its security by collectively participating in block validation. The network uses a modification of the Kademlia node discovery algorithm in a hypercube network topology. While Kademlia uses the XOR metric to define the distance between nodes, Démars uses Hamming distance [5]. This has been done to provide equal importance to all bits in the node address and thereby prevent attackers from targeting the most significant bits. All nodes in the network are associated to a wallet address, which also functions as the node ID. We also introduce the concept of random network segments or zones, which are defined by their Hamming distance from a specified node.

Messages can be targeted to specific segments but never to specific nodes. Nodes relay messages within their segments (and sometimes target messages to other segments) but whether a node is relaying a message or the originator of a message is not disclosed by the messaging protocol. This way of segmenting the network allows nodes to carry out gossiping within segments for the purposes of sharing blocks or reaching consensus without affecting the whole network, thereby reducing the data transfer costs.

3.1. Hamming Distance and Hypercube Geometry

Hamming distance [5] is a metric for measuring the difference between two strings. Since node IDs are essentially 512-bit binary strings, a similar metric can be be used as a metric for measuring the distance between nodes. Given two 512-bit address a and b, the Hamming distance between them is defined as the number of positions at which the corresponding bits differ. Similar or nearby addresses will have more bits common and have a smaller Hamming distance. DHTs which use Hamming distance as the metric for locating nodes follow the hypercube geometry [4].

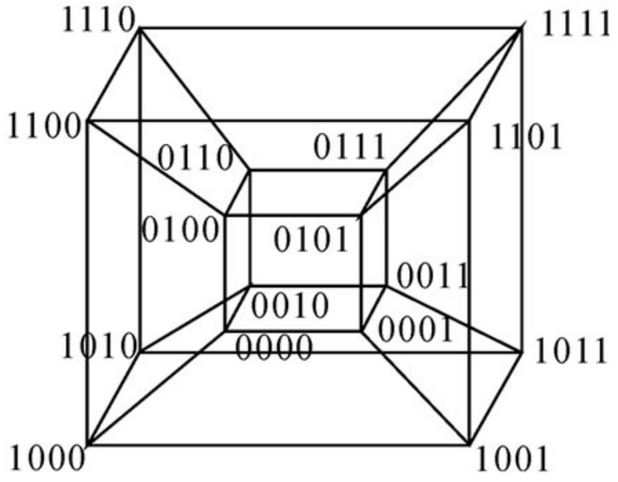

Figure 2: Hypercube of order 4

This means that nodes in the network with more similar bits in their addresses are in the same peer neighbourhood. Nodes in different neighbourhoods can be reached by querying neighbours for nodes closer to the destination address space. The advertised node ID serves not only as identification, but also provides a direct map to the location of information. Nodes joining the network find any node in the network by using DNS resolution to entry nodes - similar to the Bitcoin or other networks. Thereafter the nodes search through the network graph for their neighbourhood, defined by the presence of nodes with low Hamming distance. Once they find the desired area, they register themselves with nodes in that area.

3.2. Random Validator Zones

Consensus mechanisms often involve the entire network and incur high network transfers. A straightforward method of reducing this overhead is to localize the election process to a small subset of nodes. However, for this to be feasible, the following conditions have to be met:

- The honest majority condition of the network has to be maintained in the subset as well.
- The subset of nodes should be able to communicate within themselves efficiently.

In a binomial distribution, for a random sample of size 150, the Byzantine condition of $> 2/3$ honest users in maintained with a probability of 99.999% [26]. However, this is contingent on the distribution of wealth in the network. For example, if one malicious node holds 0.67% of the entire wealth and the sample size is 1% of the network, the malicious node can launch a shard takeover attack. The probabilistic guarantees imply than one in every 100000 blocks can be vulnerable which is not sufficient. These limitations can be overcome in different ways. The sample size can be increased so that the probability becomes infinitesimal. On the other hand, the wealth in the system can be distributed so that no single node can attain majority in a zone.

3.2.1. Distribution Condition

3.2.2. Sortition-based Validator Election

3.3. Types of Nodes

Bitcoin has normal nodes and full nodes. BFT-style systems have proposer and validator nodes. DéMars has all of them and introduces additional types of nodes to facilitate low bandwidth operations.

A useful but not critical concept to the DéMars architecture is allowing some nodes to self mark as high throughput nodes (HT nodes). HT nodes are volunteer, more powerful nodes with better data connectivity, for example desktops on uncapped ADSL connections. DéMars can operate reliably without HT nodes but at the cost of higher data transfer rates for normal mobile nodes. HT nodes act as network catalysts, improving the network robustness in terms of block storage and diversity (different ISPs). HT nodes can reduce the data transfer rates required by mobile nodes by being prioritised over normal nodes when syncing blocks. HT nodes use the same client as all other nodes but with the HT flag turned on. HT nodes do not have higher priority during validator or proposer election and there are no rewards on the blockchain for running HT nodes.

The different types of nodes are listed below:

- *Normal nodes.* The blockchain is designed so that normal nodes are practically users phones. They hold small pieces of the blockchain and can submit transactions to the network. They can also be elected to participate in block validation.

- *HT nodes.* HT or high throughout nodes are more powerful nodes with better connectivity. They help optimize network traffic. They can hold most if not all of the most recent blocks in the blockchain. They also act as normal nodes and are distributed uniformly across network segments but belong in a special overlay network for easy reachability.

- *Full nodes.* These are not required by the network but can optionally be run for historical purposes.
- *Proposer and Validator nodes.* They are elected using network-based sortition from the large number of normal nodes for each new block. They gather transactions from the network and propose a new block. They also act as validators and reach consensus about the new block.
- *Hoster nodes.* All normal nodes hold some recent blocks from the network. HT nodes hold a complete recent block record.

3.4. Other Optimizations

In addition to the infrastructure already discussed, we have tabled the below upgrades and optimizations for future implementation.

3.4.1. Additional security against DDoS attacks

We propose the transitive closure of connected accounts. In the event a proposer initiates multiple transactions at the same time, like in a DDoS attack, the connected blocks will pay a penalty, which increases exponentially. The more aggressive the attack, the greater the fees are and the less incentive there is to attack the network.

3.4.2. Timeouts

An optional system upgrade is a timeout feature to protect the network against attacks which paralyze the creation of new blocks.

4. Blockchain Structure

A blockchain is a sequence of blocks which contain validated transactional data. Normal blockchains consist of genesis and transactional blocks. The DéMars blockchain consists of additional types of blocks which facilitate its objective of reducing the storage and network transfer overheads. The blocks in the DéMars blockchain are stored as a set of buckets. All blocks linearly precede each other in time like any blockchain.

4.1. Bucketing of Blocks

All blocks on the DéMars chain are bucketed, i.e., the information in the block is separated into different buckets. Buckets in blocks correspond one-to-one to segments defined in the network. As with segments, the buckets also change at every UABC block. Buckets can be empty but all buckets must be present in every block. Buckets are linked by precedence similar to how blocks in the chain are linked. Each bucket contains the hash of the corresponding bucket in the previous block. This allows backwards traversal of the blockchain when we have a trusted block.

Buckets are stored on specific subsets of the network. Essentially, not all buckets are stored on all nodes like traditional blockchains. By replicating the buckets just enough to ensure high availability, storage costs are reduced drastically. The replication factor (the number of nodes on which a particular bucket is stored) is low for normal blocks and higher for specialized blocks (discussed shortly).

4.2. Types of Blocks

In addition to the normal genesis and transactional blocks, DéMars defines two additional types of blocks for easy traversal and retrieval on information in the chain. We describe all the different types of blocks here.

4.2.1. Genesis Block

Like all blockchains, we define a fixed genesis block which defines the initial coin distribution. Since a property of our blockchain is that it is infeasible to fork it, connecting a line of blocks to the genesis block is a guarantee for a user to be on the correct chain as long as it is valid.

4.2.2. Transaction Block

Transaction blocks are similar to regular blocks on any blockchain and contain transfers between accounts, signed by the sender. Transactions are bucketed into the buckets corresponding to the sender wallet address as well as the receiver wallet address.

4.2.3. Unspent Account Balance Commitment (UABC) Snapshot Blocks

UABC snapshots at regular intervals allow users to checkpoint account balances. These blocks allow the state of the blockchain to be verified without having to fetch all the blocks from the genesis block. UABC blocks are also bucketed by wallet addresses. These blocks are larger than regular blocks, but since only specific buckets are requested during regular lookups, the network overhead is highly minimal. Also, accounts with zero balances are not stored, resulting in a systemic pruning mechanism.

4.2.4. Jump Blocks or Skip Blocks

Jump blocks are placed at regular intervals of the blockchain. They are similar to skip lists [27] and are present at different granularities. For example, we can have jump blocks at every 1000 and also at every 100 block intervals (1000 and 100 are our granularity levels) to allow more optimized syncs. Jump blocks also contain the following:

1. SHA-256 hashes of all buckets of the previous normal block in the blockchain.
2. The SHA-256 hash of the previous normal block in the blockchain which is used as an indication of the voting segment responsible for the next jump block.
3. A UABC block bucket corresponding to the segment which is nominated to vote on the next jump block.

Multiple granularities of jump blocks work by using the same voting segment for different jump block granularities. As an example, using the granularities above, the jump block at position 1000 determines the voting segment for both the 1100th block and the 2000th block.

The different types of blocks with bucketed transactions are shown in Figure 3. The UABC block N defines the proposer segments for all blocks till the next UABC block $N + N_c$, where N_c is the checkpoint interval and defines the total number of blocks between consecutive UABC blocks. All blocks and buckets are hashed to the previous block or bucket. In addition, jump blocks are also hashed to the previous jump block. The certificate consists of the signatures of the validators. The validator set for a block is a function of the height and segment id. Selection of validators is described in the next section.

A node can sync up quickly to the network by moving forward on the high granularity jump blocks, and then switch to lower granularity jump blocks. It can also move backwards in the blockchain (in most cases only up to the last UABC checkpoint) by verifying hashes embedded in a block. To get a specific account balance or to verify transactions being proposed in a new block, only the relevant bucket chains up to the last UABC checkpoint blocks need to be requested from the network. An example scenario is shown in Figure

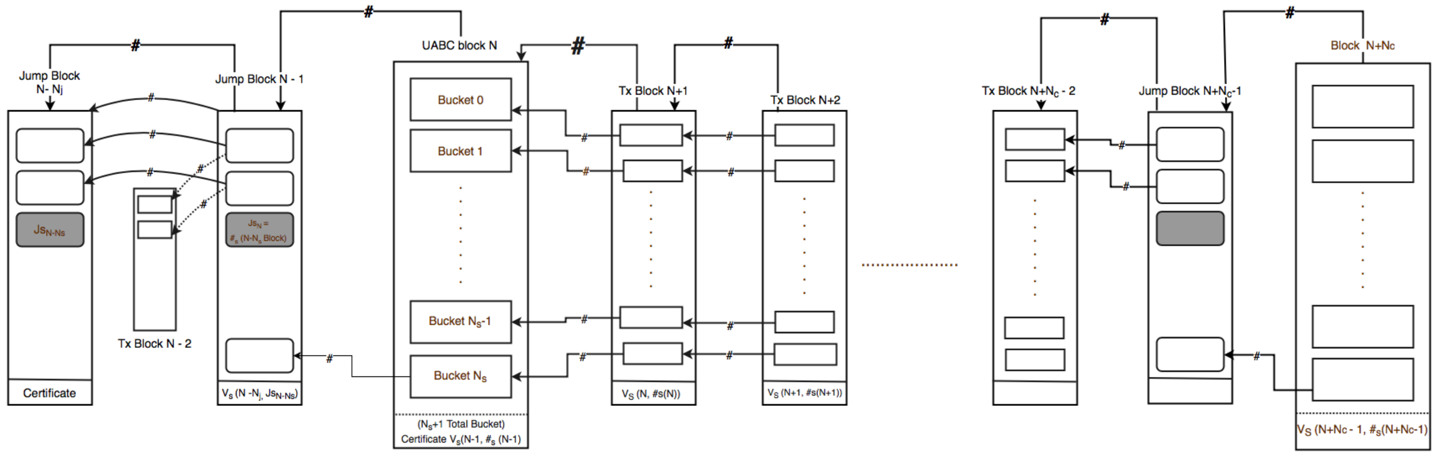

Figure 3: Different Types of Blocks on the Chain

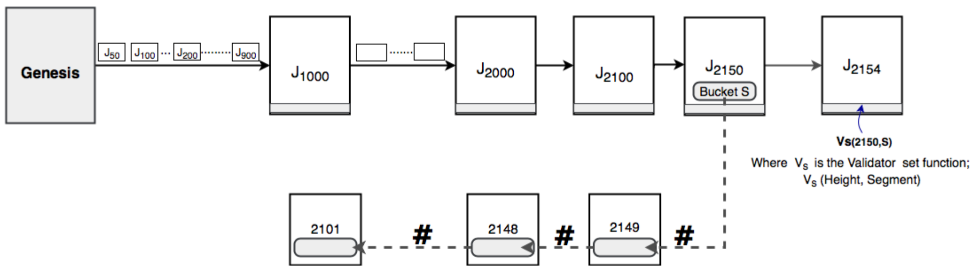

Figure 4: Catching up on the Blockchain

4. In order to validate block 2154, a node first follows the jump blocks at granularity 1000 and reaches block 2000. Then it switches to granularity 100 to reach block 2100 and granularity 50 to reach block 2150 (the latest jump block). Then it moves backwards in the chain to the latest UABC block (block 2101), downloading only the relevant buckets. This traversal is necessary to determine the balances of the accounts in the transactions.

5. Proposal and Validation of Blocks

The DéMars blockchain uses the PBFT algorithm [2] to reach consensus on the next block to be added to the chain. As such, it is able to tolerate Byzantine faults provided at least 2/3 of the total nodes are non-malicious. Since Byzantine algorithms do not scale beyond few dozens of nodes, the consensus protocol is realized among a selected committee of nodes. In order to provide security guarantees, these committees should be random and not deducible in advance. We leverage the concept of segments to randomize the validator selection so that an attacker cannot place his nodes in places which are more likely to be elected.

5.1. Validator and Proposer Selection

The previous block is hashed to give the center point of new proposer segment address. The segment size is defined by choosing a Hamming distance just large enough to ensure a minimum number of eligible addresses (validators) in that segment. The validators are selected based on their account balances in the latest UABC block. In this way a proposer segment is a subset of a network segment but because we are using XOR distances instead of Euclidean distances, these segments are difficult for attackers to identify and prepare in advance without knowing the center point address. By the time the center point address is known, the UABC block is already in place and it is too late for an attacker to influence the proposer segment.

5.1.1. Stake Eligibility Criterion

Nodes are elected to be proposer or validator nodes by firstly meeting an eligibility criterion based on the value of their stakes and then running a sortition algorithm locally to verify whether they have been selected. Nodes that have been selected broadcast consensus messages to the segment which are then propagated within the segment through a bounded gossip protocol.

Our stake based eligibility criterion is as follows:

1. From these N nodes, N_s validator nodes are chosen by sortition.
2. The one with the lowest hash is chosen as the proposer.

The importance of the eligibility criterion is to make it infeasible for an attacker with up to 1/3 of the total coins in the network to influence voting even in a segment of the network.

5.1.2. Sortition Algorithm

Nodes that meet the eligibility criterion each run a sortition algorithm seeded on the previous blocks hash. This allows nodes to determine whether they have been elected as proposers or validators and carry out actions required of their roles. The sortition function is based on verifiable random functions (VRFs) [28] which enable a node to generate a proof using his private key which can be verified by other nodes using the public key. Essentially, this is similar to the sortition in Algorand [10] but bounded in space by limiting the number of nodes which participate in the process, thereby reducing the costs of gossiping the messages.

5.2. Proposal of New Blocks and Checkpoints

Proposers gather transactions and propose a block which validators verify by communicating through a gossip protocol in the new proposer segment. There are two types of proposals: checkpoint proposals, and normal transaction proposals. For transaction proposals, proposer nodes receive transactions and include them inside the block, inside appropriate buckets.

For checkpoint proposals, transactions are generated rather than collected. Proposers gather all the blocks since the previous UABC and use it to compute the next UABC block. The amount of data transferred for a UABC checkpoint proposal and the associated consensus process is higher than a normal transaction proposal. Hence, UABC checkpoints occur less regularly in the blockchain than normal transaction blocks. Also, they are usually queried for specific buckets rather than the complete block hence subsequent operations cost less.

For normal blocks, the gathering process ends when a fixed block size (1MB) or time boundary is reached. Proposer nodes then propose a block to the segment. The validators then employ a Practical Byzantine Fault Tolerant (PBFT) algorithm [2] to reach consensus. The algorithm is not compute intensive, does not require specialized hardware and has low execution costs.

5.3. Validating a New Block

When a new block is proposed, a set of validators are elected through sortition based on the previous block. Validators verify that a proposer meets eligibility criteria and has provided proof of being sortition elected. They then check new transactions to ensure they are signed correctly (public key verification) by the account owners and that the accounts have sufficient balance to carry out the transactions. Balances are retrieved by getting the account balances of all the sender accounts in the previous UABC checkpoint then downloading all the transaction block buckets involving the sender accounts (transactions for both sending or receiving funds).

5.3.1. Consensus Algorithm

DéMars allows distributed voter or validator selection, where voters are randomly selected for each block and are not chosen from different groups.

The PBFT consensus algorithm functions in multiple phases to reach an agreement, as discussed in Section 2.3.1. Due to the message overheads, PBFT does not scale beyond a few dozens of nodes. Hence, we limit the number of validators to 100. PBFT assumes that attacker can control the Byzantine faulty nodes and the network in a coordinated way and ia able to tolerate such attacks. PBFT requires $> 2/3$ of the nodes in the system to be non-malicious.

5.3.2. Validator Security

Validator segments consist of nodes within a fixed Hamming distance from a point determined by the blockchain history (the hash of the previous block for normal blocks). This makes them incalculable in advance, since the block hash is available only after a block is committed. Several features of our blockchain ensures validator anonymity:

- By the time a subsegment based on Hamming distance is determined, an attacker cannot influence the validator set, since the eligibility is based on account balances in the previous UABC block.
- The sortition function prevents anyone from knowing the elected members even though they have the list of all eligible validators in the chosen segment. Validators can provide cryptographic proof that they are validators which can be verified by other nodes, but they cannot be predicted by an attacker.
- Using our DHT-based, semi-structured network gives us the interesting feature of nodes being able to receive messages targeted at their relevant segments or subsegments without having to ever broadcast their real node ID. As part of gossiping, nodes relay unsent messages within segments and no node can distinguish from a node emitting a message versus a node relaying a message. This provides an additional level of node anonymity and makes it hard to target nodes with specific node IDs such as expected validators or proposers.
- We use temporary private keys to sign blocks which are deleted upon signing and cannot be reused.

6. Conclusion

Blockchains provide a way of decentralizing trust by different mechanisms like PoW, PoS, etc. However, these are fraught with limitations and either require the nodes to have high storage and bandwidth or require them to trust third-parties which do. Such constraints make them difficult to realize in regions with expensive network and hardware. To address these limitations, we have proposed DéMars, a novel blockchain solution which leverages randomized network segments using Hamming distances to localize the consensus process and thereby reduce the network bandwidth required. Also, periodic checkpointing removes the need to store the complete blockchain since the genesis block. The network structure also makes the system truly democratic in nature resulting in more transparency.

References

[1] S. Nakamoto, Bitcoin: A Peer-to-peer Electronic Cash System (2008).

[2] M. Castro, B. Liskov, et al., Practical Byzantine Fault Tolerance, in: OSDI, volume 99, USENIX, pp. 173–186.

[3] S. Liu, P. Viotti, C. Cachin, V. Quéma, M. Vukolic, XFT: Practical Fault Tolerance beyond Crashes, in: OSDI, USENIX, pp. 485–500.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A Scalable Content-addressable Network, volume 31, ACM, 2001.

[5] R. W. Hamming, Error Detecting and Error Correcting Codes, Bell System Technical Journal 29 (1950) 147–160.

[6] J. Kwon, Tendermint: Consensus without Mining, Draft v. 0.6, fall (2014).

[7] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, Ethereum Project Yellow Paper 151 (2014) 1–32.

[8] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems (TOPLAS) 4 (1982) 382–401.

[9] M. Pease, R. Shostak, L. Lamport, Reaching Agreement in the Presence of Faults, Journal of the ACM (JACM) 27 (1980) 228–234.

[10] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling Byzantine Agreements for Cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, ACM, pp. 51–68.

[11] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al., Spanner: Google's Globally Distributed Database, ACM Transactions on Computer Systems (TOCS) 31 (2013) 8.

[12] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, OmniLedger: A Secure, Scale-out, Decentralized Ledger via Sharding, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 583–598.

[13] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX, pp. 279–296.

[14] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, A Secure Sharding Protocol for Open Blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, pp. 17–30.

[15] G. Danezis, S. Meiklejohn, Centrally Banked Cryptocurrencies, arXiv preprint arXiv:1505.06895 (2015).

[16] J. R. Douceur, The Sybil Attack, in: International Workshop on Peer-to-peer Systems, Springer, pp. 251–260.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, ACM SIGCOMM Computer Communication Review 31 (2001) 149–160.

[18] A. Rowstron, P. Druschel, Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-peer Systems, in: IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, pp. 329–350.

[19] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. D. Kubiatowicz, Tapestry: A Resilient Global-scale Overlay for Service Deployment, IEEE Journal on Selected Areas in Communications 22 (2004) 41–53.

[20] P. Maymounkov, D. Mazieres, Kademlia: A Peer-to-peer Information System based on the XOR Metric, in: International Workshop on Peer-to-Peer Systems, Springer, pp. 53–65.

[21] L. N. Bhuyan, D. P. Agrawal, Generalized Hypercube and Hyperbus Structures for a Computer Network, IEEE Transactions on Computers (1984) 323–333.

[22] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, J. J. Wylie, Fault-scalable Byzantine Fault-tolerant Services, ACM SIGOPS Operating Systems Review 39 (2005) 59–74.

[23] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, L. Shrira, HQ Replication: A Hybrid Quorum Protocol for Byzantine Fault Tolerance, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, USENIX, pp. 177–190.

[24] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva: Speculative Byzantine Fault Tolerance, ACM Transactions on Computer Systems (TOCS) 27 (2009) 7.

[25] R. Guerraoui, N. Knežević, V. Quéma, M. Vukolić, The Next 700 BFT Protocols, in: Proceedings of the 5th European Conference on Computer Systems, ACM, pp. 363–376.

[26] V. Buterin, On Sharding Blockchains, [Online]. Available: `https://github.com/ethereum/wiki/wiki/Sharding-FAQs`, 2018.

[27] W. Pugh, Skip Lists: A Probabilistic Alternative to Balanced Trees, Communications of the ACM 33 (1990) 668–676.

[28] S. Micali, M. Rabin, S. Vadhan, Verifiable Random Functions, in: 40th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, pp. 120–130.