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 resources. Similarly, 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 computational power. To solve these problems, we propose DéMars, a novel blockchain protocol that leverages a structured DHT to optimize the amount of data stored and transferred. This allows the network to be divided into segments, which can be navigated using the DHT protocol. The consensus group for a given block is elected from among the neighbours of a randomly selected node, which is elected from different segments in a Round-robin fashion, a process called *network-based sortition*. These nodes validate the blockchain using BFT algorithms.

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. Finally, the high storage and bandwidth requirements make these systems difficult to realize in areas with poor internet services and expensive infrastructure. 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 Kademlia. Kademlia is based on a concept of measurable node proximity by using XOR distances between nodes and provides an efficient navigation algorithm to communicate with defined neighbourhoods. This provides DMars with unambiguous network segments responsible at different points in time for electing their proposers and validators which carry out consensus messaging within the boundaries of these segments.

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 are combined allow to fully validate transactions even though nodes hold only a partial blockchain:

- Embedding unspent transaction output (UTXO) snapshot at checkpoint blocks in the blockchain.
- Making it practically infeasible to fork the blockchain through the PBFT consensus algorithm.
- The implementation of jump blocks effectively allowing a logarithmic traversal of the blockchain forward and the possibility to traverse the blockchain backwards linearly, an idea that is made possible by verifiable validator sets.
- A bucketing of block transactions that allows retrieving a subset of account balances relevant to operations . Bucketing is done by network address segments for reasons further explained below.
- Blocks and buckets being distributed in different predictable segments in the network.

Details of each of these properties and how the combination of these properties achieve low data transfer rates, low computing requirements while not compromising trustlessness 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 with most other blockchains where mobile clients have to either depend on 3rd party centralized API wallets or by trust full nodes to reduce the data transfer load (SPV wallets). With Bitcoin, less than 10,000 full nodes validate transactions, with node distribution skewed to a handful of countries. In DéMars, all user nodes 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, 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. Unspent Transaction Output (UTXO)

UTXO is a method of bookkeeping used in Bitcoin where every transaction spends outputs from prior transactions. This transaction, in turn, generates newer outputs for future transactions to process. Essentially, this stores the account balances of all wallets in a database which can be used to validate transactions. By maintaining UTXO sets, nodes joining the network do not need to download all the blocks from the genesis block. This also removes the need to run full nodes and thereby the bandwidth requirements.

2.1.2. Limitations of Existing Systems

The transactions are replicated across all nodes (or at least all full nodes) which highly increases the storage costs. Running a full node in Bitcoin requires hundreds of GBs of storage. Also, the messages are broadcast across the network thereby increasing the bandwidth costs as well. Finally, they are biased towards nodes with high computational power (proof-of-work systems) or high stake (proof-of-stake systems).

2.2. Byzantine Fault Tolerance

The Byzantine Generals' Problem [4, 5] is an agreement problem in distributed systems in the presence of malicious nodes. Pease et al. [4] 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.

2.2.1. 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 [6, 7, 8, 9]. PBFT consists on three steps:

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.

2.3. Distributed Hash Table (DHT)

A DHT is a decentralized system which a loopkup 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 [10], Chord [11], Pastry [12], Tapestry [13], etc. These systems emphasize on three major properties: decentralization, scalability and fault tolerance.

2.3.1. Kademlia

Kademlia [14] is a DHT for P2P systems, where each node is identified by its node ID and is part of an overlay network. Kademlia uses a XOR distance metric for routing queries and locating nodes in the network. Given two node identifiers, x and y, the XOR distance between them is defined as their bitwise exclusive or (XOR), $d(x, y) = x \oplus y$. Though non-Euclidean in nature, the XOR metric follows the symmetry $(d(x, y) = d(y, x))$ and triangle $(d(x, y) + d(y, z) \geq d(x, z))$ properties.

The routing table consists of lists of nodes at different XOR distances, called k-buckets. These entries are used to recursively lookup entries in the DHT.

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 slight modification of the Kademlia node discovery algorithm. All nodes in the network are associated to a wallet address. The wallet address is the node ID. We also define a fixed set of network segments are defined by their XOR distance from 0. Segments are defined by the first Bsegment bits of any node ID.

One modification from Kademlia is that nodes never disclose their actual node ID but register on the network with a random address (advertised node ID) within the network segment they are found at. 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 protocol messaging. 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, to reduce overhead data transfer costs.

Running on a Kademlia network means that nodes in the network with similar segment areas are in the same peer neighbourhood. It is therefore possible to find a path to nodes in different neighbourhoods by querying neighbours for nodes closer to the destination segment area. 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 - as per the Bitcoin network. Thereafter the nodes search through the network graph for their segment area. Once they find the segment area, they register themselves with nodes in that segment area.

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 a 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.

3.1. Types of Nodes

- *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.

4. Blockchain

Our blockchain consists of different types of blocks, linearly preceding each other in time.

4.1. 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 as long as it is valid.

4.2. Block Bucketing

All blocks on the DéMars chain are bucketed, that is information in the block is separated into different buckets. Buckets in blocks correspond one-to-one to segments defined in the network. Buckets can be empty but all buckets must be present in every block.

Buckets are linked by precedence in the same way blocks in the chain are linked by precedence. 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.

4.3. Transaction Block

Transaction blocks contain transfers between accounts, signed by the sender. Transactions are bucketed into both the bucket corresponding to the sender wallet address and the receiver wallet address.

4.4. Unspent Transaction Output (UTXO) Snapshot Blocks

UTXO snapshots at regular intervals allow users to checkpoint account balances. UTXO blocks are also bucketed by wallet addresses. Accounts with zero balances are not stored in the UTXO checkpoints resulting in a systemic pruning mechanism.

4.5. Jump Blocks

Jump blocks are placed at regular intervals of the blockchain but can also be 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 optimised syncs. Jump blocks also contain the following:

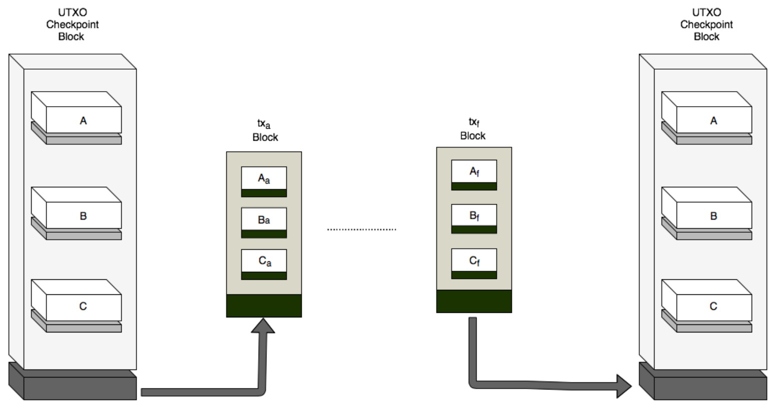

Figure 1: UTXO Commitments

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 UTXO 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 block determines the voting segment for both the 1100th block and the 2000th block. The 1900th block does not contain a UTXO block bucket and does not point to further jump blocks.

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 UTXO 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 UTXO checkpoint blocks need to be requested from the network.

5. Proposal and Validation of Blocks

5.1. Proposer Segments

The previous block is hashed to give the center point of new proposer segment address. The segment size is defined by choosing the lowest XOR distance factor to ensure a minimum of eligible addresses in that segment. 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 of knowing the center point address. By the time the center point address is known, it is too late for an attacker to influence the proposer segment.

5.2. 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. Only the top N nodes in order of stake are eligible for validation.
2. N_s nodes are chosen by sortition.

The importance of the eligibility criterion is to make it infeasible for an attacker with 50% of the total coins in the network to influence voting even in a segment of the network.

5.3. 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.

5.4. Proposal of New Blocks and Checkpoints

Proposers gather transactions and propose a block which validators will 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 validate them through a BFT consensus algorithm.

For checkpoint proposals, transactions are gathered similarly to normal transaction blocks. Proposers have to get all the blocks including the previous UTXO which is used to compute the next UTXO block. The amount of data transferred for a UTXO checkpoint proposal and consensus is higher than a normal transaction proposal which is why UTXO checkpoints occur less regularly in the blockchain than normal transaction 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 and a Practical Byzantine Fault Tolerant (PBFT) algorithm [2] is used to reach consensus. The algorithm is not compute intensive and therefore consensus does not require specialised hardware and expensive running costs.

5.5. Validating a New Block

When a new block is proposed, a set of validators are elected through sortition based on the proposed block. Validators verify that a proposer meets eligibility criteria and has provided proof of being sortition elected. They then check (public key verification) new transactions to ensure they are signed correctly 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 UTXO checkpoint then downloading all the transaction block buckets involving the sender accounts (transactions for both sending or receiving funds).

5.6. Consensus Algorithm

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

The consensus algorithm functions in multiple phases to reach an agreement. Due to the message overheads, they do not scale beyond a few dozens of nodes. As a result, some nodes are elected or delegated to take part in the consensus process. PBFT assumes that attacker can control the Byzantine faulty nodes and the network in a coordinated way and incurs high overheads in order to protect from such a strong attacker. PBFT requires $> 2/3$ of the nodes in the system to be non-malicious.

5.7. Validator Security

Validator segments will consist of nodes at a fixed XOR distance from a point determined by the blockchain history (the hash of the previous block for normal blocks).

Several features of our blockchain ensure validator anonymity:

- By the time a subsegment based on XOR distance is determined, an attacker cannot influence the validator set.
- The sortition function prevents anyone from knowing who 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 but they cannot be predicted by an attacker.
- Using our Kademlia based 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 a level of node anonymity and makes it hard to target nodes with specific node IDs such as expected validators or proposers.
- The use of temporary private keys to sign blocks which are deleted upon signing.

5.8. Optimizations

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

5.8.1. 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.

5.8.2. Timeouts

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

5.8.3. Voting Options

We are considering offering each node the ability to vote on proposed upgrades and on various community proposals.

6. Conclusion

DéMars enables blockchain transactions of bandwidth contrained mobile networks using a novel segmentation mechanism using Kademlia XOR spaces. The network structure also makes the system truly democratic in nature resulting in more transparency. The future is now.

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

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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.