Design and analysis of a decentralized relay network*

Marlin Labs†

January 30, 2018

Networks form the backbone of all peer-to-peer systems as hosts communicate with one another via the Internet. Enterprises have long recognized the importance of networks in improving user experience and productivity. As a result, a variety of techniques including the use of private lines, CDNs and SD-WANs have been developed to provide better networking. Blockchains, however, rely on the public internet which is generally modeled as an asynchronous, best effort communication system, in which guarantees of reliable communication cannot be established. Since decentralized systems can't take advantage of the techniques mentioned earlier due to their centralized nature, they accept the performance compromise imposed by the public Internet as a given.

In addition, block and transaction propagation in decentralized networks today is incentive incompatible as nodes that become aware of the information compete for the same payoffs. Babaioff et al.[1] were the first to point this out for transaction propagation in Bitcoin. Using the success of the MIT team in the DARPA Network Challenge as an example, they showed that once mining rewards diminish, without additional incentives, it would be in the best interest of miners to keep transactions they learn about private with the hope to eventually mine and claim the fees originating from them. Abraham et al.[2] pointed out that when a miner shared its mined block with its neighbour, the neighbour had an incentive to not propagate the block further and take advantage of the fact that it was amongst the first few miners who had learned about the block and get a headstart in finding the next block[1].

In this work, we present a secure way to enable decentralized systems to benefit from high-performance relay networks without compromizing decentralization. We illustrate its incentive compatibility and resistance to multiple attack vectors. Its benefits include higher throughput for blockchain platforms, lower fees, faster finality and better performing decentralized cloud services. In addition, several other use cases like its exploitation in decentralized exchanges are not beyond imagination.

*This specification is under active development and will be regularly updated.

†Web: `www.marlin.pro`, Email: `info@marlin.pro`

[1]Note that this issue is different from selfish mining where the miner himself has an incentive to keep the block private.

Contents

1 Motivation

Networking forms the backbone of any distributed application. We take a look at the current state of networking in the blockchain space and detail a few improvements that a high-performance incentivized networking layer can bring.

1.1 Networking as a scalability bottleneck

While various off-chain methods have been developed to scale blockchains, on-chain scaling is still important for security reasons. Off-chain scaling solutions usually rely on on-chain dispute resolution mechanisms. This means that the dispute resolution mechanism is limited by on-chain throughput - it can be saturated with disputes thereby necessarily letting some disputes go unresolved in favour of an attacker. The higher the throughput of the base chain, the tougher and costlier it is to perform a successful attack.

Moreover, any scaling that we can bring on-chain compounds with other off-chain methods giving a much higher net throughput increase.

Throughput. Throughput in the context of blockchains is the number of transactions per second a blockchain can support (referred to as tps)[2].

$$Throughput = \#Transactions/Block \times \#Blocks/second$$

We essentially have two pathways to scale on-chain - increase the block size or increase the block rate.

1.1.1 Block rate increase

Block rate. In Bitcoin, this follows the exponential distribution where the probability of a block not being found for T minutes or longer is

$$P(T) = e^{\frac{-T}{t_b}}$$

The number of blocks expected to be found in a certain time follows the Poisson distribution. Thus, the probability of finding n blocks in T minutes is given by

$$P(n,T) = \frac{(\frac{T}{t_b})^n * e^{\frac{-T}{t_b}}}{n!} \tag{1}$$

Propagation time. It is the time taken for a newly mined block to reach other nodes in the network (usually represented for high percentiles like 95%).

Forks. Blockchain forks occur when two different but equally valid blocks (also called uncle/orphan blocks) are produced at the same height. This lead to one of the blocks being eventually discarded in favour of the other one. A good consensus mechanism incentivizes new blocks to be produced on top of the most recent block. However, in distributed systems, the "most recent" block may not be the same across the network leading to a fork.

Forks are bad for a blockchain because of two main reasons:

1. Wasted effort - Any effort which has gone into a discarded block is wasted and could have been productively used elsewhere.
2. Transaction repudiation - Any transaction included in the discarded block is now effectively rolled back even though it was seen in the blockchain by some participants. This is the reason why we need to wait for n confirmations - to reduce the chances that the block with our transaction is eventually discarded.

[2]Different systems have different concepts as a basis. For example, in Ethereum, it doesn't make sense to talk about transactions, rather, gas is a more fundamental quantity. For demonstration and simplicity, we use transactions in the analysis.

Relationship between forks and block time. Using Eq (1) above, we get

$$\begin{aligned} P(fork) &= P(1, t_p) + P(2, t_p) + P(3, t_p) + ... \\ &= \sum_{n=1}^{n=\text{inf}} \frac{(\frac{t_p}{t_b})^n * e^{\frac{-T}{t_b}}}{n!} \\ &= 1 - \frac{1}{e^{\frac{t_p}{t_b}}}. \end{aligned} \tag{2}$$

$1 - e^{-x}$ is a monotonically increasing function for $x > 0$. Thus, the probability of forks increases as t_b is decreased without correspondingly decreasing the propagation time $T = t_p$. The propagation time is a bottleneck toward higher block rates.

1.1.2 Block size increase

Decker et al.[3] show that a block size increase leads to higher propagation time. As a spam/DDoS prevention measure, nodes need to verify any data that they receive before propagating it further. With an increase in block size, the amount of data that the nodes need to verify increases leading to higher propagation time.

Using the same analysis as in the previous section, we assert that networking (specifically, propagation time) is a bottleneck to increasing the block size as well. In addition, an incentivized networking layer can prevent spam/DDoS by introducing penalties thereby enabling additional optimizations in the form of cut-through propagation in place of the current store-and-forward propagation.

1.1.3 Examples

Following is a brief description of some systems that could benefit by leveraging Marlin:

Tendermint. Tendermint's Byzantine Consensus Algorithm relies on receiving at least 2/3 votes from fellow validators in its pre-vote and pre-commit stages. These stages are governed by time outs (assuming partial synchronicity) since in an unreliable network its hard to know whether messages were lost or are delayed. A low-latency network like Marlin could allow systems like Tendermint to decrease the value of the timeout parameter by a tenth and also decrease the number of rounds required before blocks get committed by decreasing round failures due to network losses.

$$\begin{aligned} \text{Throughput} &\propto \#\text{blocks/second} \\ &\propto \frac{1}{\text{avg time required for committing block}} \\ &\propto \frac{1}{\text{number of rounds per block}(n) * \text{time per round}} \\ &\propto \frac{1}{n * \text{timeout limit for pre-vote and pre-commit}} \end{aligned}$$

Thunder Protocol The fast-path protocol uses an aggregate signature scheme[3] to receive signatures from at least 3/4 of the committee members. However, the fallback is a slow chain like Ethereum. There's no estimate of how often the fast path might have to fallback to the slow chain and thus can benefit from a faster slow chain.

Algorand In the Byzantine Agreement consensus algorithm used by Algorand, block propagation is parallelized with collection of soft-votes by propagating block hashes (which are much smaller than the block themselves) separately. As a result by the time the block reaches every node in the network, every honest node receives 2f+1 soft-votes with high probability. However, nodes have to wait to propagate the cert-votes till they don't receive the block. Algorand like Bitcoin uses gossip to propagate blocks with verification carried out at every node to prevent spam. Increasing the

[3]Schnorr multisig requires 3 rounds with the discrete-log assumption.

speed of block propagation can thus decrease the duration of every period for every honest node hence improving the system's throughput.

Bitcoin/Ethereum Liveness and consistency in the Nakamoto protocol is subject to the mining hardness being a function of the network's maximum latency l which requires the block-time to be significantly more than l. As shown in Eq. (2), to prevent forks t_p has to be decreased. However, the Bitcoin and Ethereum network today use gossip to propagate blocks over public networks where the block is verified at each hop increasing the network propagation latency. Reducing t_p using a faster relay can lead to a proportional decrease in the value of t_b.

Dfinity Dfinity's consensus algorithm proceeds in rounds where each round involves: beacon nodes broadcasting the source of randomness (VRF); blockchain layer nodes building and broadcasting a block proposal from validated transactions via the Probabilistic Slot Protocol driven by the random beacon output; notary nodes listening to block proposals, signing/notarizing the highest-ranked block proposal and broadcasting it back to the network. The Marlin network can help in bringing down the broadcast saturation times for each of the above steps resulting in faster block times.

1.2 Networking as a decentralization bottleneck

We look at transaction propagation in the absence of a properly incentivized networking layer.

Tragedy of the commons. Garrett Hardin explained the tragedy of the commons in his essay[4] in the following way: "Picture a pasture open to all. It is expected that each herdsman will try to keep as many cattle as possible on this commons. ... Therein is the tragedy. Each man is locked into a system that compels him to increase his herd without limit, in a world that is limited. Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons."

Relayer bandwidth is a common resource. Computation, storage and bandwidth are common resources used in blockchains. While computation has an incentiveized marketplace today, none exist for bandwidth and storage[4]. As a result, blockchains today rely on the altruism of miners and full nodes to relay blocks and transactions.

In the case of blocks, miners have an incentive to relay their blocks and receive the latest blocks in order to prevent the blocks they mine from being orphaned[5]. However, they have little incentive to proactively propagate blocks mined by the other miners. They are in fact incentivised to withhold the block and propagate it only after they have mined a block on top of it. This not only saves them from bandwidth costs but also gives them an edge with respect to the other miners.

Free-rider problem. A market failure that occurs when individuals prefer to let others make the investment and yet have access to the benefits of the investment.

Transaction propagation is not incentivized. Transaction propagation is susceptible to the free-rider problem where nodes would like to save bandwidth costs by not propagating transactions and rely on others to do it.

Analysis. Consider a set of n full nodes. Each node i chooses a strategy of forwarding transactions or not, represented by $s_i \in S_i = \{0, 1\}$. Let $S = \prod_i S_i$ be the set of all possible strategy profiles. Each node has a payoff function $f_i(s)$ for every strategy profile $s \in S$. Let $f(s) = (f_1(s), f_2(s), \ldots, f_n(s))$. Given a game (S, f) as defined among n full-nodes, we attempt to analyze the Nash equilibrium, the condition under which it exists and its implications.

Let s_{-i} be the strategy profile of all players except player i, i.e. $s = (s_i, s_{-i})$. Let the utility that node gets out of the blockchain platform be represented by $u_i(s)$. A node choosing to forward transactions can only bring more utility to the blockchain platform, i.e.

$$u_i(0, s_{-i}) \leq u_i(1, s_{-i})$$

[4]Two competing proposals for storage rent exist and are summarized at https://www.reddit.com/r/ethereum/comments/ab1n2e/eli5_storage_rent

[5]ignoring selfish mining

Let the cost that a node incurs if it chooses to forward transactions be c. Assuming no incentive to forward transactions, the payoff for nodes can be written as:

$$f_i(s) = \begin{cases} u_i(0, s_{-i}) & \text{if } s_i = 0 \\ u_i(1, s_{-i}) - c & \text{if } s_i = 1 \end{cases}$$

A Nash equilibrium is said to exist at a strategy profile if the player gains nothing by changing his strategy unilaterally while other players stick to their existing strategy. Formally, for a strategy profile s^* to be a Nash equilibrium, $\forall i$ and $\forall s_i \in S_i$,

$$f_i(s_i, s^*_{-i}) \leq f_i(s^*_i, s^*_{-i})$$

If we want it to be in the rational interest of node i to forward transactions, there must exist a Nash equilibrium at $s_i = 1$. In particular,

$$\begin{aligned} & f_i(0, s^*_{-i}) \leq f_i(1, s^*_{-i}) \\ \implies & u_i(0, s^*_{-i}) \leq u_i(1, s^*_{-i}) - c \\ \implies & c \leq u_i(1, s^*_{-i}) - u_i(0, s^*_{-i}) \\ \implies & c \leq \Delta u_{ii} \end{aligned}$$

where $\Delta u_{ij} = u_i(1, s^*_{-j}) - u_i(0, s^*_{-j})$ represents the utility difference to node i as a result of node j changing its strategy unilaterally.

Assuming the nodes are fungible, the utility that nodes derive depends only on the total number of nodes forwarding transactions, i.e.

$$\begin{aligned} & u_i(s) = u_i(s') \text{ if } |s| = |s'| \\ \implies & \Delta u_{ii} = \Delta u_{ij}, \forall j \neq i \end{aligned}$$

Nodes do not gain anything directly from choosing to forward transactions. Any gains come indirectly from lower transaction confirmation times and capturing a part of the resulting ecosystem value growth.[6].

The ability to capture ecosystem value(which is a proxy for Δu_{ii}) usually follows a power law distribution in real-world systems. This is all the more true in the cryptocurrency space where holding tokens is a good(and in many cases, primary) way of capturing ecosystem growth and the power law in token wealth is even more pronounced.

This means that if a node j has the ability to capture more of the value than node i,

$$\begin{aligned} & \Delta u_{ii} < \Delta u_{ji} \\ \implies & \Delta u_{ii} < \Delta u_{jj} \qquad \text{(since } \Delta u_{jj} = \Delta u_{ji}\text{)} \end{aligned}$$

The implication is that if it is rational for node i to be forwarding transactions, it is even more rational for node j to be forwarding transactions. Node i therefore prefers to let node j forward transactions instead of itself. If node j is already forwarding transactions, it is still rational for node i to let the entity controlling node j spin up another node k to forward transactions instead of itself forwarding transactions.

This invariably leads to centralization. The biggest entities(which as per power law distribution would be small in number) will be the only ones for whom it is rational to forward transactions.

1.3 Networking as a defense mechanism

A high-performance networking layer acts as a good defense against some attacks by making it much more diffucult expensive to execute for an attacker.

[6]E.g. Bitmain is incentivized to run Bitcoin nodes so that the Bitcoin ecosystem can grow and they can continue to sell ASICs and an Ethereum holder is incentivised to run Ethereum nodes so that Ethereum's price can increase and his holdings become more valuable.

1.3.1 Selfish mining

Attack description. A selfish miner as described by Eyal et al[5] does not broadcast a block it has mined and continues building on top of it privately until it finds a competing block being propagated by honest miners. The scenario is parameterized by two variables - α, the percentage of hash power controlled by the selfish miner and the γ, the percentage of honest miners that mine on top of the block released by the selfish miner after observing that an honest miner has released a block that competes with its withheld block.

Eyal et al deduce that for a well connected selfish miner with possibly several sybil non-mining nodes across the world γ would nearly equal 1 as it could propagate its withheld block to the other honest miners as soon as a node in its sybil-network received a block produced by a honest miner. In such a scenario the blockchain system is not incentive compatible even when α is just greater than 0. On the other hand, if γ is 0, a selfish mining pool requires control over atleast $1/3$ of the total hashing power in order to carry out an attack where it receives a revenue greater than its relative size.

Countermeasure. Currently when there are two branches of the same height, the block that an honest miner builds on top of is the one it received first. As a result latency and topology play a decisive factor which can be exploited by a well-connected miner with sybils spread across the globe. While the sybil nodes slow down the already slow gossip propagation of the block produced by honest miners, caching of the block produced by the selfish miner ensures that it can be swiftly delivered to honest miners across geographies. Marlin's low-latency relay helps bring γ close to 0 by not relying on gossip to broadcast blocks through the network. A low-latency global relay ensures that the honest miner's block reaches different geographies before a sybil node aiding the selfish miner can signal its private network to release the withheld block.

1.3.2 Race attack and Finney attack

Attack description. An attacker can send a transaction to only the merchant while create a conflicting transaction that sends the amount to his own account and send it to the network. It is more likely that the latter transaction gets included in the next block thus reversing the merchants transaction. This attack only makes sense when the merchant accepts 0-confirmation transactions. In a Finney attack, the attacker himself constructs a block having the conflicting transaction and propagates it quickly across the network as soon as the merchant accepts its original transaction[7].

Vector76 attack. A Vector76 attack is like the Finney attack described above except that the merchant waits for at least one-confirmation. In order to satisfy this requirement, the attacker sends his privately mined block containing the transaction paying the merchant only to the merchant as soon as he observes a block being propagated that contains it's conflicting transaction (to prevent the merchant from propagating the block containing the legitimate transaction) and quickly tries to retrieve goods of equal value from the merchant based on the 1-confirmation.

Like before, this attack is especially easy when the merchant is not very well connected. Quoting Gavin Andreesen - "If mybitcoin was running bitcoin behind Tor, and had just one connection (through a Tor exit node) to the rest of the bitcoin network, then they'd be particularly susceptible to this 1-confirmation attack."[8]

Countermeasure. The usual precaution to deter such attacks is for the miner to avoid accepting 0-confirmation transactions or to disable incoming connections, choose specific outgoing connections[9] and only connect to well connected nodes. A relay network like Marlin is a collection of a large set of well-connected nodes. Given that most miners use it to receive and send transactions and blocks, a transaction that wishes to make its way to block would be broadcast via the Marlin network ensuring that merchants have a high chance of seeing the conflicting transaction. Moreover, merchants could forbid direct communication and only accept blocks and transactions

[7]These attacks have been discussed with respect to Dice sites at several forums, for example, https://bitcointalk.org/index.php?topic=145510.0 and https://bitcointalk.org/index.php?topic=327767.0

[8]https://bitcointalk.org/index.php?topic=36788.msg463423#msg463423

[9]https://bitcointalk.org/index.php?topic=79090.msg881283#msg881283

coming through the Marlin network ensuring that they do not have a different view as compared to the mining community.

2 The Marlin Network

2.1 Stakeholders

Nodes associated with the Marlin Protocol are classified under the following roles:

1. Marlin nodes (shortened to node for brevity) - Nodes that serve as relay nodes in the Marlin network.
2. Auditors - Nodes that ensure that the relay nodes or relay networks (defined below) are abiding by their performance and SLA guarantees.
3. Miners - Block producers in blockchain platforms that use the Marlin relay network to propagate packets and blocks.
4. Full nodes - Users(wallets, merchants, etc) that use the Marlin network to receive the latest transactions and blocks from the blockchain platforms they are interested in.

These roles are not exclusive. A miner or full node can also act as a Marlin node.

Relay network. A set of Marlin nodes that cooperate with one another to propagate packets amongst themselves and deliver them to miners or full nodes as per the protocol of the users using its services with pre-specified SLAs in advertised geographies.

The Marlin network is a marketplace of such relay networks where multiple relay networks compete to carry blocks for the same blockchain.

2.2 Architecture

The working of the Marlin network can be broken down into four parts: creation, management, packet propagation and monitoring.

Once a relay network is created using a set of available Marlin nodes, it is important that the network be smoothly managed so that poorly performing individual nodes do not harm the network as a whole. Miners are subscribed to all relay networks serving the blockchain they mine on and randomly pick a few relay networks to publish blocks to. These randomly chosen relay networks are expected to notify and deliver blocks to all subscribers with pre-negotiated performance guarantees. A monitoring mechanism in place ensures that defaulting relay networks are penalized.

The Marlin network has a native token LIN. LIN is required by Marlin nodes to be a part of a relay network which entitles them to fees collected by the network. The value of LIN required to be staked is not only a function of their expected revenue but also a reflection of the cost to its users in case it fails. Stakes are slashed in case the relay network fails to deliver blocks or transactions within certain time limits.

Whether or not a relay network fails to meet its guarantees is monitored by a decentralized network of auditors. A Schelling scheme[6] is used to incentivize the auditors to report results correctly. Our goal is to keep the auditing scheme independent of the relay network and allow third-party reporting schemes[10] to be plugged in as and when it is mutually acceptable by both relay networks and its users.

LIN is not required to be used by miners or full nodes. Payments can be made in any token that's acceptable to the interested parties.

[10]for example, integrating Path Network or reporters in decentralized oracles or prediction markets provided the users trust them to be secure for their purposes

2.2.1 Creation

What's a relay node? A node that runs the Marlin relay node software can act as a Marlin node. A good internet connection is sufficient to become a Marlin node. Commodity machines these days running optimized overlay router software can process packets at a rate of 10 Gbps which is sufficient for our use case. Nevertheless, optimized hardware for the same is also commercially available[11].

Listing a node in the Marlin Marketplace. The Marketplace is where the nodes and relay networks advertise their characteristics. Users of the Marlin network can find a relay network of their choice based on the characteristics, capabilities and price advertised at the Marketplace. A node makes a *AddNodeToMarketplace* transaction to list itself in the Marketplace which contains its IP address, median bandwidth, longitude, latitude and the public key corresponding to the address with staked LIN.

Creation of a relay network. A node can create a *CreateRelayNetwork* transaction with details of its capabilities - the protocols that it would like to serve like Ethereum), maximum number of nodes it would allow to join, the minimum and maximum number of nodes it would enforce per geographical region denoted by coordinates, the minimum stake it requires from member nodes, the minimum configuration of the nodes(like bandwidth, CPU cores) and the minimum SLA it would require constituent nodes to provide in their geographical location. Relay networks are automatically listed in the Marlin Marketplace once created albeit with an inactive flag.

2.2.2 Management

Governance policies. The parameters in the *CreateRelayNetwork* transaction are ultimately to be followed by all nodes in the relay network. While the node creating the relay network initializes them, once other nodes join the relay network they ought to have a say in the network too especially considering that requirements of users and the competitive landscape for the relay network may change over time. Such changes to the relay network parameters require in-network governance.

Nodes are allowed to create proposals to change parameters and only the nodes in the concerned relay network are allowed to vote on it. While figuring the best governance mechanism may require a few iterations, we would like to point out that the on-chain governance discussed here is not as serious as the protocol related on-chain governance discussed widely elsewhere (and attacked for being plutocratic[7, 8]). Minority nodes in this case have the flexibility to leave and create a competing relay network if they feel that their proposals are more competitive. A competing relay network is much less threatening than a forked blockchain.

Entry regulation. There are two ways a node can join a relay network. A relay network after passing a majority vote (dictated by internal governance policies) can invite a node to join the relay network through a *RelayInvitation* transaction which has an expiry period. The node joins the network by sending a *AcceptInvite* transaction before the expiry of the invitation.

A node can also make a request to join a relay network through a *JoinRequest* transaction which the nodes in the relay network can accept or reject based on a voting round the proposal for which is created automatically if the specifications of the node meet the membership criteria laid out by the relay network.

Exit regulation. Permissionless systems are susceptible to high churn rates which may have disastrous consequences for a relay network as well as its users. Thus, unannounced exits by a node are heavily punished to encourage graceful exits[12]. Every relay network sets a notice period - the minimum time a node must give a relay network before it exits. If a node exits unannounced or before its notice period ends, its stake is repeatedly slashed for not meeting SLAs until its stake reduces to nothing. Users can take into account the duration of the notice period as well as the minimum amount of staked tokens mandated by a relay network before choosing to use them.

[11] for example, https://netfpga.org/site/#/systems/3netfpga-10g/details/

[12] Godfrey et al[9] show that simply replacing a dead node by a randomly picked node followed by some advanced strategies reduce churn for a P2P multicast overlay

Nodes wishing to leave a relay network make a *AnnounceExit* transaction. If the relay network finds a node to replace the node trying to exit, it can make a *AllowExit* transaction to allow the node to exit before its notice period ends. In practice, the *AllowExit* transaction is automatically made after a voting round on the proposal created by the *AnnounceExit* transaction. Nodes can also create a *ForceExit* transaction against a node in the relay network. If the majority (dictated by governance policies) believe that the node is misbehaving or should be kicked out for some reason, the node is removed from the relay network.

2.2.3 Packet propagation

Setup. Miners, wallets and other users that use the Marlin Relay network integrate the Marlin SDK which is responsible for interaction with Marlin nodes. The Marlin architecture is modular allowing protocol developers to write plugins to customize pre- and post-processing functions and execute custom code for packets related to their protocol at Marlin nodes, full nodes or wallets. This includes strategies on how to choose amongst the different relay networks - while some protocols may choose to use only one relay network (insecure), others may randomly choose between any relay network available that meets a certain criteria while some others may hardcode a list of relay networks to choose from using a probability distribution based on price and past performance.

Block Entry. A verifiable unbiased source of randomness $\mathcal{R}$ is used to choose n relay networks from those advertising the capability to serve the platform that a miner producing the block should send the block to. $\mathcal{R}$ and n can be customized by the protocol designers if required as per their security considerations. Once configured the Marlin SDK helps maintain connections with closest entry nodes of the different relay networks.

Once a miner produces a block, he sends it to the relay network(s) chosen using $\mathcal{R}$. He may or may not have to pay a fee to the entry nodes of the relay network based on the following choice to be made by protocol designers:

1. Protocol designers can require that a fee be paid by miners to cover the costs of block verification at the entry node. This ensures that the griefing factor for DDoS attempts is low since non-Marlin block producers cover the cost for block verification. However, this entails an added delay - the time required to verify the block at first hop. Invalid signed blocks can be used as fraud proofs (which can be optimized using off-chain compute systems) to avoid stake slashing for not meeting SLA guarantees.
2. An alternative is to require miners to stake tokens. This allows the platform to eliminate the step of block verification at the first hop and penalize the block producer if the block turns out to be spam.

Block Traversal. The nodes in a Marlin relay network form a mesh network with multicast paths optimized depending on the source. The packet loss rates, bandwidth and latency of the different paths are constantly monitored to ensure that multicast delivery of blocks can be completed in as little time as possible.

Details of incentivization techniques used to get nodes to forward packets to other nodes in their relay network are elaborated in Section 3.

Block Exit. Miners subscribe to closest Marlin relay nodes to receive notifications about new blocks and transactions. To prevent squatters a small fee is levied when initially subscribing. A correctly configured Marlin SDK (by the protocol designers) will ensure that nodes do not subscribe to relay networks from which they don't hope to receive any further data due to lack of publishers in that network. As soon as a Marlin node receives a new block it sends its hash to all its subscribers. Subscribers wishing to receive the block notify the node by sending it a micropayment which is followed by an exchange of bytes of the block. To avoid the added latency coming from the Announce and Request model, nodes can directly push to a whitelisted set of miners and take payments later.

2.2.4 Monitoring

The goal of monitoring is to provide feedback on a node's performance so that governance mechanisms at various levels can kick in.

Auditing. A number of anonymous auditor nodes monitor the working of Marlin nodes by subscribing to them as any other subscriber. Periodically, the auditor nodes participate in a Schelling scheme where they vote on whether a particular node was performing well and met its SLA obligations or not. If the node was found to not meet its SLA, then its stake is slashed and it might suffer other repercussions in the network's internal governance mechanism.

Auditor reward pool. We provision an auditor reward pool to properly fund the incentivization mechanism for auditors. It needs continuous infusion of funds to run sustainably which can take multiple forms (or a combination of them):

1. Nodes periodically contribute a fee to the reward pool without which they are cutoff from Marlin. This can be viewed as a membership/maintenance fee.
2. Any stake that is slashed goes into this reward pool.
3. Whenever nodes deposit/withdraw a stake, a portion is taken as fee and added to the reward pool. This can be viewed as an entry/exit fee.

The reward pool can be separate for different protocols if needed which provides flexibility in the design.

Auditor sampling. Auditors form a separate permissionless network. Nodes nominate themselves as auditors by staking tokens and making a *RegisterAuditor* transaction with smart contract giving their geographical location. They register themselves as normal subscribers with the Marlin nodes and are indistinguishable to them (via IP or otherwise). Given a pseudo-random *seed*, nodes compute $VRF_{n.priv}(seed) \rightarrow < hash, \pi >$. If $hash/2^{hashbits} < p$ where p equals the ratio of the average number of auditors required per Schelling round and the number of registered auditors in a region, then the node participates as an auditor during the following epoch. When submitting its results in a *Commit-Reveal* scheme described below, it also submits $< hash, \pi >$ which can be checked using the node's public key to verify if the node was eligible to act as an auditor in that epoch.

Schelling scheme. The auditors are asked to vote *Yes* or *No* on whether the node has met its obligations. This happens in two phases, a *Commit* phase where auditors submit a shielded vote and a *Reveal* phase where the auditors reveal their vote. The auditors who are part of the majority outcome get a reward from the reward pool.

Assertion 2.1. Auditors are incentivized to tell the truth.
This is an example of a coordination game where an auditor maximizes his reward by "coordinating" his response with other auditors[13]. In the absence of any communication, auditors are left to guess what the other auditors will do and will converge on two Schelling points - the truth and the untruth. The two-part *Commit-Reveal* structure ensures auditors cannot glean information about the votes of other auditors and adjust their strategy accordingly.

As long as the Schelling scheme is initially seeded with everybody telling the truth, the auditors are incentivized to continue telling the truth. Any deviation from the truth leaves them with a lower payoff. Therefore, telling the truth is a Nash equilibrium.

Assertion 2.2. Auditors can't be targeted.
Nodes can target auditors and prioritize them thereby giving them a better SLA than the normal subscribers. The Schelling scheme mitigate this by incentivizing auditors to stay anonymous the same way they are incentivized to tell the truth.

As long as the Schelling scheme is initially seeded with anonymous auditors, the auditors are incentivized to remain anonymous, lest the node target them with a better SLA leading to them not being in the majority outcome of the Schelling scheme anymore. Any deviation from remaining anonymous leaves them with a lower payoff. Therefore, remaining anonymous is a Nash

[13] Even if it may not be the truth, the only thing they care about is being in the majority

equilibrium.

3 Incentivization

Related work. Given cryptoeconomic systems work solely on the basis of incentives with no presence of a centralized court to punish defectors, it is important that transaction and block propagation in blockchains be incentive compatible. A significant body of work with respect to the incentive compatibility problem of blockchains has been on withholding attacks and selfish mining[5, 10, 11]. Babaioff et al.[1] propose an incentive mechanism for a very specific topology - a forest of d-ary trees. Abraham et al.[2] argue that when a node is part of multiple propagation paths, nodes at the same distance from the source wouldn't be able to charge more than an amount x_{min} designated by the system. This analysis has two issues - (i) a system designated minimum payment can't work in practice due to the possibility of a side-channel payment mechanism undercutting the minimum (ii) it does not take into account the fact that a node that forwards packet faster has a higher value than a slower one. Ersoy et al.[12] provide an incentivization scheme that is forwarding-dominant and Sybil-resistant as defined below.

Requirements. We thus arrive at the requirements we expect from a forwarding scheme-

1. *Forwarding-dominant*: It should be a dominant strategy for nodes to forward transactions and blocks instead of withholding them.
2. *Sybil-resistant*: Nodes should not have an incentive to not create sybil identities in the propagation path in order to capture a larger share of the fee.
3. *Performance rewarding*: Nodes should have a continuous incentive to improve their performance and forward packets to their children faster.
4. *Low propagation overhead*: The scheme suggested by Babaioff et al.[1] pays $log(\mathcal{H})$ units to forwarders for every unit of fee paid to the miner who successfully mines the transaction in a block. Such a scheme seems wasteful and suboptimal in a competitive market.

Fee distribution function. While Ersoy et al.[12], Babaioff et al.[1] and Abraham et al.[2] show the existence of a fee sharing strategy that achieves desirable properties in certain settings, we refrain from imposing a fee sharing structure into the system and rather leave it to the market participants to decide. We do so primarily because of the following reasons:

1. *Side-channel attacks*: Nodes can bypass any system imposed restrictions by creating off-band payment channels where they reallocate fees.
2. *Service differentiation*: Two nodes at the same level of the propagation path can have different costs due to usage of different service providers providing different levels of performance guarantees.
3. *Orphaning parents*: A node that is part of multiple propagation paths might choose a late-coming though cheaper fee history transaction as compared to a faster more expensive one. The premium that a child node is willing to pay for speed depends on what his children are willing to pay and his risk associated with not being in the path by not choosing the faster transaction.

The values of these risks and rewards will be determined by market participants over time. Any sort of system imposed restriction will lead to nodes acting suboptimally or finding ways to circumvent the restriction.

Fee distribution mechanism. In order for nodes that are part of the propagation path to get paid their due share, there should be a mechanism for them to claim their share. Such a fee distribution mechanism should satisfy the following properties:

1. *Orphaning-resistant*: Nodes further down the propagation path have an incentive to remove some of their parents from the path history and thus claim a larger share for those that remain (including themselves).

2. *Order-preserving*: A fee distribution mechanism in which not all nodes in the propagation path are rewarded uniformly, say in a decreasing order from the source, nodes have an incentive to rearrange the nodes in the path so to put themselves (or their allies) at higher paying positions.

In order to satisfy the above two properties, a transaction and a block is modified to contain a witness which is appended to the original content body.

$$Block_{new} = Block||Witness$$
$$Transaction_{new} = Transaction||Witness$$

The witness can be implemented in the following two ways:

1. *Signature chaining*: In this approach, every node in the propagation path appends the public key of the node it is forwarding the content to and signs the body with its private key. Let n_i be the i^{th} node in the propagation path, $n.pub$ and $b.priv$ be node n's public and private key respectively and f_i be some fee metric claimed by node n_i, then the witness forwarded by n_i to n_{i+1} is as follows:

$$Witness_{n_i} =< Witness_{n_{i-1}}, n_{i+1}.pub, f_i > || < Witness_{n_{i-1}}, n_{i+1}.pub, f_i >_{n_i.priv}$$

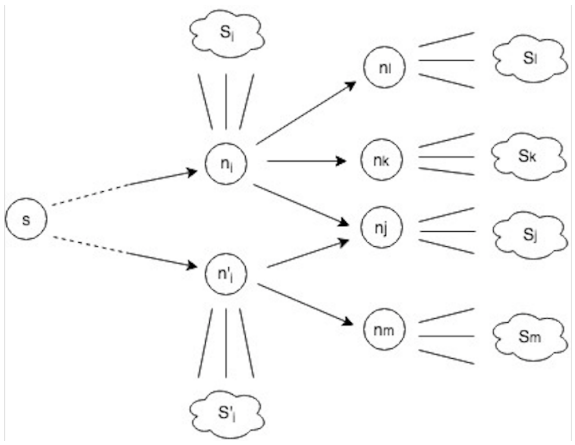
   However, this scheme requires a node to have knowledge of not only its children's IP addresses but public keys too. The scheme[13] described next resolves this issue.

2. *Key list*: A node appends a newly created key pair $< K_{pub}, K_{priv} >$ to the witness and signs the witness not including K_{priv}. The receiving node removes K_{priv} from the witness and uses it to sign the witness after appending its own public key and a new key pair. The transmitting node uses a unique key pair for every node it forwards packets too.

$$\begin{aligned} Witness_{n_i} =& < Witness_{n_{i-1}}, n_i.pub, K_{i+1}.pub, f_i > \\ & || < Witness_{n_{i-1}}, n_i.pub, K_{i+1}.pub, f_i >_{K_i.priv} \\ & ||K_{i+1}.priv \end{aligned}$$

Listed below are some notations used in the discussion that follows:

1. $\mathcal{S}_{i,s}$: The set of subscribers of node n_i including downstream Marlin nodes when the original source is s.
2. $\mathcal{S'}_{i,s}$: The set of subscribers of node n_i excluding downstream Marlin nodes when the original source is s. Since $S'_{i,s}$ is independent of s we simply denote it as S'_i.
3. $\mathcal{R}_{i,s}$: The revenue earned by node n_i including its share from the revenue of downstream Marlin nodes when the original source is s.
4. $\mathcal{R'}_{i,s}$: The revenue earned by node n_i excluding its share from the revenue of downstream Marlin nodes when the original source is s.



Assertion 3.1. Every node will be part of greater than one propagation path.

Reason. Let n_j be a Marlin node that receives packets only from node n_i when the original source is s. Let n_k be another potential source of packets for n_i. If n_i chooses to maintain a subscriber relationship exclusively with n_j, it is rational for n_i to take a fee of 1-ϵ times n_i's earnings $\mathcal{R}_{j,s}$ due to its knowledge that its rational for n_j to accept that fees and earn $\epsilon \cdot \mathcal{R}_{j,s} > 0$. In a competitive this will ideally lead to n_k proposing n_i with a fee cut of 1-2.ϵ. n_i would thus maintain a subscriber relationship with n_k too leading to a price war between n_j and n_k.

Assertion 3.2. The fee charged by every node will not be a constant x_{min}.

Reason. While we assert above that there will be a price war between nodes n_j and n_k, we disagree with the conclusion of Abraham et al.[2] of fees uniformly dropping to x_{min}. This is simply because of the value offered by differentiation of services which in our case is lower latency. Some nodes might have a competitive advantage in packet propagation either due to their location or internet service provider. Such nodes would be able to cater to their subscribers faster and thus charge a premium $\mathcal{P}$ over x_{min}. Moreover, bandwidth pricing is geography specific making x_{min} vary over regions.

Assertion 3.3. $\mathcal{P}$ does not aggravate centralization risks.

Reason. The discussion above seems to suggest that nodes running at advantageous locations or at good service providers might out compete other nodes leading to centralization of nodes in desirable networks. We argue that such centralization risks already exist with networks that are expected to use Marlin and can be curbed in similar ways. Rational miners of all platforms are expected to switch to networks that offer cheaper prices and better network performance. In fact the Internet is already centralized to an extent with submarine links being owned by a few companies at the transcontinental level and governments being able to impose censorship restrictions at national levels.

Rational Marlin nodes will price in the risks that come with the lure of predatory pricing as will its subscribers. As a result networks wishing for diversity will pay for it. Moreover, Marlin implements a suggestion that's been made for Casper to enhance diversity which requires that penalties to increase as defaults increase within the same time window.

Assertion 3.4. It is rational for a node to always forward packets.

Reason. A Marlin node can either be an independent node serving only the Marlin network or could also be or be colluding with a miner in a network using the Marlin relay network. We show that in both cases its in the interest of a rational miner to forward packets to other Marlin nodes.

Case 1. The Marlin node is independent.

If node n_i doesn't forward packets its reward is given by $\mathcal{R'}_{i,s}$. On the other hand, if n_i forwards packets its reward is given by $\mathcal{R}_{i,s} = \mathcal{R'}_{i,s} + f_i \cdot \mathcal{R}_{i+1,s}$. Since $f_i \cdot \mathcal{R}_{i+1,s} \geq 0$, it is rational for a non-colluding Marlin node to always forward packets.

Case 2. The Marlin node is colluding.

Let nodes n_i and n_j be aware of a transaction t, π_i and π_j represent the probability that nodes n_i and n_j produce a block respectively and π_0 represent the probability that nodes other than n_i and n_j who are aware of the t produce a block.

The expected reward of n_i and n_j if they both don't forward t any further is $\mathcal{R''}_{i,s} = \frac{\pi_i \cdot t}{\pi_i + \pi_j + \pi_0}$ and $\frac{\pi_j \cdot t}{\pi_i + \pi_j + \pi_0}$ respectively.

On the other hand, if n_i forwards t to n_k, its expected reward is $\frac{\pi_i \cdot t + f_i \cdot t \cdot \pi_k}{\pi_i + \pi_j + \pi_0 + \pi_k}$ which could exceed $\mathcal{R''}_{i,s}$ if f_i is set to $\frac{\pi_i}{\pi_i + \pi_j + \pi_0}$. Moreover, not only can n_j never be sure that n_i doesn't forward packets to n_k, n_k can receive those packets by acting as a normal subscriber to n_i or n_j under a sybil identity. Thus, it is rational for n_j to forward packets to n_k and derive a fee off $\mathcal{R}_{k,s}$.

As mentioned earlier, it's not necessary for f_i to equal f_j. This is not only due to service differentiation in terms of latency but also due to prior knowledge of paths. If n_i is aware that the fee chain

along the path used to propagate packets to n_j exceeds its own that is $\sum_{l=0}^{j-1} f_{l,s\to j} > \sum_{l=0}^{i-1} f_{l,s\to i}$ then f_i can be set to equal $f_j + \sum_{l=0}^{j-1} f_{l,s\to j} - \sum_{l=0}^{i-1} f_{l,s\to i}$. These are estimates that nodes will have to make over time.

Conclusion. We thus have an incentivization scheme that's forwarding-dominant and performance rewarding. Whether or not the system is Sybil resistant and has a low propagation overhead depends on the competition in the marketplace. In a market with low competition, nodes are incentivized to set extortionate rates and create Sybil identities in the path to increase their revenue share. However, in a properly functioning market where downstream nodes have a choice, rational opportunistic nodes wouldn't let the margin exist in turn decreasing the forwarding fee overhead. Sybil-resistance and performance rewarding are contradictory. As nodes continuously try to improve on performance and out-compete other nodes, if the market values the better performance, the node will have a small window of opportunity to get away with creating Sybil identities (charge a non-zero $\mathcal{P}$) while the others catch up.

4 Security

The Marlin Relay layer occupies a central position in the stack of the blockchain platforms it serves - all peers in a P2P network communicate using the networking layer. Attacks to compromise the safety and liveliness guarantees of blockchains by exploiting the networking layer already exist under certain conditions. Its important that Marlin does not significantly weaken the conditions to make such attacks viable. In the following sections we discuss some guarantees that blockchain platforms usually provide, ways in which an attacker can attack Marlin to harm Marlin or the platforms that use it followed by an exposition of counter-measures incorporated into Marlin to make it resistant to such attacks.

4.1 Preliminaries

Definition 4.1. A **blockchain protocol**[14] is defined as an algorithm pair (π^V, θ) where π^V parameterized by validity function V maintains a local state s and θ takes as input a difficulty parameter κ and s and outputs an ordered chain of blocks B. B is considered valid iff $V(B) = 1$.

Definition 4.2. Around nineteen years ago, in 2000, Eric Brewer conjectured a trade-off between consistency, availability and partition tolerance called the **CAP Theorem**. In simple terms, it states that a distributed system cannot simultaneously provide:

1. Consistency: all nodes have an identical copy of the state
2. Availability: each request eventually receives a response
3. Partition-tolerance: nodes may be partitioned into multiple groups that cannot communicate with one another

Most distributed systems today recognize that communication networks are unreliable: messages maybe delayed and sometimes lost forever. As a result partitions are inevitable and systems must forfeit either consistency or availability. Blockchains are no different - being permissionless peer-to-peer systems they are susceptible to the vagaries of public networks and thus prone to partitioning. A Bitcoin client can prefer availability by trusting transactions as soon as they are in a block (and thus risk accepting a UTXO that has been double spent in another network partition which might be dropped during conflict resolution once the partitions merge) or opt for consistency by waiting for the transactions to be at least a few blocks deep. Similarly, while some systems like Tendermint[15] explicitly favour consistency, most other blockchain platforms today favour availability, for example, Ethereum Casper[16].

4.2 Requirements

A poorly designed relay network could be threat to several important features of blockchain systems that have come to be considered a necessity. We list below some of those features and the transitive expectations they result in for a relay network such as Marlin.

1. Censorship resistance: A transaction t submitted to a network of nodes maintaining a blockchain $B = b_1, b_2, b_3, ...b_t$ is expected to be executed and included in one of the blocks b_t provided it follows the validity rules. While there can be several reasons for t not being present in B sufficient time after its shared such as, t being a double spend or the transaction fee set too low during a period of high congestion, the case that concerns the network layer is *network censorship*. Its expected of a relay network such as Marlin to ensure that transactions reach a sufficient number of nodes of the blockchain platform and leave it to the implementation of π^V to decide whether or not the transaction is valid (for example, some blockchain protocols may have a provision for blacklist addresses).
2. Fast finality: The time by which a transaction can be considered executed and irreversible depends on the speed with which both the transaction as well as the block can be propagated across the network. This requirement is an extension of the requirement above - its not just necessary that transactions get delivered to sufficient number of nodes so that they get included in a block but its also also necessary that the same happens with a low latency to provide a good user experience. Some use cases such as dice games would be impossible while block times would wildly fluctuate (in case of synchronous blockchain protocols) or fork rates would waver (in case of asynchronous protocols) without a consistently performing relay network.
3. Decentralization: It should not be possible for a single authority to control a significant portion of the blockchain network. The network layer being an important part of the protocol, its necessary to ensure that the Marlin network remain decentralized and also not be a gateway to facilitate 51% attacks over its users.
4. Cheap cost: Users expect low fees from blockchain enabled transactions. Since in the long term fees are the means to cover the costs of mining, its necessary that the net cost of mining not increase because of the fees charged by relay nodes.
5. Availability and consistency: The CAP Theorem dictates that both availability and consistency can't be guaranteed when the network is partitioned. Most blockchain system designers account this fact and make a cautious choice to design the system either in favour of availability or consistency. Network faults are a major cause for network partitioning. As a result, its important for a relay network to detect and correct faults quickly so that the periods of inconsistency or outage can be kept short. If allowed to prolong either the blockchain would come to a standstill (if it favours consistency) or lead to forks (if it favours availability) which would lead to several seemingly confirmed transactions to be reversed (double spend attacks).

4.3 Attack scenarios

4.3.1 DDoS attacks

Attack description. Marlin nodes in a region can be DDoSed thereby leading to service disruption (unavailability and slower finality) and partitioning.

Countermeasure 1. Resource isolation.

Separate resources are allocated for internal data transfer to other Marlin nodes and public data transfer to PubSub API subscribers.

Since internal resources are utilized by a small number of known Marlin peers, any DDoS on the internal resources reduces to a simple DoS attack with a small scope. This can be mitigated easily by rate-limiting/flow control and similar techniques. A DDoS-proof internal network ensures that any DDoS will be contained within a small region and won't affect the wider ecosystem.

Countermeasure 2. DDoS Prevention and Redundancy.

Multiple Marlin nodes in a region who individually protect their public-facing PubSub API against DDoS.

Protection techniques include allocation of spare capacity, threat monitoring, dynamic black hole policies, etc. This could even be outsourced to big commercial providers like CloudFlare or AWS who might provide better (and cheaper) protection because of their huge scale. Marlin nodes would advertise a SLA depending on the precautions they undertake. These guarantees (and limitations thereof) can be factored in by miners using the Marlin network while compelling Marlin nodes to ensure that the SLA is provided or have their stakes slashed.

This is further exemplified by having multiple Marlin nodes serving a region. Failure in a single node results in other nodes picking up the slack.

4.3.2 Extortion and cartelization

Attack description. Subscription prices set by Marlin nodes may be extortionate and can lead to a form of selective censorship against a few nodes or a region. Cartels can be formed between all Marlin nodes in a region or relay networks serving Marlin's users.

Countermeasures. The Marlin network mitigates this using market competition at various levels.

Nodes in a region compete to get more subscribers. Any node which tries to set extortionate prices is going to be immediately undercut by other nodes. The price can be expected to quickly go back to market rate.

Nodes can still engage in extortion by forming regional cartels. In this case, internal network governance kicks in to punish/kick out these cartels and on-board new nodes. Essentially, there is competition between nodes regarding network membership and any badly behaving node can swiftly be replaced with a good one.

Moreover, censorship can either be implemented through whitelists or blacklists. A node suspecting itself to be part of a blacklist can create Sybil identity to fetch packets thus escaping the blacklist. Whitelist based censorship is not possible given the high penalties associated with not meeting SLAs which is checked by disguised auditors.

The final line of defense is the Marlin marketplace where multiple networks compete against each other. This will ensure that any network with poor price or performance will be out-competed by better networks.

4.3.3 Tax evasion

Attack 1. Nodes might make side-channel payment arrangements with subscribers and report a low on-chain income so that a smaller amount has to be passed on to its ancestors in the propagation tree.

Countermeasure. This can be considered a case of extortion and tackled in a similar way.

Attack 2. Nodes can claim to be the origin of packets in order to not have to submit the Witness and thus share their revenues with ancestors in the propagation path.

Countermeasure. Even the first Marlin node in the propagation path has a Witness signed by the SDK of the miner using the Marlin network that has its public key. Thus no node can fake their position in the path to avoid having to share revenues.

4.3.4 Censorship against a miner

Attack description. Node(s) decide to censor blocks from a particular miner.

Countermeasures. Backup options in the short term. Stake slashing in the medium term.

Miners maintain a diverse set of Marlin nodes which might be far away as backup options in addition to nearby nodes. Even if the miner has to send the block to a backup node on the

opposite side of the planet, he incurs less than 100ms latency penalty which is good enough in the short term. An additional discussion on how native gossip in user networks can help is present in Section 4.3.5.

The censoring nodes fail to meet their SLAs and get their stakes slashed at every instance of censorship making it a very expensive attack to maintain in the medium term.

Analysis. We analyze the above situation to derive a griefing factor (g).

$$g = \frac{c_v}{c_a} \tag{3}$$

where c_a is the cost to the attacker arising from stake slashing and c_v is the cost to the victim arising from lost revenue as a result of 100ms latency penalty and higher chances of block getting orphaned.

Let t_P be the average block propagation time. Let Pr denote probability and T be a random variable modelling time between consecutive blocks and $C_T(t) = \Pr[T \leq t]$ be its CDF.

A fork occurs when a competing block is produced before the original block has time to propagate through the network.

$$\Pr[\text{fork}] = \Pr[T \leq t_P] \tag{4}$$
$$= C_T(t_P) \tag{5}$$

For example, in Bitcoin/Ethereum, PoW mining is a Poisson process. This means that the interval between blocks is an exponential distribution with mean as the desired block time(t_B):

$$\Pr[T = t] = \frac{1}{t_B} e^{-t/t_B} \tag{6}$$
$$\implies C_T(t) = 1 - e^{-t/t_B} \tag{7}$$

The cost of this attack to the victim(c_v) would be lost revenue as a result of increased fork rate. Let r_B be the block reward and r_o be the reward for orphans/uncles.

$$c_v = (C_T(t_P + 0.1) - C_T(t_P)) * (r_B - r_o) \tag{8}$$

The cost of this attack to the attacker(c_a) is the stake slashed.

$$c_a = s(n) \tag{9}$$

where n is the number of nodes getting their stake slashed and s is the stake slashing function.

4.3.5 Eclipse attacks, Routing attacks, Partitioning and Blackholing

Attack description. Related to the discussion above is a popular attack known as eclipse attacks. In a usual P2P network, in order to execute a successful eclipse attack, an adversary takes control of a sufficient number of IP addresses (eg. botnet groups, infrastructure attackers like ISP's) and monopolizes all connections to and from a victim node, isolating the victim from the rest of its peers in the network[17]. Moreover, routing attacks can poison BGP tables to divert packets from their intended course. The attacker can then:

- filter the victim's view of the blockchain leading to double spending attacks
- force the victim to waste compute power on obsolete views of the blockchain (engineering block races)
- coopt the victim's compute power for its own nefarious purposes (selfish mining)
- eclipse a fraction of miners eliminating their mining power from the rest of the network, making it easier to launch mining attacks.

In the case of Marlin such an attack can two forms:

Attack 1. Marlin nodes bribed by competing miners could try to eclipse a set of miners in a region.

Attack 2. The Marlin relay network itself is partitioned due to random network faults or by an adversary (such as an ISP blocking all Marlin related traffic from entering/exiting the region). A particular region could also be blackholed by an ISP blocking all Marlin related traffic inside the region.

Countermeasures. The countermeasures and analysis against attack 1 is similar to the discussion in Section 4.3.4. As for attack 2 a number of measures and fallbacks ensure that the effects of an attack are mitigated.

Countermeasure 1. Encryption

Marlin encrypts internal traffic and could even be run on standard ports making it difficult to distinguish from regular web traffic. This makes blocking Marlin traffic a difficult and messy task.

Countermeasure 2. Stake slashing

Marlin nodes will fail to provide their SLA guarantees and get their stake slashed. While making the cost of a bribery attack high, nodes are also incentivized to seek better alternatives and backup options in case they get partitioned or blackholed.

Countermeasure 3. Gossip as fallback

The Marlin Client installed by the users has a mechanism to gossip transactions and blocks to users in close proximity. Marlin nodes can also listen to this gossip and use this as a fallback to "punch through" any thin network partitions. As a result, while blocks and transactions can travel larger inter-continental distances quickly using the Marlin network, any regional disruptions can be countered with the existing robustness of the gossip protocol. This gives us the robustness of gossip while not losing out on the performance benefits.

Countermeasure 4. Diversification

Marlin nodes get their stake slashed based on the number of nodes which get their stake slashed in a given timeframe. Nodes thus have an incentive to decorrelate their failure modes with other nodes as much as possible. This encourages nodes to diversify their ISPs (among other things) and reduces the risk of a single party getting significant leverage in the first place.

Sidenote on the impact of CAP on Marlin. Despite our best efforts, its a given that partitions in a distributed system are inevitable. In the case of partitioning, Marlin provides liveness over safety. The inconsistency is evident in that a Marlin node does not inform subscribers about new blocks while there exists one in the partition it is disconnected from. However, we mitigate this issue through redundancy and diversification much like permissionless systems today while still not adversely affecting the performance guarantees.

4.3.6 Sybil attacks

Attack description. In permissionless settings an attacker can easily create several fake identities (key pairs are quick to generate and IP easy to spoof). This allows an attacker to present miners with a large set of relay networks each having widely distributed nodes which are all in reality just the attacker himself.

Countermeasure. We require Marlin nodes to stake LIN. which acts as a defence against a single entity making countless sybils. In the case of nodes faking geographical location, the SLA requirements ensure that such nodes would be penalized and thus find it untenable.

4.3.7 Stake reuse attack

Attack description. While requiring nodes to stake LIN prevents them from creating multiple identities without increasing their stake, a node could still list itself (with a single identity) as part

of multiple relay networks. When executed at scale, several relay networks would end up being networks constructed by the same set of nodes defeating the purpose of randomly choosing a relay network through which to broadcast.

Countermeasure. We enforce that a node cannot be part two relay networks where the intersection set of their capabilities is not null.

4.3.8 Parent orphaning

Attack description. In order to increase revenue, a set of Marlin nodes could collude and use witnesses that maximize the number of nodes from the colluding set when attributing packet sources in the propagation path for dividing payments even though they receive packets faster from a non-colluding node. Such an attack would drive nodes possibly bearing a higher cost to provide better performance out of the market thus decreasing the network's overall performance.

Countermeasure. Nodes internally prioritize nodes that include them in the propagation path during reward distribution. Thus nodes guilty of misattribution risk being deprioritized and thus losing out on revenues it could receive by propagating packets faster to its children in comparison to its competitors serving the same nodes.

References

[1] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.

[2] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. *CoRR, abs/1612.02916*, 2016.

[3] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. pages 1–10, 09 2013. doi: 10.1109/P2P.2013.6688704.

[4] Garrett Hardin. The tragedy of the commons. *science*, 162(3859):1243–1248, 1968.

[5] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[6] Vitalik Buterin. Schellingcoin: A minimal-trust universal data feed. `https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/`, Mar 2014. Accessed on 12/01/2019.

[7] Vitalik Buterin. Governance, part 2: Plutocracy is still bad. `https://vitalik.ca/general/2018/03/28/plutocracy.html`, Mar 2018. Accessed on 12/01/2019.

[8] Vlad Zamfir. Against on-chain governance. `https://medium.com/@Vlad_Zamfir/against-on-chain-governance-a4ceacd040ca#914f`, Dec 2017. Accessed on 12/01/2019.

[9] P Godfrey, Scott Shenker, and Ion Stoica. *Minimizing churn in distributed systems*, volume 36. ACM, 2006.

[10] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[11] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[12] Oğuzhan Ersoy, Zhijie Ren, Zekeriya Erkin, and Reginald L Lagendijk. Transaction propagation on permissionless blockchains: Incentive and routing mechanisms. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 20–30. IEEE, 2018.

[13] Phil Daian and Vitalik Buterin. Incentivizing a robust p2p network/relay layer. `https://ethresear.ch/t/incentivizing-a-robust-p2p-network-relay-layer/1438/5`, Mar 2018. Accessed on 12/01/2019.

[14] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

[15] Jae Kwon and Ethan Buchman. Cosmos. `https://cosmos.network/cosmos-whitepaper.pdf`, Jan 2019. Accessed on 12/01/2019.

[16] Vitalik Buterin. Casper the friendly finality gadget. `https://ethresear.ch/uploads/default/original/1X/fdbebd67c8a9671efabf4e53d6267789cd91d96c.pdf`, Oct 2017. Accessed on 12/01/2019.

[17] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. `https://cosmos.network/cosmos-whitepaper.pdf`, 2015.