Less is More: Fairness in Wide-Area Proof-of-Work Blockchain Networks

YIFAN MAO, the Ohio State University, USA
SHAILESHH BOJJA VENKATAKRISHNAN, Ithe Ohio State University, USA

Blockchain is rapidly emerging as an important class of network application, with a unique set of trust, security and transparency properties. In a blockchain system, participants record and update the 'server-side' state of an application as blocks of a replicated, immutable ledger using a consensus protocol over the Internet. Mining blocks has become lucrative in recent years; e.g., a miner receives over USD 200,000 per mined block in Bitcoin today. A key factor affecting mining rewards, is the latency of broadcasting blocks over the network. In this paper, we consider the problem of topology design for optimizing mining rewards in a wide-area blockchain network that uses a Proof-of-Work protocol for consensus. Contrary to general wisdom that a faster network is always better for miners, we show a counter intuitive result where a slower network is actually beneficial to some miners. This is because competing miners must choose neighbors that not only decrease their own latency to others, but also ensure that the latency between other miners do not decrease because of itself. We formalize this problem, and provide both theoretical analysis and experimental results to support our claim.

Additional Key Words and Phrases: blockchains, peer-to-peer, topology design, network games

ACM Reference Format:
Yifan Mao and Shaileshh Bojja Venkatakrishnan. 2018. Less is More: Fairness in Wide-Area Proof-of-Work Blockchain Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Blockchains have emerged as a secure, transparent and decentralized alternative to centralized servers for implementing diverse network applications such as payments, digital marketplace and smart contracts [12]. Blockchains are implemented over a peer-to-peer (p2p) overlay, where each node in the overlay maintains a replica of an immutable ledger called the blockchain in which transactions (e.g., payments) are recorded as they are made over the network. To function correctly, it is essential for the blockchain replicas at different nodes to be in agreement with each other which is achieved using a consensus protocol. Since the Nakamoto consensus [30] was proposed in Bitcoin in 2008, several alternative consensus algorithms have been proposed in recent years with useful performance properties [14, 22]. Nevertheless, a number of mainstream blockchains—notably blockchain based cryptocurrencies—rely on the Nakamoto consensus, also called as proof-of-work (PoW), for their operation [5, 6, 28, 30]. The combined market capitalization of these cryptocurrencies currently stands over a trillion USD [4].

In a PoW cryptocurrencies, new payment transactions are consolidated into blocks and appended to the blockchain in a process called mining. A block in Bitcoin, for instance, contains an average of around 2000 payment transactions [1]. New blocks are generated by nodes in the p2p overlay, called miners, that compete to solve a computationally difficult cryptographic hash puzzle for each

Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

new block. The computational challenge involved in creating new blocks dissuades attackers from readily generating incorrect blocks and secures the blockchain. Blocks are mined in sequence with each mined block holding a reference to a unique parent block that was mined before the block. When a miner mines a new block, it broadcasts the block over the p2p network so that other miners can start mining the next block using the received block as a parent.

Miners are rewarded in proportion to the amount of computational effort they put in for mining new blocks [24]. The amount of rewards earned by a miner depends on the number of blocks the miner mined that has been included in the blockchain. Thus miners are incentivized to mine as many valid blocks as possible to maximize their individual payoffs. Increasing the computational effort (i.e., the hashrate of a miner [36]) is a natural approach to increasing the number of blocks mined, and hence the rewards earned. For a fixed hashrate, a miner may also be able to increase rewards by being well-connected in the p2p network and/or being favorably located in the network to send and receive blocks faster than other competing miners on average [19, 21, 41]. Recent works have explored various ways including topology redesigns [26, 33, 35], fast relay networks [10, 11, 23], and coding or compressing blocks [16, 32] to reduce the latency of block propagation in the network. The conventional wisdom here is that a faster network that is able to disseminate blocks quickly is generally more desirable that a slower network, as it allows miners to hear about newly mined blocks promptly thus reducing the likelihood of mining new blocks over older blocks (a process called as forking §2). A faster network also facilitates a higher rate of mining blocks, which has positive implications for increasing the rate at which payment transactions are confirmed in the blockchain (i.e., the throughput) and reducing transaction confirmation delays [14].

In this paper, we study the impact of the network on the mining reward of each miner in PoW blockchains with wide area networks such as Bitcoin or Ethereum. Miners in these network are often geographically far away from each other resulting in significant (e.g., 100s of milliseconds or more [19]) propagation delays while broadcasting blocks [18, 19] compared to the rate of block generation in the network. For instance, blocks are mined once every 13 seconds on average in Ethereum, once every minute in Dogecoin, once every 2.5 minutes in Litecoin and once every 10 minutes in Bitcoin. In private blockchain networks, the mining rate can be configured to be even faster [38].

In these cases, the topology of the p2p overlay has a significant effect on the dynamics of block propagation and consequently the rewards earned by competing miners. We consider topology design as a game played between miners, where the number and choice of peers (miners) a miner connects with is the action and the average fraction of blocks mined by the miner that are included in the blockchain is the reward for the miner. Since analyzing the space of all possible strategies that miners can follow and their associated rewards is intractable, we study a simple class of cooperative strategies in which miners group themselves into clusters and choose neighbors primarily within their respective clusters. We also study the non-cooperative policy where the number and choice of neighbors for a miner are chosen independently by the miner.

Our main result is a counter-intuitive phenomenon where payoffs are maximized for miners that organize themselves into a tightly-connected 'dominant' cluster comprising of roughly 70% of all the miners in the network. Miners not part of this dominant cluster receive suboptimal payoffs. Interestingly, the payoffs for miners in the dominant cluster starts decreasing if the size of the cluster either increases or decreases from its optimal size of 70% of the entire network. This is because for a miner to maximize its reward, it is not only essential for the miner to be well-connected (i.e., have low latency paths) to the rest of the network in order to receive and send blocks fast, but also for the rest of the miners to *not* be well-connected so that they receive blocks slower than the miner on average. A miner that increases its connectivity to the rest of the network by increasing its number of neighbors is not only creating new low-latency paths between itself and others, but also enabling new low-latency paths between other pairs of miners through itself. On the other hand, a miner that has too few neighbors risks having paths that are of higher latency to other miners. For miners in the dominant cluster, the 'right' balance occurs by choosing an appropriate cluster size, and connecting to primarily neighbors within the cluster.

Experimentally we observe that the clustering effects mentioned above can occur naturally simply due to the non-uniform geographical distribution of miners around the world. For example, if 70% of all miners in the world are located in either Europe or North America (the rest being in other continents), then even with a constant degree random topology the miners in Europe and North America will end up being a dominant cluster with low intra-cluster latency compared to miners in other continents. As such, when considering real-world geographies, we observe an unfair advantage to miners of regions where there is a rich concentration of miners and vice versa. However, here too as before we observe that the payoffs to miners in central locations (Europe or North America, in our experiments) degrades if they increase their number of neighbors beyond a certain threshold. Miners in remote locations strictly benefit by increasing the number of connections they make.

We summarize the key contributions of the paper below.

(1) We propose a simple theoretical model to analyze the fraction of successful blocks mined by each miner in a wide-area PoW blockchain network with heterogeneous link latencies.
(2) Under a strategy where miners organize themselves in to two tightly-connected clusters with sparse connections between the clusters, we derive the payoff each miner receives. We show that the payoffs to miners in the larger dominant cluster is maximized when the size of the cluster is 70% of the network size.
(3) We perform a similar analysis when there are three clusters, and show that payoffs are maximized when the dominant cluster includes 60% of all miners in the network. We also show it is beneficial for small clusters to merge into a larger cluster, rather than remain as separate clusters.
(4) We develop an event driven simulator on Omnet++ [8] to simulate mining and block propagation over a world-wide network of 270 miners based on Bitcoin's network protocol.
(5) We present experimental results to validate our theoretical findings; we also present candidate strategies that miners can follow in order to alleviate clustering bias and maximize their rewards.

2 BACKGROUND

We use Bitcoin as a representative example of a blockchain system, to provide a background on PoW blockchain networks. Other PoW blockchains follow a conceptually similar architecture, potentially with different algorithmic parameters. Bitcoin is a digital payment system, that allows end-users to send and receive currency between each other in a secure way. Payments made over Bitcoin are recorded in an immutable ledger called the blockchain, as an ordered sequence (earlier payments appear before later payments), which allows the blockchain to verify the validity of payments and prevent double spending. All transactions made since the inception of Bitcoin in 2008 are registered in the blockchain, making it a fairly large database (over 300 Gigabytes today).

The blockchain p2p network. Bitcoin operates over a peer-to-peer network of servers and clients. Clients are end-devices running a lightweight 'wallet' application for making payment transactions. Servers are typically equipped with high-performance computational hardware (ASICs, GPUs etc.) which they use to validate transactions and extend the blockchain. Clients connect to one or more servers via TCP connections; servers accept incoming connection requests from clients. In addition servers also connect to other servers via TCP. Bitcoin specifies each server can have a maximum of 125 connections to other clients or servers.

Proof-of-Work consensus. The blockchain is structured as an ordered sequence of blocks, where each blocks contains a small number (e.g., 2000) of ordered transactions. The very first block in Bitcoin is called as the genesis block. Servers in Bitcoin each maintain a local replica of the blockchain, which they use for verifying new payment transactions and extending the blockchain. To provide a secure, trustworthy payment service, it is essential for the blockchain replicas at different servers to be consistent with each other. Consistency across the blockchain replicas in Bitcoin is achieved via the Proof-of-Work (PoW) consensus protocol [30].

When a client makes a payment in Bitcoin, it encodes the payment information as a transaction message and broadcasts it over the p2p network. Each server receives transaction messages, verifies their validity (against older payment transactions recorded in the blockchain), batches the verified transactions in to a new block and digitally signs the block . For a block to be considered valid a server must also include the solution to a computationally difficult cryptographic hash puzzle with the block. Servers solve this puzzle via brute-force guesswork; the time it takes for a server to solve a puzzle is commonly modeled as an exponential random variable whose rate is proportional to the number of hash computations per second the server's computational hardware can perform [14]. Each block also includes a reference to a unique block on the blockchain called the parent block (a block is a child block of its parent). The number of blocks encountered while traversing along the parent links starting from a block, until the genesis block is reached, is called as the height of a block. Bitcoin follows the 'longest chain protocol' for choosing parent blocks when creating a new block: a new block must point to a block whose height is the greatest on the blockchain. This process of generating a new block is called as mining.

When a server mines a block, it immediately broadcasts the block to other servers via flooding over the p2p network. Servers receiving a block verify the validity of the block (e.g., validity of transactions in the block, correctness of the solution to the cryptographic puzzle etc.). If found valid, they append the block it to their local blockchain replica and relay the block to their neighboring peers in the network. They then start mining the next block using the received block as parent provided it has the greatest height on the blockchain. Occasionally, it is possible for two blocks to be concurrently mined by different miners over the same parent block. This creates a *fork* in the blockchain, where there are multiple blocks at the same height. In this case a server mining for a new block must mine it on top of the block it *received the earliest*, from among the blocks at the same (greatest) height. Over time, one of the forked chains will eventually grow longer than others to become part of the longest chain. Only transactions included in blocks on the longest chain are considered confirmed by the network. Blocks not on the longest chain do not contribute to extending the ledger of confirmed transactions.

Mining rewards. To incentivize servers to mine new blocks, a server receives a monetary reward for each block it mines that becomes part of the longest chain. The reward comes from two sources. First, whenever a payment transaction is made over the blockchain the payer includes a transaction fee in the transaction which goes to the server that verified the transaction and included it in its block as a reward. In 2021 the average fee per transaction over Bitcoin was around 3 USD [2].

The second source of reward for a server comes from minting new cryptocurrency each time a block is mined. In Bitcoin, currently a miner is allowed to mint 6.25 bitcoins (worth over $200,000$ USD today) every time it mines a block. These newly minted bitcoins are included as a separate transaction in the block, where the payee is the server that mined the block and the payer is a special account address reserved specifically for this purpose. Today, the rewards earned from minting new bitcoins represent $> 98\%$ of the total reward earned by a miner in a block [9] (compared to reward from transaction fees which is $< 2\%$ of the overall reward). However, the amount of bitcoins that a miner can mint per block is set to exponentially diminish over the years in Bitcoin. Therefore, transaction fees would eventually become the sole source of reward for miners.

Impact of network on rewards. Mining is a lucrative business in cryptocurrencies like Bitcoin. To maximize the likelihood of a newly mined block becoming part of the longest chain, it is essential that the block is broadcast to other servers as quickly as possible. If broadcast is delayed, there is a risk that the block will lose out to a competing block mined at the same height which reached other servers earlier. Conversely, a server must also hear about newly mined blocks as quickly as possible. This is so that a server does not waste time and computation mining a block at a certain height, when a block at that height has already been mined by another server. We use the terms servers, miners and peers interchangeably in this paper. We also restrict our focus to only the overlay network of miners, and do not consider client connections as they do not have a significant impact on the block propagation dynamics.

3 SYSTEM MODEL

3.1 Network Model

We consider a simple analytical model of the blockchain network to theoretically understand mining rewards earned under different topologies. The model we use for our experiments in Section 5 is richer and more realistic as detailed in Section 5.1. Nevertheless, the simple model we discuss in this section is useful in providing intuition on the effects of WAN latencies and network topology on mining rewards.

We model the blockchain network as an undirected graph $G(V, E)$ comprising of a set of miners V and communication links (e.g., TCP links as in Bitcoin) E between them. Blocks are mined in rounds, where in each round exactly one miner mines a block and broadcasts it over the network. We count rounds starting from 1 and use $\mathbb{N} = \{1, 2, 3 \ldots\}$ to denote the set of all rounds. Each miner $v \in V$ has an associated hash power $h_v \in (0, 1)$ with $\sum_{v \in V} h_v = 1$. The miner that mines a block in a round is chosen randomly and independently of prior rounds, with h_v being the probability of choosing miner v. We consider a uniform hash power, i.e., $h_v = 1/|V|$ for our analysis; however, our results can be extended to non-uniform hash power assignment in a straightforward way.

When a miner mines a block, it broadcasts the block to all of its neighbors in G. Each link (u, v) has a latency $l_{(u,v)} \in (0, 2)$ which is the time it takes for the block to propagate from u to v (we assume $l_{(u,v)} = l_{(v,u)}$ for any edge $(u, v) \in E$) through link (u, v). When a miner v receives a block (that has not been previously received) from neighbor u, it immediately forwards the block to all of its neighbors other than u. If a miner receives a block that it has previously seen, it does not forward the block to its neighbors. In practice, blocks received are first validated by miners before being forwarded to neighbors. The time taken to validate a block is often small (few milliseconds or less), compared to propagation delays on wide area networks (tens or hundreds of milliseconds). We have therefore neglected the validation delays in our model; however, our analysis techniques continue to hold even if there are non-zero validation delays.

Rounds are modeled to occur periodically, every 1 time unit. Each miner maintains a local replica of the blockchain based on the blocks it has received so far. When a miner v is sampled to mine a block during a round, it mines a block on top of the most recent block it has seen on the longest chain in its blockchain. If there are multiple competing longest chains, the miner mines the fresh block on top of the earliest received block among all blocks at the ends of the competing chains. Before the mining process starts at round 1, we assume there is a unique genesis block that is known to all miners. We refer to the genesis block as a block mined at round 0. A block mined during round r will be referred to as block r. For any block r, we call the number of blocks on a path from the genesis block to block r on the blockchain as the height of block r. The genesis block has a height of 0.

For each blockchain network $G(V, E)$ we consider an associated latency graph $L(V, \tilde{E})$ comprising of miners V and edges $\tilde{E}$ between every pair of miners. Each edge $(u, v) \in \tilde{E}$ has a weight $\delta_{(u,v)} \geq 0$ which is the minimum time it takes for v to receive a block mined by u over network G (or vice-versa, i.e., $\delta_{(u,v)} = \delta_{(v,u)}$). The latency graph L dictates the number of blocks mined by different miners that are included in the longest chain, and therefore is going to be the focus of our analysis. We assume all miners follow protocol, and leave a discussion on various possible adversarial actions that miners can take to future work. We do not model transaction generation and broadcast.

3.2 Reward Model

When a block mined by a miner is included in the longest chain, the miner incurs a reward for mining the block as explained in Section 2. While a number of factors affect the rewards earned by a miner per block—such as the plaintext size of the transactions in the block, number of unconfirmed transactions currently in the network, height of block etc.—for simplicity we assume the reward per block included in the longest chain to be a constant. Using this model as a motivation, we consider a reward model in which the aggregate reward earned by a miner is proportional to the fraction of the blocks mined by the miner in the longest chain. For a miner $v \in V$, we denote by $F_v \in [0, 1]$ the average fraction of blocks mined by v that are included in the longest chain over a long time horizon, i.e., assuming the following limit exists

$$F_v = \lim_{R \to \infty} \frac{\sum_{r=1}^{R} \mathbf{1}_{\text{block } r \text{ is mined by } v \text{ and is included in the longest chain}}}{\sum_{r=1}^{R} \mathbf{1}_{\text{block } r \text{ is included in the longest chain}}}. \tag{1}$$

In a fair network with n miners, $F_v = 1/n$ for all $v \in V$. However, if the network is unfair, some miners have a greater than $1/n$ fraction of blocks mined by them in the longest chain, while others have less than $1/n$ fraction of blocks in the longest chain. We present how certain miners can gain such an advantage by carefully choosing whom to connect with in the network, and what the affected miners can do to alleviate their disadvantage. We reiterate that outside of choosing neighbors carefully, the miners do not deviate from protocol in any way. Choosing neighbors carefully in a blockchain network (but otherwise following protocol) is not considered as adversarial behavior. Hence, the policies we present in this paper are all "legal" methods by which some miners can obtain an unfair advantage over others.

We also model the costs incurred by miners for electricity, cooling etc. while mining blocks. If the average cost incurred per mined block (whether or not it is included in the longest chain) is not negligibly small compared to rewards obtained per block, then a miner is not only interested in maximizing the fraction of blocks it mined that become part of the longest chain, but also in minimizing the fraction of wasted blocks it mined that did not become part of the longest chain. Note that maximizing the fraction of blocks mined by a miner included in the longest chain is distinct from minimizing the fraction of blocks mined by a miner that have been excluded from the longest chain. For a miner $v \in V$, we let W_v be the fraction of blocks mined by v that are excluded from the longest chain over a long time horizon, i.e., assuming the following limit exists

$$W_v = \lim_{R \to \infty} \frac{\sum_{r=1}^{R} \mathbf{1}_{\text{block } r \text{ is mined by } v \text{ and is excluded from the longest chain}}}{\sum_{r=1}^{R} \mathbf{1}_{\text{block } r \text{ is mined by } v}}. \tag{2}$$

Depending on the precise values for reward earned by a miner per block in the longest chain, and the cost incurred per mined block, a miner v may be interested in maximizing a weighted difference

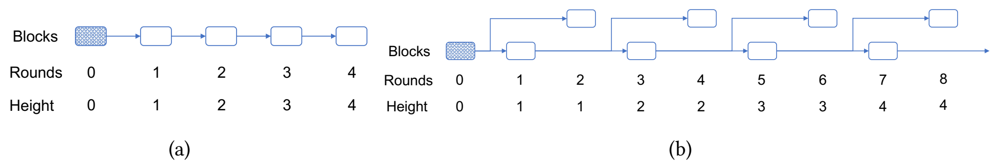


Fig. 1. Evolution of the blockchain if miners are organized as a single cluster. (a) There is no forking if the link delay is less than the round length, (b) half of the blocks are in a forked chain if the link delay exceeds the round length.

of its F_v and W_v values. We treat F_v and W_v as separate performance metrics in the topologies that we consider.

3.3 Objective

Blockchain networks use a decentralized algorithm for constructing the p2p network. We view topology construction as a game between the n miners, and study simple classes of topologies that the miners can construct to maximize their rewards and/or minimize wasted cost.

We assume each miner $v \in V$ is located at a fixed location within a wide area. The location of the miners induces an underlying latency $l_{(u,v)}$ between every pair of miners u and v. If u and v have a link between them, this is the latency of sending a block from u to v (or vice-versa) over this link. If u and v do not have a link between them, this is the latency with which a block sent from u to v (or vice-versa) would propagate had there been a link between u and v. We assume a miner u has knowledge about $l_{(u,v)}$ for all $v \in V$, but does not know about the precise locations of the other nodes. We provide two control knobs to miners that they can use to select their neighbors:

(1) *Degree.* A miner v specifies a degree d_v which allows v to make d_v outgoing connections;
(2) *Choice of neighbors.* A miner v specifies the set of d_v miners to which to connect to.

We assume each miner knows the IP address of all other miners in the network.

4 ANALYSIS

4.1 One Cluster

As a warm up, we first consider the simple case where the latency graph L is such that $\delta_{(u,v)} = \epsilon$ for all $(u,v) \in \tilde{E}$ where $\epsilon > 0$. In other words, each miner is equidistant from all other miners. In this paper, we do not provide a systematic analysis of all the different network topologies G that can result in a given latency graph L. We instead resort to heuristic topology constructions with latency graphs that reasonably approximate a desired latency graph.

We consider two cases depending on the value of ϵ: (1) $0 < \epsilon \leq 1$ and (2) $1 < \epsilon < 2$. We do not consider the case where $\epsilon > 2$ as it is unlikely for a blockchain system to be configured such that the average interval between successive blocks is significantly less than the time it takes for a block to propagate to a majority fraction (e.g., 90 percent) of the network.

Case 1: If $\epsilon \leq 1$ then when a block is mined in a round, it is received by all other miners in the network before the start of the next round. Therefore any block that is mined at the next round, is going to be mined on top of the block mined in the previous round. This implies there would be no forks in the blockchain at any time, and a unique longest chain would grow at each miner as rounds progress. Figure 1(a) illustrates this process.

Case 2: Next, consider the case when $1 < \epsilon < 2$. In this case, a block mined during a round $r \in \mathbb{N}$ will not be received by any other miner in time when the next round $r + 1$ begins. Whereas all blocks mined up until round $r - 1$ will be received by all the miners before the start of round $r + 1$. If $r + 1$ is an even number, then a block mined at round $r + 1$ will be mined on top of block $r - 2$. Whereas if $r + 1$ is an odd number, then block $r + 1$ will be mined on top of block $r - 1$. This is because for an even $r + 1$ blocks $r - 2$ and $r - 1$ are both at the same height in the blockchain. However, block $r - 2$ reaches the miner of round $r + 1$ before block $r - 1$. Therefore block $r + 1$ will be mined on top of block $r - 2$. For an odd $r + 1$, block $r - 1$ will be at a height that is strictly greater than the height of block $r - 2$. Hence block $r + 1$ will be mined over block $r - 1$ in this case. Figure 1(b) illustrates how the blockchain evolves when $1 < \epsilon < 2$. Thus, at each height of the blockchain there is a fork and only half of all blocks mined are included in the longest chain.

The two cases outlined above motivate the following theorem. We omit the proof as it is straightforward.

Theorem 1. *For a single cluster latency graph L with inter-miner latency ϵ, we have*
(i) if $0 < \epsilon \leq 1$, then for any v we have $\mathbb{E}[F_v] = 1/n$ and $\mathbb{E}[W_v] = 0$;
(ii) if $1 < \epsilon < 2$, then for any v we have $\mathbb{E}[F_v] = 1/n$ and $\mathbb{E}[W_v] = 1/2$.

Implication. It is unlikely for topologies in real world blockchain networks to induce a latency graph L with the same shortest-path latencies $\delta_{(u,v)}$ between every pair of miners u and v. However, the result in Theorem 1 also holds if the shortest path latencies are not all the same, but within the same range of values: if $0 < \delta_{(u,v)} < 1$ for all u, v then claim (i) in the Theorem holds, and if $1 < \delta_{(u,v)} < 2$ for all u, v then claim (ii) holds. A random topology is likely to produce shortest path latencies that are roughly similar for all miners, if the miners are all concentrated within a single geographic region (e.g., same continent or country). The magnitude of the latency values can be controlled by varying the number of neighbors (i.e., the degree) that a miner chooses to have. A large degree ensures that latencies are small, and vice-versa. If the cost of mining blocks is negligible compared to rewards earned, Theorem 1 says that it does not matter what degree miners choose (so long as the network is connected). Whereas if the reward depends on the number of blocks mined, then a higher degree is desirable as it would result in a smaller latency value and minimize forking.

4.2 Two Clusters

Next, we consider the case where there are two clusters of miners, where miners in each cluster have low-latency paths to each other but a high latency to miners of the other cluster. This can happen, e.g., if within a cluster miners are densely connected, whereas links between the clusters are sparse. To model this, we suppose the latency graph L comprises of two sets V_1 and V_2 of miners with $|V_1| = pn$ and $|V_2| = (1-p)n$ where $0.5 < p < 1$. The latency between any two miners $u, v \in V_1$ is $\delta_{(u,v)} = \epsilon$ where $0 < \epsilon < 1$. Similarly, the latency between any two miners $u, v \in V_2$ is also $\delta_{(u,v)} = \epsilon$ where $0 < \epsilon < 1$. However, the latency between a miner $u \in V_1$ and a miner $v \in V_2$ is $\delta_{(u,v)} = \Delta$, where $1 < \Delta < 2$. We assume $\Delta - 1 > \epsilon$, to model the case where the inter-cluster miner latencies are significantly larger than the intra-cluster miner latencies. This assumption implies if block i is mined by cluster 1 and block $i + 1$ is mined by cluster 2, then the $(i + 1)$-th block is propagated to all miners in cluster 2 before they receive the i-th block (and vice-versa if the i-th block is mined by cluster 2 and the $(i + 1)$-th block is mined by cluster 1).

We analyze this case by first observing that there are distinct *phases* that occur in any sample path of blockchain evolution. To illustrate this, consider the example shown in Figure 2 where the sequence of clusters that mine blocks in each round are as follows: 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, i.e., blocks 1–4 are mined by cluster 1, block 5–7 are mined by cluster 2 etc. For this example, the blockchain evolves as shown in Figure 2. The different phases in this evolution are as follows.

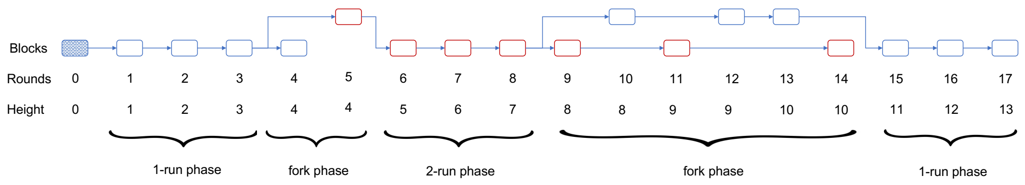


Fig. 2. Evolution of the blockchain when there are two clusters. Blocks in blue are mined by cluster 1 miners, while blocks in red are mined by cluster 2 miners. The three types of phases (1-run, 2-run and fork) that occur during the evolution are also marked.

For the first three rounds, there are no forks. This is because the block mined in round 1 by a miner in cluster 1 reaches the miner of second block within ϵ time units, allowing the second miner to build its block on top of the first block. Similarly the third block is built on top of the second block. Each of these three blocks is part of the longest chain. We call such a sequence of contiguous blocks that are all mined by miners within cluster 1 as a *1-run* phase. Similarly, if there is a continuous sequence of blocks that are all mined by miners of cluster 2 (e.g., blocks 6, 7, 8), we call those blocks as belonging to a *2-run* phase. It can be seen that blocks mined during a 2-run phase also belong to the longest chain.

Next, consider a sequence such as 2, 1, 2, 1, 1, 2 where every two consecutive blocks is mined by an opposing pair of clusters such as 2, 1 or 1, 2. We call such a phase as a *fork phase*. From Figure 2, we see that during a fork phase the blockchain has two competing forks that grow in parallel. We formalize this observation in the following lemma.

Lemma 1. *For a two cluster network with a latency graph as discussed in this section (§4.2), during a fork phase there are two competing forks with miners in each cluster growing a distinct fork.*

Proof. Consider a fork phase sequence $c_1, c_2, \ldots, c_{2i}$, where c_j for $j \in \{1, \ldots, 2i\}$ denotes the cluster that mined the j-th block in the fork phase sequence. Let $b_1, b_2, \ldots, b_{2i}$ be the sequence of blocks that are mined during this phase. Supposing the fork phase started right at round 1, meaning the first block b_1 mined by c_1 is mined on top of the genesis block. At the beginning of the second round, none of the miners in cluster c_2 would have received b_1. Therefore b_2 is mined on top of the genesis block as well. Thus there are two forks at the end of two rounds, one due to each cluster. Similarly during the third round, if $c_3 = c_1$ then block b_3 is going to be mined on top of b_1, whereas if $c_3 = c_2$ then block b_3 is going to be mined on top of b_2. This is because if $c_3 = c_1$ then the miner of b_3 would have received b_1 but not b_2 at the beginning of the third round. Since the fork ending with b_1 is the longest fork known the miner of b_3, it mines the block b_3 on top of b_1. Analogously, if $c_3 = c_2$ then the miner of b_3 receives b_2 before b_1. Since both forks are at equal height but b_2 was received earlier than b_1, the miner of block b_3 mines b_3 on top of b_2. We can similarly argue that if $c_4 = c_2$ then b_4 is mined on top of b_2, and if $c_4 = c_1$ then b_4 is mined on top of b_1. Continuing this reasoning, by induction we can see that throughout the duration of the fork phase there are two forks (one per cluster) that increases in height in parallel.

If the fork phase did not start right at round 1, let c_0 be the cluster that mined the block b_0 in the round before the fork phase started and let b_{-1} be the block mined in the round before b_0 was mined (it is possible b_{-1} is the genesis block). Clearly $c_0 \neq c_1$, since this would mean c_0 is also in the fork phase. Therefore $c_0 = c_1$ and block b_1 is mined on top of block b_0. Moreover, the height of b_{-1} is strictly less than the height of b_0. For otherwise b_{-1}, b_0 would also be in the fork phase. Therefore b_2 is also mined on top of b_0. From here on, we continue the same argument as in the case where the fork phase starts at round 1 to show the lemma. □

A fork phase ends when there are two consecutive blocks mined by the same cluster. E.g., if $c_1, c_2, \ldots, c_{2i}$ is a sequence of clusters that mined blocks during a fork phase and if c_{2i+1}, c_{2i+2} are the clusters that mined the two blocks following the fork phase, then if $c_{2i+1} = c_{2i+2}$ the fork phase ends. When a fork phase ends, among the two competing forks during the fork phase one of them "wins out" and become part of the longest chain. If $c_{2i+1} = c_{2i+2} = 1$ then the fork of cluster 1 wins out, whereas if $c_{2i+1} = c_{2i+2} = 2$ the the cluster 2's fork wins out (see Figure 2). Thus when a fork wins out at the end of a fork phase, all blocks mined by the winning cluster become part of the longest chain. Note that is is possible for a new fork phase to begin starting with the second round after a fork phase ends (i.e., from the round of c_{2i+2}).

To compute $\mathbb{E}[F_v]$ and $\mathbb{E}[W_v]$ for a miner v in this two cluster setting, for any sequence of blocks mined $b_1, b_2, \ldots$ in each round starting from the genesis block, let $P_1, P_2, \ldots$ denote the sequence of phases that occur across the rounds. E.g., in Figure 2, P_1 is a 1-run phase that lasts during rounds $1, 2, 3$; P_2 is a fork phase that lasts during rounds $4, 5$; P_3 is a 2-run phase that lasts during rounds $6, 7, 8$; P_4 is a fork phase in rounds $9, 10, 11, 12, 13, 14$; P_5 is a 1-run phase during round $15, 16, 17$. For a miner $v \in V$, let P_i^v be the number of blocks mined by v during phase P_i that is included in the longest chain. Similarly, let P_i^c be the number of blocks mined by cluster $c \in \{1, 2\}$ during phase P_i that is included in the longest chain. Supposing the network runs for k phases. Then, the expected number of blocks M_c mined by cluster c that are in the longest chain over the k phases is

$$M_c = \sum_{i=1}^{k} P_i^c \Rightarrow \mathbb{E}[M_c] = \sum_{i=1}^{k} \mathbb{E}[P_i^c]. \tag{3}$$

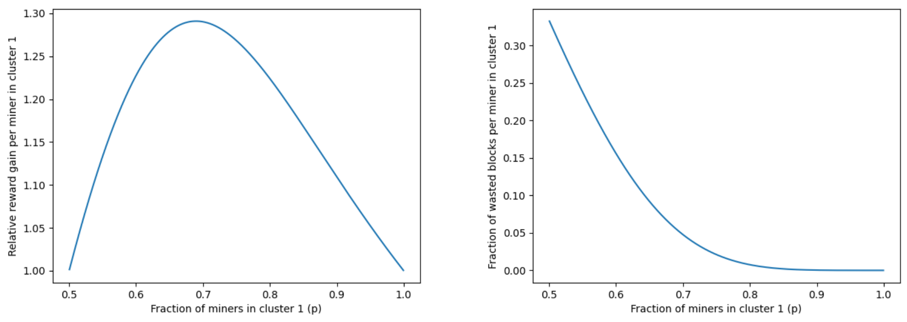
Since the phases are identically distributed, it suffices to compute $\mathbb{E}[P_1^c]$ to evaluate $\mathbb{E}[M_c]$ in Equation (3). Using $\mathbb{E}[M_c]$ we can compute $\mathbb{E}[F_v]$ as shown below.

Theorem 2. *For a two cluster latency graph L discussed in this section (§4.2) with $|V_1| = p|V|$ and $|V_2| = (1-p)|V|$, we have* $\mathbb{E}[F_v] =$

$$\frac{1}{n}\left[\frac{p}{1-p} + \frac{2p^2(1-p)}{(1-2p(1-p))^2}\right] \Big/ \left(\left[\frac{p^2}{1-p} + \frac{2p^3(1-p)}{(1-2p(1-p))^2}\right] + \left[\frac{(1-p)^2}{p} + \frac{2(1-p)^3 p}{(1-2p(1-p))^2}\right]\right) \tag{4}$$

for any $v \in V_1$.

Proof. We consider three possible cases:
(i) P_1 is a 1-run phase. Supposing the phase lasts for $i \geq 1$ rounds. The probability of this happening is $p^{i+1}(1-p)$, since the phase ends only when the $(i+1)$-th block is mined by cluster 1 and the $(i+2)$-th block is mined by cluster 2. All blocks mined by cluster 1 during this phase belong to the longest chain.
(ii) P_1 is a 2-run phase. Again suppose the phase lasts for $i \geq 1$ rounds. The probability of this event happening is $(1-p)^{i+1}p$ by the same argument as before. None of the blocks mined during this phase belong to cluster 1.
(iii) P_1 is a fork phase. Now, suppose the phase lasts for $2i$ rounds for $i \geq 0$. Further suppose that cluster 1 wins this phase, i.e., blocks mined by cluster 1 become part of the longest chain after the phase ends. The probability of this occurring is $(2p(1-p))^i p^2$. Here $2p(1-p)$ is the probability that in every consecutive pair of rounds, one of the blocks is mined by one cluster and the other block is mined by the other cluster. p^2 is the probability that at the end of the fork, two blocks are mined by cluster 1 which ensures that cluster 1's fork wins out after the phase ends. In this case all i blocks mined by cluster 1 during the phase become part of the longest chain. Similarly $(2p(1-p))^i(1-p)^2$ is the probability that cluster 2 wins out at the end of the phase. However, in this case none of the blocks mined by cluster 1 become part of the longest chain.



(a) (b)

Fig. 3. (a) Relative gain over the fair value (i.e., $\mathbb{E}[F_v]/(1/n)$) of a miner v in the dominant cluster with varying cluster size. (b) Percentage of blocks mined by v that are not in the longest chain as a function of cluster size.

From the above three cases, we have

$$\begin{aligned}\mathbb{E}[P_1^1] &= \sum_{i=1}^{\infty} \left(ip^{i+1}(1-p) + i(2p(1-p))^i p^2\right) \\ &= \frac{p^2}{1-p} + \frac{2p^3(1-p)}{(1-2p(1-p))^2} \\ \Rightarrow \mathbb{E}[M_1] &= k\left[\frac{p^2}{1-p} + \frac{2p^3(1-p)}{(1-2p(1-p))^2}\right]. \end{aligned} \tag{5}$$

We can similarly compute the expected number of blocks mined by cluster 2 as

$$\mathbb{E}[M_2] = k\left[\frac{(1-p)^2}{p} + \frac{2(1-p)^3 p}{(1-2p(1-p))^2}\right]. \tag{6}$$

Since each miner in cluster 1 is equally likely to mine a block, by symmetry we have

$$\mathbb{E}[M_v] = \frac{k}{n}\left[\frac{p}{1-p} + \frac{2p^2(1-p)}{(1-2p(1-p))^2}\right] \quad \forall v \in V_1. \tag{7}$$

In the limit of the number of phases approaching infinity, using concentration bounds we can show that the expected percentage of blocks mined by a miner in cluster 1 can be given as

$$\mathbb{E}[F_v] = \frac{\mathbb{E}[M_v]}{\mathbb{E}[M_1] + \mathbb{E}[M_2]} \tag{8}$$

where $\mathbb{E}[M_1], \mathbb{E}[M_2], \mathbb{E}[M_v]$ are as above. □

Thus, the percentage of blocks mined by a miner in V_1 could be significantly different from the fair allocation of $1/n$ in this two cluster case. In Figure 3(a) we plot the ratio between $\mathbb{E}[F_v]$ as in Theorem 2 and the fair allocation of $1/n$ as a function of the cluster size parameter p. We see that the ratio is maximum at 1.29 when the cluster size is roughly 69% of the total number of miners in the network.

Corollary 1. *For a two cluster network with latencies as discussed in this section (§4.2), $\mathbb{E}[F_v]$ for any $v \in V_1$ exceeds the fair fraction of $1/n$ by 29% when $p = 0.69$.*

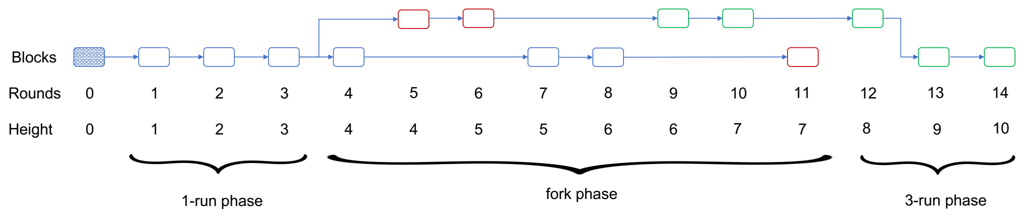


Fig. 4. Evolution of the blockchain when there are three clusters. Blocks mined by each cluster are denoted using a different color. The figure also illustrates the different phases that occur during the evolution.

Implication. Theorem 2 and Corollary 1 imply that it is beneficial for some miners to form connections with others on the network in such a way that there is a well-connected 'dominant' cluster of size approximately 70% of the overall network size. The per-miner gain diminishes if the size of the dominant cluster either increases or decreases from this optimal value. For miners that have been excluded from this dominant cluster, we show in Section 4.3 that being well-connected with each other is beneficial even if they are not able to achieve a low latency path to the dominant cluster. How miners can coordinate to form such clusters in a fully distributed way is beyond the scope of this work, and may be considered in future works.

We now present the fraction of blocks mined by a miner that are included in the longest chain.

Theorem 3. *For a two-cluster network with latency graph as in this section (§4.2), we have*

$$\mathbb{E}[W_v] = \left(\frac{2(1-p)^3}{(1-2p(1-p))^2}\right) \Big/ \left(\frac{p}{1-p} + \frac{2p^2(1-p)}{(1-2p(1-p))^2} + \frac{2(1-p)^3}{(1-2p(1-p))^2}\right). \tag{9}$$

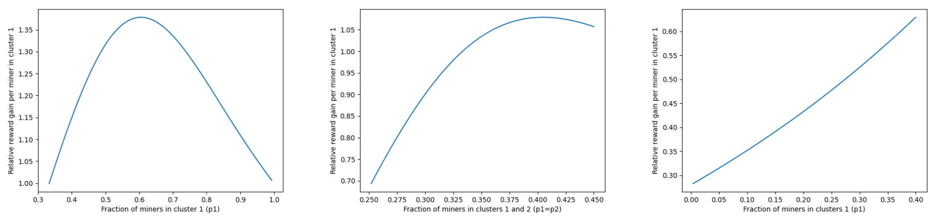
(Proof in Appendix A).

Implication. Figure 3(b) plots $\mathbb{E}[W_v]$ for a miner $v \in V_1$ as a function of p. Unlike $\mathbb{E}[F_v]$, we see that $\mathbb{E}[W_v]$ is monotonically decreasing with p. When $p = 0.7$ the wastage fraction is already less than 5 %. Thus, depending on the relative magnitude of the rewards obtained per mined block included in the longest chain and cost incurred per mined block, miners may be incentivized to form dominant clusters that are only a fraction of size of the total network size.

4.3 Three Clusters

Next, we consider the case when there are three clusters in the network V_1, V_2, V_3 with $|V_1| = p_1 n, |V_2| = p_2 n, |V_3| = p_3 n$ for $p_1 \geq 0, p_2 \geq 0, p_3 \geq 0$ with $p_1 + p_2 + p_3 = 1$. We assume the latency between two miners within the same cluster is ϵ with $0 < \epsilon < 1$ while the latency between two miners in different clusters is Δ with $1 + \epsilon < \Delta < 2$. Consider an example where the cluster to mine a block during each round is given by the sequence $1, 1, 1, 1, 2, 2, 1, 1, 3, 3, 2, 3, 3, 3$ as shown in Figure 4. As in the two cluster case, we observe distinct phases that occur in the sequence. During rounds 1 to 3 we have a 1-run phase, where blocks are mined by cluster 1 and all blocks belong to the longest chain. During rounds 4 to 11 we have a fork phase, where there are two competing chains that are growing in parallel. From round 12 onwards we have a 3-run phase where blocks are mined by cluster 2.

Notice that the number of forks occurring during a fork phase is still two, as in the two cluster case. This is due to our assumption that $\Delta < 2$, which guarantees each miner to hear about at least the last but one block at the beginning of a round. While the 1-run, 2-run and 3-run phases are analogous to the two cluster case, the fork phase in this case behaves differently than before. Specifically, when a fork phase ends it is no longer true that a single cluster emerges as the winner of the phase. Indeed, in Figure 4 we see that the winning fork includes blocks mined by all three



(a) (b) (c)

Fig. 5. (a) Relative gain of a miner v in the dominant cluster, when the other two clusters are of equal size. (b) Relative gain of a miner v in cluster 1, when clusters 1 and 2 are of the same size. (c) Relative gain of a miner v in cluster 1, when cluster 3's size is fixed at 60% of the network size.

clusters. This makes the analysis more involved compared to the two cluster case. We defer the proof to Appendix B of the following theorem.

Theorem 4. *For a three cluster network with latency graph as described in this section, we have*

$$\mathbb{E}[F_v] = \frac{\mathbb{E}[M_1]}{p_1(\mathbb{E}[M_1] + \mathbb{E}[M_2] + \mathbb{E}[M_3])}, \tag{10}$$

for any $v \in V_1$ where $\mathbb{E}[M_1], \mathbb{E}[M_2], \mathbb{E}[M_3]$ are as given in Appendix B.

Implication. Figure 5 presents how $\mathbb{E}[F_v]$ varies under different values for parameters p_1, p_2, p_3 for a miner v in the first cluster. First, we consider the case where cluster 1 is the unique dominant cluster, while cluster 2 and 3 are equal sized smaller clusters. Figure 5(a) plots the percentage of blocks mined by a miner in cluster 1 as its size varies between 33% to 100% of the network size. The optimum gain for miner v occurs when the size of its cluster is around 60% of the network size.

Next, we consider a setting where there are two dominant clusters, cluster 1 and 2, both of the same size. Figure 5(b) plots the rewards for miner v in this scenario when the dominant cluster size varies between 25% to 45% of network size. We see that the relative gain increases beyond 1 only when the dominant cluster size is greater than 33% of the network size.

In the third plot, we consider a scenario where there is a single dominant cluster of size equal to 60% of entire network. We then vary the size of cluster 1 between 0 and 40% as shown in Figure 5(c). We observe that the larger the size of cluster 1, the better it is for miner v. But in all cases, the relative gain stays below 1.

5 EVALUATION

In this section we present experimental evaluation to highlight reward bias due to clustering discussed in Section 4, in real-world settings. We then present candidate protocols that miners can adopt in order to alleviate the bias and increase their profits.

5.1 Simulation Model

We conduct our experiments on OMNeT++ 5.6.2, an event driven simulator [40], by implementing a detailed model based on the Bitcoin protocol. There are 246 miners in our simulation, with each miner mining blocks, broadcasting messages and maintaining a local replica of the blockchain in a completely decentralized fashion. Note that this model we have used in the simulations, is significantly different from the simple analytical model discussed in Section 3.

Mining. We apply Bitcoin's mining and broadcasting strategy in our simulation. At the start of the simulation, all the nodes mine the block on top of the genesis block. To simulate mining, each miner has an independent random timer that expires according to an exponential distribution with a default mean of 15 seconds. We use the same rate for all miners, to model uniform hash power across all miners. When a miner's timer expires, it generates a block on top of the block with the greatest height on its blockchain, and broadcasts it immediately to its connected neighbors.[1] When a miner is waiting for its timer to expire, if it receives a new block that extends its current longest chain, it updates it local blockchain and resets the timer to mine on top of the new received block.

Block broadcast. Miners receive and generate messages of different types, each of which causes the state of the miner to be updated as follows.

* A block message has an identifier and a reference to the identifier of the block's parent. We do not include any transactions in a block message; nor do we model bandwidth limitations in the network. When a miner receives a block message, it first validates the block by verifying it points to a valid parent block in its blockchain. If the received block is valid and extends the longest chain in the blockchain by one, the miner adds the block in to its blockchain. It then resets its exponential timer to mine a new block over the latest received block. We set a duration of 1 ms at each miner for this validation operation. Upon validation, the miner immediately sends out the block message to all of its neighbors except the one it received it from.
* A miner may also receive a block message, whose parent block is unavailable at the miner's local blockchain replica. In this case, the miner sends a 'require' message back to the sender of the block requesting to also send the parent block. The received block is meanwhile stored within an orphan block list at the miner. We set a duration of 1 ms at each miner for this operation as before. When a miner receives a require message, it sends back the requested block as a 'response' message. In case the response block message received to the require message is also such that its parent block is unavailable, a second require message is sent. This process continues until all blocks from the genesis to the first received block are available.
* If a miner receives a block that is not part of the longest chain, it adds the block in to the blockchain, and does not relay the block to any of the neighbors.

Reward. Miners earn rewards by mining blocks that get included in the longest chain. We measure reward by computing the percentage fraction of blocks mined by a miner that is included in the longest chain, i.e., the F_v values for each miner v as defined in Equation (1). While it is likely that the blockchain replicas at different miners are not completely consistent with each other, it is typically the case that the longest chain except for the last few blocks is consistent across miners. So we discard the last 100 blocks in the longest chain, and collect the percentage of valid blocks among the remaining blocks in the longest chain for each miner. Each experiment is run for a sufficient duration that more than 100,000 blocks are present in the longest chain at the end of the simulation. This ensures there are at least a 100 blocks mined per miner within the longest chain. For each network topology, we run 10 independent simulations to gather confidence in the reported results.

Network model. We consider 246 miners distributed across cities around the world. There are 94 miners from Europe, 83 from North America, 37 from Asia, 12 from South America, 11 from Africa and 9 from Australia. This geographical distribution of miners is representative of the Bitcoin network's miner distribution [3] that has a majority of nodes at Europe and North America. The propagation delay of messages between any two miners is set according to the global ping statistics dataset collected on July 19th & 20th, 2020 from WonderNetwork [7]. We take half of the average

[1]Ties are broken as explained in Section 2.

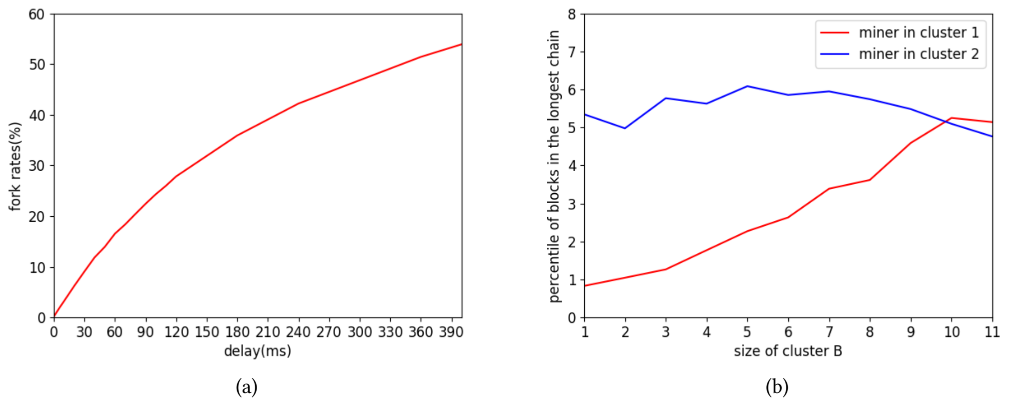


Fig. 6. (a) Percentage of wasted block (i.e., that are not part of the longest chain) on a single cluster with varying link latency. (b) With two clusters, the blue curve shows the percentage of blocks in the longest chain mined by a miner in cluster 1 whereas the red curve shows the percentage of blocks in the longest chain mined by a miner in cluster 2, with varying cluster sizes.

measured round-trip times as the propagation delay between any two miners that are neighbors in a topology. The median link propagation delay in the dataset is 69 ms. We consider all messages to be small in size, and do not model bandwidth at the miners. We have set the delay in processing messages to be a low value of 1 ms, as in reality miners (e.g., in Bitcoin) have a significant compute power available to them. The primary broadcast delay bottleneck in our experiments is therefore the WAN propagation delay.

5.2 Results

5.2.1 Validating the theoretical results of Section 4. We first conduct experiments to validate the generality of our theoretical results, as there are important differences between the theoretical and the Omnet++ model. For example, in Omnet++ blocks are mined by miners using independent exponential timers, and there is no notion of a round like in the theoretical model. We set up a 20-node network on our simulator, where each node has an expected 6 sec mining time per block. Thus, on average one block is mined every $6000/20 = 300$ ms. The network topology is a complete graph with a constant link delay on all links. We vary the link delay from 0 ms to 400ms across experiments.

Figure 6(a) shows the fraction of wasted blocks (i.e., blocks that are not part of the longest chain) as the link latency varies from 0 ms to 400 ms. We see that miners can efficiently communicate their blocks in a low delay network and avoid forking, but the efficiency decreases with a higher delay. This qualitatively captures the essence of Theorem 1, where we showed if link delay is less than the round length there is no forking, whereas if link delay is greater than round length (but less than twice the round length), there is 50% forking. From Figure 6(a) we observe that, indeed when link latency starts to exceed the average block arrival interval (300 ms) the percentage of wasted blocks is around 50%.

Next, we evaluate a 2-cluster network. With the same 20 miners, we split them into 2 clusters, cluster A and B. Miners in the same cluster have a 30ms delay between each other, but have a 360

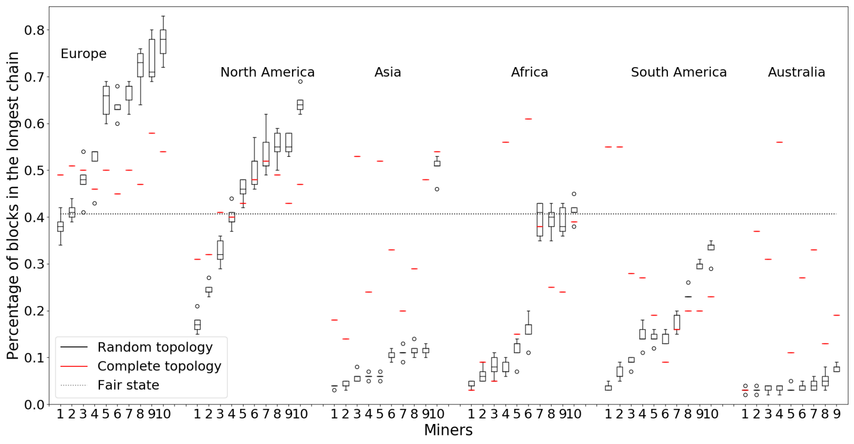


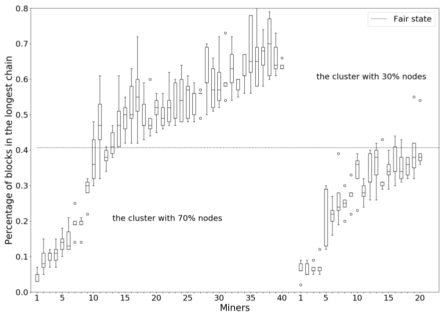
Fig. 7. Fraction of blocks in longest chain as mined by miners in different continents under random and complete graph topologies. The confidence intervals are omitted for the complete graph results for clarity.

ms delay to nodes in the other cluster. We adjust the size of cluster B to increase from 1 until 11, while the size of cluster A decreases from 19 to 9.

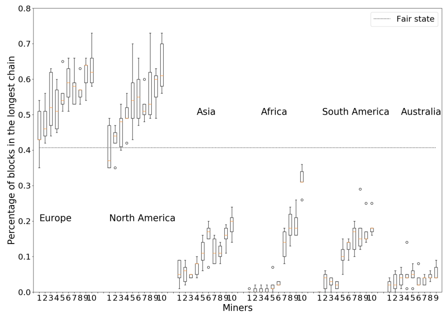
In Figure 6(b) we pick 2 arbitrary miners A_i and B_i. Miner A_i is chosen such that it is always in cluster A during our experiments, and miner B_i is chosen such that it is always in cluster B. We record the percentage of valid blocks mined by A_i, B_i in the longest chain to learn about their rewards with changing size of the cluster they belong to. In a fair network, we expect each of A_i and B_i to have $1/20 \times 100 = 5\%$ of their blocks in the longest chain. Miner A_i starts around 5% in a 19-node network where the cluster A occupies most of the longest chain blocks and distributes it to its 19 miners. Miner A_i gets the most percentage around size 14 after which the reward decreases back to 5% when the size of cluster A matches that of cluster B with 10 nodes each. Node B_i gets a higher percentage and a greater slope with the increasing size of cluster B. The large size cluster can easily defeat the small size cluster and get a higher percentage of blocks in the longest chain. The larger cluster has an advantage at all sizes, as long as it is larger than the other cluster. But the maximum benefit appears at cluster size 14 in our 20 node network. After that, even if the larger cluster plunders a larger cake there are more miners in the cluster to share the cake.

Thus, the results in the experiments follow the theoretical model's conclusions.

5.2.2 Geographic location-induced clustering of miners. On a world-wide network, the distribution of where miners are located implicitly creates a clustering in the network, even if the topology is chosen in a geography-independent way (e.g., random). So before we provide any optimized topology designs, we first investigate how much benefit/hurt miners receive owing to their geography. For this experiment, we consider the 246 miner network outlined in Section 5.1, with link latencies following the WonderNetwork ping dataset. We consider a random topology and the complete graph topology. Similar to the topology followed by Bitcoin network today, the random topology randomly picks a fixed number of neighbors for each node. Due to the smaller size of our network, we set a (out-)degree of 6 for each miner rather than 8 as followed by Bitcoin [27]. Each miner reaches out to 6 random miners and builds connections to them; it also accepts any incoming



(a)



(b)

Fig. 8. Percentage of blocks in longest chain mined by miners in the dominant and non-dominant clusters under (a) random clustering policy, (b) a geography-based clustering policy.

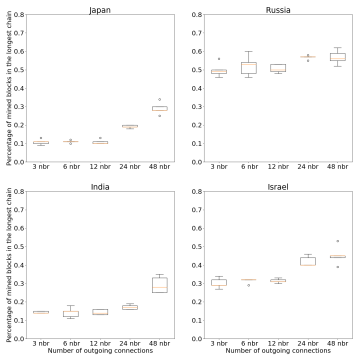
connection requests from other miners. The complete graph topology connects each miner to the remaining 245 miners. Thus each miner has a direct connection to every other miner.

Figure 7 shows the results, where we have plotted the percentage of blocks in the longest chain mined by miners in each of the 6 continents. For brevity, we have randomly chosen 10 miners from each continent and have included only their results in the figure. We see that the Europe and North America are the dominant clusters. In the 246 node network, there is a fair value of around 0.4% (= 100/246) with uniform hash power. However, most of the European and North American miners get a greater percentage than the fair value while miners in other continents get below their fair share in both topologies. In the complete graph topology, Europe and North America get slightly closer to the fair state, but there is still a large difference between them and other regions.

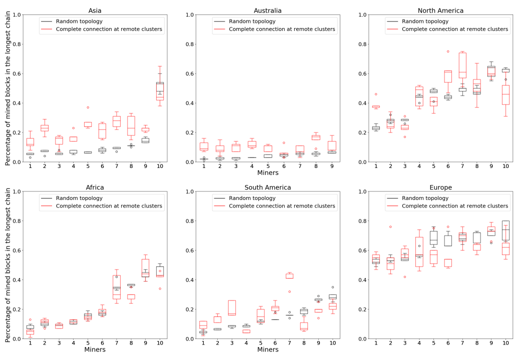
We can explain the difference between Europe, North America and the other regions using the theoretical results of Section 4. As a large cluster with low link delay within the cluster, Europe and North America can get a higher acceptance rate in their mined blocks, as it corresponds to there being a single dominant cluster in the network. Since Europe has 94 miners and North America has 83 miners, together they have 72% of all miners in the network, which we have seen is also a good size to get the maximum percentage gain for every miner in the cluster.

5.2.3 Manually clustering miners into two clusters. Geography-induced clustering under a random topology provides a natural advantage to the European and North American nodes. Next, we consider manually interconnecting miners to form clusters. We first implement this manual clustering policy by randomly splitting the nodes into two groups, containing 70% and 30% of miners respectively. The first group is the dominant cluster, while the second group is the non-dominant cluster. To mimic the theoretical 2-cluster latency graph model, we would like miners in a cluster to have low latency between them and miners across clusters to have a high latency. We therefore connect each miner to every other miner within its own cluster. Whereas, we leave only 20 connections overall between the 2 clusters.

Figure 8(a) shows that the results, where we only plot a random subset of miners from each group for brevity as before. We observe that miners from the large cluster get a greater percentage of blocks because of being part of the larger cluster. Whereas, even European and North American miners have a hard time reaching the fair state if they are in the smaller cluster.



(a)



(b)

Fig. 9. (a) Increasing the degree of miners in Asia strictly benefits them. (b) Tightly interconnecting miners in non-central regions strictly benefits most of the non-central miners.

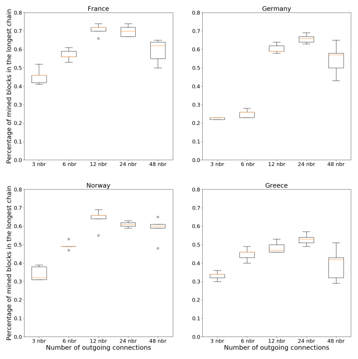
Next, we consider another topology where we take Europe and North America together as the large cluster and the remaining continents as the small cluster. As before, within a cluster we connect each miner to all other miners in the cluster, and leave only 20 connections between clusters. In Figure 8(b), the percentage distributions are narrower compared with the random topology in Figure 7. Clustering by regions gives a more stable performance in each continent.

5.2.4 How to alleviate bias due to clustering. In light of the reward biases due to clustering that has been highlighted thus far, a natural question that arises is what can miners negatively affected by the clustering do to lessen the impact of clustering on their payoff. From the results of Section 4, two potential solutions arise.

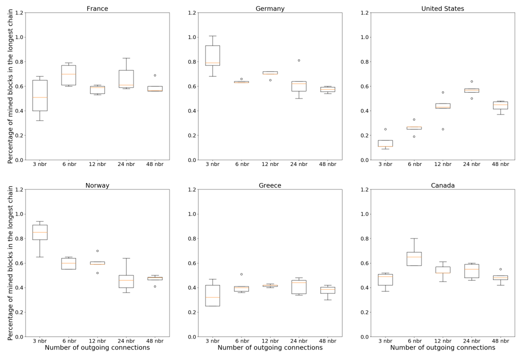
- First, if the miner has knowledge about the existence of a dominant cluster in the network, it should attempt to connect to as many miners within the cluster as possible and become a part of it.
- When this is not possible, if the miner has knowledge about other miners that are not included in the dominant cluster it should attempt to connect to as many of those miners as possible to form as large a non-dominant cluster in the network as possible.

To test these solutions, we conduct experiments where we adjust the neighbors of miners in non-central locations, namely Asia, Africa, Australia and South America to see if their reward improves. We first focus on miners only in Asia. Due to the relatively small number of Asian miners, and relatively large propagation delays to other miners, most Asian nodes get lower than their fair share in the previous experiments. By keeping the rest of the network topology fixed (each miner makes 6 random outgoing connections), we adjust each Asian miner to choose 3, 6, 12, 24 and 48 neighbors across different experimental runs.

Figure 9(a) plots the percentage rewards obtained by 4 randomly selected miners in Asia under this experiment. With increasing number of neighbor connections, all 4 Asian nodes get a higher reward percentage. Miners in Russia and Israel are closer to Europe so they perform better than the other 2 nodes at first. However, they have a lesser relative performance improvement with 48 outgoing neighbor connections compared with the miners in Japan and India.



(a)



(b)

Fig. 10. (a) Increasing the degree of miners in Europe hurt them at high degree (b) Increasing the degree of miners in Europe and North America.

Next, we consider the policy where we take miners in continents except Europe and North America to form a single tightly-connected non-dominant cluster. With the neighbors for Europe and North American miners unchanged, each miner in other continents choose to connect to every other miner in the other continents. Figure 9(b) plots the performance for 10 randomly picked miners from each continent. Asian and Australian miners get significant benefit compared to the random topology. African nodes are only slightly affected, with performance rather similar to that of the random topology. South American miners exhibit two distinct trends: miners in the north of the continent (closer to North America) which perform well under a random topology see some loss; while miners in the south of the continent get a performance increase. European and North American miners lose a bit in their reward percentages. Thus, the policy of tightly interconnecting miners in non-central locations benefits most of these miners. Miners in centrally located clusters or close to the central cluster bear the corresponding loss.

Next, we verify whether increasing the number of neighbors also benefits miners in the central region of Europe and North America. By keeping the degree at other miners fixed at 6, we increase the number of random connections at European miners to vary as 3, 6, 12, 24 and 48. Figure 10(a) plots the performance for 4 randomly chosen miners in Europe. While the performance increases initially with increasing degree, after about a 12 neighbors any further increase in number of neighbors hurts the miners. Thus, less is more here for the European miners.

Since there are 2 central regions, we also consider a policy where we increase the neighbor degree at both European and North American miners, while keeping the degree of other miners fixed. Figure 10(b) plots performance of 4 randomly picked nodes from Europe as in Figure 10(a) and 2 randomly picked nodes from North America. Different from Figure 9(a), the performance here tends to to drop with more neighbors.

Based on the results in Figure 9 and Figure 10, the ideal strategy for each miner varies. Centrally located miners need to limit their connections, as more neighbors hurt them. Miners in non-central regions need to expand their neighborhood size, as the more the better. There is a conflict between these ideal strategies, because when non-central miners increase their number of connections to central miners, it inadvertently also increases the number of connections from the central miners which hurts the central miners. Therefore central miners are incentivized to reject these extra connections. The best strategy then for the non-central miners is to connect to the miners that are willing to accept their connection requests. Ultimately, this creates a situation where non-central miners have a large number of interconnection links among themselves, while central nodes enforce a strict control of their connections.

6 RELATED WORK

The effect of network delay on mining rewards in blockchains has been studied in prior works [39]. E.g., Cao et al. [15] show that with a limited degree bound, high hash power mining pools are incentivized to connect with other hash power mining pools to reduce block propagation delay and increase their mining rewards. Gencer et al. [21] highlight that the top miners are generally more successful in including blocks in the main chain, on Bitcoin and Ethereum networks. In Putri et al. [34], the authors look at the effect of network latency on miners executing a selfish-mining attack [20]. These works do not consider the effect increasing connectivity of a miner has on the connectivity of other miners.

Enabling a faster broadcasting of blocks for all the miners, even if it does not necessarily result in strict reward improvements for some (or all) miners, allows blocks to be mined faster thereby increasing the transaction confirmation throughput of the system. Thus a number of prior works have proposed techniques to accelerate block dissemination. Relay networks such as Falcon [10], Fibre [11], bloXroute [23] use a network of dedicated servers to transport blocks quickly between distant geographical regions instead of relying on p2p gossip. Kadcast [35] proposes a structured p2p overlay with UDP and forward error correction to achieve a more efficient broadcast compared to Bitcoin's random topology with TCP links. Perigee [26] presents an algorithm to explore and exploit neighbors to search for the best set of neighbors to connect to, for optimizing broadcasting delay. Other innovations, e.g., Bitcoin's compact block relay [17] also help to reduce broadcast time by reducing the size of each block. Nagayama et al. [29] study how progress in Internet speeds in recent years have led to reductions in broadcast time in Bitcoin.

Another line of work considers how miners can deviate from a prescribed mining protocol to selfishly garner benefit [13, 37]. Eyal et al. [20] present a selfish mining attack, wherein a pool of colluding miners withhold mined blocks and reveal them at a later point in time, thus deceiving other miners into performing the futile task of mining blocks over a forked chain. Empirical analysis of the Bitcoin blockchain shows a large frequency of short time intervals between consecutive blocks, implying selfish mining may be happening in practice [31]. Lewenberg et al. [25] uses game theoretic tools to analyze how members of a mining pool may share rewards between them. They show under certain cases participants are incentivized to keep switching between pools.

7 CONCLUSION

We have considered the problem of topology design for maximizing miner rewards in wide-area PoW blockchain networks. A miner may decrease its latency to other miners by carefully choosing its neighbors on the overlay, and/or by increasing the number of neighbors. Contrary to general understanding that achieving a lower network latency to other miners is strictly beneficial to miners, we have shown that a miner receives the most benefit if it is well-connected to a dominant cluster of roughly 70% of miners in the network, and *not* well-connected to the remaining miners. Designing fully decentralized algorithms that can achieve this optimal clustering is an interesting direction for future work. Our result exposes an inherent weakness in PoW blockchains, where miners can always organize themselves in order to selfishly benefit by denying the fair share of rewards to miners of the non-dominant cluster. Beyond network-level strategies, it would be useful to propose protocol changes (e.g., at the consensus layer) that incentivize miners to obtain their fair share of rewards.

REFERENCES

[1] [n. d.]. Average Transactions Per Block. https://www.blockchain.com/charts/n-transactions-per-block.

[2] [n. d.]. Bitcoin Average Transaction Fee. https://ycharts.com/indicators/bitcoin_average_transaction_fee.

[3] [n. d.]. Bitnodes network snapshot. https://bitnodes.earn.com/nodes/.

[4] [n. d.]. Cryptocurrency Market Capitalization. https://coinmarketcap.com/.

[5] [n. d.]. Dogecoin. https://dogecoin.com/.

[6] [n. d.]. Ethereum. https://ethereum.org/en/.

[7] [n. d.]. Global Ping Statistics. https://wondernetwork.com/pings.

[8] [n. d.]. OMNeT++ Discrete Event Simulator. https://omnetpp.org/.

[9] [n. d.]. Transaction Fees Percentage in Block Reward. https://btc.com/stats/fee.

[10] 2020. Falcon. https://www.falcon-net.org/.

[11] 2020. FIBRE. https://bitcoinfibre.org/.

[12] Joe Abou Jaoude and Raafat George Saade. 2019. Blockchain applications–usage in different domains. *IEEE Access* 7 (2019), 45360–45381.

[13] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. 2016. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security* 12, 8 (2016), 1967–1978.

[14] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. 2019. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 585–602.

[15] Tong Cao, Jérémie Decouchant, Jiangshan Yu, and Paulo Esteves-Verissimo. 2021. Characterizing the Impact of Network Delay on Bitcoin Mining. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 109–119.

[16] Nakul Chawla, Hans Walter Behrens, Darren Tapp, Dragan Boscovic, and K Selçuk Candan. 2019. Velocity: Scalability improvements in block propagation through rateless erasure coding. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 447–454.

[17] Matt Corallo. 2017. Compact block relay. BIP 152.

[18] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. 2016. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*. Springer, 106–125.

[19] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*. IEEE, 1–10.

[20] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*. Springer, 436–454.

[21] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. 2018. Decentralization in bitcoin and ethereum networks. In *International Conference on Financial Cryptography and Data Security*. Springer, 439–457.

[22] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principle*. 51–68.

[23] Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. 2018. bloxroute: A scalable trustless blockchain distribution network whitepaper. *IEEE Internet of Things Journal* (2018).

[24] Joshua A Kroll, Ian C Davey, and Edward W Felten. 2013. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Proceedings of WEIS*, Vol. 2013. 11.

[25] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. 2015. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*. Citeseer, 919–927.

[26] Yifan Mao, Soubhik Deb, Shaileshh Bojja Venkatakrishnan, Sreeram Kannan, and Kannan Srinivasan. 2020. Perigee: Efficient peer-to-peer network design for blockchains. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. 428–437.

[27] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2015. Discovering bitcoin?s public topology and influential nodes. *et al* (2015).

[28] Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and Richard Brooks. 2016. A brief survey of cryptocurrency systems. In *2016 14th annual conference on privacy, security and trust (PST)*. IEEE, 745–752.

[29] Ryunosuke Nagayama, Ryohei Banno, and Kazuyuki Shudo. 2020. Identifying impacts of protocol and internet development on the bitcoin network. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–6.

[30] Satoshi Nakamoto. 2008. A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.

[31] Till Neudecker and Hannes Hartenstein. 2019. Short paper: An empirical analysis of blockchain forks in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 84–92.

[32] A Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, and Brian Levine. 2017. Graphene: A new protocol for block propagation using set reconciliation. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 420–428.

[33] Sehyun Park, Seongwon Im, Youhwan Seol, and Jeongyeup Paek. 2019. Nodes in the bitcoin network: Comparative measurement study and survey. *IEEE Access* 7 (2019), 57009–57022.

[34] Bellia Dwi Cahya Putri and Riri Fitri Sari. 2018. The effect of latency on selfish-miner attack on block receive time bitcoin network using NS3. In *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*. IEEE, 1–5.

[35] Elias Rohrer and Florian Tschorsch. 2019. Kadcast: A Structured Approach to Broadcast in Blockchain Networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 199–213.

[36] Meni Rosenfeld. 2014. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009* (2014).

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

[38] Markus Schäffer, Monika Di Angelo, and Gernot Salzer. 2019. Performance and scalability of private Ethereum blockchains. In *International Conference on Business Process Management*. Springer, 103–118.

[39] Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. 2019. A theoretical model for fork analysis in the bitcoin network. In *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 237–244.

[40] András Varga and Rudolf Hornig. 2008. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. 1–10.

[41] Luming Wan, David Eyers, and Haibo Zhang. 2019. Evaluating the impact of network latency on the safety of blockchain transactions. In *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 194–201.

A PROOF OF THEOREM 3

Proof. The proof follows a similar argument as in the proof of Theorem 2. As before let $P_1, P_2, \ldots, P_k$ be a sequence of k-phases starting from the genesis block. The expected number of blocks mined by a miner $v \in V_1$ during these k phases is given by Equation (7). Let $\tilde{P}_1^c$ be the expected total number of blocks mined by cluster c in phase 1. During a 1-run phase all blocks mined belong to cluster 1. During a 0-run phase none of the blocks mined belong to cluster 1. During a fork phase, half of all blocks mined belong to cluster 1. Therefore,

$$\mathbb{E}[\tilde{P}_1^1] = \sum_{i=1}^{\infty} \left(ip^{i+1}(1-p) + i(2p(1-p))^i p^2 + i(2p(1-p))^i (1-p)^2 \right) \tag{11}$$

$$= \frac{p^2}{1-p} + \frac{2p^3(1-p)}{(1-2p(1-p))^2} + \frac{2p(1-p)^3}{(1-2p(1-p))^2}. \tag{12}$$

Hence the expected total number of blocks mined by a miner $v \in V_1$ across k phases is

$$\mathbb{E}[\tilde{M}_v] = \frac{k}{n}\left[\frac{p}{1-p} + \frac{2p^2(1-p)}{(1-2p(1-p))^2} + \frac{2(1-p)^3}{(1-2p(1-p))^2} \right]. \tag{13}$$

Combining Equation (13) with Equation (7), the Theorem follows. □

B PROOF OF THEOREM 4

As in the two cluster case, let $P_1, P_2, \ldots, P_k$ be k phases starting from the genesis block. It suffices to analyze the expected number of blocks mined by cluster 1 during the first phase as the phases are identically distributed. Let M_1 be the number of blocks mined by cluster 1 that is included in the longest chain in phase P_1. If P_1 is a 1-run phase, then all blocks mined during the phase belong the longest chain. The contribution of this term in the overall expression for $\mathbb{E}[M_1]$ is

$$\sum_{i=1}^{\infty} p_1^i p_1 (1-p_1) i = \frac{p_1^2}{1-p_1}, \tag{14}$$

by the same reasoning as in the two cluster case.

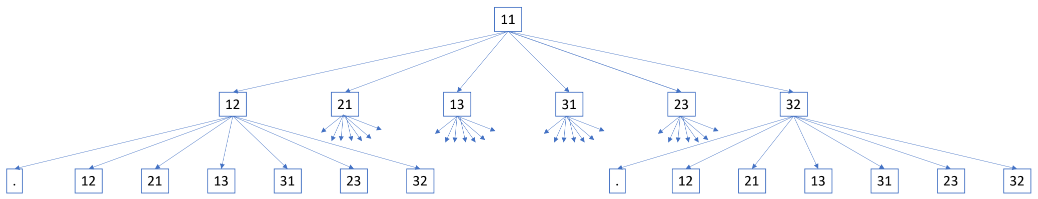


Fig. 11. Possible sample paths that can occur during a fork phase when there are three clusters.

Next, suppose P_1 is a fork phase. Let the blocks mined during this phase be from clusters $c_1, c_2, \dots, c_{2l-1}, c_{2l}$ where c_i denotes the cluster that mined block i in the phase. Let c_{2l+1}, c_{2l+2} be the clusters that mined blocks $2l+1, 2l+2$ after the fork phase ends. We must have $c_{2l+1} = c_{2l+2}$. We observe that the winner among blocks $2l-1, 2l$ is determined by c_{2l+1}, c_{2l+2}. If $c_{2l-1} = c_{2l+1}$, then c_{2l-1} is the winner; else if $c_{2l} = c_{2l+1}$ then c_{2l} is the winner; if $c_{2l-1} \neq c_{2l} \neq c_{2l+1}$ then c_{2l-1} is the winner. Similarly, the winning cluster during round $2l-3, 2l-2$ is determined by the winning cluster during rounds $2l-1, 2l$. Let $c^*_{2l-1,2l}$ be the winning cluster during rounds $2l-1, 2l$. If $c_{2l-3} = c^*_{2l-1,2l}$ then c_{2l-3} is the winner; else if $c_{2l-2} = c^*_{2l-1,2l}$ then c_{2l-2} is the winner; if $c_{2l-3} \neq c_{2l-2} \neq c^*_{2l-1,2l}$ then c_{2l-3} is the winner. By continuing this process we can determine the winning cluster during each consecutive pair of rounds in the fork phase.

To analyze the contribution of the fork phase in the overall expression for $\mathbb{E}[M_1]$, consider Figure 11 which shows the different possible sample paths in the fork phase that ends with two blocks mined by cluster 1. A sample path starts with the leaf node, and goes up the tree until it reaches the root. Each sample path has a certain probability of occurring and has a certain number of blocks mined by cluster 1 that is included in the longest chain. For a node (i, j) in Figure 11 in which cluster i wins, let α^i_{ij} be the average number of blocks mined by cluster 1 that are included in the longest chain in the subtree rooted at that node. Similarly, for a node (i, j) in Figure 11 in which cluster j wins, let α^j_{ij} be the average number of blocks mined by cluster 1 that are included in the longest chain in the subtree rooted at that node. By using the fact that winner of node is dictated by the winner of its parent in the tree, we have following recurrence relations:

$$\alpha^1_{12} = 1 + p_1p_2\alpha^1_{12} + p_2p_1\alpha^1_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^1_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^3_{32} \tag{15}$$

$$\alpha^2_{12} = p_1p_2\alpha^2_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^2_{32} \tag{16}$$

$$\alpha^1_{13} = 1 + p_1p_2\alpha^1_{12} + p_2p_1\alpha^1_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^1_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^3_{32} \tag{17}$$

$$\alpha^3_{13} = p_1p_2\alpha^1_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^3_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^3_{23} + p_3p_2\alpha^3_{32} \tag{18}$$

$$\alpha^2_{23} = p_1p_2\alpha^2_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^2_{32} \tag{19}$$

$$\alpha^3_{23} = p_1p_2\alpha^1_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^3_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^3_{23} + p_3p_2\alpha^3_{32} \tag{20}$$

$$\alpha^2_{21} = p_1p_2\alpha^2_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^2_{32} \tag{21}$$

$$\alpha^1_{21} = 1 + p_1p_2\alpha^1_{12} + p_2p_1\alpha^1_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^1_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^3_{32} \tag{22}$$

$$\alpha^3_{31} = p_1p_2\alpha^1_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^3_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^3_{23} + p_3p_2\alpha^3_{32} \tag{23}$$

$$\alpha^1_{31} = 1 + p_1p_2\alpha^1_{12} + p_2p_1\alpha^1_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^1_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^3_{32} \tag{24}$$

$$\alpha^3_{32} = p_1p_2\alpha^1_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^3_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^3_{23} + p_3p_2\alpha^3_{32} \tag{25}$$

$$\alpha^2_{32} = p_1p_2\alpha^2_{12} + p_2p_1\alpha^2_{21} + p_1p_3\alpha^1_{13} + p_3p_1\alpha^3_{31} + p_2p_3\alpha^2_{23} + p_3p_2\alpha^2_{32}. \tag{26}$$

Once we solve for all the α's in the above system of equations, we can compute the contribution of the fork phase term in $\mathbb{E}[M_1]$ as

$$\begin{aligned} &p_1^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^1 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^1 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^3)+\\ &p_2^2(p_1p_2\alpha_{12}^2 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^2)+\\ &p_3^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^3 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^3 + p_3p_2\alpha_{32}^3), \end{aligned} \tag{27}$$

as a fork phase can end with two blocks that are either both mined by cluster 1, both mined by cluster 2 or both mined by cluster 3. Combining this with Equation (14) we have

$$\begin{aligned} \mathbb{E}[M_1] &= p_1^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^1 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^1 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^3)+\\ &\quad p_2^2(p_1p_2\alpha_{12}^2 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^2)+\\ &\quad p_3^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^3 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^3 + p_3p_2\alpha_{32}^3)+\\ &\quad \sum_{i=1}^{\infty} p_1^i p_1(1-p_1)i \qquad (28)\\ &= p_1^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^1 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^1 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^3)+\\ &\quad p_2^2(p_1p_2\alpha_{12}^2 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^1 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^2 + p_3p_2\alpha_{32}^2)+\\ &\quad p_3^2(p_1p_2\alpha_{12}^1 + p_2p_1\alpha_{21}^2 + p_1p_3\alpha_{13}^3 + p_3p_1\alpha_{31}^3 + p_2p_3\alpha_{23}^3 + p_3p_2\alpha_{32}^3)+\\ &\quad \frac{p_1^2}{1-p_1}. \qquad (29) \end{aligned}$$

The expressions for $\mathbb{E}[M_2]$ and $\mathbb{E}[M_3]$ can be similarly computed as above. We can also simply interchange p_1 with p_2 in the above to compute $\mathbb{E}[M_2]$, and interchange p_1 with p_3 in the above to compute $\mathbb{E}[M_3]$.