Strategic Peer Selection Using Transaction Value and Latency

Kushal Babel*
Cornell University, Jump Crypto
New York, USA

Lucas Baker
Jump Crypto
New York, USA

ABSTRACT

Many blockchains utilize public peer-to-peer networks to communicate transactions. As activity on blockchain-based DeFi protocols has increased, there has been a sharp rise in strategic behaviour from bots and miners, commonly captured by the notion of Maximal Extractable Value (MEV). While many works have focused on MEV arising from the smart contract layer or consensus layer, in this work we study how a strategic agent can maximise realisable MEV through the optimal choice of network peers.

Specifically, we study how existing definitions and algorithms for latency optimization [14, 17] can be augmented with information about the transactions themselves in order to optimize peering algorithms. We formally model this optimization objective for two classes of consensus protocols : 1) time-based ("fair ordering") protocols and 2) single leader-based protocols. We present an efficient local algorithm for choosing peers strategically, and evaluate our algorithm on real world data to show that it outperforms benchmark algorithms that either choose peers randomly or do not exploit information about blockchain transactions.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; • **Networks → Network structure**; *Network algorithms*; *Network simulations.*

KEYWORDS

MEV; P2P networks; latency; blockchains; strategic behaviour

ACM Reference Format:
Kushal Babel and Lucas Baker. 2022. Strategic Peer Selection Using Transaction Value and Latency. In *Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security (DeFi '22), November 11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3560832.3563440

*This work was done during the first author's internship at Jump Crypto.

1 INTRODUCTION

The emergence of blockchain-based financial markets through decentralized finance (DeFi) has created both a new form of protocol-based financial infrastructure and a new class of strategic opportunities that rely on these mechanisms. For blockchains that utilize peer-to-peer (P2P) networks to communicate pending and finalized transactions, many such opportunities can be observed in advance by miners and/or validators, who can then profit from their ability to arbitrarily include, exclude, and reorder transactions. Collectively referred to as MEV (Maximal Extractable Value), this phenomenon was first documented in [8] and then empirically and formally studied in [6, 19, 21]. In addition to miners, sophisticated actors such as bots and arbitrageurs can exploit pending transaction information to extract MEV profits for themselves. The interaction between these actors has given rise to a growing ecosystem, with tools that coordinate MEV activity among bots and miners/validators while attempting to minimize negative externalities. However, this infrastructure also introduces significant centralization and consensus risks to the blockchain itself.

MEV can arise from the application layer (smart contracts), consensus layer (rewards to protocol participants), or network layer. Prior research has primarily studied MEV at the application [6, 10, 20] and consensus [9, 13] levels. However, the ability to compete for MEV profits is also linked to latency in observing pending transactions at the network layer. This information advantage in *transaction propagation* can be exploited by faster participants to extract MEV. Maximizing network-level performance generally relies on centralized or third-party infrastructure such as private node relays, but recent work has shown that it is possible to reduce latency independently, i.e. from the perspective of an individual node, via the Peri algorithm [17]. Peri (inspired by the Perigee algorithm [14]) introduces a local approach to strategic peer selection that significantly improves on random peer selection without requiring global knowledge of network topology.

As formulated, Peri does not incorporate transaction data or metadata in order to optimize the choice of peers. In this work, we initiate the study of how latency information may be combined with blockchain transaction data to inform the selection of peers by a strategic agent. We show that a strategic agent can consistently extract higher MEV by adaptively choosing peers based on both latency and the value of transactions to the agent itself. We also discuss how the class of underlying consensus protocol and the parameters of the protocol, such as number of consensus committee members or block production schedule, impacts the effectiveness of the weighted selection process. Finally, we evaluate our algorithm (MEV-Peri) against baselines of random peering and Peri [17] on simulated latency models of P2P networks and estimated transaction flow from real world Ethereum mainnet activity.

Our contributions:

- Model the objective of maximising MEV-weighted *adversarial advantage* (generalized latency reduction) for (1) time-based ordering protocols and (2) leader-based protocols.
- Introduce an algorithm, MEV-Peri, to maximise a node's expected MEV-weighted advantage from peer selection.
- Develop simulation methods to evaluate our algorithm based on real estimates derived from Ethereum mainnet activity.

Paper overview: In Section 2, we provide a background discussion of P2P structure, the Peri and Perigee algorithms [14, 17] and MEV. We then develop our model in Section 3 along with a discussion on how different protocol parameters affect the peering algorithm and gain from optimization. We discuss our algorithm, MEV-Peri, in Section 4, and evaluate MEV-Peri on different models of P2P networks and consensus protocols in Section 5. In Section 6, we discuss observations, mitigations, and areas for future work.

2 BACKGROUND AND RELATED WORK

2.1 Network Structure and Consensus

P2P Networks. Participants of many permissionless blockchains such as Ethereum are organized as nodes in a peer-to-peer(P2P) network structure. Each node in this P2P network is connected to other nodes ("peers") for receiving and sending information about pending transactions and finalized blocks. A new node joining the network bootstraps itself by connecting to either a hard-coded set of peers or a known set from a previous session. Once connected, a node can *discover* other nodes using Distributed Hash Table protocols such as Kademlia [15] or Chord [16]. Crucially, every node is free to choose its own peering strategy and number of peers. The default peering strategy in most blockchain clients selects for known, stable, and compatible nodes.

Mempool Transactions. Once a transaction is cryptographically signed with a user's public key, it can be broadcast from any source node to the rest of the P2P network through a gossip protocol. Note that while the signed transaction cannot be tempered without detection, most blockchains expose the transaction data (and metadata) in clear such that any network participant can learn all the information about the transaction. Each node maintains a pool of unconfirmed transactions that pass basic validity checks (e.g. sufficient transaction fees), collectively referred to as the *mempool.* For each transaction m in its own pool, a node chooses a small random subset of peers and sends them the digest $H(m)$; recipients then request the full transaction if they do not have the data associated with the digest $H(m)$. Miners (or *validators* in proof-of-stake networks) choose a subset of their own TxPool to form into a proposed *block*, or series of transactions. Which block is accepted depends on the network's chosen *consensus protocol*, which enables distributed participants to agree on a valid block history. The chosen block is then added to the global ledger, and the process repeats.

2.2 MEV

Defining MEV. Because transaction ordering within a block may have significant financial repercussions for participants in that block, there is also value in the miner's ability to choose the ordering and inclusion of transactions. The total profit that a miner could realise through such operations is referred to as Maximal Extractable Value, or MEV [4]. While there are many forms of MEV-linked transactions, two of the most common examples are sandwiching, adding transactions before and after a user transaction to take advantage of price movements, and frontrunning, where a miner imitates a profitable user transaction that precedes or replaces the original [1]. Conservatively, the amount of realised MEV has been estimated at $650 million or more since 2020 [2].

Latency and Consensus. Latency can be introduced either at the network level during propagation or at the node level due to internal computation; we focus on the network level. The impact of latency may vary based on design restrictions around transaction ordering. In practice, there is typically a transaction fee mechanism [12] such that all transactions within a block are ranked in decreasing order of fees paid, unless the miner proposing the block has an incentive to act otherwise. We refer to this mechanism as a *leader-based protocol.* One alternative to the above mechanism is a *time-based ordering protocol*, where transactions are processed according to a first come first serve (FCFS) scheduling rule [9]. It is possible that the properties of time-based ordering protocols make harvesting MEV much more difficult.

Private Relays. In the last couple of years, private trusted relay services and MEV markets such as Flashbots [3] have emerged, which provide a suite of tools for coordination among users, bots, and miners/validators. Two tools are of relevance here: firstly, the service provides an auction mechanism that links miners with bots to facilitate efficient MEV extraction, essentially frontrunning-as-a-service (FaaS). The auction provider acts as an intermediary between bots or *searchers*, which pick up user transactions from the public mempool and submit them along with MEV extracting transactions, and miners or validators, which accept these transaction *bundles* and include them in blocks. Secondly, the service provides a tool allowing users to submit transactions directly to miners/validators in frontrunning-protected bundles. Note that in this work, we are only concerned with transactions that are sent to the public P2P network, and exclude privately submitted transactions.

2.3 Peri and Perigee

Low-latency mempool data availability can be provided via centralized services such as bloXroute, which offer private node relays that can offer meaningful latency advantage to an agent in both receiving other users' mempool transactions and propagating own transactions to the miners/validators. However, it is also possible to decrease latency without use of such services via *strategic peering.* Recent work on this subject aims to minimize *triangular latency*, by adding a node a such that all fastest paths between source nodes s and target nodes t must include a. The strategic peering algorithm Peri [17] (inspired by Perigee [14]) offers a localized strategy that improves significantly on naive peering, and can also be used in concert with private relays such as bloXroute. [17] also shows that the latency optimization problem, in general, is NP-hard.

3 MODEL

3.1 Strategic Behaviour

The strategies available to a given agent can be broadly divided into two categories, (1) those that consider pending transactions from other users, and (2) those that do not, instead relying only on the latest state of the blockchain. In the second case, the choice of peers can only affect latency in observing the updated state, because all confirmed transactions are ordered (at least partially) and propagated with priority. Assuming the block time is large compared to the median latency between nodes, all *reasonably* connected nodes in the P2P network will have a complete view of the full block history regardless of peer selection. Hence, we mainly focus on strategic behaviour that depends on learning the flow of pending transactions from other users, with the aim of issuing *frontrunning transactions* which reach miners/validators first.

3.2 Objective Formalisation

We model the P2P network as an undirected graph. Let the network be denoted by $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ before the addition of the agent node a. Agent a inserts itself into the P2P network with an initial peer set $\mathcal{P}$, producing a new network $\mathcal{N}' = \{\mathcal{V}', \mathcal{E}'\}$ where $\mathcal{E}' = \mathcal{E} \bigcup_{p \in \mathcal{P}} (a, p)$ and $\mathcal{V}' = \mathcal{V} \cup \{a\}$. Let the latency distribution between node x and y be denoted by $\mathcal{L}_{\mathcal{N}}(x, y)$.

Let the set of all transactions be denoted by $\mathcal{T}$. We model the transaction flow from different nodes as a distribution $\mathcal{F} : \mathcal{V} \rightarrow \mathcal{T}$ with the utility function $\mathcal{U}_a : \mathcal{T} \rightarrow \mathbb{R}$ for the strategic agent a such that $\mathcal{U}_a(tx)$ denotes the utility of transaction tx. We estimate the utility function based on real world data in Section 5.1. The probability of realising the utility of a transaction tx may depend on the transaction propagation in the P2P network $\mathcal{N}$, the transaction tx itself, and the consensus mechanism M to finalise the transactions. We represent this probability by $\mathbb{P}_M(\mathcal{L}_{\mathcal{N}}, tx)$. The agent's objective to maximize its expected sum of utility over all transactions can be formulated as

$$\sum_{s \in \mathcal{V}, tx \sim \mathcal{F}(s)} \mathcal{U}_a(tx) \times \mathbb{P}_M(\mathcal{L}_{\mathcal{N}}, tx) \tag{1}$$

We assume that the processing time of agent a requires to fire its own transaction is negligible compared to the network latencies and block production schedule. We now model this objective concretely for two broad classes of consensus protocols.

3.3 Time-based ordering protocols

Recent works such as Aequitas [9] and Wendy [11] propose time-based ordering protocols on a first-come-first-serve (FCFS) schedule. These protocols make use of a committee (with possibly some byzantine nodes) to decide the final ordering of transactions which is *largely* consistent with the order in which each honest committee node observes the transactions. These protocols guarantee that if a certain fraction γ of nodes in the committee $C \subset \mathcal{V}$ observe tx_1 before tx_2, then tx_1 is ordered before tx_2. This ordering mechanism is typically independent of the transaction content itself. We adapt the notion of triangular latency advantage [17] to obtain the probability of realising the utility (opportunity to frontrun) from other users' transactions:

$$\mathbb{P}_{FCFS}(\mathcal{L}_{\mathcal{N}}, tx) = [\![\sum_{c \in C} [\![(\mathcal{L}_{\mathcal{N}}(s, c) > \mathcal{L}_{\mathcal{N}'}(s, c))]\!] \geq \gamma \times |C|]\!] \tag{2}$$

where,

$$[\![X]\!] = \begin{cases} 1, & \text{if } X \text{ is true} \\ 0, & \text{otherwise} \end{cases}$$

and the addition of the agent's peers $\mathcal{P}$ in Network $\mathcal{N}$ induces the Network $\mathcal{N}'$ as described in Section 3.2

The objective function to choose the optimal peer set can now be formulated as:

$$\underset{\mathcal{P}}{\operatorname{argmax}} \sum_{s \in \mathcal{V}, tx \sim \mathcal{F}(s)} \mathcal{U}_a(tx) \times \mathbb{P}_{FCFS}(\mathcal{L}_{\mathcal{N}}, tx) \tag{3}$$

Intuitively, the objective function tries to find a peer set $\mathcal{P}$ that maximises the weighted utility of transactions for which the latency between the source node and the committee can be *short-circuited* by the agent.

3.4 Leader-Based Protocols

We now model single leader-based protocols where the leader has the freedom to choose the order of transactions among the pending transactions available to it. We assume that the agent's transaction, if reaches the leader before the next block, is attractive enough to be included before the victim's transaction. Otherwise, the agent does not take any action and its utility for the victim transaction is 0. Note that this can be achieved by either fee-based ordering or MEV auctions run by private relayers such as Flashbots. We model a stochastic block production schedule, since a fixed interval block production schedule (such as in proof-of-stake blockchains) is a degenerate case of the same. The block production schedule follows a Poisson distribution (such as that used in proof-of-work blockchains) with average rate of block production denoted by λ. Let the last block produced be at timestamp t_0. The probability of realising the utility of transaction tx issued by source s at time $t(> t_0)$ is given by the following exponential decay distribution:

$$\mathbb{P}_{leader-based}(\mathcal{L}_{\mathcal{N}}, tx) = e^{-\lambda * (\Delta t)} \tag{4}$$

where Δt is given by,

$$t - t_0 + \mathcal{L}_{\mathcal{N}}(s, a) + \mathcal{L}_{\mathcal{N}}(a, leader) \tag{5}$$

For protocols where the *leader* is unpredictable (such as PoW Ethereum), in practice the leader can be reached by making use of private relay networks such as Flashbots.

Remark. While we model leader-based protocols, we focus our optimization(Section 4 and evaluation(Section 5) on time-based protocols since they are more sensitive to latency advantages. The most popular leader-based protocol, Ethereum, has an average block interval of approximately 13 seconds, shifting the MEV game away from latency optimization to execution efficiency. However, peering algorithms would be more effective in leader-based protocols for inter-block times approaching 1 second or less, or for transactions submitted within the last second of a fixed block interval (as with PoS Ethereum).

4 PEERING ALGORITHM

Optimizing the triangular latency advantage for a set of source and target nodes is proven to be NP-hard [17]. Thus, our objective function from equation 3 is likely NP-hard for nontrivial values of the utility function and γ. In order to arrive at an approximate solution, we propose the MEV-Peri Algorithm (Algorithm 1), similar to Peri [14, 17]. We propose the following score function which can be calculated by the agent from its local view of the P2P network (since no single node has the entire view of the P2P network).

Algorithm 1: MEV-Peri algorithm, adapted from Peri [17] and Perigee [14]

```
Input: Network: $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, Peer budget: $n$, Replacement Ratio: $r$;
Output: Optimal Peer Set: $\mathcal{P}^*$;
Requires: $1 \le n \le |\mathcal{V}|$, $0 < r < 1$;
$\mathcal{P} \leftarrow$ random($\mathcal{V}, n$);
for epoch = 1,2,... do
    Sleep(epoch period);
    $\phi \leftarrow$ Init Score Map;
    for $p \in \mathcal{P}$ do
        $\phi(p) \leftarrow$ score(p);
    end
    $\mathcal{P} \leftarrow \mathcal{P} -$ least($\phi, r$);
    $\mathcal{P} \leftarrow \mathcal{P} \cup$ random($\mathcal{V}, r$);
end
```

$$\texttt{score}(p) = \sum_{\text{observed } tx} [\mathcal{U}_a(tx) \times (TS(tx) - TS_p(tx))] \qquad (6)$$

where, $TS_p(tx)$ is the timestamp at which the agent a receives the transaction tx from peer p and $TS(tx)$ is the earliest timestamp at which the agent receives the transaction from *any* of its peers. Intuitively, the score function captures the latency penalty for the peer p relative to the fastest peer weighted by the utility of the transaction it delivers. Note that we assume all peers deliver the transaction. In practice, if some peer does not deliver a transaction, its penalty can be replaced by an appropriate timeout value. We collect the scores of all the peers in each epoch, and then replace the lowest r fraction with random peers. Note also that we do not ban the evicted peers, so that if their transaction flow changes, they can be found by the agent in the future epochs.

5 EVALUATION

We conduct our analysis via simulation because direct experimentation on mainnet is prohibitively expensive (and ethically questionable). In order to measure the efficacy of our algorithm, we would have to issue valid transactions from our agent to frontrun other users' transactions, then observe their timestamps at the target nodes or order in the finalised transactions. This is because invalid transactions are not propagated by any known P2P network. Each valid DeFi transaction on Ethereum mainnet costs at least \$3 at the time of writing, so evaluation over a meaningful period and sample size would require tens of thousands of dollars.

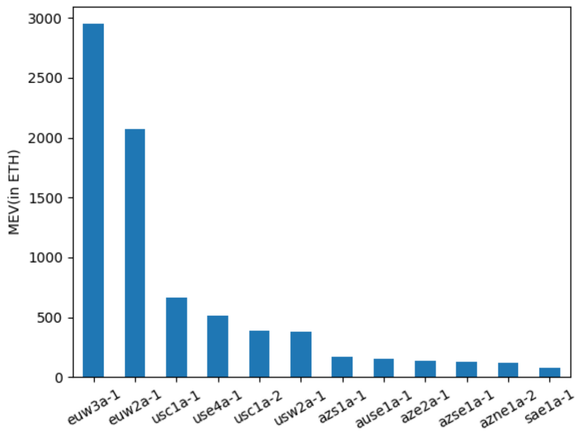

Figure 1: Total MEV attributed to geographically distributed nodes over the period of May 15 - August 14. Nodes in Europe have the highest share, followed by nodes in the United States.

We make a simplifying assumption in order to create our simulation framework: the latency of transaction propagation is independent of the utility of the transaction. In practice, these two distributions might not be completely independent. For example, nodes that host services for many retail users (eg: Infura, Alchemy) are likely to produce transactions with high utility. Those same nodes are also likely to be well connected, leading to lower latency propagation of their issued transactions. We measure the correlation in Section 5.1 to show the strength of our assumption in practice.

In order to run this simulation we require two components: 1) a simulation framework to generate P2P network topologies and inter-node latencies, 2) an estimate for our score function which itself depends on the transaction flow distribution and utility of those transactions. For the first element of our simulation, we instantiate P2P network topologies and latencies using the scale-free graph models [5]. Most organically formed P2P networks, including blockchains [18], have the characteristics of scale-free graphs. The scale-free model fits most naturally in environments where new nodes are likeliest to form edges with higher-degree nodes, creating a rich-get-richer dynamic. This is the case when, for instance, a node samples its initial peer set randomly from the peers of existing nodes.

5.1 Transaction Flow and Utility Estimation

We use the MEV of the transaction as the natural utility function, since it upper-bounds the value extracted by any agent.[1] In order to calculate the score function, we need a distribution of the total MEV over the source nodes in the network. Firstly, we must understand *which nodes* produce MEV-containing transactions. In typical P2P

[1]Note that any practical deployment of MEV-Peri should slot in the utility function specific to the agent.

networks, it is quite challenging to establish the association between a transaction and its source node. In fact, such an association would be an attack on the privacy in the P2P network. Fortunately, we only need a distribution of MEV over the nodes. To that end, we instantiate 12 geographically distributed Ethereum nodes and monitor their mempool (specifically, the timestamps of pending transactions). As a first approximation, we instead attribute the MEV of a transaction *tx* to the node that first observed *tx*. We collected the timestamps of 116 million unique transactions from May 15 to August 14. This gives us an estimate of the transaction flow across the nodes in a P2P network. We must also estimate the MEV for each MEV-containing transaction. For this, we use the public data of miner rewards for *bundles of transactions*, available via the Flashbots Blocks API, collecting information about 98k unique transactions from the above time period. Since miner rewards make sense only at a bundle level, we consider the MEV of a transaction equal to the average miner reward paid per transaction in the bundle. This is reasonably accurate since the vast majority of bundles have 2-3 transactions, consisting of victim transactions and bot transactions. Each transaction in the bundle has utility to a strategic agent, either for similarly extracting MEV from the victim transaction or for undermining the competing bot transactions. Figure 1 shows the distribution of MEV across our 12 geographically distributed nodes. Like most natural phenomena, this distribution appears to follow a power law. Hence, we fit a discrete power law to this observed data, and estimate the power law parameters using Maximum Likelihood Estimation [7]. The discrete power law is governed by the following Probability Density Function(PDF):

$$PDF(x) = \frac{x^{-\alpha}}{\zeta(\alpha, x_{min})} \tag{7}$$

where

$$\zeta(\alpha, x_{min}) = \sum_{n=0}^{\infty} (n + x_{min})^{-\alpha} \tag{8}$$

is the Hurwitz zeta function. We fix x_{min} to the MEV value of the `aze2a-1` node, i.e. the "poorest" node, and use the following closed form solution from [7] to find the optimal power law scaling parameter α:

$$\hat{\alpha} = 1 + n \left(\sum_{i=1}^{n} ln \frac{x_i}{x_{min} - 1/2} \right)^{-1} \tag{9}$$

We find $\hat{\alpha} = 1.72 \pm 0.20$. We use this estimate of the power law distribution to assign transaction flow and utility to the nodes in our simulation. In order to refine our estimate, we could spawn more Ethereum nodes, or better yet attribute each *tx* to the peer that gossiped *tx*, thus multiplying our sample size by the number of peer connections for each node. However, we defer these refinements to future work.

Remark. We study the correlation between the ranking of nodes based on our MEV estimation and the ranking simply obtained by their latencies. This correlation for our dataset is 0.65, suggesting that incorporating transaction flow in our score function can indeed give us advantage obtainable based on latency optimization alone.

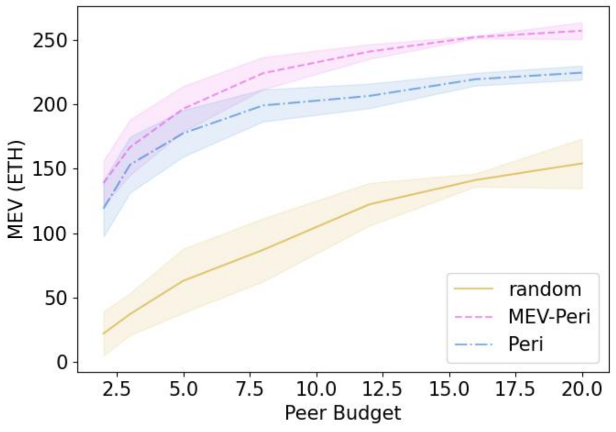

Figure 2: MEV-weighted advantage of different peering algorithms as a function of the peer budget. See Section 5 (specifically, Section 5.2) for more details.

5.2 Simulation

Equipped with the estimate of the transaction flow and utility functions, we augment the open source simulator released by the authors of [17] to add transaction flow and utility function estimates for the generated P2P graphs. We then implement the MEV-Peri algorithm (Section 4) to measure MEV-weighted advantage and benchmark it against a random peering strategy and Peri. We set the number of epochs to 800, with 1000 nodes in each P2P graph and replacement ratio of 0.25 for both MEV-Peri and Peri. We set the size of committee to 100 (10% of the nodes) with $\gamma = 0.5$. We run the simulation for different peer budgets ranging from $n = 2$ to $n = 20$. We instantiate the MEV distribution in the transactions according to the distribution ($\hat{\alpha} = 1.72$) obtained in Section 5.1 with an average MEV per epoch as 1 ETH. Our results are summarised in Figure 2. We note that MEV-Peri outperforms both a random peering algorithm and Peri, which does not take transaction data or metadata into account. Naturally, all algorithms give higher advantage with more peers, and higher peer budget also increases MEV-Peri's relative advantage against Peri.

6 DISCUSSION

We note that peering algorithms can be applied only for public mempool activity, and do not cover activity happening inside private mempools enabled by relayers such as Flashbots. It may appear that for faster chains, the latency component of a peering algorithm optimization routine would be more important than the transaction utility component. However, we observe that nodes of faster chains are also likely to gossip less, so it may be quite important to choose the *right* nodes for transaction flow.

Mitigation. While robust design of network topology and preventing strategic peering is quite challenging, if not impossible, we note that blinding the transaction data and metadata could mitigate against strategic behaviour at the network layer.

Our framework of combining transaction utility with network latency to optimise peering suggests the following avenues for future work:

- Multi-agent model: Sophisticated actors are likely to run multiple agent nodes in order to gain better visibility of the mempool activity. Hence, a multi-agent peering model would be of significant practical interest.
- Our work suggests that estimating transaction utility is key to peering optimization. Hence, more comprehensive and possibly agent-specific MEV strategy should be slotted in to estimate transaction utility.
- We do not currently consider the stability and reliability of peer connections. Depending on network topology, there may be independent utility from ensuring a quorum of safe and stable peers.

REFERENCES

[1] 2022. Maximal extractable value (MeV). https://ethereum.org/en/developers/docs/mev/

[2] 2022. MeV explore. https://explore.flashbots.net/

[3] 2022. Welcome to flashbots. https://docs.flashbots.net/

[4] 2022. What is miner-extractable value (MeV)? https://blog.chain.link/what-is-miner-extractable-value-mev/

[5] Ré ka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 1 (jan 2002), 47–97. https://doi.org/10.1103/revmodphys.74.47

[6] Kushal Babel, Philip Daian, Mahimna Kelkar, and Ari Juels. 2021. Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts. Cryptology ePrint Archive, Paper 2021/1147. https://eprint.iacr.org/2021/1147 https://eprint.iacr.org/2021/1147.

[7] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. Power-Law Distributions in Empirical Data. *SIAM Rev.* 51, 4 (nov 2009), 661–703. https://doi.org/10.1137/070710111

[8] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy (S&P)*. 910–927. https://doi.org/10.1109/SP40000.2020.00040

[9] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. 2020. Order-Fairness for Byzantine Consensus. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 451–480.

[10] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. 2022. Towards a Theory of Maximal Extractable Value I: Constant Function Market Makers. https://doi.org/10.48550/ARXIV.2207.11835

[11] Klaus Kursawe. 2020. Wendy, the Good Little Fairness Widget: Achieving Order Fairness for Blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies* (New York, NY, USA) *(AFT '20)*. Association for Computing Machinery, New York, NY, USA, 25–36. https://doi.org/10.1145/3419614.3423263

[12] Yulin Liu, Yuxuan Lu, Kartik Nayak, Fan Zhang, Luyao Zhang, and Yinhong Zhao. 2022. Empirical Analysis of EIP-1559: Transaction Fees, Waiting Time, and Consensus Security. arXiv:arXiv:2201.05574

[13] Dahlia Malkhi and Pawel Szalachowski. 2022. Maximal Extractable Value (MEV) Protection on a DAG. https://doi.org/10.48550/ARXIV.2208.00940

[14] 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* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 428–437. https://doi.org/10.1145/3382734.3405704

[15] Petar Maymounkov and David Mazières. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. *Peer-to-Peer Systems* (2002), 53–65. https://doi.org/10.1007/3-540-45748-8_5

[16] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (San Diego, California, USA) *(SIGCOMM '01)*. Association for Computing Machinery, New York, NY, USA, 149–160. https://doi.org/10.1145/383059.383071

[17] Weizhao Tang, Lucianna Kiffer, Giulia Fanti, and Ari Juels. 2022. Strategic Latency Reduction in Blockchain Peer-to-Peer Networks. https://doi.org/10.48550/ARXIV.2205.06837

[18] Taotao Wang, Chonghe Zhao, Qing Yang, Shengli Zhang, and Soung Chang Liew. 2021. Ethna: Analyzing the Underlying Peer-to-Peer Network of Ethereum Blockchain. *IEEE Transactions on Network Science and Engineering* 8, 3 (2021), 2131–2146. https://doi.org/10.1109/TNSE.2021.3078181

[19] Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. 2021. On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols. In *2021 IEEE Symposium on Security and Privacy (S&P)*. 919–936. https://doi.org/10.1109/SP40001.2021.00113

[20] Liyi Zhou, Kaihua Qin, and Arthur Gervais. 2021. A2MM: Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges. https://doi.org/10.48550/ARXIV.2106.07371

[21] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. 2021. High-Frequency Trading on Decentralized On-Chain Exchanges. In *2021 IEEE Symposium on Security and Privacy (S&P)*. 428–445. https://doi.org/10.1109/SP40001.2021.00027