Lesson 9 - MEV

> ## MEV - Maximum Extractable Value
>
> **TL;DR** The additional value that can be extracted outside of the core protocol and how decentralisation & censorship resistance is affected by that.
>
> **Miner-extractable value (MEV)**: We introduce the notion of MEV, value that is ***extractable by miners directly from smart contracts as cryptocurrency profits***. One particular source of MEV is ordering optimization (OO) fees, which result from a miner's control of the ordering of transactions in a particular epoch. PGAs and pure revenue opportunities provide one source of OO fees. We show that MEV creates systemic consensus-layer vulnerabilities. - *Flashboys 2.0* ***(2019)***
>
> ### Key ideas:
>
> - Validators have the power to decide ordering and inclusion of transactions.
> - They can extract extra value by ordering transactions, or selling the right to order transaction.
> - This creates consensus-layer vulnerabilities because incentives become mal aligned to the core protocol.
>
> ### Ethereum Architecture:
>
> Ethereum is susceptible to MEV in the a few ways:
>
> All transactions are available in a public mempool before they are mined
>
> All transaction data is public
>
> Transactions can be cloned
>
> Slow / each block is 12 seconds
>
> There a lot of complexity (Defi ect) => Emergent unintended phenomena
>
> ### Is MEV unique to Ethereum?
>
> No, hypothetically MEV can also be seen on Bitcoin. The incentives to censor Lightning channels or to double-spend colored coins are technically MEV. However, Bitcoin is inherently less exposed to MEV than blockchains like Etheruem.
>
> The reason for that lies in the complexity and "statefulness" of the respective blockchain:

> The rate at which MEV accumulates on a given blockchain is generally proportional to the complexity of its application-layer behavior.
>
> Ethereum is a general purpose blockchain so cannot bound this complexity.
>
> MEV incentives cannot be easily mitigated without altering Ethereum's UX.
>
> Other highly complex blockchains such as Solona also have high levels of MEV.

Main types of MEV

Arbitrage & Liquidations

- Arbitrage is the largest source of MEV revenues.
- Arbitrage does helps market efficiency in general.
- But it's incredibly competitive and this creates negative externality such as:
  - incentives to centralise,
  - collude
  - frontrun transactions.

Atomic arbitrage

- Reduced inventroy risk
- no timing risk
- (Atomic) Either you profit or transaction fails
- Flashloan can be used for collateral (less competitive due to gas requirements)

Cross domain MEV

- It is more difficult and risky mainly because you can't do atomic arbitrage.
- Bridges are slow so capital is needed on both chains.
- Transactions must be executed simultaneously on multiple chains.
- Create Incentives for validators to collude and try to monopolise order flow across multiple chains.
- Even if validators don't collude there are economies of scale to running multiple validators on multiple chains.
- Unchecked this could lead to a winner takes all dynamic and more centralisation.

If you want to go deep into the theory:
Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value

Liquidations

- Lending protocols Aave, Compound, and Maker.

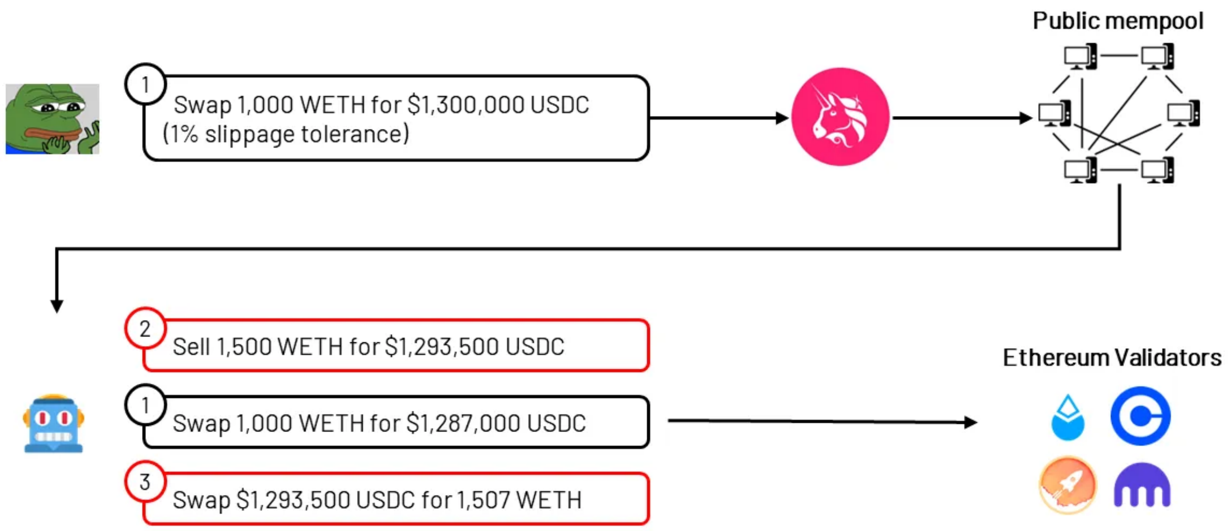
- Similar to arbitrage, liquidation events are highly competitive. During sharp market downturns, the competition to liquidate borrowers has led to enormous gas fees.

Font Running

Generalised front running bots: Introductory Video from Scott Bigelow

https://www.youtube.com/watch?v=UZ-NNd6yjFM

- **Front Running**: Monitoring mempool for profitable transactions then submitting them but with higher gas fees.
- **Back Running**: monitoring of a mempool to execute a transaction immediately after a pending target transaction.
- **Sandwich Attack**: Combination of front & back running to sandwich a trade.


Public mempool
1 Swap 1,000 WETH for $1,300,000 USDC (1% slippage tolerance)
Ethereum Validators
2 Sell 1,500 WETH for $1,293,500 USDC
1 Swap 1,000 WETH for $1,287,000 USDC
3 Swap $1,293,500 USDC for 1,507 WETH

MEV analytics: https://eigenphi.io/mev/ethereum/sandwich

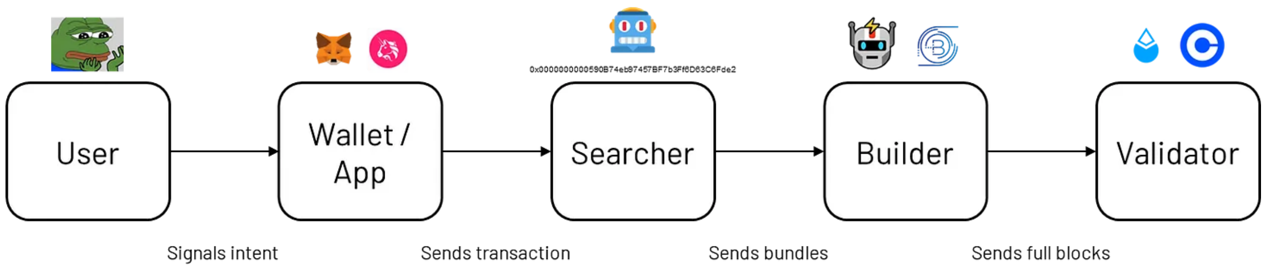
Priority Gas Auctions

- Without an effective fee market, competition for MEV can lead to Priority Gas Auctions.
- Searchers and bots compete against with increasingly higher gas fees

This leads to:

- Network congestion, which may cause other transactions to get stuck or dropped by the network
- High and volatile gas prices, crowding out other crypto users and disadvantaging less savvy participants
- Failed bids that unnecessarily consume block space (even failed transactions use up block space)

Transaction Flow



User: Anyone who would like to express and submit intent to change the state of the blockchain.

Wallet / Application: The user interface that translates user intent into a blockchain transaction.

Searcher: Entities that monitor the mempool and submit transactions to extract MEV.

Builder: Aggregate transactions from various sources to create a full block, ideally one that maximizes rewards.

Validator: Perform consensus duties, such as proposing and attesting to blocks.

MEV Optimisations:

- Using vanity addresses with leading 0000
- Using multicalls contract
- Simulating transactions to target same block
- Simulating locally
- Decopile tools https://decompile.tools/contract/1/0xa69babef1ca67a37ffaf7a485dfff3382056e78c
- Libevm talk: https://www.youtube.com/watch?v=qj3CHc58GKE
- It's a trap: https://www.youtube.com/watch?v=zGz410AJEEs
- MEV day https://flashbots.notion.site/MEV-DAY-836f88806995412dabc1c7bb7ce4e830

Transaction Propagation Services

Pre Merge approach - Flashbots auction

Flashbots Auction provides a private communication channel between Ethereum users and miners for efficiently communicating preferred transaction order within a block.

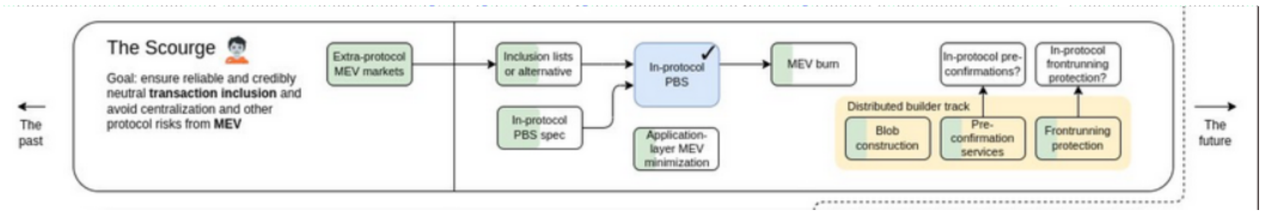
Flashbots Auction consists of mev-geth, a patch on top of the go-ethereum client, along with the mev-relay, a transaction bundle relayer.

Flashbots mitigated the negative effects of PGAs and front running but it created other problems.

Endgame

Endgame Vitalik: https://vitalik.ca/general/2021/12/06/endgame.html

There's a high chance that block production will end up centralized: either the network effects within rollups or the network effects of cross-domain MEV push us in that direction in their own different ways. But what we can do is use protocol-level techniques such as committee validation, data availability sampling and bypass channels to "regulate" this market, ensuring that the winners cannot abuse their power.


The past
The Scourge
Goal: ensure reliable and credibly neutral transaction inclusion and avoid centralization and other protocol risks from MEV
Extra-protocol MEV markets
Inclusion lists or alternative
In-protocol PBS spec
In-protocol PBS
Application-layer MEV minimization
MEV burn
In-protocol pre-confirmations?
In-protocol frontrunning protection?
Distributed builder track
Blob construction
Pre-confirmation services
Frontrunning protection
The future

Ethereum roadmap:
https://twitter.com/VitalikButerin/status/1588669782471368704/photo/1

Proposer Builder Separation (PBS)

(Status: In development not implemented/Not a solved problem)

TL;DR: Split the roles of validators/builders so that the centralisation forces of MEV are contained in builder layer.

- Builders are highly specialized actors that construct candidate blocks and make bids to get them included
- Proposers are validators that naively accept the highest bid
- Goal: builders absorb economies of scale, proposers stay decentralized

But how How to stop validators from stealing the MEV from searchers?

Ideas:

- Encrypt transactions: commit reveal. Trusted hardware etc.
- Use ZK-snarks
- Cr lists. Proposer submits a list of must include transactions to the builder.

Explainer video: https://www.youtube.com/watch?v=fAgrIdyWIqc

Sources:
Hitchhikers guide to Ethereum
Proposer Builder Separation

MEV-Boost (Flashbots) an interim solution

Mev-Boost is an offchain implementation of PBS

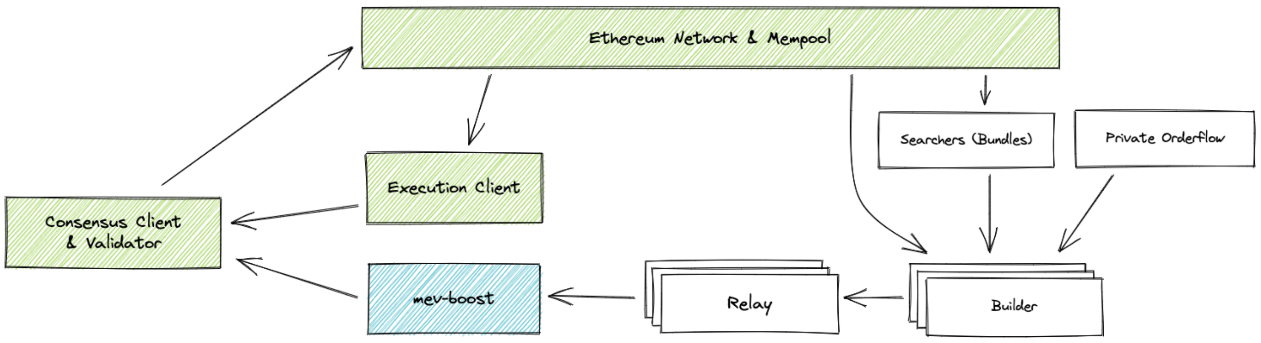
It is an interim solution until PBS can implemented at protocol level. It's optimised for liveness and resilience.

How it works:

- **Searchers** take transactions from the public mempool, potentially adding their own, and arrange them into bundles.
- **Block builders** are services/providers that will aggregate various bundles and transactions into block templates.
- The **relay** receives these block templates (also referred to as execution payloads), and will verify their validity.
- The **MEV-boost** component is middleware which handles communication with the relays, the profit-switching logic (for selecting the most valuable payload), and a fallback mechanism in the case of some system failure.

Using a middleware instead of direct modification of the consensus clients allows for maintaining each component independently and provide cross client compatibility with minimal changes to the engine api.

Source: https://ethresear.ch/t/mev-boost-merge-ready-flashbots-architecture/11177

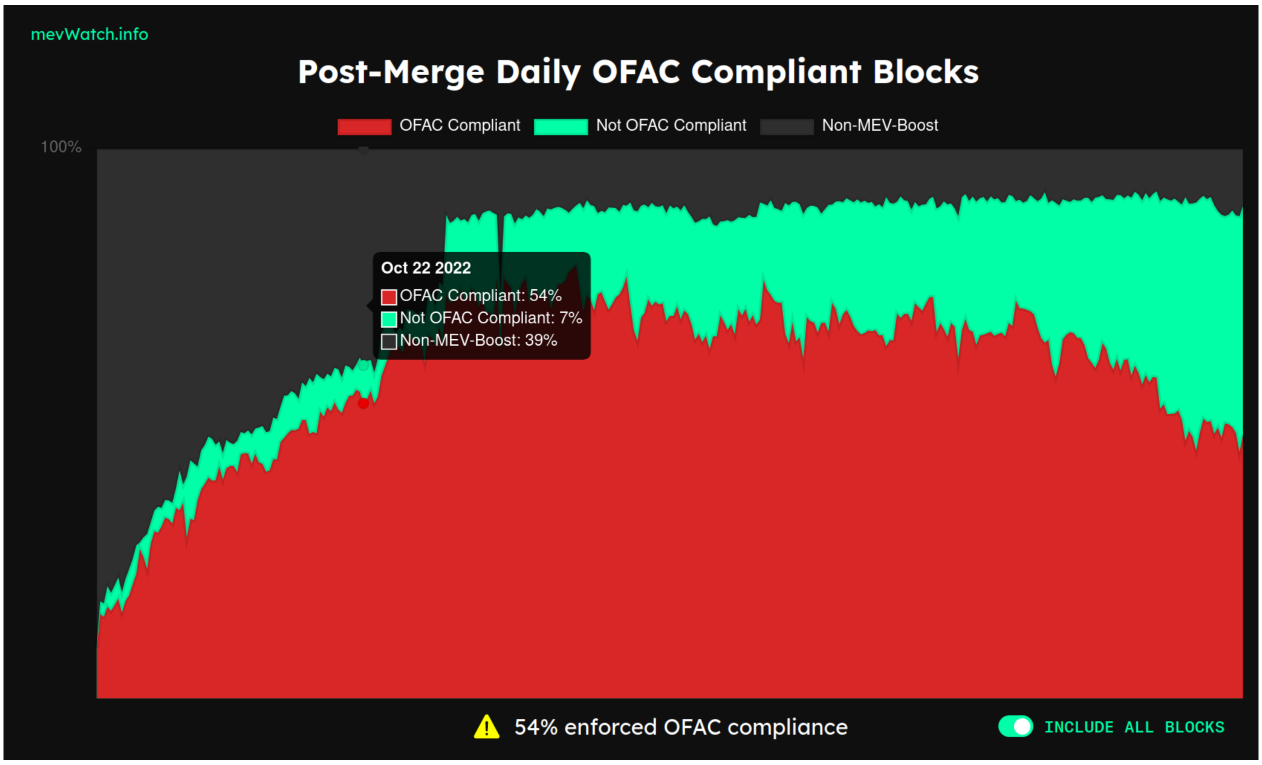


Beginner's Guide to mev-boost: https://writings.flashbots.net/writings/beginners-guide-mevboost/

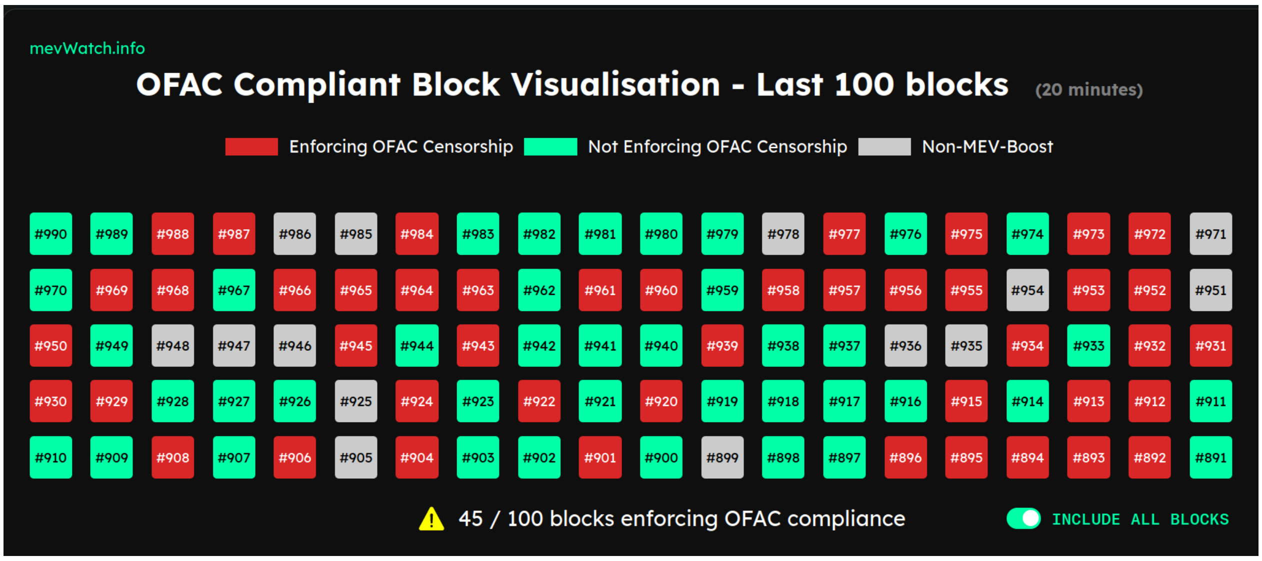
MEV-Boost, How Does it Work & What's Missing: https://www.youtube.com/watch?v=PS9SlHGJlrU

Censorship Risks

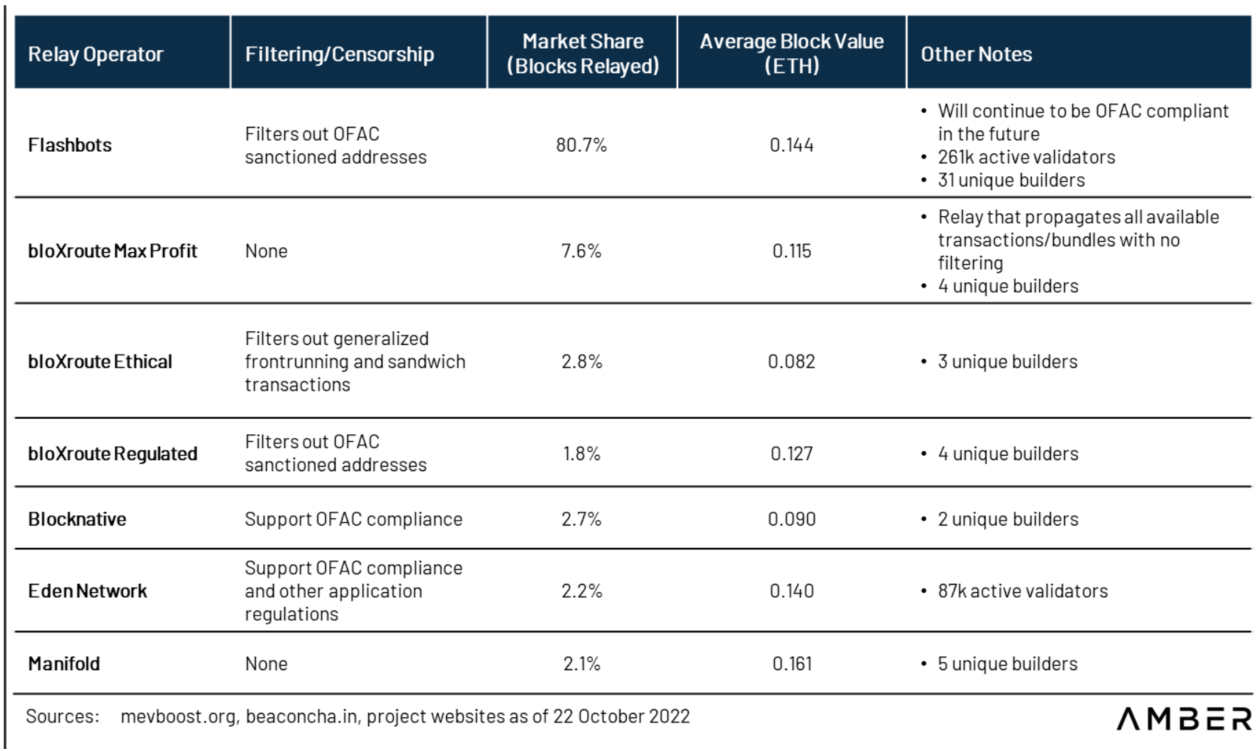
Only a handful of entities run relays and builders, and some of those entities have to abide by local regulators. Notably, several relays and builders, including Flashbots' own ones, filter out OFAC-sanctioned addresses. They will not include any transactions that interact with any smart contract or addresses specified by the US Treasury Department.



- Important to note that we only need a small number of non-censoring blocks to prevent censorship



Most agree that best mitigation for now is having relay diversification & long term to implement PBS.



Relay Operator	Filtering/Censorship	Market Share (Blocks Relayed)	Average Block Value (ETH)	Other Notes
Flashbots	Filters out OFAC sanctioned addresses	80.7%	0.144	• Will continue to be OFAC compliant in the future • 261k active validators • 31 unique builders
bloXroute Max Profit	None	7.6%	0.115	• Relay that propagates all available transactions/bundles with no filtering • 4 unique builders
bloXroute Ethical	Filters out generalized frontrunning and sandwich transactions	2.8%	0.082	• 3 unique builders
bloXroute Regulated	Filters out OFAC sanctioned addresses	1.8%	0.127	• 4 unique builders
Blocknative	Support OFAC compliance	2.7%	0.090	• 2 unique builders
Eden Network	Support OFAC compliance and other application regulations	2.2%	0.140	• 87k active validators
Manifold	None	2.1%	0.161	• 5 unique builders

Sources: mevboost.org, beaconcha.in, project websites as of 22 October 2022

AMBER

More stats:

- https://mevboost.pics/
- https://mevboost.pics/

How to get started

Use flashbots with meta mask to protect against front running and sandwich attacks

- https://docs.flashbots.net/flashbots-protect/rpc/quick-start/
- https://docs.bloxroute.com/introduction/fast-protect

Listening to mempool

Using ether js we can listen to mempool

```
var ethers = require("ethers");
var url = "ADD_YOUR_ETHEREUM_NODE_WSS_URL";

var init = function () {
  var customWsProvider = new ethers.providers.WebSocketProvider(url);

  customWsProvider.on("pending", (tx) => {
    customWsProvider.getTransaction(tx).then(function (transaction) {
      console.log(transaction);
    });
  });
};

init();
```

Listen to contract events

```
const ethers = require("ethers");
const url = 'wss://eth-mainnet.g.alchemy.com/v2/...Key here...'
const abi = '[{....}]'

const customWsProvider = new ethers.providers.WebSocketProvider(url);
//Curve Aave stable Swap
//https://etherscan.io/address/0xDeBF20617708857ebe4F679508E7b7863a8A8
EeE#code
const contract = new
ethers.Contract('0xDeBF20617708857ebe4F679508E7b7863a8A8EeE', abi,
customWsProvider)


contract.on("AddLiquidity", (provider, token_amounts, fees) => {
    console.log(provider, token_amounts, fees);
});
```

Send transaction bundles via Flashbots

```
const ethers = require("ethers.js");
const {
    FlashbotsBundleProvider,} = require("@flashbots/ethers-provider-
bundle");
const provider = new ethers.providers.JsonRpcProvider({
    url: ETHEREUM_RPC_URL,});
    // Standard json rpc provider directly from ethers.js. For example
you can use Infura, Alchemy, or your own node.
const authSigner = new ethers.Wallet(
"0x0000000000000000000000000000000000000000000000000000000000000000");
// `authSigner` is an Ethereum private key that does NOT store funds
and is NOT your bot's primary key.// This is an identifying key for
signing payloads to establish reputation and whitelisting
const flashbotsProvider = await FlashbotsBundleProvider.create(
provider,  authSigner);// Flashbots provider requires passing in a
standard provider and an auth signer

const signedBundle = await flashbotsProvider.signBundle([  {
signer: SOME_SIGNER_TO_SEND_FROM,    transaction:
SOME_TRANSACTION_TO_SEND,  },]);
const bundleReceipt = await flashbotsProvider.sendRawBundle(
signedBundle,  TARGET_BLOCK_NUMBER);
```

Example Bots

- Sandwich (Out of date and not we do not recommend running sandwich bots in general) Repo
- Uniswap V3 python https://github.com/BowTiedDevil/degenbot/tree/main/arbitrage
- Hummingbot https://hummingbot.org/

Resources

- MEV-sbc Workshop: https://flashbots.notion.site/MEV-sbc-Workshop-6dcb3501d7e647b98b703b6312c3e95b
- Overview of MEV: https://amberlabs.substack.com/p/extractable-value
- General overview of Ethereum + PBS: https://members.delphidigital.io/reports/the-hitchhikers-guide-to-ethereum/
- MEV day : https://www.youtube.com/watch?v=zGz410AJEEs
- https://medium.com/iosg-ventures/suave-entering-the-new-mev-decade-dc95ee91a603

Other Approaches

Some protocols have taken steps to prevent MEV
such as using Verifiable Delay Functions in order to prevent gaming of transaction ordering, as Solana has done on its base layer to ensure transactions are ordered by time of arrival, or simply delegate ordering to something like Chainlink's Fair Sequencing Service (FSS)

Fair Sequencing Service

See Blog
"In a nutshell, the idea behind FSS is to *have an oracle network order the transactions sent to a particular contract SC*, including both user transactions and oracle reports. Oracle nodes ingest transactions and then reach consensus on their ordering, rather than allowing a single leader to dictate it. Oracle nodes then forward the transactions to the contract *SC*. They sequence these transactions by attaching nonce or sequence numbers to them or sending them in batches."

You can read more about Fair Sequencing Services in the Chainlink 2.0 White paper (Section5): https://research.chain.link/whitepaper-v2.pdf

5.2 FSS Details

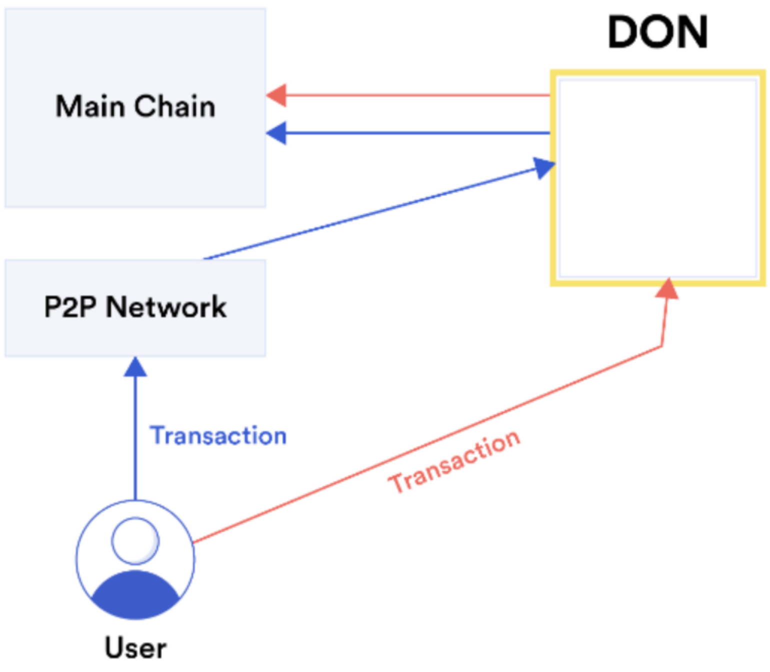


Figure 12: Order-fair mempool with two different transaction paths: direct and mempool-based.

1. Direct: The direct approach is conceptually simplest, but requires changes to user clients so that transactions are sent directly to the Decentralized Oracle 48 Network nodes, rather than to the nodes of the main chain.
   The DON collects user transactions destined to a specific smart contract SC and orders them based on some ordering policy. The DON then sends the ordered transactions to the smart contract on the main chain. Some ordering mechanisms also require the direct approach because the user that creates a transaction must cryptographically protect it before sending it to FSS.

2. Mempool-based: To facilitate the integration of FSS with legacy clients, the DON can use Mempool Services (MS) to monitor the main chain's mempool and collect transactions.