Polygon FastLane

By Alex W (AKA ‘Thogard’) / FastLane Labs

Summary:

The ‘Polygon FastLane’ (‘PFL’) is a protocol designed to compensate participating validators, reduce transaction spam, and improve network health by monetizing propagation bottlenecks in the p2p (peer-to-peer) layer of the Polygon blockchain. The protocol seeks to achieve this in a way that:

1. Incurs no transaction delay beyond the limit of what is already possible.
2. Reduces the economic incentive for spam during the sprints of participating validators.
3. Does not facilitate transactions that cause overt harm to Polygon’s users (does not directly encourage ‘sandwich’ attacks or front-running).
4. Does not require validators to run custom code on their validator node
5. Potentially lowers the data transfer costs for validators and searchers.

Background:

Polygon is a Proof of Stake (‘PoS’) blockchain; the transaction inclusion and ordering is determined by special nodes called validators. The validator that will ‘mine’ (IE determine transaction inclusion and sequencing) a block is predetermined by the HEIMDALL layer and based on the ‘stake’ (IE the amount of the native token, MATIC, held in an escrow-like account that is potentially forfeited if misbehavior is detected). The larger the stake, the more blocks a validator will be assigned to mine. Block assignments come in sets of 64 blocks called sprints. The selected validator will mine 64 blocks in a row before another validator takes over.

The p2p (‘peer-to-peer’) layer of the Polygon blockchain is comprised of nodes (IE blockchain clients) that are each connected (IE ‘peered’) to a subset of all of the nodes that comprise the blockchain. While all validators and sentries are nodes, not all nodes are validators or sentries.

Spam is a serious issue on the Polygon blockchain. The origin of this problem can be traced to a line of code in the BOR code base:

https://github.com/maticnetwork/bor/blob/master/eth/handler.go#L623

(Note: this line of code is also present in GETH—this is not a Polygon-specific problem)

Transaction Propagation:

When a transaction is created and begins propagating through the p2p layer to reach the validator, at each node the transaction broadcast could follow one of two distinct propagation paths; it is sent 'directly' to P(√(peers) / peers) of its peers and it is 'announced' to the remaining set of peers. Whether a specific peer receives 'direct' propagation or 'announced' propagation is random*. For the purposes of this abstract, it is important to know that **'direct' transaction propagation results in approximately a 500ms speed advantage over 'announced' transactions** (on BOR clients that haven't been modified). For example, if a searcher sends a transaction to a peer with a peer count of 100, there is a 10% (P(√(100) / 100)) chance that the transaction will be propagated directly and a 90% chance that it will be announced and therefore subject to the 500ms delay.

**(Note: the random assignment of direct or announced propagation is due to the peer set being a map, and maps in golang are not ordered.)*

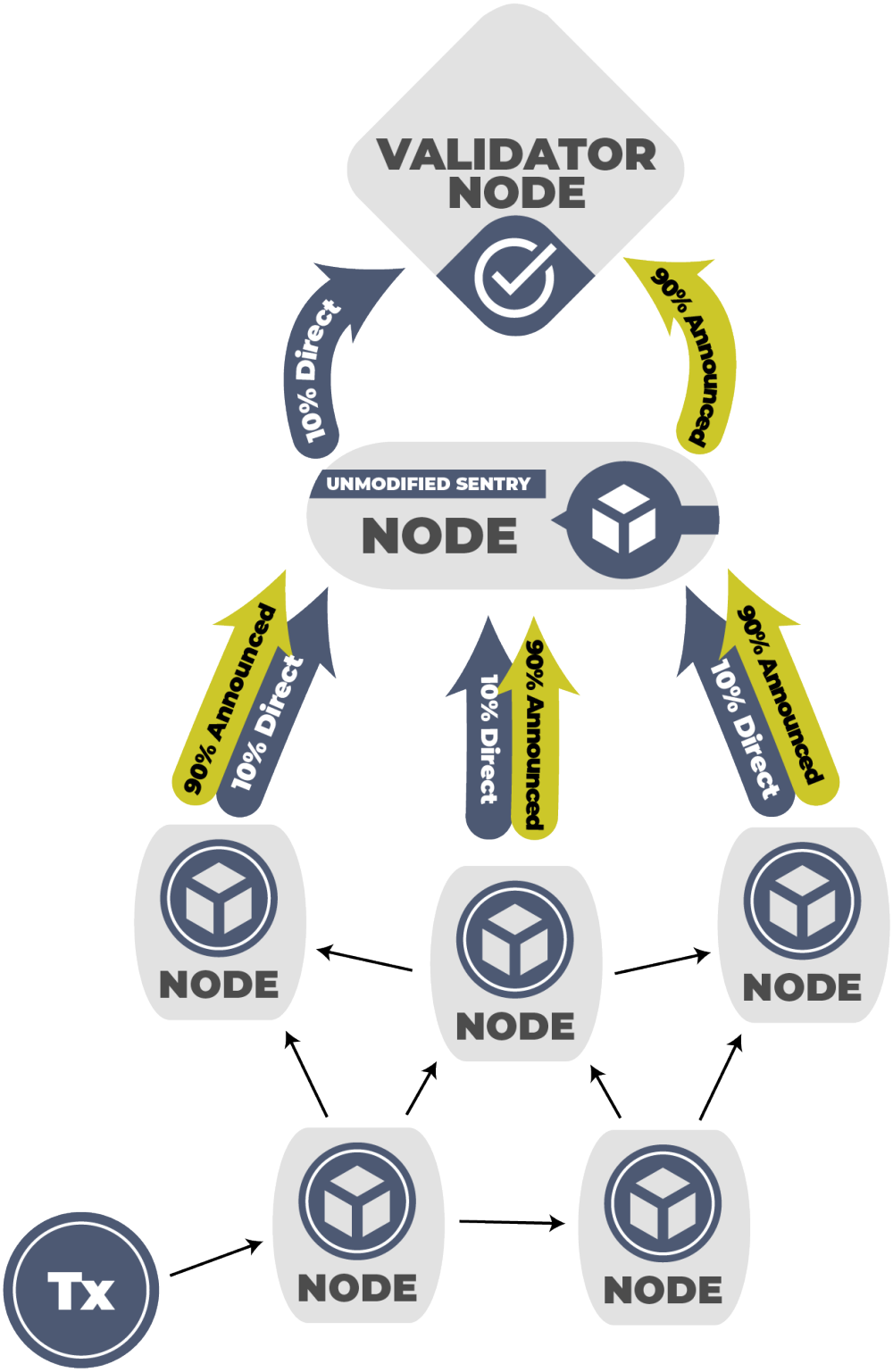
Although transaction priority inside of a block is initially determined by the value of the gasPrice* ** parameter, when a validator receives multiple transactions that have identical gasPrice values the transaction priority is determined chronologically—the first transaction **received by the validator** is placed first in the set of transactions with identical gasPrice values in a block.

**(Note: After EIP-1559, the parameters are 'maxPriorityFeePerGas' and 'maxFeePerGas.' For the sake of simplicity, we will use 'gasPrice' in their place.)*

***(Note: This assumes no other MEV mechanism such as MEV-Bor being run by the validator. As we will discuss later, this is not necessarily a safe assumption.)*

On Polygon, the majority of validator nodes are not 'peered' ('peered' means connected via the p2p protocol for the exchange of information such as transaction data) in the standard format. Rather than peering just any node, validator nodes often have only a few (or one) peers, called 'sentries.' These sentries connect to the standard number of peers via the p2p protocol and serve as a buffer between the validator nodes and the broader network. While they are an effective safety mechanism to prevent against DDOS-type attacks, they also act as a bottleneck: there is no route to a validator other than through its sentry. This bottleneck is significant. An example: imagine that Node A has a transaction that Node B does not have. Even if Node A does not directly propagate the transaction to Node B, Node B still has dozens of other peered nodes, each of which also might propagate the transaction directly to it. Validator nodes, however, don't have other peers from whom they can receive direct transaction propagation. **If the validator's sentry does not propagate the transaction directly to the validator node, then the validator node will be forced to receive the transaction via 'announcement,' thereby incurring the 500ms delay.**

Illustration of default transaction propagation:



Searcher Transactions:

Two of the most profitable types of arbitrage (arbitrage is a series of trades that generates profit while controlling for risk/delta) on the Polygon blockchain are ‘backruns’ and ‘liquidations.’ Both of these styles of arbitrage can have only one winner per opportunity created, but have multiple searchers (‘searchers’ are operators of MEV bots) competing for them. Furthermore, both of these styles’ opportunities are created by a transaction in the mempool: for backruns the opportunity is usually created by a large trade transaction sent to a DEX (IE decentralized exchange—an exchange on the blockchain itself) router contract, and for liquidations the opportunity is usually created by a price update transaction sent to a price oracle contract. The winner of the opportunity—and therefore the profit—is the searcher whose transaction is placed directly behind the transaction that created the opportunity.

Due to the randomness created by the ‘direct’ vs ‘announced’ propagation styles discussed above, the arbitrage transaction that lands directly behind the opportunity-creating transaction is not always the first one sent. Imagine a validator that has one peered node—its sentry node. That sentry node has 100 peers, one of which is the validator. In this hypothetical scenario, there are 3 nodes (including the validator’s sentry) in between the RPC node that receives the opportunity and the validator that sequences the transactions. Each of these nodes has 100 peers. Therefore, each node represents a 10% chance (P(√(100) / 100)) of ‘direct’ propagation and a 90% chance of ‘announced’ propagation and the ensuing 500ms delay.

Not only does the searcher want their transaction to land at the validator as fast as possible, they also want their transaction to use the same *propagation pattern* as the opportunity-creating transaction.* While on its route to the validator, if the opportunity-creating transaction is propagated directly by the first node, announced by the second, announced by the third, and propagated directly on the final hop from the sentry to the validator, then the optimal path for the searcher transaction would also have two direct propagation hops and two announced hops. If the opportunity-creating transaction is delayed and the searcher’s transaction is not, then the searcher’s transaction will arrive too soon and not be profitable. Inversely, if the opportunity-creating transaction isn’t delayed and the searcher’s transaction is, then the searcher’s transaction will arrive too late and another searcher will have won the opportunity’s profit.

*(*Note: we are aware that we are oversimplifying the statistics of propagation in the p2p layer in order to avoid unnecessary complexity. This is done to aid in understanding the root problem.)*

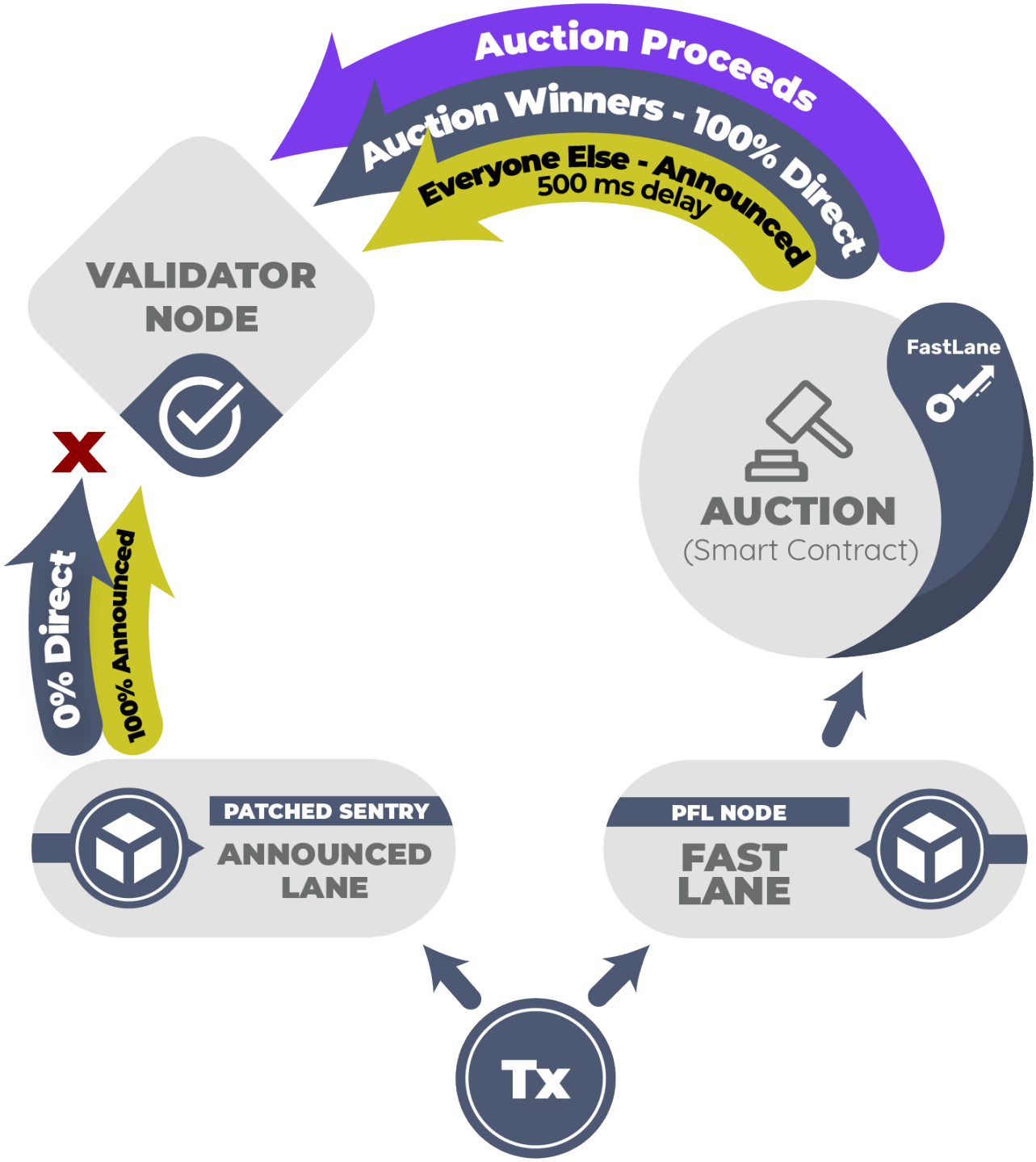
The end result of this randomness is incentivized spam: if an opportunity presents with $1,000 in potential profit and with a transaction cost of only $0.01 and each of the searcher’s transactions has a 1% (hypothetical) chance of matching the propagation pattern of the opportunity-creating transaction, then the rational course of action for a profit-seeking searcher is to send multiple transactions. In fact, as the ratio of the profit to the cost of the transaction increases, the optimal number of searcher transactions also increases. This has led to situations in which over 1000 transactions from a single searcher have been observed for a single liquidation opportunity.

PFL Overview:

The Polygon FastLane will reduce the incentive for spam by removing the randomness from the ‘direct’ vs ‘announced’ propagation dynamic during the sprints of participating validators. The system works as follows:

- Searchers who detect an opportunity-creating transaction in the ‘mempool’ can submit that transaction and their own transaction via the PFL searcher relay.
- The searcher’s submission to the relay will be a bundle consisting of two transactions:
    - The opportunity-creating transaction.
    - The searcher’s transaction.
- Upon receipt of a searcher bundle containing a new opportunity-creating transaction, a new auction will be started with a duration of approximately 100ms to 400ms (TBD).
- The PFL searcher relay(s) will immediately propagate the opportunity-creating transaction to as many peers as possible, excepting the validator node(s).
- Any additional searchers who submit a bundle with the same opportunity-creating transaction before the end of the auction will be placed into the same auction.
- At the end of the auction, the opportunity-creating transaction and the searcher transactions will be sent to the targeted validator.
    - The opportunity-creating transaction will arrive at the validator first.
    - The searchers’ transactions will arrive in order of the size of their validator payments.
- If a searcher’s transaction is successful, the opportunity will be considered ‘fulfilled’ and all subsequent searchers’ transactions targeting the same opportunity will automatically revert.
- The searcher’s payment to the validator will be held in escrow in the PFL Validator Vault contract.
- At the end of each week, the proceeds from all PFL Auctions during that week will be added to validator’s balance and will become eligible for withdrawal by the validator at any time.

Diagram of the transaction propagation for participating validators:



Implementation:

Validators:

Participating validators will need to reach out to the PFL team and be approved for participation. Once approved, participating validators will need to perform two actions:

1. The validator will need to peer their validator node to at least one PFL sentry node.

AND

2a. The validator will need to modify their own sentry node(s) to block the random chance of directly propagating a transaction to their validator node. This is the removal of roughly 5 lines of code in the BOR code base. A patch (which we call the 'Sentry Patch') will be provided for them that can be run after updating to the most recent BOR version.

OR

2b. If the validator has no sentries of its own or hasn't patched its sentries with PFL's 'Sentry Patch' then the validator will have to make sure that the validator node's only peers are PFL Nodes. If this is the choice selected by the validator, it is *strongly* recommended that the validator maintains a 'stand-by' sentry that remains unpeered by the validator but that can be peered during periods of PFL node maintenance

(Note: we strongly recommend 2a over 2b.)

Upon completion of steps 1 and 2, participating validators will become eligible to receive proceeds from the PFL Auctions. To collect their revenue, they can visit the PFL Validator Vault and initiate a withdrawal.

Searchers:

Searchers will need to follow four steps in order to participate in PFL auctions.

1. Searchers must add support for the PFL Auction contract to their own contract. This process is similar to implementing flash loans or flash swaps.
2. For each EOA ('account') the searcher plans to use for signing their searcher transactions in PFL bundles, the searcher must first 'enable' that EOA on the PFL Auction contract. This only needs to happen once per EOA per searcher contract.
3. Each searcher must build their searcher transaction, sign it with an 'enabled' EOA from #2, and place it inside a PFL bundle consisting of the opportunity-creating transaction and the searcher's transaction.
4. The searchers must then submit their PFL bundles to the PFL Relay via API.

Auction Details & Rationale:

There are two core components in the PFL system that are designed to work in parallel. The first component is the PFL Node and the second is the PFL Auction Contract.

PFL Nodes:

A PFL Node is a heavily modified BOR node. The node performs multiple functions:

1. Connects with other PFL Nodes to create a fast network for rapid bundle propagation via a private intranet.
2. Peers with participating validators via BOR's standard p2p protocol.
3. Receives searcher bundles sent via the PFL Searcher Relay.
4. Initiates and processes PFL auctions, runs safety checks, and determines the ordering of the participating searchers' transactions.
5. Widely propagates the opportunity-creating transaction to all peers except the validator at the start of the auction.
6. Delivers the 'opportunity-creating' transaction and the searcher transactions to the validator in the correct order.

One of PFL's primary goals is to create an open and free market for searchers to compete in, thereby increasing auction revenue for participating validators. By widely propagating the opportunity-creating transaction, we hope that more searchers are able to see the opportunity before its completion and participate in the bidding. While this may seem straightforward, there is another element to this decision that we'd like to discuss: PFOF ('payment for order flow') protocols.

PFOF protocols function by keeping opportunity-creating transactions private, having a singular searcher create a transaction with a very low validator payment, and submitting the bundle to a validator via private relay. By accepting bundles directly from these PFOF protocols, validators ensure that they miss out on revenue that they otherwise could have received had the private transactions been made public and therefore subject to bidding wars between searchers.

PFL is not opposed to PFOF protocols – in fact, we view them as potential partners– but we do want to make sure that participating validators are fairly compensated when a PFOF protocol uses the PFL system. We therefore plan to create separate, private APIs for PFOF protocols. These APIs will function similarly to the PFL Searcher Relay but will *not* widely propagate the opportunity-creating transaction. These APIs will only be available to PFOF protocols who have completed a revenue-sharing agreement with PFL. The profit from these agreements will be sent to the PFL Validator Vaults, similar to the searcher auctions proceeds. Furthermore, just like the searcher auction transactions, these private bundles from PFOF protocols will be identifiable afterwards via the block explorer.

PFL Auction Contract:

The PFL Auction Contract is designed for use both by searchers and validators. It has two distinct parts: the Validator Vault and the Searcher Relay.

Validator Vault:

The PFL Validator Vault is where validators' proceeds are stored. For safety and gas efficiency purposes, we've implemented a 'rounds' system. At the end of a round, all of the revenue a validator has accrued will become available for withdrawal. Each round will last approximately a week, but that length could be shortened based on feedback from validators.

Searcher Auction:

The PFL Searcher Auction will be the primary focus of any searcher participating in PFL auctions. We strongly recommend that searchers visit the GitHub repo (https://github.com/Polygon-Fast-Lane/auction/tree/main/contracts) to see the auction code (in the 'jit-relay' folder) and implementation examples (in the 'jit-searcher' folder.)
(Note: folder names may have changed since the time of this writing.)

Below is an overview of key points & the rationale behind certain design choices:

1. All searchers' transactions must call the 'submitFastLaneBid' function in the PFL Auction Contract. Any searcher transaction with a 'to' parameter that isn't the PFL Auction Contract will be rejected by the PFL Node.

2. In the call arguments, the searcher must specify their bid amount, the transaction hash of the 'opportunity-creating' transaction, their own smart contract address, and the input data (bytes) with which to call their smart contract.

3. Because the PFL Node will deliver *all* searcher transactions in the order of their bid amount, the PFL Searcher Auction will first check to see if this auction has already been completed. The 'auction ID' is the keccak of the opportunity transaction hash and the transaction's gasPrice – if another searcher transaction has already completed without revert using that ID, all subsequent transactions with the same ID will automatically revert. *(Note: the addition of the gasPrice to the ID is to prevent an attacker from 'blocking' the opportunity via a higher gas price.)*

4. After a few more safety checks, the PFL Searcher Auction will then call the searcher's contract with the input data specified in the call arguments. Please note that due to the potential risk of exploitation, a few mandatory safety requirements have been added, such as the EOA approval check. In a future iteration of this contract, we may relax the safety features on PFL's end and entrust searchers to handle them directly, however due to the new nature of this system we are currently going with the less gas efficient but safer route. *(Note: please see our github for examples of how to integrate the safety checks into your own contract.)*

5. Once the call of the searcher's contract has completed, the Searcher Auction then checks to see if the searcher paid the promised amount. If so, it allocates the amount paid to the validator who mined the block and then locks the auction ID so that all subsequent searcher transactions for this opportunity will revert. If not, the entire transaction reverts – undoing not just the changes to the

Searcher Auction but the changes/trades that the searcher's contract made too – and the searcher with the next highest bid will then get a chance.

Sending all of the searchers' transactions from the PFL Node to the validator was a deliberate design choice. Due to Polygon's rapid blocks and BOR's block sealing mechanism, it is likely that searchers' transactions may not land in the next block – they likely will land two blocks past the current block. Furthermore, because the PFL system does not violate or alter the validator's transaction ordering system (price/time), it is likely that even if the searchers' transactions landed just one block past the current block there would still be transactions ahead of it (due to higher gasPrice) in the block. This means that the block state at the time the transactions are executed will not be known with 100% confidence. Because of this, and because of the potential attack vector of blocking off an auction and then not paying to force a no-winner scenario, the decision was made to allow all of the searchers' transactions a chance to win. The auction ID is in place to allow the cheapest reverts possible… but I mean… c'mon.. this is Polygon. It's a fraction of a fraction of a penny.
(Note: This design choice may be situationally altered when we expand to other chains.)

Because the PFL system does not alter the validators' transaction ordering rules, ecosystem users would be subject to less damage in the event of a reorg than if more invasive MEV systems were used. This is because the new block's ordering rules would still match the reorged block's ordering rules, leading to greater likelihood of similar block state at execution for each transaction.

As an example, imagine a validator that decided to order transactions in its blocks based on time received rather than by price. If that validator's blocks were reorged by another validator who also ordered by time received, we would expect the original blocks and the new blocks to have some differences but to look somewhat similar. If, however, that validator were to be reorged by a validator who ordered transactions by gasPrice and size, we would expect the new blocks to be *completely* different. Transactions that were in the original validator's first block could end up in the last block of the reorganizing validator if they had low gasPrice. As a general rule: the less correlated the transaction ordering rules are between validators, the greater the likelihood and magnitude of change of state during a reorg for a user, and therefore the greater potential for damage to the user. Preserving the natural transaction ordering rules is a conscious design choice for PFL.

Next Stage:

Our next stage will be focused on the launch of two coins: shMATIC and PFL. The shMATIC token will be a liquid staking derivative of MATIC with MEV-augmented yield. The extra yield will come from a portion of the revenue from each PFL Auction, which will be bridged from Polygon Mainnet to Ethereum Mainnet at the end of each PFL Auction round. The PFL token will be the governance token for the PFL DAO, which will be tasked with allocating shMATIC's stake among participating validators, integrating new validators into the PFL protocol, determining revenue share agreements with PFOF protocols, and eventually the decentralization and management of PFL's infrastructure. This system will encourage new users to stake with PFL validators, increasing their block frequency and thereby increasing their total MEV revenue. More details will be announced shortly.

FAQ:

Q: Will using PFL lead to more ‘reorgs’?

A: No. The goal of the PFL system is to allow for seamless validator participation *without* requiring any changes on the validator node itself. In fact, PFL should reduce reorgs for a variety of reasons:

1. The validator node will be subject to fewer ‘spam’ transactions due to spam no longer being incentivized.
2. The validator node will now be peered with a PFL Node, which will be peered into the PFL network. This will allow for faster propagation of its own blocks, thereby reducing the likelihood of the validator’s backup not seeing its blocks and creating its own, leading to a reorg.
3. Validators are incentivized to continue to run the official BOR repo and stay up-to-date on the latest patches. The PFL system removes the incentive for validators to experiment with MEV by making modifications of their own that could lead to reorgs.

Q: Can PFL be used to censor transactions?

A: No. All transactions will still be received by all validators. We are not changing **what** the validators receive, we are only altering **when** they receive. Furthermore, we are only adjusting the receipt timing inside of the boundaries that already exist in the system. No transaction will be delayed more than was already likely. Think of the ‘Polygon FastLane’ as a way to remove randomness, disincentivize spam, increase validator revenue, and reward efficient searchers.

Q: What happens if my bundle arrives after the auction ends?

A: We suggest speeding up or otherwise altering your code. If your bot runs from just one node in Germany and the validator of the current sprint is based in Singapore, our advice would be to set up another node of your own in Singapore. All we can offer you is a speed increase, but how valuable that speed increase is depends on your own code and infrastructure.

Q: If a validator wishes to keep its own sentry node peered to the validator node, what code change will it have to make?

A: The specific change is in the **bor/eth/handler.go** file beginning at **line 481**:

Existing Code:

```
for _, tx:= range txs {
    peers:= h.peers.peersWithoutTransaction(tx.Hash())
    // Send the tx unconditionally to a subset of our peers
    numDirect:= int(math.Sqrt(float64(len(peers))))
    for _, peer:= range peers[:numDirect] {
        txset[peer] = append(txset[peer], tx.Hash())
    }
    // For the remaining peers, send announcement only
    for _, peer:= range peers[numDirect:] {
        annos[peer] = append(annos[peer], tx.Hash())
    }
```

Modified Code:

```
for _, tx:= range txs {
    peers:= h.peers.peersWithoutTransaction(tx.Hash())
    // For all peers, send announcement only
    for _, peer:= range peers {
    annos[peer] = append(annos[peer], tx.Hash())
}
```

A patch will be provided so that validators may quickly and easily implement this change on their sentries after each BOR update/upgrade. The PFL team will not maintain an independent BOR repo—instead, participating validators are encouraged to use the standard BOR repo provided by the Polygon team and to apply PFL's 'Sentry Patch' on top of the latest official release. The 'Sentry Patch' should *not* be applied to the validator nodes, although doing so would not lead to any instability.

Please see below for the sentry patch and instructions:

https://github.com/Polygon-Fast-Lane/sentry-patch

Q: I'm a validator and I don't understand any of that—can I still participate?

A: Yes, absolutely. The only prerequisites for participating are being able to install a patch on your sentry nodes (see above) and having your validator node add a PFL Node as a trusted peer. Running a validator node on polygon is extremely difficult—we're confident that if you can keep your node online then implementing the PFL protocol will be a walk in the park for you.

Q: Who runs / controls the PFL sentries? Are they centralized?

A: The nodes are maintained by the PFL team. While the nodes themselves are centralized, the overall implementation of the PFL system is **not** centralized as long as validators maintain their own sentries. We are one path to the validators, but we are not the *only* path.

Q: What happens if the PFL auction contract gets hacked?

A: We've designed the auction contract in such a way that it rarely holds substantial funds. If the contract is exploited then the validators will lose any balances in the Validator Vault, which is unfortunate, but we will identify the breach and take steps to ensure that the validators will be able to collect at the end of the next round. We encourage validators to withdraw any balances as frequently as possible to help minimize the potential damage an exploit could cause.

(Note: if any white hat hackers demonstrate the attack vector on the blockchain without telling us first, we'll encourage them to keep a percentage of the amount taken in exchange for giving us assistance in removing the attack vector in the next iteration of the contract.)

Q: Where can I see the code for the PFL Nodes?

A: At this time, the code is not available for public viewing. We wish to make it as hard as possible for clever searchers to game the system.

Q: Is this service similar to BloxRoute?

A: Although there may be some similarities in *how* we achieve our goals, the goals themselves are completely different. If anything, we are BloxRoute's opposite in that we seek to achieve transaction differentiation by slowing transactions down rather than speeding it up. We hope to work closely with them in the future on joint projects leveraging the combination of both systems.

Q: Is the Polygon team aware of this project? Do they support it?

A: While the Polygon team is aware of this project and providing guidance on design decisions, at this time they neither endorse nor condemn it. If a validator is curious about participation, we advise them to *privately* reach out to the Polygon team and simply ask them.

Q: What's more profitable, PFL or Flashbots?

A: While there are some similarities between PFL and MEV-Geth (the pre-merge Flashbots client), Flashbots post-merge has evolved into something that is no longer comparable. We have tremendous admiration for the Flashbots team and do not see them as competitors. The PFL system was designed as an alternative to the original MEV-Geth (or MEV-BOR) with a focus on achieving similar goals but with the constraints of 'no custom validator code' and 'no altering transaction ordering rules.' Because of this, 'sandwich' attacks are not facilitated by our system, lowering our profit potential. One of the benefits of our system, though, is our ability to collect revenue from relationships with PFOF protocols that otherwise would have bypassed fair validator payments. We do not know which is the larger amount.

Q: What token / currency will the bidding take place in?

A: At this time, we intend for the bidding to take place in MATIC, but are open to receive feedback and suggestions from validators and searchers.

Q: Are there any rules against a validator running MEV-Bor and participating in the PFL program simultaneously?

A: All participating validators must receive approval from the PFL team before searchers can bid on their 'FastLane' access. If a validator is upfront about usage of MEV-Bor, we will not exclude them from the auction. *Searchers are responsible for ensuring that their transactions revert if not profitable.* Our current plan is to include performance and profit metrics on the auction's front end—these metrics will help identify validators that are running MEV-Bor, their own local bot, or anything else that eats into the value of 'FastLane' access. We are not necessarily against validators performing these actions—we just want to make sure that searchers are bidding with full information. In other words, as long as validators are honest and transparent about their other MEV strategies, and searchers can price this information into their bids, then we're OK with it, too.

Q: Will searchers be able to peer directly with the validators when they win an auction? Will the searchers know the validator enode, peer ID, or IP address?

A: No, searchers will not be directly peered with the validator nor will they have any way of determining the validator's enode address, ip address, or peer ID.

Q: This isn't really MEV! The 'Miner' isn't running any custom code!

A: Technically this is 'SEV', with the S standing for either 'Sentry' or 'Speed'—your choice. But we do think this falls into the category of 'Maximal Extracted Value.'

Q: Is there any cost for validators to participate?

A: Not only is there no cost, we expect the monthly data usage of the validator nodes to go down due to two factors: spam being disincentivized and fewer transactions being directly propagated (preventing the direct propagation of duplicate transactions). This is all in addition to the proceeds from the auctions that the validators will receive.

Q: What do participating validators need to do to collect their auction proceeds?

A: Go to the PFL Validator Vault and hit that withdraw button.

Q: Can validators participate in PFL privately?

A: If there is demand for private participation and the Polygon team approves the design, then we could implement this concurrently with the planned PFOF protocol integration. That being said, the Polygon blockchain and the transactions on it are all public – it would not be difficult for a motivated developer to re-purpose PFL's open source analytical tools and detect any validators who are privately participating.

Q: What happens if the PFL sentry nodes crash? Can the validators' own sentry nodes still function even though they've been modified with the 'sentry' patch?

A: Yes—in the event that the PFL sentry nodes **all** crash but your own sentry node does not, your validator node will still be able to function just fine. Every transaction that an unmodified sentry node would have received and propagated will still be received and propagated by your patched sentry node. In other words, nothing will be censored even if the PFL sentry nodes crash.

Q: Can I have my BloxRoute gateway peered directly to my validator?

A: No, although we strongly encourage you to have it peered directly to your patched sentry.