SoK: Preconfirmations

Aikaterini-Panagiota Stouka[1], Conor McMenamin[1], Demetris Kyriacou[3], Lin Oshitani[1], Quentin Botha[2]

[1]*Nethermind Research*
[2]*Research Institute for Cryptoeconomics, WU Vienna*
[3]*Imperial College London*

October 3, 2025

Abstract

In recent years, significant research efforts have focused on improving blockchain throughput and confirmation speeds without compromising security. While decreasing the time it takes for a transaction to be included in the blockchain ledger enhances user experience, a fundamental delay still remains between when a transaction is issued by a user and when its inclusion is confirmed in the blockchain ledger. This delay limits user experience gains through the confirmation uncertainty it brings for users. This inherent delay in conventional blockchain protocols has led to the emergence of preconfirmation protocols – protocols that provide users with early guarantees of eventual transaction confirmation.

This article presents a Systematization of Knowledge (SoK) on preconfirmations[1]. We present the core terms and definitions needed to understand preconfirmations, outline a general framework for preconfirmation protocols, and explore the economics and risks of preconfirmations. Finally, we survey and apply our framework to several implementations of real-world preconfirmation protocols, bridging the gap between theory and practice.

1 Introduction

In recent years, the scalability of blockchain systems has been driven by advances within blockchain protocol design (see [1–5]), but also by out-of-protocol innovations in the transaction lifecycle. A particularly important yet relatively underexplored stage of this lifecycle is the period between when a transaction is submitted through a wallet and when it is confirmed on-chain. During this interval, users face uncertainty about if, when, and/or how their transactions will appear on the blockchain. This window of uncertainty has motivated the emergence of preconfirmation protocols, through which users can receive stronger assurances about transaction inclusion or ordering before final consensus is reached.

Preconfirmations have rapidly gained traction across blockchains, particularly on additional layers built on top of them such as Layer 2s (see Section 3). There, preconfirmations serve diverse goals ranging from improving user experience to enabling new application primitives [6–10]. For example, traders desire predictable execution in DeFi, wallets seek to minimize friction for mainstream adoption, and cross-domain systems depend on timely bridging guarantees. Despite the growing relevance of preconfirmations, the design space remains fragmented, with projects re-inventing solutions in isolation and often overlooking subtle trade-offs between trust, latency, and decentralization.

This article presents a Systematization of Knowledge (SoK) on preconfirmations. We survey the landscape of existing approaches, categorize their underlying mechanisms, and highlight both the promises and pitfalls of this rapidly evolving space. By placing current designs within a broader taxonomy, we aim to clarify their assumptions, identify unaddressed challenges, and provide a framework for reasoning about future directions.

Ultimately, preconfirmations should be seen as a structural component of blockchain systems with implications for protocol security, economics, and blockchain adoption. This work contributes a unified lens for evaluating the role of preconfirmations, setting the stage for deeper inquiry into how preconfirmations can be integrated in creating predictable, secure, and user-friendly transaction pipelines.

[1]The latest version of this document is hosted on GitHub at: https://github.com/NethermindEth/sok-preconfirmations

1.1 Organization of the SoK

The structure of the main body of the SoK is as follows:

- In Section 3, we present the fundamental concepts of blockchain that are essential for understanding preconfirmation protocols.
- In Section 4, we present the fundamental definitions and terms used in preconfirmation protocols (e.g., what a preconfirmation is, its types, and who the entities in a preconfirmation protocol are).
- In Section 5, we outline the structure of a preconfirmation protocol as a sequence of steps and describe how different types of preconfirmation protocols vary at each step.
- In Section 6, we focus on research that has been/needs to be conducted on the economics of preconfirmation protocols. This includes, for example, studies on the revenues of entities that provide guarantees to users, as well as analyses of how users should reward these entities with tips to incentivize their participation in the protocol and encourage honest behavior.
- In Section 7, we discuss the risks that can arise from preconfirmation protocols – both for the entities involved in the protocol (e.g., users requesting guarantees and entities providing them) and for the underlying blockchain protocol itself.
- Equipped with a general understanding of preconfirmation protocol concepts, in Section 8 we present several preconfirmation protocols that are currently in production. When a protocol differs slightly from the skeleton described in Section 5, we explain how it deviates from the general structure.
- We conclude in Section 9.
- In Appendix A, we discuss how preconfirmations can be formulated and provided for intents, as opposed to transaction-based preconfs.

2 Related Work

This SoK builds on previous work conducted to aggregate and organise preconfirmation knowledge. Most notably, [11] contains a collection of resources and articles related to preconfirmations that are periodically updated. Attempts have also been made to explain fundamental preconfirmation concepts [12, 13]. This SoK builds on these works, standing as a complete guide to all preconfirmation-related concepts.

3 Background

This section outlines the established blockchain concepts necessary to understand preconfirmations. Although preconfirmations are blockchain-agnostic, in this document, we restrict our discussion to preconfirmations on Ethereum.

3.1 Blockchain Fundamentals

For the purpose of this document, we consider a **blockchain** as both the protocol and data structure used to derive a replicated state machine among a set of distributed, potentially distrusting, peers/nodes. Although the blockchain protocol defines rules intended to produce a single shared state for observers who follow the blockchain protocol, conflicting states can still emerge temporarily. **Consensus mechanisms** are used to resolve these conflicts and determine the single shared state of the system. In this document, we assume blockchains derive a single shared state. Furthermore, we focus on transaction-based blockchains which form the majority of blockchains today. These blockchains advance the shared state by sequentially executing an ordered list of transactions organized in a block. Every block links to the previous block. The canonical list of transactions that derive the single shared state is denoted as the **blockchain ledger**. **Inclusion** is the process of adding a transaction to the ordered list of transactions, and **execution** is the computation of a transaction's result, thereby updating the shared state. This distinction is critical for delineating the types of preconfirmations that can be given.

Many of the most popular blockchains fall under the category Proof of Stake (PoS) blockchains. In PoS blockchains, the ledger is extended by entities known as validators, who hold stake in the blockchain

– that is, they possess some amount of the blockchain's native currency. These validators are typically selected to extend the ledger with a probability proportional to the size of their stake.

In Ethereum, which belongs to this category of blockchains, validators need to lock a minimum amount of the native currency, called Ether (ETH), to be able to participate in extending the blockchain ledger. This locked amount is referred to as **collateral** or deposit. In addition to proposing new blocks, validators also attest to (vote on) the blocks proposed by other validators. Validators attesting to or proposing a block receive rewards minted by the system. In addition, they can earn rewards through the tips of the transactions they include in the block they propose. These tips are known as **transaction fees**. If validators behave maliciously in verifiable ways specified by the rules of the protocol, they get penalized by losing a portion of their collateral – a process known as **slashing**.

Ethereum is a smart-contract enabled blockchain. **Smart contracts** can be considered as a collection of code/functions, data, and state within the global state of a blockchain [14] that enable applications to be built on and trustlessly executed by transactions on the underlying blockchain. At a high level, a smart contract consists of an algorithm and public data, both of which are created and updated through transactions originated by users and executed by validators. In the context of preconfirmations, smart contracts serve as critical components for coordinating and enforcing preconfirmations.

3.2 Ethereum's Block-Building Pipeline

In Section 3.2, we highlight the key components of the Ethereum blockchain that are critical for understanding preconfirmations. In more detail, we describe the process through which blocks are built and eventually proposed, known as the block-building pipeline.

Transaction Propagation and Mempools Users create transactions through their digital wallet applications. These transactions are then sent to nodes, which maintain transaction mempools and expose standardized interfaces called Remote Procedure Call (RPC) endpoints. The nodes perform basic validation checks on each transaction (e.g., verifying the sender's balance) before adding it to their mempool and potentially propagating it across the peer-to-peer (P2P) network. Nodes that do not propagate certain transactions are said to retain a private mempool. Validators select transactions from the mempool to construct an ordered list of transactions for inclusion in their blocks. Even if a transaction passes the initial validation checks before entering the mempool, it may still become invalid during execution if, for example, a prior transaction in the mempool drains the sender's balance.

Proposers and Attesters: Selection and Responsibilities Ethereum progresses in epochs each consisting of 32 slots of 12 seconds. During every slot, there is one eligible validator selected to construct, sign, and propose a block. This validator, known as proposer, is pseudo-randomly selected from the set of registered validators. Beyond the rewards earned by following the Ethereum protocol (see Section 3.1), the proposer can extract additional value – known as maximal extractable value (MEV) [15] – by intentionally manipulating the list of transactions proposed within the slot. For example, the trade prices offered by a type of decentralized exchange called Automated Market Maker (AMM) change after every trade. A proposer can profit from a user's buy order by including their own buy order in front of the user's buy order – **front-running** the buy order – and/or including their own sell order behind the user's buy order – **back-running** the user's buy order. Together, front-running and back-running the same transaction is known as **sandwiching** [15]. Apart from proposing blocks, validators are elected into committees that vote on the validity (as specified by the Ethereum protocol) of proposed blocks – a process known as attesting.

The task of constructing a block can be unbundled from the tasks of signing and proposing the block. This is known as Proposer-Builder Separation (PBS) [16]. PBS enables proposers to delegate the construction of their block to specialized entities known as builders. In PBS, proposers select the transaction list to be proposed, referred to as the `ExecutionPayload` (see [17]), by running an auction among builders. Builders bid in the auction, competing to have their own transaction lists proposed, with the winning payment paid to the proposer. Note that when the proposer selects a transaction list, they can only see a commitment to it, not the entire list. The complete list is revealed only after the proposer has made their selection. Another entity, known as the relayer (not to be confused with preconfirmation gateways introduced later), provides fair exchange [18] of the committed transaction list between proposers and builders. MEV-Boost [19], an implementation of PBS for Ethereum, is the primary source of blocks on Ethereum with over 90% of blocks being sourced through MEV-Boost at the time of writing [20].

Proposer Lookahead The proposer lookahead specified by the Ethereum protocol provides advance notice to validators of their proposing duties. According to EIP-7917 [21] which is a candidate for inclusion in Ethereum's upcoming Fusaka upgrade [22] the proposer schedule for epoch $N + 1$ becomes known and accessible – e.g., via smart contracts – at the start of epoch N. This lookahead is critical for enabling preconfirmations, as it allows future proposers to begin offering preconfirmations before their assigned slot arrives.

3.3 Ethereum Scaling Solutions

To enable a decentralized network to keep up with and reach consensus on state updates in blockchains, throughput is more limited compared to centralized solutions where a single trusted party maintains and updates transaction records. To increase throughput while still preserving key features of the blockchain (e.g., decentralization), scaling solutions in the form of additional layers have emerged. The original blockchain, in our case Ethereum, will be denoted by layer 1 (L1), and its canonical list of blocks with transactions by **L1 blockchain ledger**. The other layers will be called layer 2 (L2s), L3s, or generally LNs for some $N > 1$. These LNs are execution environments that typically move one or more of the L(N-1) resources outside of the critical path of L(N-1) state progression. At the time of writing, the majority of LN users exist at L2, so without loss of generality, we will focus on L2s when discussing these blockchain scaling solutions. To the best of our knowledge, all statements about L2s and their relationship to the L1 can apply to any LN and its respective L(N-1).

Deploying as an L2 instead of L1 removes the need for the L2 to establish its own consensus protocol or dedicated economic security for securing state transitions (although these features can be added). L2s can tap into the user base of the L1, allowing L1 users to opt in to locking their funds and using them within the L2. Many L2 solutions exist, including rollups, validiums, and Plasmas [23]. As in L1, L2 state progresses through proposers proposing blocks, establishing a canonical list of blocks, and some eventual finalization of these blocks/commitments to these blocks on L1 to form the **L2 blockchain ledger**. L2 state is then derived from this L2 blockchain ledger. L2 solutions can differ in how they finalize L2 state or in the type of data they post to L1. However, these details are beyond the scope of this document. With respect to preconfirmations, two key L2-specific concepts are relevant:

1. **L2 proposer selection:** A key factor impacting preconfs is whether an L2 is based or non-based.
   (a) **Based L2:** Delegates block proposing to the proposers of L1 [24, 25].
   (b) **Non-based L2:** Uses a dedicated proposer election mechanism. Within this category of non-based L2s, a key differentiating factor is whether there is one or many proposers. When there is only one proposer, preconfirmations are greatly simplified, at the cost of centralization, and the negative effects that this brings (e.g., single point of failure, monopolization). These single proposer non-based L2s are, rather appropriately, sometimes referred to as centralized sequencer L2s. Some examples of non-based L2s include Optimism [6], Arbitrum [7], ZKsync [8], and Starknet [26].
2. **L2 governance:** Governance is a protocol for determining how a set of designated entities can modify some or all of the blockchain protocol specification. These permissions may stem from being a founder, a stakeholder, or being elected within the protocol. Governance can have many roles, including upgrading the chain, adjusting protocol parameters, or selecting entities for certain privileged roles – such as proposer or overseer (introduced in Definition 12) – within the blockchain protocol (see [27–29]). In Ethereum, the rules governing how blockchain ledger progression and state derivation change via social consensus among validators. This governance-related consensus occurs outside of Ethereum's protocol rules.

4 General Definitions and Concepts of Preconfirmations

With this background in hand, we are ready to introduce and discuss preconfirmations. At a high level, a preconfirmation in the blockchain context is a promise/commitment that the blockchain ledger will satisfy some property at some point in the future. The overwhelming majority of preconfirmations being considered, both in production and development, involve promises about the inclusion of a specific transaction or transactions in the transaction list of a block that will eventually be part of the blockchain ledger.

To formally define preconfirmations, we use the concept of logical predicates. A predicate function determines whether a given input to the function possesses a specific property. In the blockchain context, a predicate function taking as input a blockchain ledger, as formed at a specific point in time, e.g., a slot in Ethereum, would return true if and only if the property defined by the predicate function is satisfied by the blockchain ledger. In this context, we define a preconfirmation as follows:

Definition 1. *For a given blockchain, a* ***preconfirmation (preconf)*** *is a commitment to some predicate function* `f` *such that: there exists a point in time – after the preconfirmation is issued – such that if the blockchain ledger at that time is given as input to the predicate, it will return true.*

For preconfs, the commitment can be considered as a promise. The exact structure, significance, and strength of the commitment depends on the individual protocol, and entity providing the commitment. For example, a commitment from an upcoming proposer is different to a commitment from an entity who does not control proposal rights – the proposer can directly influence the blockchain ledger. A commitment from an entity who is financially punished for not satisfying the commitment is different to a commitment from an entity who faces no repercussion for not satisfying the commitment. Who can offer preconfs, the relative strength of preconf commitments, preconf incentives, and preconf enforcement mechanisms are expanded on throughout the remainder of this document.

Additionaly, we define **conditional preconfs** as preconfs for which the corresponding promise is valid only if some pre-defined condition is met. The condition is also a predicate that takes as input the blockchain ledger.

Definition 2. *A* ***conditional preconf*** *for a promised predicate* `f` *and a conditional predicate* `c` *is a commitment that if the condition* `c` *is satisfied by the blockchain ledger in the future, then the predicate function* `f` *will also be satisfied by the blockchain ledger at some point after the preconf is issued.*

For example, a conditional preconf could promise that a transaction will be included in the blockchain ledger if a specific block builder is selected to build the block for a given slot. This is the type of preconf builders can offer in mev-commit [30]. Another example of a conditional preconf could be a promise that a transaction will be included in the blockchain ledger if the base fee (i.e, minimum required fee payable to the protocol) in a given block is below some threshold.

Preconfs, whether conditional or not, are only meaningful if the committed predicates can be satisfied. To describe this notion of predicate satisfaction, we introduce the concept of fulfillment, which holds for all preconfs, including conditional preconfs, as follows:

Definition 3. *A preconf is considered* ***fulfilled*** *if and only if the promised predicate specified by the preconf is satisfied by the blockchain ledger. Satisfying a preconf's promised predicate is equivalent to fulfilling the preconf.*

Preconfs can be implemented in many different ways. Key factors that distinguish preconf implementations include who is providing the preconf, the predicate involved in the preconf, and blockchain governance, if it exists. The type of preconf determines which applications and users benefit from them. We distinguish and explain these varying preconf implementations throughout this document. In this document, we will focus on transaction-based preconfs, defined in the following section.

Other types of preconfs not requiring commitments to specific transactions or transaction sequences are possible. These are generally referred to as intent preconfs. An example of an intent is a request to exchange a specific amount of token A for some minimum amount of token B without a specific user transaction. Although we focus on transaction-based preconfs in the main part of this document, we provide some information on intent preconfs in Appendix A.

4.1 Transaction-based preconfs

Transaction-based preconfs are preconfs with predicates that depend on the existence of a transaction or sequence of transactions in the blockchain ledger. Some examples of this type of predicate are the following:

1. A given transaction is included in the blockchain ledger in an arbitrary position.
2. A given transaction appears at a specific position in the blockchain ledger, e.g., after a specific sequence of transactions.

Below, we formally define different types of preconfs based on the structure of their predicates.

The main types of preconfs studied in the literature are **inclusion** and **execution** preconfs (see [12, 31–33]), defined as follows:

Definition 4. *For a given blockchain and transaction* tx_0*, an* ***inclusion preconf*** *is a preconf where the predicate function returns true if* tx_0 *is included in the blockchain ledger given as input.*

Generally, inclusion preconfs are useful when the primary concern of the user or application is ensuring that the transaction is included. Use cases include simple transfers or posting sequences of L2 transactions (batches) to L1. However, inclusion preconfs provide no guarantee of how a transaction executes, which becomes problematic for transactions seeking to act on **contentious state** [32, 33].

Definition 5. ***Contentious state*** *is a state element upon which concurrent transactions attempt to act, creating a race condition among transactions to act on the state element first. This state element may be derived directly from the existing blockchain ledger, or from transactions that are expected to be included in the blockchain ledger (e.g., preconfed transactions building on the blockchain ledger)*

Examples of contentious state include arbitrage and liquidation opportunities on a blockchain. For a transaction trying to act on such opportunities, or contentious state in general, execution preconfs are more comprehensive commitments than inclusion preconfs [34].

Definition 6. *For a given blockchain, transaction* tx_1 *and sequences of transactions* stx_1*, an* ***execution preconf*** *is a preconf where the promised predicate is a function that returns true if* tx_1 *is included in the blockchain ledger immediately after* stx_1*.*

We refer to transactions that have received inclusion preconfs or execution preconfs as inclusion preconfed and execution preconfed transactions, respectively. When the context of inclusion preconf or execution preconf is clear or unimportant, we simplify this terminology to just preconfed transactions.

Providing execution preconfs is more complex and comes with additional block-building constraints compared to providing inclusion preconfs. Execution preconfs can only be issued either (i) for the current slot, or (ii) for a future slot, provided that the transactions to be included in all preceding slots have already been preconfed. This is because execution preconfs require knowledge of the sequence of transactions in the blockchain ledger preceding the transaction to be preconfed.

4.2 Preconfers

Preconfs are provided by preconfers. The entity that can perform the preconfer role depends on the layer and sequencing model considered.

Definition 7. *A* ***preconfer*** *is an entity that provides preconfs, with* ***preconfing*** *being the act of issuing a preconf by a preconfer.*

Preconfers have varying abilities to fulfill preconfs. Block proposers – whether for L1 or L2 – are candidates for preconfers. This is because proposers can enforce fulfillment of their preconfs by virtue of their control over block construction and proposal. Although proposers have the option to act as preconfers – since they are entities capable of fulfilling preconfs – doing so may increase their operational complexity. Proposers can overcome this complexity by delegating their preconfing duties to gateways.

Definition 8. *In the context of preconfs, a* ***gateway*** *is a specialized entity that preconfs on behalf of a proposer.*

As gateways lack direct control over block proposal/signing, preconfs sourced from gateways depend on the ability of the gateway to enforce fulfillment by the proposer [35–37]. How exactly gateways fulfill preconfs to the proposer is the focus of Section 5.5.

Another preconfer candidate is builders in the current MEV-Boost PBS paradigm. Builders as preconfers is one of the core focuses of the mev-commit preconf protocol [30]. Recall that in MEV-Boost, builders construct blocks and bid for the right for their block to be selected for proposal by the proposer. Given a builder must construct a block without knowing whether or not the block will be proposed, preconfs offered by MEV-Boost builders are conditional on preconfing builders winning the MEV-Boost auction [38, 39]. That being said, at the time of writing, three builders typically win more than 90% MEV-Boost auctions [20]. Therefore, acquiring preconfs from these builders may be a sufficient commitment for some users and applications.

5 The Preconfirmation Pipeline

This section analyzes the general flow of a preconf protocol and the roles of its key participants. The flow begins when a candidate expresses interest in joining the protocol as a preconfer. Upon meeting the protocol's registration requirements, the candidate becomes eligible for election as a preconfer. The protocol then enters a repeated interaction phase involving users who submit preconf requests and preconfers who respond to them with preconf responses[2]. The final phase of preconf protocols focuses on the fulfillment of preconf promises and the on-chain publication (inclusion in the blockchain ledger) of any preconfed transactions. An optional addition to the flow concerns the punishment of preconfers who behave maliciously. Fig. 1 illustrates a general overview of the preconf protocol flow.

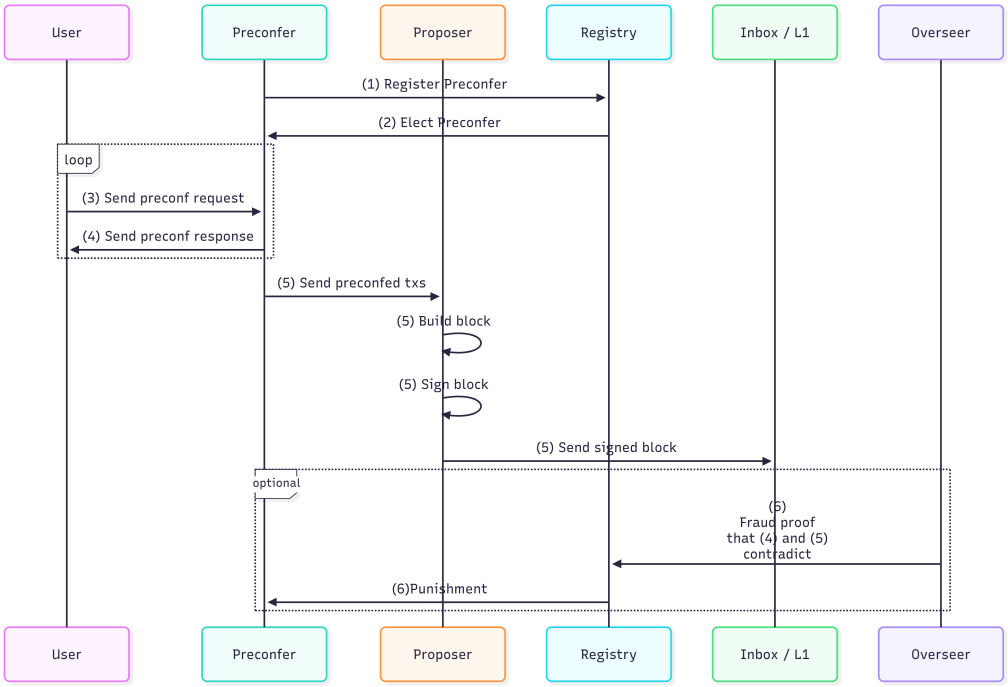

Figure 1: Preconf Protocol Flow.

The remainder of this section goes through this preconf protocol flow, which we decompose into the six steps of preconfing.

5.1 Step 1: Preconfer Registration

The first step in the preconf protocol flow is preconfer registration (see Fig. 1). Preconfer registration involves candidates signaling that they want to become a preconfer. For this signaling to result in a successful registration, the candidate must meet all applicable eligibility criteria. This registration is typically managed by a smart contract known as the preconf registry, for which we give details in Section 5.1.1. Each preconf registry has the freedom to set its own criteria for its preconfers. Some example criteria include:

- Proof that a registrant is a proposer. Generally, preconfers must have the ability to perform inclusion/execution of a preconfed transaction. Some preconf protocols may insist that preconfers be proposers of the underlying blockchain protocol. Specifically, registrants may need to prove:
  - Membership of the L1 proposer set for L1 and based L2 preconfs.
  - Governance committee approval for non-based L2 preconfs.
- Deposit of some minimum collateral amount that may be subject to slashing [40, 41].

[2]In some preconf protocols, this phase may involve a trusted party, known as overseer, who signs the preconf responses along with the preconfer and penalizes them in real-time if the preconfer deviates from the protocol's instructions. This is a real-time punishment that is thoroughly discussed in Section 5.6.3.

- That a registrant is not blacklisted see [38].

5.1.1 The Preconfer Registry

In this section, we present what the smart contract preconf registry is usually responsible for, and we elaborate on the minimum collateral an entity may need to deposit to be registered as a preconfer. In Section 3 we mentioned that every validator needs to deposit collateral to participate in block proposal and attestation in Ethereum. Here, we discuss what slashable collateral is in the context of preconf protocols.

Definition 9. *In the context of preconfs and a specific preconfer,* ***slashable collateral*** *is an amount of tokens that can be slashed in the case of that preconfer's misbehavior.*

With the definition of slashable collateral in the context of preconf protocols established, we can now present the definition of the preconf registry.

Definition 10. *A* ***preconfer registry*** *is a smart contract wherein entities can opt in to preconfing, registering themselves as preconfers. This smart contract can: (i) verify whether a candidate meets the applicable eligibility criteria; (ii) hold collateral and enforce monetary penalties on preconfers deemed to have acted maliciously in accordance with contract rules; and (iii) determine the preconfer schedule.*

Preconfer registries [40, 42] can play an important role in their respective blockchains. For L2s, preconfer registries can be tightly coupled with the proposer election mechanism, including having full control over proposer election (for more details, see Section 5.2).

Preconfer registries where proposers can opt in to preconfing are expected to require preconfers to deposit some slashable collateral. In such protocols, the preconfer registry would be entrusted to manage the collateral on behalf of preconfers. This includes slashing of the collateral when deemed necessary by any registry-specified rules, known as slashing conditions [41, 43]. These slashing conditions should in theory reflect the behaviours that are desired/acceptable from a preconfer. Slashing of collateral, and alternative punishment methods are thoroughly discussed in Section 5.6. Note that in the case of conditional preconfs, the preconfer should only be eligible for penalization if the preconf condition is satisfied but the promised predicate is not satisfied (see [30] for more details on conditional preconf slashing).

Note that any preconfer collateral directly locked up in a preconfer registry cannot be reused for other purposes. Therefore, protocols may want to rely on third-party restaking platforms for sourcing slashable collateral, such as the EigenLayer platform [44]. Restaking allows collateral that secures the underlying blockchain to be reused for multiple purposes, making it an attractive and capital-efficient way to source collateral. Of course, repurposing slashable collateral for multiple protocols introduces risks related to stake being slashed in multiple protocols at once. Most, if not all collateral-dependent protocols design slashing rules assuming all of the deposited collateral is available for slashing. As such, if collateral in one protocol can be reduced without slashing or withdrawing from that protocol, the implied security of the collateral is weakened. For more information on the trade-offs of re-staking collateral, see [45].

Lastly, in an attempt to make preconfing accessible to solo-stakers with limited resources, collateral delegation has been proposed [40]. Collateral delegation allows delegates to delegate stake to a delegatee who wishes to act as a preconfer but does not possess the required collateral. In such a setting, if a delegatee violates a slashing condition, it would be the delegates' stake that would be slashed. Sharing of preconfer rewards would be one reason for delegating collateral, although delegation comes with significant risk for delegates (see Section 7)).

Universal Registry Contract At the time of writing and to the best of our knowledge, all preconf protocols implement and deploy their own standalone registry contract. The Universal Registry Contract (URC) [42, 46] aims to address the fragmentation risks that standalone preconfer registries inherit. The URC aims to be a generic open-source contract that removes the need to create and maintain individual preconfer registry contracts for each preconf protocol. The URC's promise is that individual protocols can customize their slashing and collateral requirements outside of the core URC. The URC aims to standardize:

- Preconfer registration and collateral posting.
- How slashing condition violations are reported and enforced. Specifically, the URC defines a common interface and process for enforcing slashing, without tying itself to any specific slashing

logic.

With a single shared preconf contract like the URC, there is an easier path for preconf adoption for all stakeholders. One preconfer registry simplifies the tasks of assessing and comparing preconf protocols, slashing conditions, preconf guarantees and preconf risks (see [47] for more details on the risks and benefits of the URC specifically).

5.2 Step 2: Preconfer Election

Preconfer election is a crucial part of most preconf protocols. This section examines how preconfers are elected. As already mentioned, in many L2s, the preconfer registry and proposer election are aware of each other. In non-based L2s [6–8], the two tasks are handled by the same smart contracts. In contrast, for the L1 and based L2s, the proposer election mechanism is typically not aware of the preconf registry, or even that preconfs are taking place. In all cases, the preconfer registry must be consulted to identify the schedule of preconfers, and for which slots preconfs can be provided. The preconfer schedule depends on (i) the type of the preconf (inclusion or execution) and (ii) how blockchain proposers delegate block-building.

How the type of preconf affects the preconfer schedule

- **Execution preconfs:** If the current proposer of a blockchain is not a preconfer, then execution preconfs cannot be credibly provided, even if future proposers in the proposer lookahead are preconfers. This is because the current proposer has control over the next blockchain ledger update. As such, future preconfers cannot know the blockchain ledger on which they will act. Note that this does not apply to conditional preconfs that can be provided, for example, by a builder in MEV-Boost [19], even if the builder cannot be certain they will win the block-building auction.
- **Inclusion preconfs:** Such preconfs can be issued by many, if not all preconfers. This is because the fulfillment of inclusion does not depend on the current state of the blockchain ledger.

How block-building delegation affects schedule

- **L1 and based L2s proposer preconfs:** Proposers are determined via the L1 proposer election mechanism. L1 preconfer registries – while not responsible for electing L1 proposers – can still determine the preconfer schedule, e.g., through EIP-7917 [21]. The preconfer schedule can be determined by finding the intersection of the registered preconfer set and the proposer lookahead, and taking into account whether the preconf is of type inclusion or execution. For based L2s, if the lookahead yields no eligible registered preconfer, a fallback mechanism (e.g., drawing from a static whitelist or randomly sampling from the registry) can be used to prevent gaps in preconf coverage (see [48]).
- **Non-based L2s proposer preconfs:** Within non-based L2s, there is a large proposer-preconfer election design space. The preconfer registry can either read from the outputs of a standalone proposer election mechanism as in L1 and based L2 preconfs, or be integrated into the proposer election mechanism.
- **Block builder preconfs:** When block builders issue preconfs, the preconfer schedule cannot be predetermined but rather depends on the outcome of the block-building auction. In this case, all preconf-registered block builders can issue conditional preconfs, conditioned on the corresponding block builder winning the block builder auction. Which of the builders wins the auction and thus is responsible for fulfilling the preconfs is then determined after preconfing has taken place. Given the probabilistic nature of block-builder preconfs being fulfilled, it may make sense for users to source conditional preconfs from more than one registered block builder [39].

5.3 Step 3: Preconf Request

As soon as a registered preconfer is elected, users who wish to receive preconfs can begin submitting preconf requests (see Fig. 1). Depending on the preconf protocol, users can route preconf requests to the preconfer via P2P [30] or through RPC endpoints [49, 50]. The request structure can vary from protocol to protocol. Preconf requests may include the following:

- **Transaction(s):** Requests are expected to include the transaction(s) for which the user wishes to receive a preconf[3].

- **Preconf tip:** The preconf tip is the compensation to be paid to the preconfer for the preconf. In some preconf protocols, the tip is atomically bundled with the transaction that requests a preconf. This means that the preconfer cannot redeem the tip without including the transaction in the blockchain ledger (for more details, see [55, 56]). An idealized tipping mechanism protocol should ensure that a preconfer is only rewarded with a preconf tip if both the preconf is fulfilled, and that preconfer is responsible for fulfilling the preconf.

  Regardless of how the tipping mechanism is implemented, the preconf tip stands a key incentive for preconfers to provide preconfs. As such, pricing models for preconf tips are crucial. Preconf tip pricing is handled in detail in Section 6.2.

- **Preconf type:** Provided that the protocol and/or the preconfer support both inclusion and execution preconfs, users should specify which type of preconf they want.

Requests may also contain more nuanced parameters:

- **Deadline:** A deadline parameter sets the latest possible slot/time after which a preconf request should be considered void, and its transaction(s) no longer includable by the preconfer [31].

- **Tip decay/escalator mechanism:** The preconf tip may need to change during the lifetime of a preconf request to reflect the preconf requester's urgency for the preconf to be fulfilled. A tip decay or escalator mechanism can achieve this. Tip decay mechanisms can incentivize individual preconfers to respond in a timely fashion to maximise the tip revenue they receive [57]. Tip escalators have also been discussed for regular transactions in [58]. Tip escalators depend on multiple entities competing to fulfill a transaction or preconf first to collect the fee/tip. This makes such a tip escalator more appropriate for inclusion and conditional execution preconfs, where multiple preconfers may have the ability to fulfill a preconf.

- **Preconf penalty:** Instead of relying on a predetermined penalty system, it is possible that users define penalties for preconfer faults related to a specific request through a request parameter [59,60]. The idea is to allow users and preconfers to mutually agree on a level of cryptoeconomic security where a "one-size-fits-all" approach does not suffice.

- **Latest preconfed state:** Protocols can allow users to specify a commitment to a transaction sequence that includes: (i) the transactions present in the blockchain ledger at the beginning of the current blockchain slot, and (ii) any transactions that have been preconfed. This gives users control over the transaction sequence their request acts on. [56]

- **Privacy preferences:** Privacy-preserving techniques may be applied to preconf requests to hide key request data from preconfers before they issue the preconf. Techniques include public-key encryption, or trusted intermediaries as they used in MEV-Boost today [50, 56, 60, 61].

5.4 Step 4: Preconf Response

After observing a preconf request and verifying its validity, the preconfer must then choose whether to provide a preconf or not. This section describes the process of responding to preconf requests. When a preconfer responds to a preconf request, the digital artifact sent to the user is the preconf. Generally, preconfers should be incentivized by the protocol to provide timely preconfs to users in order to improve user experience. To incentivize timely preconfs, a suitable tipping mechanism is required (as discussed in Section 5.3 and Section 6.2), along with a protocol that ensures the preconfer receives the tip only if the preconf is provided no later than the agreed-upon time (as discussed in Section 5.4.1).

The response to a preconf request stands as a signed commitment to include or execute the preconfed transaction. In many preconf protocols, preconfs also serve as evidence for punishing dishonest preconfers who fail to fulfill stated commitments (e.g., in URC described in Section 5.1.1). Preconfs must be signed by the preconfer. Additionally, preconfs may include (but are not restricted to):

[3] Other preconf protocols allow users to request intent fulfillment [51] or ahead-of-time commitments to include a transaction where the executable transaction has not been specified yet [52–54]. These latter preconfs are sometimes referred to as block-space commitments.

- **Unique identifier to a preconf request:** A preconf must contain a unique mapping to the request it is directed at. A hash of a signed request is such a mapping [62].
- **Block number containing the requested transaction or validity period:** Some preconf protocols may require the preconfer to disclose the number of the block that will contain the requested transaction, or commit to the period during which the preconf will be fulfilled [50].
- **Latest preconf state:** The preconfer may be required to provide a commitment to the the latest preconfed state (explained in Section 5.3) at the instant when the preconf was provided.
- **Commitment to additional slashing conditions:** Certain preconf protocols may require a preconfer to commit to additional slashing conditions – e.g., as specified by the associated preconf request.
- **Timestamp:** The time at which the preconf was issued.

Apart from the content of a preconf, the frequency of preconfs can also vary. Preconfs can be published in several ways:

- Immediately upon commitment to fulfill the preconf, known as streaming (for example, as discussed in [35]). Requiring the preconfer to publish a preconf before being allowed to continue preconfing could act as a disincentive to accept tips without providing timely preconfs.
- At regular time intervals.
- In batches – e.g., after the preconfer has committed to preconfing ten transactions, the preconfer publishes a preconf batch including these ten transactions.
- Upon request, granting access to previously issued preconfs.

The frequency with which preconfs are expected, enforced, and actually provided has implications for all entities in a preconf protocol. For example, streaming preconfs is resource intensive for a preconfer, both from an infrastructure and a pricing perspective (see Section 6.2 for more details on how preconf frequency affects preconf pricing and preconfer revenue). On the other hand, streaming preconfs may offer certain users an improved user experience compared to batch-preconfing (see [35]). In turn, this improved user experience could translate into increased preconf tips for the preconfer.

5.4.1 Fair-Exchange of Preconf Requests and Responses

Equally important to the core concepts of preconf requests and responses is the fair exchange of these requests and responses.

Definition 11. *The **fair exchange problem** states that during a mutual exchange of items between two entities, it is vital to guarantee that both entities or neither entity receive the expected item. No entity should have the ability to gain an advantage by misbehaving or quitting the transaction prematurely [18, 63].*

In the context of preconfs, users should have some expectation of how and when preconfs are received after submitting a request. This expectation depends on many factors, including but not limited to the type of preconf requested, the tip being paid, the cost to provide and fulfill the preconf, the relative demand for transactions/preconfs at the time the request was sent, and the value that a preconfer can extract from the preconf beyond the tip. Crucially, a preconf user's expectation has a time component built in – users typically want preconfs sooner rather than later. This makes the fair exchange of preconfs more complex than eventual fair exchange of request and response. In [38], the concept of timely fair exchange is introduced to explain this user preference in the context of preconf protocols. Previous work on fair exchange proved that it is impossible to enforce fair exchange without a trusted third party [64]. As such, preconf protocols need to introduce some form of trusted third party in order to provide meaningful guarantees of timely fair exchange. Note that a smart contract can serve as a trusted party – assuming that the L1 satisfies certain security guarantees and the contract is correctly implemented – but only in cases where there exists programmable proof of the preconfer's misbehavior that the contract can independently verify. For example, the smart contract must have access to the proposer lookahead mechanism to confirm whether a specific entity signing a preconf was indeed elected as a preconfer for a given slot.

Definition 12. *In the context of preconfs, an* ***overseer*** *is an entity or set of entities that is trusted to observe the actions of preconfers, and provide signals to the protocol in the case of a preconfer's deviation from protocol rules or expected actions. These signals can include proofs of deviation, although some deviations may not be provable, depending instead on overseer trust. Overseer signals may be communicated on-chain through protocol smart contracts, or off-chain through P2P communication layers.*

Several overseer election mechanisms have been discussed. The primary distinction of these protocols is based on whether the overseer can signal preconfer misbehavior in-protocol or not. In-protocol signaling allows for explicit preconfer punishment, something which is discussed in Section 5.6. Such overseers can be elected through governance and/or be required to hold some large amount of a blockchain's native token for which the preconfs are being provided. Both of these mechanisms align overseers with the success and reliability of the preconf protocol.

For overseers who are expected to signal preconfer deviation out-of-protocol, several solutions have been discussed to date. In [38], users are collectively identified as a suitable overseer in certain preconf protocols. Users can use the threat of withholding orderflow from a misbehaving preconfer to incentivize preconfers to follow protocol rules. The effectiveness of orderflow withholding as a threat depends on the expected profit of a preconfer from preconfs. This in turn is influenced by the frequency of being elected as a proposer/preconfer and the availability of block-space (see [65]).

Wallet providers operating as out-of-protocol overseers on behalf of users is also discussed in [38]. As wallet providers are more sophisticated than a typical blockchain user, wallet providers may be better suited to identify preconfer deviations. In a similar vein, [35] presents the concept of gateways (see Definition 8) and relayers (see Section 3.2) acting as overseers on behalf of users by monitoring preconfer behavior and withholding requests or tips. It is important to note that overseers and preconfers should be distinct entities in each preconf protocol. For example, a relayer can assume the roles of a preconfing gateway and an overseer, but not in the same preconf protocol.

5.5 Step 5: Fulfillment

It is a preconfer's ability to fulfill a preconf that provides a preconf protocol with utility. This section focuses on the different mechanisms that preconfers can use to fulfill preconfs. These fulfillment mechanisms depend on many factors, including but not limited to:

- What layer of blockchain the preconfs apply to.
- How much of a block is being preconfed, and how any remaining block-space is filled when only a percentage of a block is being used for preconfs.
- Whether the preconfer is a proposer or gateway.

These tradeoffs are examined in the following Fulfillment Tradeoffs subsections.

5.5.1 Fulfillment Tradeoffs: Based vs Non-Based

For L1 and based L2 preconfs, the L1 proposer controls proposal of L1 blocks, and so can always propose an L1 block containing any and all preconfs. For non-based L2s with a single proposer, fulfillment is also controlled by the proposer themselves.

In the case of non-based L2s with mutliple proposers, the rules of the underlying L2's blockchain ledger dictate a preconfer's ability to fulfill preconfs. In some non-based L2s, it may be sufficient to just propagate the block to the P2P layer for consensus. However, some non-based L2s require that an L2 proposer's blocks must appear in the L1 ledger by a certain proposal deadline to be considered valid. In this case, an L2 preconfer can only fulfill their preconfs if an L1 proposer includes the corresponding L2 blocks on L1 by the proposal deadline, which is not guaranteed.

5.5.2 Fulfillment Tradeoffs: Full Block Preconfs vs Partial Block Preconfs

In the case where a full block is being used for preconfs, a preconfer can build the block themselves and ensure fulfillment of all preconfs. In the case where only a partial block is used for preconfs, and the final block is built at proposal time, potentially including regular transactions, there must be a mechanism in place to ensure the final block fulfills all preconfs. A simple algorithm for achieving this is including

all preconfed transactions first in a block, followed by the remainder of the block. However, when any of these preconfed transactions are inclusion preconfs, this simple algorithm is likely sub-optimal for a proposer with regard to capturing block-building revenue. As inclusion preconfs can be placed anywhere in a block, a proposer can strategically place these preconfs to maximize their MEV revenue.

That being said, this problem of maximizing the value of blocks given some constraints has parallels to the ordered knapsack problem [66], which requires significant sophistication to solve for the knapsack sizes and number of items that modern blockchains provide. This problem has been referred to as the Online Block Packing problem [67]. To allow proposers to outsource this complex optimization problem while still receiving revenue from the solution, the Constraints API [68] has been proposed as a tool to allow proposers to offer preconfs (or apply arbitrary constraints on a block), and then outsource the building of a block satisfying all preconf predicates to a set of builders, analogously to how proposers outsource block-building through MEV-Boost [19].

5.5.3 Fulfillment Tradeoffs: Proposer vs Gateway

Proposers are perfectly placed to provide preconfs, given their ability to include preconfs in their own blocks. Gateways on the other hand require additional machinery to ensure that a proposer proposes a block which includes all preconfs provided by the gateway. The relationship between the proposer and gateway is crucial to the mechanism that must be used to fulfill gateway preconfs.

If the gateway trusts the proposer, a straightforward message-passing algorithm suffices to fulfill preconfs from gateway to proposer to the blockchain ledger. Depending on whether or not the gateway is also tasked with sourcing the block for the proposer will dictate the time at which the preconfs must be sent back to the proposer, or whether the proposer even needs to observe the preconfs. In the case where the gateway also sources the proposer's block, for example, if the gateway also runs MEV-Boost, the proposer can simply sign the block header and have the gateway propagate the final block to the P2P network, as in MEV-Boost.

If a gateway does not trust a proposer, the gateway would require cryptoeconomic guarantees that the proposer will propose a block that includes all preconfs provided by the gateway. Cryptoeconomic guarantees from the proposer are especially important when the gateway is liable to be punished in the case where preconfs are not fulfilled. Such a guarantee could include shared punishment when preconfs are not fulfilled, or the existence of a mutually trusted overseer to arbitrate faults [38, 69].

In [70], it is proposed to penalise entities based on the specific reason a preconf was not fulfilled. If no block was published for the relevant slot, the proposer is penalized. Conversely, if a block was proposed but the preconf was still not fulfilled, the gateway is held responsible and penalized.

5.5.4 Designing Robust Fulfillment Mechanisms

In any preconf protocol, it is possible that a preconfer fails to fulfill their preconfs. Although this non-fulfillment may be malicious, non-fulfillment of preconfs can also be unintentional. Some thought has been given to protecting against unintentional failures to fulfill preconfs. The authors of [71] suggest that preconfers can collaborate with subsequent preconfers through revenue sharing to preconf the same transaction sequences in the event that the original preconfer fails to fulfill the preconfs themselves. This approach is termed **chaining** preconfs. Although the exact incentives and guarantees of chaining are unexplored, chaining provides a protocol design direction to protect preconfers and their users from the risks of failing to fulfill preconfs.

5.6 Step 6: Preconfer Faults & Enforcement Mechanisms

Preconfer faults are part-and-parcel of preconfing, standing in the way of the ideal notion of perfectly-reliable preconfs. Preconf enforcement mechanisms are an important tool in disincentivizing faults and ensuring that preconfers either fulfill preconfs according to any preconf restrictions, or are punished.

5.6.1 Preconfer Faults

There are various faults that a malicious or negligent preconfer can commit that jeopardize the smooth operation of a preconf protocol. Fig. 2 presents a decision tree that categorizes preconfer faults. These preconfer faults are described in the remainder of this subsection.

Definition 13. *A **safety fault** occurs when the proposer publishes a block which violates a preconf predicate to which the preconfer committed.*

Note that a safety fault will never be committed by an honest preconfer who is also a proposer. In such cases, the punishment is expected to be severe [72].

Unlike safety faults, preconf liveness faults do no clearly apply to all blockchains. We define liveness faults for L1s and based L2s in-line with the seminal work on based preconfirmations [72].

Definition 14. *A **liveness fault for L1 and based L2 preconfs** occurs when an L1 proposer does not publish an L1 block for a particular slot, meaning all preconfed transactions for that slot are not included in the respective blockchain ledger.*

If we attempt to apply a similar definition to non-based L2s, preconf liveness faults are in many cases equivalent to a liveness failure of the underlying L2 – which is conceptually different to the well-understood liveness fault of L1s and based L2s. As such, we do not consider liveness faults with respect to non-based L2s.

Unlike safety faults, liveness faults can be accidental and non-malicious, for example, due to a power outage or internet downtime. Therefore, preconf protocols may prefer a lighter penalty for liveness faults, as proposed in [72], or to introduce fulfillment mechanisms that mitigate the risk of preconfers being slashed for accidental violations, as discussed in Section 5.5.4.

Although idleness faults are not well-documented, it is natural for some preconf protocols to require and enforce that preconfers issue preconfs under certain conditions, e.g., preconfing all valid requests, or preconfing all requests paying tips greater than a certain amount.

Definition 15. *An **idleness fault** occurs when a preconfer receives valid preconf requests but neglects their duty and fails to provide preconf services.*

As illustrated in Section 5.4 and Section 6.2, there are various factors that can influence a preconfer's decision to provide a preconf, such as the preconf tip and the incompatibility of conflicting transactions. As idleness faults depend on requests being delivered to the preconfer but not being responded to, this introduces a subjective observability requirement that could potentially be handled by a trusted party such as an overseer as presented in Section 5.4.1, in Definition 12.

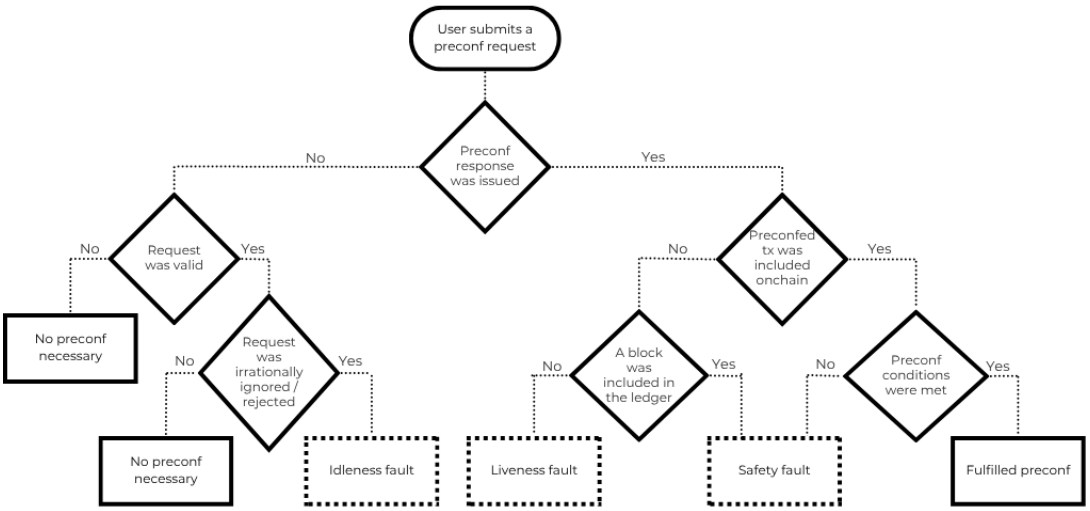

Figure 2: Preconfer Fault Decision Tree

5.6.2 Enforcement Mechanisms and Proofs of Malicious Behaviour

Preconf protocols rely on some form of economic incentive to encourage honest preconfer behaviors. These incentives are provided by mechanisms that either explicitly or implicitly punish malicious preconfer behavior.

Definition 16. *An **enforcement mechanism** triggers punishment of a preconfer when the preconfer satisfies one or more punishing conditions.*

There are many possible enforcement mechanism implementations. Key differences between enforcement mechanisms include whether the overseer responsible for observing punishing conditions is permissionless or permissioned, the type of proof required to trigger preconfer punishment, and the level of punishment [38].

In a permissioned overseer setting, observation and triggering of punishments are driven by the permissioned overseer. Generally, the proofs required to initiate punishment for permissioned overseers are less rigorous. Punishing conditions do not necessarily need to be proven to be satisfied, potentially only requiring an overseer signature. When proofs are not required to trigger punishments, this raises concerns about overseer abuse [38].

In a permissionless setting, anyone can act as an overseer to initiate punishment via an enforcement mechanism. Contrary to the permissioned setting, permissionless overseers must be required to submit conclusive and indisputable evidence to trigger explicit preconfer punishments via the enforcement mechanism. Such evidence may include a signed preconf request, response pair disagreeing with the blockchain ledger. When indisputable evidence is not available, some form of arbitration between preconfer and overseer(s) must take place. As triggers are disputable in this case, this exposes permissionless overseer enforcement mechanisms to incorrect or unsatisfactory triggering and punishment [38, 46, 73].

Another important aspect related to the indisputable evidence required to trigger a preconf punishment is computational complexity. The authors in [34] discuss how this complexity depends on the type of preconf – inclusion versus execution. For execution preconfs, it is sufficient to prove that, at the specific position where the preconfed transaction is expected to appear, a different transaction exists. In contrast, for inclusion preconfs, the overseer must prove that a preconfed transaction is absent from every position in all blocks proposed after when the preconf was given until any imposed deadline for fulfillment. This deadline can be protocol-imposed, or imposed by the request (see Section 5.3).

5.6.3 Punishments

	Direct	**Indirect**
Real-time	Overseer interrupts preconfs or proposals	Downgrade reputation to affect orderflow
Ex-post	Slashing	Downgrade reputation to affect orderflow
	Temporary black-listing	Forfeit revenue while blacklisted
	Temporary stake freezing	

Table 1: Preconfer Punishment Mechanisms

As mentioned in the previous section, preconfer punishments can be divided into **direct** and **indirect** punishments, depending on how punishments are applied to a preconfer's net worth. Direct preconfer punishments involve preconfer collateral slashing, while indirect preconfer punishments target revenue from preconfs [38]. For example, if a preconfer is punished through blacklisting [38] for a specific period, the preconfer forfeits all revenue from preconfs during that time.

Preconfer punishments can also be divided into **real-time** and **ex-post** punishments, depending on when punishments are applied [38]. In real-time punishment, preconfer's actions are monitored by an overseer and any misbehaviour is immediately detected preventing the preconfer from making any profit. For instance, in real-time punishments, the preconf responses may also need to be signed by the overseer. This means that the issuance process can be disrupted not only when the preconfer is idle, but also if the overseer fails to perform their role. This creates a dependency on overseer liveness for the preconf protocol to function.

Ex-post punishments penalize dishonest preconfers retroactively. This removes the preconfer as a liveness dependency for the preconf protocol. However, ex-post punishments typically require preconfers to lock up collateral that can be slashed. If any of the slashing conditions are violated, the preconfer loses a predetermined amount of their collateral. Moreover, a preconfer can be temporarily blacklisted, i.e., be deprived of their right to provide preconfs or have their stake temporarily frozen. In addition to these explicit punishments, enforcement mechanisms can also limit the preconf orderflow of dishonest

preconfers or downgrade their reputation to affect future orderflow [38, 59], implicitly reducing preconf revenue.

6 Preconf Economics

This section focuses on preconf economics, with particular focus on how the revenue of proposers is expected to change given preconfs, how preconfs are priced, and the general economic viability of preconfs. Although the enforcement mechanisms described in Section 5.6.3 fall under the heading of preconf economics, we omit any additional discussion on punishments in this section.

By construction, preconfs require proposers to commit to block-space earlier than block proposal time. However, without enforcement of preconfs or explicit preconf rewards, preconfers have an incentive to withhold preconf responses as long as possible. All else equal, delaying the issuance of preconfs allows proposers to build blocks using up-to-date information, with the number of transactions in the mempool and number of possible transaction orderings increasing with time. These are all factors that contribute to a proposer's ability to extract MEV.

In contrast, user demand for preconfs translates to users gaining utility by receiving a confirmation of transaction execution or inclusion before a proposer proposes a block, as already mentioned in Section 5.4.1.

To address the tension of users wanting preconfs, and proposers wanting to wait as long as possible before a response is provided, proposer incentives to provide preconfs are needed. We have already seen in Section 5.6 that the preconfer role can be subjected to punishment conditions to incentivize certain behaviours, including timely responses to requests. Just as punishments can be applied for slow responses, incentives in the form of preconf tips can be provided for timely responses.

The remainder of this section considers how preconfs affect proposer revenues, and how preconf tips play an important role in the adoption of preconfs. We conclude the section by describing several preconf tip pricing models that have been discussed in the preconf literature. Equipped with well-founded preconf tip pricing models, users, proposers, and preconfers alike can use or adapt these models as required, and make informed decisions about preconf tip pricing.

6.1 Impact of Preconfs on Proposer Revenue

Several works have attempted to model the effect of preconfs on expected proposer revenue [32,33,74–76]. Generally, preconfs increase demand for block-space by offering users an additional mechanism to express that demand. Unfortunately for proposers, block-space demand does not directly translate to increased proposer rewards. This is because many blockchains, including Ethereum and its L2s, deterministically increase transaction base fees (fees that go to the blockchain protocol and not to the proposer) [77] when demand is high. Therefore, sustained demand resulting in increased base fees reduces the relative priority fee/tip that a proposer can capture. Demand aside, there are two key takeaways from existing literature on proposer revenue from preconfs:

1. Inclusion preconfs are a relatively straightforward way for proposers to increase their revenue from proposing without disrupting their expected revenue from existing block-building strategies, such as MEV-Boost, commit boost, or otherwise [32, 78].

2. Execution preconfs require preconfer sophistication and preconf-specific tip revenue to make execution preconfs economically viable for proposers with the option not to preconf [33, 74, 75].

In the following subsections, we provide a detailed breakdown of how these takeaways are established, first for inclusion preconfs and then for execution preconfs.

6.1.1 Inclusion Preconfs & Proposer Revenue

Inclusion preconfs provide proposers with a potential source of revenue through preconf tips without a clear need for increased proposer sophistication [32, 76, 78]. The study provided in [76], although based on limited mainnet data from different preconf protocols, supports the thesis that preconfs add value to typical Ethereum blocks.

In [32, 78], it is identified that proposers can use even basic preconf pricing models to offer preconfs and expect to increase their overall revenue. The key factors at play here are:

1. **Flexibility:** Inclusion preconfs give proposers the ability to choose where preconfed transactions are included in a block. Specifically, a transaction receiving a preconf can be placed anywhere in a block by a proposer.

2. **Ease of Pricing:** The further down in the transaction list of a block a transaction

   is positioned, the less value and volatility its expected fee (or preconf tip) has in dollar terms. These low price, low price-volatility parts of the blocks are where inclusion preconfs are expected to compete for inclusion in the blockchain ledger, given the flexibility that inclusion preconfs give to propsoer with respect to transaction positioning in a block.

   This makes the pricing of inclusion preconfs straightforward for a proposer, as the expected error of pricing models is low. For example, a preconfer who knows that 50% of transactions pay less than 1$ per 100,000 gas used can happily sell 100,000 gas inclusion preconfs for 1$ or more until the total gas consumed by inclusion preconfs is more than 50% of the available block-space (gas represents a unit of computation, and each Ethereum block has a limit on the total amount of gas that can be consumed by its transactions).

3. **Compatibility with Block-Building Auctions:** Thanks to designs like Commit-Boost [79] and the Constraints API [68], any preconfs committed to by a preconfer can be communicated to and enforced upon builders. These implementations are practical implementations of earlier theoretical designs [80]. Such designs allow proposers to benefit from MEV-Boost style competition (see PBS in Section 3.2) among builders on the remaining parts of the block, enhancing proposers' revenue.

Inclusion preconfs therefore stand as an easy-to-price revenue add-on for proposers. Of course, pricing sophistication is always possible. Proposers can likely increase expected revenue by using more sophisticated pricing models compared to the conservative inclusion preconf pricing models presented here. However, expected or required sophistication is not without risks (see Section 7).

A caveat to using models that estimate preconf revenue by extrapolating regular transaction revenues, as is done in [32] and [78], is provided in [76]. The authors point out that one must be careful when looking at simple revenue averages since they can be heavily skewed by large outlier blocks with very high rewards. Although outliers are explicitly removed in both [32] and [78], this warning remains relevant to anyone looking to use such pricing models.

6.1.2 Execution Preconfs & Proposer Revenue

Unlike inclusion preconfs, execution preconfs must be executed in a specific way, typically at a specific position in a specific sequence of transactions. This restriction has implications on a proposer's expected revenue when offering execution preconfs. Although the proposer is acting as a preconfer, we use the term proposer in-keeping with the purpose of this section – analysing the effect of preconfs on the proposer's revenue. The models presented in [75], [33], and [74] represent the main contributions in the space at time of writing.

The models in [75] and [33] are presented by the same set of authors. In these works, the authors claim that execution preconf protocols can be classified under two general classes of preconf protocols. These classes are independent-sub-slot auctions (ISSAs) [75] and dependent-sub-slot auctions (DSSAs) [33]. In-keeping with the language of these works, we will refer to At a high-level, these classes are described as follows:

- **Independent sub-slot auctions (ISSAs):** At its core, ISSAs propose proposers splitting slots into sub-slots and running independent MEV-Boost-style auctions to build sub-blocks at each sub-slot. Each of the winning sub-blocks is signed by the proposer, and stand as the preconfs for that sub-slot. Given these preconfed sub-blocks, the final proposed block is the concatenation of each sub-block's transaction lists. The crucial property of ISSAs in contrast to DSSAs is the independence of the sub-slot auctions, where at each sub-slot, the proposer is trying to maximize the revenue from the current sub-slot auction. The ISSA framing is aligned with multi-round MEV-Boost [61], where it is also suggested to run independent MEV-Boost-style auctions to build sub-blocks at each sub-slot.

- **Dependent sub-slot auctions (DSSAs):** As in ISSAs, DSSAs split a proposer's slot into sub-slots of arbitrary length, where sub-blocks are built at each sub-slot with the final proposed block being the concatenation of all sub-blocks. The distinction between DSSAs and ISSAs is that, at each sub-slot in DSSAs, the proposer chooses the sub-block which maximizes the expected revenue

of the entire slot. This expected revenue includes the transaction fees from the block, preconf tips, and any MEV that can be captured, as well as the expected revenue of future sub-blocks.

Choosing sub-blocks that maximize a slot's expected revenue is an NP-Hard packing problem (think Knapsack Problem where the value of the contents and the size of the Knapsack are continuously changing) for proposers. As such, there is a high degree of proposer sophistication required to run a DSSA.

Under this classification, an execution preconf protocol being an ISSA or DSSA has significant impact on expected proposer revenues. The main results on ISSAs and DSSAs are summarized as follows:

- **ISSAs:** It is estimated in [75] that, *ignoring any expected preconf tips*, preconfing using ISSAs with eight equal length sub-slots reduces a proposer's expected rewards for a slot by 50% compared to a single end-of-slot MEV-Boost auction. At the limit, it is estimated that ISSAs with infinitesimally short sub-slots reduce a proposer's expected revenue by 74% compared to a single end-of-slot MEV-Boost auction, *ignoring any expected preconf tips*.
- **DSSAs:** In [33], it is proven that, *ignoring any expected preconf tips*, a proposer's expected revenue through DSSAs is greater than or equal to the expected revenue from a single end-of-slot MEV-Boost auction. This result is derived by observing that a proposer's expected revenue from preconfing nothing at each sub-slot must be equal to that of running end-of-slot MEV-Boost. Therefore, any alternative maximization strategy chosen by a proposer must have increased the expected proposer revenue.

It is important to emphasize that these initial results exclude expected preconf tips from their analyses. Although the expected revenues from ISSAs appear pessimistic in comparison to DSSAs, it can be argued that the preconf tips in ISSAs are expected to be higher than the tips in DSSA [33]. The reasoning here is that ISSAs enable and incentivize users to pay to update the entire blockchain state, including contentious state (see Definition 5) at each sub-slot. This is in comparison to DSSAs, where the proposer/preconfer on behalf of the proposer is expected to strategically delay updates to contentious state in order to maximize the full slot's revenue. The effect of proposers preferring to delay contentious state updates to capture more MEV has been formally proven to exist [81].

In [81], it is proven that the value lost by liquidity providers in decentralized exchanges is strictly increasing in the time between state updates. As mentioned previously, ISSAs decrease the time between state updates compared to the normal block-building pipeline or DSSAs. In turn, liquidity providers under ISSAs would lose less money, meaning lower fee requirements and more liquidity for users. This creates a fly-wheel where users would be incentivized to trade more with decentralized exchanges, in turn paying more transaction fees and preconf tips.

The simulation-based conclusions from [74] are in line with the more formal results of [33]. Both articles identify that proposers who consider the problem of building sub-blocks to maximize the expected revenue of the entire block stand to profit from preconfs. In [74], the authors argue that proposers are likely to ignore preconfs on contentious state unless there are clear overpayments to act on this contentious state. This is because the sophisticated entities that normally pay most to act on contentious state pay more as slots progress, in line with the results from [81]. It is again noted that despite not preconfing on contentious state, proposers can increase their expected revenue by preconfing on uncontentious state. The authors conclude by acknowledging the significant increase in sophistication required to preconf, whether through differentiating between contentious and uncontentious state, or determining over-payments on a per-transaction basis.

6.2 Preconf Tip Pricing

Preconf tips are a critical component in any preconf protocol, in the same way that transaction fees are critical for blockchains. Preconf tips provide an incentive for preconfers to preconf a request, allowing users to express their preference for how quickly a request should be responded to (see Section 5.3). Therefore, the pricing of preconf tips to strike the balance between user and preconfer preferences to minimize costs yet maximize utility stands as an important area of research and development. Preconf pricing models vary in importance depending on whether or not the preconfer is trusted.

Preconfer trust can be as a result of an overseer enforcing preconfs (see Section 5.6.2) and punishing preconfers (see Section 5.6.3) for not preconfing fee-paying requests, or where the preconfers themselves are trusted, as in non-based single-proposer L2s [6–8]. In such trusted preconfer protocols, preconfers

are expected to preconf any request paying a tip above some protocol-specified threshold. This threshold is sometimes referred to as the L2 base fee. This L2 base fee is intended to cover an L2 transaction's costs, such as the costs of posting the transaction to the respective data-availability layer, or proving the transaction as part of a batch, where applicable. In systems where preconfs are expected to be given regardless of the preconf tip, preconf tip pricing is not meaningful.

Preconf tip pricing becomes relevant for untrusted preconfer protocols, where the preconfer has freedom to respond positively or negatively to requests. In such settings, one must instead consider preconfers as trying to maximize the revenue of their entire preconf slot e.g., through MEV extraction. Maximizing revenue requires a preconfer to understand the value being forfeited when providing a preconf. Understanding this forfeited value, in theory, allows a preconfer to accept any preconf tip paying more than this amount. Although there is no established pricing method for preconfs, there are several resources that provide information on how preconfs can be priced [32, 78]. Pricing methodologies can be separated into those for inclusion preconfs [32, 78], and those for execution preconfs, where no formal model for pricing has been established to the best our knowledge.

6.2.1 Inclusion Preconf Pricing

For inclusion preconf pricing, several models have been put forward [32, 78]. In [32], it is identified that block-space demand, measured in ETH per block-gas used, is highest for the first transactions in the transaction list of a block, and least valuable at the end of a block. Specifically, cumulative transaction fees in block-space follow a lognormal distribution. This aligns with the intuition that first access to contentious state has value for MEV extractors (see Definition 5).

As already mentioned in Section 6.1.1, given that inclusion preconfs need only compete for the least valuable block-space, inclusion preconfs can be priced in an almost identical way to this least valuable block-space. Given the logarithmic nature of cumulative transaction fees versus block-space, [32] demonstrates that basic price curves can be utilized by proposers to price inclusion preconfs for any amount of block-space. As an extension of this pricing curve, [32] identifies a pricing surface to incorporate both the amount of gas to be used by the preconf, and the amount of gas already consumed in the block.

The approach of [78] follows a similar methodology to [32], suggesting the use of historical transaction fees to price inclusion preconf tips. Rather than pricing inclusion preconfs using historical data for an entire block, [78] models inclusion preconf tips using transactions fees from the middle of blocks' transaction lists. This subset of transactions represents transactions paying some premium without competing for contentious state, which, as the authors argue, represents a similar class of transaction orderflow as inclusion preconfs. Both articles identify a need to increase preconf tips as more preconfs are provided and remaining block-space declines.

The models created in both [32] and [78] leave room for refinement and improvement, albeit with the trade-off of increased preconfer sophistication requirements. Possible model adaptions can incorporate up-to-date estimates of block-building revenue or more granular per-unit block-space pricing models.

6.2.2 Execution Preconf Pricing

To the best of our knowledge, there have been no meaningful pricing models proposed for execution preconfs. For a preconfer, pricing execution preconfs translates to continuously observing and analyzing incoming preconf requests throughout a slot. This is more computationally complex than building a single whole block at the end of a slot. Given the oligopolistic nature of MEV-Boost auctions [82] to build these whole blocks, the more complex task of execution preconfing and execution preconf pricing stands as a hard open problem for preconfs.

An indicator of this additional complexity exists on Arbitrum [7], where block proposals are controlled by a single proposer. Arbitrum's Timeboost [83] allows this Arbitrum proposer to outsource block-building through an auction. The winners of this auction are announced ahead-of-time, with winners given an exclusive fixed-time advantage to propose partial blocks before the whole blocks are completed (transactions can be appended to the partial blocks) and proposed by the proposer. Auction winners can use this time advantage to preconf transactions, for themselves or for others at some cost. As such, auction bids reflect the total preconf revenue that can be captured by leveraging this exclusive time advantage. Unfortunately, the algorithms being used to participate meaningfully in Timeboost and its auctions remain closed-source.

According to [84], more than 90% of Timeboost auctions are being won by two entities – an early sign that preconf pricing will follow similar, if not more extreme, centralization trends as those observed in

the MEV-Boost markets. For comparison and as previously mentioned, at any given time in the last two years, two to three builders have been responsible for 90% of blocks proposed through MEV-Boost [82].

6.3 Summary of Preconf Economics

Preliminary research suggests that while preconfs can create new revenue streams for block proposers, especially via inclusion preconfs, there are also scenarios where preconfs complicate or even reduce revenue (particularly for execution preconfs unless managed optimally). Pricing mechanisms for these services are still an open research problem, and practical implementations like Arbitrum's Timeboost are providing initial data points. The next section discusses various risks that these protocols entail, complementing these economic analyses with broader considerations.

7 Preconf Risks

This section focuses on the risks, summarized in Table 2, that accompany the addition of preconf protocols to a blockchain's block-building pipeline. The added sophistication and infrastructure that preconfs require compared to existing block-building pipelines bring risks and considerations for all entities. In this section, we detail some of the key risks and considerations that preconfs bring. The headline risks are summarized in Table 2.

Although intended to be comprehensive, more risks may surface as more preconf protocols emerge and underlying blockchains continue to evolve (e.g., through the introduction of new EIPs for Ethereum [65]).

	Preconfer	Proposer	User	Blockchain
Implementation	✓	✓	✓	✓
Slashing	✓	✓		✓
Reputation	✓	✓		✓
Liveness	✓	✓	✓	✓
Legal	✓	✓		
Centralization	✓	✓	✓	✓
Congestion	✓	✓	✓	✓

Table 2: Preconf Risks

7.1 Implementation Risk

Preconf protocols are additional technological components in an ever-growing blockchain tech stack. Like all blockchain components, preconf protocols must be thoroughly tested and audited to ensure protocols act as intended, are free of bugs, and are resilient to failures of other components in a blockchain's tech stack. Implementation risk increases the likelihood and threat of all other risks, even in scenarios where stakeholders are following the rules of the preconf protocol. The main defense against this is robust protocol design and thorough auditing of preconf protocols.

7.2 Slashing

As described in Section 5.1 and Section 5.6.3, *slashing* is an ex-post, direct penalty applied to slashable collateral locked by preconfers and imposed via enforcement mechanisms (see Section 5.6.2). While slashing is meant to disincentivize undesired preconfer behavior in a preconf protocol, there is the potential for unfair or accidental slashing. The risk of being slashed is relevant for several entities in a preconf protocol: Preconfers, who post slashable collateral; proposers, who inherit penalties from preconfers; and the underlying blockchain, whose economic security may be affected by mass slashing.

7.2.1 Slashing: Preconfer

As described in Section 5.6.1, preconfers risk losing some or all of their slashable collateral for committing faults. Safety and liveness faults are provable as they involve a direct contradiction between a signed preconf response and what is observable in the blockchain ledger (assuming a common view of the entities

on the blockchain ledger). While proof by contradiction is sufficient to slash a preconfer committing a safety fault, or proposer committing a liveness fault (assuming the preconf registry can access relevant blockchain ledger data), a trusted overseer is required to attribute idleness faults (see Section 5.6). The risk of being slashed depends on the type of fault the preconfer (perhaps unintentionally) committed:

- **Safety fault:** Safety faults should never be committed by an honest preconfer who is also a proposer (see [72] and Section 5.6.1). In such cases, the punishment is expected to be severe. When the preconfer is not a proposer, the preconfer may risk being slashed for a safety fault (e.g., [70]) if they either fail to make the proposer aware of any preconfed transactions in time for proposal, or the proposer omits specific preconfed transactions at proposal time and no fault attribution mechanism exists to exonerate the preconfer. Fault attribution can, for example, be implemented by having a relay honestly report the list of preconfed transactions that was made known to the proposer [69].
- **Idleness fault:** A preconfer can be slashed even without malicious intent, for example, if their server is down or there is a network issue and they fail to publish preconfs on time. In a protocol with a permissioned overseer, there is the additional risk that a malicious overseer may unfairly slash the preconfer even when no fault is actually committed.
- **Liveness fault:** Usually, only if the preconfer is a proposer, the preconfer is punished for a liveness fault (for more details, see Section 5.6). When the preconfer is not a proposer, liveness faults usually result in penalties for the proposer (e.g., [70]).

7.2.2 Slashing: Proposer

The risk of a proposer being slashed for their own or the gateway's misbehavior depends on whether a safety or liveness fault is committed:

- **Safety fault:** As previously mentioned, if the proposer has delegated preconfing to a gateway and no fault attribution mechanism is in place [69], safety faults typically result in the preconfer – rather than the proposer – being penalized (e.g., [70]).
- **Liveness fault:** Liveness faults are missed slots and the fault of the proposer. The proposer is usually penalized even if they have delegated preconfing to a gateway, except in cases where a fault attribution mechanism is in place to detect malicious behavior by the preconfer, such as delays in sending preconfs.

  The proposer risks being slashed even for missing slots accidentally (see Section 5.6.1). To protect themselves against this risk, proposers could request that preconfers chain their preconfs [71], allowing preconfs to be fulfilled by subsequent proposers in the lookahead in return for a proportion of the tip revenue. Preconf protocols can amplify the economic impact of missing slots in L1, where proposers already incur the opportunity cost of missing out on proposer rewards and revenue.

7.2.3 Slashing: Blockchain

As discussed in Section 5.1, prospective preconfers are required to post slashable collateral. By imposing barriers to entry, preconf protocols:

- Require careful cryptoeconomic protocol design to ensure that incentives are aligned in a way that incentivizes entities to participate.
- Risk centralizing the service's provision to a few, well-capitalized entities.

The latter has risks for the underlying blockchain especially when slashable collateral is sourced via the capital-efficient means of restaking. In this case, a single malicious or accidental act by a preconfer could trigger cascading slashings across any protocols also secured by the restaked collateral. If a widespread slashing event were to occur, the security of both the underlying blockchain and the specific preconf protocol would be affected (e.g., see [85] and [45]). Therefore, dedicated collateral used by preconfers to register is less risky for the preconf protocol than restaked collateral.

7.3 Reputation

Unlike the direct financial penalties of slashing, *reputation loss* is an indirect and ex-post punishment (see Section 5.6.3) meant to reduce the future revenue of the affected preconfer. As an attribute, an entity's reputation can affect their standing both inside the preconf protocol, which may lead to revenue loss, and outside of the context of a preconf protocol. The risk of diminished reputation concerns multiple entities in a preconf protocol: Preconfers, who face direct reputational damage and indirect revenue loss from their own misbehavior; proposers, who may be held accountable for the behavior of preconfers; and the blockchain protocol, which may suffer reputational damage if preconf protocols are perceived by users to be unfair or extractive (e.g., proposers allocate most of the block-space to preconfed transactions, leaving little room for users who do not want to preconf their transaction).

7.3.1 Reputation: Preconfer

The reputation of a preconfer can be damaged by any of the faults they commit. The risk of diminished reputation extends to the case where a preconfer is perceived to be responsible for non-fulfillment, although the proposer had committed a (potentially accidental) liveness fault (see Section 7.2.2). Recall that in such cases, it is typically the proposer – not the preconfer – that is penalized by the preconf protocol. However, users may not take the time to determine which entity actually committed the fault, especially in the absence of a fault attribution mechanism. Diminished reputation could make users, or their wallets acting as permissionless overseers, less likely to send future preconf requests to the misbehaving preconfer. This is known as reduced orderflow and is described in detail in Section 5.6.3. By reducing orderflow to preconfers with diminished reputation, affected preconfers are unable to capture the preconf tip revenue they otherwise would have received.

The risk of reputational damage for a preconfer can extend beyond any foregone revenue within the preconf protocol itself to other protocols or business activities in which the preconfer is engaged. Some preconfers may choose to operate their preconf services using separate entities, thereby mitigating the risk that diminished preconfer reputation spreads to the preconfer's identity elsewhere.

7.3.2 Reputation: Proposer

Proposers risk being held accountable for the actions of the preconfers they delegate to, as users are unlikely to distinguish between the two entities. The proposer's reputation may be harmed if the preconfer commits a safety fault – even if the proposer is not at fault and is not penalized by the preconf protocol. Diminished reputation of proposers by association with malicious or faulty preconfers may be of particular concern for entities operating many validators as part of a larger set of business offerings. This is because the previously described risk of reputation spread from misbehavior within a preconf protocol to other, external activities in which the preconfer is engaged, applies.

7.3.3 Reputation: Blockchain

The behavior of entities in and specific implementation of preconf protocols could affect the underlying blockchain's reputation. If preconfs are perceived to be highly extractive, for example through DSSAs (see Section 6.1.2), it could negatively affect the user experience and consequently the reputation of the blockchain. In the long run, the loss in the blockchain's reputation could result in (i) an exodus of users and applications in search of a blockchain with better user experience and (ii) a drop in the native currency's exchange rate.

7.4 Liveness

Liveness risks affect multiple entities in a preconf protocol: Preconfers, who may be unable to fulfill their preconfs due to proposer downtime; proposers, who risk being slashed and losing rewards for missed slots; and users, whose preconf requests may go unfulfilled.

7.4.1 Liveness: Preconfer

A preconfer's ability to fulfill their preconfs depends on the liveness of both the proposer and the underlying blockchain. As mentioned in Section 7.2.1, preconfers can be punished for idleness faults, regardless of whether the preconfer was at fault or not.

7.4.2 Liveness: Proposer

Proposers are responsible for proposing valid blocks. In the context of preconf protocols, a missed slot is punished via slashing and foregone revenue for the proposer (see Section 7.2.2). Missed slots may happen for reasons both internal and external to the proposer:

- **Internal:** As discussed in Section 5.6.1, the proposer may go offline for any number of (potentially accidental) reasons – such as power outages or internet downtime – making them unable to propose. Solutions to allow proposers to insure themselves against accidental liveness faults through chaining have been discussed in the same section.
- **External:** If the proposer, or their delegated gateway, outsources final block construction (see Section 5.5.3), the proposer risks the full block containing preconfed transactions not being propagated in time.

From the perspective of the preconf protocol, these cases are indistinguishable. As with the fair attribution of safety faults described in Sections 7.2.1 and 7.2.2, in the case of liveness faults caused by external factors, exonerating the proposer will likely require the involvement of a trusted third party.

7.4.3 Liveness: User

Users specify a preconf tip to the preconfer in expectation of preconf fulfillment (see Section 5.3). Preconf fulfillment mainly depends on the preconfer and proposer being online and non-malicious:

- **Preconfer:** If the preconfer is found by an overseer to be committing an idleness fault (see Section 5.6.1), users' preconf requests will remain unfulfilled, and, if the preconfer does not come back online, may eventually expire.
- **Proposer:** As described in Section 7.4.2, the proposer may commit a (potentially accidental) liveness fault for both internal and external reasons. If the users' preconfs were chained, they will be fulfilled, only in a later slot. If not chained, users' preconfs will remain unfulfilled and will eventually expire.

The cryptoeconomic security of the preconf protocol has a direct impact on the risks faced by users. For instance, users are better protected when the financial penalties imposed on the preconfer and proposer for failing to fulfill the preconf exceed any potential rewards from misbehavior. Additionally, the risk that users face is closely related to the costs incurred by a malicious preconfer or proposer whose sole intent is to disrupt the preconf protocol. When penalties are inadequate, the consequences can be significant [86,87]. Finally, the risk for users can be mitigated if users are compensated for non-fulfillment, e.g., in [30].

7.4.4 Liveness: Blockchain

In some cases, liveness faults in preconf protocols may disrupt the liveness of the underlying blockchain, for example:

- When a proposer delegates preconfing to a gateway, and the gateway delays – or fails entirely – to deliver the preconfs, the proposer may be delayed in proposing their block, potentially resulting in the block being excluded from the blockchain.
- When the preconfer is also the proposer, then:
  - Their downtime can affect both the liveness of the preconf protocol and the liveness of the underlying blockchain.
  - If the cryptoeconomic design of the preconf protocol creates incentives for a preconfer – who also acts as a proposer – to intentionally miss a slot (e.g., when they are scheduled to propose two consecutive blocks and can extract more MEV by skipping the first), it can negatively impact the liveness of the underlying blockchain.

7.5 Legal Risk

Preconfs are contracts between the user and the preconfer. As such, contract law [88] may apply when arbitrating failed fulfillment of a preconf. This presents a potential legal risk for proposers and preconfers with respect to preconfing. The exact legal implications of preconfs will emerge as preconf protocols mature.

7.5.1 Legal: Preconfer

A failure to fulfill preconfs could find users seeking recompense for damages caused by the preconfer. The legal entities behind preconfers may be held liable through legal channels if a failure to fulfill a preconf results in financial harm to a user. Recall that a preconf is a commitment that the blockchain ledger at a future time satisfies some predicate. Users, as a result of receiving a preconf, may decide to initiate other actions. For example, a preconf to buy ETH on a decentralized exchange may cause a user to sell ETH on a centralized exchange in an attempt to profit from arbitrage. If the preconf does not get fulfilled, users may be unable to retroactively cancel their sell order, potentially resulting in financial loss.

To mitigate the risk of preconfers being targeted for recompense through legal systems, preconfers may choose to include explicit disclaimers about lack of liability for any losses incurred by users as a direct or indirect result of interacting with the preconf protocol.

7.5.2 Legal: Proposer

As described in Sections 7.2.2 and 7.3.2, proposers that delegate preconf duties inherit some of the risks that preconfers face. If a delegated preconfer fails to fulfill its preconfs, the proposer could likewise face legal challenges due to user damages.

7.6 Centralization

Preconf protocols exert centralizing pressures on blockchains. This can be seen through the simultaneous increase in revenue that preconfs offer to preconfers (see Section 6) and the increased sophistication required to preconf vs not preconfing (see Section 6.1. This combination incentivizes blockchain entities to offer preconfs, which either means outsourcing preconfing (and at least partial block-building) to professional entities with economies of scale, or the blockchain entities themselves becoming preconfers, managing all necessary preconf infrastructure, pricing, and risk independently. Regardless of which of these paths emerge, preconfs risk the unexpected and likely undesired consolidation of blockchain roles into a small set of highly-sophisticated entities.

In the following sub-sections we explore in more detail the centralization pressures, effects, and mitigations being considered for each of the blockchain entities we consider.

7.6.1 Centralization: Preconfer

The centralization risk to preconfers from preconfing centers around the expected competitiveness of preconfing. Preconfing's sophistication requirements likely means consolidation of the preconfer set into a small number of dominant players. This means tight profit margins and high barriers to entry for all but the most sophisticated preconfers.

7.6.2 Centralization: Proposer

The potential revenue from preconfs could force proposers to participate in preconf protocols. For low-resource/unsophisticated proposers, this likely means delegation of preconfing to a small group of gateways/builders in the same way that block-building is outsourced today. Given the added complexity of preconfing compared to normal block-building, this small group of delegates may become dominated by one or two oligopolies, in-line with the centralization seen in MEV-Boost [82]. Oligopolistic delegates can elect to extract rents from proposers, especially if preconfing becomes ingrained in a blockchain's block-building pipeline. These oligopolies also become points of failure, which could prevent a proposer from proposing if delegate failure were to occur.

One proposed direction towards mitigating the risks of oligopolies forming among preconfer delegates is the preconfirmation sauna [89]. In this proposal, the authors advocate for the development of generalized preconfirmation infrastructure and tooling to enable preconfers and proposers to reduce the friction of interactions and on-boarding of entities to preconf protocols. The prevents vendor lock-in, which is a key factor in enabling oligopoly abuses.

7.6.3 Centralization: User

As the blockchain "consumer", users face a degraded user experience if centralization through preconfs materialize. Users risk degradation across each of censorship resistance, fee increases, and fault tolerance. These effects are all synonymous with service provider monopolization.

Apart from centralization of the block-building pipeline being a risk for users, there are also centralization pressures on users themselves. Users must calculate appropriate preconf tips, customs penalties (see [59]), create valid preconf requests, and identify the correct preconfers or endpoints to send their requests to.

In the case of preconf tips, Section 6 described how tips are expected to be more complex and volatile than regular EIP-1559 transactions. This makes the setting of preconf tips potentially dangerous, which in turn means likely outsourcing and additional fees for these services.

7.6.4 Centralization: Blockchain

If the underlying blockchain does not expect a blockchain entity to engage in preconfs, there is likely some centralization being introduced that the blockchain is not designed for.

While centralization may be an acceptable design trade-off for scalability in many L2s (see Section 3.3), blockchains that are designed to have decentralized and permissionless roles may see centralization as unacceptable. This is particularly true for Ethereum. Without proposer decentralization, Ethereum's censorship resistance and fault tolerance guarantees fail [90]. Unfortunately, these guarantees are already failing, as even now, proposers are outsourcing block-production en-masse to the same two or three entities through MEV-Boost [82] (see Section 3.2). Preconfs stand to exacerbate this risk for decentralized blockchains, given the additional complexity that preconfs entail (see Section 6.2).

The protection closest to being deployed against centralization effects in Ethereum is FOCIL (fork-choice enforced inclusion lists), with a full specification available under the heading of EIP-7805 [91], and incentive analysis of candidate transaction fee mechanisms in [92]. FOCIL allows non-proposing validators to enforce transaction lists upon block proposals in Ethereum, assuming that non-proposing validators will remain decentralized and majority honest. This is despite the evidenced rationality of validators when elected as proposers, through their active use of MEV-Boost. As such, FOCIL alone stands as a temporary fix for an ever-centralizing Etheruem validator set.

To tackle the misalignment of proposer incentives with other validator incentives, several protections have been discussed which aim to separate the role of the proposer from other validator roles, most notably attesting, but potentially also participation in FOCIL-like protocols for censorship resistance [93–95]. Worryingly, interest in these solutions has been small, with development pending, to the best of our knowledge. Given the threat of centralization to decentralized systems is large, this demands increased awareness, investigation, analysis and development of centralization protections.

7.7 Congestion

Preconf protocols put the underlying blockchain and blockchain infrastructure at risk of becoming congested. This is for two interrelated reasons:

- Entities are incentivized to vertically integrate with preconfers to maximize their ability to act on contentious state [96].
- Users may spam preconfers with requests to address the randomness of delivery times and value of interacting with contentious state first [61, 96].

This heavy traffic strains infrastructure, increases costs, and in the case of spam, results in wasted blockchain resources. The existing block-building pipeline isolates and contains congestion risk on the builder and relay side, with the proposer only required to select block-headers, and the resulting blocks containing little-to-no failed transactions in order to maximize a builder's revenue (builders do not receive fees from failed transactions).

Congestion impacts all preconf protocol entities: Proposers and preconfers, who are required to understand how congestion impacts revenue and how to handle worst-case loads; users, whose costs and inclusion/execution time may become more volatile; and the underlying blockchain and its applications, which may not be equipped to handle large amounts of traffic. In addition to creating inequalities among users, latency races can incentivize geographical centralisation [96]. A potential mitigation to congestion risk is an auction-based system that batches preconfs in sub-slots instead of streaming continuously [33, 61]. Through batching, proposers and preconfers can outsource congestion management to builders, a technique already employed in Ethereum's block-building pipeline (see Section 3.2).

8 Existing Protocol Case Studies

At this point of the SoK, we are equipped with sufficient generalized preconf knowledge to examine several in-production preconf protocols in detail. For ease of reading and consistency, we map each of the examined protocols to the six-step preconf pipeline framework from Section 5.

8.1 Optimism

This subsection examines the preconfs inherent in a non-based L2 with a centralised proposer (see Section 3.3), using Optimism [6, 97] as a case study. Although not explicitly labeled a "preconf protocol", the confirmations provided by the proposer serve the same practical function for users, offering guarantees of the blockchain ledger before it is posted to L1.

1. **Registration:** The registration process is permissioned and centralized. There is a single entity named Optimism's sequencer which is operated by the Optimism Foundation. There is no explicit preconfer registry or collateral requirement.

2. **Election:** The election is trivial and predetermined. As a network with a single, designated block producer, the sequencer is implicitly the elected preconfer for every L2 block.

3. **Request:** Users who wish to receive a sequencer confirmation submit a standard signed L2 transaction directly to the sequencer's private mempool.

4. **Response:** Upon receiving a transaction, the sequencer determines the transaction's execution order, computes its resulting state change, and includes it in an "unsafe" L2 block. The response to the user is a high-fidelity execution preconf. It is not merely a promise of future inclusion but a firm commitment to a specific ordering and outcome, complete with a post-transaction state root (a commitment to the L2 blockchain ledger as it will be formed after the inclusion of these transactions in this specific order). Nevertheless, unsafe transactions may not be finalized if the sequencer fails to publish the block to Ethereum within the sequencing window or if it re-organises the unsafe blocks.

5. **Fulfillment:** Fulfillment occurs in two stages. First, a separate batcher process compresses transaction data from unsafe blocks and submits it to the L1, at which point the L2 block status is upgraded from "unsafe" to "safe". Second, once the L1 block containing this data is finalized by Ethereum's consensus (meaning that it is included in the Ethereum blockchain ledger), the L2 transaction is considered "finalized", completing the sequencer's promise.

6. **Punishment:** The punishment mechanism is primarily indirect and ex-post (see Section 5.6.3). There is no automated on-chain process to slash the centralized sequencer for misbehavior. Instead, sequencer punishment relies on reputational damage and the threat of users migrating away from the ecosystem. This model is a stepping stone, as the long-term roadmap for the OP Stack ecosystem includes plans for a decentralized set of sequencers.

Fair-exchange (see Section 5.4.1) of preconfs is trusted to take place.

8.2 Taiko's "Permissionless Preconfs"

This subsection introduces the design of *permissionless preconfs* for Taiko, a based rollup, as a concrete case study of how based rollups can implement based preconf mechanisms[4].

1. **Registration:** Happens through the Universal Registry Contract (URC) (see Section 5.1.1).

2. **Election:** Uses an *optimistic lookahead scheme*: the first preconfer of each epoch posts the lookahead for the next epoch, consisting of those in the L1 proposer lookahead who have opted in to Taiko preconfs, which anyone can challenge with a fraud proof if incorrect. EIP-7917 [21] is used during these fraud proofs to verify that the submitted lookahead matches the canonical proposer schedule (see Section 3.2). If no proposer is registered, a fallback preconfer is used.

3. **Request:** Standard L2 transactions submitted through the public L2 mempool.

[4]Public documentation, to be posted here: `https://github.com/taikoxyz`, missing at time of writing. This will be updated as soon as the documentation is made publicly available.

4. **Response:** The elected preconfer collects transactions from the public L2 mempool, builds an L2 block, and publishes a signed commitment to the hash of the transaction list together with the transaction list itself every two seconds. This interval is a protocol parameter and may be adjusted over time. Because this commitment specifies the full set of transactions in their canonical execution order, L2 client software can locally execute them to derive the latest L2 state – making this effectively an execution preconf (see Definition 6 and Section 4.1). To explicitly end their preconf duties for the slot, the preconfer issues an *end-of-preconf* message, which commits to stop preconfing and hands over preconf duty to the next preconfer.

5. **Fulfillment:** The preconfer must include the preconfed transaction list in their transaction batch posted to L1 before their elected window closes, either in their own L1 block or via the L1 public mempool.

6. **Punishment:** Triggered when there is a mismatch between what was sent in the response and what was eventually fulfilled. Any entity can slash a preconfer for mismatched transaction lists, missed submissions, or equivocating on end-of-preconf message (e.g., publishing an additional L2 block after issuing an end-of-preconf message).

To address the **fair-exchange problem**, Taiko employs an overseer, governed by the Taiko DAO, that monitors preconf timing and can blacklist operators who delay publication, incentivizing timely preconfs.

8.3 Primev's mev-commit Protocol

In this section, we describe how the mev-commit protocol [30,98] issues preconfirmations, or *commitments* in the preconf protocol's terminology. While we follow the general structure introduced in Section 5, it is important to note that sometimes mev-commit slightly departs from that framework.

The central idea of mev-commit is to enable L1 block builders to provide users with enforceable, yet conditional, assurances about transaction inclusion or execution before the corresponding L1 block is built. To achieve this, the protocol maintains a dedicated mev-commit blockchain with fast block times. When a user requests a preconf by submitting a bid, a block builder may issue a conditional preconf where the condition is that the builder is selected to build the specified L1 block.

To issue such a preconf, the builder records a cryptographic commitment on the mev-commit chain. These commitments hide all details of the bid but can later be opened to allow public verification and settlement. The commitments are supposed to be opened by the block builder after building the corresponding L1 block. If the builder fails to open, e.g., because the promised conditions were not met, the user can instead open the commitment to prove non-fulfillment. In this way, mev-commit enforces accountability for preconfs without sacrificing confidentiality at the bidding stage.

1. **Registration:** Block builders enroll in the protocol through the *Provider Registry*, a smart contract deployed on the mev-commit chain.

2. **Election:** The protocol does not implement an election mechanism. Instead, providers issue conditional preconfs that are valid only if the block they build is annexed to the L1 chain at the specified height.

3. **Request:** Users may request different types of conditional preconfs, ranging from simple inclusion preconfs to execution preconfs.

4. **Response:** Upon receiving a preconf request, a block builder may ignore it or issue a preconf by publishing a cryptographic commitment to the mev-comit chain. The request specifies all conditions for fulfillment. Importantly, the associated bid (the fee a user offers for a preconf) decays over time from submission, incentivizing builders to respond quickly to maximize profit.

5. **Fulfillment:** When a builder B produces an L1 block at height h, all preconfs issued by B for height h become enforceable. If B fails to honor any such preconfs, penalties apply.

6. **Punishment:** The penalty for failing to fulfill a preconf is specified in the request as a compensation amount the user will receive in such cases. If the preconf is not honored, this compensation is transferred to the user, and the preconf bid is refunded.

To address the **fair-exchange problem**, the protocol utilises a tip decay mechanism, and an oracle that monitors the L1 chain. The actual tip amount is computed using the timestamp of when the commitment was issued as recorded on the mev-commit chain. This incentivizes builders to issue preconfs in a timely manner after receiving the bid. The oracle then looks at opened commitments and checks whether the conditions for fulfillment are met. If so, the builder receives the preconf tip specified in the bid. If not, the builder is penalized by transferring the compensation amount specified in the bid to the user

8.4 ETHGas

ETHGas [99,100] is a preconf protocol which allows upcoming Ethereum proposers to sell commitments to their block-space. ETHGas is integrated with existing PBS infrastructure (see Section 3.2) using Commit-Boost [101], a modified version of the standard MEV-Boost [19] client that enables ETHGas stakeholders to create and manage preconfs.

1. **Registration:** L1 proposers wishing to register with the ETHGas protocol must run Commit-Boost and also post collateral. The collateral can be posted by restaking using EigenLayer (see Section 5.1.1) or deposited directly into the ETHGas collateral smart contract [102].

2. **Election:** ETHGas does not run its own election process but rather provides a platform for these pre-determined proposers to sell commitments for the block-space they are already entitled to produce. Furthermore, a preconfer can choose to sell inclusion preconfs, execution preconfs or whole block commitments [99, 100].

3. **Request:** Users submit requests and purchase preconfs through the API of ETHGas Exchange. Proposers can sell preconfs directly through ETHGas Exchange while less sophisticated proposers may choose to delegate preconfing process to a third party. Inclusion preconfs can be purchased up to 32 slots in advance, while whole block commitments can be secured up to 64 slots in advance [99, 103].

4. **Response:** After a request is received and accepted, the elected L1 proposer for the target slot uses their validator key to sign a cryptographic commitment to the user's request. This signed message, which is delivered to the buyer via the ETHGas Exchange, serves as the preconf [99,104].

5. **Fulfillment:** The ETHGas Exchange streams preconfs to block builders. The builders then construct a valid block that respects these commitments and submit it to a relay. The proposer fetches the header of the most profitable block from the relay, signs it, and submits it to the network. The relay then publishes the full block body. The relay can also deliver a fall-back block in case a builder fails to produce a conforming block in time [105,106].

6. **Punishment:** After a block is finalized, the ETHGas back-end system automatically verifies if the proposer fulfilled their preconfs. It checks for a verifiable on-chain discrepancy between the signed preconfs and the actual content of the block proposed for that slot (or if the slot was left empty). If a violation is detected, the system initiates a process to slash the proposer's collateral, compensating the affected user [102].

9 Conclusion

As blockchain systems evolve, preconfs have emerged as a key mechanism for providing users with guarantees during the waiting period between submitting a transaction to the P2P network and its inclusion in the blockchain ledger. By doing so, preconfs bridge the gap between the scalability of the underlying blockchain protocol and the practical needs of end-users and applications. This SoK outlines the fundamental structure of preconf protocols and presents several preconf protocols currently in production. Furthermore, it reviews existing research and highlights areas for further study in the economics of preconf protocols, such as the revenues earned by preconfers and the pricing mechanisms through which users tip them. Finally, it discusses potential risks faced by all entities involved in preconf protocols, as well as implications for the underlying blockchain protocol.

While no single approach to preconfs has become dominant, preconf protocols in general are likely to play a increasingly important role in shaping the evolution and adoption of Ethereum and other

blockchain ecosystems, particularly in latency-sensitive use-cases. Ultimately, the path forward for preconfs requires further research and development. Preconf adoption introduces new complexities, such as more complex economic incentives and trust assumptions, which must be carefully analyzed and balanced. Collaboration across research, preconf protocols, and underlying blockchain systems will be essential to ensure that preconfs enhance user experience without compromising the security of the blockchain protocols. In this regard, preconfs are not merely an optimization, but a foundational element in the broader effort to make blockchains truly usable at scale.

Acknowledgments

We thank Christian Matt and Bernardo Magri, both Primev, for their contributions to the SoK. This SoK has received funding from Taiko, and the Ethereum Support Program – Grant ID FY25-2142. Special thanks to Taiko whose pioneering mission and ongoing successes in implementing preconfirmations inspired this work.

Legal Disclaimer

This article has been prepared for the general information and understanding of the readers. No representation or warranty, express or implied, is given by Nethermind as to the accuracy or completeness of the information or opinions contained in the above article. No third party should rely on this article in any way, including without limitation as financial, investment, tax, regulatory, legal, or other advice, or interpret this article as any form of recommendation.

References

[1] M. Fitzi, P. Gaži, A. Kiayias, and A. Russell, "Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition," Cryptology ePrint Archive, Paper 2018/1119, 2018. [Online]. Available: https://eprint.iacr.org/2018/1119

[2] C. Che, S. Li, and X. Wang, "Manifoldchain: Maximizing Blockchain Throughput via Bandwidth-Clustered Sharding," in *Proceedings of the 32nd Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, CA: Internet Society, Feb. 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/manifoldchain-maximizing-blockchain-throughput-via-bandwidth-clustered-sharding/

[3] M. Fitzi, P. Gazi, A. Kiayias, and A. Russell, "Proof-of-stake blockchain protocols with near-optimal throughput," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 37, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:210704648

[4] R. Neiheiser and E. Kokoris-Kogias, "Anthemius: Efficient & modular block assembly for concurrent execution," 2025. [Online]. Available: https://arxiv.org/abs/2502.10074

[5] Ethereum, "Scaling." [Online]. Available: https://ethereum.org/developers/docs/scaling/

[6] Optimism, "Optimism," GitHub repository. [Online]. Available: https://github.com/ethereum-optimism/optimism/tree/51eeb76efeb32b3df3e978f311188aa29f5e3e94

[7] Arbitrum, "Arbitrum Docs." [Online]. Available: https://docs.arbitrum.io/welcome/arbitrum-gentle-introduction

[8] ZKsync, "ZKsync Docs." [Online]. Available: https://docs.zksync.io

[9] Luban, "Taiko Preconf Design." [Online]. Available: https://hackmd.io/@luban/HJY_X9y1le

[10] J. Labs, "Low Latency Block Updates (Shredstream)." [Online]. Available: https://docs.jito.wtf/lowlatencytxnfeed/

[11] Fabric, "Awesome Based Preconfirmations." [Online]. Available: https://eth-fabric.github.io/website/education/awesome-based-preconfs#introduction

[12] G. Spasov, "Preconfirmations Glossary & Requirements." [Online]. Available: https://hackmd.io/@Perseverance/Sy4a_BX2p

[13] Luganodes, "Preconfirmations: Explained." [Online]. Available: https://www.luganodes.com/blog/preconfirmations-explained/

[14] Ethereum, "Introduction to Smart Contracts." [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/

[15] Chainlink, "Maximal Extractable Value (MEV)." [Online]. Available: https://chain.link/education-hub/maximal-extractable-value-mev

[16] Ethereum, "Proposer-builder separation." [Online]. Available: https://ethereum.org/roadmap/pbs/

[17] ——, "The Merge – The Beacon Chain," GitHub repository. [Online]. Available: https://github.com/ethereum/annotated-spec/blob/master/merge/beacon-chain.md?utm_source#executionpayload

[18] M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, ser. CCS '97. New York, NY, USA: Association for Computing Machinery, 1997, pp. 1–6. [Online]. Available: https://doi.org/10.1145/266420.266424

[19] Flashbots, "Mev-boost," GitHub repository. [Online]. Available: https://github.com/flashbots/mev-boost

[20] A. Wahrstätter, "Mev-boost dashboard." [Online]. Available: https://mevboost.pics/

[21] L. Oshitani and J. Drake, "Eip-7917: Deterministic proposer lookahead." [Online]. Available: https://eips.ethereum.org/EIPS/eip-7917

[22] T. Beiko, A. Stokes, and A. Dietrichs, "EIP-7607: Hardfork Meta - Fusaka." [Online]. Available: https://eips.ethereum.org/EIPS/eip-7607

[23] V. Buterin, "How do layer 2s really differ from execution sharding?" [Online]. Available: https://vitalik.eth.limo/general/2024/05/23/l2exec.html

[24] J. Bostoen, "Examining the Based Sequencing Spectrum." [Online]. Available: https://research.chainbound.io/examining-the-based-sequencing-spectrum

[25] J. Drake, "Based rollups—superpowers from L1 sequencing." [Online]. Available: https://ethresear.ch/t/based-rollups-superpowers-from-l1-sequencing/15016

[26] Starknet, "Starknet Docs: Transactions." [Online]. Available: https://docs.starknet.io/learn/protocol/transactions

[27] Arbitrum, "The state of Arbitrum's progressive decentralization." [Online]. Available: https://docs.arbitrum.foundation/state-of-progressive-decentralization

[28] Starknet, "Starknet's token: STRK." [Online]. Available: https://governance.starknet.io/learn/starknet's_token:_strk

[29] Optimism, "Governance in Season 8: The Next Phase." [Online]. Available: https://optimism.mirror.xyz/JR5YEsK9-bM6At6c6iC5RiNNE4XXi0sMp3ytINq0wXw

[30] Primev, "Documentation - Understanding mev-commit." [Online]. Available: https://docs.primev.xyz/v1.1.0/concepts/network-overview

[31] G. Spasov and D. Ivanov, "Preconfirmations for Vanilla Based Rollups," GitHub repository. [Online]. Available: https://github.com/LimeChain/based-preconfirmations-research/blob/main/docs/preconfirmations-for-vanilla-based-rollups.md

[32] F. Casey-Fierro, C. McMenamin, L. Feroleto, and F. Mosterts, "A Pricing Model for Inclusion Preconfirmations." [Online]. Available: https://research.lido.fi/t/a-pricing-model-for-inclusion-preconfirmations/9136

[33] C. McMenamin, "Analysing Expected Proposer Revenue from Preconfirmations." [Online]. Available: https://research.lido.fi/t/analysing-expected-proposer-revenue-from-preconfirmations/8954

[34] J. Delong, "A Taxonomy of Preconfirmation Guarantees and Their Slashing Conditions in Rollups." [Online]. Available: https://ethresear.ch/t/a-taxonomy-of-preconfirmation-guarantees-and-their-slashing-conditions-in-rollups/22130

[35] Matthew, "The Preconfirmation Gateway ˜ Unlocking Preconfirmations: From User to Preconfer." [Online]. Available: https://ethresear.ch/t/the-preconfirmation-gateway-unlocking-preconfirmations-from-user-to-preconfer/18812

[36] I. Shaik, "Ahead-of-Time Block Auctions To Enable Execution Preconfirmations." [Online]. Available: https://ethresear.ch/t/ahead-of-time-block-auctions-to-enable-execution-preconfirmations/21345

[37] F. Mosterts and J. Bostoen, "Delegation in Bolt: Outsourcing Sophistication While Preserving Decentralization." [Online]. Available: https://research.chainbound.io/delegation-in-bolt-outsourcing-sophistication-while-preserving-decentralization

[38] C. McMenamin, "Preconfirmation Fair Exchange." [Online]. Available: https://ethresear.ch/t/preconfirmation-fair-exchange/21891

[39] C. Matt, "Leaderless and Leader-Based Preconfirmations." [Online]. Available: https://ethresear.ch/t/leaderless-and-leader-based-preconfirmations/19971

[40] Matthew, "Credibly Neutral Preconfirmation Collateral: The Preconfirmation Registry." [Online]. Available: https://ethresear.ch/t/credibly-neutral-preconfirmation-collateral-the-preconfirmation-registry/19634

[41] Spire Labs, "Preconfirmation Registry," GitHub repository. [Online]. Available: https://github.com/spire-labs

[42] Fabric, "Universal Registry Contract." [Online]. Available: https://eth-fabric.github.io/website/development/l1-components/urc

[43] Primev, "Documentation - Registering as a Provider." [Online]. Available: https://docs.primev.xyz/v1.1.0/get-started/providers/registering-a-provider

[44] EigenCloud, "Restaking Overview." [Online]. Available: https://docs.eigencloud.xyz/products/eigenlayer/restakers/concepts/overview

[45] B. Wee, "Restaking 101: A Primer on EigenLayer." [Online]. Available: https://medium.com/%40benhwx/restaking-101-a-primer-on-eigenlayer-ad9bc69875bc

[46] Fabric, "GitHub - Universal Registry Contract," GitHub repository. [Online]. Available: https://github.com/eth-fabric/urc

[47] Matthew, "Unified Preconf Registry." [Online]. Available: https://hackmd.io/@mteam/unfiedpreconfregistry

[48] G. Spasov and D. Ivanov, "Vanilla Based Sequencing." [Online]. Available: https://ethresear.ch/t/vanilla-based-sequencing/19379

[49] Luban, "Taiyi - off-chain components." [Online]. Available: https://docs.luban.wtf/learn/architecture/off_chain_components/overview

[50] Cairo, "Towards an implementation of based preconfirmations leveraging restaking." [Online]. Available: https://ethresear.ch/t/towards-an-implementation-of-based-preconfirmations-leveraging-restaking/19211

[51] G. Konstantopoulos and Q. Kilbourn, "Intent-Based Architecture and Their Risks." [Online]. Available: https://www.paradigm.xyz/2023/06/intents

[52] S. Brown, "Proposer-Commitment Infrastructure in Ethereum." [Online]. Available: https://simbro.medium.com/proposer-commitment-infrastructure-in-ethereum-61ad3b31f05f

[53] D. Van der Werff and A. Matthews, "Opportunities and Considerations of Ethereum's Blockspace Future." [Online]. Available: https://frontier.tech/ethereums-blockspace-future

[54] TL;DR The Latest in Defi Research, "Blockspace Futures." [Online]. Available: https://www.thelatestindefi.org/tldr/tldr-request-for-papers/blockspace-futures

[55] Espresso Systems, "The Derivation Pipeline." [Online]. Available: https://hackmd.io/@EspressoSystems/the-derivation-pipeline#Signing-atomic-bundle-transactions

[56] E. Davidson, "Analyzing BFT & Proposer-Promised Preconfirmations." [Online]. Available: https://ethresear.ch/t/analyzing-bft-proposer-promised-preconfirmations/17963

[57] Primev, "Documentation - Bid Decay Mechanism." [Online]. Available: https://docs.primev.xyz/v1.1.0/concepts/bids/bid-decay-mechanism

[58] Q. Kilbourn, "Order flow, auctions and centralisation II: order flow auctions." [Online]. Available: https://collective.flashbots.net/t/order-flow-auctions-and-centralisation-ii-order-flow-auctions/284

[59] J. Burian, "User-Defined Penalties: Ensuring Honest Preconf Behavior." [Online]. Available: https://ethresear.ch/t/user-defined-penalties-ensuring-honest-preconf-behavior/19545

[60] Primev, "Documentation - Bid Structure." [Online]. Available: https://docs.primev.xyz/v1.1.0/concepts/bids/bid-structure

[61] L. Oshitani, "Based Preconfirmations with Multi-round MEV-Boost." [Online]. Available: https://ethresear.ch/t/based-preconfirmations-with-multi-round-mev-boost/20091

[62] Primev, "Documentation - Commitments." [Online]. Available: https://docs.primev.xyz/v1.1.0/concepts/commitments

[63] N. Asokan, "Fairness in electronic commerce," Ph.D. dissertation, CAN, 1998, uML Order No. GAXNQ-32811.

[64] H. Pagnia and F. C. G. Darmstadt, "On the Impossibility of Fair Exchange without a Trusted Third Party," 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID:11671049

[65] A.-P. Stouka and C. McMenamin, "Future-Proofing Preconfirmations." [Online]. Available: https://ethresear.ch/t/future-proofing-preconfirmations/22618

[66] S. G. Kolliopoulos and G. Steiner, "Partially ordered knapsack and applications to scheduling," *Discrete Applied Mathematics*, vol. 155, no. 8, pp. 889–897, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X0600388X

[67] A. Ben-Eliezer and N. Nisan, "Online Block Packing." [Online]. Available: https://www.arxiv.org/pdf/2507.12357

[68] Fabric, "Constraints API Specification," GitHub repository. [Online]. Available: https://github.com/eth-fabric/constraints-specs/blob/main/specs/constraints-api.md

[69] ——, "Fault Attribution," GitHub repository. [Online]. Available: https://github.com/eth-fabric/constraints-specs/blob/main/specs/fault-attribution.md

[70] RISE Book, "Based preconfirmations." [Online]. Available: https://docs.risechain.com/rise-stack/based-preconfs.html

[71] Matthew, "Avoiding Accidental Liveness Faults for Based Preconfs." [Online]. Available: https://ethresear.ch/t/avoiding-accidental-liveness-faults-for-based-preconfs/19888

[72] J. Drake, "Based preconfirmations." [Online]. Available: https://ethresear.ch/t/based-preconfirmations/17353

[73] Fabric, "GitHub - Example Slasher Implementations," GitHub repository. [Online]. Available: https://github.com/eth-fabric/urc/tree/main/example

[74] U. Natale, "Preconfirmations under the NO lens." [Online]. Available: https://ethresear.ch/t/preconfirmations-under-the-no-lens/19975

[75] F. Casey-Fierro and C. McMenamin, "Estimating the Revenue from Independent Sub-Slot Auction Preconfirmations." [Online]. Available: https://research.lido.fi/t/estimating-the-revenue-from-independent-sub-slot-auction-preconfirmations/8801

[76] M. Akdeniz, "Measuring validator economics under preconfirmations: Early mainnet evidence." [Online]. Available: https://ethresear.ch/t/measuring-validator-economics-under-preconfirmations-early-mainnet-evidence/

[77] T. Roughgarden, "Transaction fee mechanism design for the ethereum blockchain: An economic analysis of EIP-1559," *CoRR*, vol. abs/2012.00854, 2020. [Online]. Available: https://arxiv.org/abs/2012.00854

[78] B. Robaglia, U. Natale, and M. Moser, "Pricing Transactions for Preconfirmation." [Online]. Available: https://ethresear.ch/t/pricing-transactions-for-preconfirmation/21802

[79] Commit-Boost, "Commit-Boost Docs - Introduction." [Online]. Available: https://commit-boost.github.io/commit-boost-client/

[80] M. Neuder, "Resistance is not futile; CR in mev-boost." [Online]. Available: https://ethresear.ch/t/resistance-is-not-futile-cr-in-mev-boost/16762

[81] J. Milionis, C. C. Moallemi, and T. Roughgarden, "Automated market making and arbitrage profits in the presence of fees," in *Financial Cryptography and Data Security*, J. Clark and E. Shi, Eds. Cham: Springer Nature Switzerland, 2025, pp. 159–171.

[82] dataalways, "History of Block Builders." [Online]. Available: https://dune.com/queries/3598078/6062171

[83] Arbitrum, "Timeboost." [Online]. Available: https://docs.arbitrum.io/how-arbitrum-works/timeboost/gentle-introduction

[84] Entropy Advisors, "Arbitrum Timeboost: Auction Winner Breakdown." [Online]. Available: https://dune.com/entropy_advisors/arbitrum-timeboost

[85] Beincrypto, "DeFi in Crisis: Restaking Protocols Are Devouring Liquidity." [Online]. Available: https://beincrypto.com/restaking-protocols-are-devouring-liquidity/

[86] D. Kuhn, "What the DOJ's First MEV Lawsuit Means for Ethereum." [Online]. Available: https://www.coindesk.com/opinion/2024/05/15/what-the-dojs-first-mev-lawsuit-means-for-ethereum

[87] D. Marzec and L. Thibault, "Subverting the total eclipse (of the heart)." [Online]. Available: https://hackmd.io/@dmarz/total-eclipse-of-the-relay

[88] G. Longstaff, "What is contract law?" [Online]. Available: https://www.law.ac.uk/resources/blog/what-is-contract-law/

[89] C. McMenamin, "The Preconfirmation Sauna." [Online]. Available: https://ethresear.ch/t/the-preconfirmation-sauna/19762

[90] P. Daian, "Decentralized crypto needs you: to be a geographical decentralization maxi." [Online]. Available: https://collective.flashbots.net/t/decentralized-crypto-needs-you-to-be-a-geographical-decentralization-maxi/1385

[91] T. Thiery, F. D'Amato, J. Ma, B. Monnot, T. Tsao, J. Kaufmann, and J. Song, "EIP-7805: Fork-choice enforced Inclusion Lists (FOCIL)." [Online]. Available: https://eips.ethereum.org/EIPS/eip-7805

[92] A.-P. Stouka, J. Ma, and T. Thiery, "Multiple proposer transaction fee mechanism design: Robust incentives against censorship and bribery," 2025. [Online]. Available: https://arxiv.org/abs/2505.13751

[93] B. Monnot, "Unbundling staking: Towards rainbow staking." [Online]. Available: https://ethresear.ch/t/unbundling-staking-towards-rainbow-staking/18683

[94] T. Thiery, "Three-Tier staking (3TS) - Unbundling Attesters, Includers and Execution Proposers." [Online]. Available: https://ethresear.ch/t/three-tier-staking-3ts-unbundling-attesters-includers-and-execution-proposers/21648

[95] C. McMenamin, "Appointed Execution Propoers: Because the Proposer you know..." [Online]. Available: https://ethresear.ch/t/appointed-execution-proposers-because-the-proposer-you-know/19284

[96] L. Oshitani, "Strawmanning Based Preconfirmations." [Online]. Available: https://ethresear.ch/t/strawmanning-based-preconfirmations/19695

[97] Optimism, "Optimism Docs." [Online]. Available: https://docs.optimism.io/

[98] Primev Team, "mev-commit Whitepaper." [Online]. Available: https://whitepaper.mev-commit.xyz/

[99] K. Lepsoe, "Introducing ETHGas and Realtime Proposer Commitments to the Lido Community." [Online]. Available: https://research.lido.fi/t/introducing-ethgas-and-realtime-proposer-commitments-to-the-lido-community/9018

[100] ETHGas, "ETHGas Docs - Overview." [Online]. Available: https://docs.ethgas.com/our-technology/overview

[101] ——, "ETHGas preconf commit-boost module," GitHub repository. [Online]. Available: https://github.com/ethgas-developer/ethgas-preconf-commit-boost-module

[102] ——, "ETHGas Docs - Validators." [Online]. Available: https://docs.ethgas.com/blockspace-actors-and-integrations/validators

[103] ——, "ETHGas Docs - Commitment Buyers." [Online]. Available: https://docs.ethgas.com/blockspace-actors-and-integrations/commitment-buyers

[104] ——, "ETHGas Docs - The ETHGas Architecture." [Online]. Available: https://docs.ethgas.com/our-technology/the-ethgas-architecture

[105] ——, "ETHGas Docs - Builders & Sequencers." [Online]. Available: https://docs.ethgas.com/blockspace-actors-and-integrations/builders-and-sequencers

[106] ——, "ETHGas Docs - Relays." [Online]. Available: https://docs.ethgas.com/blockspace-actors-and-integrations/relays

[107] S. Monn and B. Bengisu, "Erc-7521: General intents for smart contract wallets." [Online]. Available: https://eips.ethereum.org/EIPS/eip-7521

[108] IQ.wiki, "Anoma." [Online]. Available: https://iq.wiki/wiki/anoma

[109] C. DAO, "Solvers." [Online]. Available: https://docs.cow.fi/cow-protocol/concepts/introduction/solvers

[110] Blackwing, "Intent Solver Architecture." [Online]. Available: https://docs.blackwing.fi/core_concepts/intent_solver

A Intents and Intent Preconfs

In an intent-based architecture, users submit intents instead of fully specified transactions (e.g., see [51, 107]). Intents are artifacts that describe a user's desired outcome without prescribing the exact execution path. More specifically, an intent includes a predicate that defines the conditions under which the user permits their funds to be moved (see [108]). For example, a user might express a willingness to exchange at most X units of token A for Y units of token B, without specifying the particular route or counterparty. Entities known as solvers collect these intents and bundle them into a single transaction, which they then submit to the blockchain for inclusion. By aggregating multiple intents – including potentially their own – solvers may extract MEV.

The structure of intents, their execution logic, the nature of solvers, and the incentive mechanisms (such as tips sent by the users or MEV extraction) vary depending on the specific intent-based architecture. While the design of preconfs in these systems remains largely unexplored, to the best of our knowledge, intent requests should resemble preconf requests. Moreover, intent solvers may act as preconfers who commit to fulfilling intents; if they succeed, they receive a reward (e.g., see [109]), but if they fail, they may be slashed (see [110]).