Boost+: Equitable, Incentive-Compatible Block Building

Fan Zhang
Yale University

Feb 2nd, 2026

Based on joint work with Mengqian Zhang, Sen Yang, and Kartik Nayak.

My Group at Yale CS

- We are a group working on security of decentralized systems.
- Recent topics
  - MEV (today's talk)
  - Censorship
  - Anonymity
  - Verifiability

Members

Fan Zhang
Faculty
Website

Aviv Yaish
Postdoc
Website

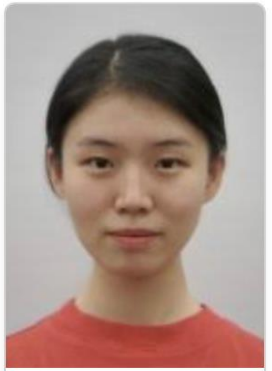
Tiantian Gong
Postdoc
Website

Sarisht Wadhwa
PhD Student
(Duke)
Website

Sen Yang
PhD Student
Website

Giannis Kaklamanis
PhD Student
Website

Wenhao Wang
PhD Student
Website



Yunhao Wang
PhD Student
Website

MEV (Daian et al '19)

Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges

Philip Daian, *Cornell Tech*, phil@cs.cornell.edu
Steven Goldfeder, *Cornell Tech*, goldfeder@cornell.edu
Tyler Kell, *Cornell Tech*, sk3259@cornell.edu
Yunqi Li, *UIUC*, yunqil3@illinois.edu
Xueyuan Zhao, *CMU*, xyzhao@cmu.edu
Iddo Bentov, *Cornell Tech*, ib327@cornell.edu
Lorenz Breidenbach, *ETH Zürich*, lorenz.breidenbach@inf.ethz.ch
Ari Juels, *Cornell Tech*, juels@cornell.edu

Ordering manipulation

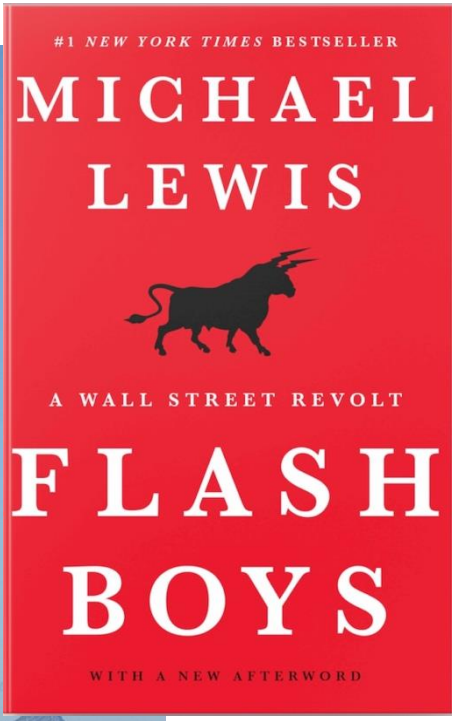
```
Alice: sell @ $90
Adv  :  buy @ $90
Adv  : sell @ $100
```

Frontrunning attack

Bob	-10
Adv	**+10**

Adv "extracted" $10 by ordering txns cleverly

- How to insert txns?
- E.g., HFT firms gain timing advantage through co-location, low-latency networks, etc



Ordering manipulation

```
Alice: sell @ $90
Adv  :  buy @ $90
Adv  : sell @ $100
Bob  :  buy @ $100
```

Frontrunning attack

Bob	-10
Adv	**+10**

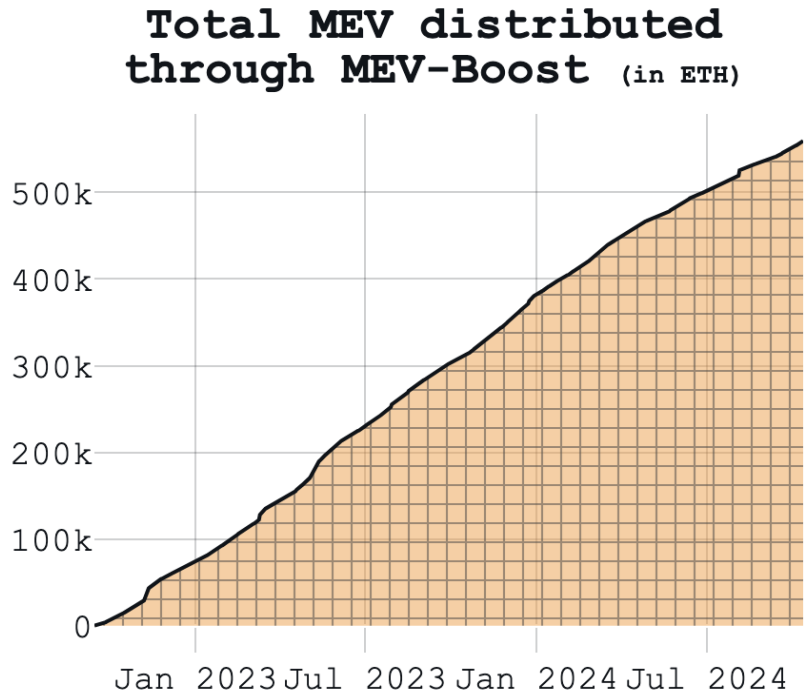
Adv "extracted" $10 by ordering txns cleverly

- How to insert txns to exchanges on blockchain?
- Block creators *completely* controls transaction ordering.
- Power → money (aka MEV)
- Miner/Maximal Extractable Value (MEV) refers to the profits gained from manipulating the ordering of transactions.

Quantitative estimation of MEV

MEV Transaction Profit Leaderboard

Type	Contract	Profit	Time(UTC)
Arbitrage	0x5dC…26fB7	$48,863.70	2024-10-12
Sandwich	0x000…20E49	$31,459.77	2024-10-09
Arbitrage	0x63d…1CfeB	$27,878.23	2024-10-09
Arbitrage	0x580…dc86c	$23,773.39	2024-10-11
Arbitrage	0xfd7…FfF8c	$20,951.53	2024-09-20
Arbitrage	0xdD3…dD189	$19,505.23	2024-09-19
Arbitrage	0xF46…FcD5c	$18,241.68	2024-09-20
Arbitrage	0x6C9…40Eac	$16,995.31	2024-09-20
Arbitrage	0x454…C8480	$16,953.08	2024-10-12
Arbitrage	0xd45…96221	$16,928.25	2024-09-21

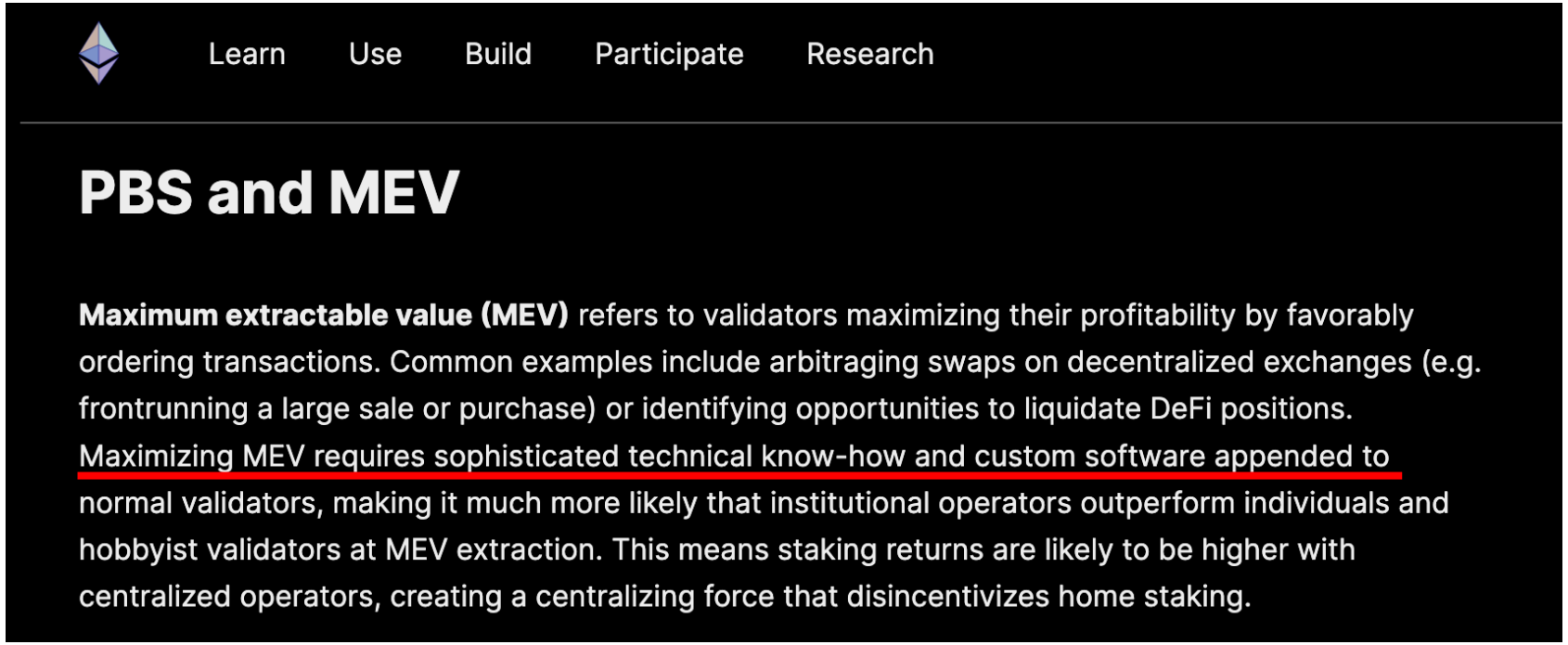


Over **550K ETH (~$1.3B)** has been extracted on Ethereum!

Security implications of MEV

- **User loss**: some MEV extraction directly harm users
  - E.g., sandwich attacks
- **Network congestion**: wasteful transactions due to MEV extraction
  - MEV bots consume approximately 40% of processing capacity on Solana
  - During a period in 2025, arbitrary on Base ate almost 100% of added throughput.
- **Consensus instability**: mev dominates fees
- **Centralization force**: Resourced validators (e.g., institutional operators) may drive small validators (e.g., home stakers) out of the MEV market.

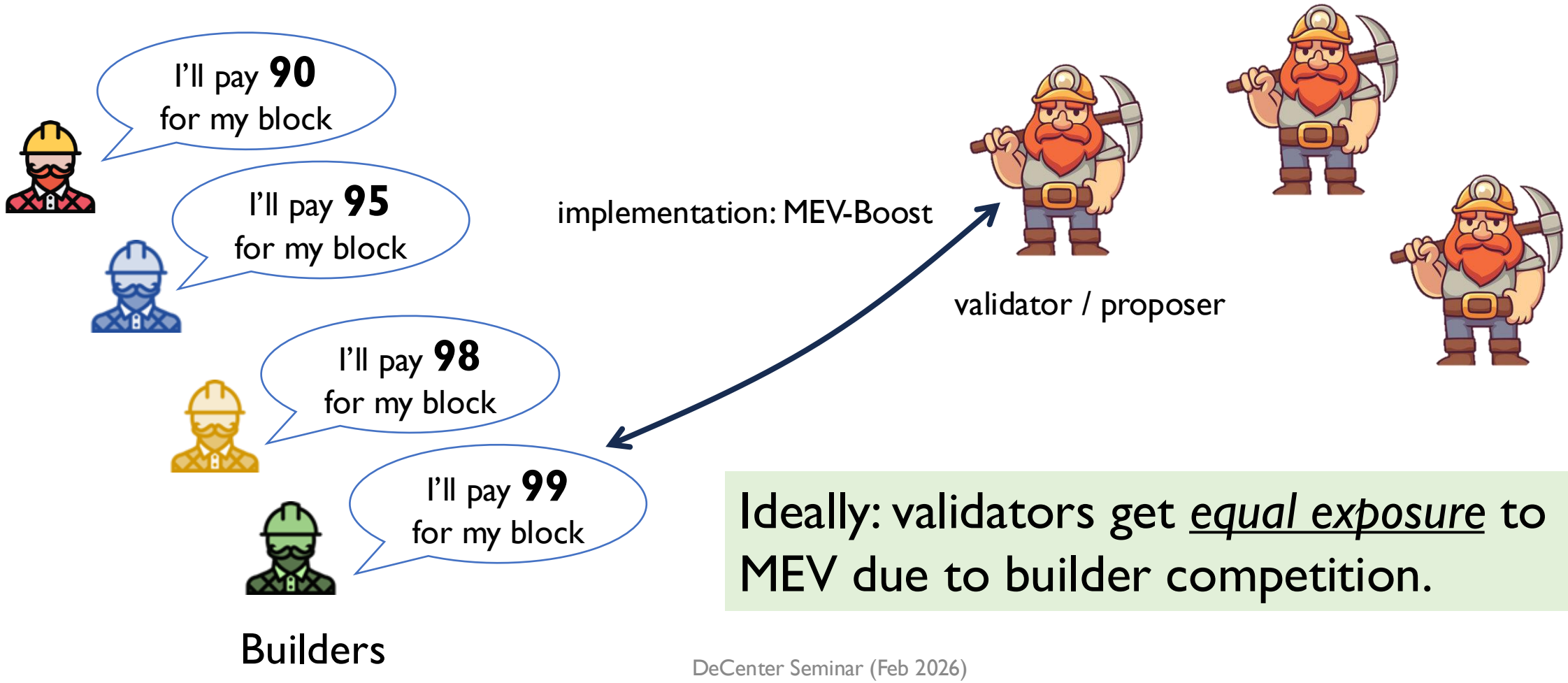
Ethereum's MEV strategy



- Response: Proposer Builder Separation (PBS)

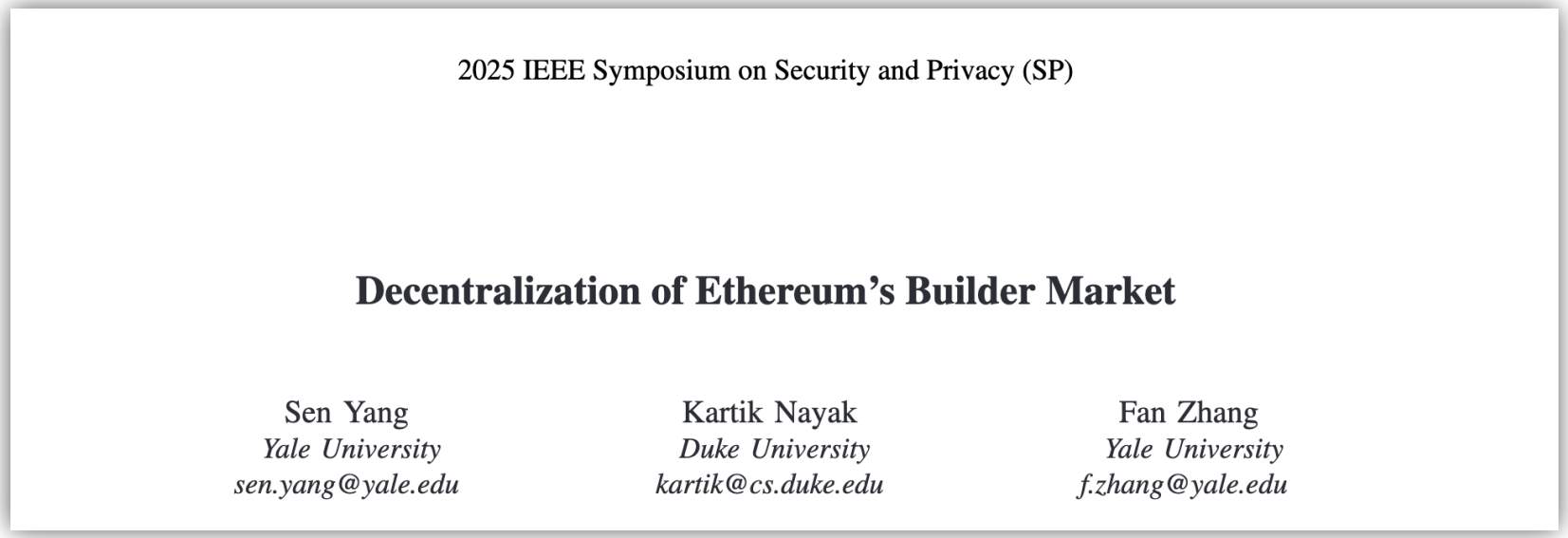
Ethereum's MEV strategy

- Idea: outsource of block building (i.e., MEV extraction) to builder market.



Ideally: validators get *equal exposure* to MEV due to builder competition.

How well does builder market work?



2025 IEEE Symposium on Security and Privacy (SP)

Decentralization of Ethereum's Builder Market

Sen Yang
Yale University
sen.yang@yale.edu

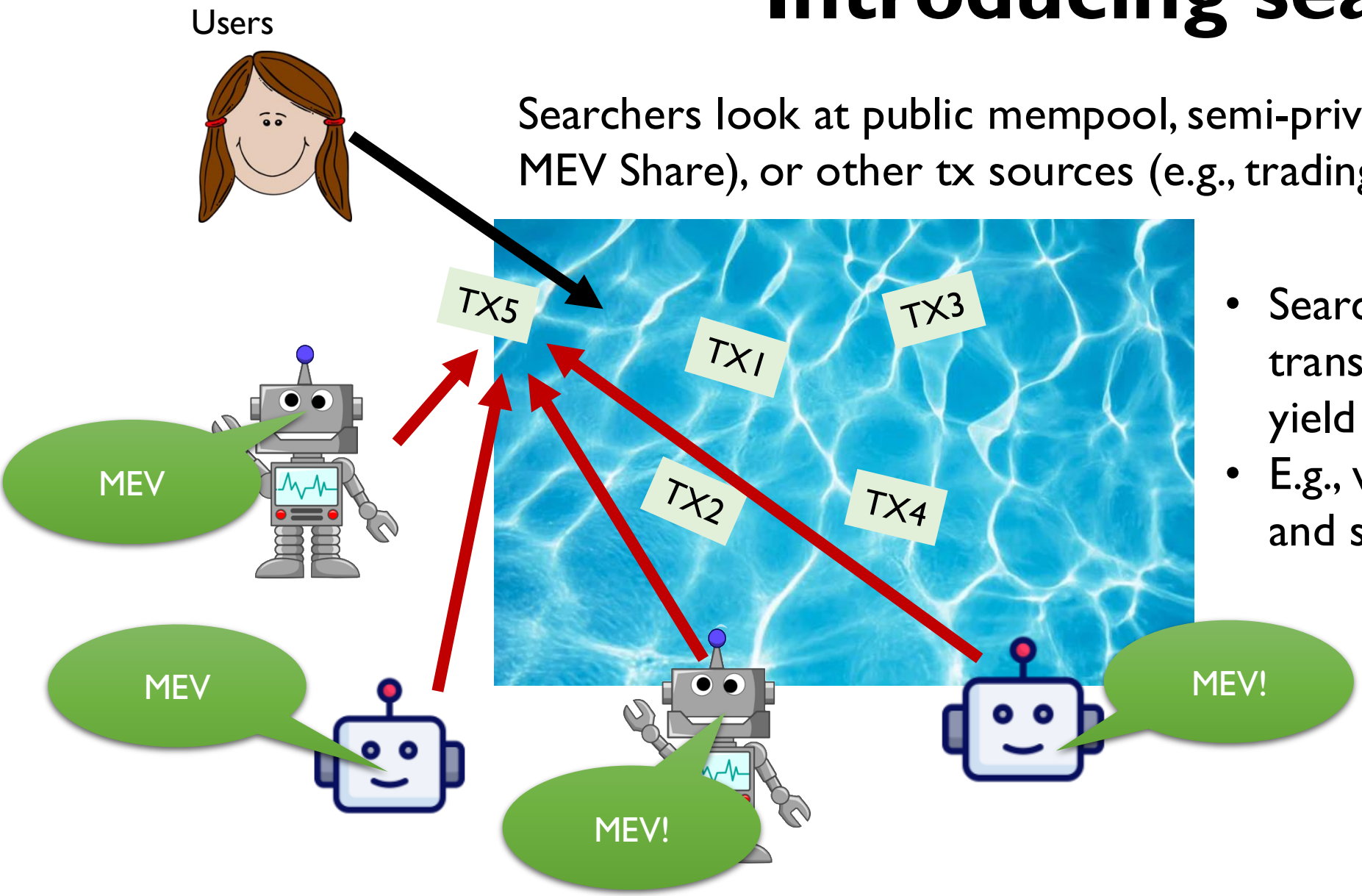
Kartik Nayak
Duke University
kartik@cs.duke.edu

Fan Zhang
Yale University
f.zhang@yale.edu

- Block building involves three parties: searchers, builders, and proposers.
- PBS only manages Builder-Proposer interaction. Searcher-builder interaction is left *un-regulated*...
- …which has led to collusive patterns known as **integration**
- …Boost+ aims to prevent integration.

Introducing searchers

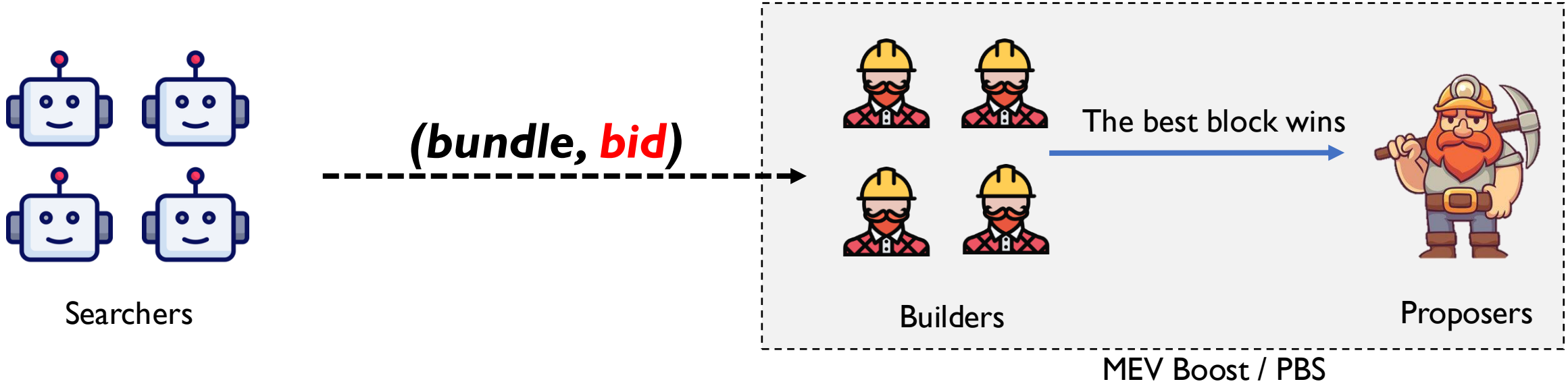
Searchers look at public mempool, semi-private channels (e.g., MEV Share), or other tx sources (e.g., trading bots, wallets)



- Searchers create sequence of transactions ("bundles") that yield MEV
- E.g., via arbitrage, liquidations, and sandwich

Searchers <> Builders <> Proposers

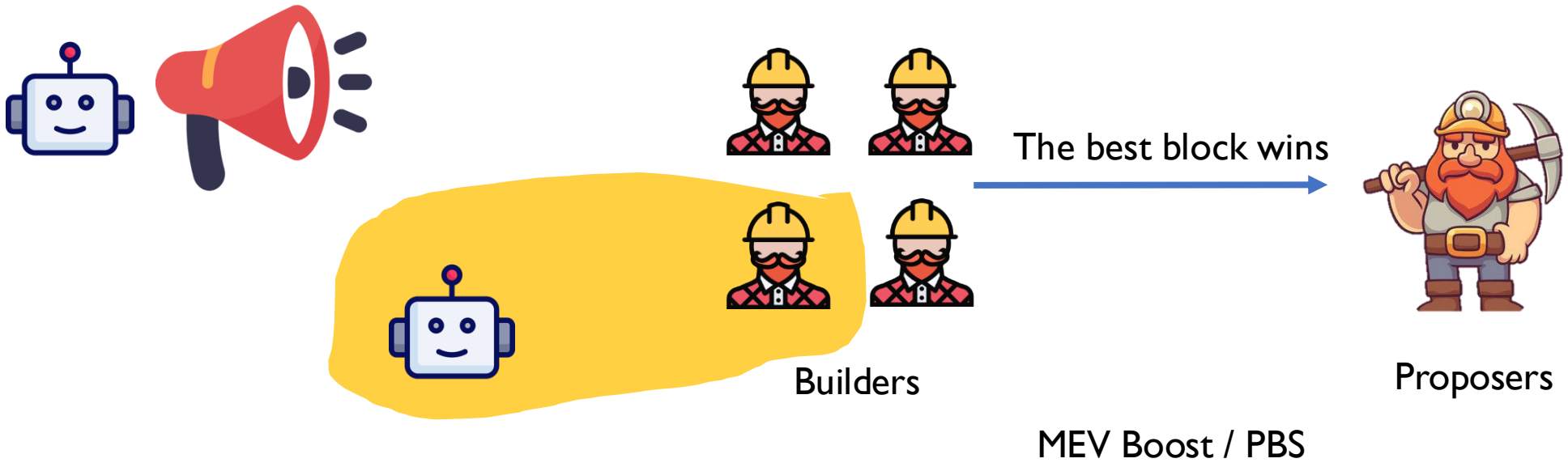
image: Flaticon.com



- Searchers send bundles to builders, offering a bid for inclusion
- Builders build blocks out of bundles (and other txns)

Searcher-builder integration

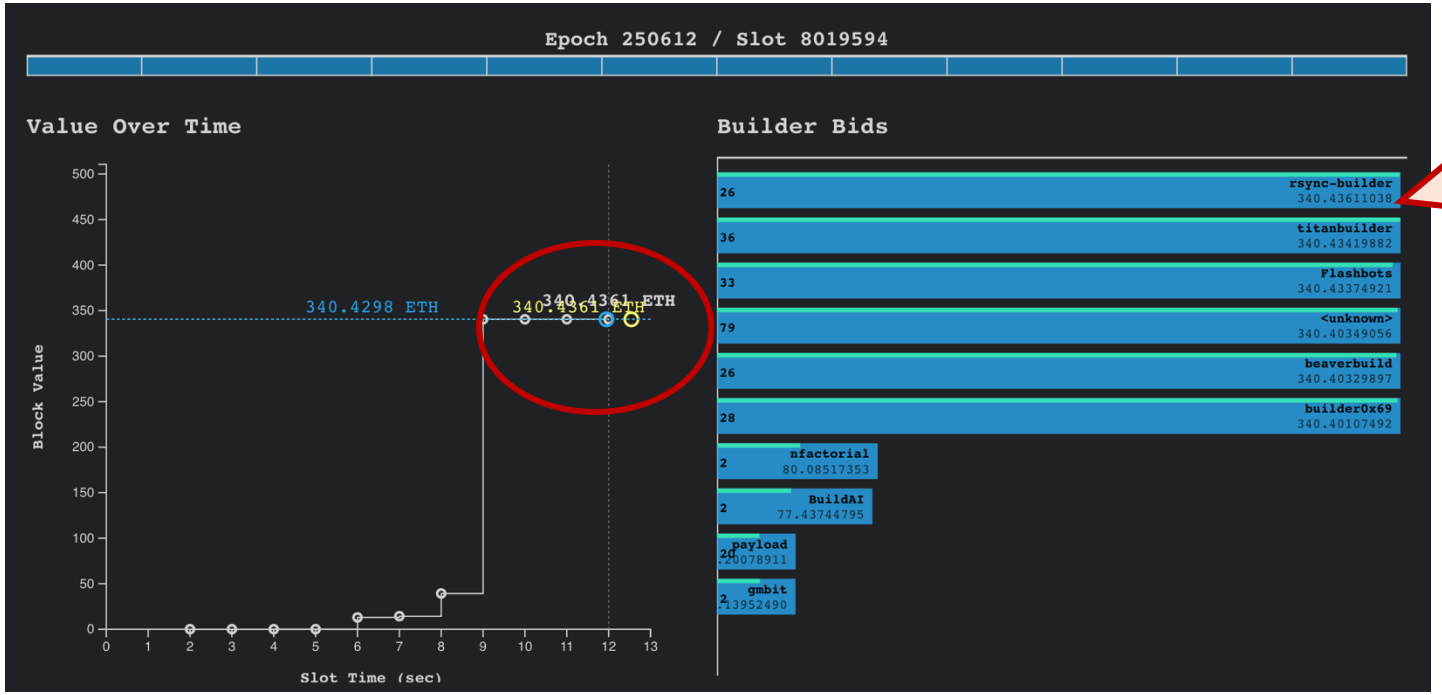
image: Flaticon.com



- Searchers can choose which builders to contact
- Some multicast (c.f. Flashbots' forward service)
- But searcher may also be (almost) exclusive w a builder (off-chain deals)
  - E.g., Banana Gun (s) and Titan (b)
- Why? To avoid competition in MEV Boost/PBS

Example: Incentive for integration

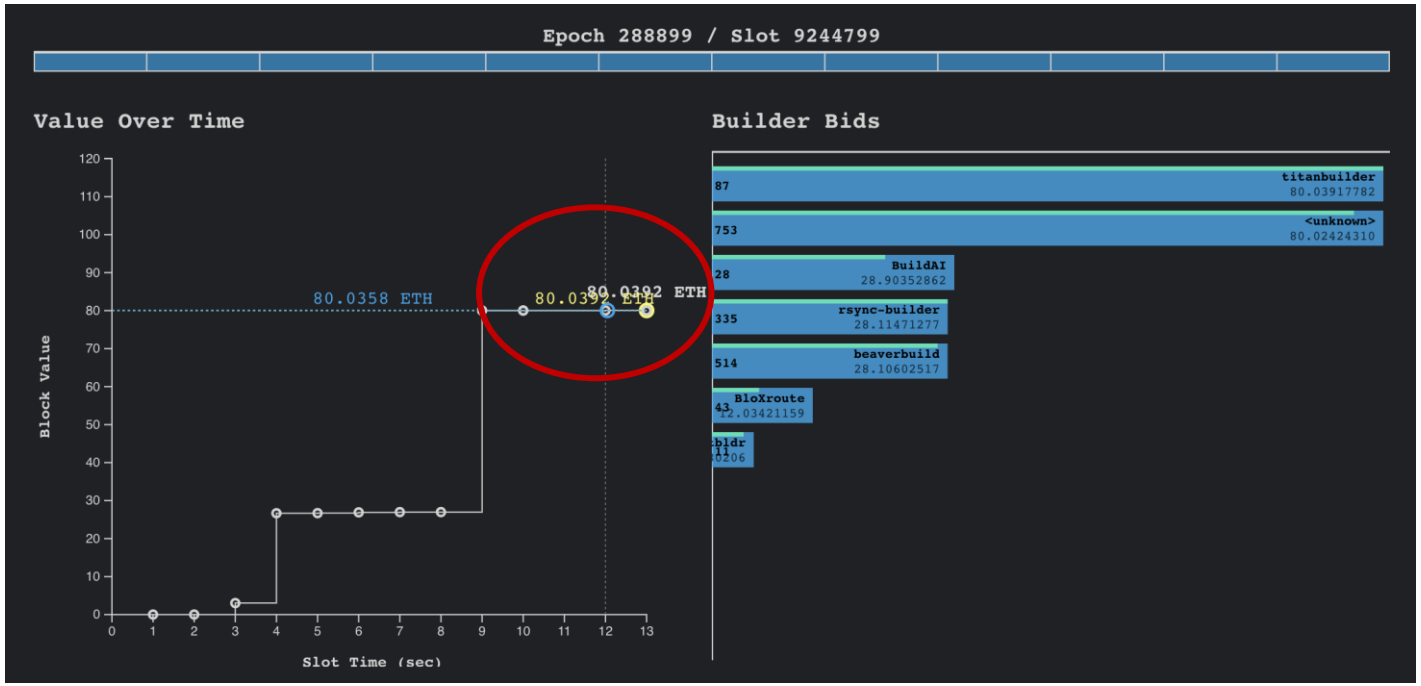
- For example, in slot 8019594, about 340 ETH came from Banana Gun (OF), and all 340 ETH was captured by the proposer.



All top builders bid 340 ETH.

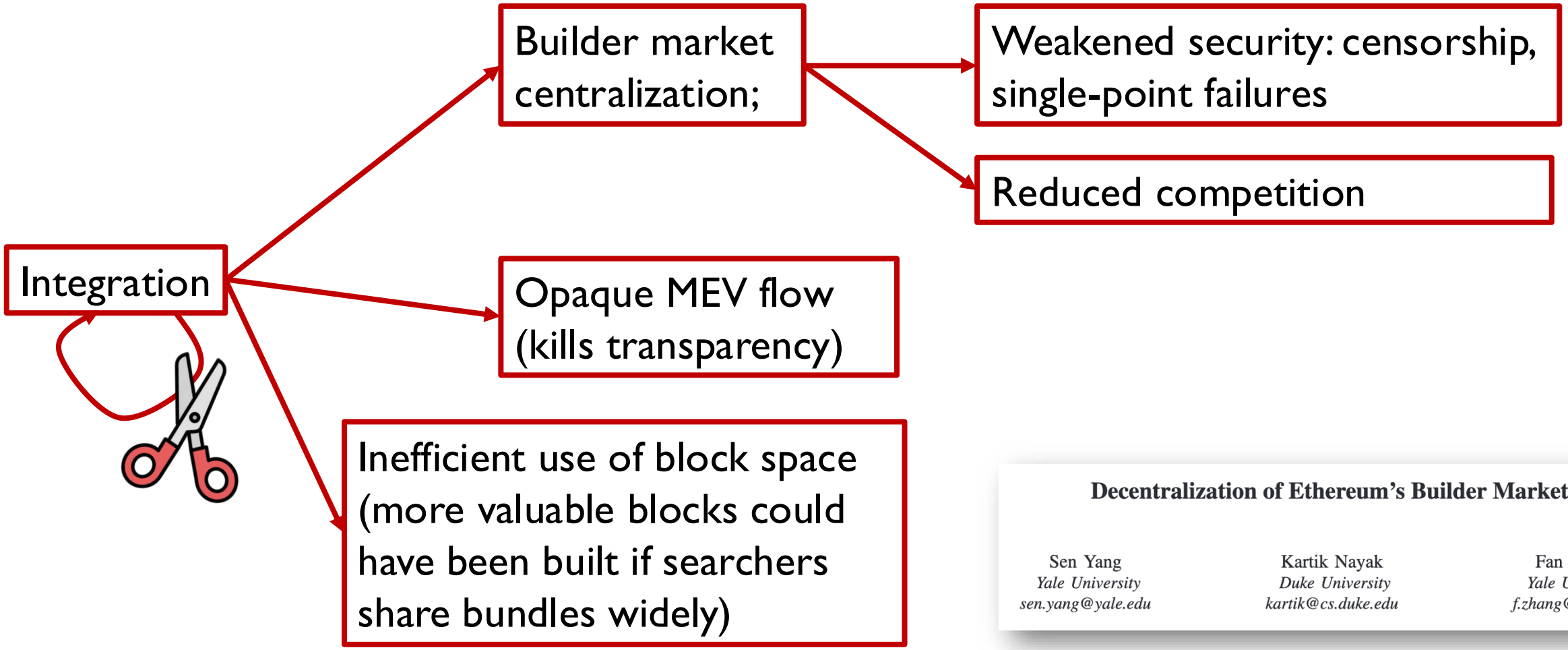
Example: Incentive for integration

- In slot 9244799, 208 ETH came from Banana Gun, and only Titan received it.
- The second-best bid 80 ETH. Titan won 80+epsilon.
- 128 ETH can be shared between Banana Gun and Titan!



Titan dominates this auction.

Implications of integration



Decentralization of Ethereum's Builder Market

Sen Yang
Yale University
sen.yang@yale.edu

Kartik Nayak
Duke University
kartik@cs.duke.edu

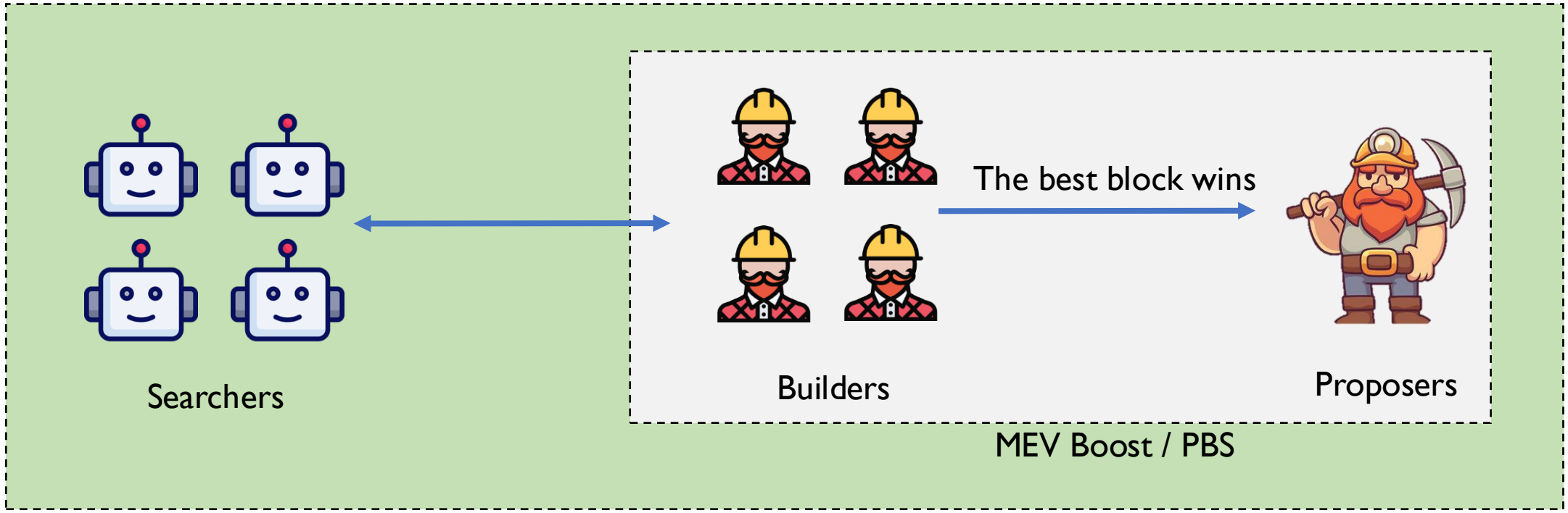
Fan Zhang
Yale University
f.zhang@yale.edu

Boost+: Equitable, Incentive-Compatible Block Building

Joint work with Mengqian Zhang, Sen Yang, Kartik Nayak

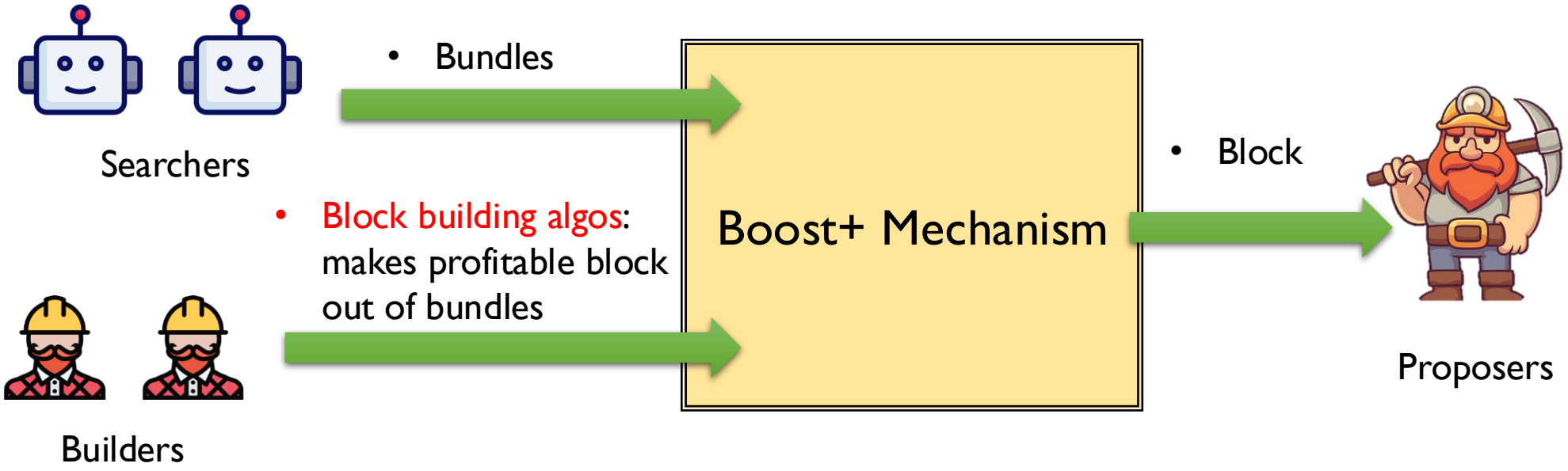
Observation

- MEV Boost leave searcher-builder interaction open, and the system gravitates towards integration
- A solution must explicitly consider searchers as participants.

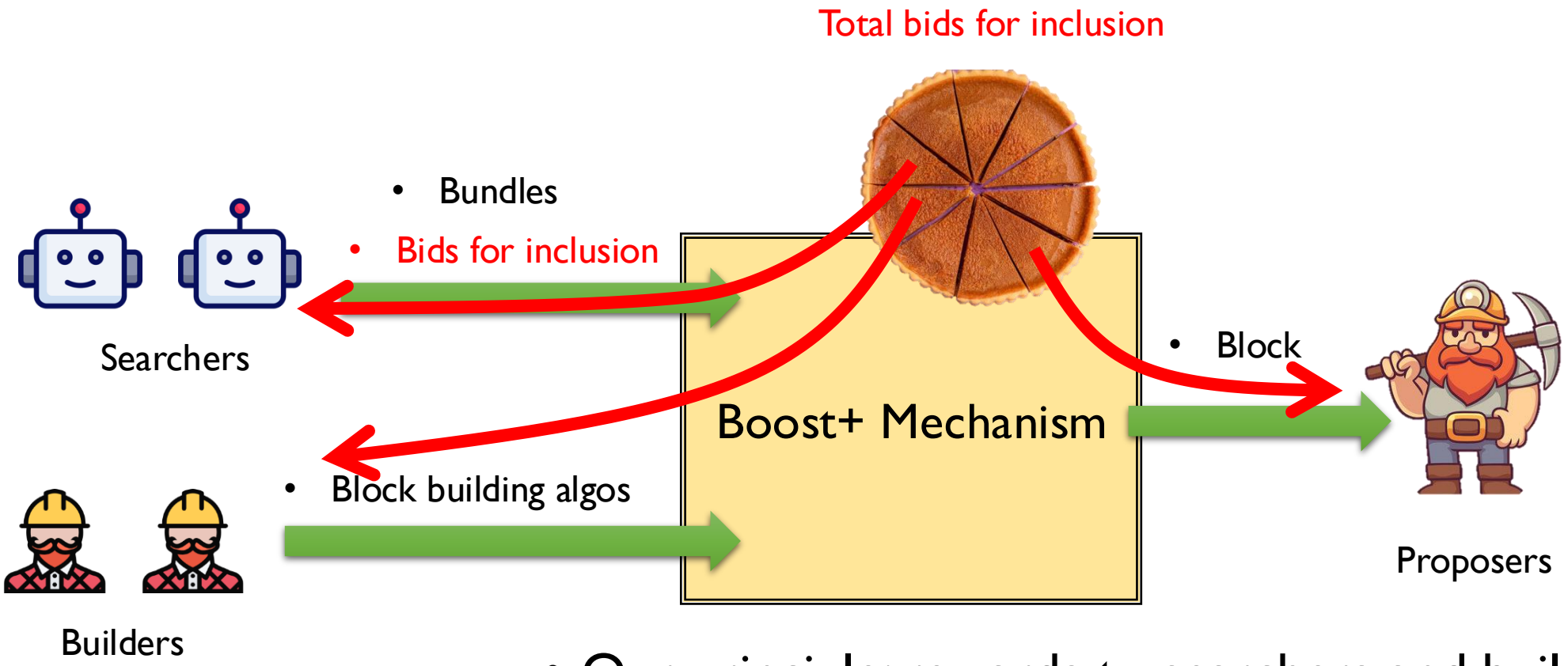


Boost+ Architecture

- Rethinking block building if there was no MEV-Boost/PBS
  - Key difference from BuilderNet (their goal is to build a distributed builder *in* MEV Boost)
- Justification: for blockchains without PBS or where auctions are too slow to run



Boost+ Architecture



- Our principle: rewards to searchers and builder depend on the marginal contribution of bundles & algorithms.

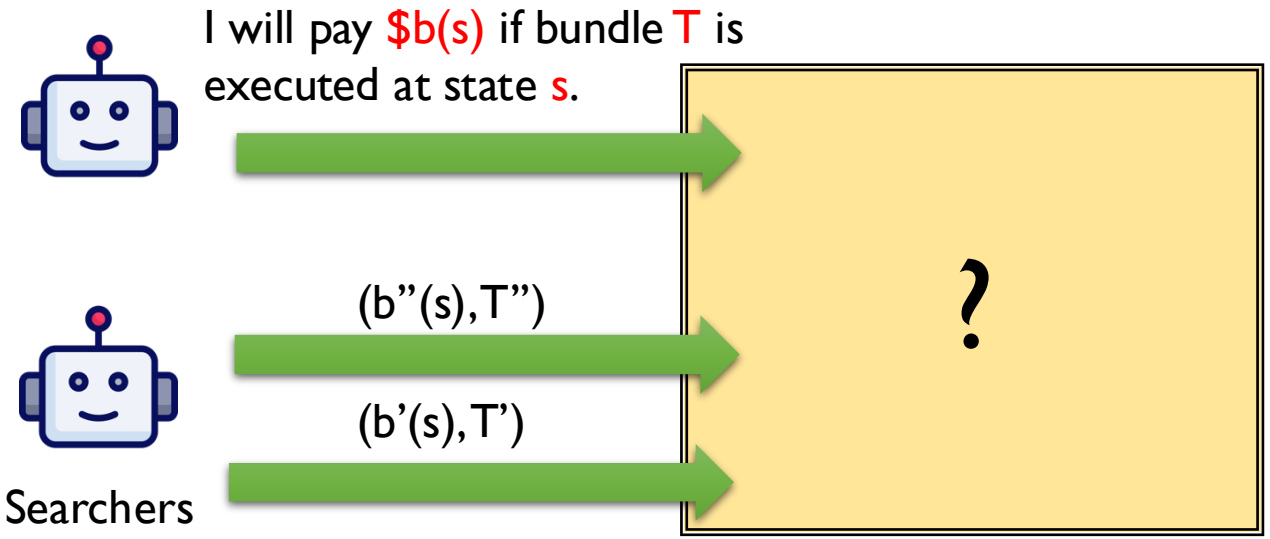
Goals of the mechanism design

- This architecture offers opportunity to achieve formal guarantees for searcher-builder interactions, including

- Truthful bidding by searchers.
  - ➔ Necessary for efficiently allocating the scarce block space.
- Truthful bidding by builders.
  - ➔ Necessary for fair builder rewards
- Equal access to all order flows.
  - ➔ Integration is not beneficial

Modeling searchers

- Searcher's willingness to pay for a bundle depends on blockchain state
- E.g., arb bots: if gain(s)> 0, bids 0.8*gain(s); bids 0 otherwise.
- Captured by a bidding function b(s) over state s



VCG mechanism can be used to

- identify the state that maximizes $\sum_i b_i(s)$, and
- ensure that truthful reporting is dominant.

But impractical!

Attempt #1: Direct application of VCG

Inputs

- M: a set of bundles from searchers
- $b_i(s)$: bidding function for bundle

Result:

- DSIC for searchers
- Welfare-maximizing
- Computationally inefficient

VCG Mechanism:

- Compute the optimal block that maximizes the total fee by all bundles $\omega^* = \arg\max_{\omega} \sum_{i \in M} b_i(\omega)$
- ω^* is output; each bundle is charged $b_i(\omega^*)$
- For each bundle i, compute a refund

$$r_i = \sum_{j \in M} b_j(\omega^*) - \max_{\omega \in \Omega} \sum_{j \neq i} b_j(\omega)\,;$$

Best block value

Best block value without i

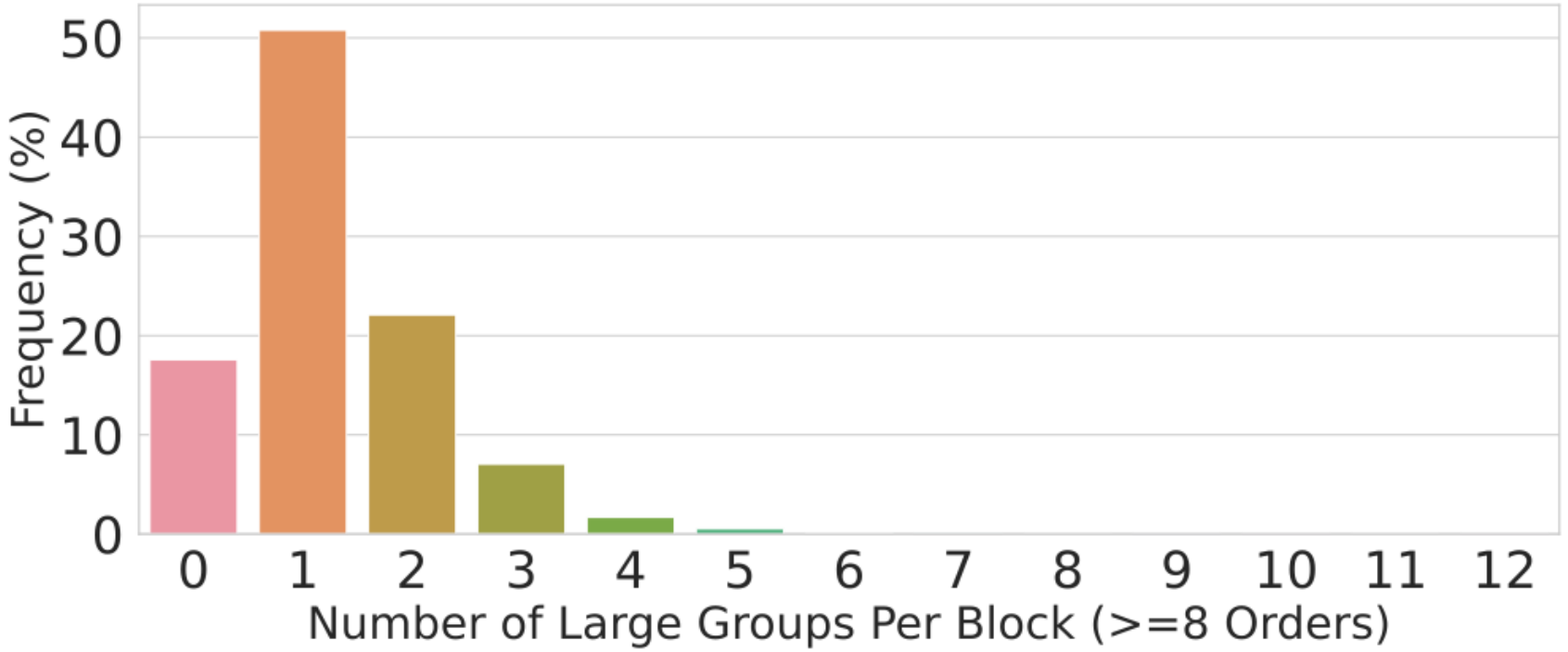
- Insert txns to ω^* to send r_i to bundle i
- Insert txns to send total pay $\sum_{i \in M}(b_i(\omega^*) - r_i)$ to the proposer

Attempt #2: Reducing search space

- To be practical, we
  - only considering ordering of bundles, and
  - search for a good (maybe not best) ordering while maintain truthfulness.
- Observations:
  - If two transactions do not touch any shared storage, they are independent
  - Relative ordering of independent transactions do not matter.
- Idea:
  1. Group transactions into conflict groups (two txns in the same group if they are not independent)
  2. Exhaustively search for an optimal ordering for small groups (e.g., < 8)
  3. For larger groups, apply ad-hoc rules to identify good ordering
  4. Combine ordered groups to get final block

Inspiration: conflicts of Ethereum TXs

- Analyzed tx independence in 10,000 blocks in Feb 2025
- ~18% has no large conflict groups (>8)
- ~50% has only one.



Attempt #2: reduce search space

- Idea: Replace exhaustive search in the VCG mechanism (in attempt #1) with this efficient algorithm $\mathcal{A}$
  - $\mathcal{A}$ is called the default algorithm in the paper
- Truthfulness holds as long as $\mathcal{A}$ is "doesn't look at bids"

- Theorem: For any $\mathcal{A}$ satisfying the following two requirements, the mechanism $\text{VCG}_{\mathcal{A}}$ is efficient and DSIC for searchers.
  - $\Omega_{\mathcal{A}}$ independent of bids, i.e., $\Omega_{\mathcal{A}}(M, \mathbf{b}) = \Omega_{\mathcal{A}}(M, \mathbf{b}')$ for $\forall\ \mathbf{b}, \mathbf{b}'$.
  - $\mathcal{A}$ can efficiently compute optimal and counterfactual blocks.

Attempt #2: Efficient VCG-based Mechanism

VCG-based Mechanism $\mathrm{VCG}_{\mathcal{A}}$:

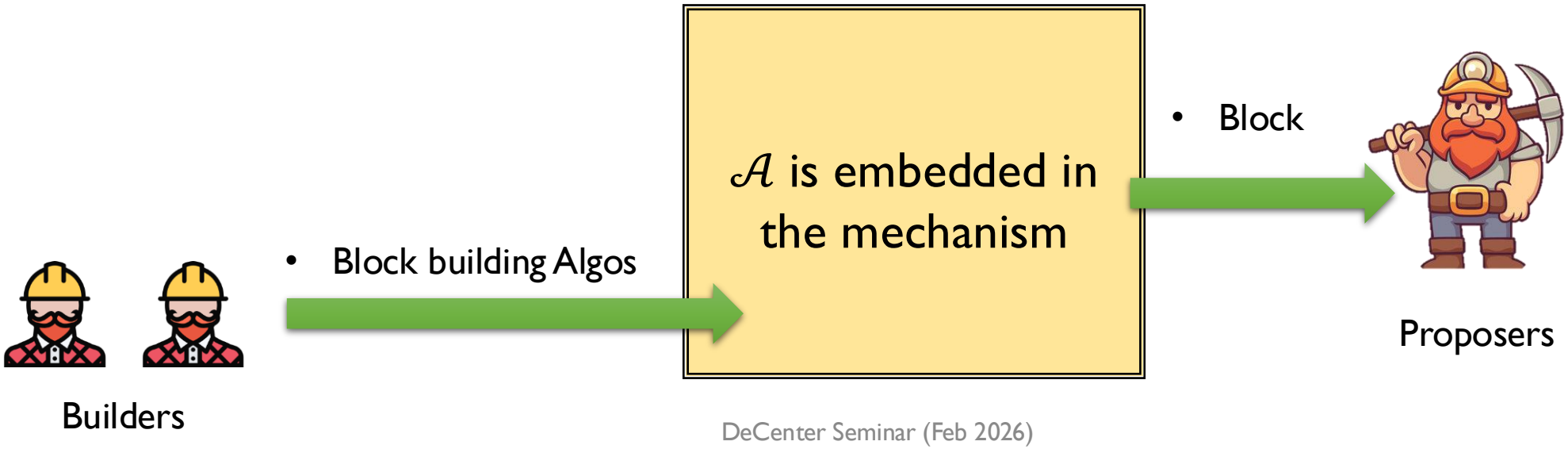
1. Run $\mathcal{A}$ to get the "optimal" blocks o^* and $\{o_{-i}\}_{i \in M}$, and outputs o^* as the final block;
2. Charge each bundle i the amount $b_i(o^*)$;
3. Refund each bundle $r_i(\mathbf{b}) = \sum_{j \in M} b_j(o^*) - \sum_{j \neq i} b_j(o_{-i})$;
4. Transfer the remaining amount $\sum_{i \in M}(b_i(o^*) - r_i(\mathbf{b}))$ to the proposer.

Theorem: For any $\mathcal{A}$ satisfying the two requirements, the mechanism $\mathrm{VCG}_{\mathcal{A}}$ is efficient and DSIC for searchers.

Tradeoff: runs in poly time, remains IC, but sacrifice social welfare (sum of total fees).

Keep improving: incorporate builders

- Idea: leverage builder's algorithms to improve block value
- Challenge: builder algorithms can be arbitrary
  - They may not satisfy the requirement of Thm 2
  - It seems even impossible to *test* if they do
- Approach: keep the default algorithm $\mathcal{A}$ to establish searcher reward



Builder's algorithm

- Modeled after practical builder's algorithm (e.g. flashbouts/rbuilder)
  - Inputs: bundles
  - Outputs: block, bid

- Additional requirements: Can't insert new transactions
  - Consistent with our view: building = ordering bundles
  - This prevents integration and frontrunning
  - Easy to verify: just compare inputs and outputs

Our Mechanism

Input: set of bundles $M = (b_i)$, and submitted algorithms $B_1, \cdots, B_n$

Fixed: default algo $\mathcal{A}$

1. Run $\mathcal{A}$ on M to generate a default block o^* and $\{o_{-i}\}_{i \in M}$.
   - Let $\beta_0 = \sum_{j \in M} b_j(o^*)$, the block value built by $\mathcal{A}$
2. For each bundle $i \in M$, compute the refund $r_i = \sum_{j \in M} b_j(o^*) - \sum_{j \neq i} b_j(o_{-i})$
3. Run all builder-submitted algorithms on M to produce n blocks $\{B_1, \cdots, B_n\}$ w/ bids $\{\beta_1, \cdots, \beta_n\}$.
   - Let B^* be the block with the highest bid $\beta^* = \max_{j \in [n]} \{\beta_j\}$.
   - Let be the second-highest bid β' among all n+1 blocks
4. Determine the final output.

Our Mechanism (cont'd)

4. Determine the final output.

Case (4a): $\beta_0 \geq \beta^*$ (default wins)

- Final block: o^*
- Builder: pay nothing.
- Searcher:
  - Pay $b_i(o^*)$
  - Refund: r_i for $i \in M$
- Proposer revenue: $\beta_0 - \sum_{i \in M} r_i$.
- (Identical to attempt #2)

Case (4b): $\beta_0 < \beta^*$ (a builder wins)

- Final block: B^*
- Builder: winner receives $\sum_i b_i(B^*) - \beta'$
- Searcher:
  - Pay $b_i(B^*)$
  - Refund: r_i for $i \in M$
- Proposer revenue: $\beta' - \sum_{i \in M} r_i$.
- Basically: builder pays 2nd price. Searcher refund is determined by the default algorithm.

Refund Analysis

Claim 1 (Non-negativity). The refund to each bundle $i \in M$ is non-negative.

Proof sketch. By the definition of $o^* = \arg\max_{o \in \Omega_{\mathcal{A}}} \sum_{i \in M} b_i(o)$, we have

$$\sum_{j \in M} b_j(o^*) \geq \sum_{j \in M} b_j(o_{-i}) \geq \sum_{j \neq i} b_j(o_{-i}).$$

➔ Individual rationality: Truthful searchers receive non-negative utility.

Theorem 3 (Budget balance). The total refunds to searchers and builders do not exceed the total payment received by our mechanism.

Proof sketch. By the definition of $o_{-i} = \arg\max_{o \in \Omega_{\mathcal{A}}} \sum_{j \neq i} b_j(o)$, $\sum_{j \neq i} b_j(o_{-i}) \geq \sum_{j \neq i} b_j(o^*)$.

$$\rightarrow r_i \leq b_i(o^*) \rightarrow \sum_{i \in M} r_i \leq \sum_{i \in M} b_i(o^*) = \beta_0.$$

Incentive Analysis

(Assumption: searchers and builders operate independently, i.e., without integration.)

Theorem 4 (DSIC for builders). Our mechanism is DSIC for builders.

Proof sketch. For builders, our mechanism = second-price auction with reserve β_0.

Theorem 5 (DSIC for searchers in default-dominating scenario). Given a bundle set where the default algorithm outperforms all builder-submitted algorithms for all bid functions, bidding truthfully is a dominant strategy for searchers.

Integration under Boost+

- Integration still possible in a limited form
- E.g., Banana Gun may submit txns as ciphertext under key K. The same key K is embedded in the algorithm from Titan.
- Equivalently, searcher has to two options:
  - *Participate*: submit bundles to Boost+, available to all algorithms.
  - *Integrate:* privately send bundles to their integrated builders.

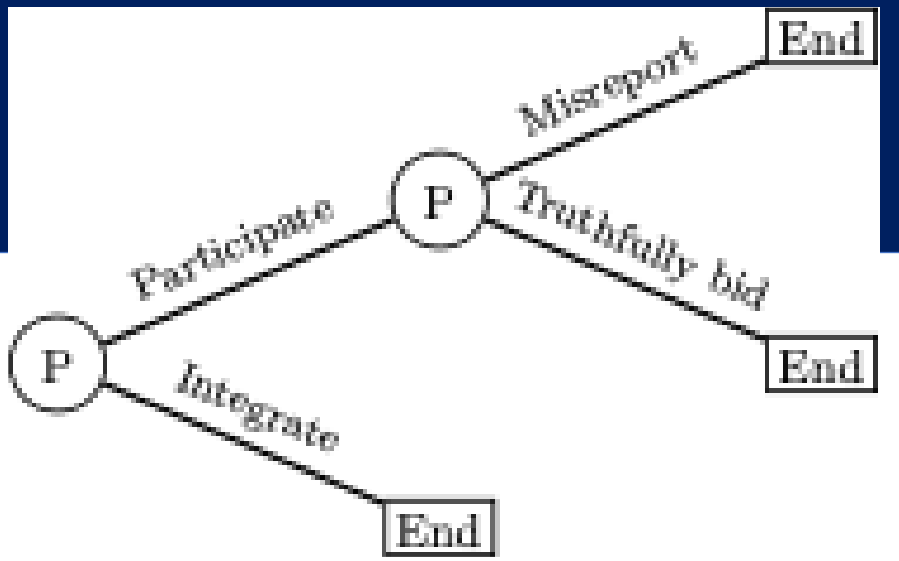
- Key observation: Searchers can't, for example, share bundles to integrated builders + default algorithm, but not others.

Conflict-free Bundles

- A bundle is called conflict-free if its execution neither affects nor depends on that of any other bundle in the set.
- Can be identified by the graph analysis we talked about.

- A fairly strong requirement!
  - Example: Banana Gun's bundles targeting niche meme-coin liquidity pool.

Integration Analysis (cont'd)



- Consider a conflict-free bundle i and a builder j.
- Two-step decision for i:
  1. Choose whether to integrate or participate;
  2. Conditional on participation, truthfully bid or misreport.

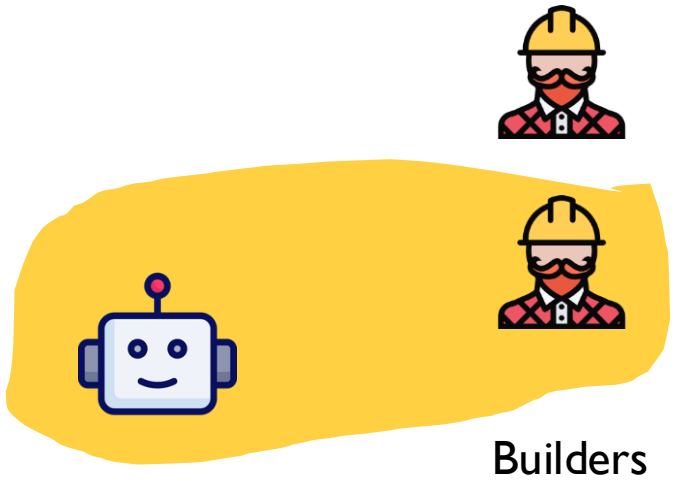
Theorem 7. *Participating* + *bidding truthfully* weakly dominates all alternative strategies with respect to the joint utility of (i, j).

Intuition.

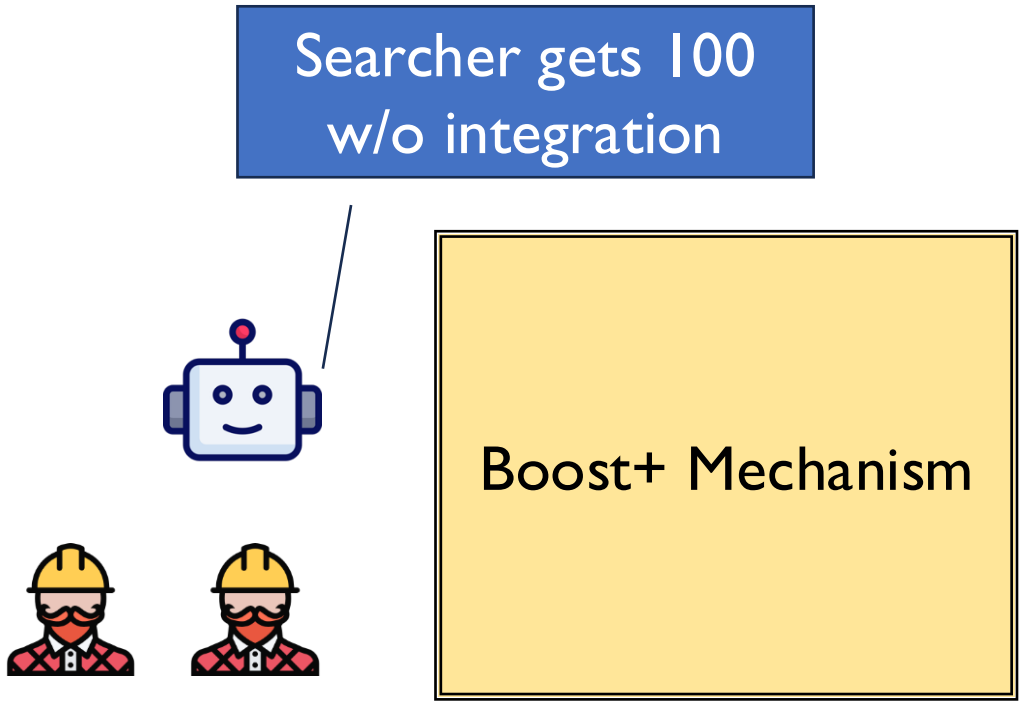
- Under truthful participation, conflict-free bundles always receives full refund.
- Integration can only reduce the chance of inclusion or cause the integrated builder to win while paying more than the true gain, thus lowering their joint utility.

Example: incentive for integration

Consider a searcher with a conflict-free bundle of 100 ETH.



In MEV-Boost, integration increases the utility by 100.

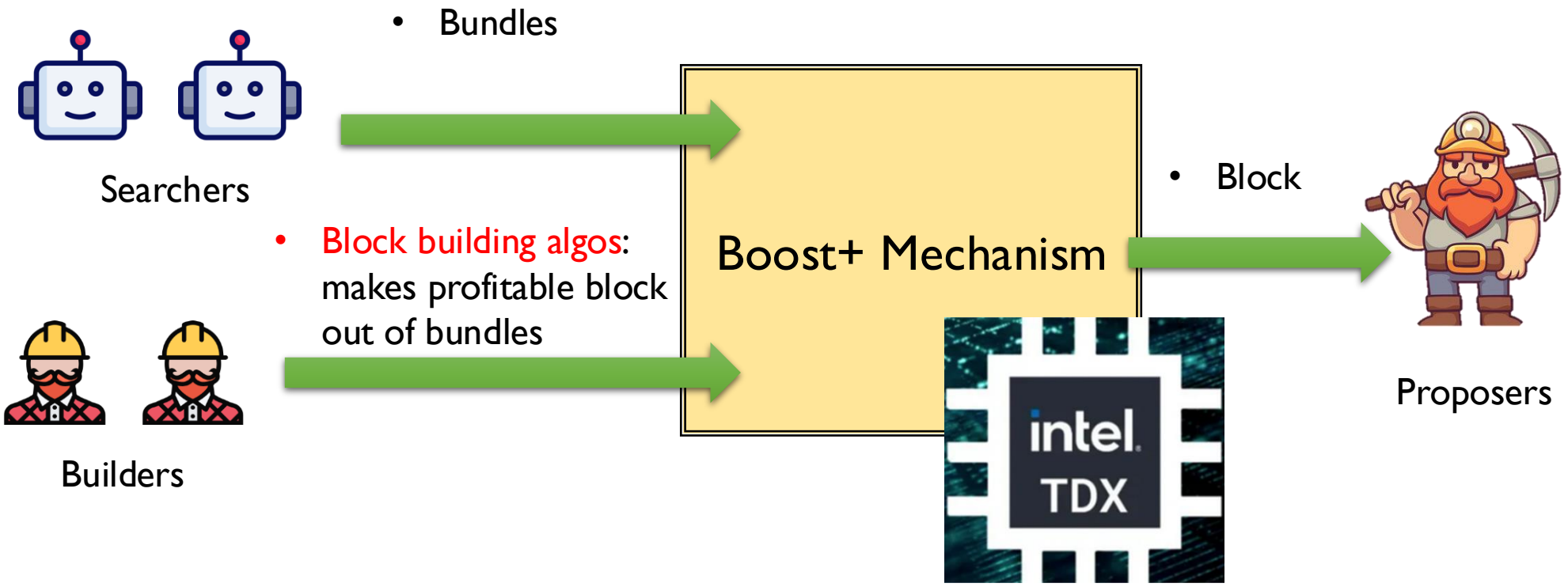


In Boost+:

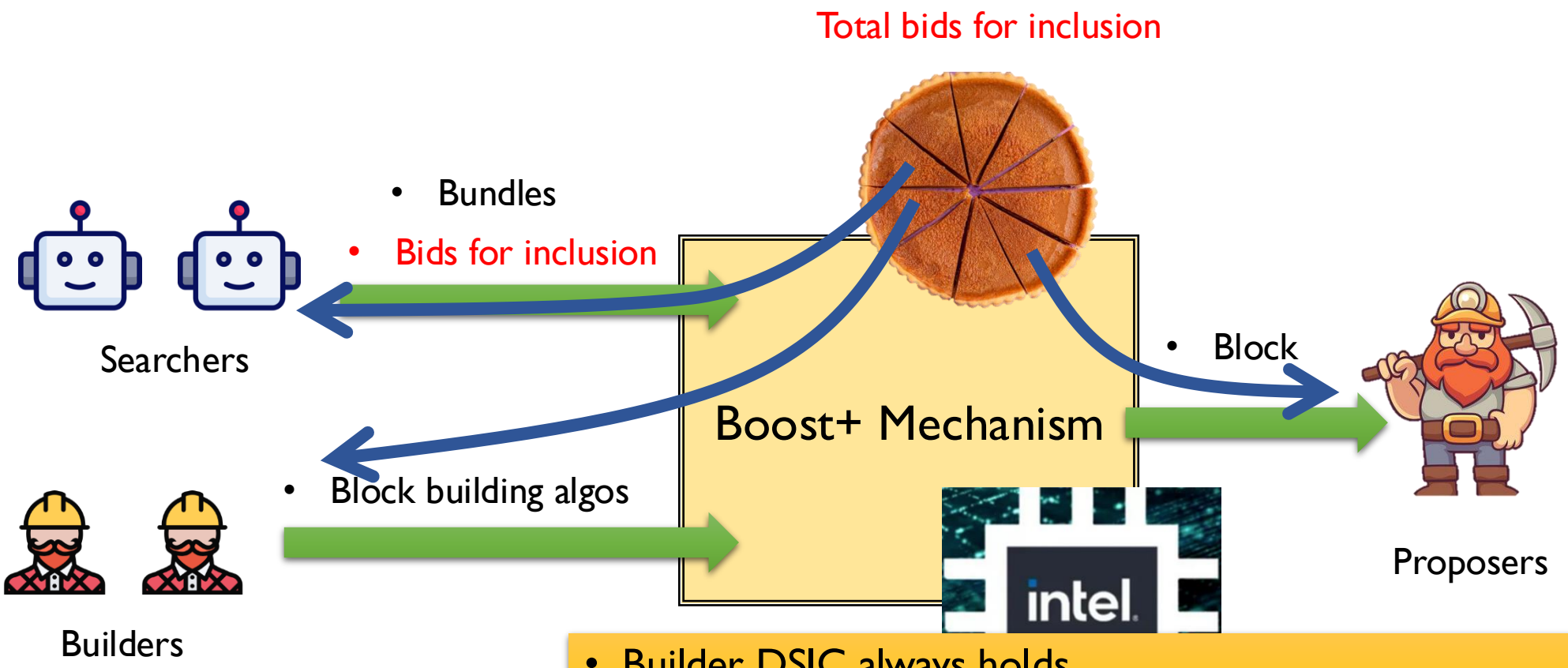
- Don't integrate: utility = 100
- Integrate: utility <= 100

Boost+ recap

- For verifiability, the whole process run in a TEE (Trusted Exec. Environment)



Boost+ recap



- Builder DSIC always holds
- Searcher DISC always holds for *conflict-free* bundles
- General searcher DISC holds when the default algo wins

Evaluation of default algorithm

- Things are nice if the default algorithm wins frequently.
- How well do our algorithm do?
- Implemented in ~900 lines of Rust code.
- Evaluation
  - We "went back in time" and run default algo along with real-world algos
  - Inputs include mempool transactions and private bundles from Flashbots
  - record the runtime of each algorithm and the block value of the built block.
- The default algorithm is the best in ~53% of the blocks.
  - Pretty interesting, given that the default algorithm "doesn't look at the bid"

Evaluation of default algorithm

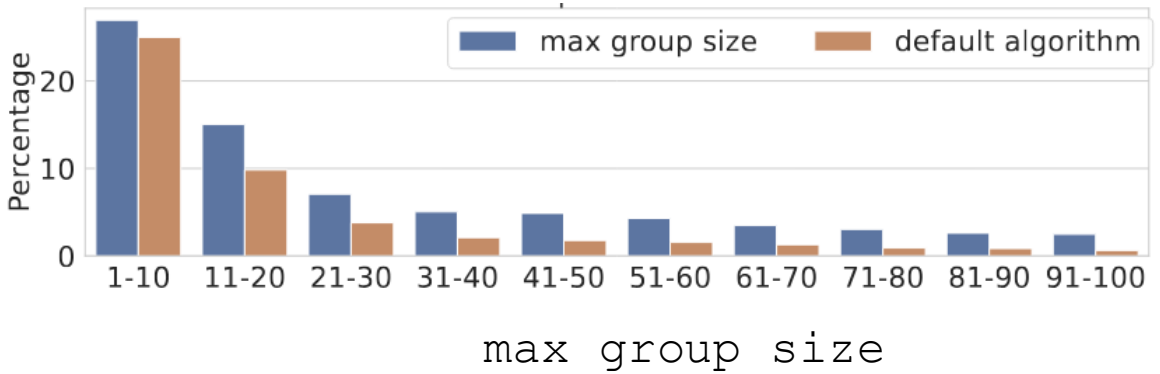


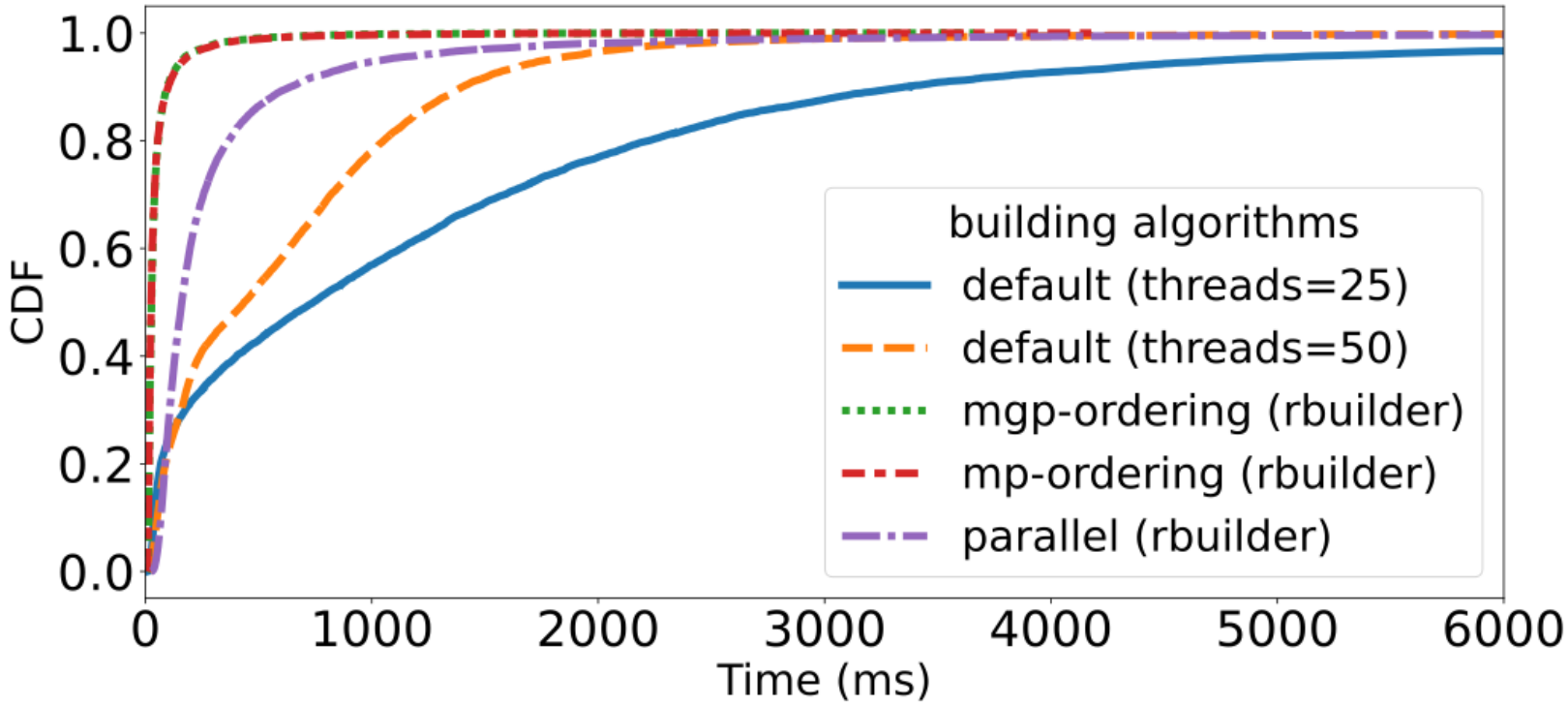
Figure 6: Distribution of the maximum group size in each block, and the fraction of all blocks where the default algorithm is the best in the block with that maximum group size.

Break down based on group size:

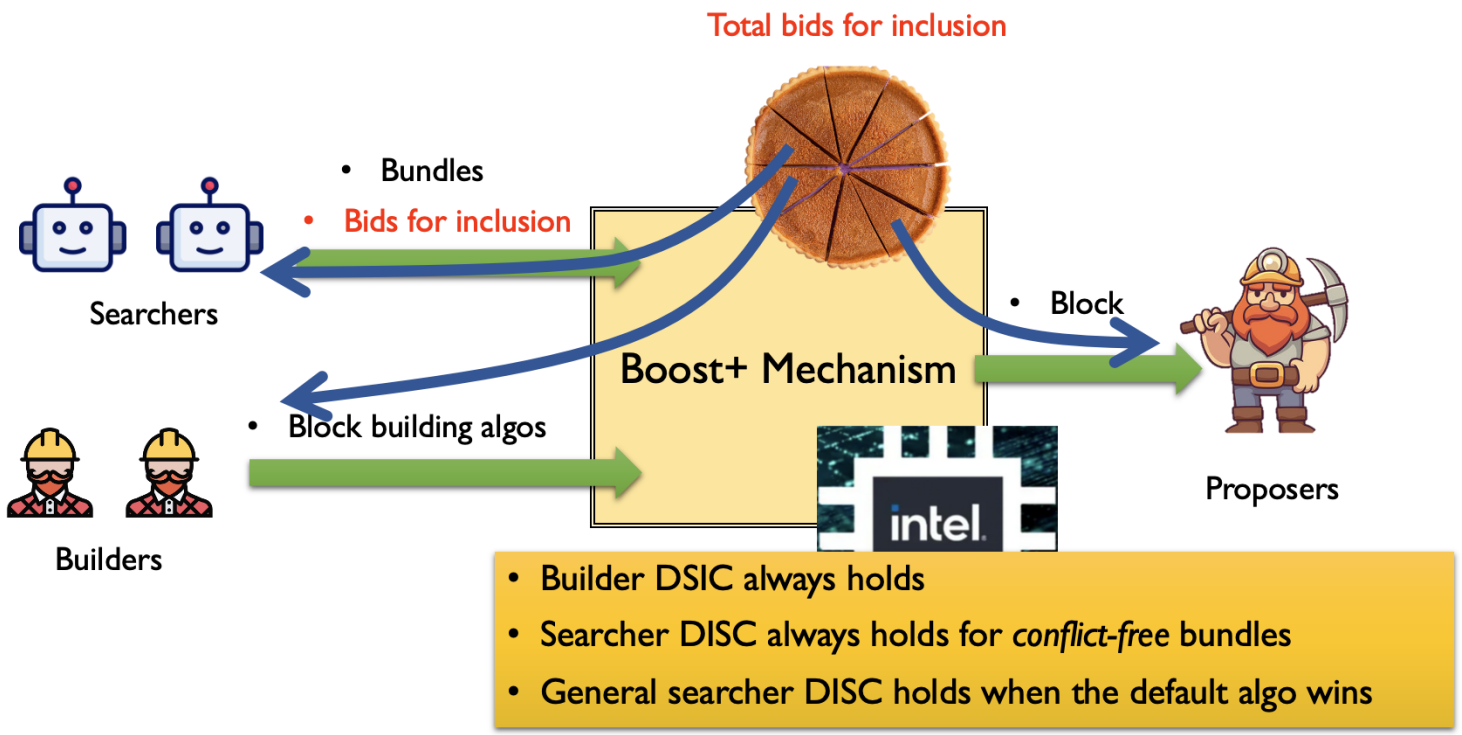
- When the max group size in the block is less than 10, the default algorithm can achieve the optimal results.
- Its winning rate declines as the maximum group size increases.

Runtime Performance

- With parallelism, the runtime of the default algorithm is comparable to that of the STOA builder algorithm (rbuilder).

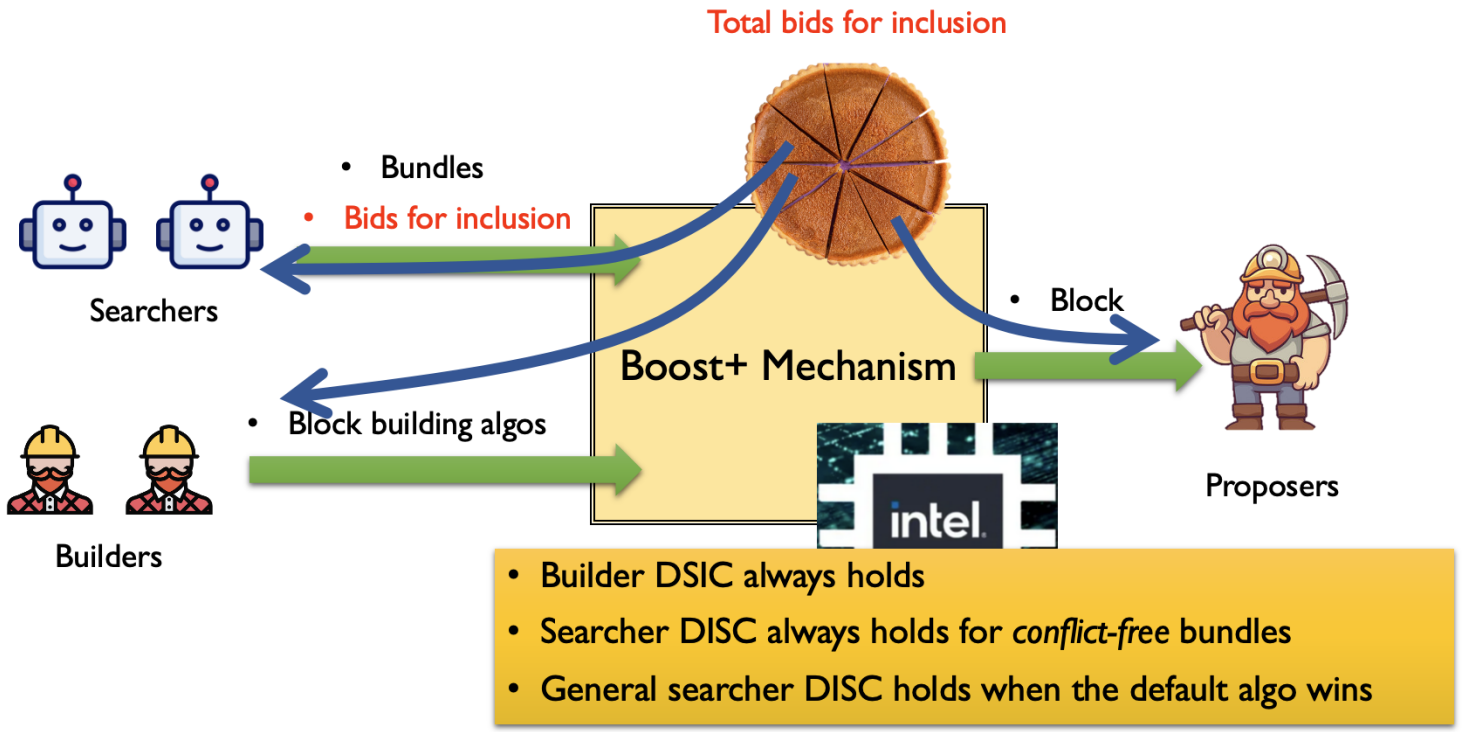


Summary: re-thinking block building



- Boost+ is a block building pipeline (alternative to MEV Boost)
- Builds on a default algorithm that "doesn't look at bids"
- Rewards searchers and builders for marginal contribution to the pie
- Whole thing runs in TEEs for verifiability and confidentiality.

Summary: re-thinking block building



- Future works
  - Achieve searcher-DISC for non-conflict-free bundles or show impossibility
  - Searchers cannot predict which algorithm will win. Can we leverage that to achieve nice properties?
  - How to improve the default building algorithm?

Backup slides

Model (Searchers)

- M: set of all bundles submitted by searchers.
- Ω: set of all feasible blocks that can be constructed from M.
- Each bundle $i \in M$ has a private valuation $v_i(\omega)$ for each block $\omega \in \Omega$.
- Each bundle $i \in M$ specifies a bid function $b_i(s_i(\omega))$, abbreviated as $b_i(\omega)$.
  - $s_i(\omega)$: execution state of bundle i in block ω.
  - The bid can be constant, e.g., a fixed tip for inclusion.
  - Or state-dependent, e.g., proportional to realized revenue (measured by the balance change).

Model (Builders)

- N: set of n block-building algorithms submitted by builders.
- All algorithms operate on the same bundle set M.
- Algorithm $j \in N$ outputs a candidate block B_j and a bid β_j.

Block-building Mechanism

- Input: a set M of bundles, their bidding functions $\mathbf{b}$, and n builder-submitted algorithms.
- Output: a block and the net payment of each participant.
  - A player may pay or get paid, settled by transactions in the output block.
  - Use a refund-based payment design: first charge searchers/builders according to their signed bids and then refund some amount after the outcome is determined.

Utility Functions

Given a mechanism in which block $\omega \in \Omega$ is selected,

- Utility of searcher $i \in M$ is

$$v_i(\omega) - b_i(\omega) + r_i$$

- Utility of builder $j \in N$ is

$$\begin{cases} \sum_{i \in M} b_i(\omega) - \beta_j + r_j, & if\ B_j = \omega; \\ 0, & otherwise. \end{cases}$$

Centralization