Workshop

Agent-based simulation of Execution Tickets

DC7 SEA

15/11/24
2:30 pm - 4:00 pm

Pascal
@pascalstichler

ephema

Agenda

Theory Part (2:30 pm - 3:00 pm)

1. Recap: What are Execution Tickets again?
2. Research Insights: Open mechanism design choices?
3. Simulation Insights: What have we found out so far?

Workshop Part (3:00 pm - 3:55 pm)

4. Introduction to the Simulation Setup
5. Collaboratively setting up the simulation
6. Run your own simulations

```
model = Model(
    params=sys_params,
    initial_state=initial_state,
    state_update_blocks=[
        {
            # 1. Update market meta data
            'policies': {
                'update_market': p_update_market
            },
            'variables': {
                'slot': v_update_slot,
                'epoch': v_update_epoch,
                'tickets': v_update_tickets,
                'current_ticket_id': v_update_current_ticket_id,
                'MEV_per_slot': v_update_MEV_per_slot,
                'Volatility_per_slot': v_update_Volatility_per_slot
            }
        },
        {
            # 2. Purchase Tickets
            'policies': {
                'purchase_tickets': p_purchase_tickets
            },
            'variables': {
                'tickets': v_update_tickets_after_purchase,
                'ticket_holder': v_update_ticket_holder_after_purchase,
                'ticket_price': v_update_ticket_price,
                'total_MEV_captured': v_update_total_MEV_captured
            }
        },
        {
            # 3. Secondary Market
            'policies': {
                'secondary_market': p_secondary_market,
            },
            'variables': {
                'tickets': v_update_tickets_sm,
                'ticket_holder': v_update_ticket_holder_sm
                # More?
            }

        },
        {
            # 4. Redeem tickets
            'policies': {
                'redeem_tickets': p_redeem_tickets
            },
            'variables': {
                'tickets': v_update_tickets_after_redemption,
                'ticket_holder': v_update_ticket_holders_after_redemption
            }
        }
    ]
)
```

$1.7bn

MEV-Boost payments since The Merge (Sept 15, 2022)[1]

That's ~ $2mn per day 🤯

[1] 573,889 ETH as of Nov 11, 2024 based on https://mevboost.pics

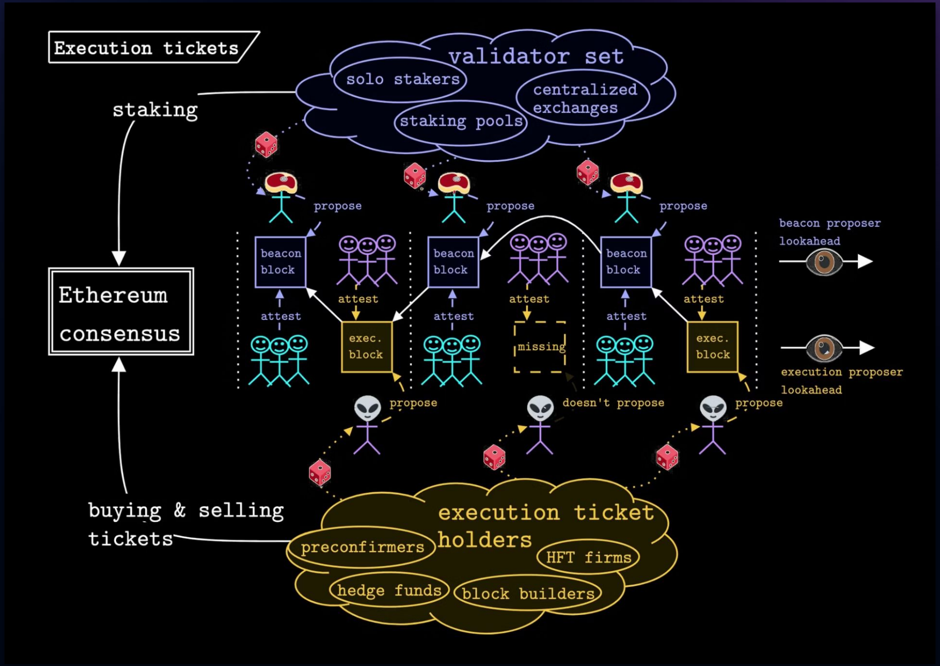
Execution Tickets as a new idea to allocate block space

- Introduced by Justin Drake in late 2023 as **Attester-Proposer Separation (APS)**
- Core idea is to **separate the beacon block from the execution block** and sell the right to create execution blocks, however randomize it
- Allows to **capture MEV at the protocol level**
- Randomization is there to prevent multi-block MEV and liveness issues

Recap: How execution tickets are processed

The process in more detail:[2]

1. During the beacon round, the selected beacon proposer **proposes a beacon block**
2. Proposer proposes the beacon block that contains some form of **inclusion list**
3. Beacon attesters vote on the validity and timeliness of the beacon block
4. Execution round: a randomly selected ET can **propose an execution block**
5. The execution attesters vote on the timeliness and validity of the execution block



[2] Neuder, M. (2023, December). Summary of Execution Tickets. Retrieved from https://ethresear.ch/t/execution-tickets/17944

Several guiding research questions come up with the idea of Execution Tickets

1. When laying out the design space, what are **objectives** to focus on?

2. How to **measure** them?

3. What are **possible configurations**?

4. What will be the **outcome** with regards to the measurement params?

Overview of optimization objectives

	Objectives	Measurement Metrics
Optimization Parameters	1. Decentralization 2. MEV Capture 3. BPIC (Block Producer Incentive Compatible)	1. Market Share, Nakamoto-coefficient & Herfindahl - Hirschman Index (HHI) 2. MEV-Share Protocol
Pricing Behavior	1. Price Predictability 2. Price Smoothness 3. Price Accuracy	1. Garman & Klass Measure 2. Var(Δp) 3. MEV-Share Protocol

Overview of possible mechanism designs

- Based on the introduced requirements we have **outlined 6 promising mechanism designs** to test in a simulation
- Therefore, the **seven most important specification parameters** have been defined
- Further, based on literature review and expert input the mechanism designs have been constructed

	Simple FPA auction	**JIT second price slot auction**	**Flexible 1559-style**	**Fixed SPA**	**Flexible, refundable AMM**	**Fixed, resellable FPA**
Amount of tickets	Fixed	Fixed	Flexible	Fixed	Flexible	Fixed
Expiring tickets	Yes	Yes	No	No	No	No
Refundability	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated)	No (unallocated & allocated)
Resalability	No (unallocated & allocated)	Yes (allocated)	Yes (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated & allocated)
Enhanced Lookahead	No	Reduced	Yes for Execution Validators	Yes for Execution Validators	No	No
Pricing Mechanisms	FPA	SPA	1559-style	SPA	AMM	FPA
Target Amount	32	1	undefined	1024	undefined	1024

Results of simulation runs

Based on over 300 runs of the simulation with different configurations we can conclude:

- None of the configurations scores particularly **well on decentralization**
- Secondary **markets with JIT auctions reduce centralization** by enabling specialized ticket holders to acquire tickets JIT
- With regards to MEV capture, **auction formats and AMM-style pricing scores well**, while EIP-1559 style pricing lags in the price adaption
- Auction formats with longer lookahead periods provide high price predictability and smoothness

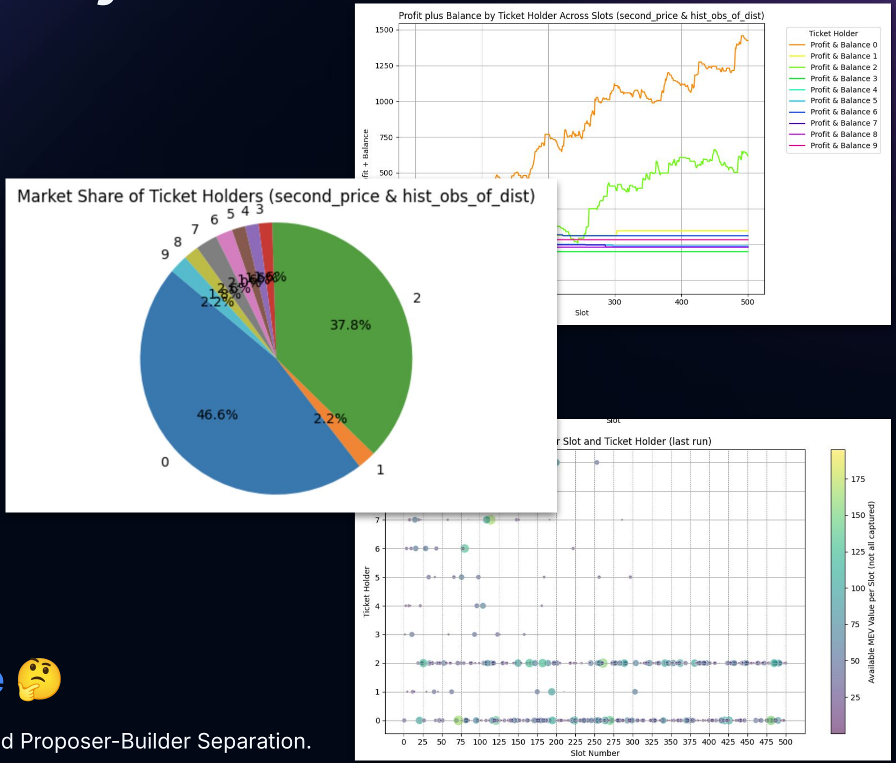
Overview of Simulation Results[3]

Objective	Metric	Simple FPA auction	JIT SPA slot auction	Flexible 1559-style	Fixed SPA	Flexible, refundable AMM	Fixed, resellable FPA
Decentral-ization	Market share	93.2%	52.2%	76.2%	69.8%	89.0%	48.9%
	Nakamoto-coefficient	1	1	1	1	1	1.9[4]
	Herfindahl-Hirschman Index	8718.0	4741.7	6062.7	5950.9	7991.0	3458.1
MEV Capture	MEV-Share Protocol	79.8% / 89.1%	76.6% / 77.5%	40.2%	69.8% / 81.7%	75.8 % / 84.0 %	77.2 %
Price Predictability	GK Measure	0.008	3.8	1.095	0.006	0.010	0.042
Price Smoothness	V(Δp)	0.026	1398.6	91602.8	0.012	0.057	0.521
Price Accuracy	MEV-Share Protocol	79.8%	77.5%	40.2%	69.8%	75.8%	77.2%

Objective 1: Decentralization

The currently (bad) news: With the current configurations there is a risk of high execution layer centralization forces

- ❏ In all current configurations we observe high centralization forces
- ❏ It is driven by bidders working with expected values compared to actual value in-flight pricing
- ❏ Enabling a secondary market reduces the centralization
- ❏ Finding aligns with existing literature (e.g. (Bahrani et al., 2024)[3])



→ **Objective 1** (execution layer decentralization) remains a **design challenge** 🤔

[3] Bahrani, M., Garimidi, P., & Roughgarden, T. (2024). Centralization in Block Building and Proposer-Builder Separation. http://arxiv.org/abs/2401.12120

Objective 2: Capturing MEV

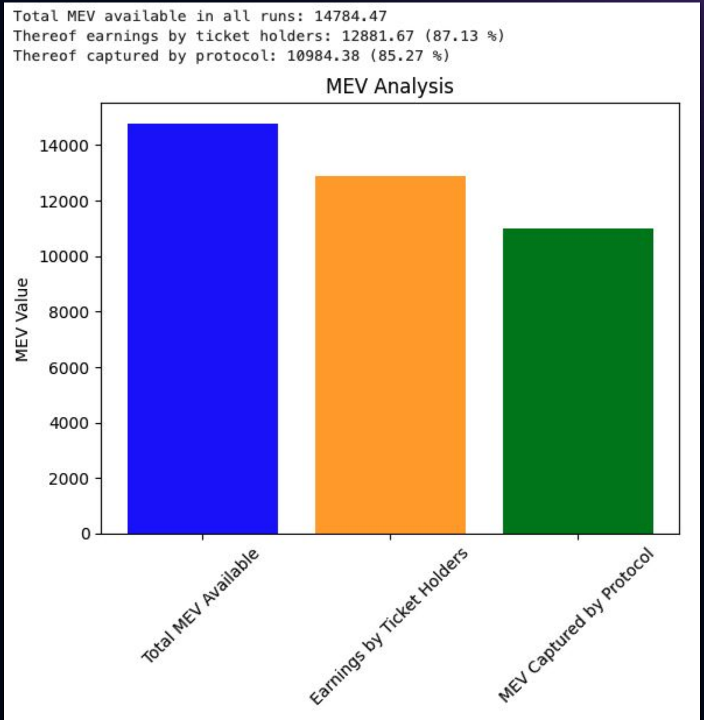
The good news: With the right pricing mechanism a high rate of captured MEV seems possible

- We observe generally a high MEV capture of 70 - 90 % in different configurations

- Overall auction based formats and AMM-style pricing performs well

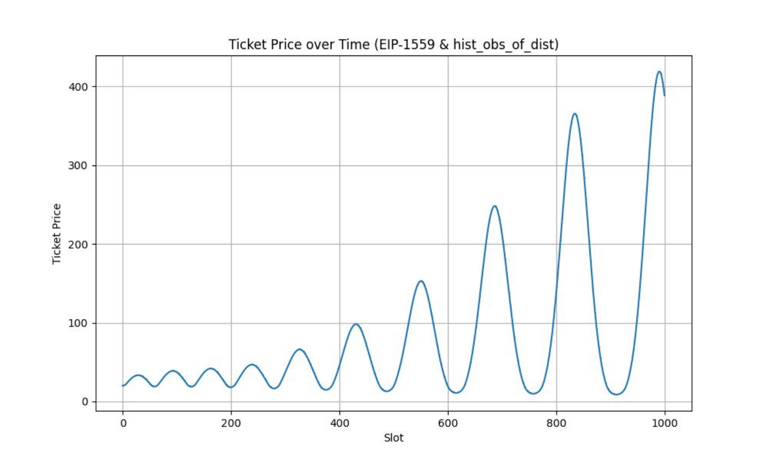
- A notable exception is 1559-style pricing that has a lower MEV capture (as outlined above)

→ **Objective 2** (MEV capture) seems to be on **a good track** ✅

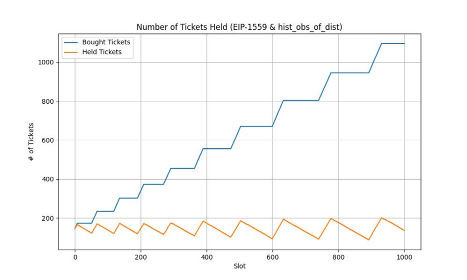


Deep dive on EIP-1559 style pricing

With a high number of newly maximal issued tickets in a EIP-1559 style pricing mechanism and a low adjustment factor prices start self-reinforcing oscillating



→ If EIP-1559 style pricing is chosen, the **adjustment factor needs to be higher than 12.5 %** and / or **adapting after every ticket sale** (not only slot)



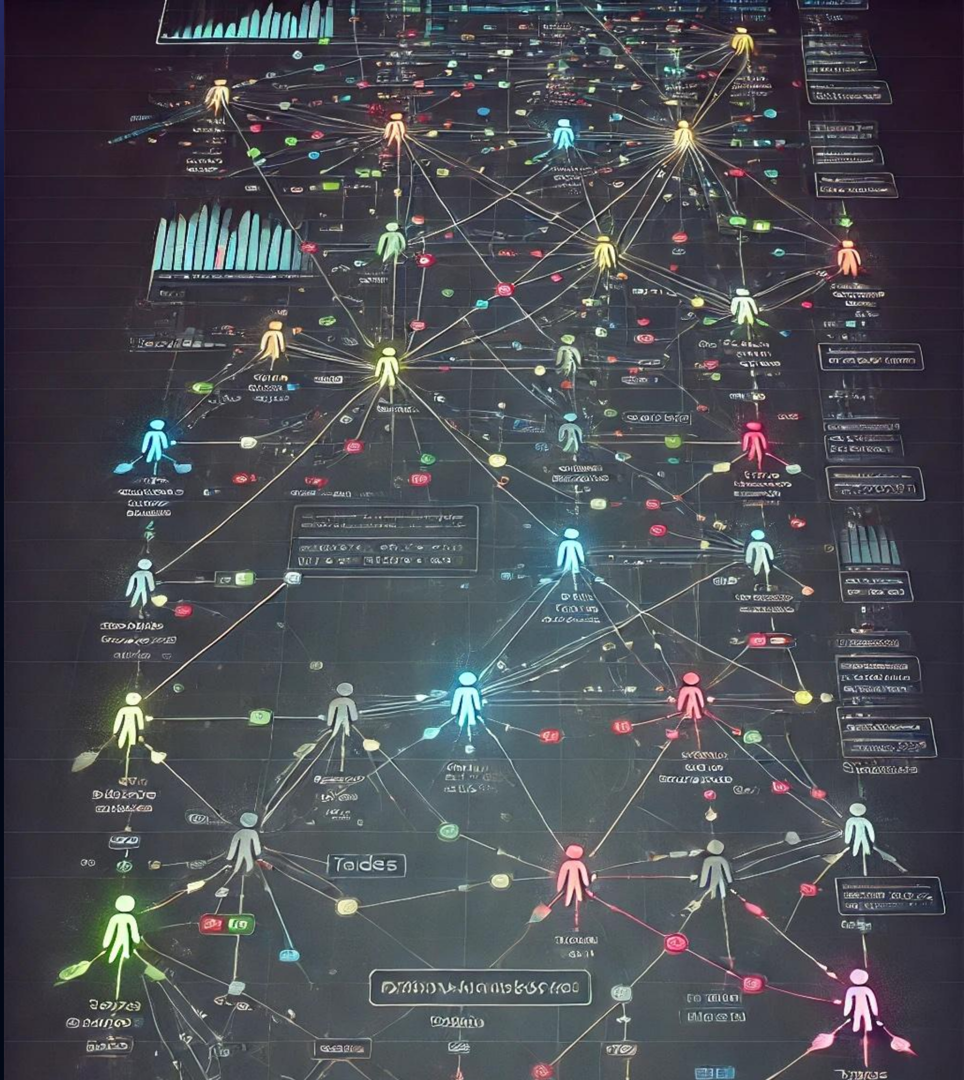
Findings on ticket attributes

- **Fixed vs. Variable Ticket Amounts:** The choice depends on the pricing mechanism—fixed amounts suit auctions, while variable amounts are better for EIP-1559 and AMM-style pricing
- **Expiring Tickets:** Non-expiring tickets are preferable because expiring tickets reduce MEV capture and complicate pricing due to the risk of expiry
- **Refundability:** Tickets should be non-refundable to avoid added complexity, as refundability doesn't significantly impact market dynamics
- **Secondary Market:** Allowing a secondary market enhances decentralization and MEV capture by enabling specialized holders to acquire tickets when they can maximize value; it's advisable to embrace this despite potential challenges

Workshop part

Run your own simulations

Before break?



Workshop idea: Design your own configuration and run your own simulation

1. You can get together in **groups** (optional)
2. First step: **Download and run** the current version of the simulation (next slides details)
3. Second step: Based on literature and your personal experience **design a configuration** for execution tickets that you want to test
4. Third step: **Run your own** version of the **simulation**
5. Last step: Share your results in the Notion documentation (optional)

	Simple FPA auction	**JIT second price slot auction**	**Flexible 1559-style**	**Fixed SPA**	**Flexible, refundable AMM**	**Fixed, resellable FPA**	**?**
Amount of tickets	Fixed	Fixed	Flexible	Fixed	Flexible	Fixed	
Expiring tickets	Yes	Yes	No	No	No	No	
Refundability	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated)	No (unallocated & allocated)	
Resalability	No (unallocated & allocated)	Yes (allocated)	Yes (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated & allocated)	
Enhanced Lookahead	No	Reduced	Yes for Execution Validators	Yes for Execution Validators	No	No	
Pricing Mechanisms	FPA	SPA	1559-style	SPA	AMM	FPA	
Target Amount	32	1	undefined	1024	undefined	1024	

Simulation Setup (1/3)

How the simulation works

Simulation process (using radCAD)

1. Update market meta data
2. Purchase Ticket round
3. Run secondary market (if enabled)
4. Redeem Tickets

File Structure

- ❏ Main File: *'Execution Ticket Simulation.ipynb'*
- ❏ Support Classes: *models.py, purchase_functions.py, utils.py & calc_statistics.py*
- ❏ Results Folder

Agents (Ticket Holders)

- ❏ Divided into 3 groups: **top, medium & tail builders** [4]
- ❏ Randomly different attributes on: MEV capturing abilities, aggressiveness, initial funds, discount factor etc.

```
model = Model(
    params=sys_params,
    initial_state=initial_state,
    state_update_blocks=[
        {
            # 1. Update market meta data
            'policies': {
                'update_market': p_update_market
            },
            'variables': {
                'slot': v_update_slot,
                'epoch': v_update_epoch,
                'tickets': v_update_tickets,
                'current_ticket_id': v_update_current_ticket_id,
                'MEV_per_slot': v_update_MEV_per_slot,
                'Volatility_per_slot': v_update_Volatility_per_slot
            }
        },
        {
            # 2. Purchase Tickets
            'policies': {
                'purchase_tickets': p_purchase_tickets
            },
            'variables': {
                'tickets': v_update_tickets_after_purchase,
                'ticket_holder': v_update_ticket_holder_after_purchase,
                'ticket_price': v_update_ticket_price,
                'total_MEV_captured': v_update_total_MEV_captured
            }
        },
        {
            # 3. Secondary Market
            'policies': {
                'secondary_market': p_secondary_market,
            },
            'variables': {
                'tickets': v_update_tickets_sm,
                'ticket_holder': v_update_ticket_holder_sm
                # More?
            }

        },
        {
            # 4. Redeem tickets
            'policies': {
                'redeem_tickets': p_redeem_tickets
            },
            'variables': {
                'tickets': v_update_tickets_after_redemption,
                'ticket_holder': v_update_ticket_holders_after_redemption
            }
        }
    ]
)
```

[4] Yang, S., Nayak, K., & Zhang, F. (2024). Decentralization of Ethereum's Builder Market. arXiv preprint arXiv:2405.01329.

Parameters to tweak and adjust

- To adjust the simulation mainly change the parameters in cell 3 of the Notebook:
  - TIMESTEPS: # of slots in the simulation
  - RUNS: # of times the simulation is run
  - `'Sys_params'` : configuration of params

```
TIMESTEPS = 100
RUNS = 1
#  Define System Parameters
sys_params = {
   'selling_mechanism': 'first_price',
   'max_tickets': 100,
   'initial_ticket_price': 20,
   'MEV_scale': 30,
   'slots_per_epoch': 32,
   'number_of_ticket_holders': 10,
   'secondary_market': True,
   'price_vola': {'mean': 0, 'sigma': 0.2},
   'agent_bidding_strategy': 'optimal_heuristic_bidding',
   'EIP-1559_max_tickets': 4,
   'EIP-1559_adjust_factor': 8,
   'AMM_adjust_factor': 25,
   'expiry_period': None,
   'reimbursement_factor': None}
```

- Optional:
  - Adjust properties of ticket holders in `class TicketHolderAgent` `(Support-Classes/models.py)`
  - Adjust the bidding mechanisms of ticket holders in `(Support-Classes/models.py)`

Run your own simulation

1. Sign-up to Google Colab & Copy Notebook via ‘File’ > ‘Save a copy in Drive’
2. Setting up the Simulation:
   a. Download the repo from Github (for support classes): https://github.com/ephema/ET-Simulation
   b. Move repo to Google Drive root folder
3. Run the simulation via ‘Runtime and ‘run all’
4. Design your own configuration and adjust the simulation
5. [Opt] Adjust the configuration of Ticket Holders and their bidding strategies

Link to Notion Docu:

bit.ly/ET-Simulation

Run your own simulation and show us the results

	Simple FPA auction	JIT second price slot auction	Flexible 1559-style	Fixed SPA	Flexible, refundable AMM	Fixed, resellable FPA	?
Amount of tickets	Fixed	Fixed	Flexible	Fixed	Flexible	Fixed	
Expiring tickets	Yes	Yes	No	No	No	No	
Refundability	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated)	No (unallocated & allocated)	
Resalability	No (unallocated & allocated)	Yes (allocated)	Yes (unallocated & allocated)	No (unallocated & allocated)	No (unallocated & allocated)	Yes (unallocated & allocated)	
Enhanced Lookahead	No	Reduced	Yes for Execution Validators	Yes for Execution Validators	No	No	
Pricing Mechanisms	FPA	SPA	1559-style	SPA	AMM	FPA	
Target Amount	32	1	undefined	1024	undefined	1024	

Overview of Simulation Results[17]

Objective	Metric	Simple FPA auction	JIT SPA slot auction	Flexible 1559-style	Fixed SPA	Flexible, refundable AMM	Fixed, resellable FPA	?
Decentral-ization	Market share	93.2%	52.2%	76.2%	69.8%	89.0%	48.9%	
	Nakamoto-coefficient	1	1	1	1	1	1.9	
	Herfindahl-Hirschman Index	8718.0	4741.7	6062.7	5950.9	7991.0	3458.1	
MEV Capture	MEV-Share Protocol	79.8% / 89.1%	76.6% / 77.5%	40.2%	69.8% / 81.7%	75.8 % / 84.0 %	77.2 %	
Price Predictability	GK Measure	0.008	3.8	1.095	0.006	0.010	0.042	
Price Smoothness	V(Δp)	0.026	1398.6	91602.8	0.012	0.057	0.521	
Price Accuracy	MEV-Share Protocol	79.8%	77.5%	40.2%	69.8%	75.8%	77.2%	

Link to Notion Docu:

bit.ly/ET-Simulation

Thank you!

Research project generously supported by

ecosystem support program

X/Twitter: @ephemalabs

Appendix

Preliminary Takeaways - tbd if slide needed

Preliminary Takeaways

- **Objective 1** of decentralization **remains a challenge** and needs to be further investigated
- **Objective 2** of capturing a maximal amount of MEV at protocol level **looks promising**
- **EIP-1559 style pricing cannot** be **directly transferred** to Execution Tickets. Auction based formats as well as AMM-style pricing shows promising results
- Non-expiring and non-refundable tickets are preferable
- Enabling a secondary market fosters decentralization and seems operationally more feasible

Economic evaluation of auction mechanisms

TFM	MMIC?	DSIC?	OCA-proof?
FPA	yes (Cor. 5.3)	no (obvious)	yes (Cor. 5.12)
SPA	no (Ex. 3.5)	almost[28]	almost (Rem. 5.13)
β-burn FPA	yes (Cor. 5.3)	no (obvious)	no (Cor. 5.17)
1559	yes (Cor. 5.3)	usually (Thm. 5.7)	yes (Cor. 5.14)
β-burn 1559	yes (Cor. 5.3)	usually (Rem. 5.9)	no (Cor. 5.18))
tipless	yes (Cor. 5.3)	yes (Thm. 5.5)	usually (Cor. 5.15+Rem. 5.16)

Source: Roughgarden, T. (2021) Transaction Fee Mechanism Design. Retrieved from https://arxiv.org/pdf/2106.01340.pdf.

From an game theory perspective certain properties are important as well

BPIC (Block Producer Incentive Compatible)

A transaction fee mechanism (TFM) is BIPC, if it expects a block producer to publish a block that **maximizes its private valuation plus the net fees earned**.[2]

E.g. SPA is not BIPC

OCA-proofness (Off-Chain Agreement)

A system's resilience against off-chain alternatives **that could pareto-improve over a on-chain outcome**.[3]

E.g. β-burn FPA not OCA-proof

DSCI (Dominant-Strategy Incentive Compatible)

Refers to a situation where if a miner/proposer follows the allocation rule *x*, **each user is best off playing their dominant strategy** regardless of the bids of other users.
E.g. FPA not DSCI

[2] Bahrani M, Garimidi P, Roughgarden T. (2023). Transaction Fee Mechanism Design with Active Block Producers. Retrieved from https://arxiv.org/abs/2307.01686v2
[3] Roughgarden, T. (2021) Transaction Fee Mechanism Design. Retrieved from https://arxiv.org/pdf/2106.01340.pdf.

Multi-block MEV: Historical MEV boost payment distribution