Economic organisation in the Ethereum protocol

Digital Finance Seminar Series — Columbia University

Barnabé Monnot

Robust Incentives Group, Ethereum Foundation

Aim of the talk

Last week, Columbia Cryptoeconomics workshop at Columbia!

If you didn't make it:

1. Why? 😢
2. A chance to get an overview!

If you made it:

1. Some overlap (sorry! New York is too exciting 🗽🚕🎭🎷🍕🍔)
2. **My hope:** A new way of seeing things you already know!

Section 1

Protocol-agent problem

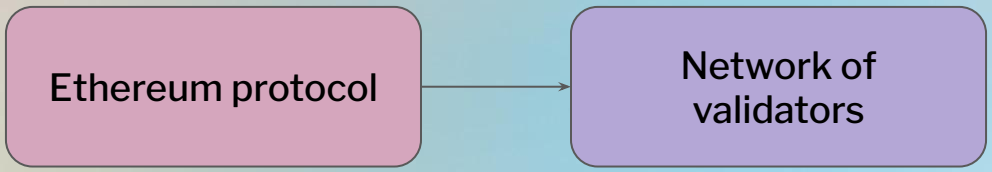
Protocol-agent problem

There is a thing called the **Ethereum protocol**

Validators run the protocol, but do they run it **honestly**?

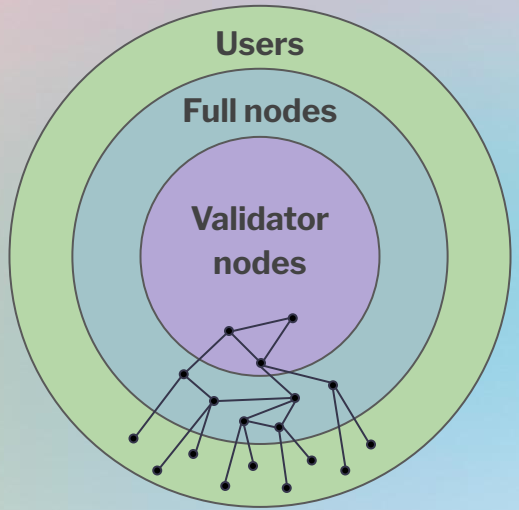
System of **rewards and penalties** ensures compliance, but is it enough?

Today: Understand what the protocol is, how economic organisation follows



Who validates?

Full nodes observe execution of the Ethereum protocol (**“read-only”**)
Validators participate in protocol instantiation (**“read-write”**),
i.e., **consensus formation** and **block construction**

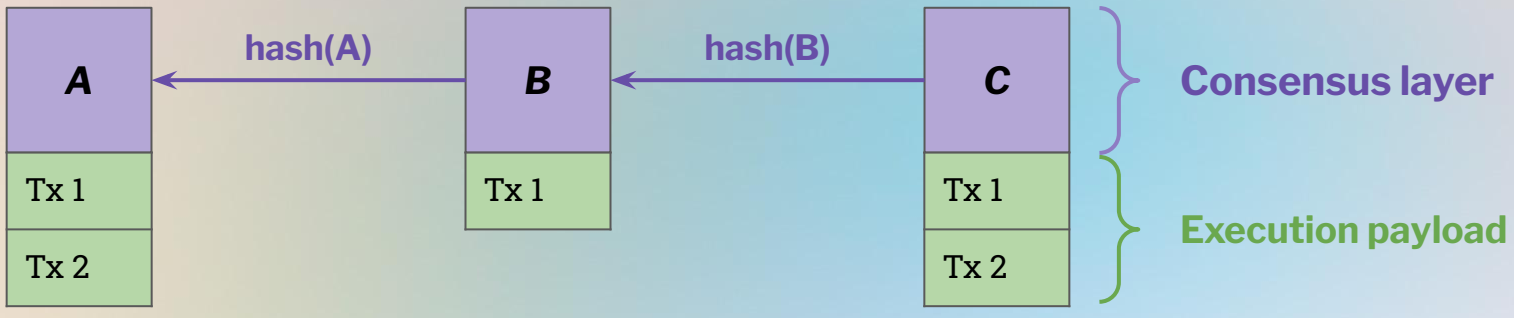


Ethereum protocol instantiates a blockchain

Blockchain = data structure, chain of blocks

Blocks contain:

- **Consensus data** (e.g., hash of previous block)
- **Executable transactions** (executable wrt some execution model)

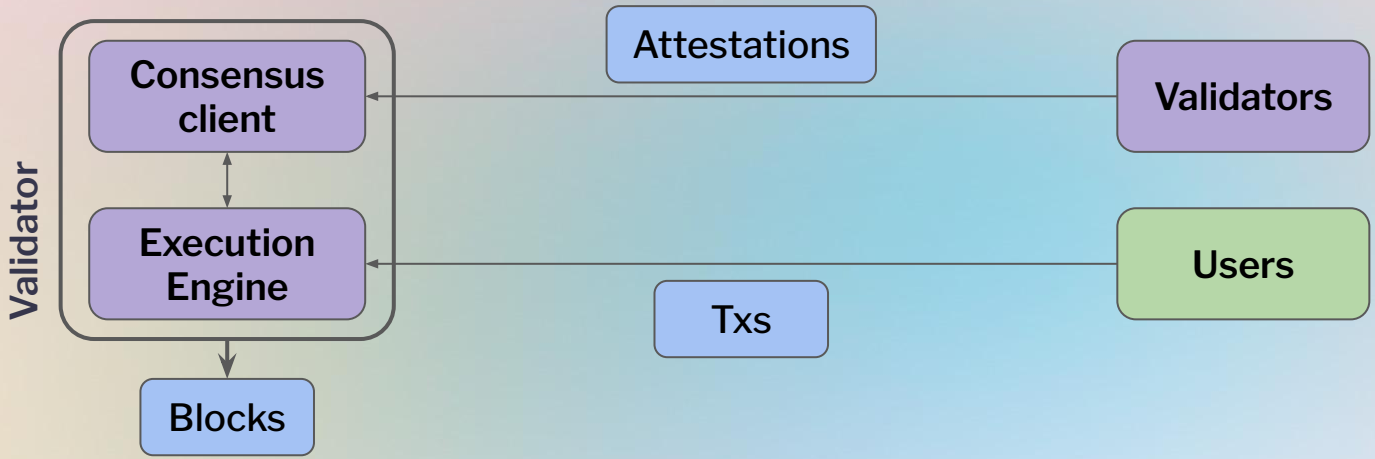


Validators

To become a validator, you **lock up 32 ETH** in the deposit contract

After activation, validator is called upon to perform consensus duties:

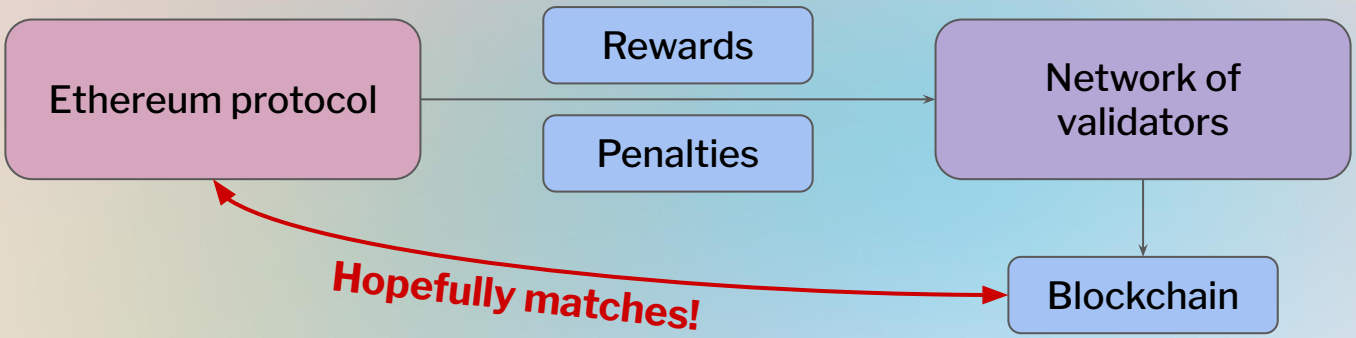
- **Block proposals:** Make a block containing consensus and execution data
- **Attestations:** Provide their view of the consensus, finalise blocks



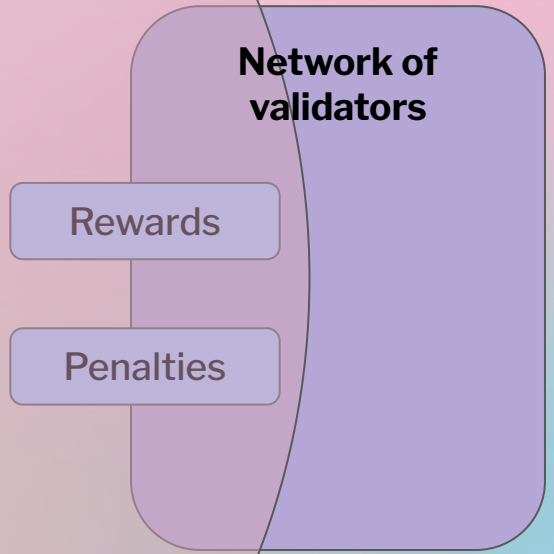
Protocol rewards

Protocol specifies rewards and penalties

- **Rewards**
  - **Block reward** for block proposal/correct voting
  - **Transaction fees** from execution payload
- **Penalties:** Inactivity penalty + Slashing for attributable consensus fault



Ethereum protocol



Section 2

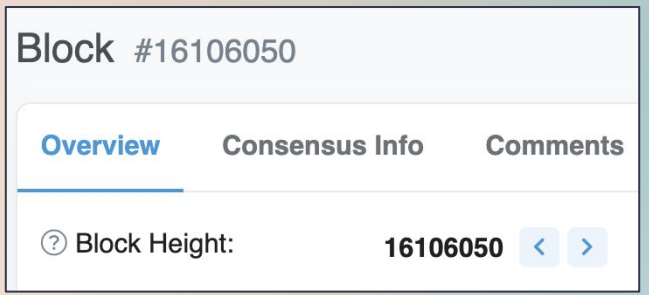
Supplying network resources

Gas

All nodes receive all blocks, execute and verify all transactions.
Transactions consume:

- **Compute** (basic operations)
- **Storage** (short-term memory, state accesses, historical data/function calls)
- **Bandwidth** (raw size of the block/messages in bytes)

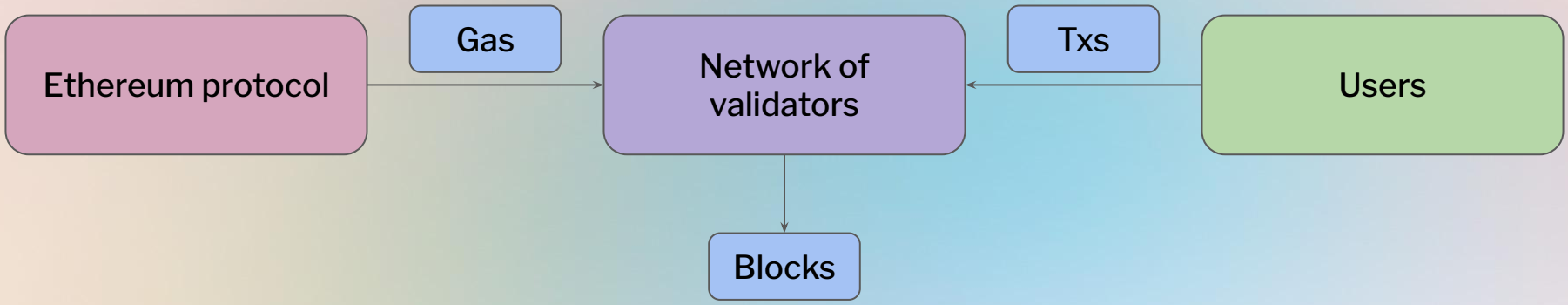
Execution payloads consume gas, a measure of resource use.



Validators as block producers

Protocol lets validators-as-block-producers **consume** resources
Protocol constrains supply to guarantee **low verification costs**

Validators produce **blocks**,
meeting **demand for transactions** with **supply of resources**



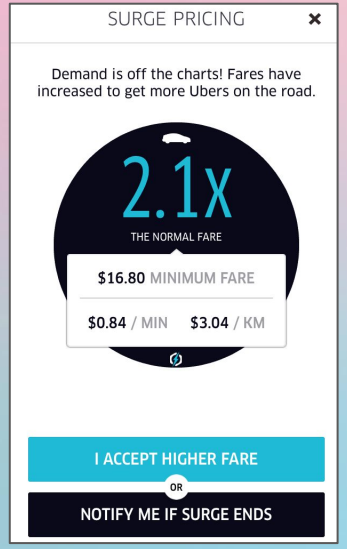
EIP-1559

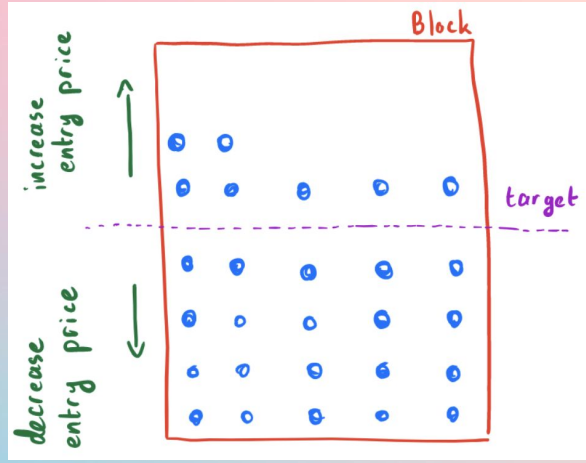
Since August 2021, protocol-mandated reserve price per gas

- **Demand signal:** Blocks target 15 million gas use, block limit 30 million
- **Update rule:** Block filled > 15 million => reserve price increases

Reserve price is a **congestion price**!

See **Roughgarden '21** for reference





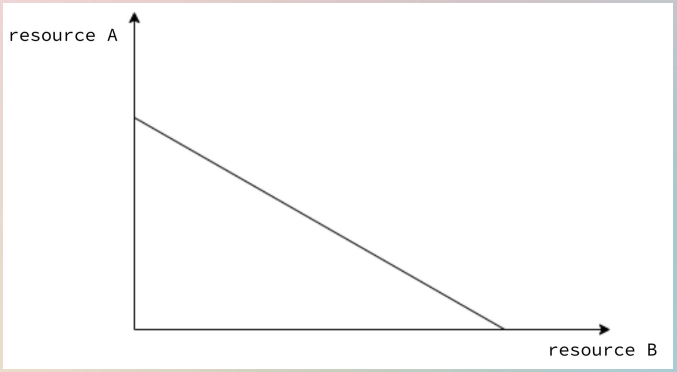
$$b_{n+1} = b_n \cdot (1 + \gamma \cdot \frac{g_n - T}{T})$$

Multi-dimensional EIP-1559

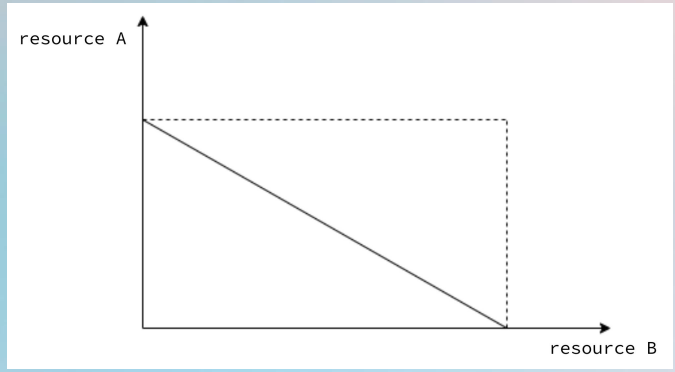
Gas is numéraire with **fixed relative prices** between resources ⇒ **Inefficient!**

> “3 units of execution = 1 bytes kept by all nodes forever…”

Multi-dimensional gas ⇒ **Floating** relative prices!



Fixed relative prices

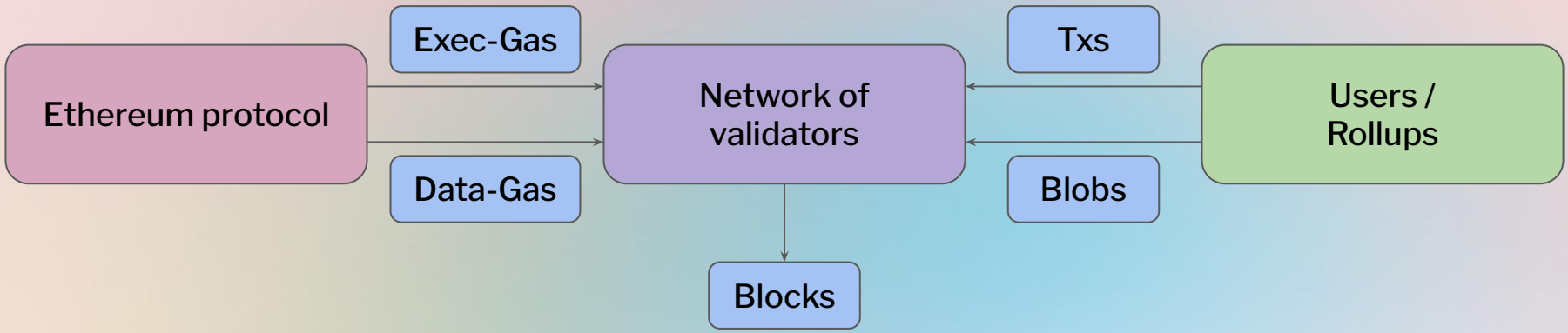


Floating relative prices

h/t @adietrichs

2-dimensional EIP-1559: Data-gas

Start with **two dimensions** with EIP-4844, “exec-gas” and **data availability**
Data availability => consumed by rollups, L2 solutions secured by Ethereum



Summary so far

Ethereum protocol **specifies a set of rules, behaviours** to follow

- Some it can **reward**, some it can **penalise**...
- Ultimately, **validators** are the ones executing the protocol
- **Full nodes** backstop incorrect execution, e.g., **validator capture**

Ethereum protocol also **supplies some amount of resources** over time

- Resources are **costly** to the network!
- All nodes supply these **physical** resources
- Impacts **consensus formation** (e.g., too big nodes => nodes drop out)

Questions or thoughts? 🌱

Section 3

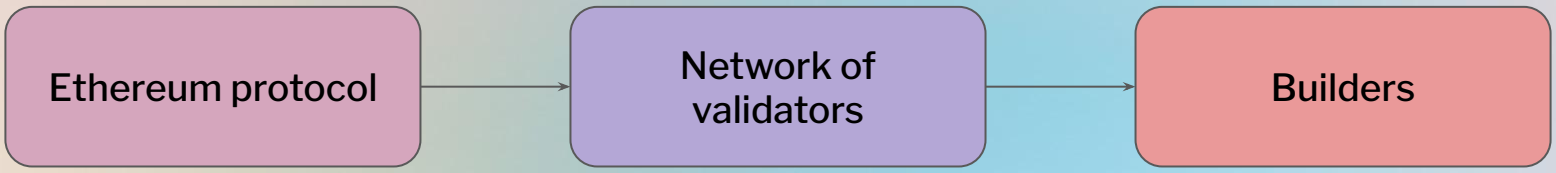
Proposer-agent problem (a.k.a Proposer-Builder Separation)

Proposer-agent problem

Ethereum protocol entrusts its execution in the hands of validators.

Turns out, **validators themselves outsource some of their core functions!**

Let's see why and how.



Execution payload

Block proposer is free to make the **execution payload** however they like

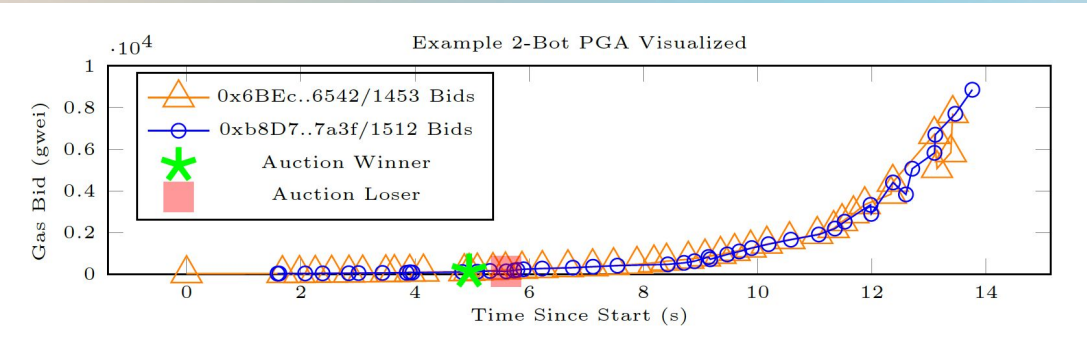
No inclusion or ordering consensus, only one inclusion constraint:

All transactions must pay the prevailing **EIP-1559 basefee**

Order transactions in **decreasing order of gas price** is ~incentive-compatible

"First spot" in block has great value (capture arbitrage, react to market)

=> Induce higher value payments to proposer



Daian, Philip, et al. "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges." *arXiv preprint arXiv:1904.05234* (2019).

Ordering matters!

There is **value** to acting upon some state of the world on-chain

$v(s)$ = Value of transaction executed on state s

Example:

- $v(\{\text{ arbitrage exists }\})$ = 1,000,000
- $v(\{\text{ arbitrage is gone }\})$ = 0

Whoever attempts to capture arbitrage is willing to spend up to 1,000,000

Ultimately, who is best-placed to capture arbitrage?

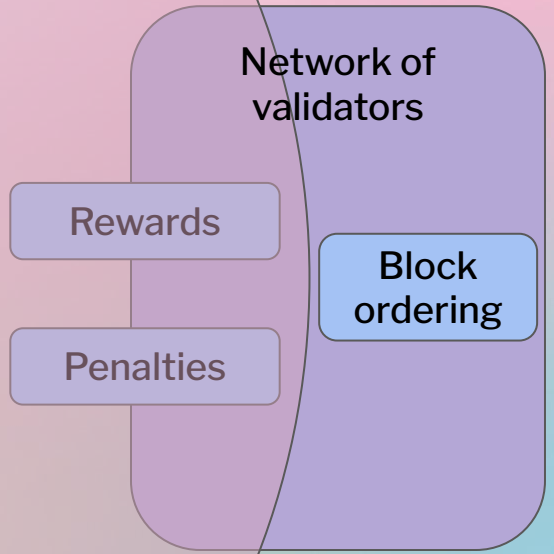
Block proposer, final word on transaction ordering, insert their own tx

How to allocate first slot?

We can think through alternatives:

- **No allocation:** Priority Gas Auctions, “ad hoc” English auction, wasteful
- **Randomise:** Not IC for proposer, also wasteful, “spray and pray”
- **First-come first-served:** Induces latency games, redirect value towards network infrastructure (e.g., HFT-style fast pipes)
- **Private channels:** User/proposer collusion, favours “dark” pools/vertical integration/exclusive order flow
- **Auctions:** Flashbots Auction, open market for **bundles** vying for top-of-block inclusion

Ethereum protocol



(h/t Alex and Quintus at Digital Finance Seminar)

Why you should care about Maximal Extractable Value (MEV)

by Quintus Kilbourn & Alex Obadia (Flashbots)

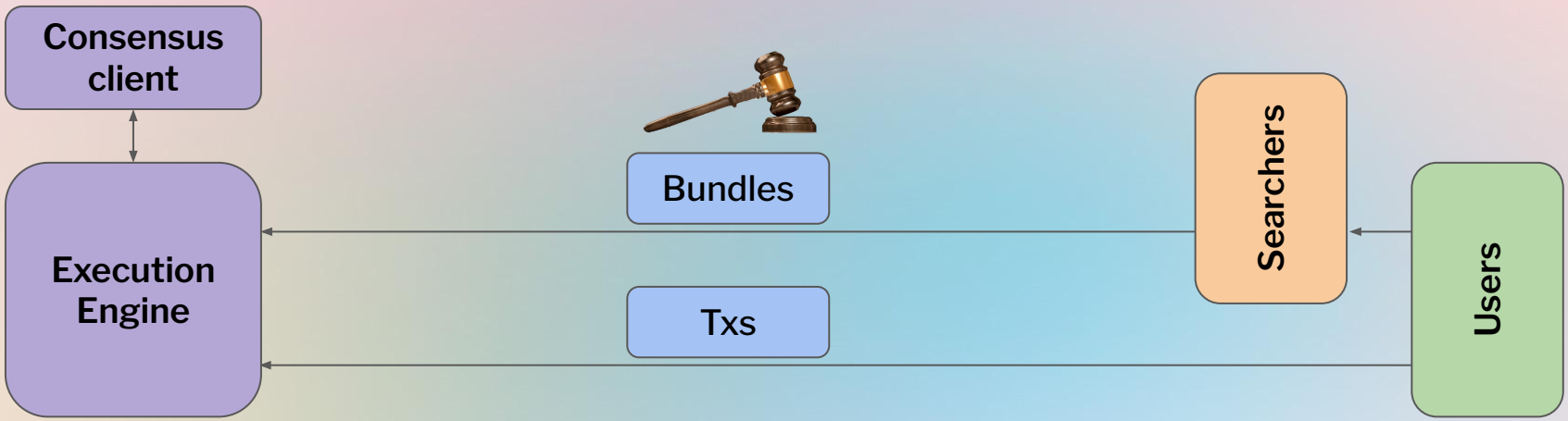
Today we give the **protocol's perspective** on all of this

Alex and Quintus go deeper into the structure of MEV and its extraction

Block construction in Proof-of-Work

Let **searchers** bid for **bundle** inclusion at top-of-block

Auction between **block proposer** and **searchers**

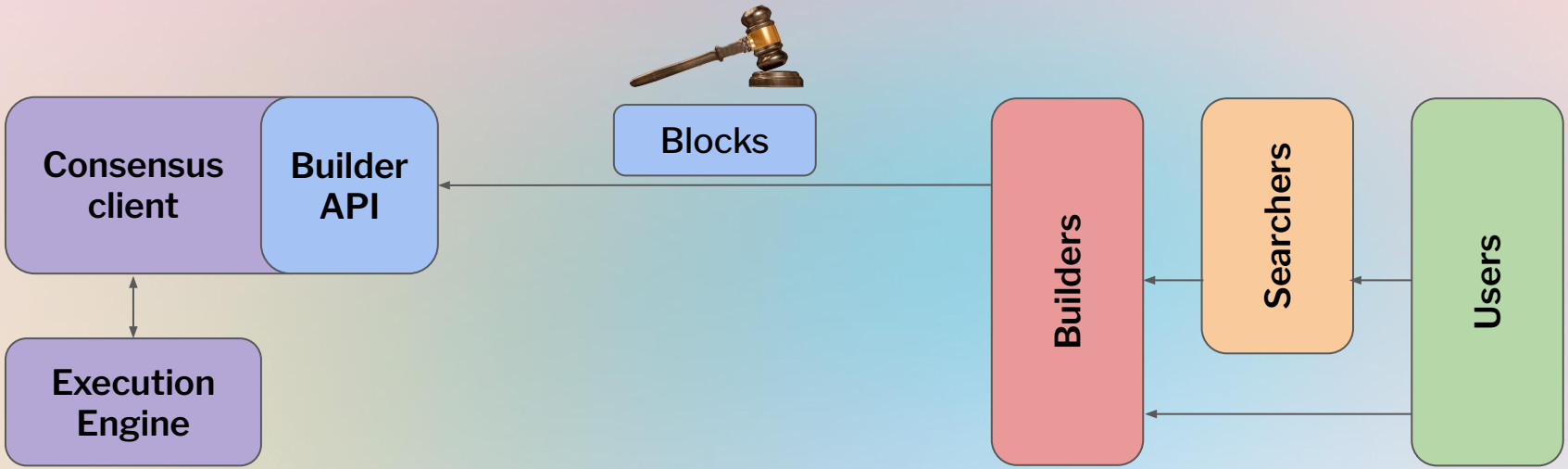


Block construction in Proof-of-Stake

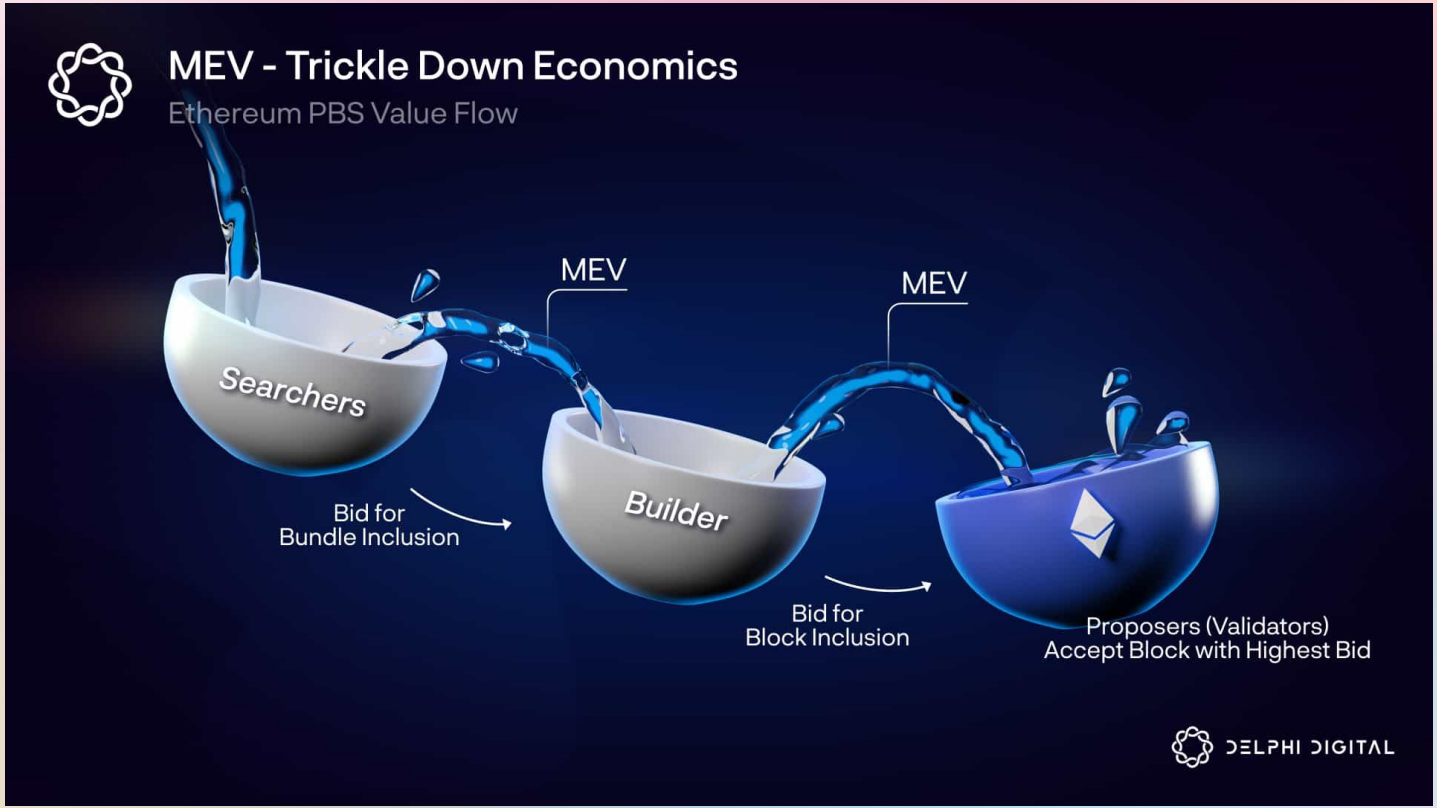
In PoS, **auction for top-of-block** doesn't work

Trust model is different with solo validators...

Idea: Validators receive full blocks from **builders**, builders **bid** for inclusion



MEV trickle down (J. Charbonneau)



The present of PBS: mev-boost for PoS Ethereum

Bidding phase

Builders send full block + bid to **Relay**

Relay validates bids (block validity + bid amount)

Relay = third-party for *fair exchange*

Bid selection

Proposer receives bids from **Relays**

Default: **mev-boost** selects highest bid

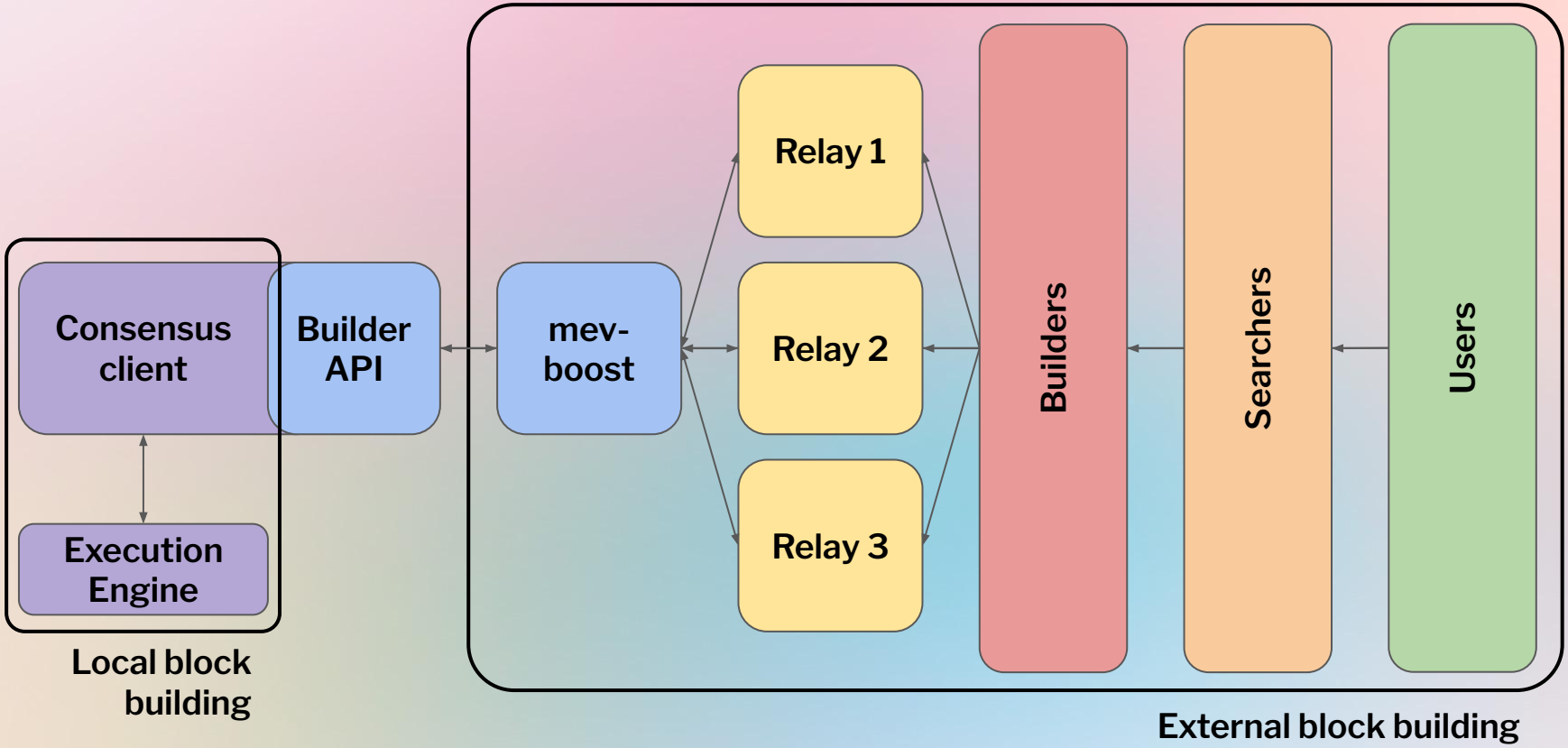
Proposer signs bid, can no longer make another block

Delivery

Relay receives signed bid

Relay releases full block to **Network**

Block-building today (in Proof-of-Stake)



Section 4

Seeing like a protocol

Out-of-protocol markets

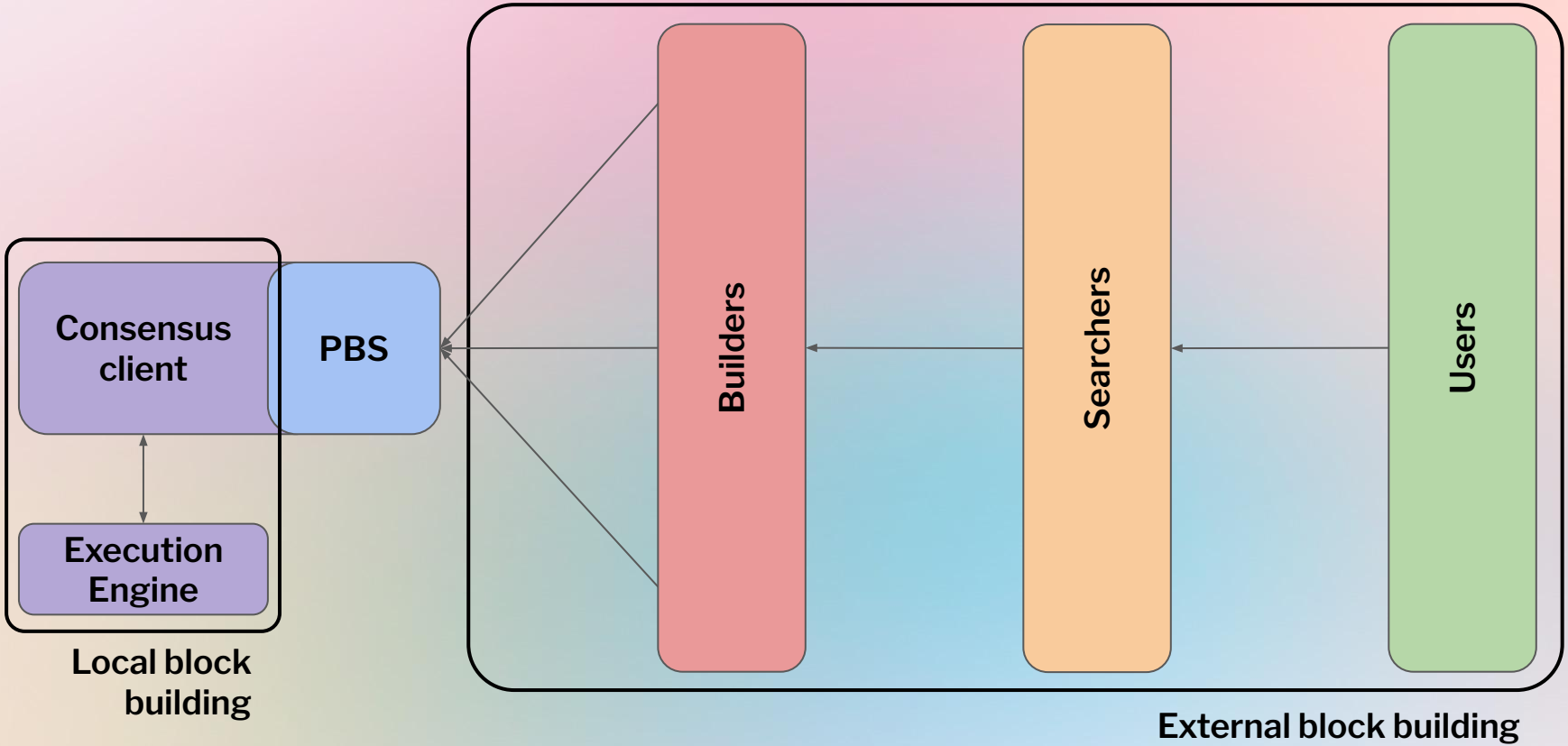
PBS is expressed today **out-of-protocol**, via mev-boost

Relays are trusted parties, validate the good being sold,
guarantee payments to **Proposer** in case of **Builder** failure

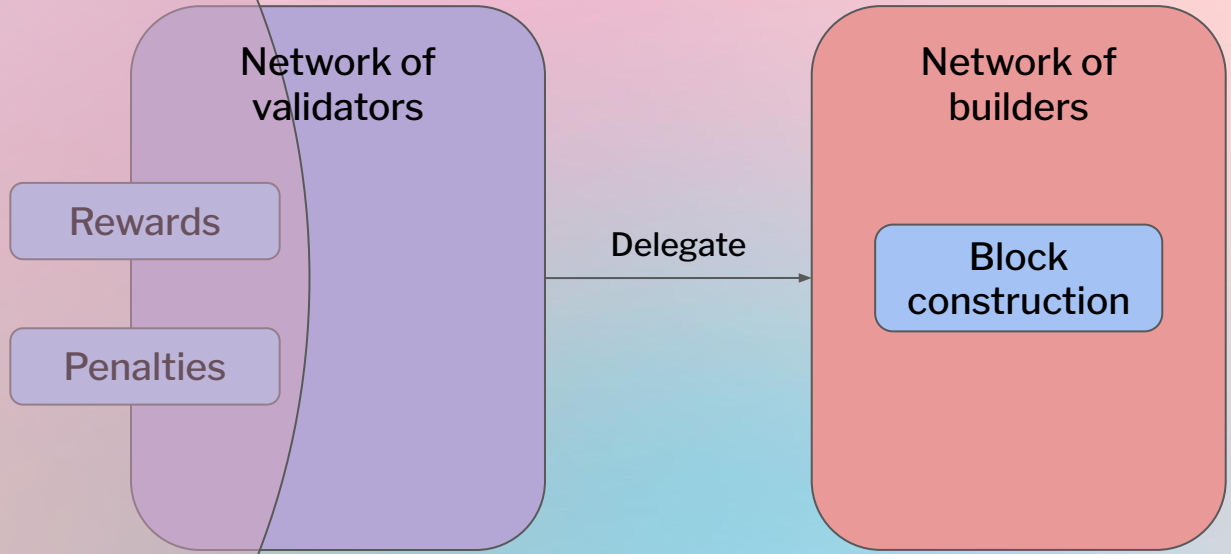
Is this enough?

Should the protocol recognise the separation and entrench it?

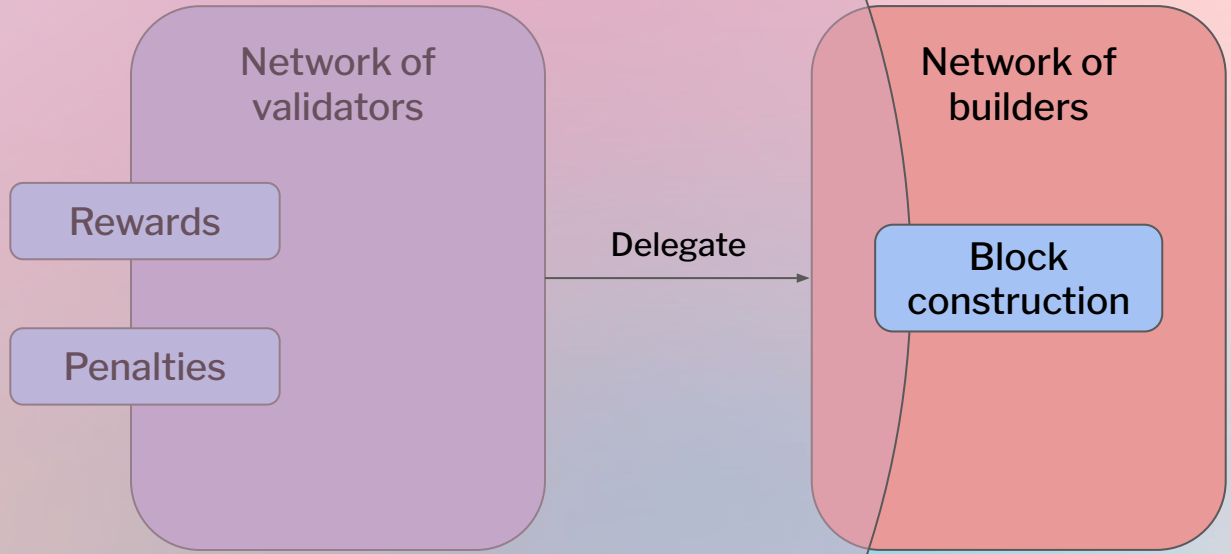
Block-building tomorrow? (In-protocol PBS)



Ethereum protocol



Ethereum protocol



Why delegate?

Builder separation for the **execution payload** “happened to” the protocol…

But perhaps natural **division of labor**?
Making a block is hard, but ***verifying*** it is easy

- Builders summoned for the one-shot block **construction**,
  increase protocol supply or maximise efficiency of available resources
- All nodes verify the work is done, since **verification** is cheap
- All proposers receive commensurate rewards,
  no preference to sophisticated/trusted proposers

Lean into it: **Proposer-Builder Separation**

Opportunities of outsourcing: Danksharding

Earlier we discussed **data-gas**, consumed by rollups.

- EIP-4844 plans to provide on average **1 MB/block** extra data
- This is **~100x** more data than rollups consume *today*!!

Danksharding: Provide on average **16 MB/block** extra data!

- Builder makes a megablock, high bandwidth requirements
- Rest of the network validates availability with **Data Availability Sampling**
- DAS has low bandwidth/compute requirements

Max capacity can't be provided without builder

- (or massive increase of full node bandwidth, which is ❌)

Risks of outsourcing: Censorship resistance

Ethereum protocol: Include any and all transaction which pays the fee

Protocol-Proposer problem: Individual proposers may censor transactions

Large **decentralisation** of the validator set ⇒ Someone includes!

But proposers outsource block production to much less decentralised agents…

Proposer-Builder problem: Builders may censor individual transactions, despite proposer's preferences

Solution: Proposers require inclusion of some transaction list by their builders

Works better if protocol is involved, otherwise rational to keep list empty…

“Proposer-Builder Separation” separation

A market structure

Asymmetry between block ***construction*** and block ***verification***

Cost of block construction is potentially high (in resources, or expertise), but one-shot

Cost of block verification remains low

Proposer-Builder Separation:
The proposer outsources block construction to third-parties

An allocation mechanism

What does the proposer outsource exactly?

What is the space of contracts the protocol lets proposers enter into?

Outsource the whole block? Part of it?
Sell the rights in advance?
According to some inclusion list?

Many ways to do PBS!

What is Ethereum? 💫

Where do Ethereum's concerns stop?

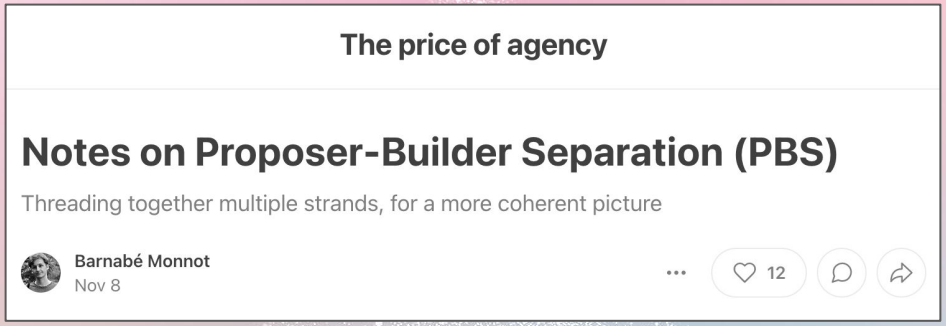
- **At the client level?** Provide more ways for **out-of-protocol markets** to organise?
- **At the market structure?** Recognise builder role, make sure proposer is protected from builder failure? Moral hazard?
- **At the allocation mechanism?** Determine some/all markets/mechanisms, fully specify contracting space between proposers and third-parties

Some (incomplete) ways to think about it

Risk for the protocol? Safety, liveness, throughput? Does it maximise welfare? Outsourcing may be good! Sometimes, more incentive-alignment.

Risk for the proposer? Should the protocol protect them?

Thank you!



https://barnabe.substack.com/p/pbs

Barnabé Monnot

Robust Incentives Group (RIG), Ethereum Foundation

barnabe@ethereum.org

@barnabemonnot