

Post-Merge Client Architecture

Adrian Sutton

Lead Blockchain Engineer, ConsenSys

About Me



CONSENSYS



TEKU




HYPERLEDGER

BESU


The background features a complex geometric pattern of overlapping triangles and lines. The colors transition from warm tones (orange, yellow) on the left to cool tones (teal, green, blue) on the right. The text is centered in a dark, monospace-style font.

Everything you ever need to know
about Ethereum clients...



Everything you ever need to know
about Ethereum clients...

In 25 minutes...



Everything you ever need to know
about Ethereum clients...

In 25 minutes...

Hmm...



Ethereum Clients

Three Key Things To Know

Commonalities

Networking

- Peer discovery & management
- Sybil resistance
- DoS prevention
- Peer scoring
- Gossiping new data

Blockchain

- Tracking the block tree
- Re-org handling
- Operation/transaction mempools
- Sync
- State + transition function

Differences

Pretty much all the specific technologies

- LibP2P vs DevP2P
- Discv5 vs Discv4/DNS
- SSZ vs RLP
- Sha256 vs keccak
- BLS vs ECDSA

Slot Timing

+0s
Block



New slot begins every 12s.

Selected producer should create and publish their block as quickly as possible.

+4s
Attestations



Validators scheduled for this slot produce an attestation.

Attests to validator's view of the current chain head.

Produced earlier if block already imported.

+8s
Aggregates



Aggregators gather individual attestations into aggregates.

Can only aggregate matching attestations.

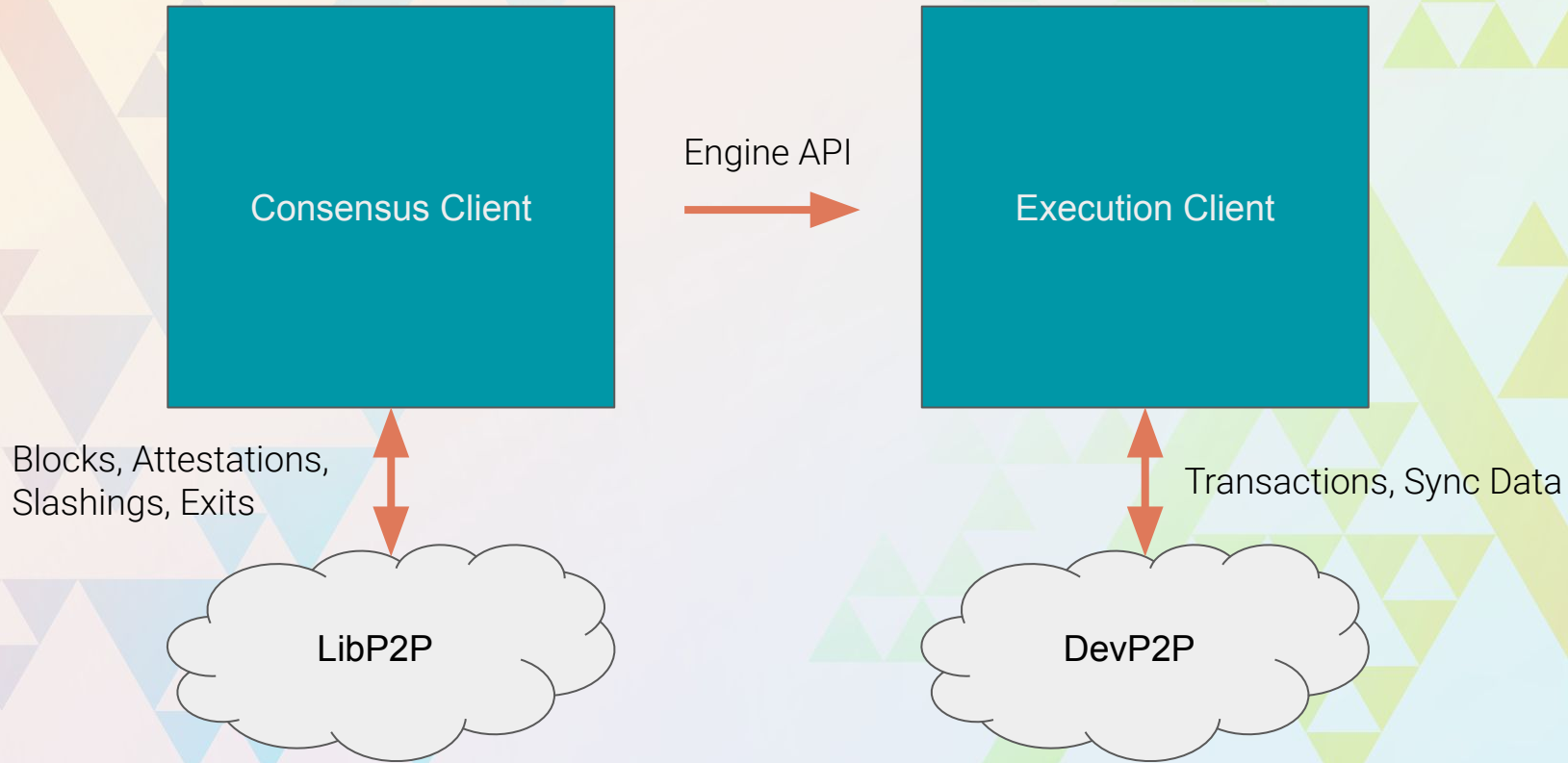
Key Differences - Large Execution State

- Storing world state efficiently is a huge challenge for execution clients
- Multiple different strategies for
 - Syncing
 - Pruning
 - Indexing
- Major performance challenges



The Post-Merge World

Deployment Model



Solution: Light Consensus Client

Build a light consensus client into execution clients.

Pros:

- Simple for users to run a basic node
- Reduced system requirements

Cons:

- Will always trail one slot behind head
- Not suitable for validator nodes
- Reduced security guarantees



Solution: Combine Clients

Combine the consensus and execution clients into a single executable.

Pros:

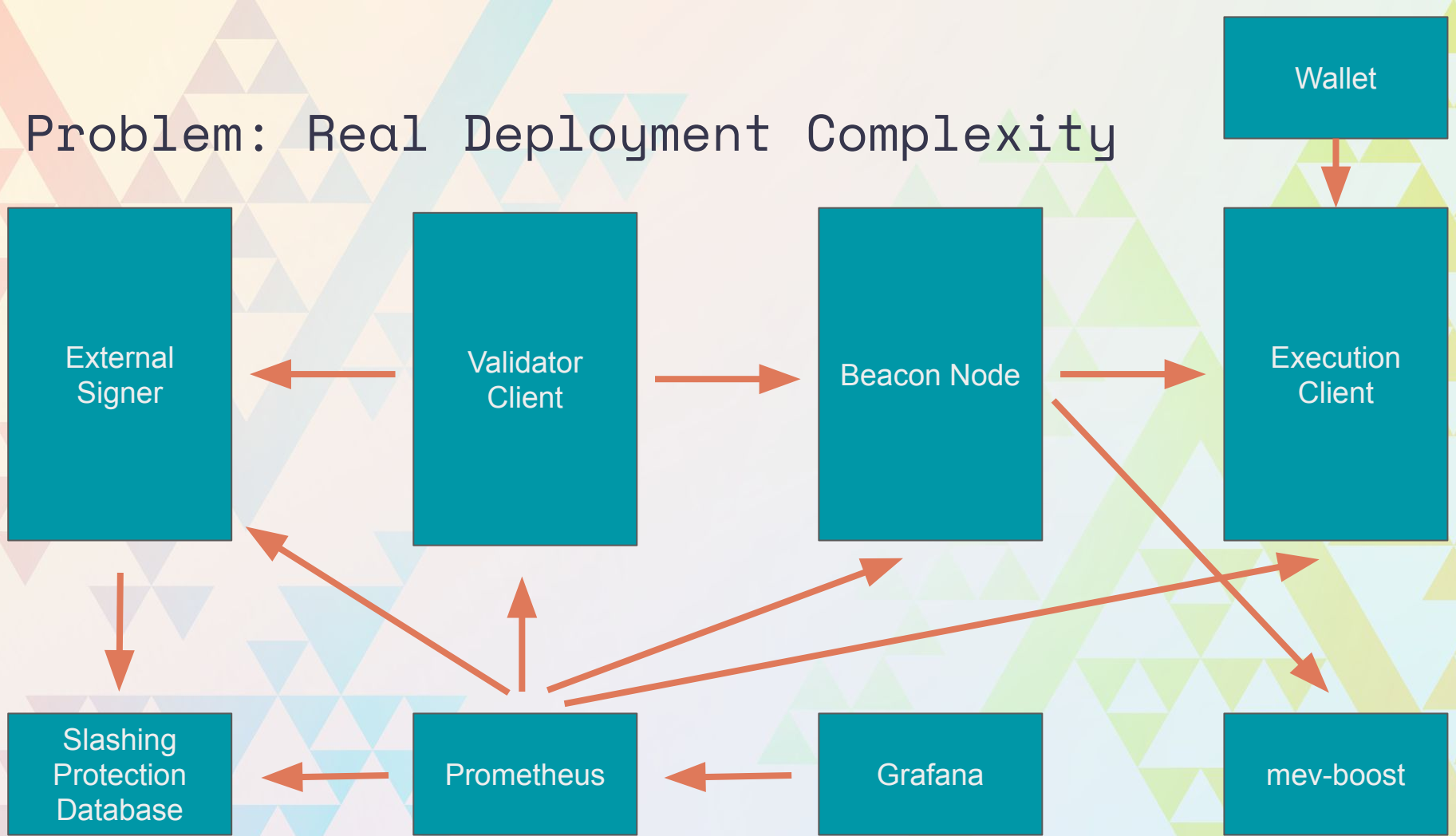
- Simple for users to start a new node

Cons:

- Bad for client diversity
- Dependency conflicts
- Increases coupling and cognitive load for core devs
- Encourages client scope creep



Problem: Real Deployment Complexity



Solution: EthOS[†]

Move coordination up a layer by using/improving/building tools to manage configuration and integration of different clients and components.

Pros:

- Maintains decoupling of consensus and execution clients
- Decouples development model from deployment model
- Can include a full suite of functionality, monitoring etc
- Already exists in [eth-docker](#), [eth-wizard](#), [DAppNode](#), [Stereum](#) and others

Cons:

- Hides complexity rather than removing it
- Extra layer means yet more software to build and maintain

[†] Not actually an OS





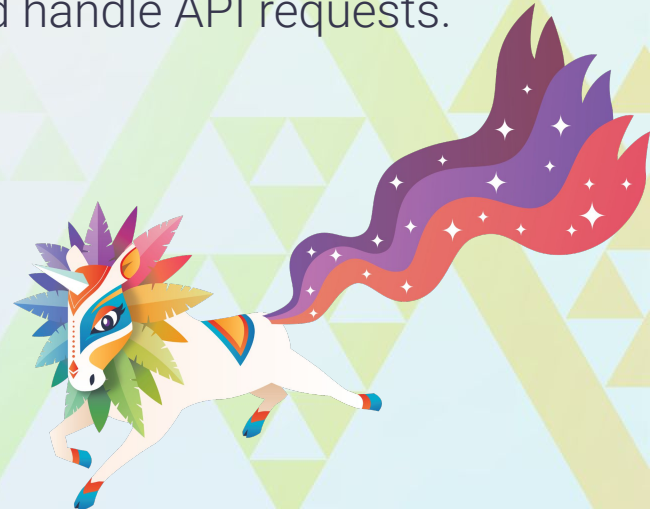
More Problems & Opportunities

Problem: Data Duplication

Consensus clients and execution clients both store the execution payload data.

All those transactions add up to a lot of disk space.

Both clients need to access the data to send to peers and handle API requests.



Solution: New engine API Methods

engine_getPayloadBodies - <https://github.com/ethereum/execution-apis/pull/146>

Efficiently request execution payload bodies from the execution client.

Pros:

- De-duplicates transaction data
- Batch requests for data make it reasonably efficient

Cons:

- More requests for the execution client database to handle
- Increases coupling between clients



Solution: Prune Historic Blocks

Store duplicate transactions but prune blocks aggressively.

Consensus specs only require storing ~5 months worth of blocks.

Pros:

- Simpler than deduplicating transactions
- Consensus disk space requirements become almost static

Cons:

- Doesn't help archive nodes
- Older blocks become harder to find / potentially unavailable.



Problem: Non-canonical Blocks and Re-orgs

EL *can* return ACCEPTED and not execute non-canonical blocks

- Risks putting the consensus client into optimistic sync mode
- Might cause validator duties to be missed
- May make switching to that fork slower

Take-away:

- Short re-orgs need to be highly optimised to avoid missing validator duties
- Long re-orgs should be very rare and may not need to be so optimised

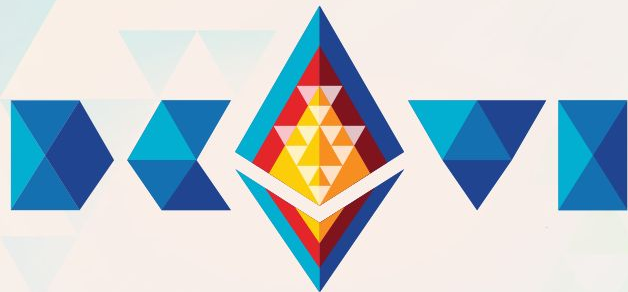


Learn from “the other side”

Embrace multi-component

Clean up some loose ends





Thank you!

Adrian Sutton

Lead Blockchain Engineer, ConsenSys

adrian.sutton@consensys.net



[@ajsutton](https://twitter.com/ajsutton)