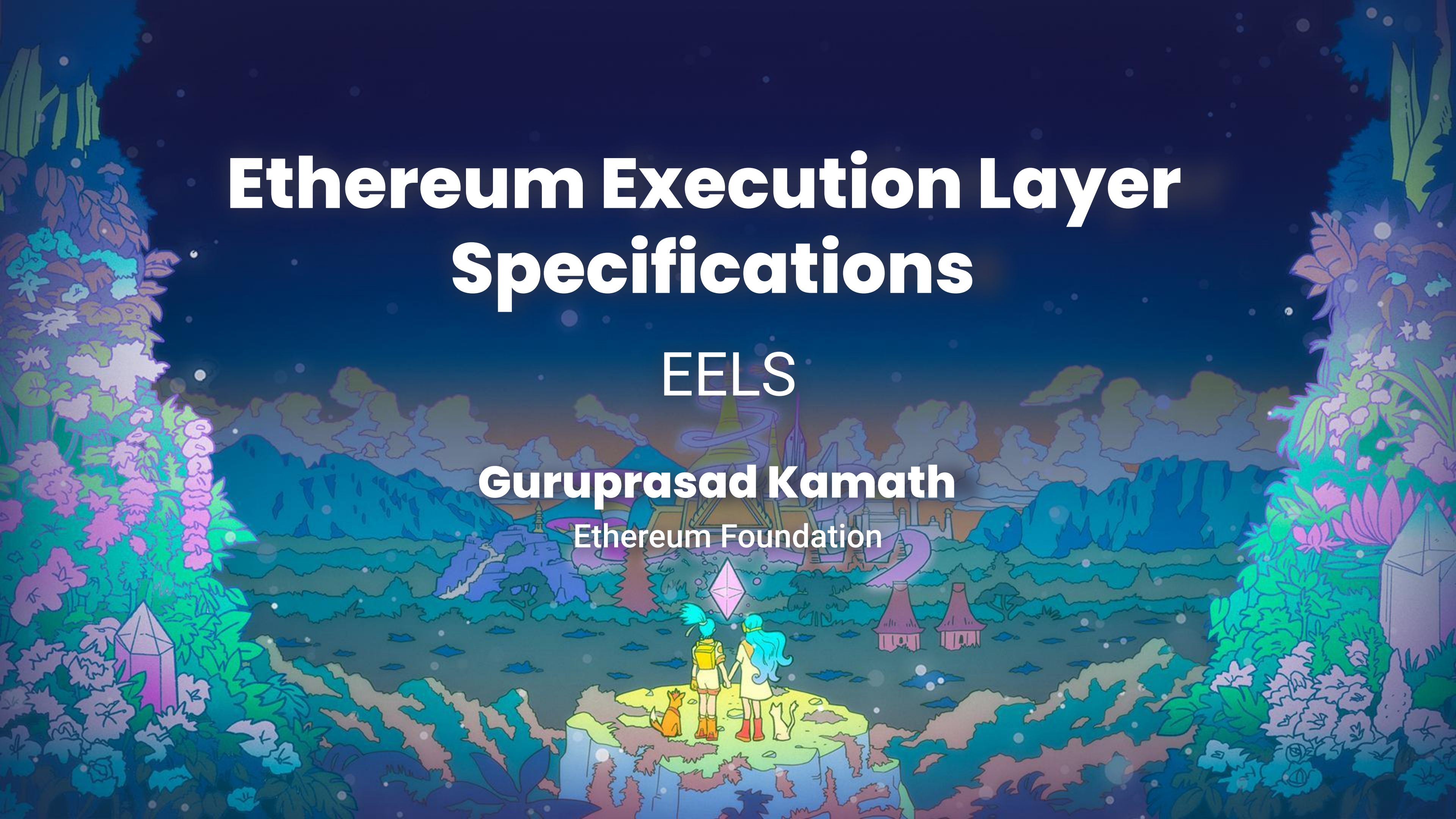


Ethereum Execution Layer Specifications

EELS

Guruprasad Kamath

Ethereum Foundation

Section 1

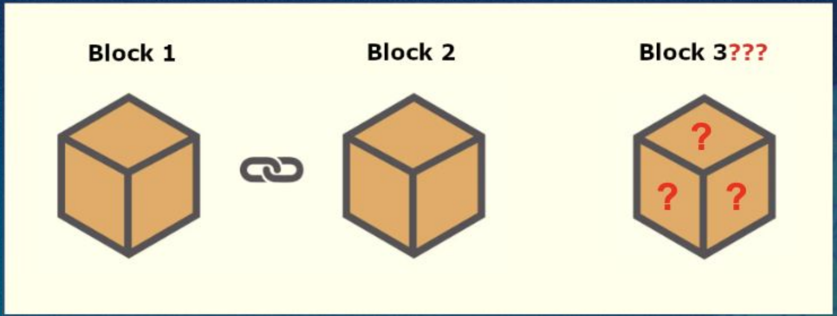
What is EELS?

What is EELS?

Specify the execution layer

State transition function

- Is the new block valid?
- If yes, what is the new state?



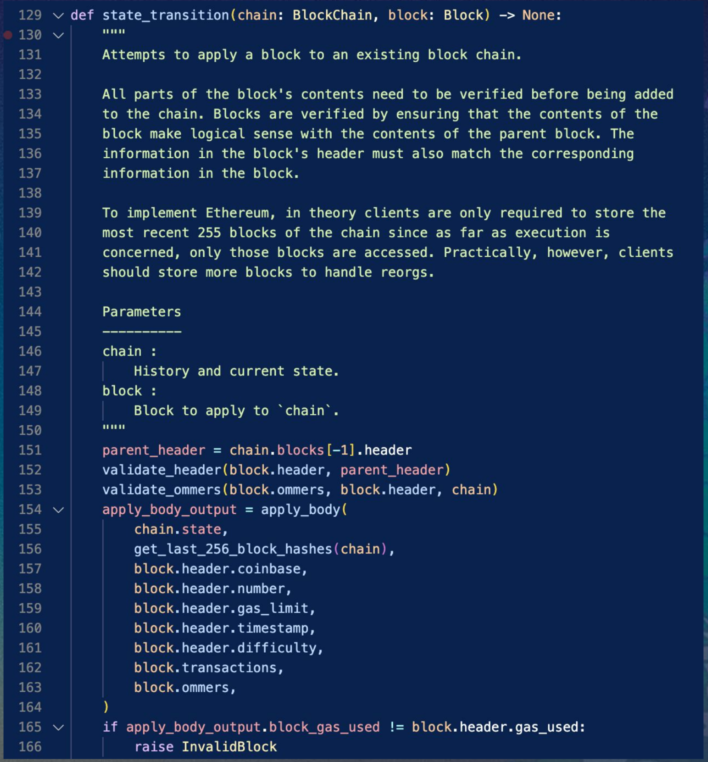
Not specified

- Re-orgs
- Networking
- JSON-RPC
- …

https://github.com/ethereum/execution-specs

What is EELS?

- Written in Python
- Executable
- Optimized for readability
    - Extensively documented
    - Almost pseudo-code
- Playground for prototyping new EIPs

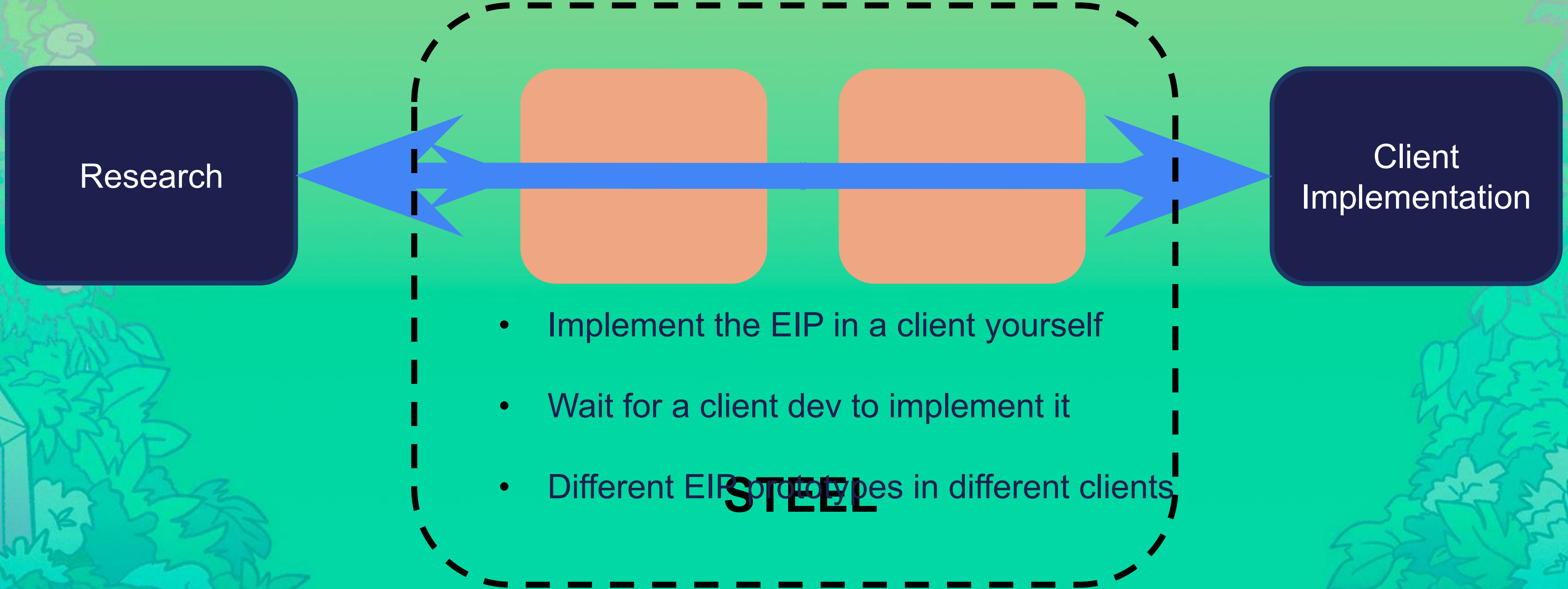


```python
def state_transition(chain: BlockChain, block: Block) -> None:
    """
    Attempts to apply a block to an existing block chain.

    All parts of the block's contents need to be verified before being added
    to the chain. Blocks are verified by ensuring that the contents of the
    block make logical sense with the contents of the parent block. The
    information in the block's header must also match the corresponding
    information in the block.

    To implement Ethereum, in theory clients are only required to store the
    most recent 255 blocks of the chain since as far as execution is
    concerned, only those blocks are accessed. Practically, however, clients
    should store more blocks to handle reorgs.

    Parameters
    ----------
    chain :
        History and current state.
    block :
        Block to apply to `chain`.
    """
    parent_header = chain.blocks[-1].header
    validate_header(block.header, parent_header)
    validate_ommers(block.ommers, block.header, chain)
    apply_body_output = apply_body(
        chain.state,
        get_last_256_block_hashes(chain),
        block.header.coinbase,
        block.header.number,
        block.header.gas_limit,
        block.header.timestamp,
        block.header.difficulty,
        block.transactions,
        block.ommers,
    )
    if apply_body_output.block_gas_used != block.header.gas_used:
        raise InvalidBlock
```

Section 2

Why do we need EELS?

EL Development Cycle



Research

Client Implementation

- Implement the EIP in a client yourself
- Wait for a client dev to implement it
- Different EIP prototypes in different clients

STEEL

Advantages of using EELS

- Faster iteration cycle for development
- Throw light on possible weird EVM edge cases
- One-stop shop for EIP prototyping
  - Interaction between EIPs
  - Leverage EELS tooling (test filling, code analysis etc.)
- Closer integration with EEST
- Support from the EELS team

Section 3

Where is EELS right now?

We are here!

- All forks upto and including Prague
- Working implementation of EOF
- Default test filler for EEST
- Consume all tests
- Verify mainnet blocks (upto a Cancun)

We’re heading here!

- First to implement EIPs/updates
- More tooling for EIP authors
- Integrate in the EIP process
- Participate in Devnets

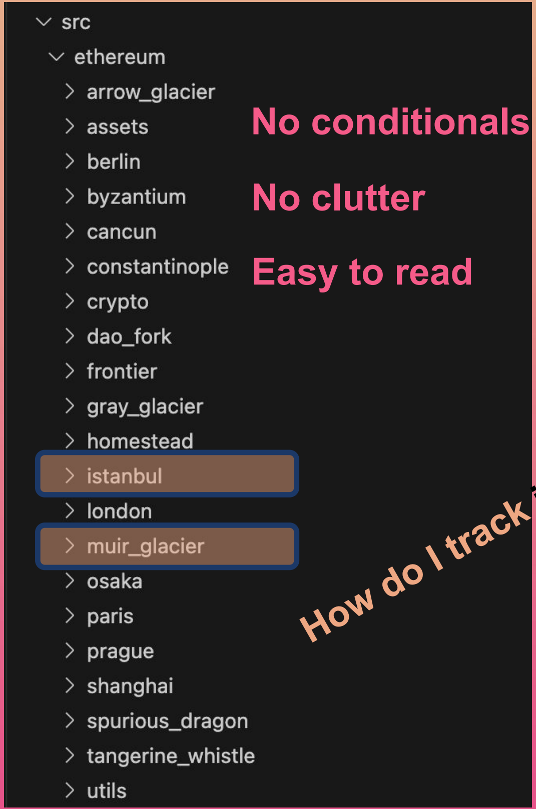
Section 4

How do I use EELS?

How do I use EELS?

- ethereum/execution-specs
- Support Python 3.10+
- Forks live on Mainnet
  - master branch
- Forks under development
  - forks/<FORK NAME>
  - eips/<FORK NAME>/<EIP>

Separate folder for each fork



```
⌄ src
  ⌄ ethereum
    › arrow_glacier
    › assets
    › berlin
    › byzantium
    › cancun
    › constantinople
    › crypto
    › dao_fork
    › frontier
    › gray_glacier
    › homestead
    › istanbul
    › london
    › muir_glacier
    › osaka
    › paris
    › prague
    › shanghai
    › spurious_dragon
    › tangerine_whistle
    › utils
```

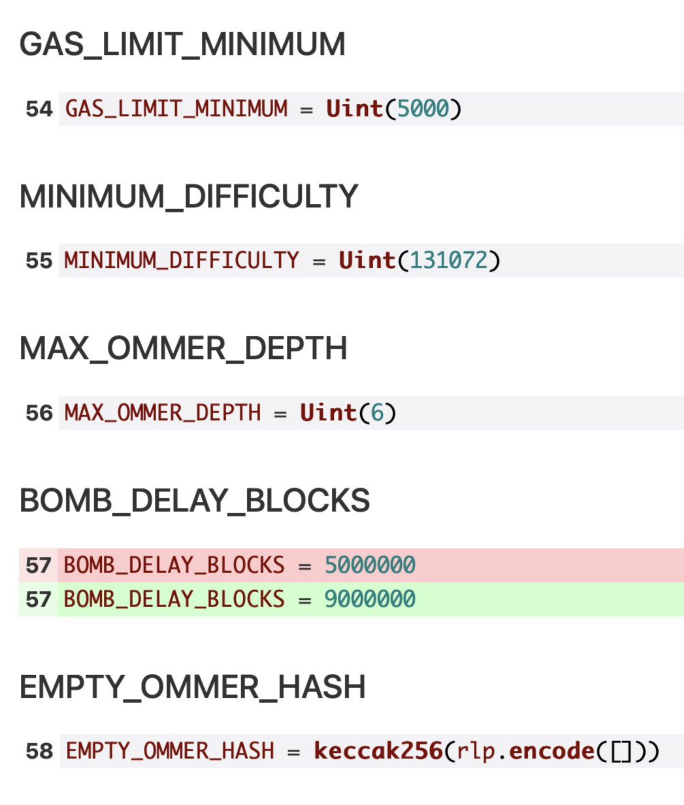
No conditionals

No clutter

Easy to read

How do I track individual upgrades?

Diff Documentation focussed on upgrades



GAS_LIMIT_MINIMUM

```
GAS_LIMIT_MINIMUM = Uint(5000)
```

MINIMUM_DIFFICULTY

```
MINIMUM_DIFFICULTY = Uint(131072)
```

MAX_OMMER_DEPTH

```
MAX_OMMER_DEPTH = Uint(6)
```

BOMB_DELAY_BLOCKS

```
BOMB_DELAY_BLOCKS = 5000000
BOMB_DELAY_BLOCKS = 9000000
```

EMPTY_OMMER_HASH

```
EMPTY_OMMER_HASH = keccak256(rlp.encode([]))
```

Team

Sam
@samwilsn

Peter
@peterdavies

Guru
@gurukamath

How can you contribute?

👉 Use EELS and provide feedback

👉 Implement your proposed EIP

Thank you!