Blockchains Meet Distributed Hash Tables:  
Decoupling Validation from State Storage

Matteo Bernardini, Diego Pennino, and Maurizio Pizzonia
The problem
The problem

Scalability
The problem

Scalability

Consensus
The problem

Scalability

Consensus

Problems:

Storage
The problem

- Scalability
- Consensus

Problems:

- Time of Synchronization
The problem

Scalability

Consensus

Problems:
- Time of Synchronization
- Storage capacity
The problem

Scalability

Consensus

Problems:

✓ Time of Synchronization
✓ Storage capacity
✓ Validation needs Storage

Storage
Overview of our work

Full Node divided into two roles:
Overview of our work

Full Node divided into two roles:

1. **Storage role:**
   - tunable occupied space

Main idea:
- a partitioned Ledger
Overview of our work

Full Node divided into two roles:

1. **Storage role**: ✔ tunable occupied space

2. **Validation role**: ✔ small chain with a constant size

Main idea:
- a partitioned Ledger
- validation from a secure starting point which **IS** NOT the local ledger
Storage Role
Ethereum Storage Role: the Ledger

State elements

\[
\begin{align*}
k_0 &= 000 & V_0 \\
k_1 &= 001 & V_1 \\
k_2 &= 010 & V_2 \\
k_3 &= 011 & V_3 \\
k_4 &= 100 & V_4 \\
k_5 &= 101 & V_5 \\
k_6 &= 110 & V_6 \\
k_7 &= 111 & V_7
\end{align*}
\]
Ethereum Storage Role: the Ledger

Prefix tree

State elements

$V_0$

$k_0 = 000$

$V_1$

$k_1 = 001$

$V_2$

$k_2 = 010$

$V_3$

$k_3 = 011$

$V_4$

$k_4 = 100$

$V_5$

$k_5 = 101$

$V_6$

$k_6 = 110$

$V_7$

$k_7 = 111$
Ethereum Storage Role: the Ledger

Prefix tree - Merkle Hash Tree (MHT)
Prefix tree - *Merkle Hash Tree (MHT)*

- Prefix tree
- Merkle Hash Tree (MHT)
**Ethereum Storage Role:** the Ledger

**MHT - the proof**

```
\begin{align*}
\text{h(h}_0\text{|h}_1) \\
&\text{h(h}_0\text{|h}_0) \quad \text{h(h}_1\text{|h}_1) \\
&\text{h(h}_0\text{|h}_0) \quad \text{h(h}_0\text{|h}_1) \\
&\text{h(V}_0) \quad \text{h(V}_1) \quad \text{h(V}_2) \quad \text{h(V}_3) \quad \text{h(V}_4) \quad \text{h(V}_5) \quad \text{h(V}_6) \quad \text{h(V}_7) \\
\end{align*}
```
Ethereum Storage Role: the Ledger

\[ h(h_0 | h_1) \]

\[ h(h_{00} | h_{01}) \]

\[ h(h_{00} | h_{01}) \]

\[ h(V_0) \]

\[ h(V_1) \]

\[ h(V_2) \]

\[ h(V_3) \]

\[ h(V_4) \]

\[ h(V_5) \]

\[ h(V_6) \]

\[ h(V_7) \]
**Ethereum Storage Role:** *the Ledger*

**MHT - the proof**

- **h(h₀|h₁)**
- **h(h₀₀|h₀₁)**
- **h(h₀₀₀|h₀₀₁)**
- **h(V₀)**
- **h(V₁)**
- **h(V₂)**
- **h(V₃)**
- **h(V₄)**
- **h(V₅)**
- **h(V₆)**
- **h(V₇)**
Storage Role of a Node: 
Distributed Hash Table (DHT)

Hash table (key-value)
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

P2P
Storage Role of a Node: 
Distributed Hash Table (DHT)

Hash table (key-value)

P2P

Authority of a key = it stores its value
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

- put\((k,v)\)
- get\((k)\)

Primitives:

P2P

Authority of a key = it stores its value
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

Primitives:

➢ put(k,v)
➢ get(k)

Properties:

Authority of a key = it stores its value
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

Primitives:
- \(\text{put}(k,v)\)
- \(\text{get}(k)\)

Properties:
- Autonomy and decentralization

Authority of a key = it stores its value

P2P
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

Primitives:
- `put(k, v)`
- `get(k)`

Properties:
- Autonomy and decentralization
- Fault tolerance

Authority of a key = it stores its value

P2P
Storage Role of a Node: Distributed Hash Table (DHT)

Hash table (key-value)

Primitives:
➢ put(k,v)
➢ get(k)

Properties:
✔ Autonomy and decentralization
✔ Fault tolerance
✔ Scalability

Authority of a key = it stores its value
OUR Storage Role: the small Ledger

\[ MHT + DHT = \text{pruned Authenticated Data Structure (pADS)} \]
OUR Storage Role: the small Ledger

\[ MHT + DHT = \text{pruned Authenticated Data Structure (pADS)} \]

Authority:

- \( k_0 = 000 \)  \( V_0 \)
- \( k_1 = 001 \)  \( V_1 \)
OUR Storage Role:  

the small Ledger

\[ MHT + DHT = \text{pruned Authenticated Data Structure (pADS)} \]
**OUR Storage Role:** the small Ledger

**pruned Authenticated Data Structure (pADS)**

\[ k_0 = 000 \]

\[ V_0 \]

\[ k_1 = 001 \]

\[ V_1 \]
**Our Storage Role:**  *the small Ledger*

**pruned Authenticated Data Structure** (*pADS*)

\[
\begin{align*}
k_0 &= 000 \\
V_0 &= 0 \\
k_1 &= 001 \\
V_1 &= 1
\end{align*}
\]
OUR Storage Role: the small Ledger

pruned Authenticated Data Structure (pADS)

Pivot block
**OUR Storage Role:** Transaction

**Traditional Transaction**
- Sender(s)
- Receiver(s)
- Operation(s)
- Signature(s)

**Key ‘a’:**
- Value
- Proof
- Number of block

**Key ‘b’:**
- Value
- Proof
- Number of block

**Key ‘c’:**
- Value
- Proof
- n. pivot block
Validation Role
OUR Validation Role: truncated blockchain
OUR Validation Role: *truncated* blockchain
OUR Validation Role: truncated blockchain
OUR Validation Role: truncated blockchain
OUR Validation Role: truncated blockchain
OUR Validation Role: *block schema*

- Traditional Block Elements
OUR Validation Role: *block schema*

Traditional Block Elements

\[ t_1: \{ (k_0,V_0), (k_3,V_3) \} \]
\[ t_2: \{ (k_1,V_1), (k_2,V_2), (k_3,V_3) \} \]
\[ t_3: \{ (k_0,V_0), (k_1,V_1) \} \]

*keys involved in transactions with their values*
OUR Validation Role: *block schema*

Traditional Block Elements

$t_1$: \{ (k_0, V_0), (k_3, V_3) \}

$t_2$: \{ (k_1, V_1), (k_2, V_2), (k_3, V_3) \}

$t_3$: \{ (k_0, V_0), (k_1, V_1) \}

keys involved in transactions with their values

$pADS \tau$

$k_0 \rightarrow k_1 \rightarrow k_2 \rightarrow k_3$
OUR Validation Role: Validation block

New Block
OUR Validation Role: Validation block

New Block

\[ k_0 \quad k_1 \]
OUR Validation Role: Validation block

New Block

r

k₀

k₁
OUR Validation Role: **Validation block**

New Block
OUR Validation Role: **Validation block**
OUR Validation Role: Validation block

Pivot
OUR Validation Role: **Validation block**

Pivot
OUR Validation Role: Validation block

Pivot
OUR Validation Role: Mining block
OUR Validation Role: **Mining block**

\[
t_1: \{ (k_2, V_2), (k_3, V_3) , \text{ proofs, n. pivot blocks} \}
\]

\[
t_2: \{ (k_1, V_1), (k_2, V_2), (k_3, V_3) , \text{ proofs, n. pivot blocks} \}
\]

\[
t_3: \{ (k_0, V_0), (k_3, V_3) , \text{ proofs, n. pivot blocks} \}
\]
OUR Validation Role: **Mining block**

\[
t_1: \{ (k_2, V_2), (k_3, V_3) \text{, proofs, n. pivot blocks} \}
\]

\[
t_2: \{ (k_1, V_1), (k_2, V_2), (k_3, V_3) \text{, proofs, n. pivot blocks} \}
\]

\[
t_3: \{ (k_0, V_0), (k_3, V_3) \text{, proofs, n. pivot blocks} \}
\]
Recap: Transaction/block lifecycle
1. The transaction creator gets from DHT proofs of the involved keys
1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
1. The transaction creator gets from DHT proofs of the involved keys.

2. The transaction creator broadcast the transaction with proofs.

3. Miners use transaction proofs and τ's of d+f locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm.
1. The transaction creator gets from DHT proofs of the involved keys

2. The transaction creator broadcast the transaction with proofs

3. Miners use transaction proofs and $\tau$'s of $d+f$ locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm

4. The agreed block is broadcasted to all
Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
3. Miners use transaction proofs and $\tau$'s of $d+f$ locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm
4. The agreed block is broadcasted to all
5. All nodes validate the new block and update their pivot block
Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys

2. The transaction creator broadcast the transaction with proofs

3. Miners use transaction proofs and \( \tau \)'s of \( d+f \) locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm

4. The agreed block is broadcasted to all

5. All nodes validate the new block and update their pivot block

6. DHT nodes that are authority for keys involved in the new block also update their pADS
Conclusion and Future works

- Store a small amount of data (pADS, DHT, truncated chain)
- Validation without Ledger (proof and pivot)
- First synchronization of a node performed in less than a minute

-process implementation of this approach
- extensive tests
Thank you for your attention