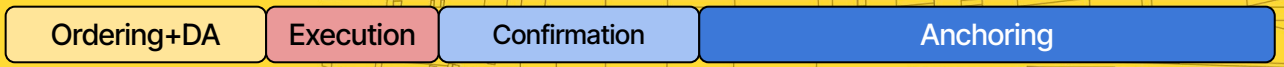
Movement

The modular future of Move

Andreas Penzkofer, PhD
Senior Research Engineer

Modular design

of Move Stack chains
\+ Movement Network



Move VM

- **Parallel Execution:** Parallel execution engine (BlockSTM) enhances transaction throughput and scalability.

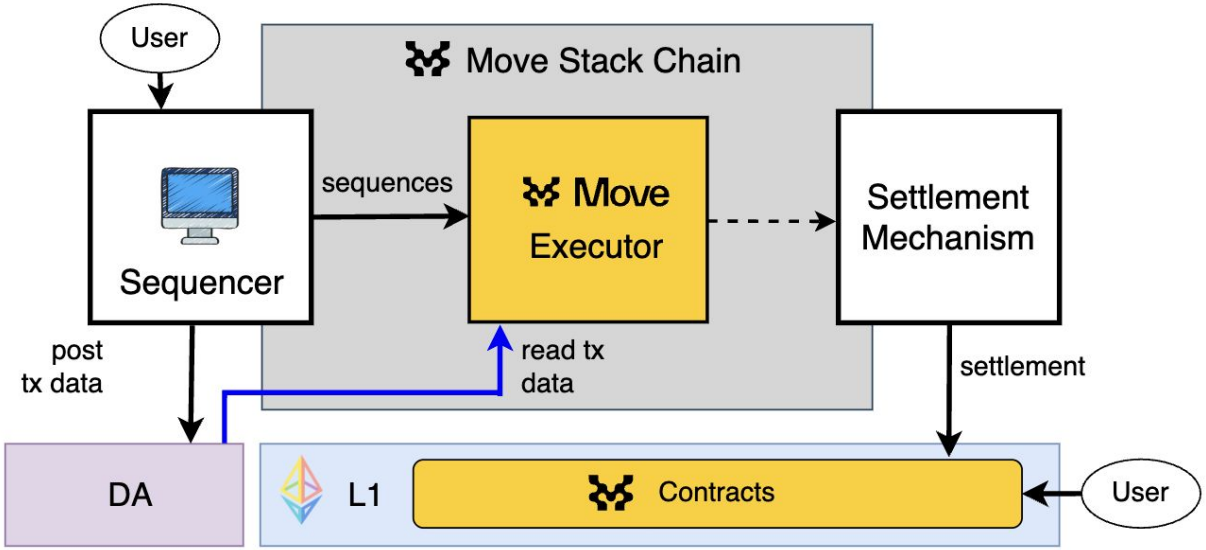
  *Benchmark performance: 160,000 tps **

- **Enhanced Security:** Bytecode **runtime verification** and **formal verification.**

  Ensures executable code is safe, correct, and adheres to stringent standards. Thus preventing vulnerabilities and reinforcing blockchain integrity.

* https://aptos.dev/en/network/blockchain/execution

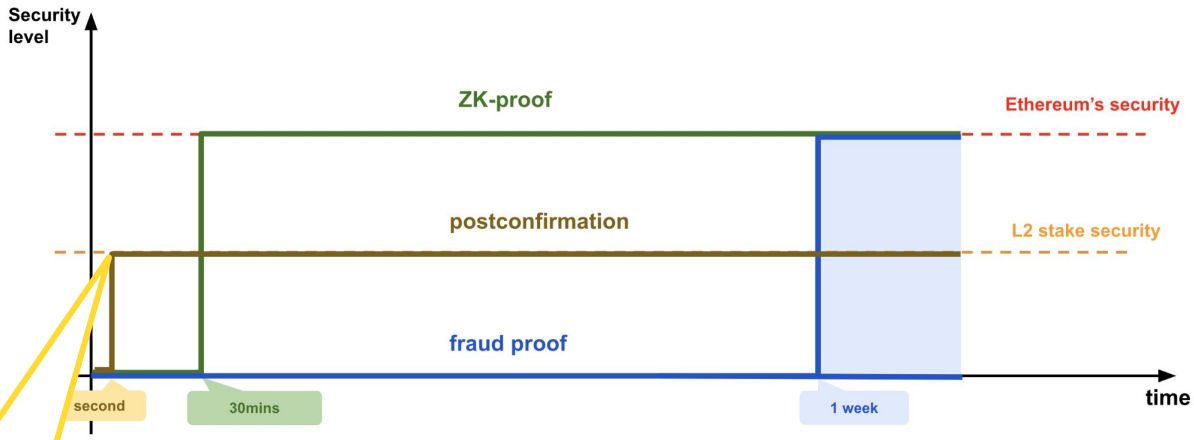
Move Stack Chain



- **Modular Architecture:** Modular architecture for creating customizable rollups or sidechains.
- **Freedom on Settlement:** Fraud proofs, ZK-proofs, Fast Finality Settlement (sidechain)
- **Flexible Component Selection**: Enables flexible component selection (e.g., sequencers, data availability, settlement).
- **Standardization**: Promotes standardization across chains for better developer and user experience.

Move Stack Side Chain

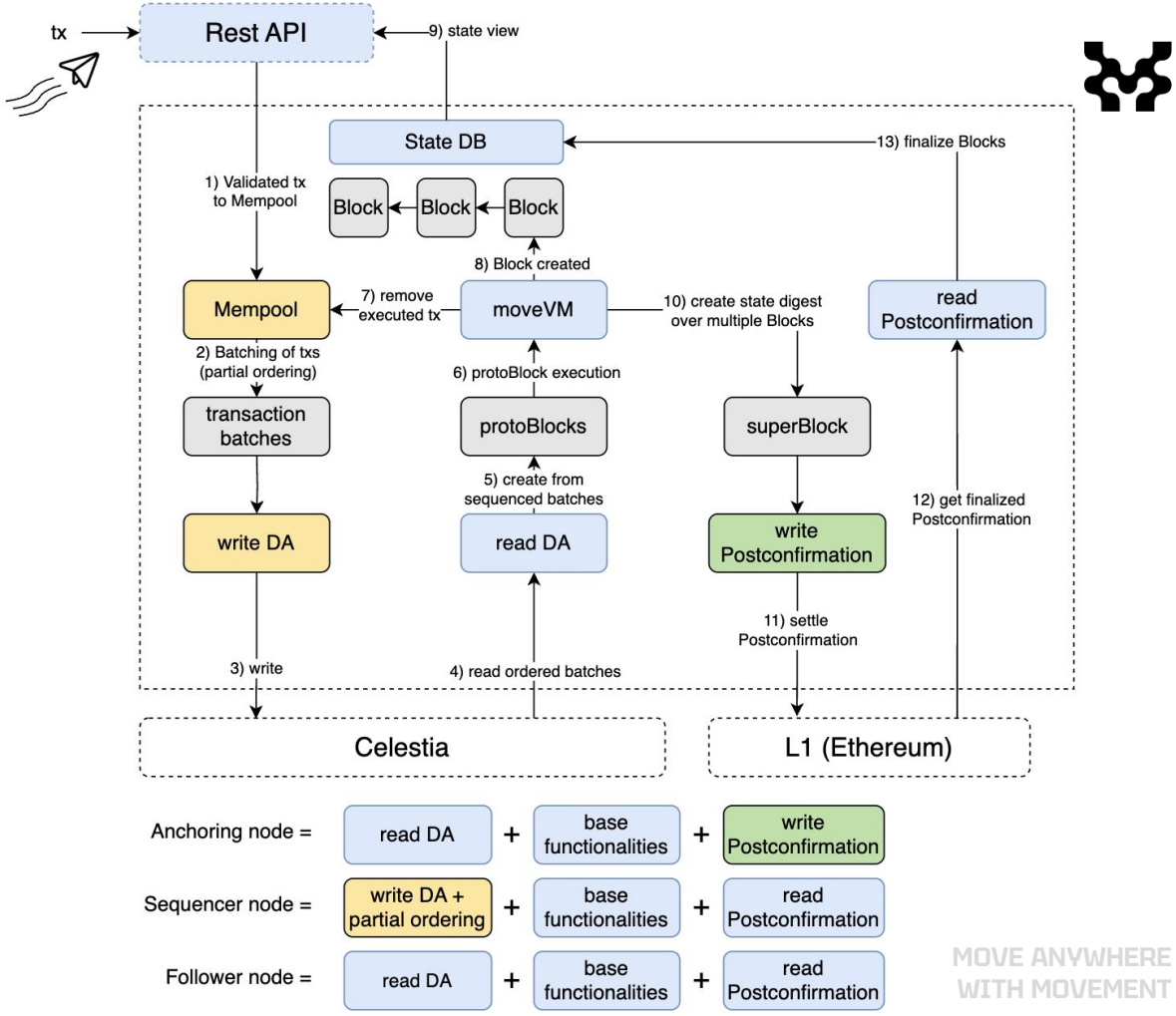
Fast Finality Settlement



Postconfirmations give early finality guarantees and make LATE rollbacks extremely unlikely

Move Stack Side Chain

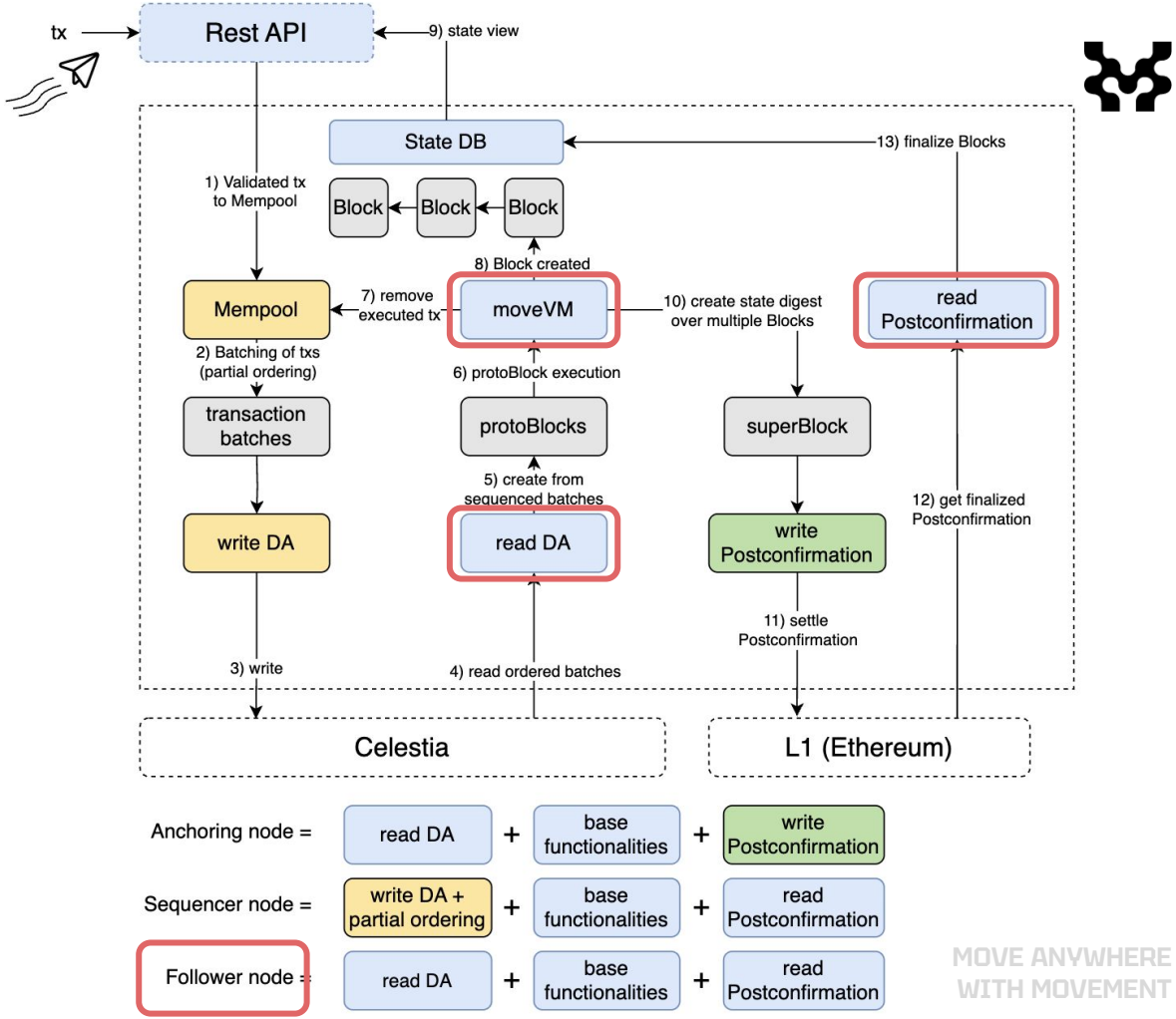
Transaction Lifecycle



- Permissioned sequencer
- Consensus and DA via Celestia
- Confirmation Layer and Anchoring via Fast Finality Settlement

Move Stack Side Chain

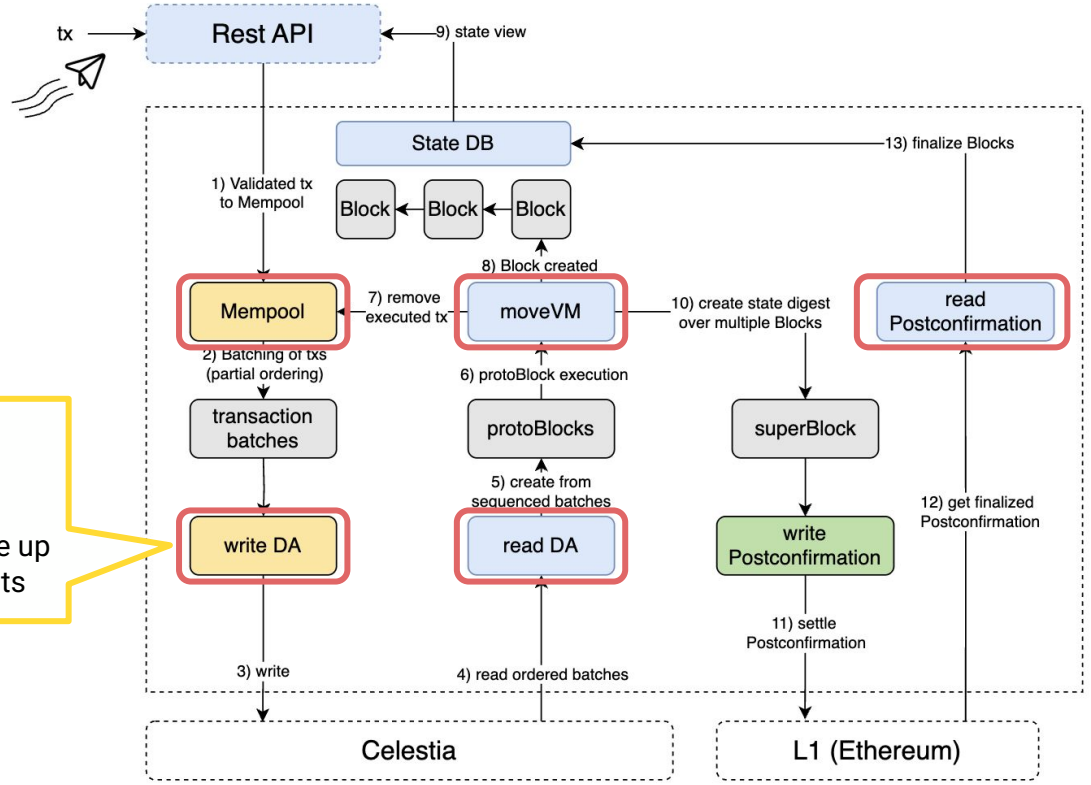
Transaction Lifecycle



- Permissioned sequencer
- Consensus and DA via Celestia
- Confirmation Layer and Anchoring via Fast Finality Settlement

Move Stack Side Chain

Transaction Lifecycle



Permissioned sequencer:

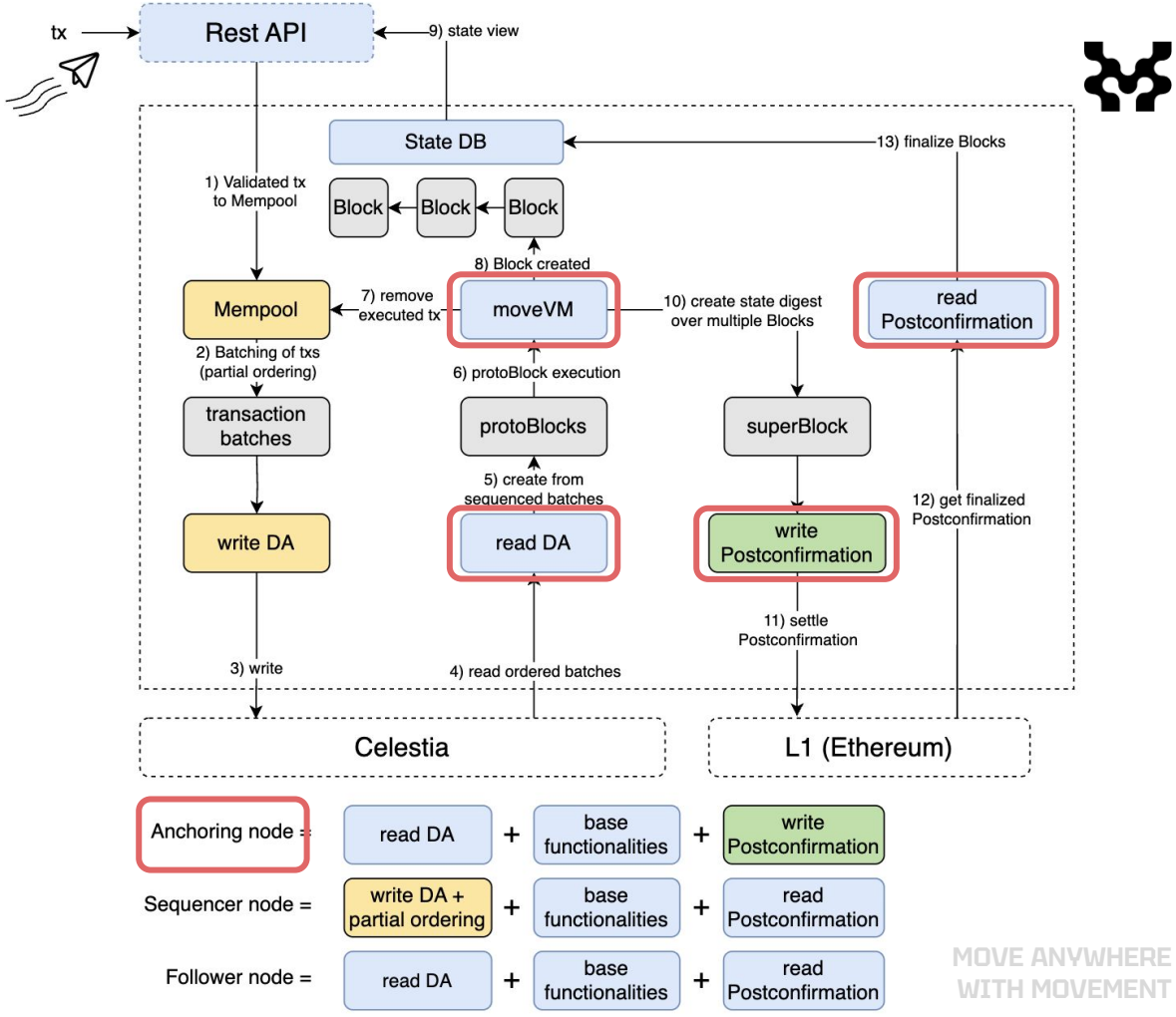
No need to give up sequencer rights

- Permissioned sequencer
- Consensus and DA via Celestia
- Confirmation Layer and Anchoring via Fast Finality Settlement

Anchoring node =	read DA	+	base functionalities	+	write Postconfirmation
Sequencer node =	write DA + partial ordering	+	base functionalities	+	read Postconfirmation
Follower node =	read DA	+	base functionalities	+	read Postconfirmation

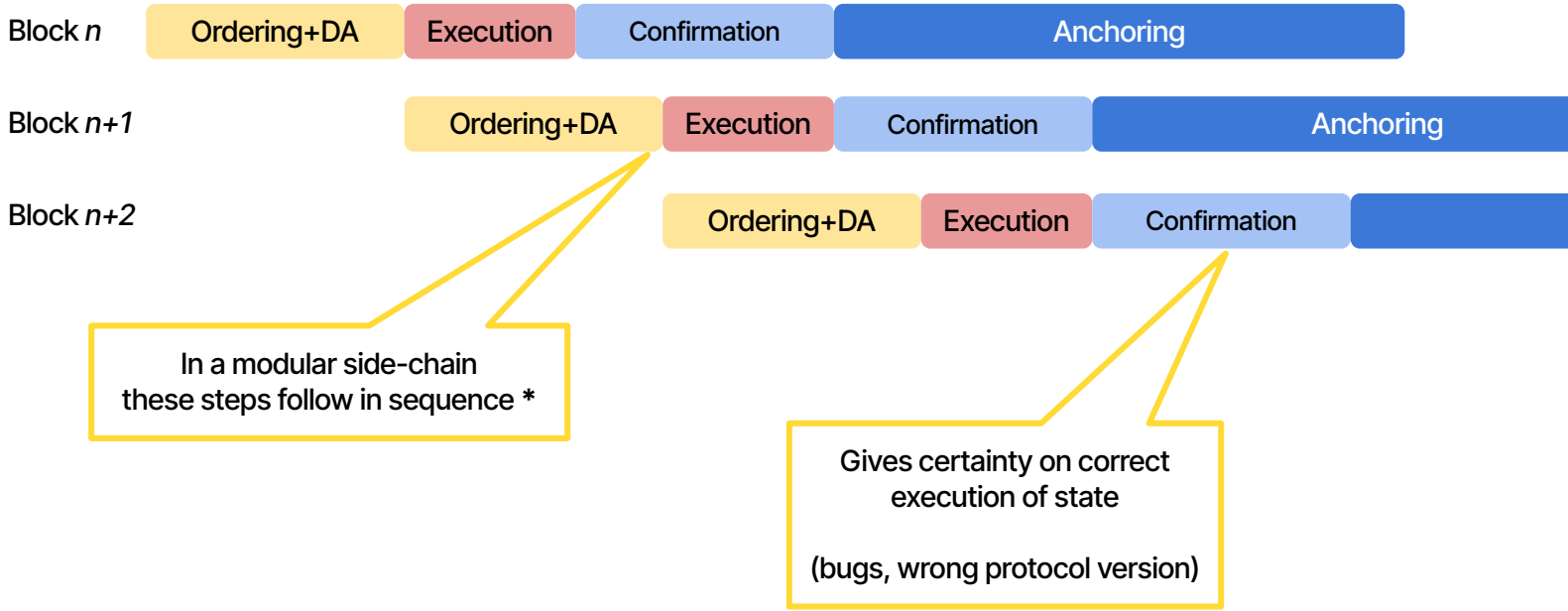
Move Stack Side Chain

Transaction Lifecycle



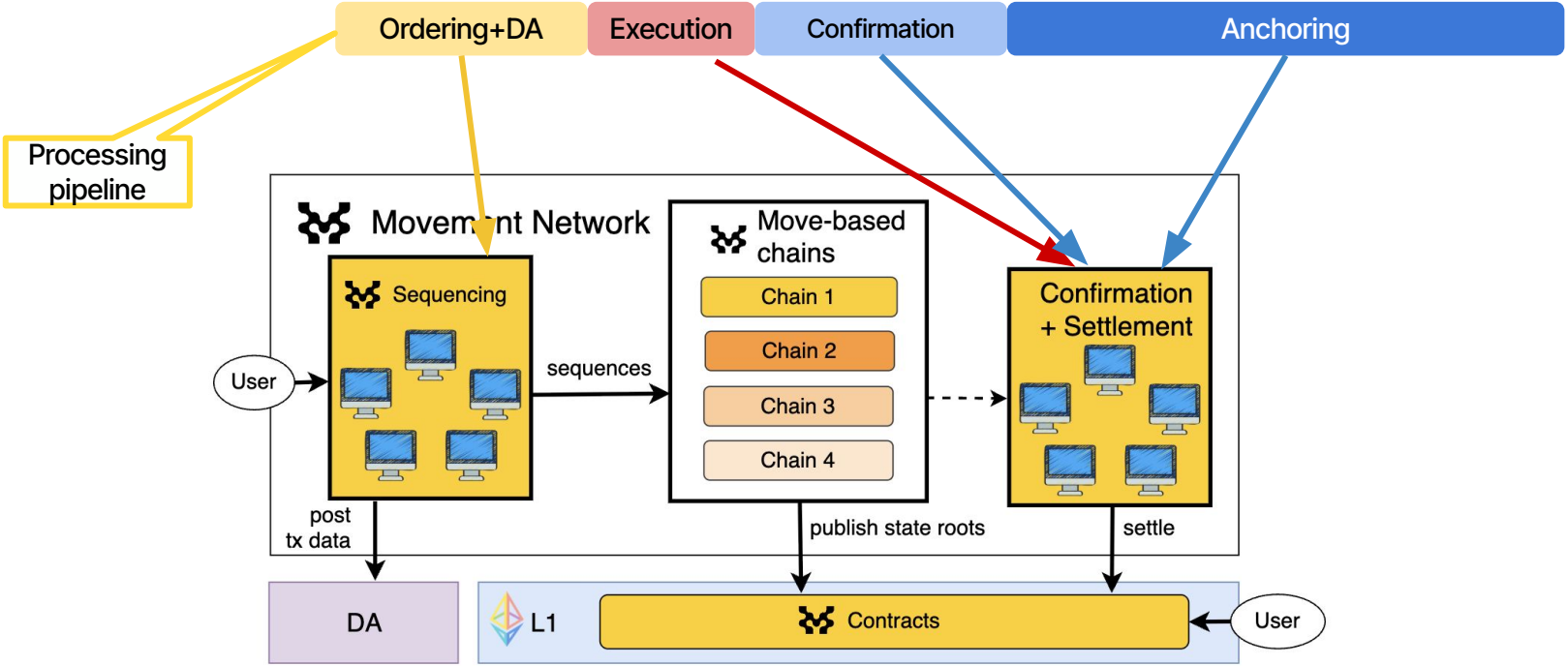
- Permissioned sequencer
- Consensus and DA via Celestia
- Confirmation Layer and Anchoring via Fast Finality Settlement

Processing steps of transactions

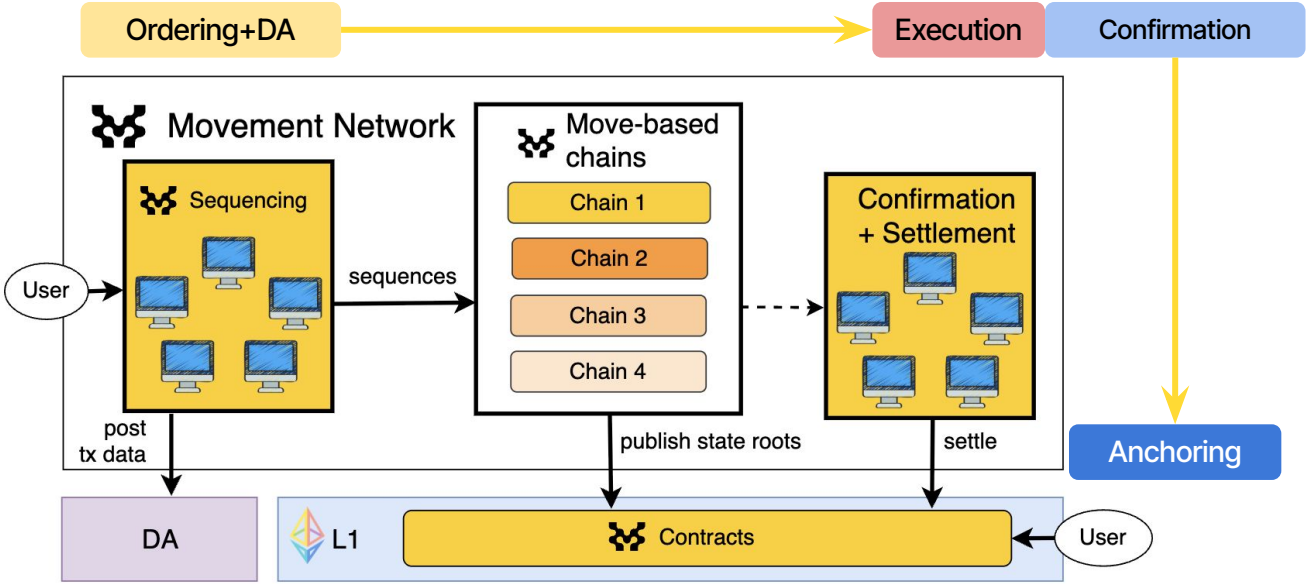


* Zaptos (https://arxiv.org/abs/2501.10612) shows we may parallelize some of these steps to achieve lower latency

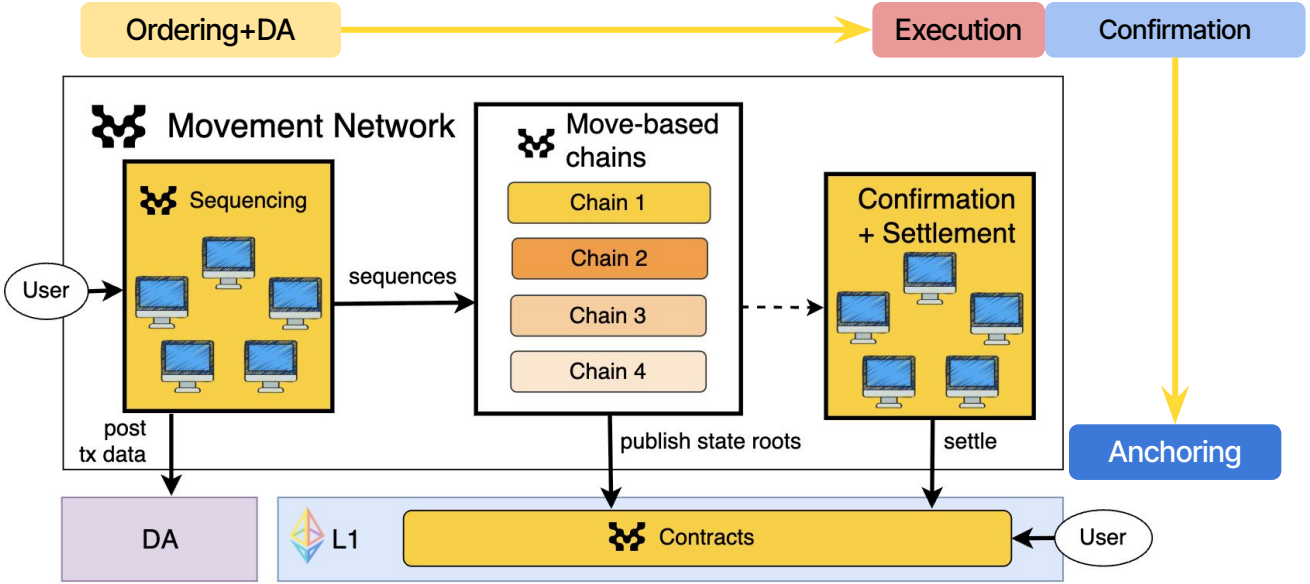
Modular Cluster Design



Modular Cluster Design

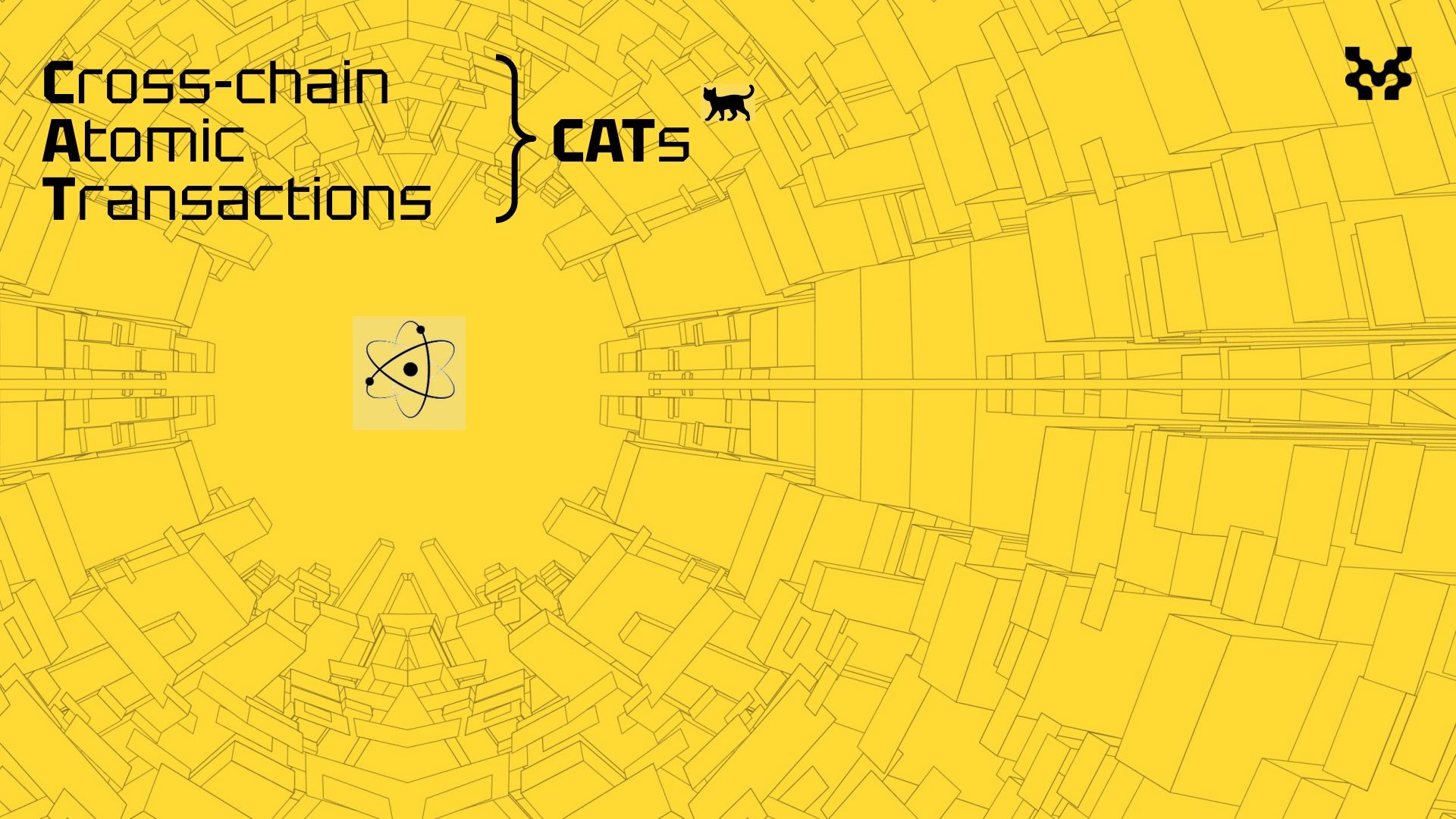


Modular Cluster Design



- **App-Specific Chains:** customized features, while sharing liquidity and infrastructure.
- **Cost-Effective & Scalable:** deploy new chains with reduced costs, leveraging a modular design and compatibility with common L2 approaches.
- **Interoperability:** through shared data availability, fast settlement and (some degree of) sharing of sequencing rights

Cross-chain Atomic Transactions } CATs

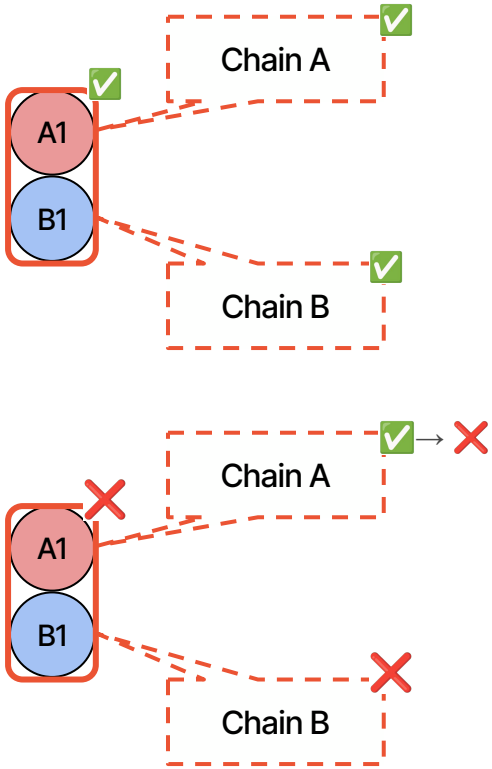


Challenges

with modular multi-chain ecosystems

- **Fragmented Liquidity**
  Capital is split across chains, reducing market depth and hurting DeFi efficiency.
- **Capital inefficiency**:
  Protocols must replicate logic for liquidity across chains.
- **Developer overhead**:
  Maintaining multi-chain support adds major complexity.
- **Cross-Chain Latency**:
  Asynchronous transfers and interactions are slow, breaking real-time composability.
- **No Atomicity**:
  Multi-chain actions can't execute all-or-nothing, exposing users to failure and loss.
- **Trust in external bridges:**
  Frequently bridges from ecosystem-foreign parties must be employed for transfers.

Solution



CATs Generic composable atomic actions on multiple chains

Multiple chains apply state transitions that should happen together or not at all.

Similarity to Burn and Mint bridge

Bidirectional implications

Contagion of weakest link

Systemization of Knowledge: Cross-chain Token Bridges and Risk
Uri Lee (Imperial College London, United Kingdom)

Dependency graphs

in heterogeneous chain-systems

- **Latency and liveness**

  Better predict finalities and identify least-latency impacting ordering.

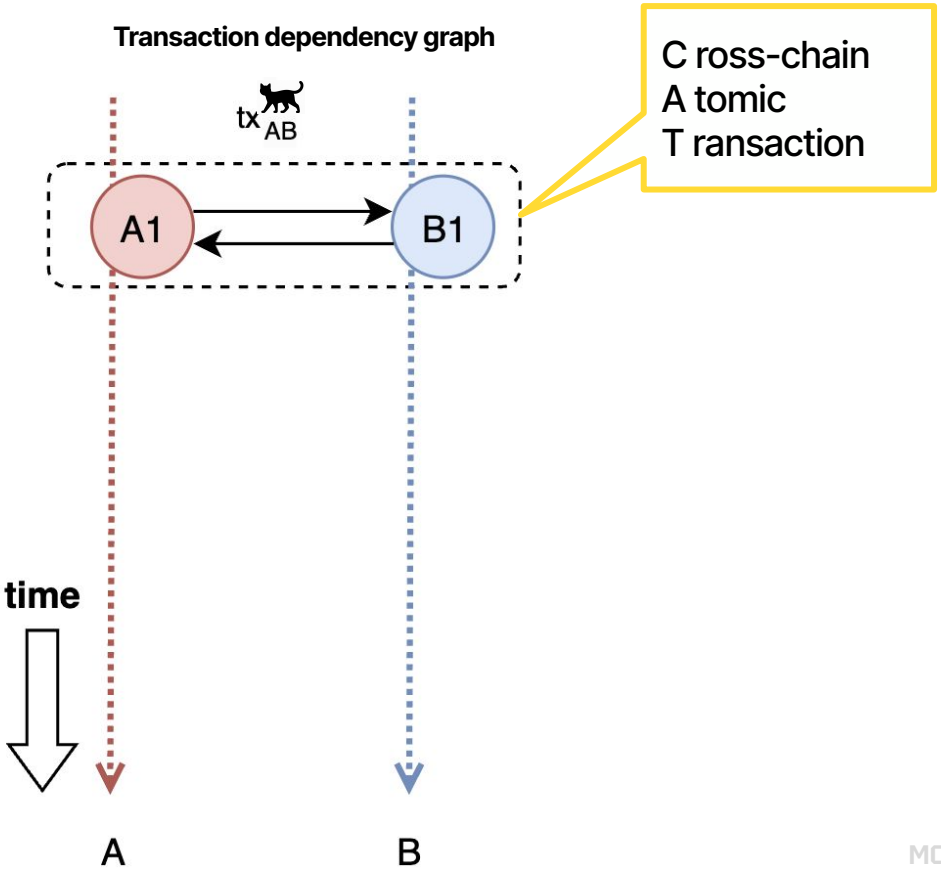
- **Safety**

  Identifies security risks by showing how security spreads across chains.

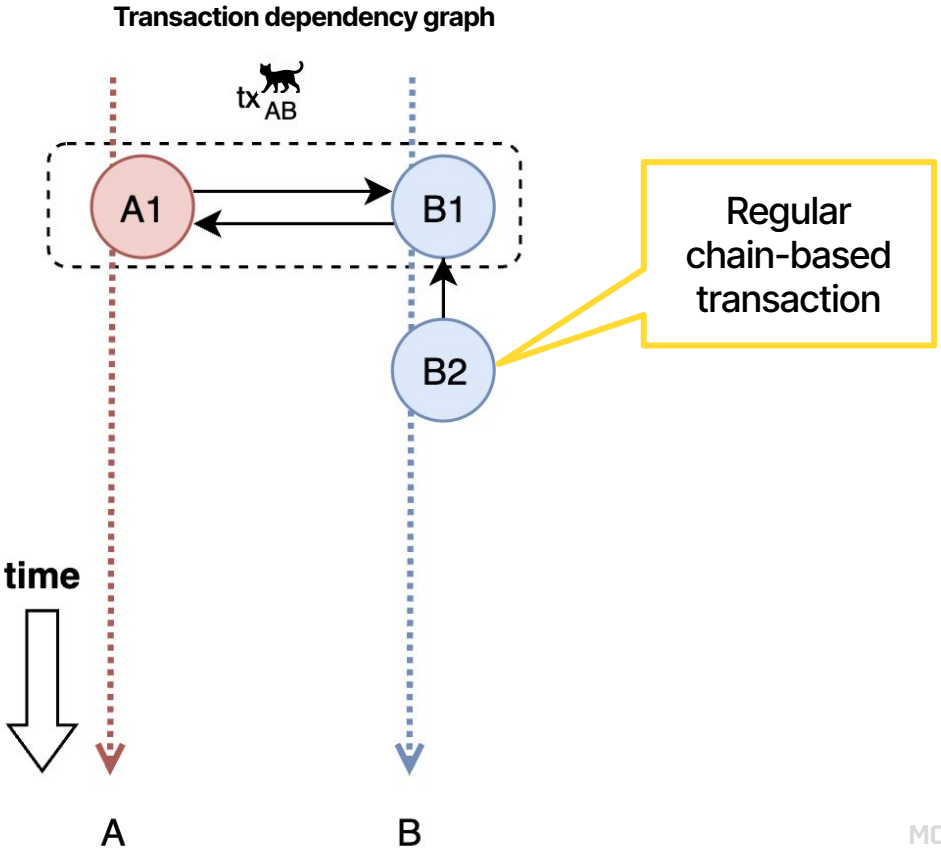
  Improved planning for joining interoperable chains.

Why should dependency graphs be considered?

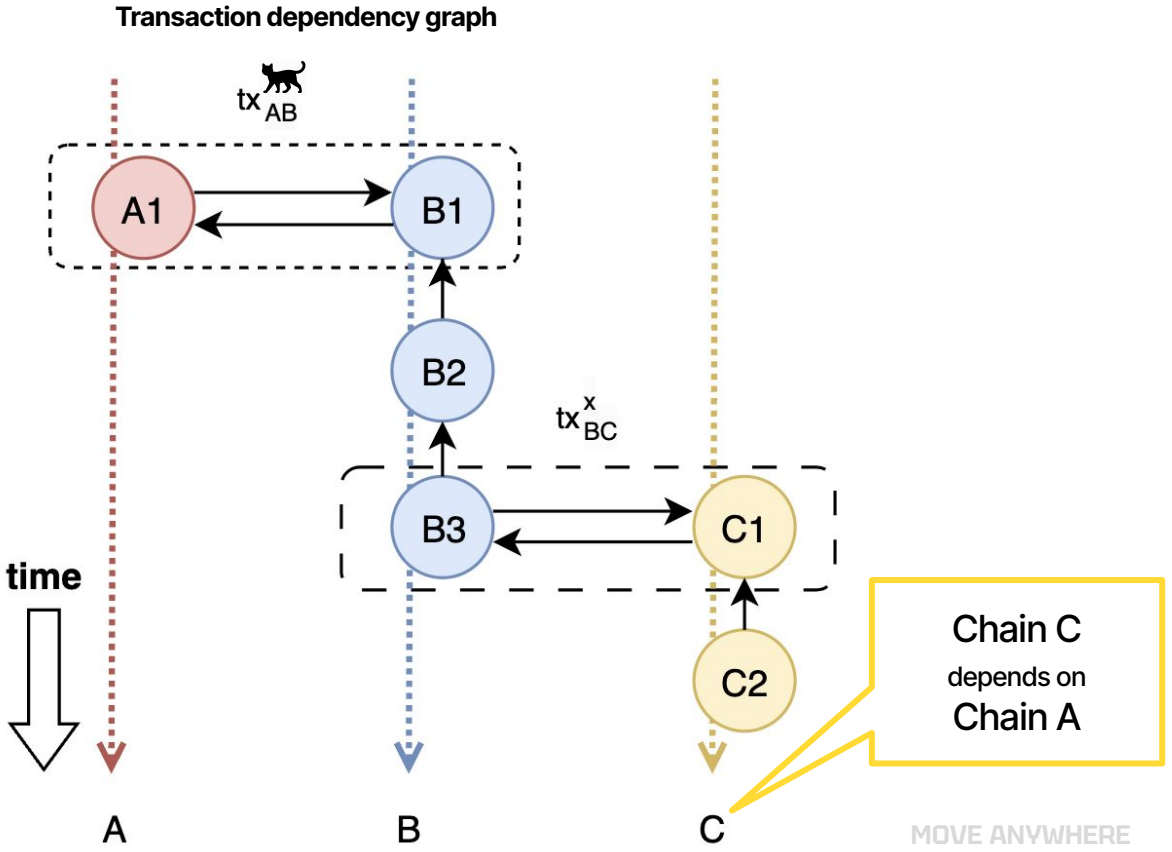
Transitive transaction dependencies



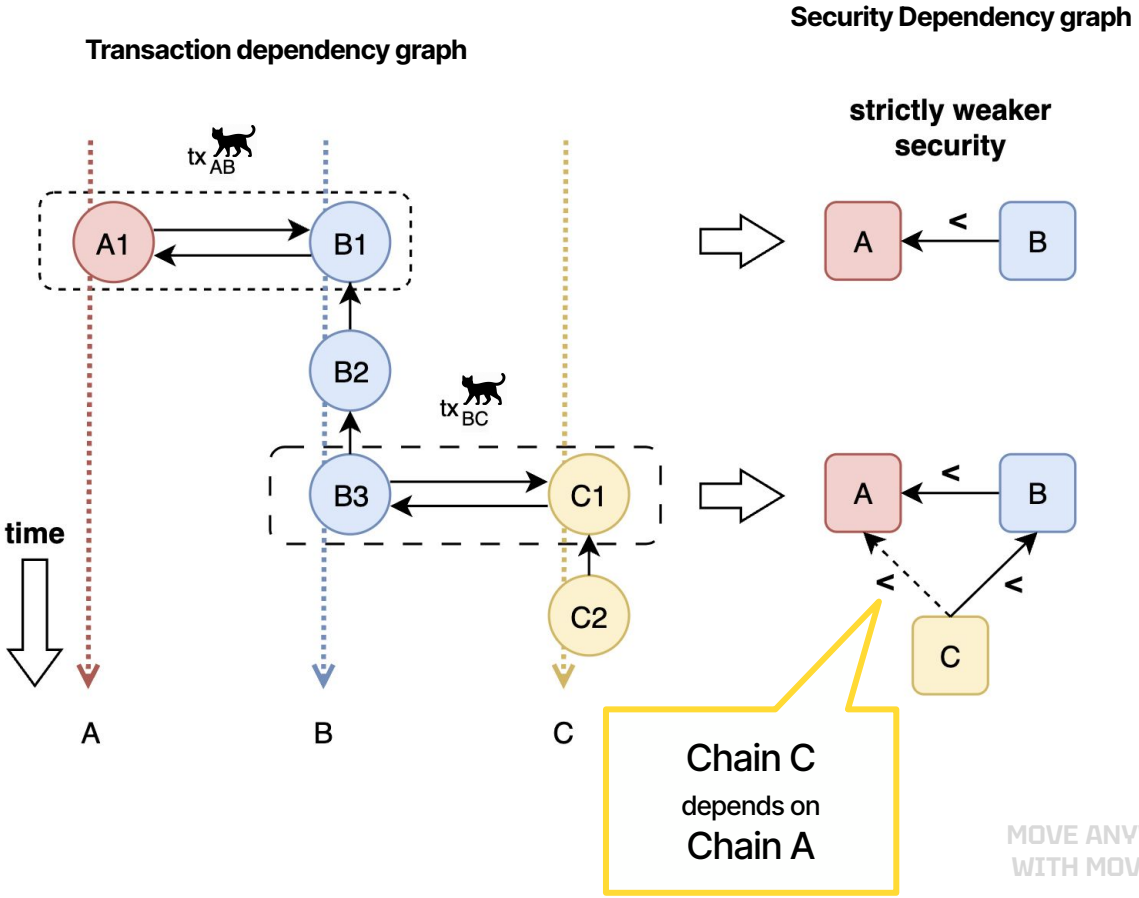
Transitive transaction dependencies



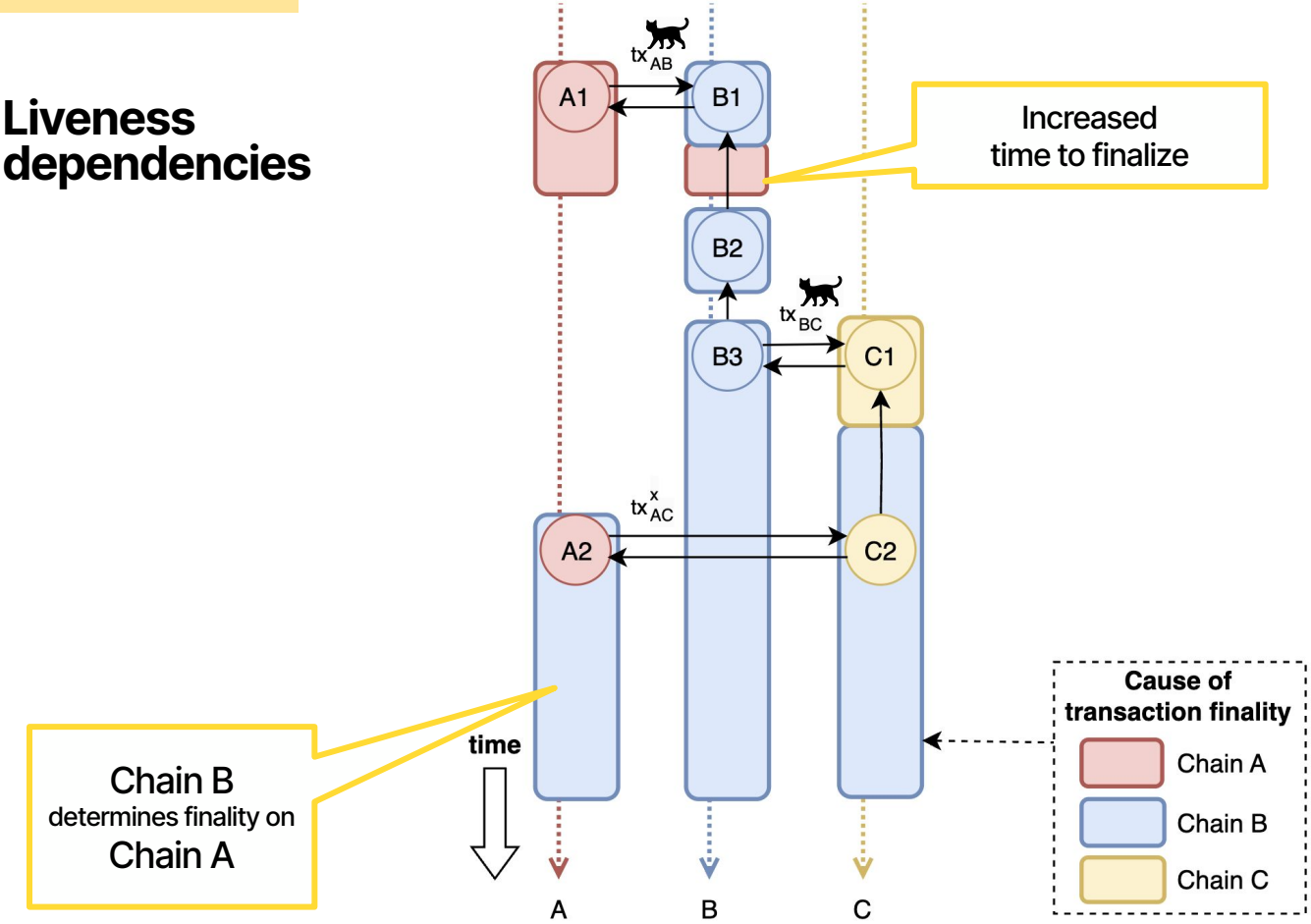
Transitive transaction dependencies



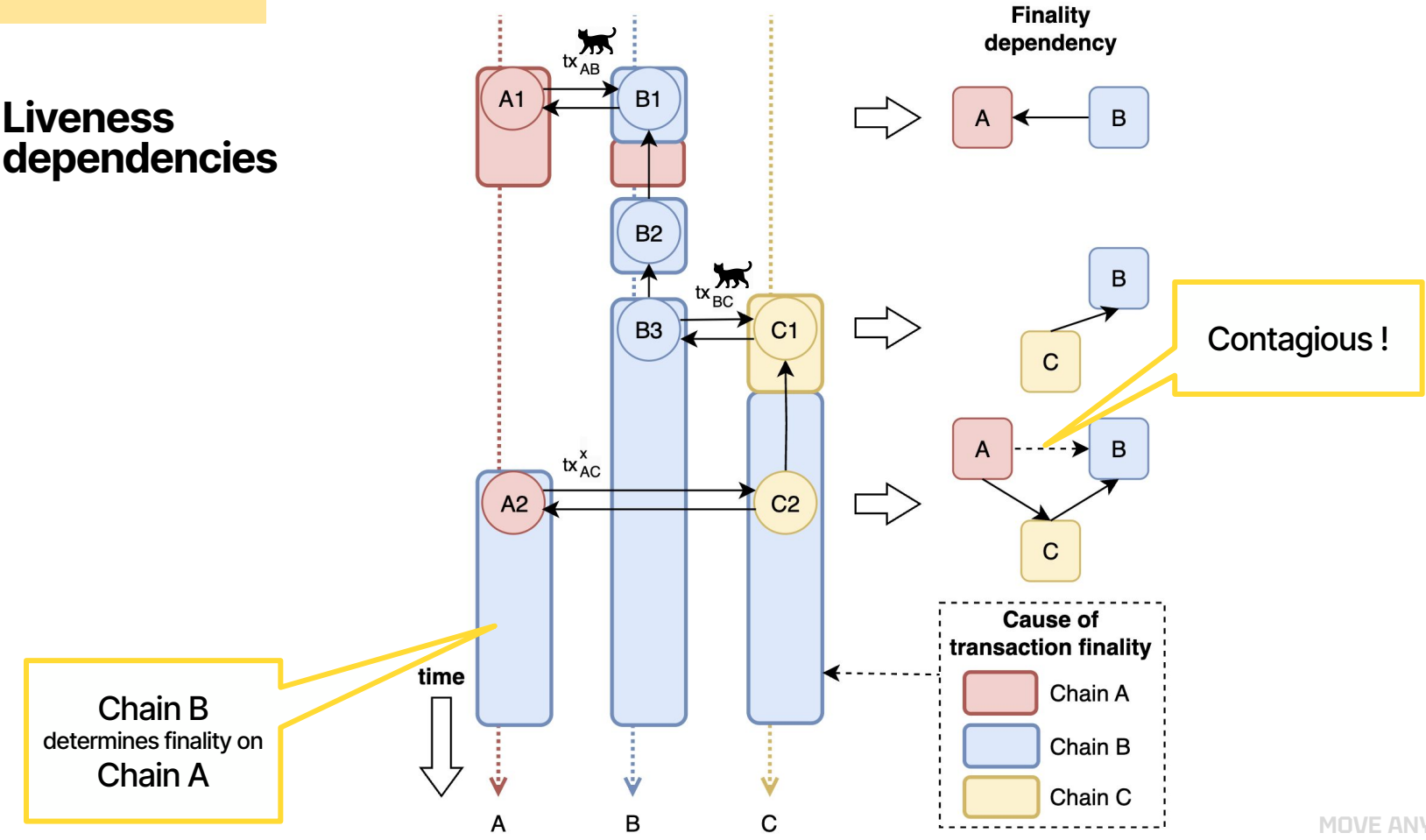
Safety dependencies



Liveness dependencies

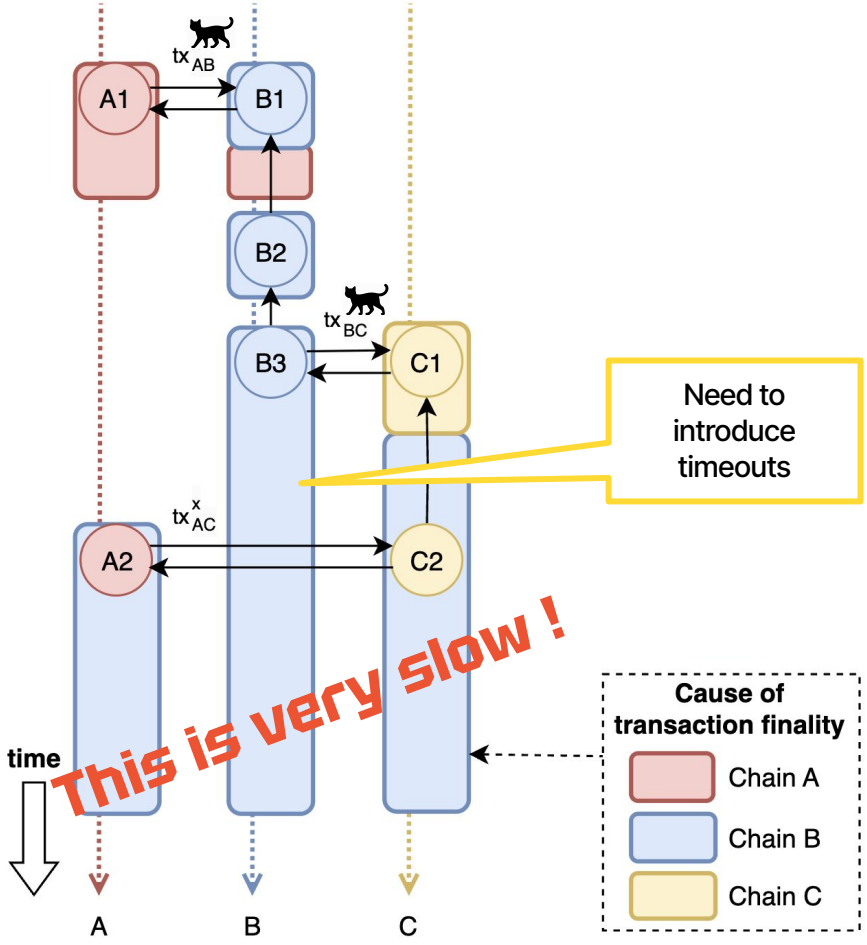


Liveness dependencies



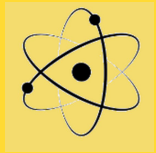
Latency:

Slow Chain !

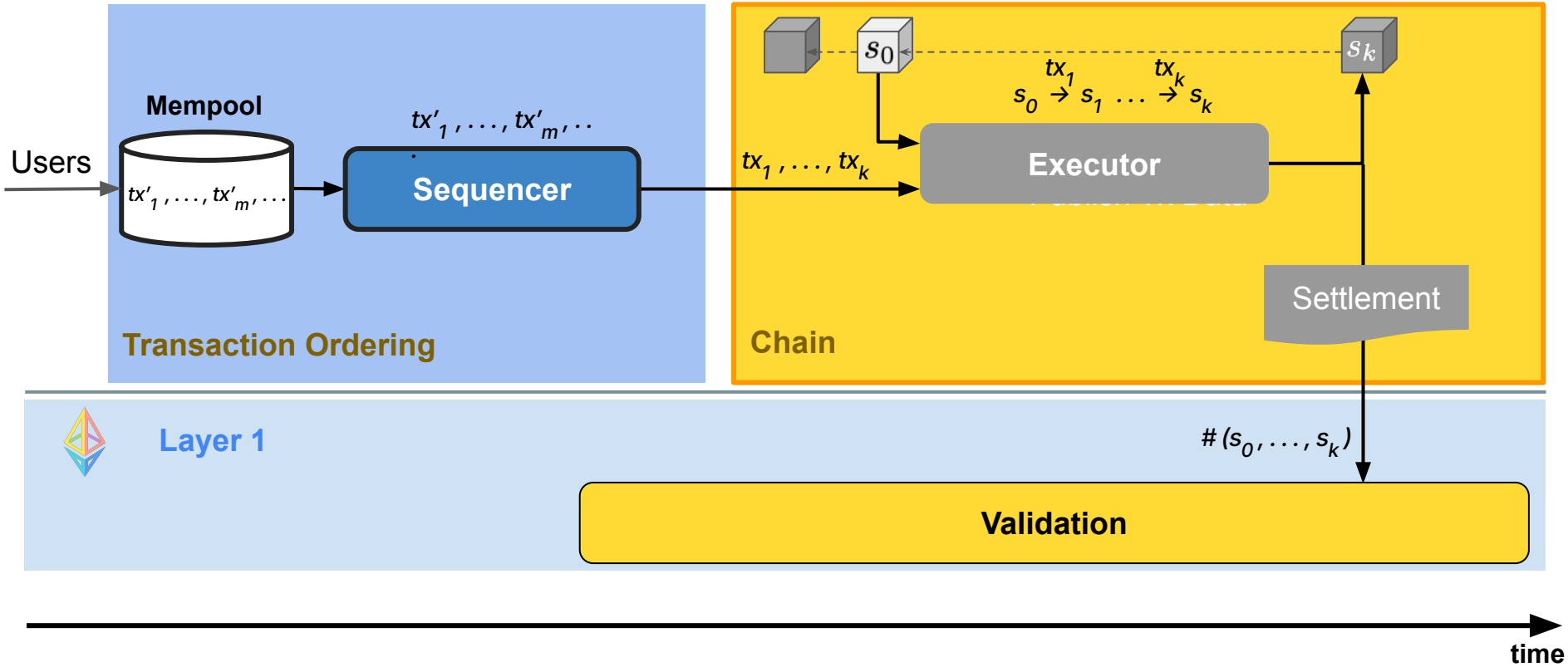


Interoperability solution

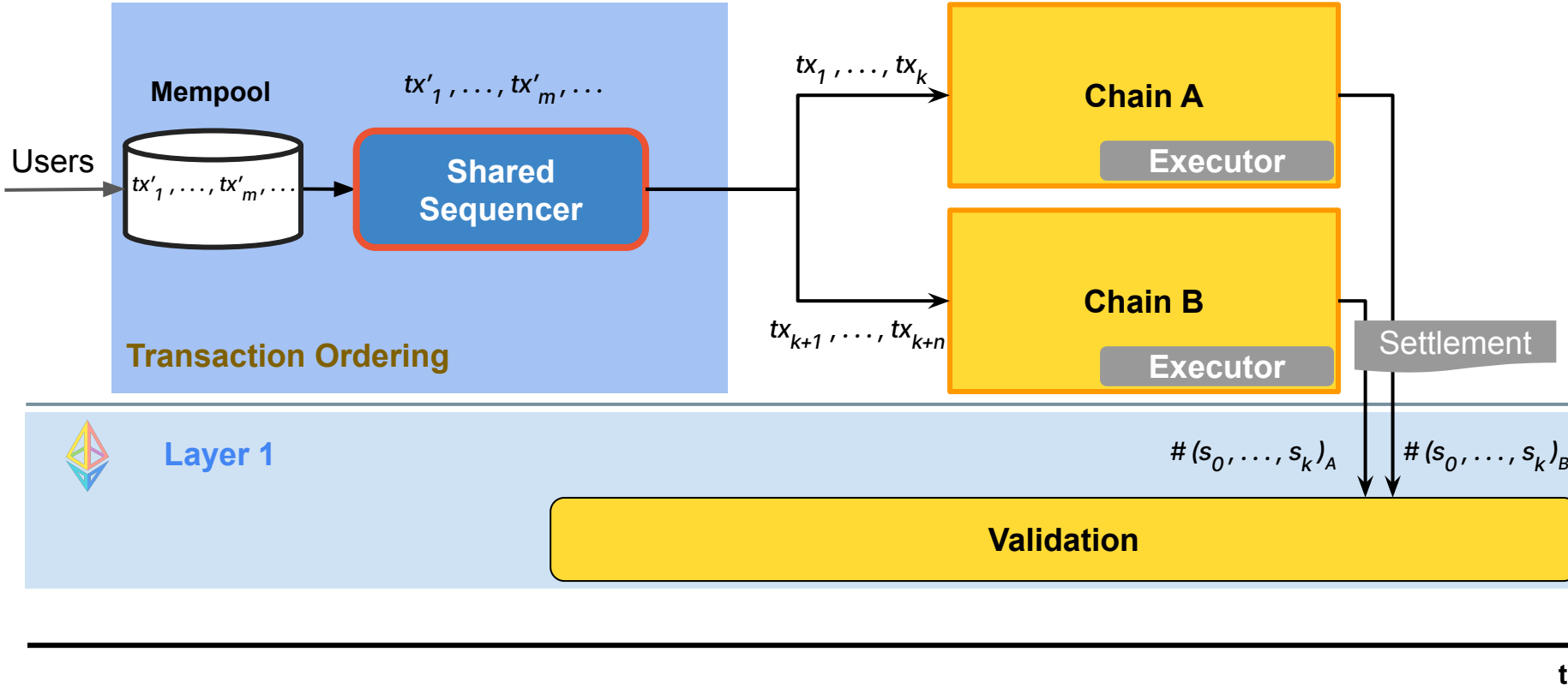
for side chains and L2s

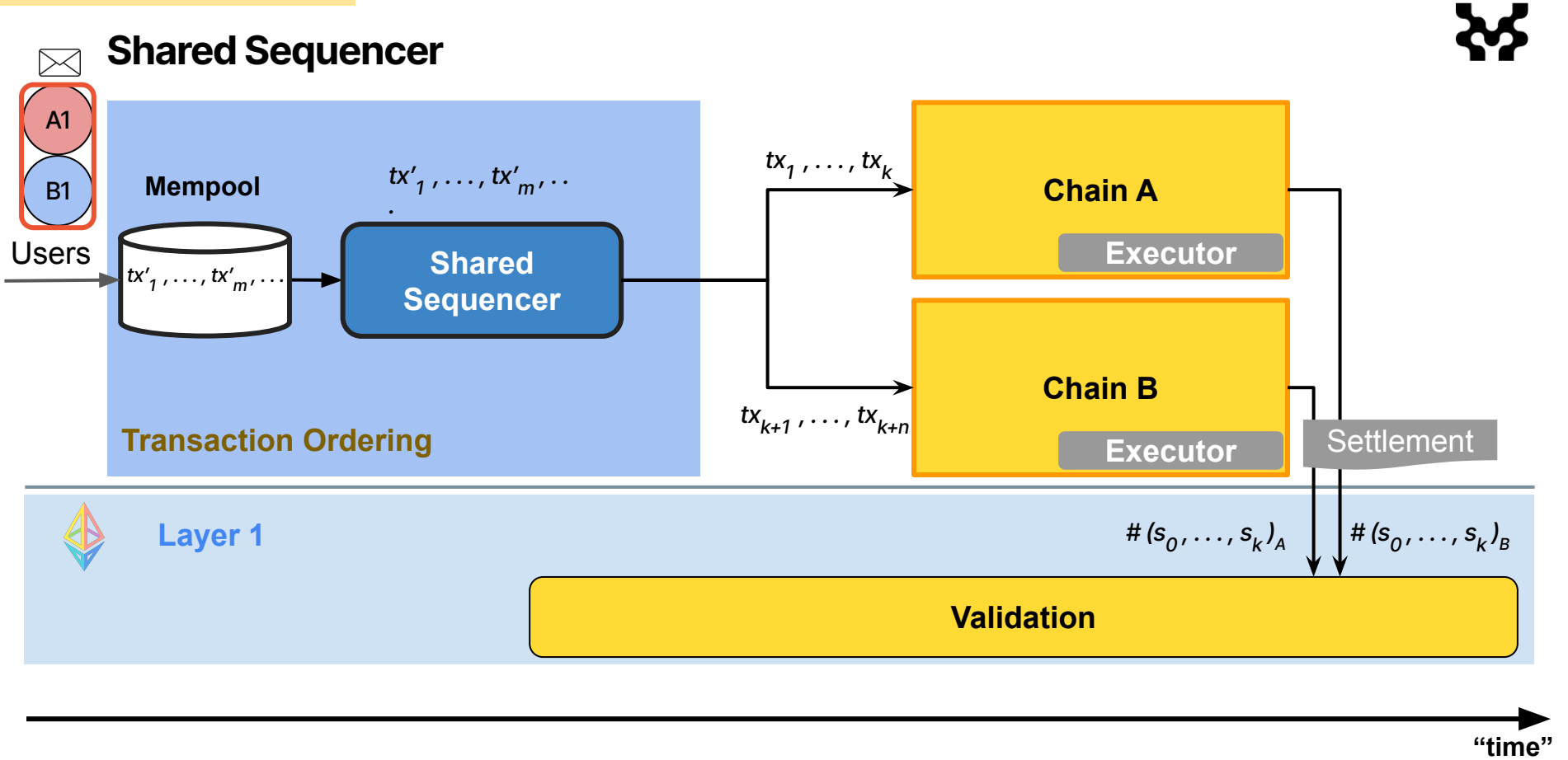


Move Stack Chain

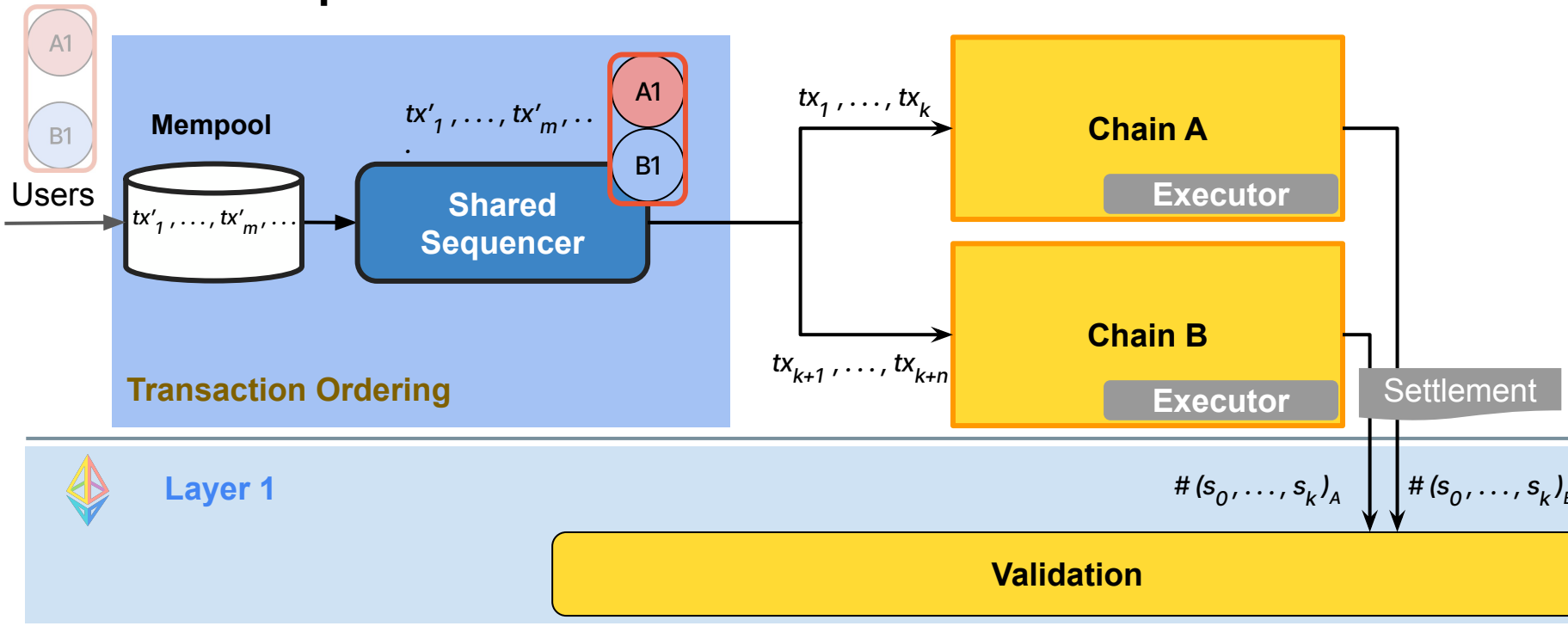


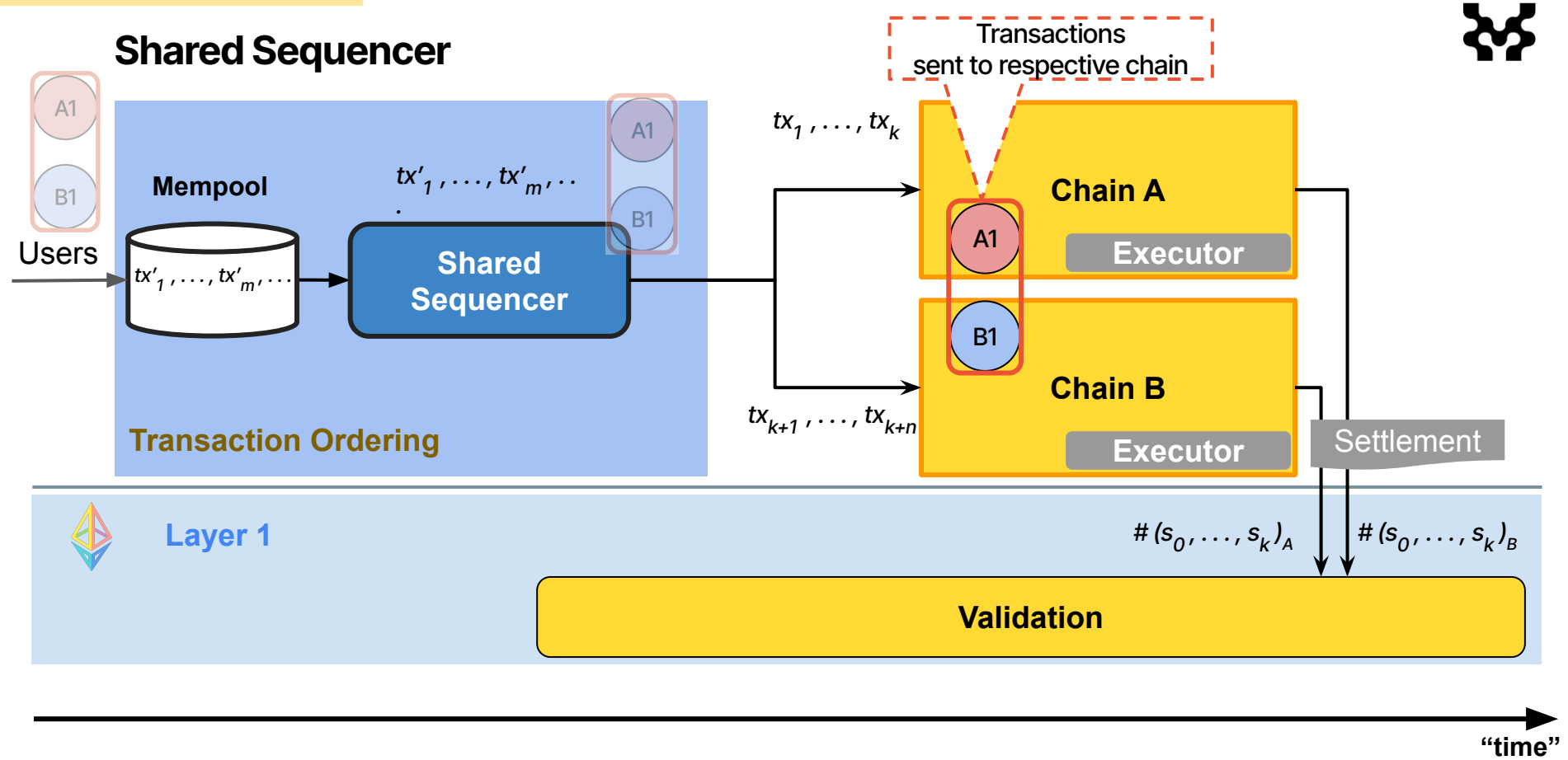
Shared Sequencer




Shared Sequencer
A1
B1
Users
Mempool
$tx'_1, \ldots, tx'_m, \ldots$
$tx'_1, \ldots, tx'_m, \ldots$
Shared Sequencer
Transaction Ordering
$tx_1, \ldots, tx_k$
$tx_{k+1}, \ldots, tx_{k+n}$
Chain A
Executor
Chain B
Executor
Settlement
Layer 1
$\#(s_0, \ldots, s_k)_A$
$\#(s_0, \ldots, s_k)_B$
Validation
"time"

Shared Sequencer




Shared Sequencer
Transactions sent to respective chain
A1
B1
Users
Mempool
$tx'_1, \ldots, tx'_m, \ldots$
Shared Sequencer
A1
B1
Transaction Ordering
$tx_1, \ldots, tx_k$
Chain A
A1
Executor
B1
Chain B
Executor
$tx_{k+1}, \ldots, tx_{k+n}$
Settlement
Layer 1
$\#(s_0, \ldots, s_k)_A$
$\#(s_0, \ldots, s_k)_B$
Validation
"time"

Sequencer:

pre-execution

vs

inclusion-only

Types of Sequencers

with execution**:**

- Sequencer simulates all transactions (powerful builder)
- Should **only** submit **valid transactions**
- requires chain state awareness
- increased centralization risks.
- **does not scale well !**

without execution**:**

- sequencer only orders transactions
- separation of concerns
- simplifies trust but needs **frequent synchronization** across chains.

Sequencer:

pre-execution

vs

inclusion-only

Coordination requirement :
Trusted third party *

Types of Sequencers

with execution:

- Sequencer simulates all transactions (powerful builder)
- Should **only** submit **valid transactions**
- requires chain state awareness
- increased centralization risks.
- **does not scale well !**

without execution:

- sequencer only orders transactions
- separation of concerns
- simplifies trust but needs **frequent synchronization** across chains.

Execution Approaches

optimistic execution:

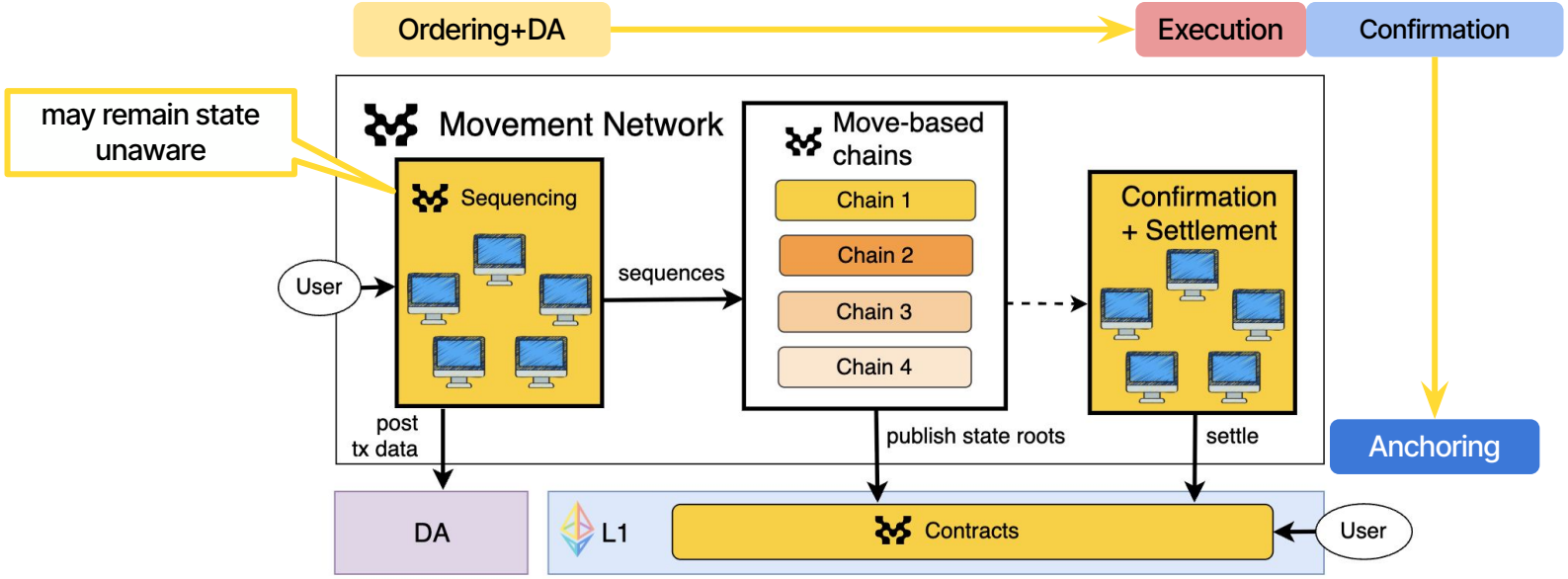
- applies transactions immediately, risks rollbacks if dependencies fail, may slow down throughput

simulation, but execution after synchronization**:**

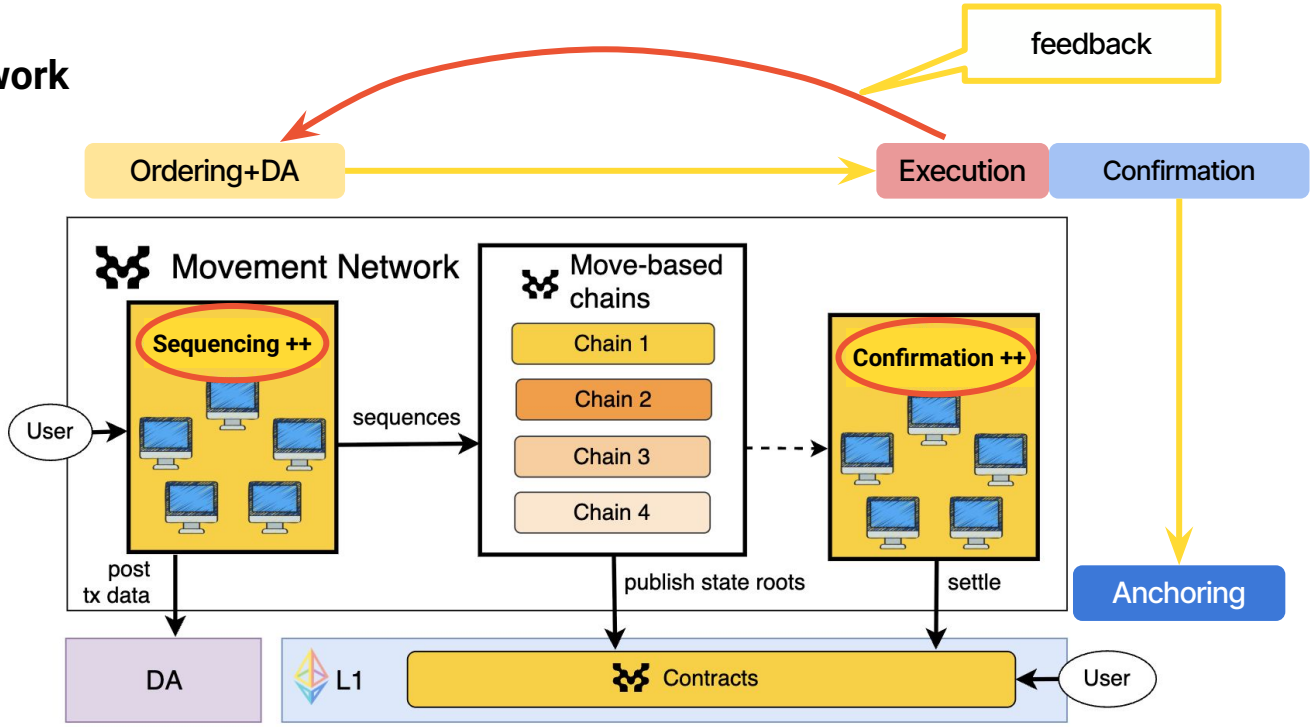
- Waits for transaction confirmation - **locks state** until success
- ensures consistency but adds complexity and latency

* “Cross-blockchain transactions are not feasible in practice without the participation of a trusted third party”
Rafael Belchior et al : A Survey on Blockchain Interoperability: Past, Present, and Future Trends

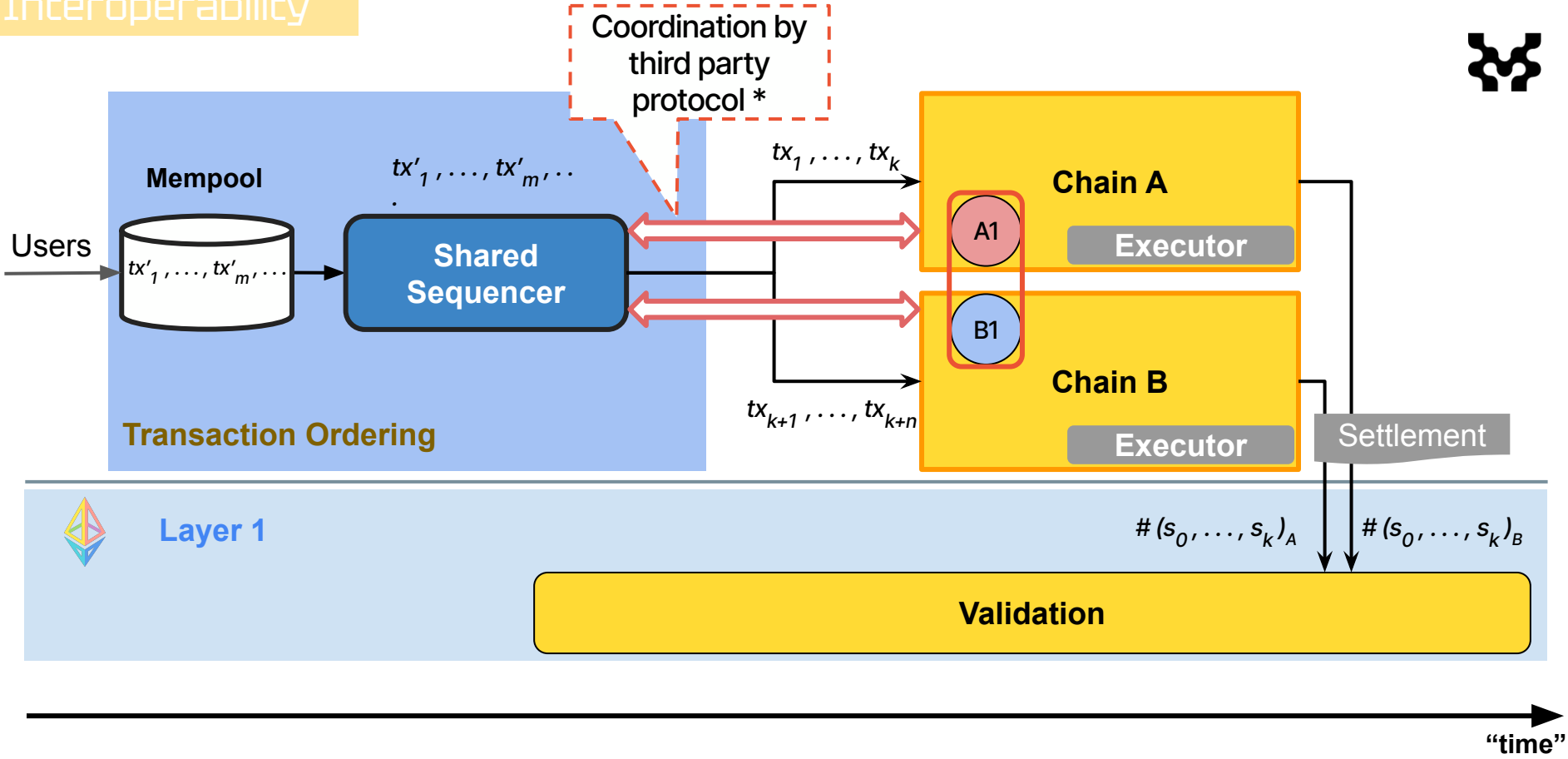
Move Stack Chain
in the
Movement Network



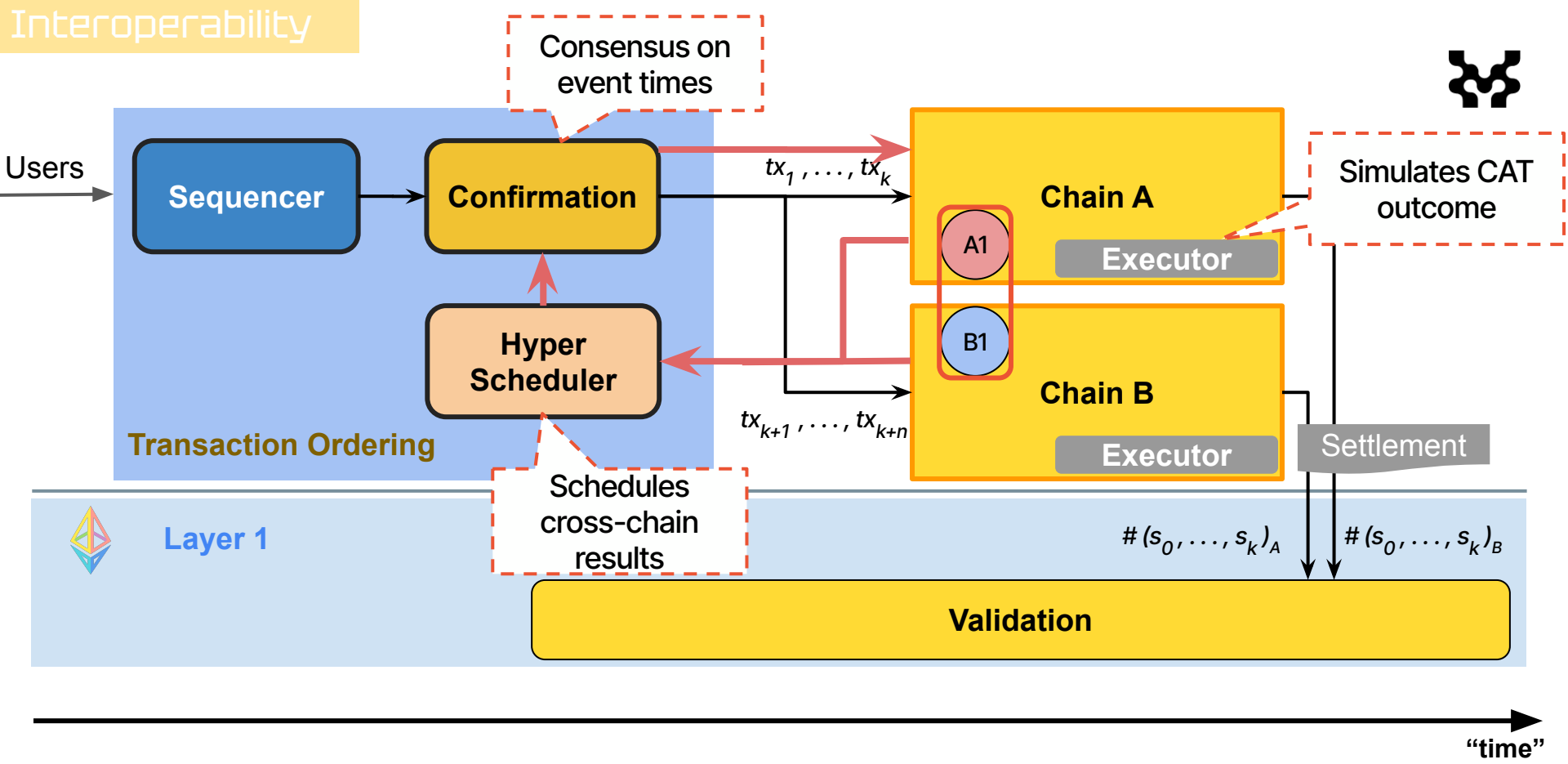
Move Stack Chain
in the
Movement Network

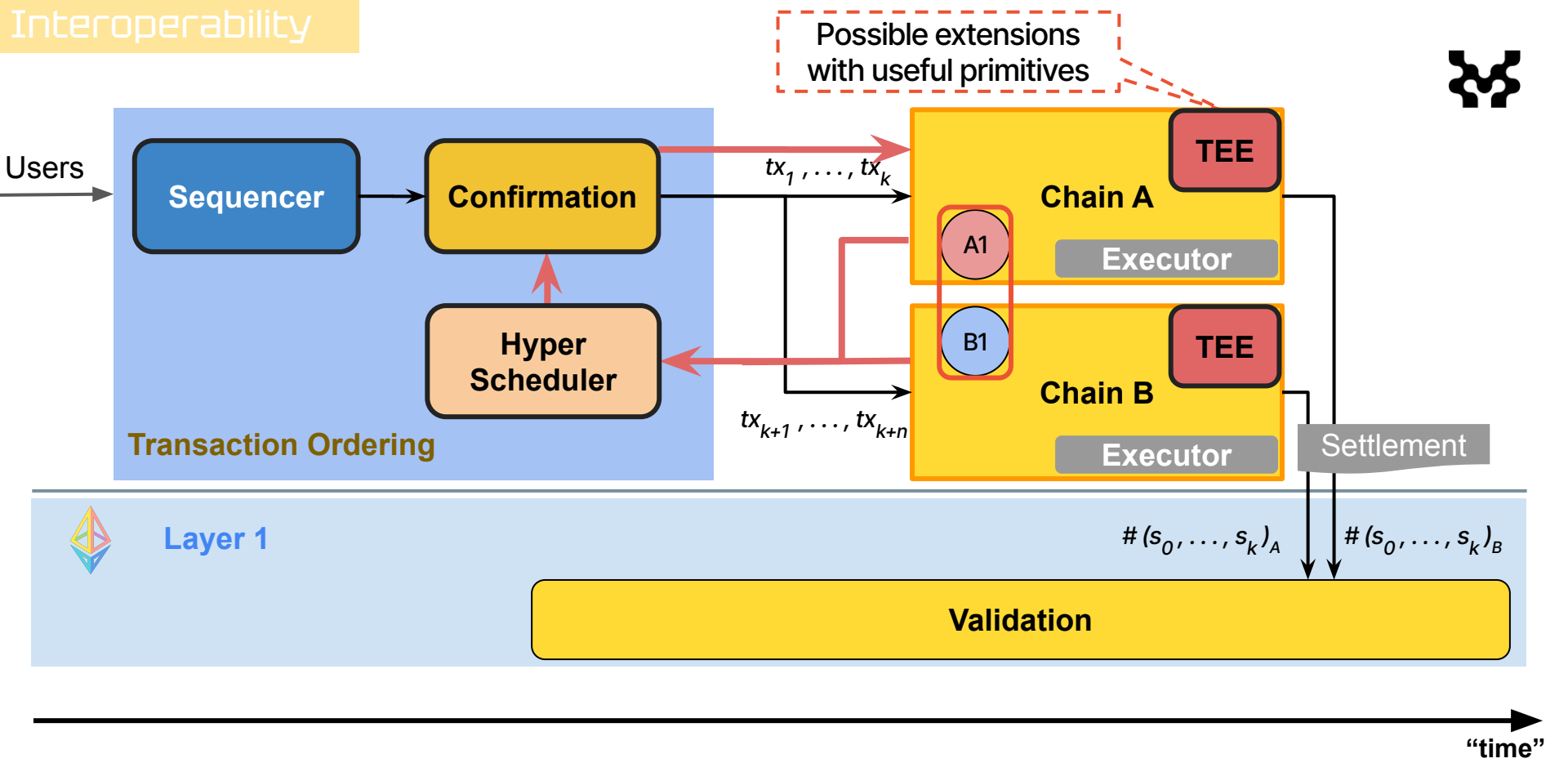


Interoperability



* for example, Yuandi Cai et. al (Huazhong University): Atomicity for cross-chain applications through layered state commitment




Possible extensions
with useful primitives
Users
Sequencer
Confirmation
Hyper
Scheduler
Transaction Ordering
$tx_1, \ldots, tx_k$
$tx_{k+1}, \ldots, tx_{k+n}$
TEE
Chain A
A1
Executor
B1
TEE
Chain B
Executor
Settlement
Layer 1
$\#(s_0, \ldots, s_k)_A$
$\#(s_0, \ldots, s_k)_B$
Validation
"time"

Thank You

Movement