China Blockchain Conference

A Fast and Scalable DAG-Based Consensus with an Underlying Backbone Chain

Zhiqiang Liu

Shanghai Jiao Tong University

November 25, 2018

Background

DAG Ledger

- IOTA
- Byteball
- Hashgraph

Our Thinking and Work

- Two-Layer Structure
- Instantiation
- DAG Formalization in a Nutshell
- Brief Idea of Our Security Analysis

Background

Great potentials of distributed ledger

- Online payment
- Transactions of digital assets
- Smart digital contracts
- ...

Scalability Bottleneck of Blockchain

- Bitcoin: 7 tps
- Ethereum: 15 tps
- Visa: 56,000 tps (2015)
- Alipay: 256,000 tps (11.11, 2017)

Straightforward Approaches to Solve the Scalability Bottleneck

- Increase Block Size $\rightarrow$ High Throughput $\rightarrow$ Longer Propagation Time

Straightforward Approaches to Solve the Scalability Bottleneck

- Increase Block Size $\rightarrow$ High Throughput $\rightarrow$ Longer Propagation Time
- Decrease Block Interval $\rightarrow$ High Throughput $\rightarrow$ Instability from Forks

Do We Really Need a Block to Organize Transactions?

DAG Ledger

Using DAG as a Ledger

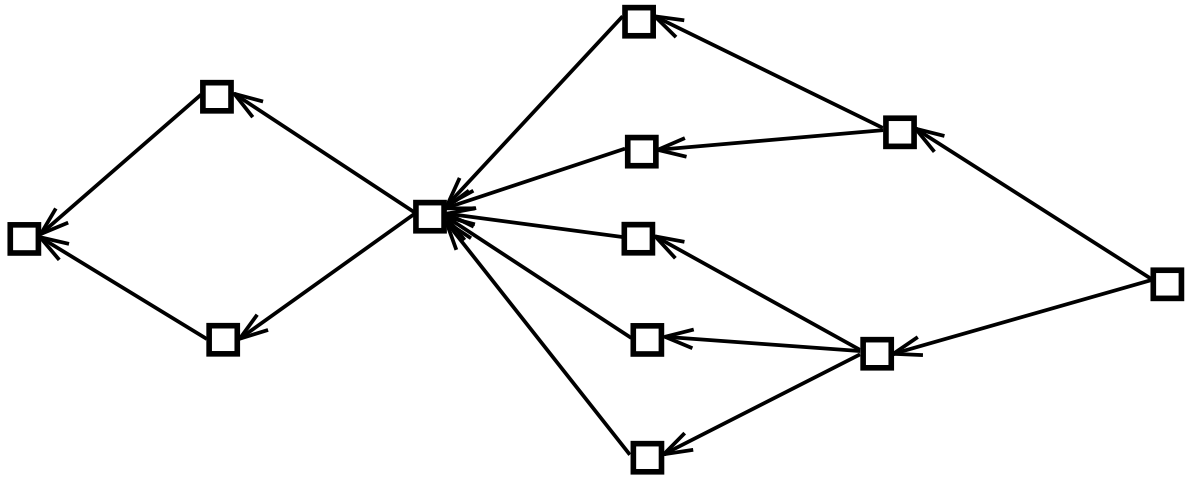


Figure: The Ledger of Transactions

- What if we organize transactions themselves into a *Dircted Acyclic Graph?*

Using DAG as a Ledger

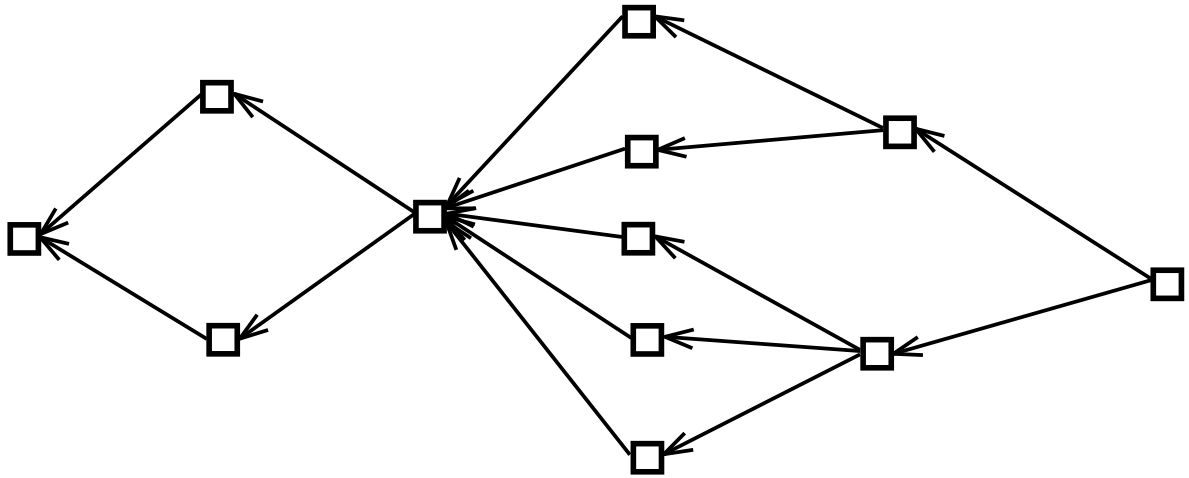


Figure: The Ledger of Transactions

- What if we organize transactions themselves into a *Dircted Acyclic Graph?*
- The virtue of DAG: concurrency

IOTA

IOTA

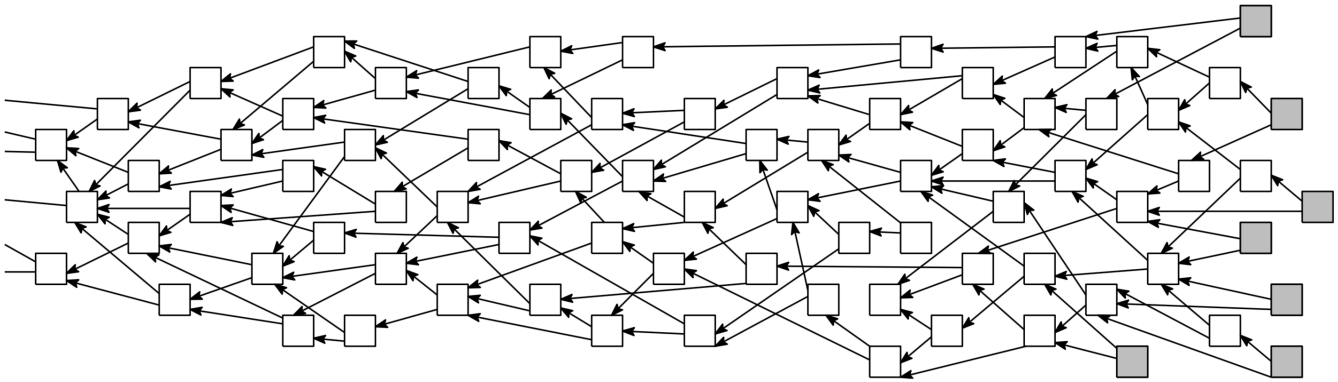


Figure: The Tangle

- Users must work to approve other transactions
- Each transaction contains a small PoW (weight)
- When the cumulative weight reaches the threshold, the transaction gets confirmed

IOTA

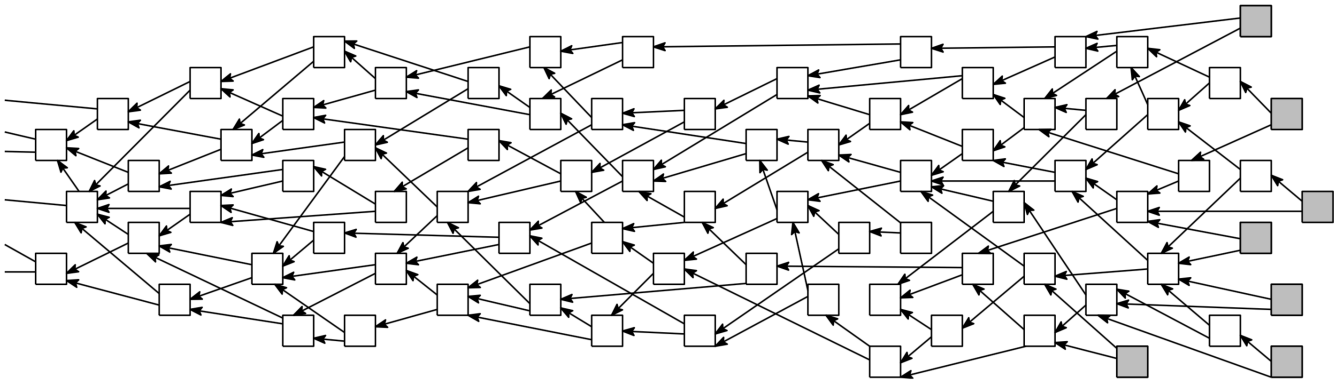


Figure: The Tangle

- No transaction fee
- High throughput
- Support for IoT

IOTA

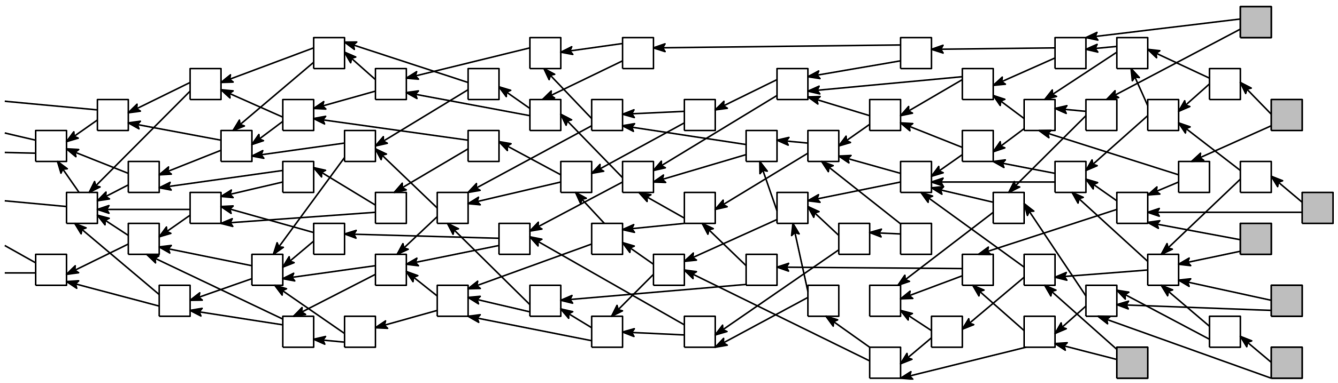


Figure: The Tangle

- Hard to determine a fixed threshold
- Security of the Tangle relies on high load
- Needs a Coordinator run by IOTA Foundation to issue periodic milestones to confirm validate transactions

Byteball

Byteball

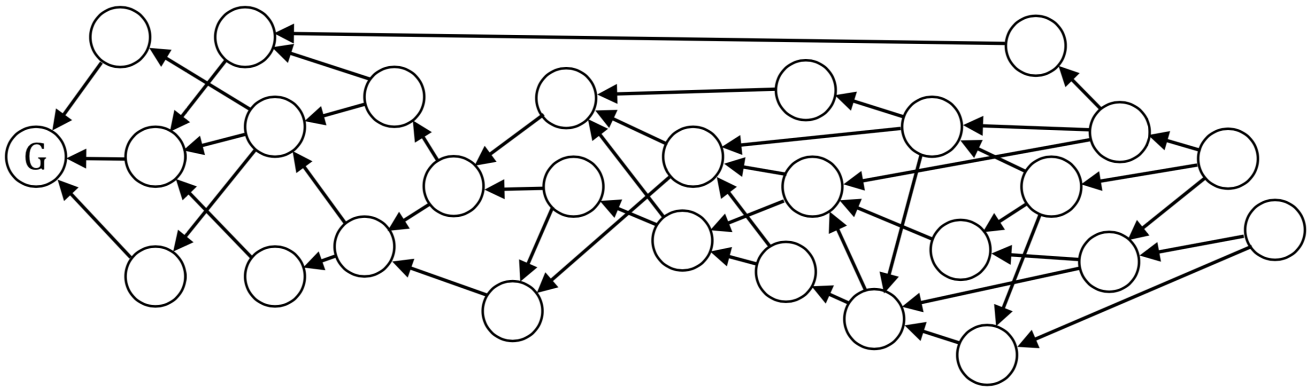


Figure: Byteball

- 12 witnesses to derive a main chain in the DAG
- Serialize all the transactions based on the main chain

Byteball

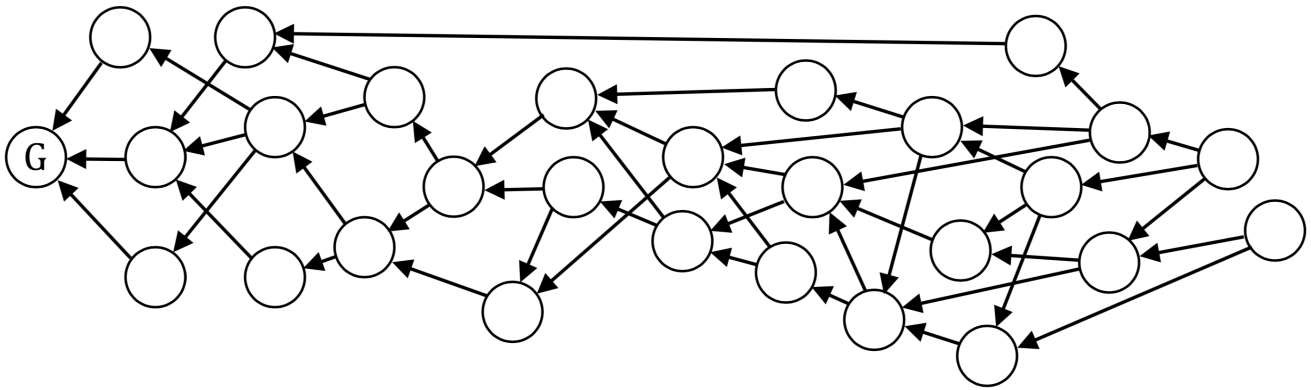


Figure: Byteball

- Multi-functions of units
- Very low transaction fees
- Deterministic transaction finality

Byteball

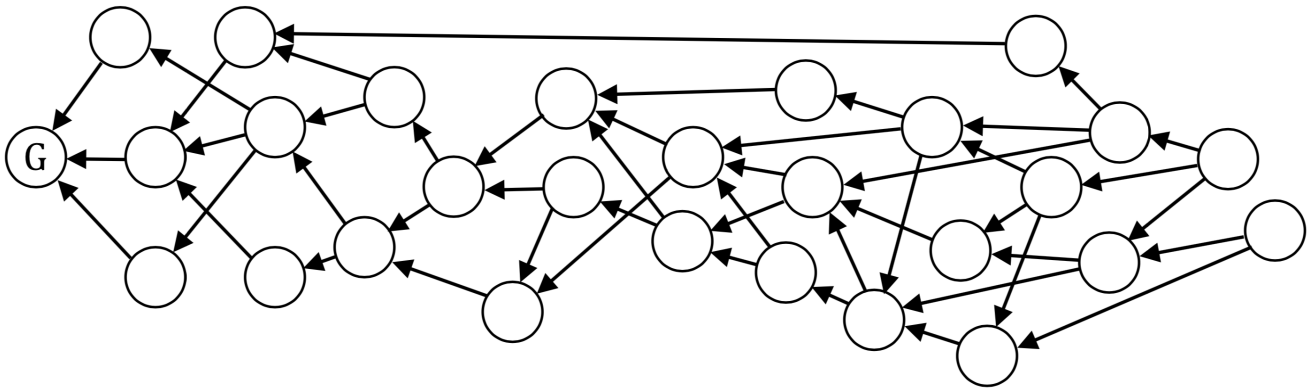


Figure: Byteball

- No security proof
- There exist bugs in its consensus algorithm
- Centralized in a certain way

Hashgraph

Hashgraph

- Record transactions and communication history with events and hashes
- Virtual voting to reach a consensus

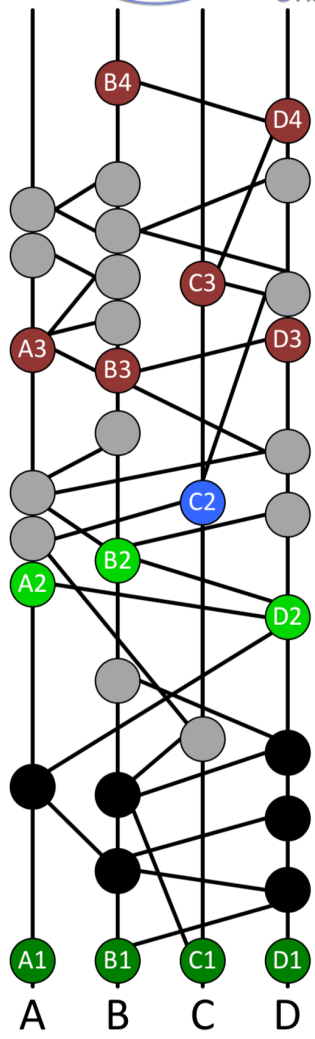


Figure: Hashgraph

Hashgraph

- Low computation (No PoW)
- Lower communication complexity (Leaderless BFT vs. PBFT)

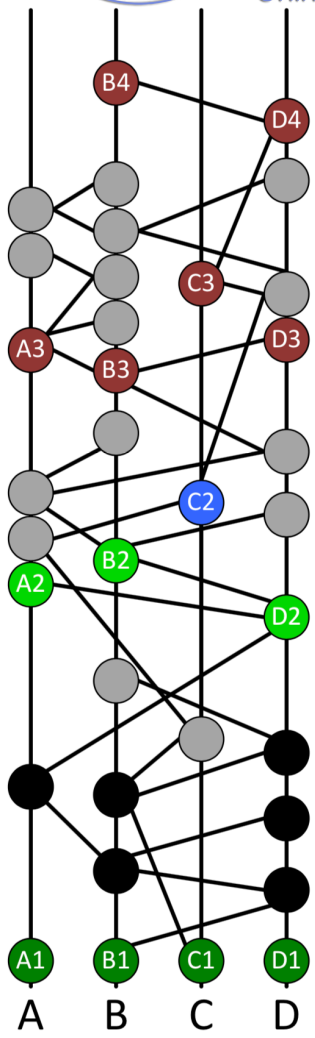


Figure: Hashgraph

Hashgraph

- 39 known and reputable nodes (real centralized)
- No support for dynamic join and leave

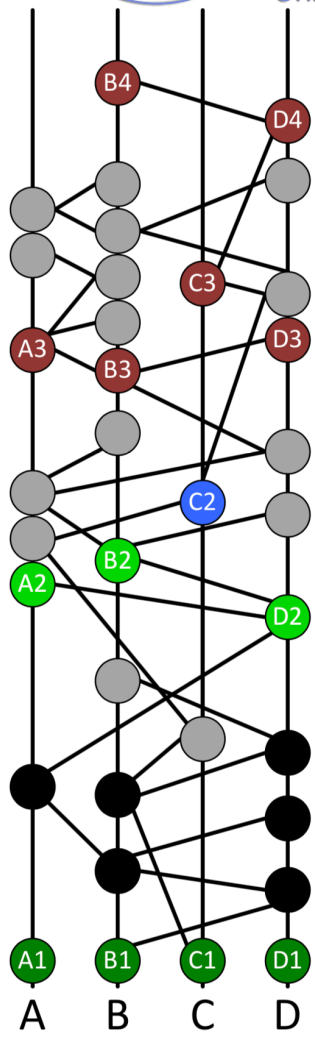


Figure: Hashgraph

Is There a Way to Realize a Fast Decentralized DAG Ledger?

Our Thinking and Work

DAG Ledger

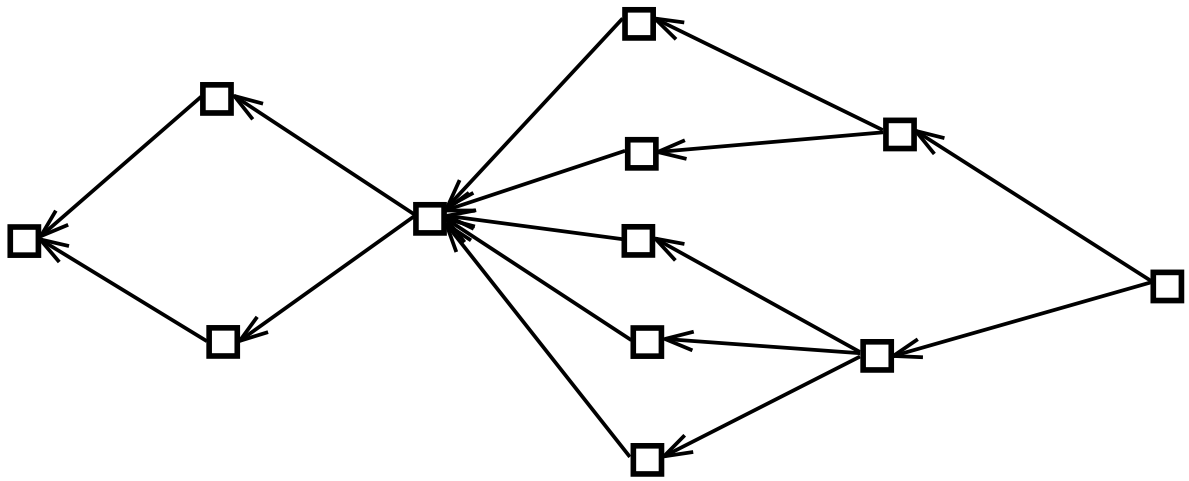


Figure: The Ledger of Transactions

- How to achieve consensus (transaction serialization) over DAG?

DAG Ledger

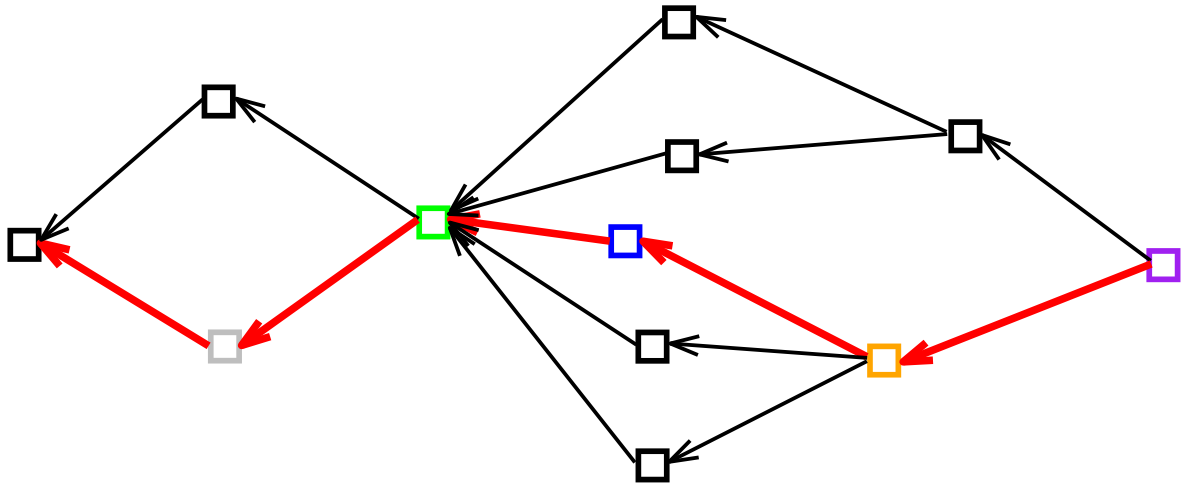


Figure: Main Chain

1. Find a *Main Chain* in the DAG

DAG Ledger

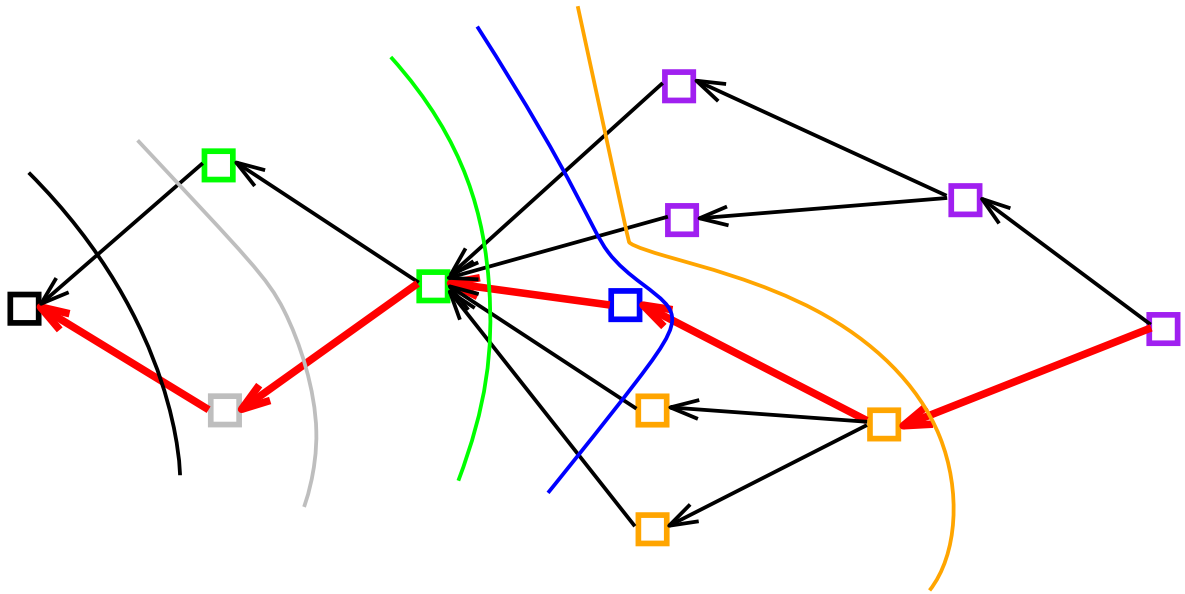


Figure: Rounds Division

1. Find a *Main Chain* in the DAG
2. Divide transactions into rounds

DAG Ledger

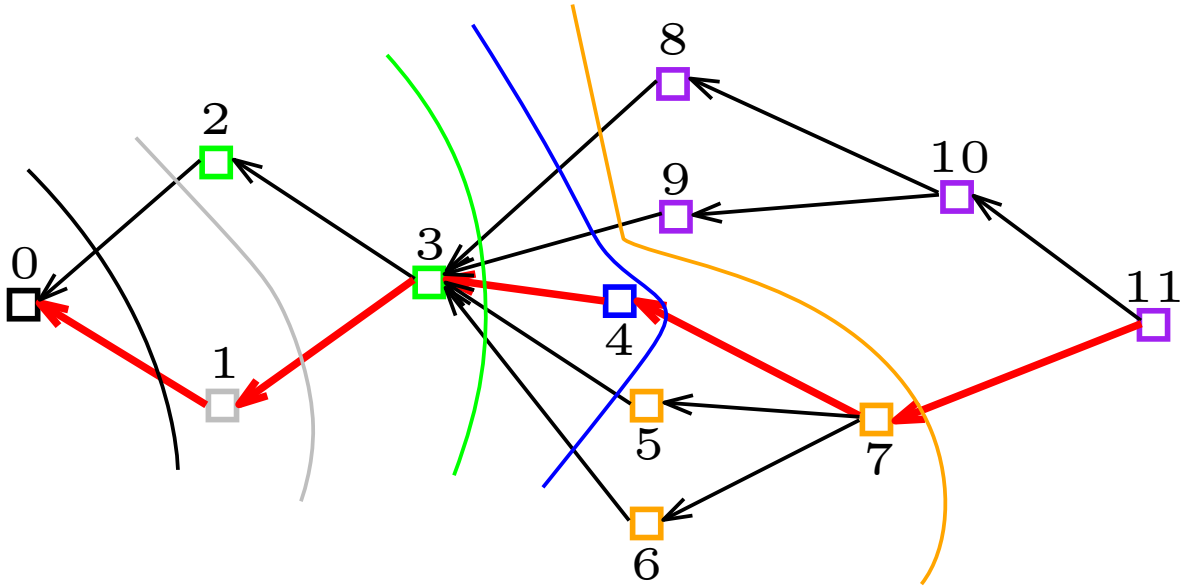


Figure: Transaction Serialization

1. Find a *Main Chain* in the DAG
2. Divide transactions into rounds
3. Transaction serialization

DAG Ledger

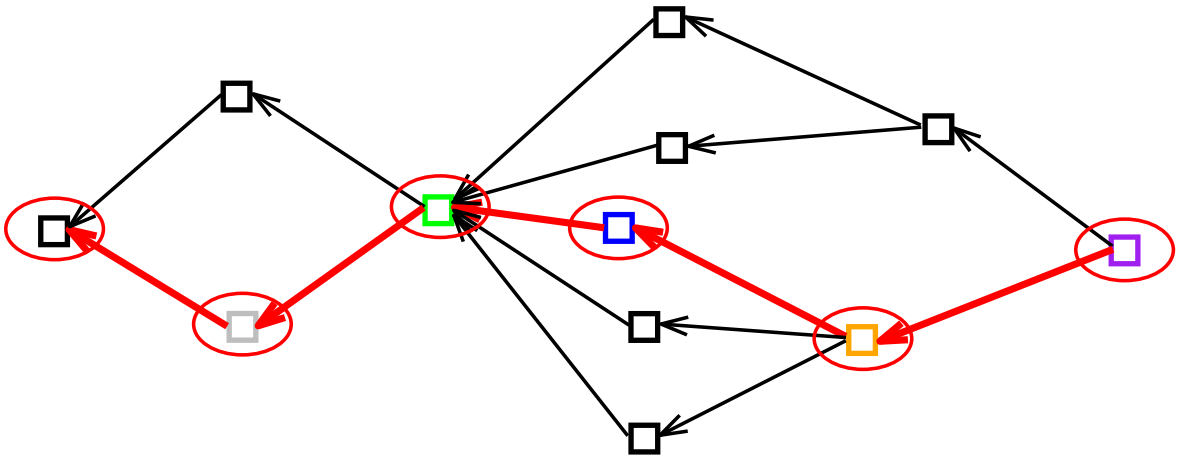


Figure: Main Chain Consists of Key Nodes

- Where come these key nodes?

The Entity

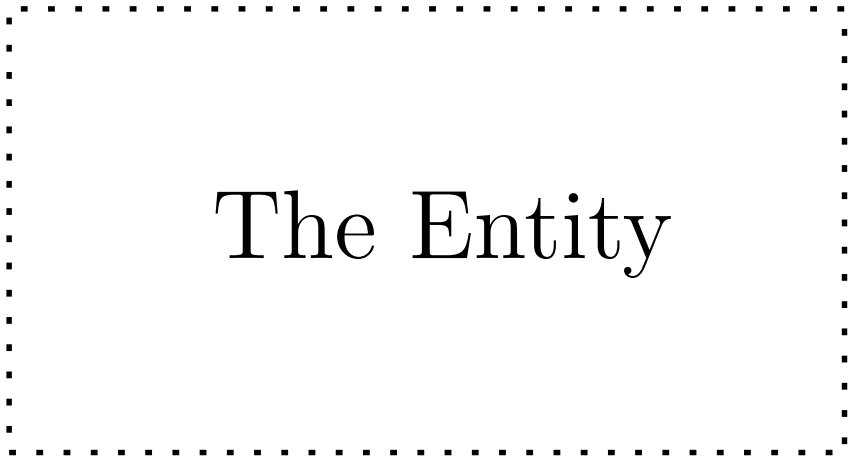


Figure: The Entity

- Assume there exists an entity issuing key nodes periodically

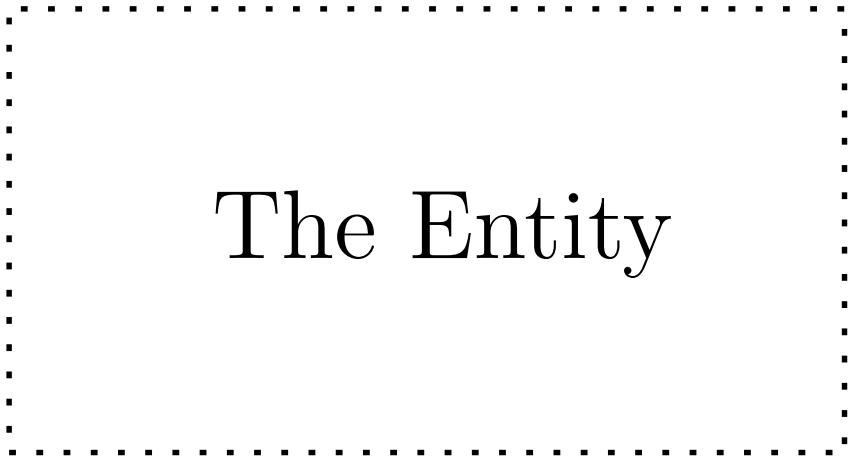


Figure: The Entity

- Assume there exists an entity issuing key nodes periodically
- The entity should be generated in a *decentralized* manner

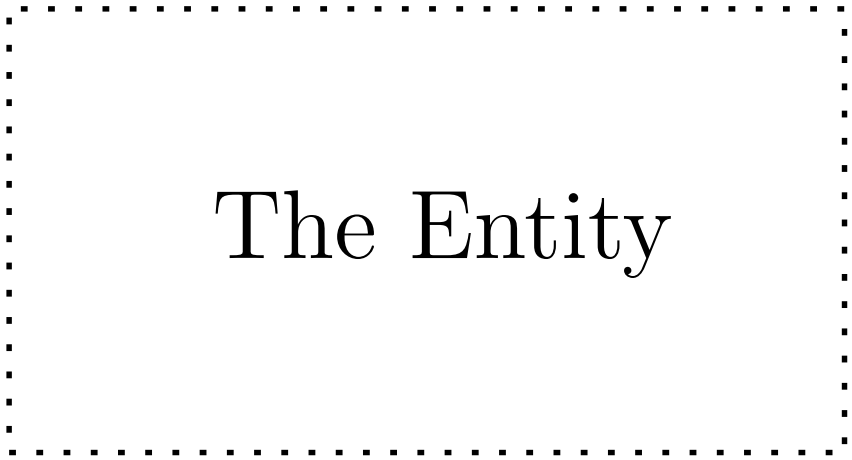


Figure: The Entity

- Assume there exists an entity issuing key nodes periodically
- The entity should be generated in a *decentralized* manner
- *Consistency and Liveness* should be guaranteed

Two-Layer Structure

Two-Layer Structure

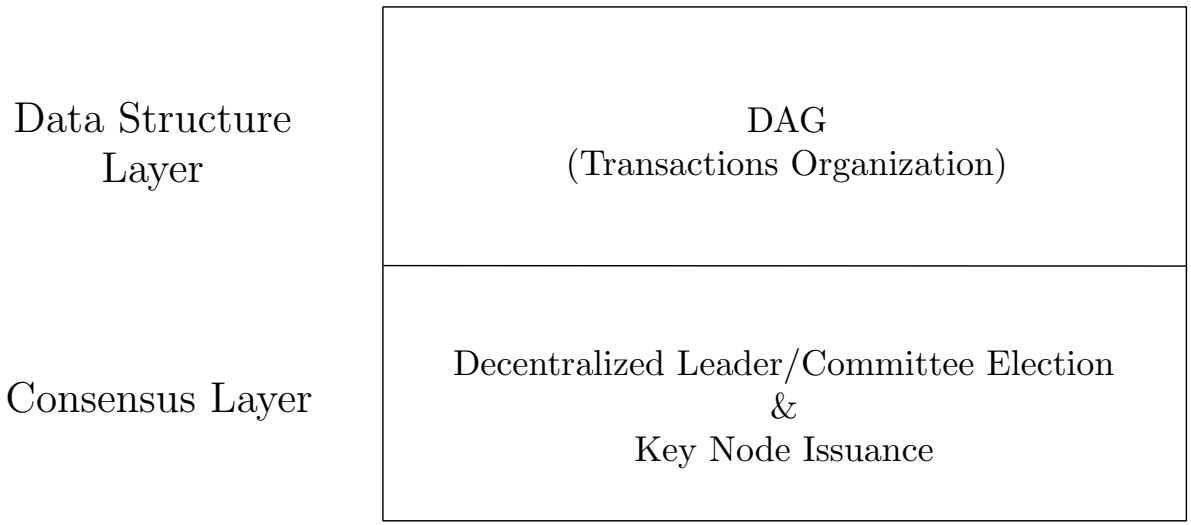


Figure: Two-Layer Structure

- Data Structure Layer: Transactions organization using a DAG

Two-Layer Structure

China Blockchain Conference

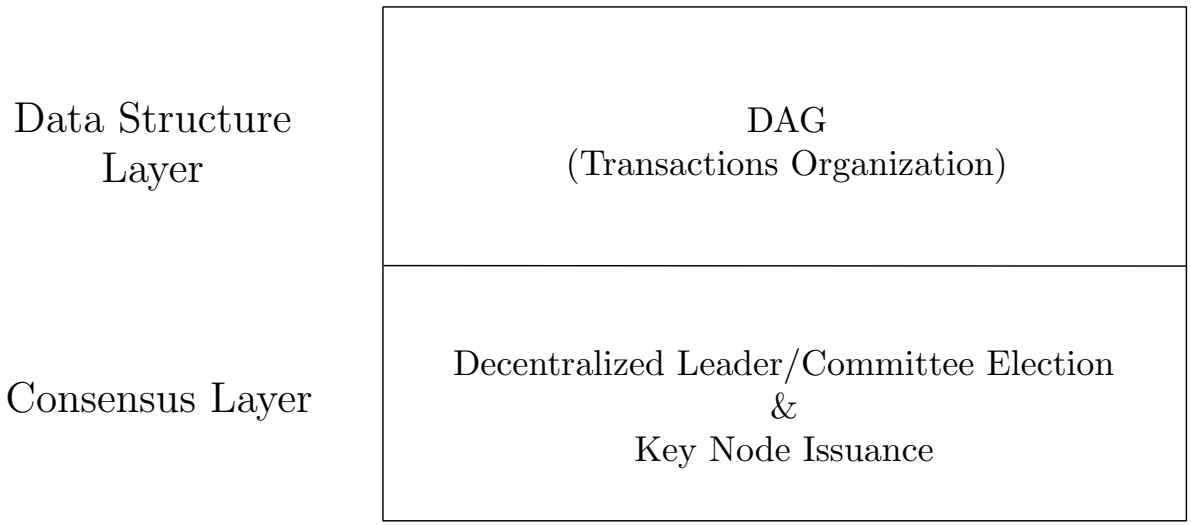


Figure: Two-Layer Structure

- Data Structure Layer: Transactions organization using a DAG
- Consensus Layer: Decentralized leader/committee election & key node issuance

Instantiation

The Committee

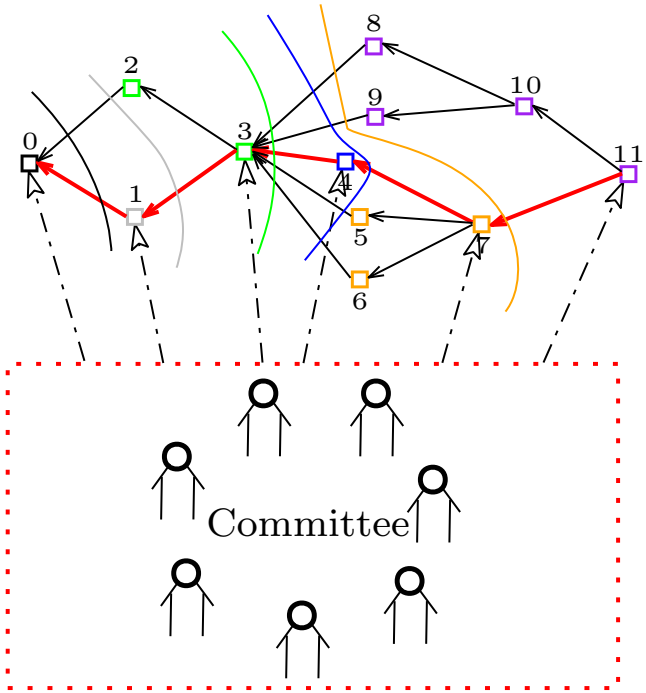


Figure: Consensus under a Committee

- Committee in a whole serves as the entity

The Committee

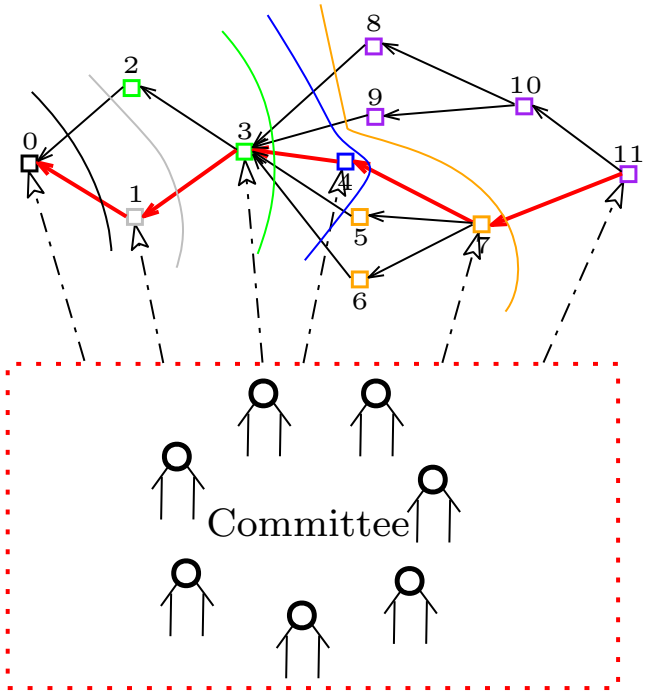


Figure: Consensus under a Committee

- Committee in a whole serves as the entity
- Its members run a BFT based protocol (with collective signing) to produce key nodes and broadcast them to the DAG

The Committee

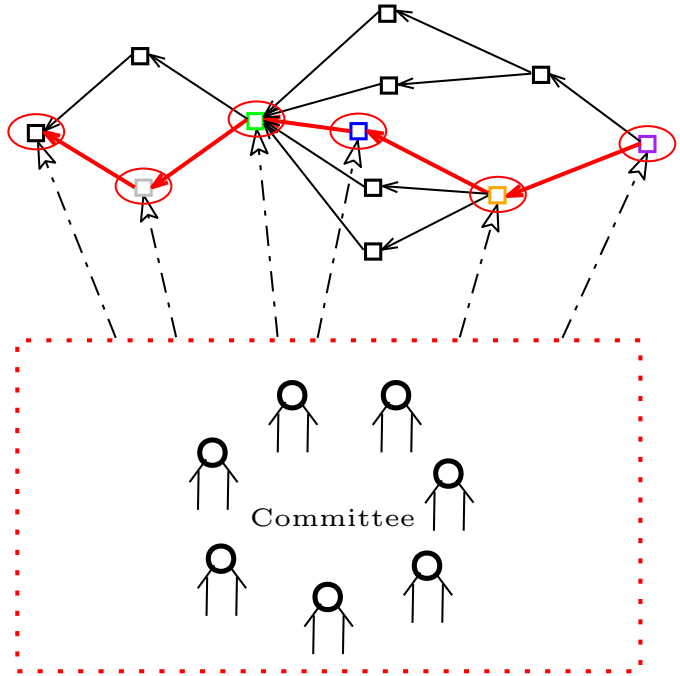


Figure: Consensus under a Committee

- Where comes the committee?

Backbone Chain

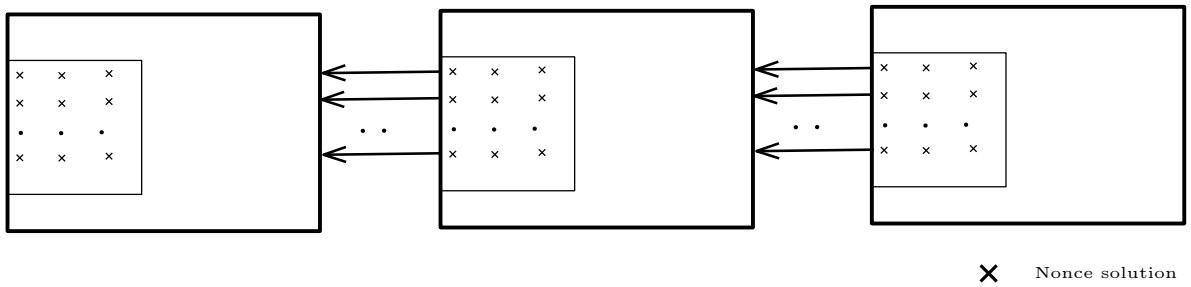


Figure: Committee Generated through the Backbone Chain

- Each committee is contained in a mining block

Backbone Chain

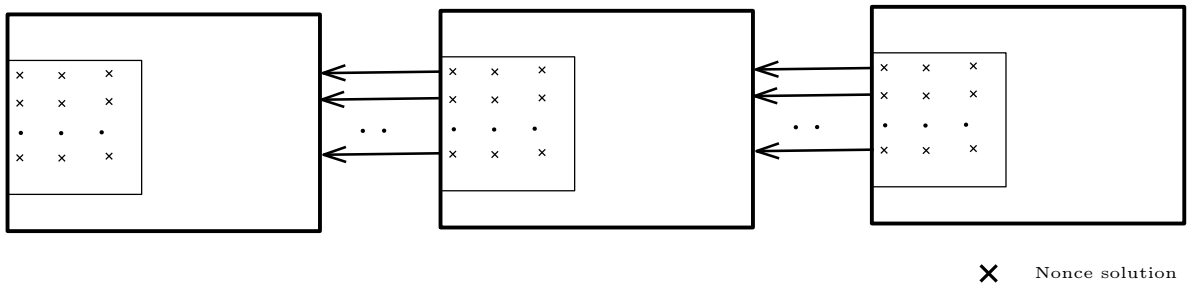


Figure: Committee Generated through the Backbone Chain

- Each committee is contained in a mining block
- PoW mining $\rightarrow$ Solutions for the puzzle $\rightarrow$ Consensus by the current committee $\rightarrow$ New committee

The Full Vision

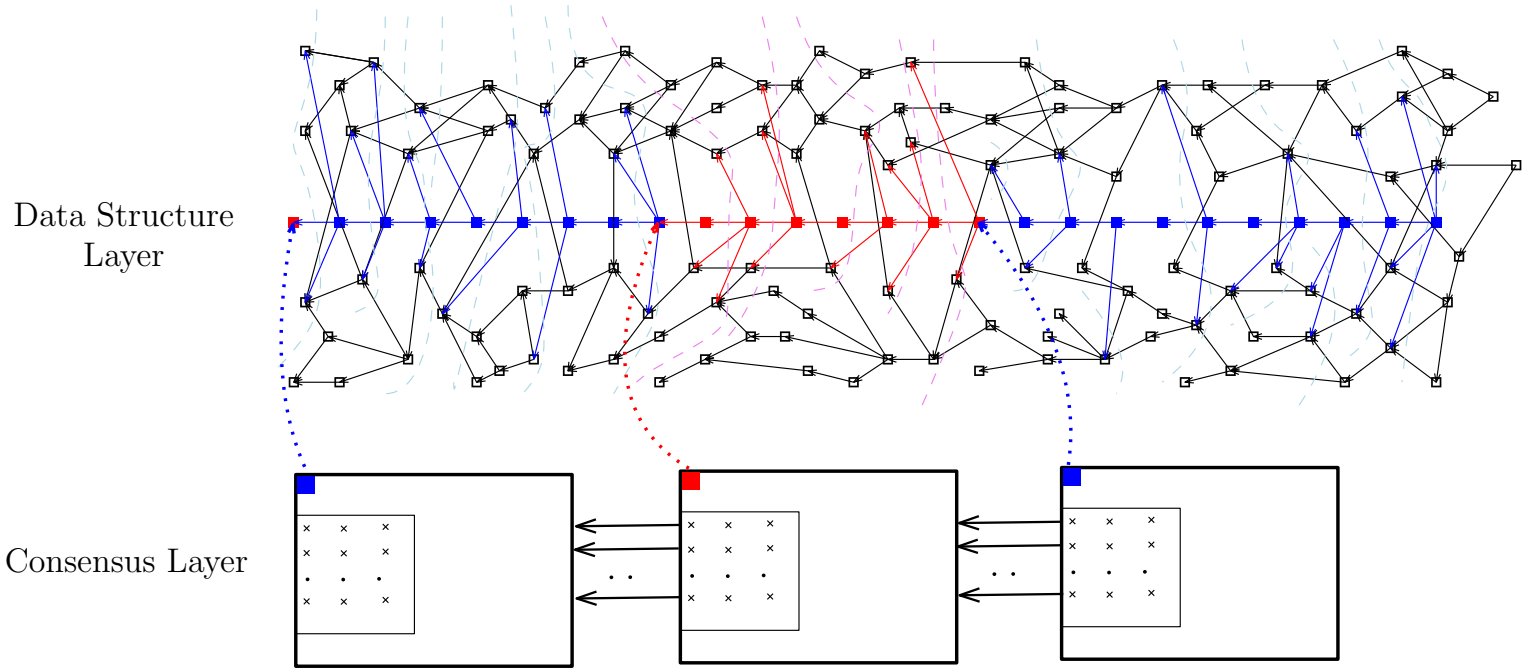


Figure: The Full Vision

Feature

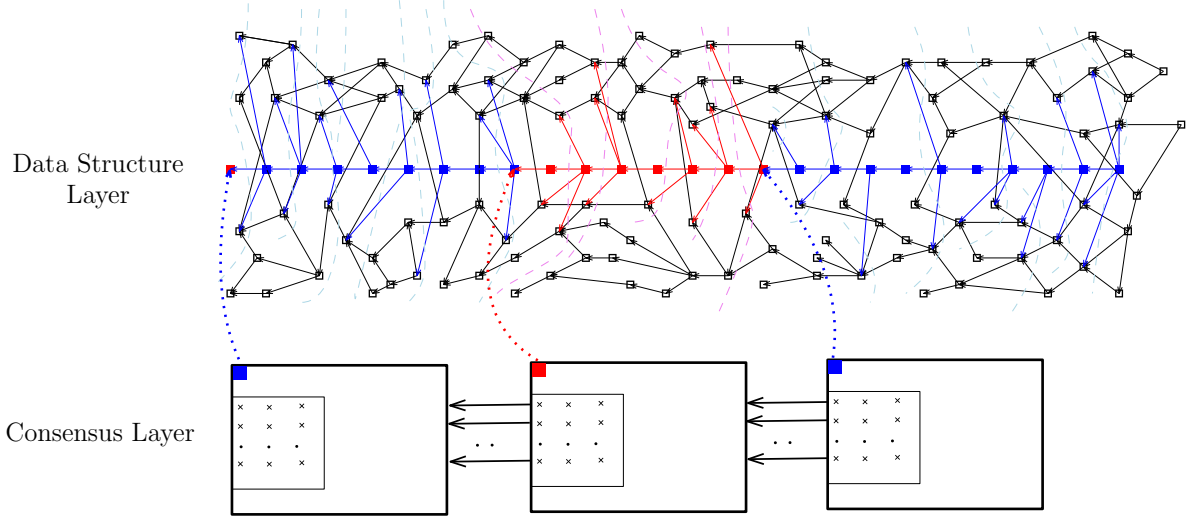


Figure: The Full Vision

- High throughput
- Fast settlement
- Deterministic finality

China Blockchain Conference

DAG Formalization in a Nutshell

The DAG

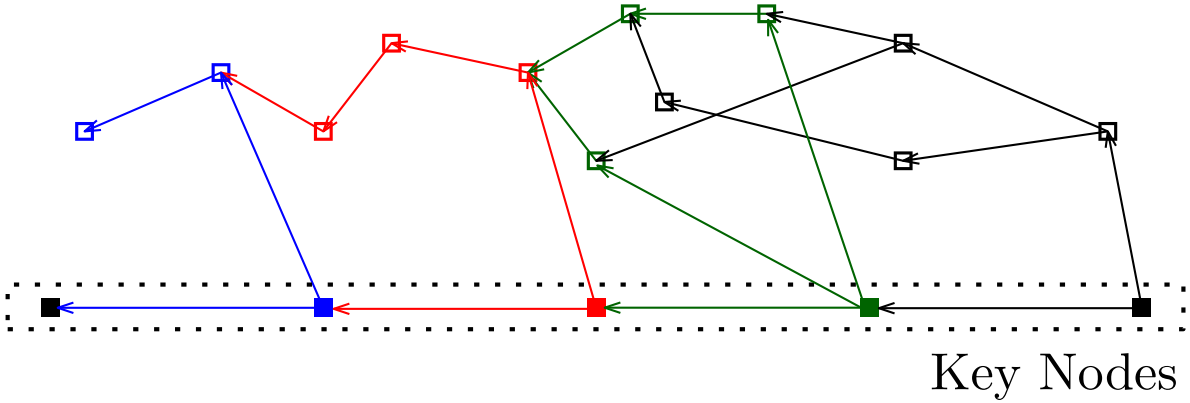


Figure: The Ledger of Transactions

- The ledger consists of transactions nodes and key nodes.

The DAG

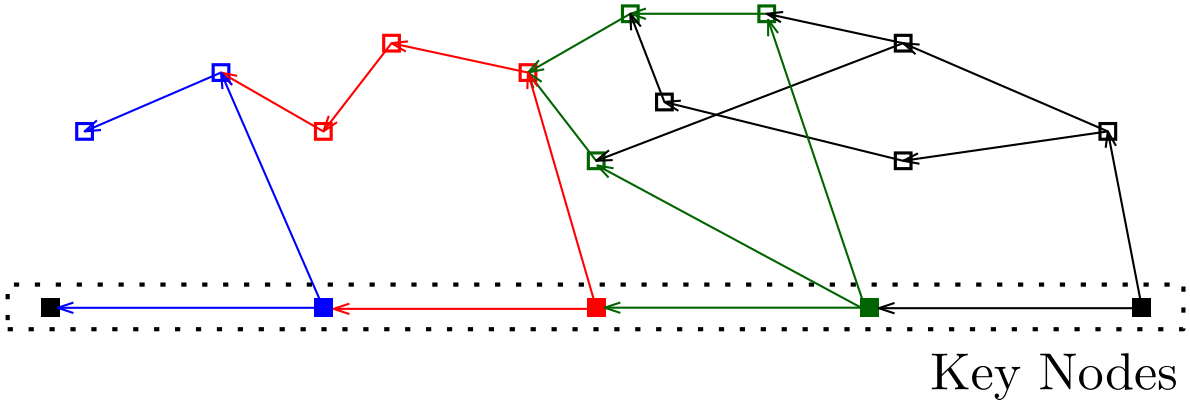


Figure: The Ledger of Transactions

- The ledger consists of transactions nodes and key nodes.
- Each transaction node (a "□" in the figure) denotes a transaction to be validated.

The DAG

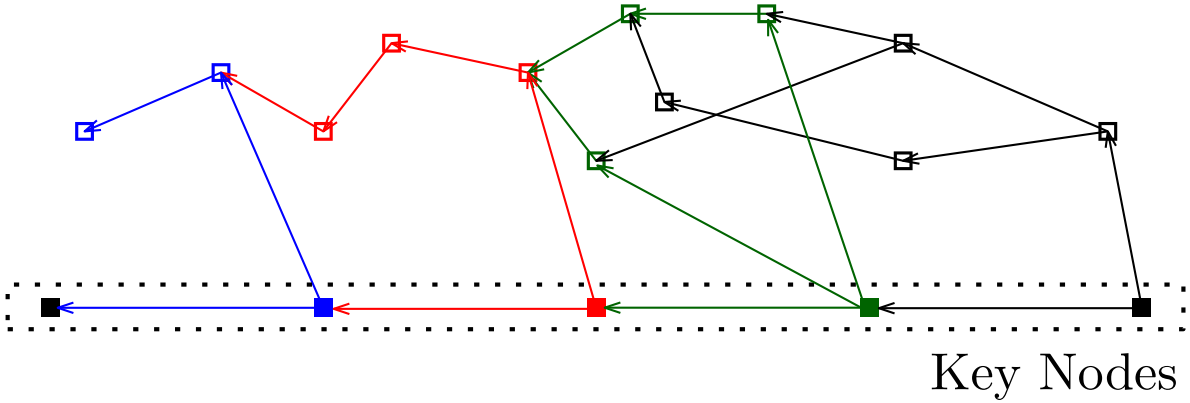


Figure: The Ledger of Transactions

- The ledger consists of transactions nodes and key nodes.
- Each transaction node (a "□" in the figure) denotes a transaction to be validated.
- Key nodes are issued by the committee to linearize all transaction nodes (we will soon see how it works).

An Intuitional Description

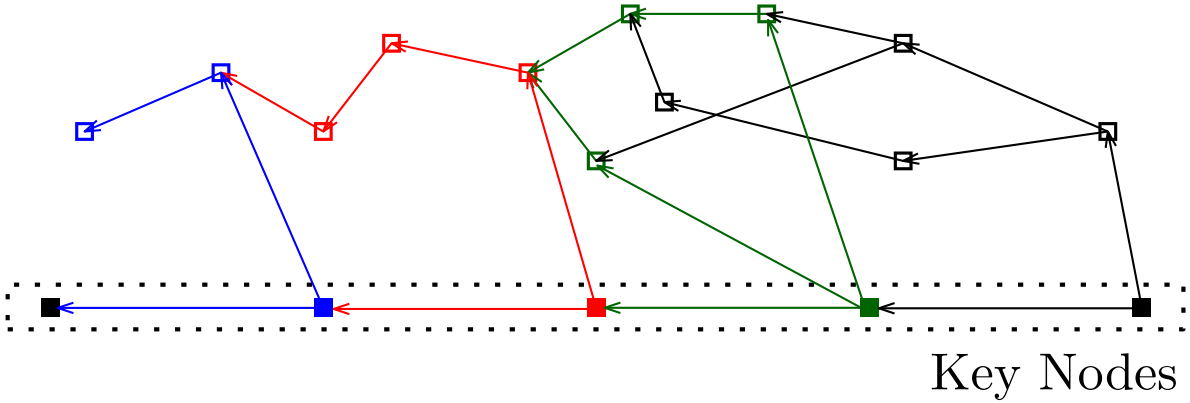


Figure: The Ledger of Transactions

- Each key node "confirms" several new transaction nodes (marked by different colors).

An Intuitional Description

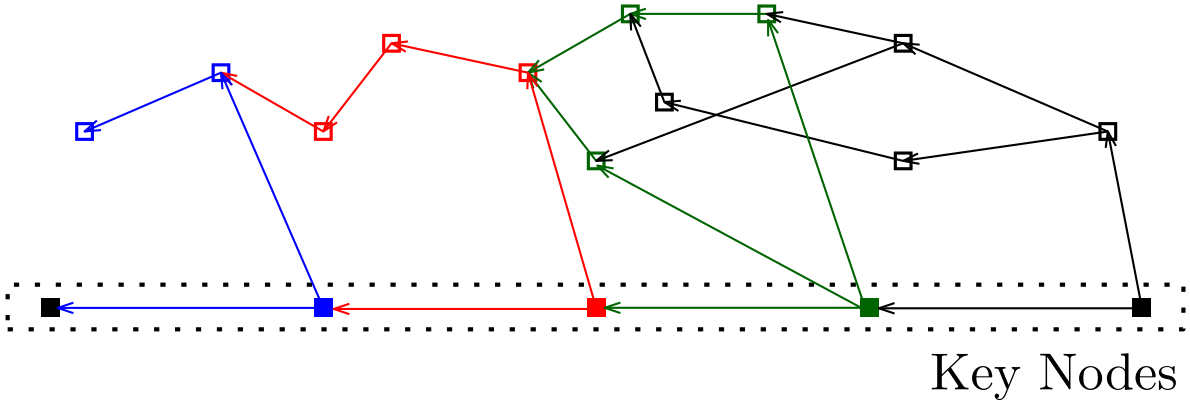


Figure: The Ledger of Transactions

- Each key node "confirms" several new transaction nodes (marked by different colors).
- These newly confirmed nodes are appended to the ledger with a determined ordering.

An Intuitional Description

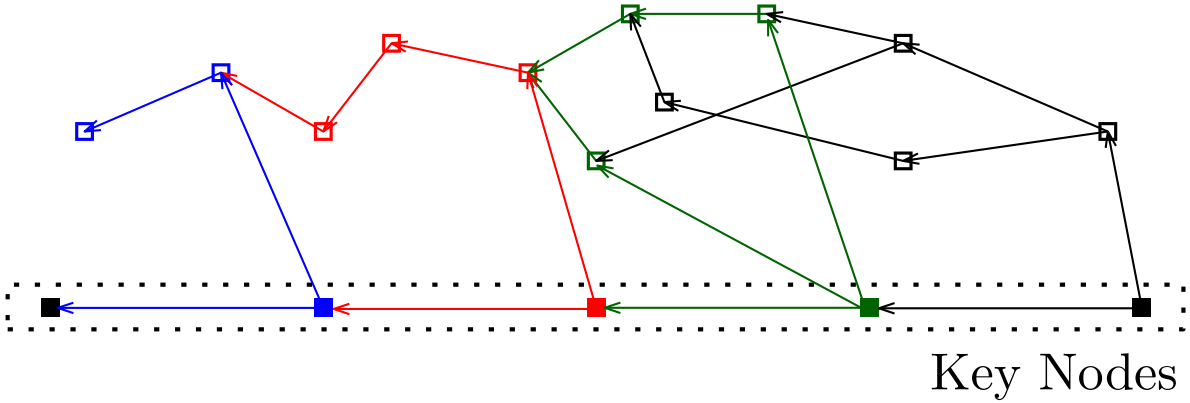


Figure: The Ledger of Transactions

- Each key node "confirms" several new transaction nodes (marked by different colors).
- These newly confirmed nodes are appended to the ledger with a determined ordering.
- A restrict proof of consistency is shown in our full paper (to be released soon).

Vertex & Edge Set

- A (directed) graph is a pair $G = (V, E)$, where

Vertex & Edge Set

- A (directed) graph is a pair $G = (V, E)$, where
- $V = V_{\text{tx}} \cup V_{\text{key}}$ $(V_{\text{tx}} \cap V_{\text{key}} = \emptyset)$ is the vertex set;

Vertex & Edge Set

- A (directed) graph is a pair $G = (V, E)$, where
- $V = V_{\mathsf{tx}} \cup V_{\mathsf{key}}$ $(V_{\mathsf{tx}} \cap V_{\mathsf{key}} = \emptyset)$ is the vertex set;
- $E \subseteq V \times V$ is the edge set.

Admissible DAG

A (directed) graph $G = (V = V_{\text{tx}} \cup V_{\text{key}}, E)$ $(V_{\text{tx}} \cap V_{\text{key}} = \emptyset, E \subseteq V \times V)$ is called an admissible DAG iff

1. It is a DAG.

Admissible DAG

A (directed) graph $G = (V = V_{\text{tx}} \cup V_{\text{key}}, E)$ ($V_{\text{tx}} \cap V_{\text{key}} = \emptyset$, $E \subseteq V \times V$) is called an admissible DAG iff

1. It is a DAG.
2. It is a succinct DAG (so that new nodes are encouraged to refer tip nodes).

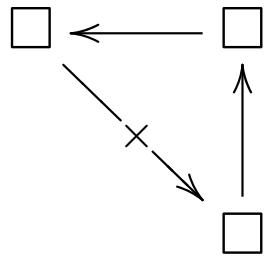
Admissible DAG

A (directed) graph $G = (V = V_{\text{tx}} \cup V_{\text{key}}, E)$ ($V_{\text{tx}} \cap V_{\text{key}} = \emptyset$, $E \subseteq V \times V$) is called an admissible DAG iff

1. It is a DAG.
2. It is a succinct DAG (so that new nodes are encouraged to refer tip nodes).
3. Key nodes form a totally ordered chain.

Formal Description of An Admissible DAG

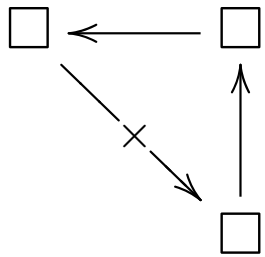
- **DAG.**



$$\forall \ell \in [|V|] \setminus \{1\}.\ \forall (v_1, v_2, \ldots, v_\ell) \in V^\ell.$$
$$\big(\forall i \in [\ell - 1].\ (v_i, v_{i+1}) \in E\big) \Rightarrow (v_\ell, v_1) \notin E.$$

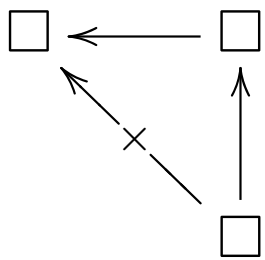
Formal Description of An Admissible DAG

- **DAG.**



$$\forall \ell \in [|V|] \setminus \{1\}.\ \forall (v_1, v_2, \ldots, v_\ell) \in V^\ell.$$
$$\big(\forall i \in [\ell - 1].\ (v_i, v_{i+1}) \in E\big) \Rightarrow (v_\ell, v_1) \notin E.$$

- **Succictness.**

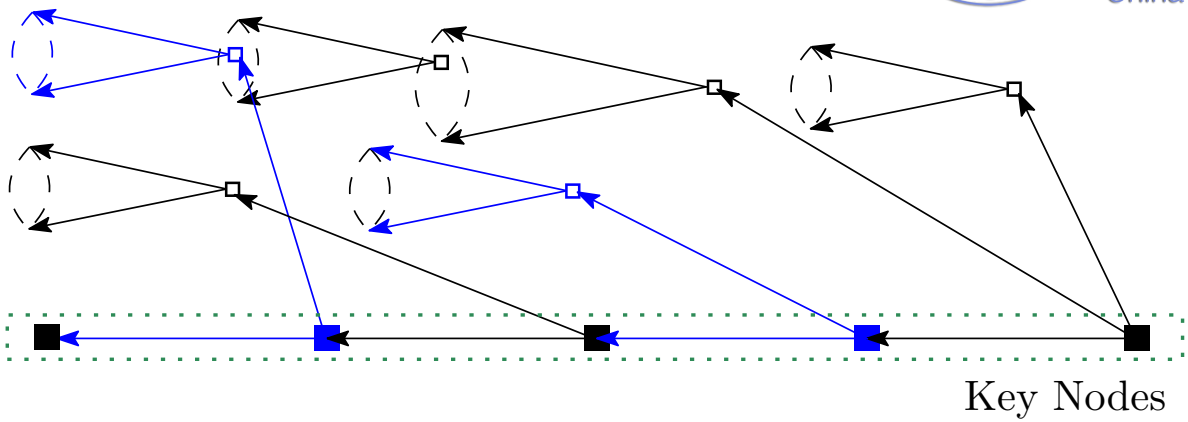


$$\forall \ell \in [|V_{\mathsf{tx}}|] \setminus \{1, 2\}.\ \forall (v_1, v_2, \ldots, v_\ell) \in V_{\mathsf{tx}}^\ell.$$
$$\big(\forall i \in [\ell - 1].\ (v_i, v_{i+1}) \in E\big) \Rightarrow (v_1, v_\ell) \notin E.$$

Formal Description of An Admissible DAG

- **Ordered Key Node Chain.** $V_{\text{key}} = \left\{u_1, u_2, \ldots, u_{|V_{\text{key}}|}\right\}$ satisfies that $\left(\forall i \in \left[|V_{\text{key}}| - 1\right]. \ (u_{i+1}, u_i) \in E\right)$ $\wedge \left(\forall i, j \in [|V_{\text{key}}|]. \ i \neq j + 1 \Rightarrow (u_i, u_j) \notin E\right)$

Recursion Tree



The recursion tree $\mathrm{Rec}(p) = (V', E')$ $(p \in V)$ of a newly added key node p in an admissible directed acyclic graph $G = (V, E)$ is defined as the subgraph of G with

$$\begin{cases} V' = & \{p\} \cup \Big\{ v \in V \;\Big|\; \exists \ell \in [|V|]. \\ & \exists (u_0 = p, u_1, u_2, \ldots, u_\ell = v) \in V^{\ell+1}. \\ & \Big(\forall i \in [\ell]. \; (u_{i-1}, u_i) \in E \Big) \Big\} \\ E' = & \Big\{ (u, v) \in E \;\Big|\; u \in V' \Big\}. \end{cases}$$

Reversed Breadth-First Traverse Sequence

We denote RBF(G) as the reverse of the breath-first traverse sequence of (sub)graph G.

Total Order

- We assume an admissible DAG $G = (V, E)$ with key units $V_{\text{key}} = \{u_1, u_2, \ldots, u_\ell\}$. The total order of each vertex is defined as its position in the sequence

$$\text{Total}(G) = \text{RBF}(\text{Inc}(u_1))||\text{RBF}(\text{Inc}(u_2))||\ldots||\text{RBF}(\text{Inc}(u_\ell))$$

  if it is included in $\text{Inc}(u_i)$ for any $i \in [\ell]$, or infinity on the other case.

Total Order

- We assume an admissible DAG $G = (V, E)$ with key units $V_{\text{key}} = \{u_1, u_2, \ldots, u_\ell\}$. The total order of each vertex is defined as its position in the sequence

$$\text{Total}(G) = \text{RBF}(\text{Inc}(u_1))||\text{RBF}(\text{Inc}(u_2))||\ldots||\text{RBF}(\text{Inc}(u_\ell))$$

  if it is included in $\text{Inc}(u_i)$ for any $i \in [\ell]$, or infinity on the other case.
- **Theorem 1**: Total ordering is well-defined for an admissible DAG.

Brief Idea of Our Security Analysis

Proof Roadmap for Classical Distributed Consensus

- First Step: Common Prefix, Chain Quality, Chain Growth.
- Second Step: Consistency, Liveness.

Obviously, this roadmap is not suitable to our scheme!

Our Security Goals

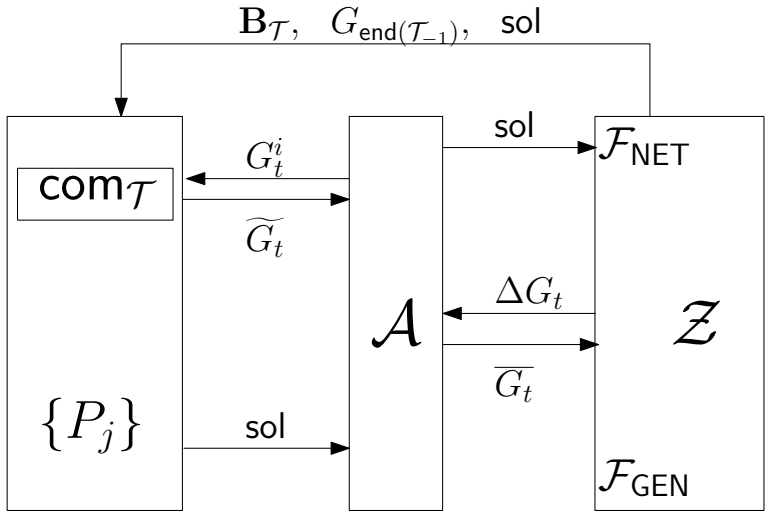


Figure: Our Execution Model

What we need prove is essentially:

- Common Prefix. $\mathsf{Total}(\overline{G}_s) \preccurlyeq \mathsf{Total}(\overline{G}_t)$ for all $s \leq t$.

Our Security Goals

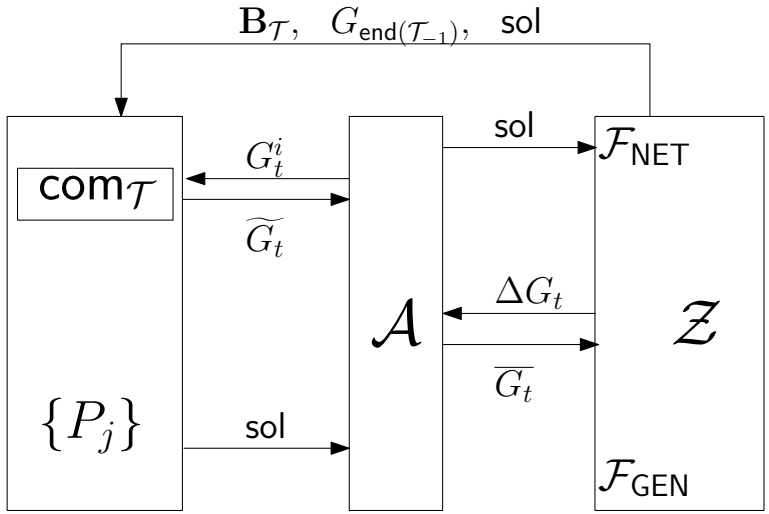


Figure: Our Execution Model

What we need prove is essentially:

- Common Prefix. $\mathsf{Total}(\overline{G}_s) \preccurlyeq \mathsf{Total}(\overline{G}_t)$ for all $s \leq t$.
- Liveness. $\Delta G_t \sqsubseteq \overline{G}_{t+2\delta}$ for all t.

The proof

Welcome to find our proof in the full paper

Thanks for listening!