

Mysticeti

The new core of the Sui blockchain

Alberto Sonnino

Byzantine Fault Tolerance



$> 2/3$



Byzantine Fault Tolerance



$\geq 2f+1$



$3f+1$

Partial Synchrony



Keeping the Talk Short

In scope

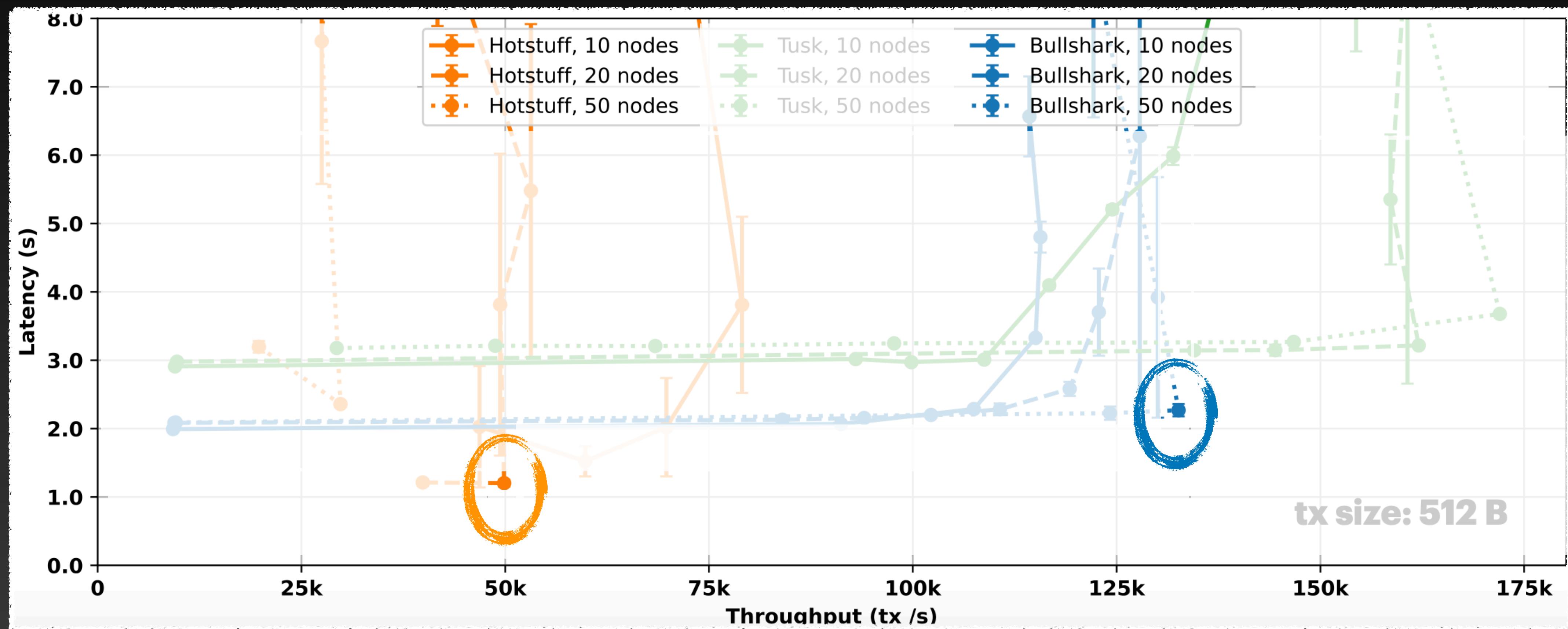
- Ordering (quorum-based)



Not in scope

- Nodes selection?
- Committee reconfiguration?
- Transactions execution?
- Transactions language?
- Financial incentives?
- etc

Why? Latency



Why? Crash Faults

In a year of running Sui:

Why? Crash Faults

In a year of running Sui:

- How many Byzantine faults?

Why? Crash Faults

In a year of running Sui:

- How many Byzantine faults? 0

Why? Crash Faults

In a year of running Sui:

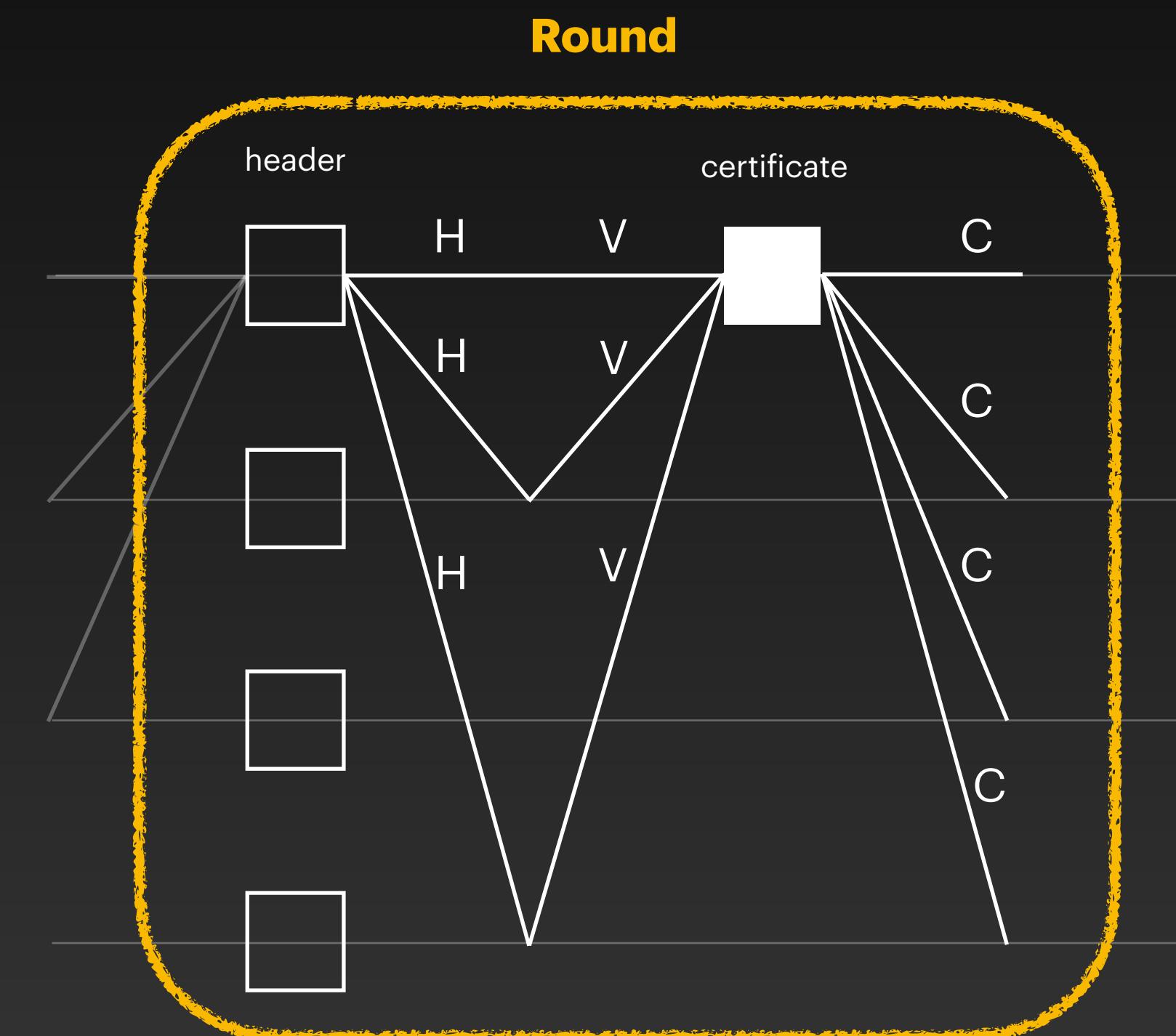
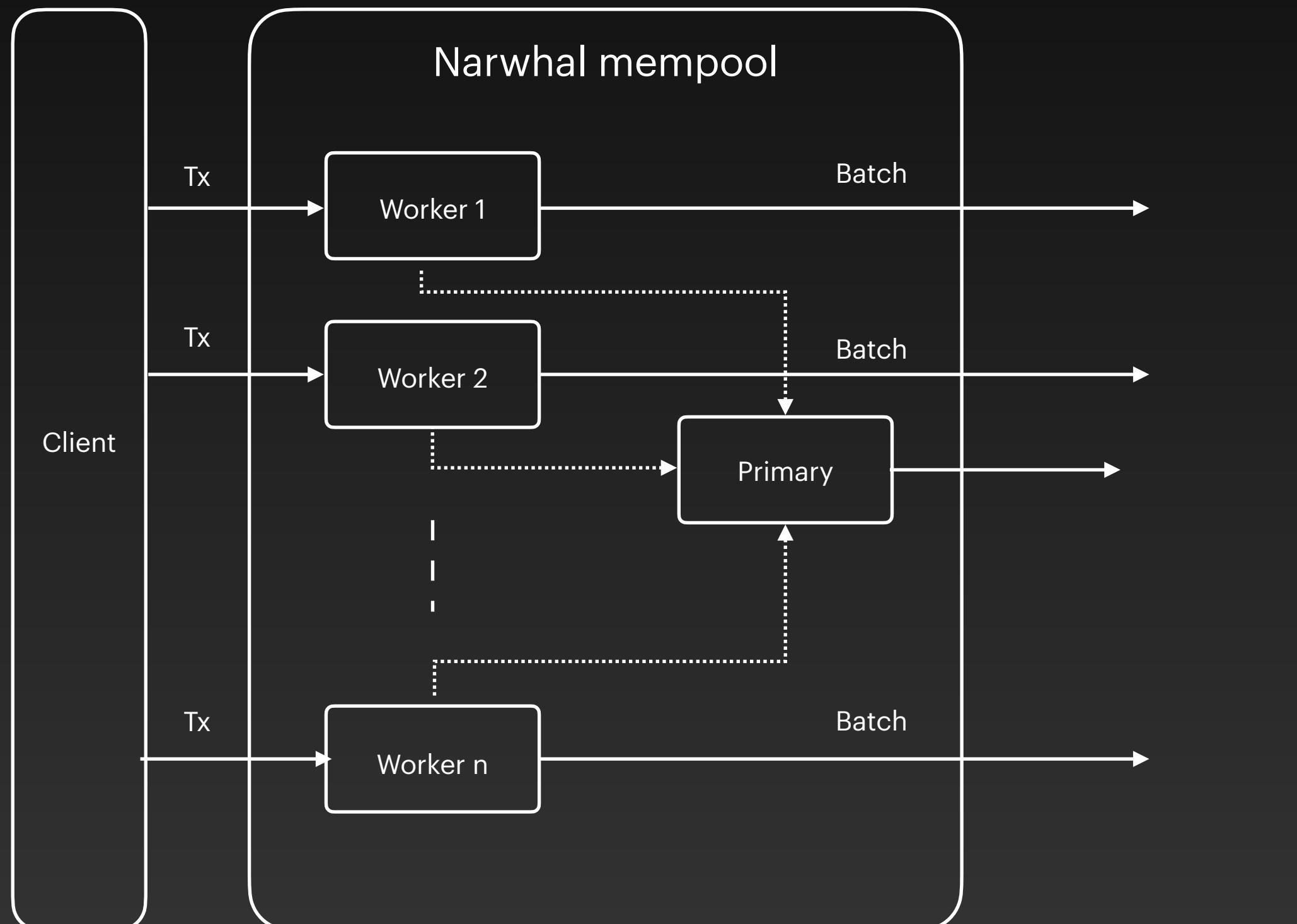
- How many Byzantine faults? 0
- How many Crash faults?

Why? Crash Faults

In a year of running Sui:

- How many Byzantine faults? 0
- How many Crash faults? 😭

Why? Engineering Complexity

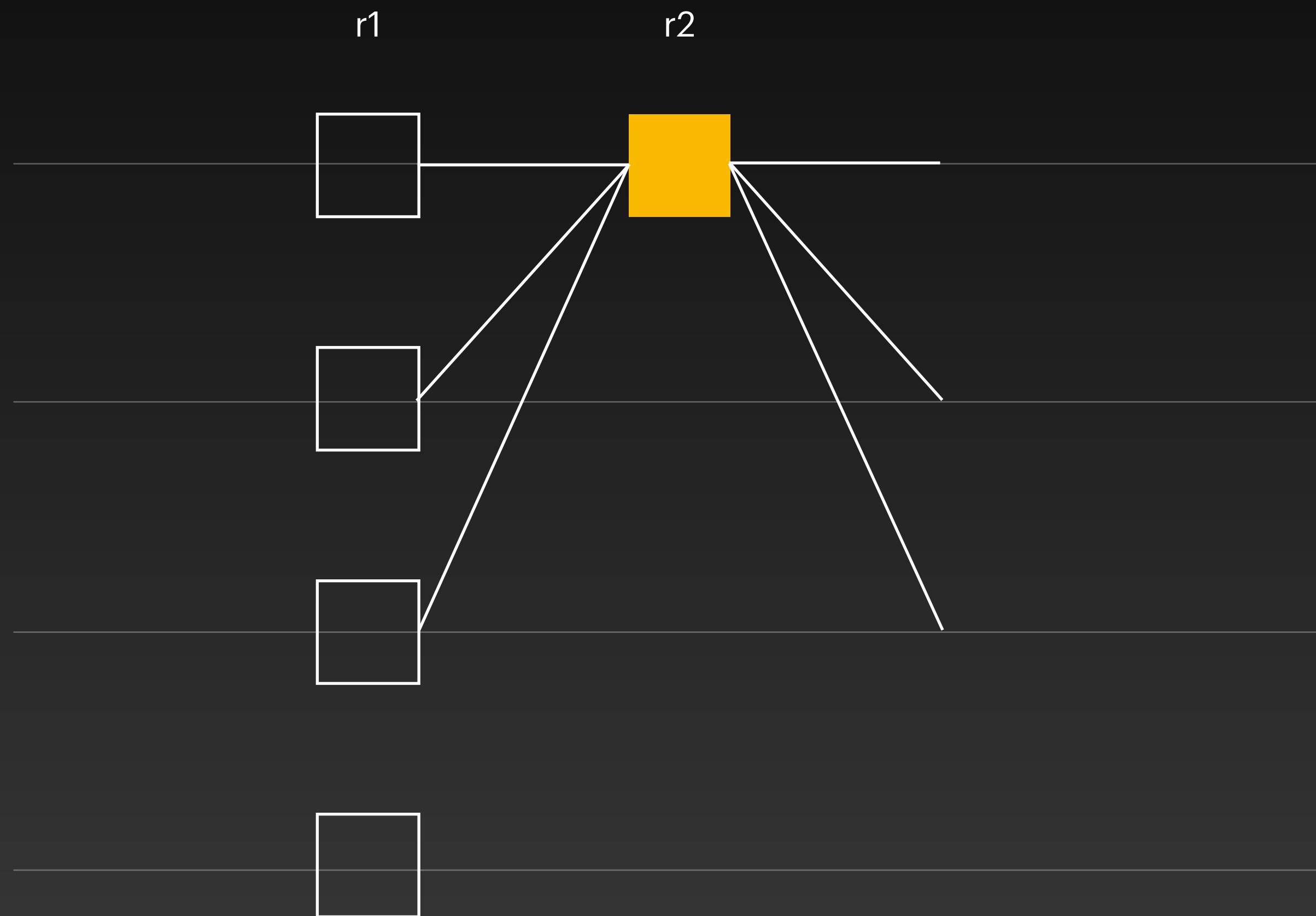


The Mysticeti DAG

Uncertified DAG

The Mysticeti DAG

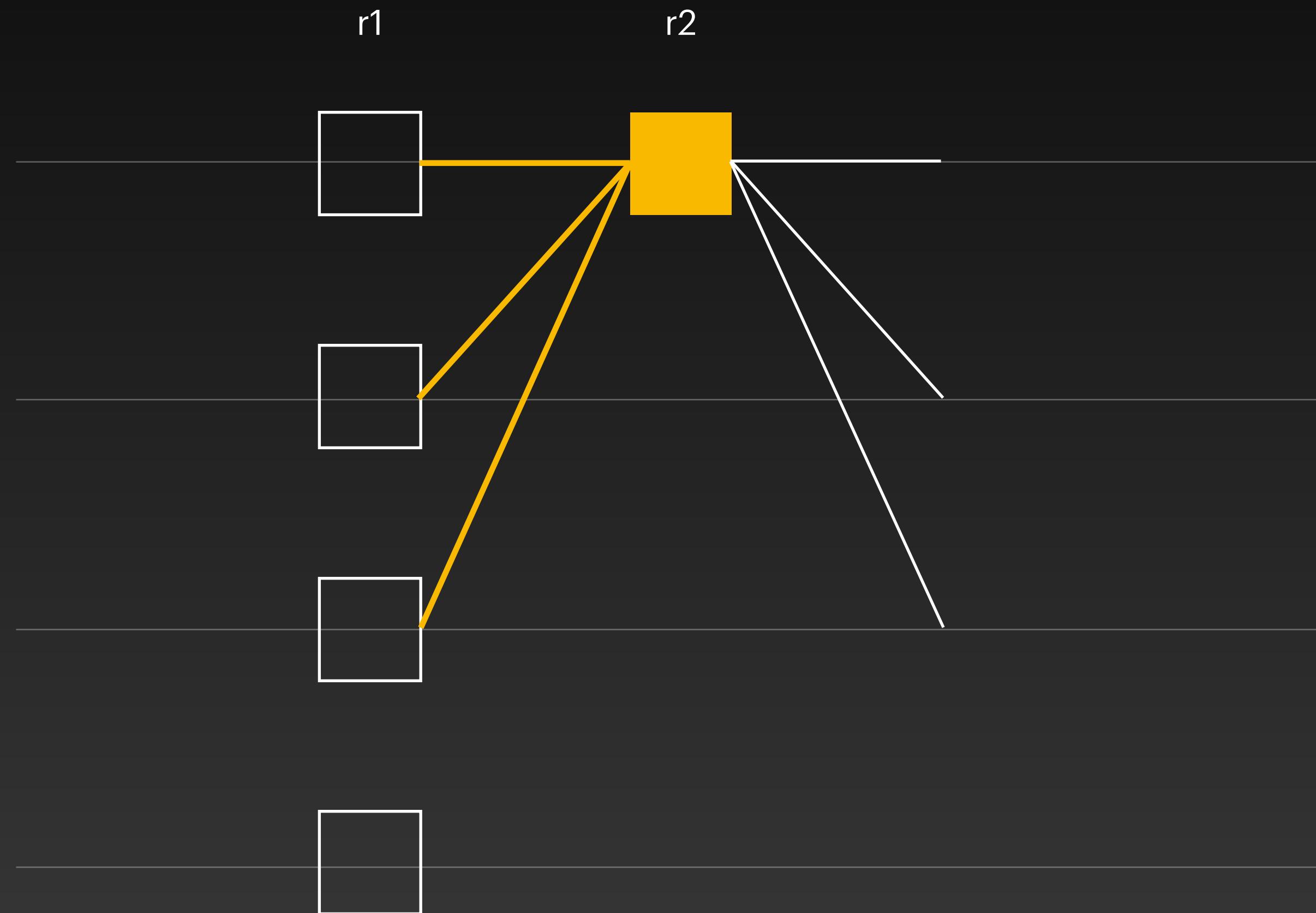
Block Creation



- Round number
- Author
- Payload (transactions)
- Signature

The Mysticeti DAG

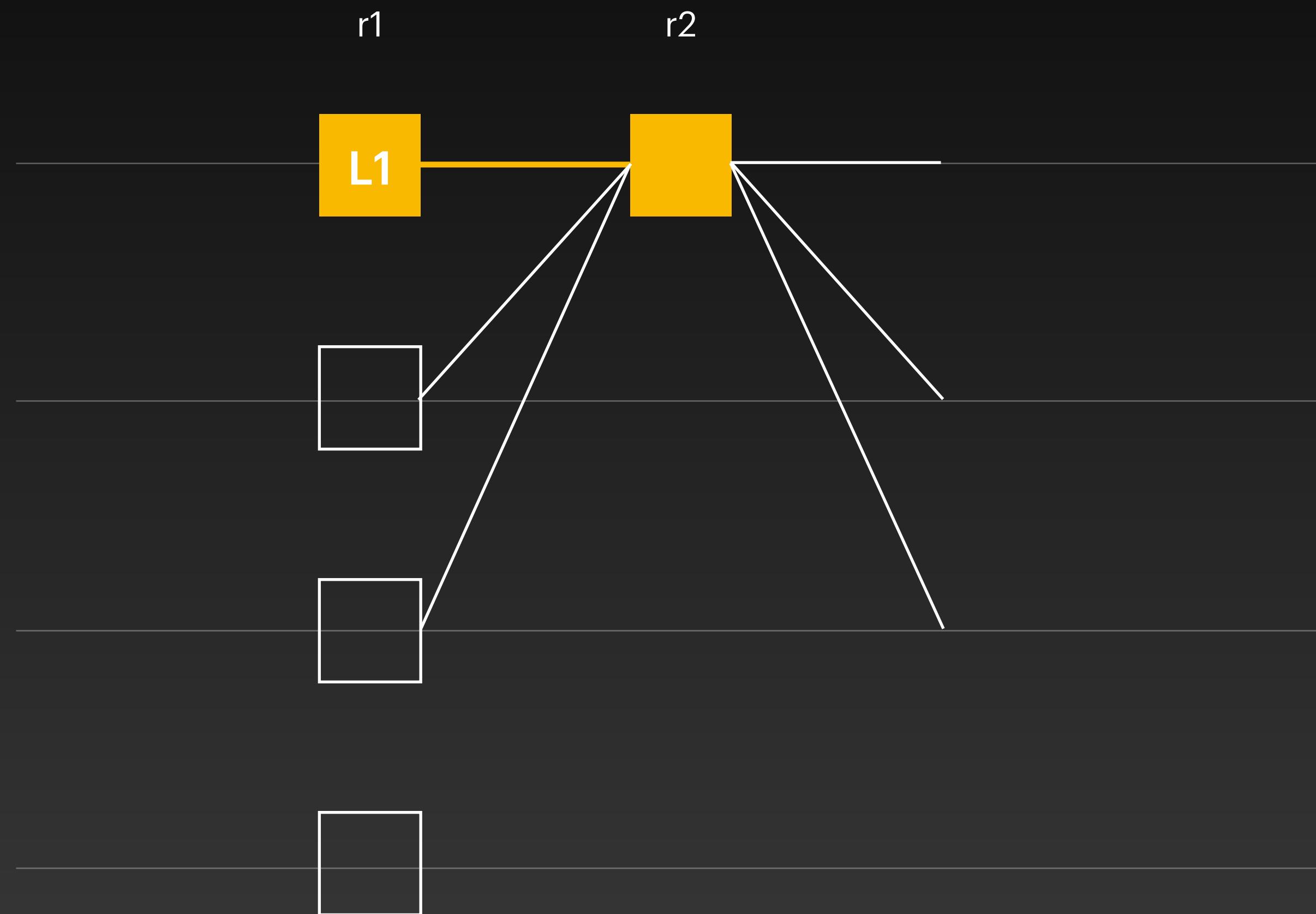
Rule 1: Link to $2f+1$ parents



- Total nodes: **$3f+1 = 4$**
- Quorum: **$2f+1 = 3$**

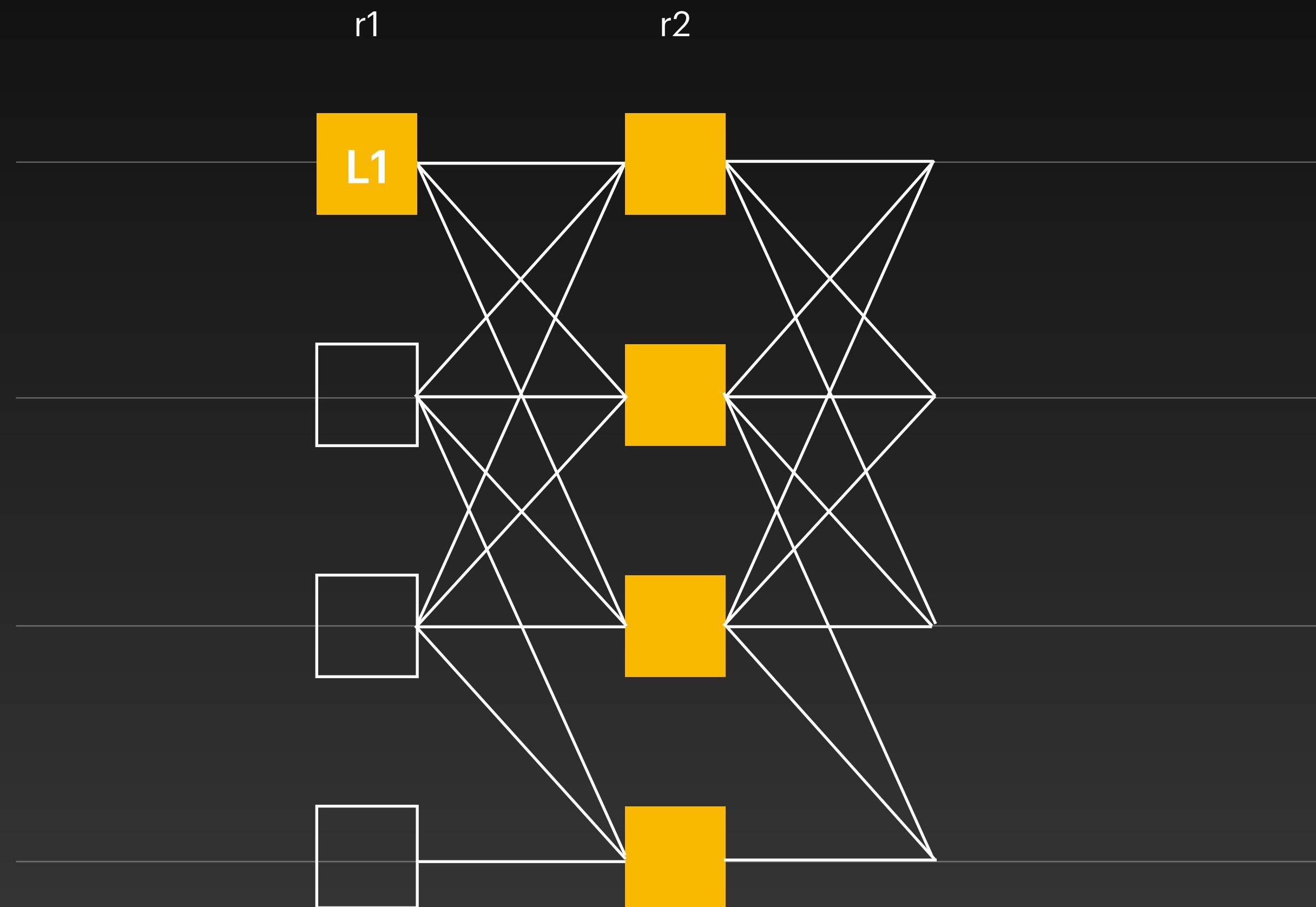
The Mysticeti DAG

Rule 2: Every node waits and links to leaders

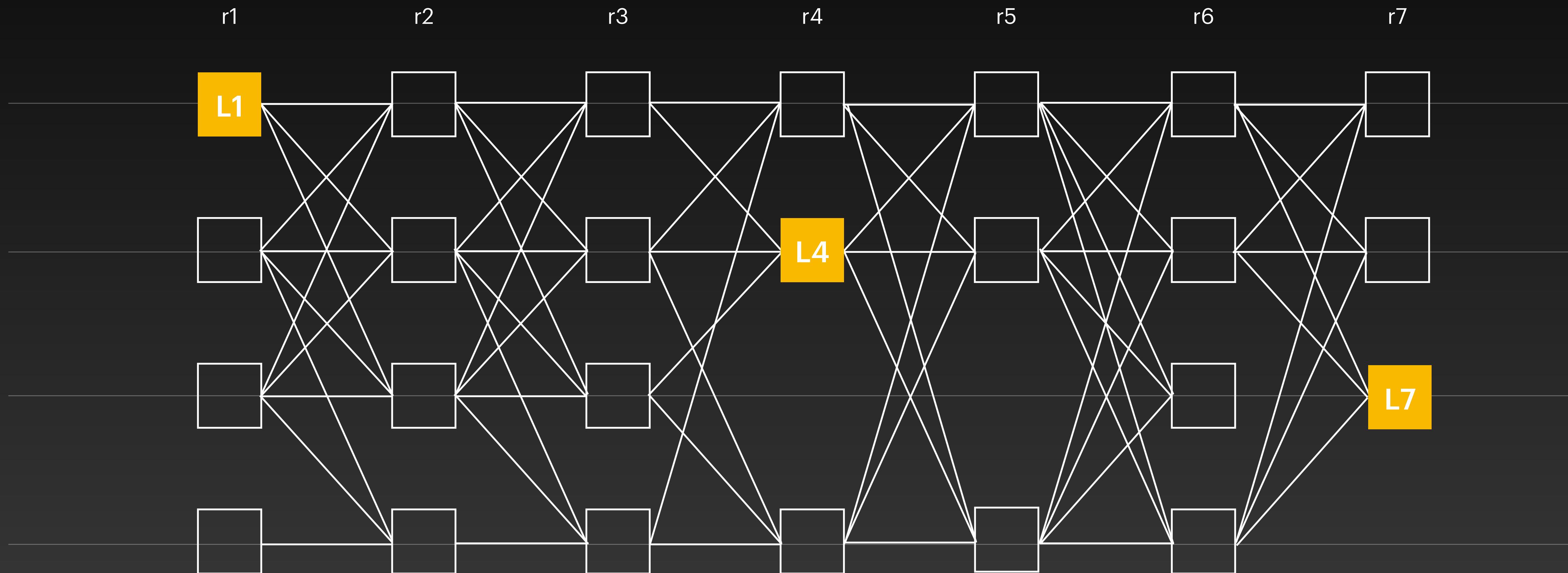


The Mysticeti DAG

Rule 3: All node run in parallel



The Mysticeti DAG

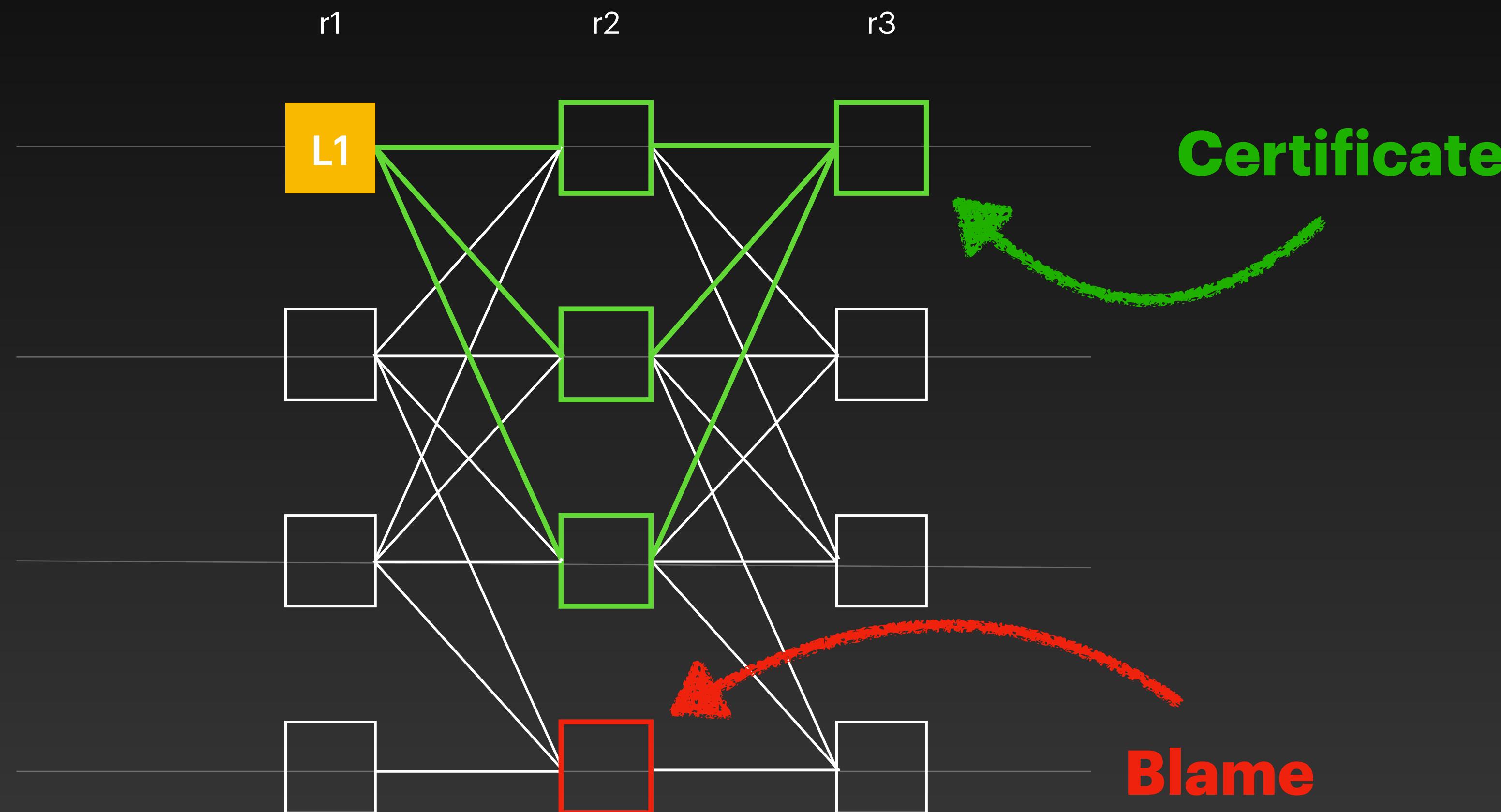


Main Ingredient:

All messages embedded in the DAG

- Fewer signatures
- Isolated engineering component
- Define interpretable patterns on the DAG
- Run multiple protocols on the same DAG

Interpreting DAG Patterns



Two Protocols, One DAG

Mysticeti-C Consensus

- No rounds without leader
- Multiple leaders per round

Mysticeti-FPC Adding Fast Finality

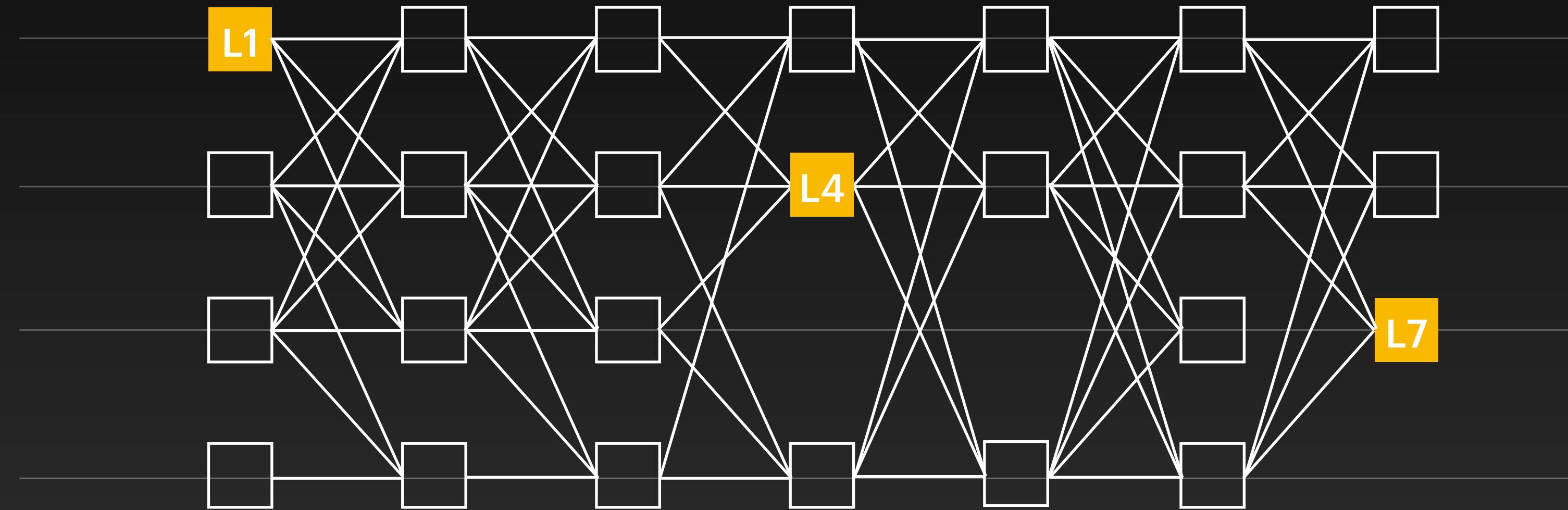
- Interpret BCB on DAG

Mysticeti-C

The consensus protocol

End Goal

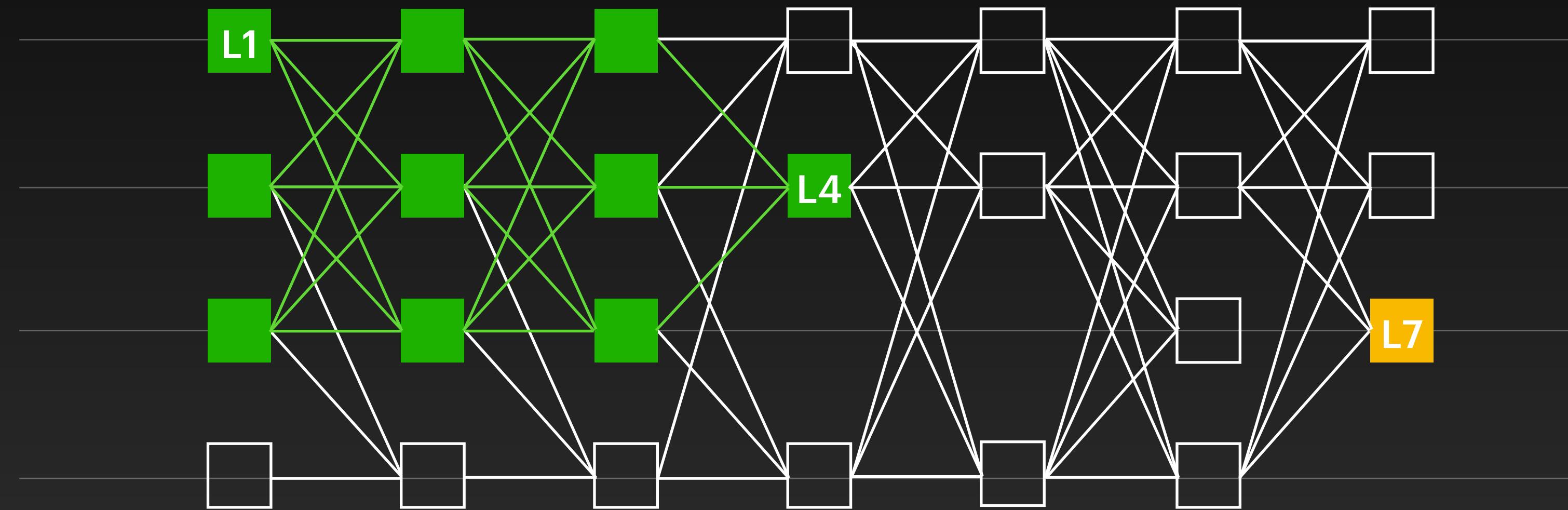
Ordering leaders



- We focus on ordering leaders: L1 L4 L7

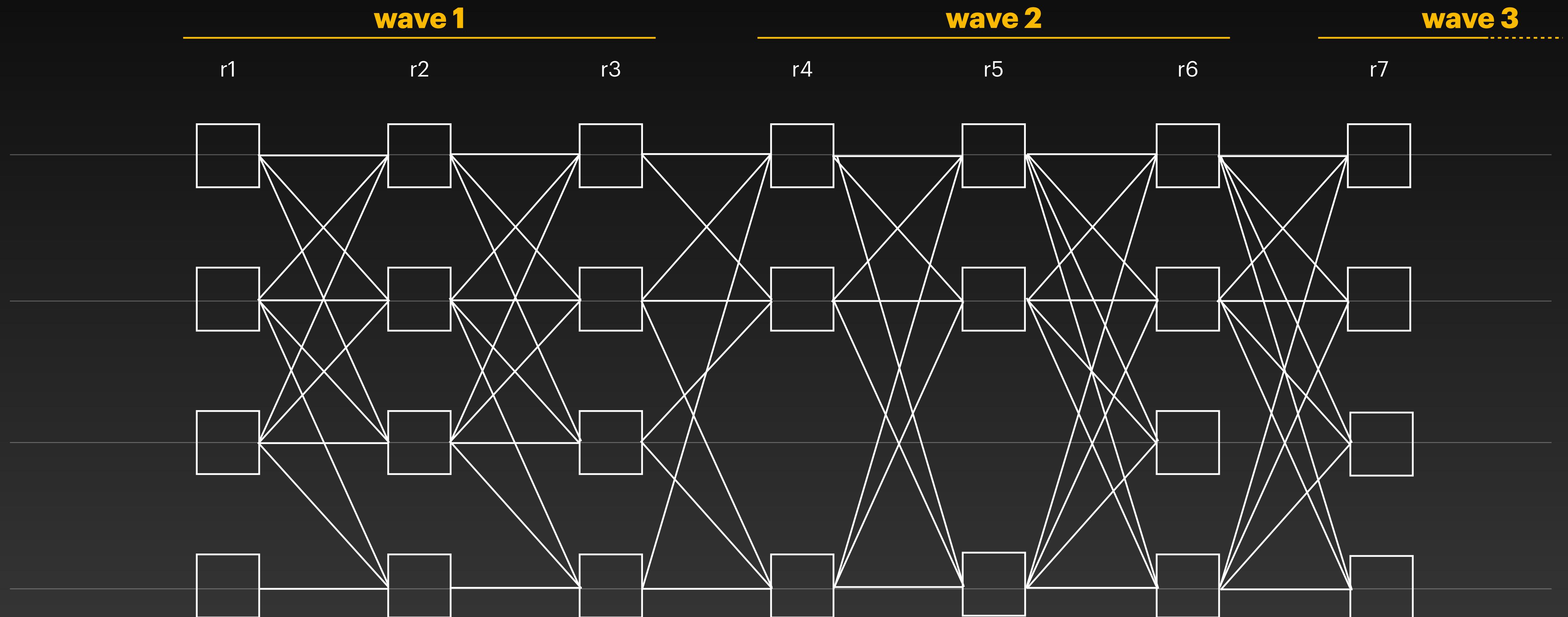
End Goal

Ordering leaders

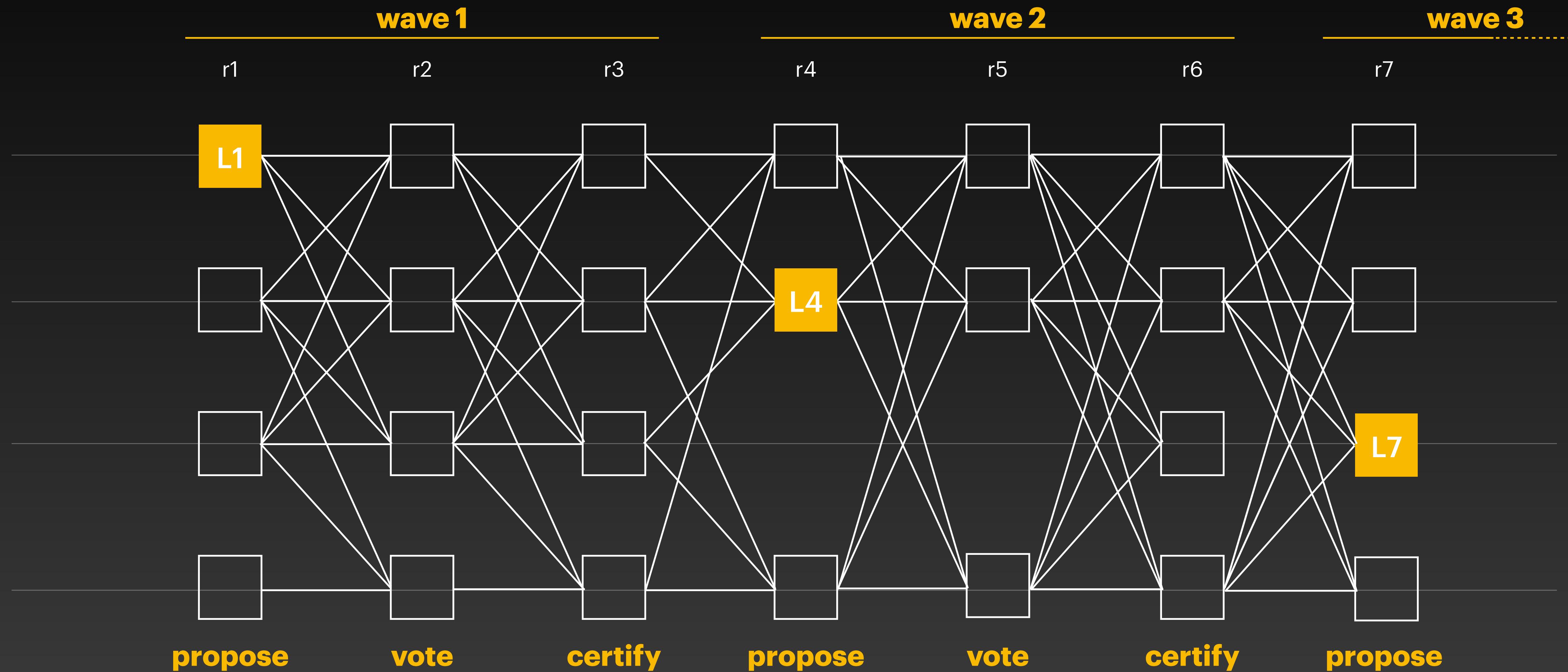


- We focus on ordering leaders: L1 L4 L7
- Linearising the sub-DAG is simple

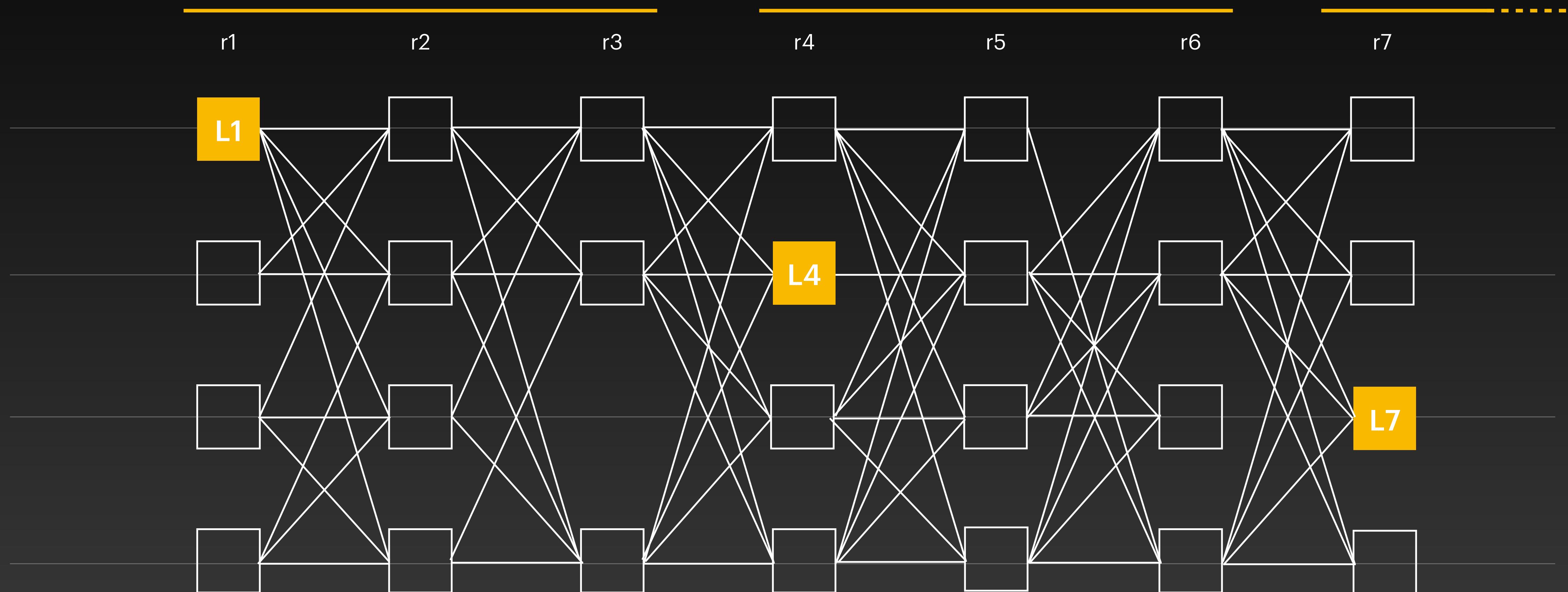
DAG Structure



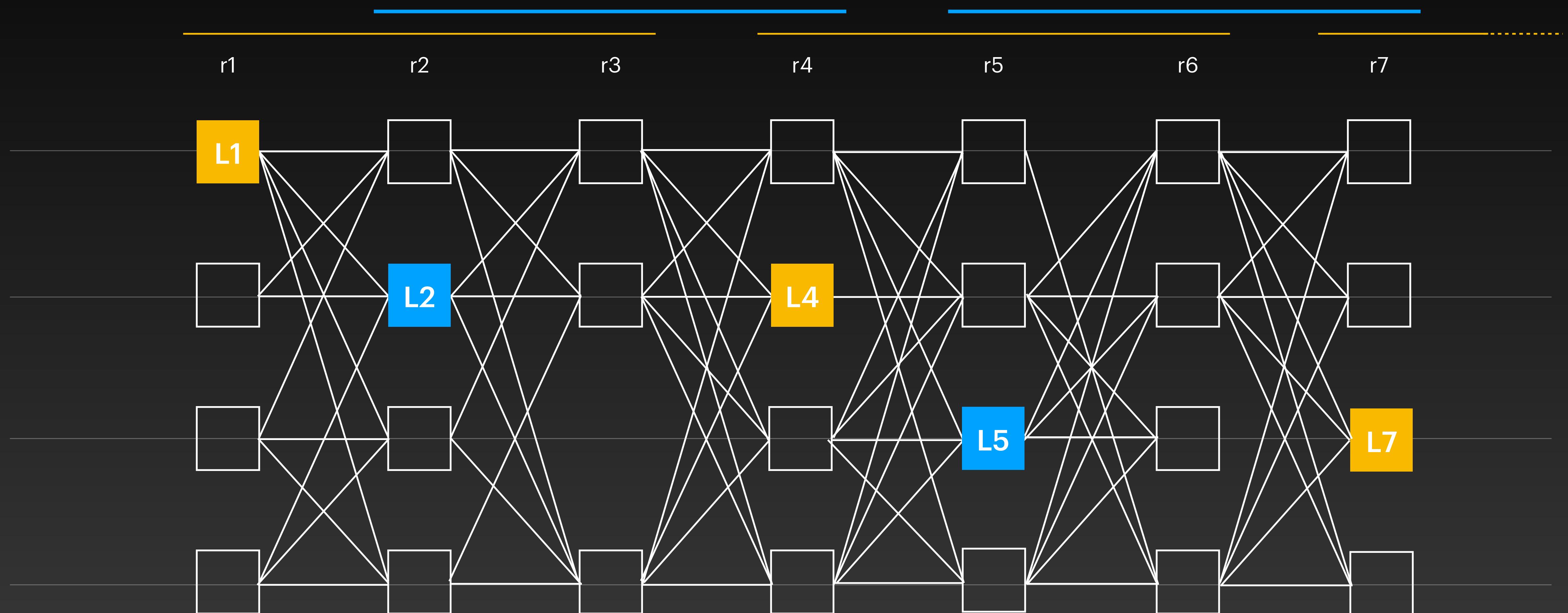
DAG Structure



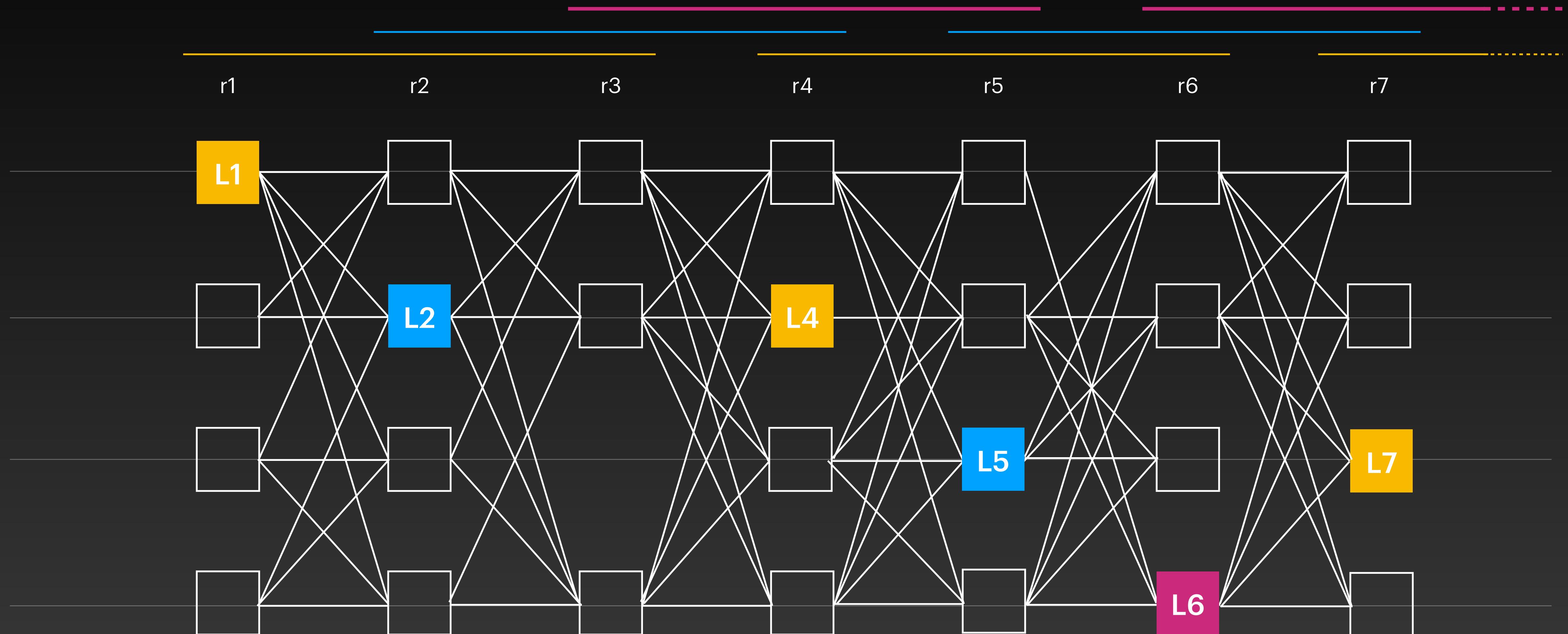
DAG Structure



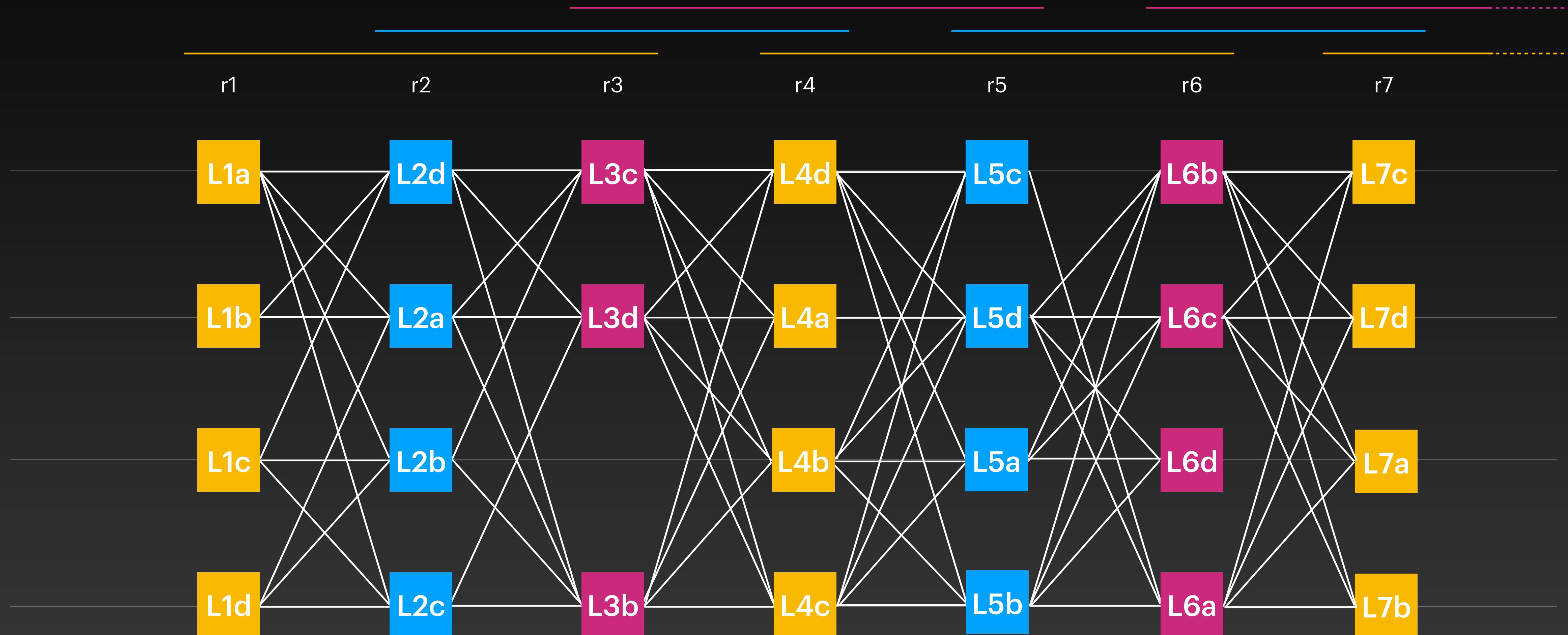
DAG Structure



DAG Structure

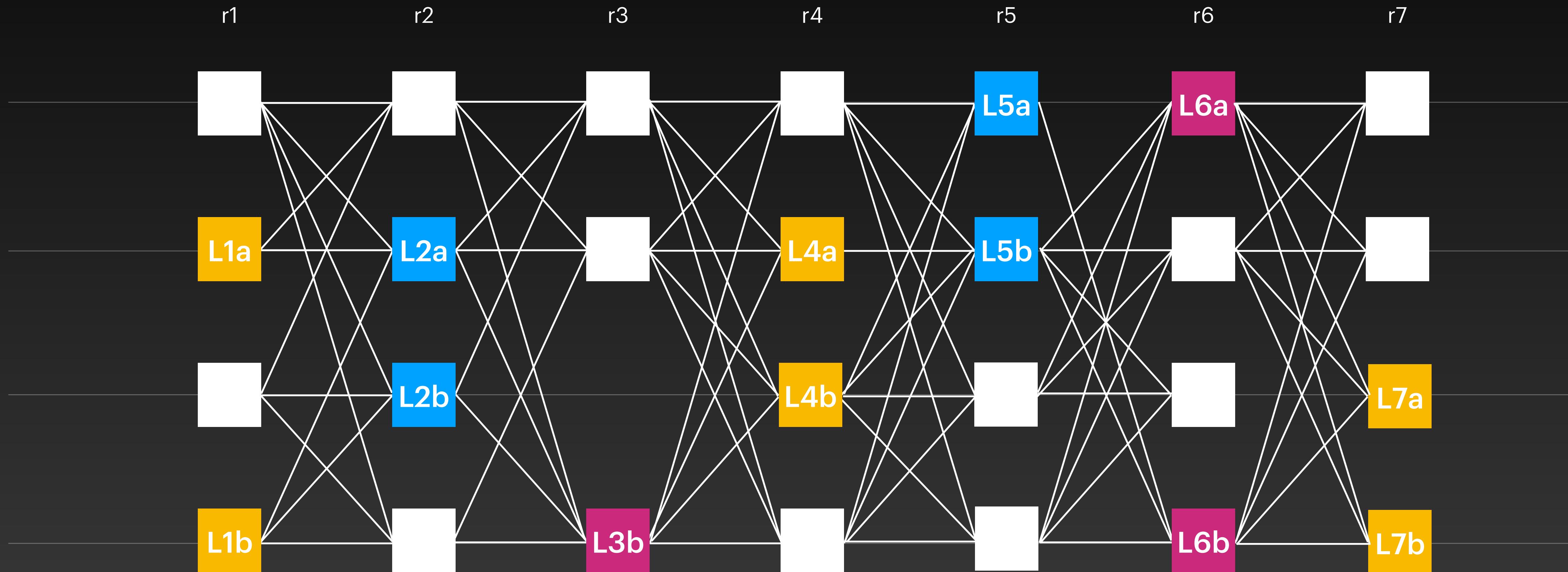


DAG Structure



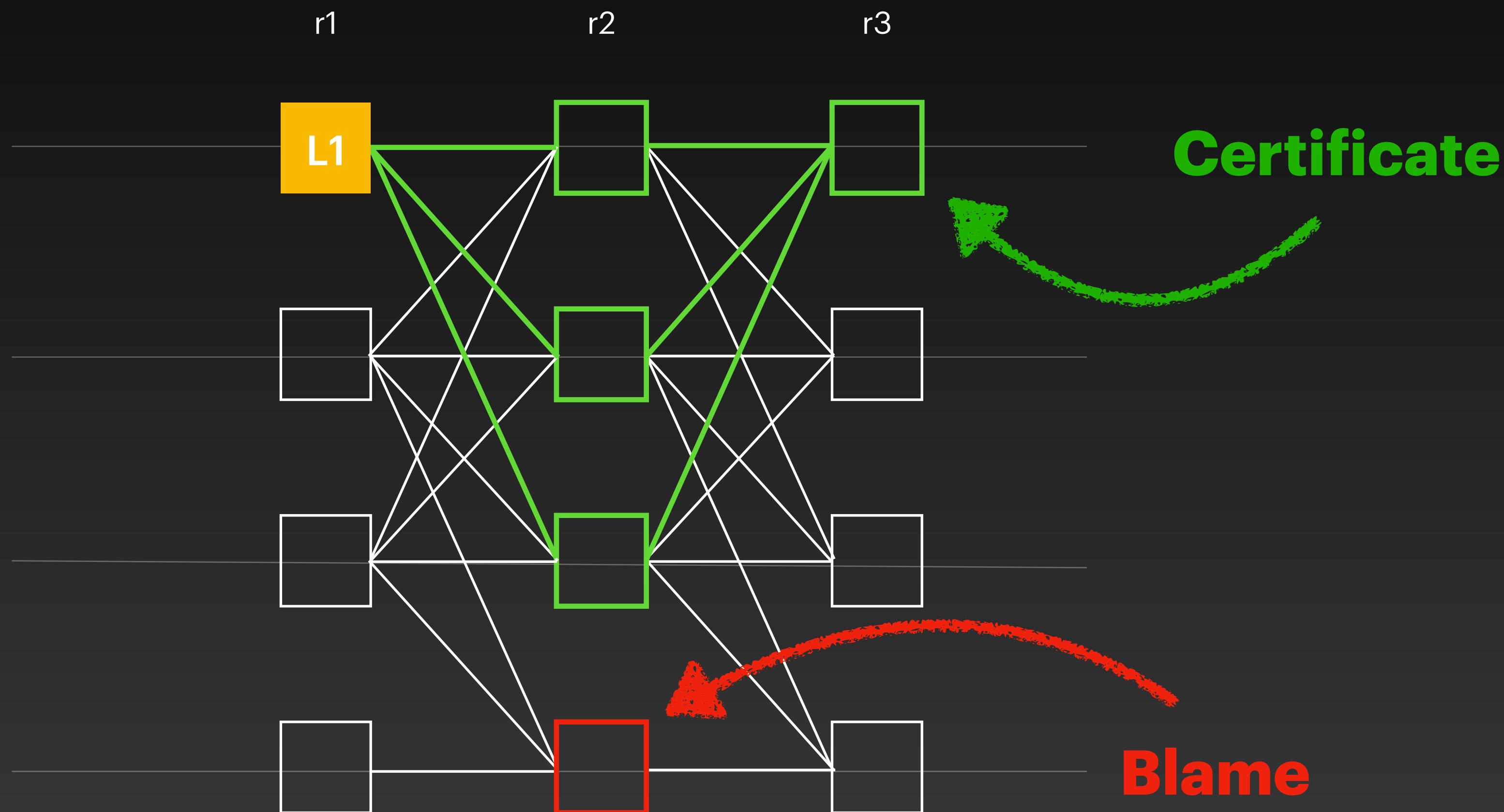
Practical Implementation

Select only 2 leaders per round



Interpreting DAG Patterns

Reminder



Direct Decision Rule

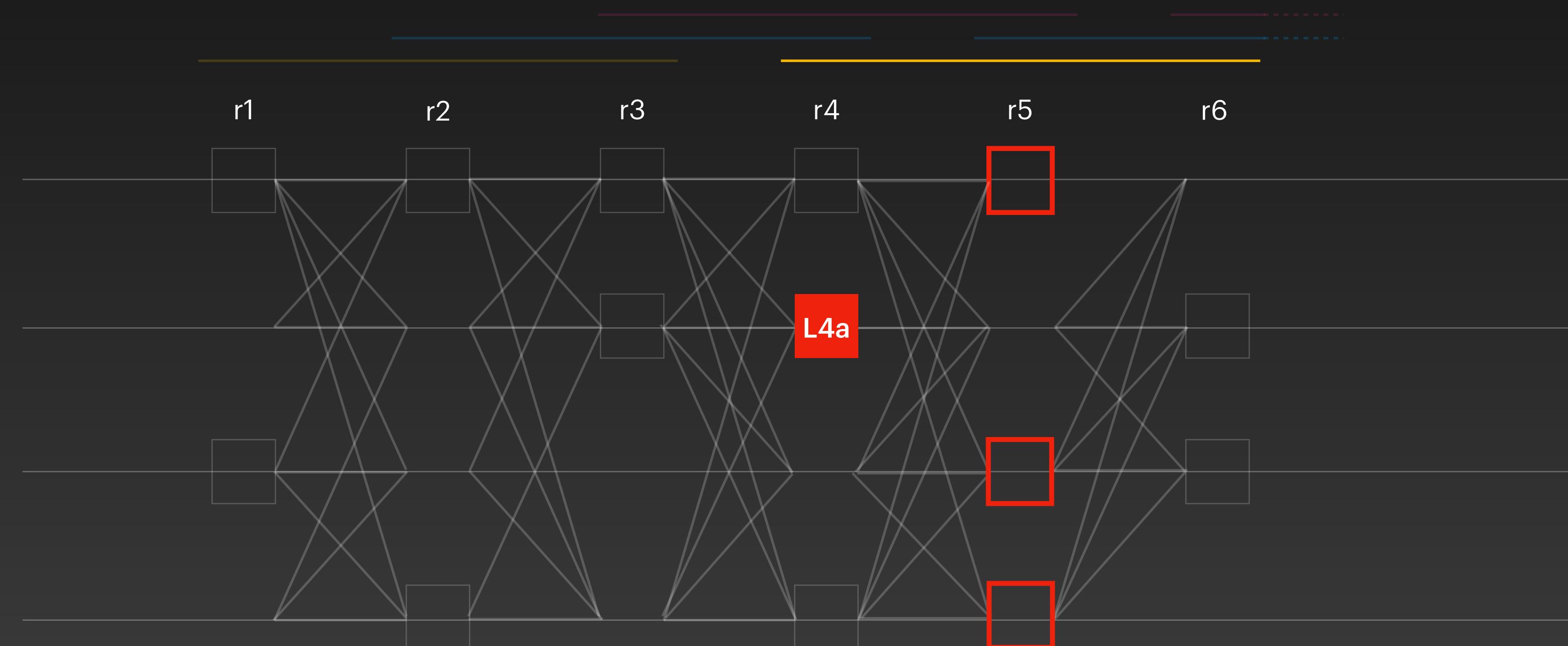
On each leader starting from highest round:

- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certificates
- **Undecided** otherwise

Direct Decision Rule

On each leader starting from highest round:

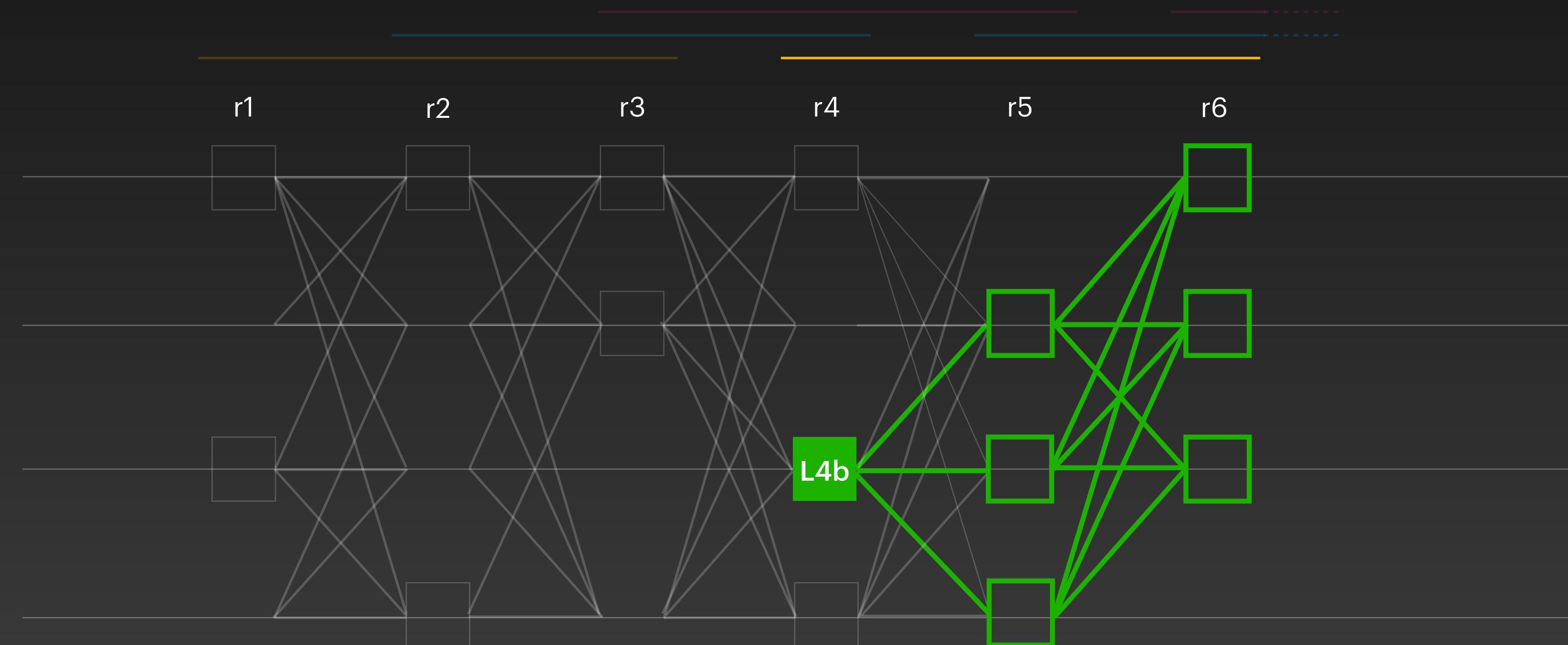
- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certificates
- **Undecided** otherwise



Direct Decision Rule

On each leader starting from highest round:

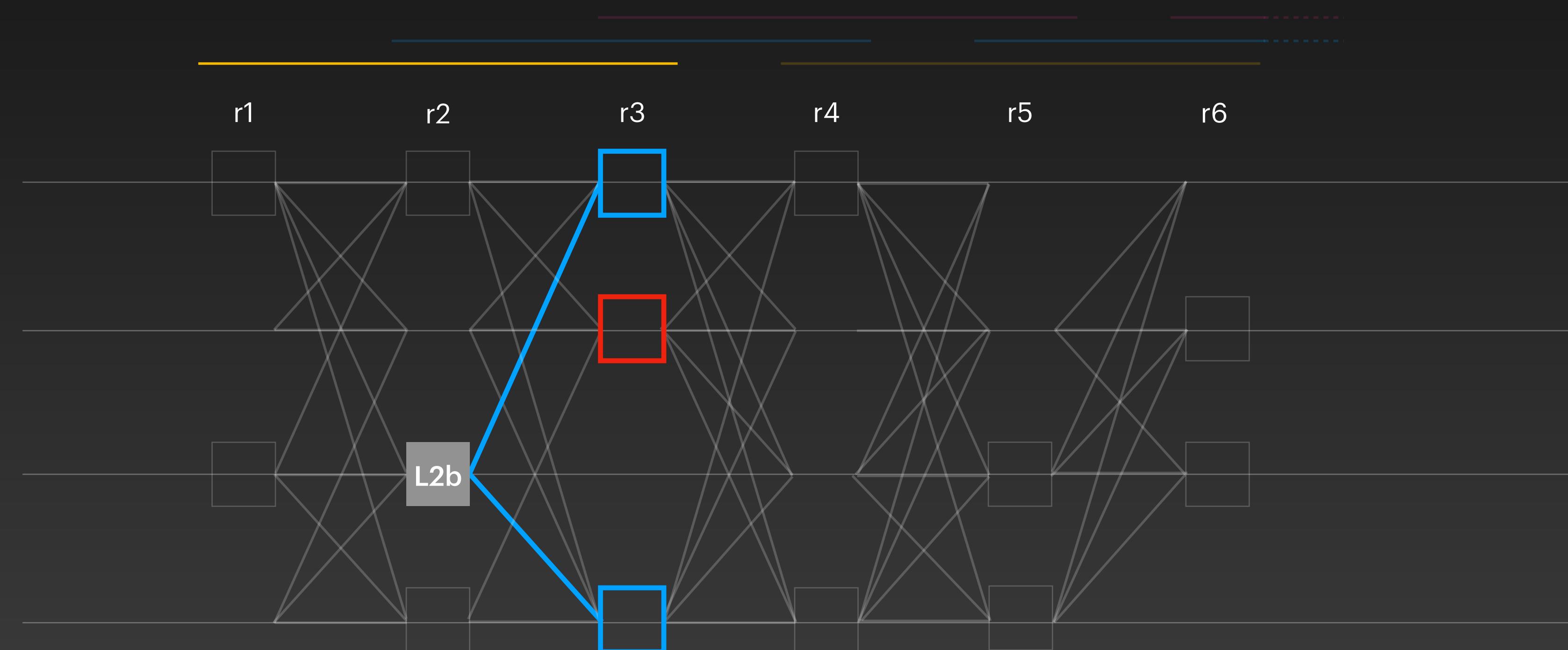
- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certificates
- **Undecided** otherwise



Direct Decision Rule

On each leader starting from highest round:

- **Skip** if $2f+1$ blames
- **Commit** if $2f+1$ certificates
- **Undecided** otherwise

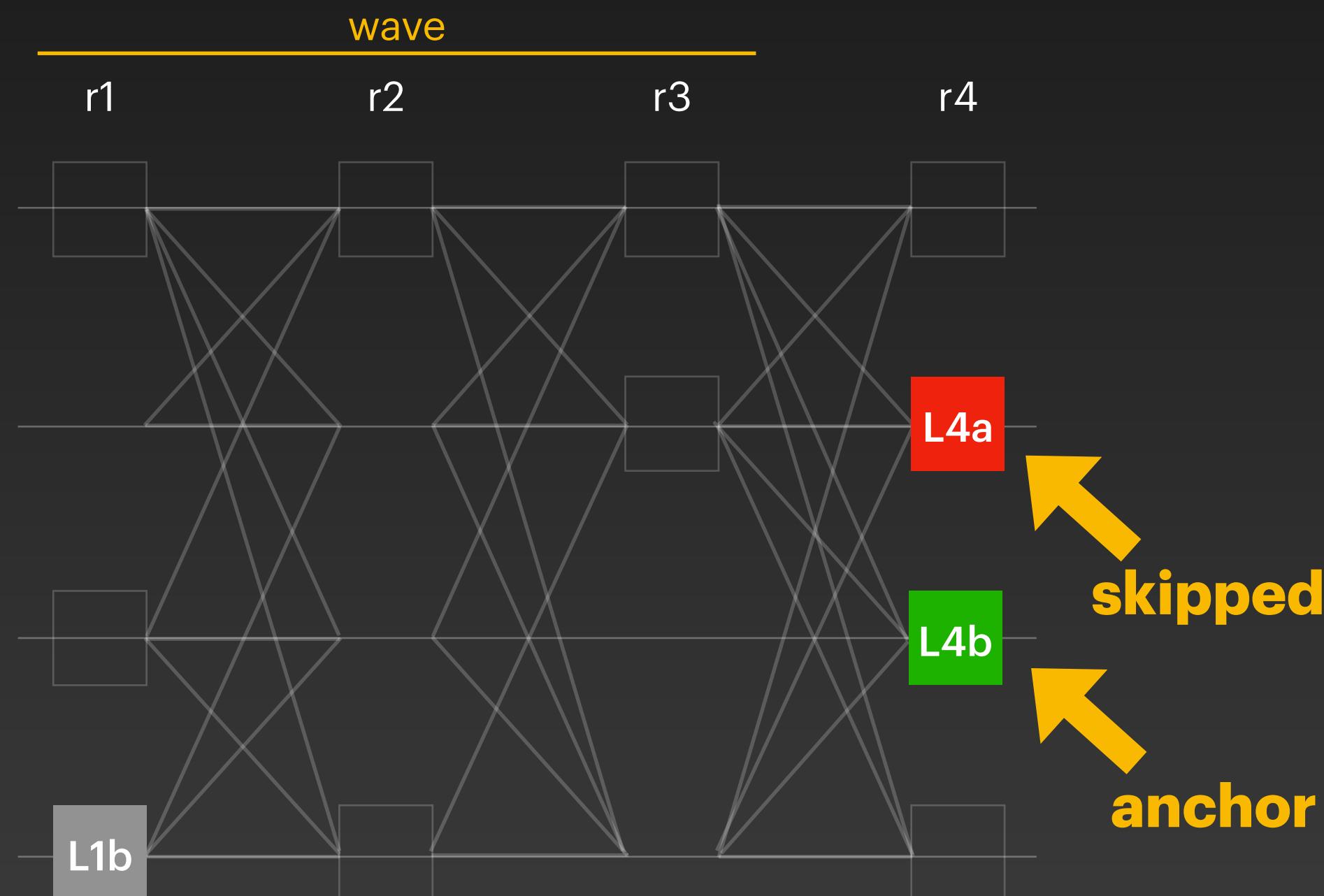


Indirect Decision Rule

Indirect Decision Rule

1. Find Anchor

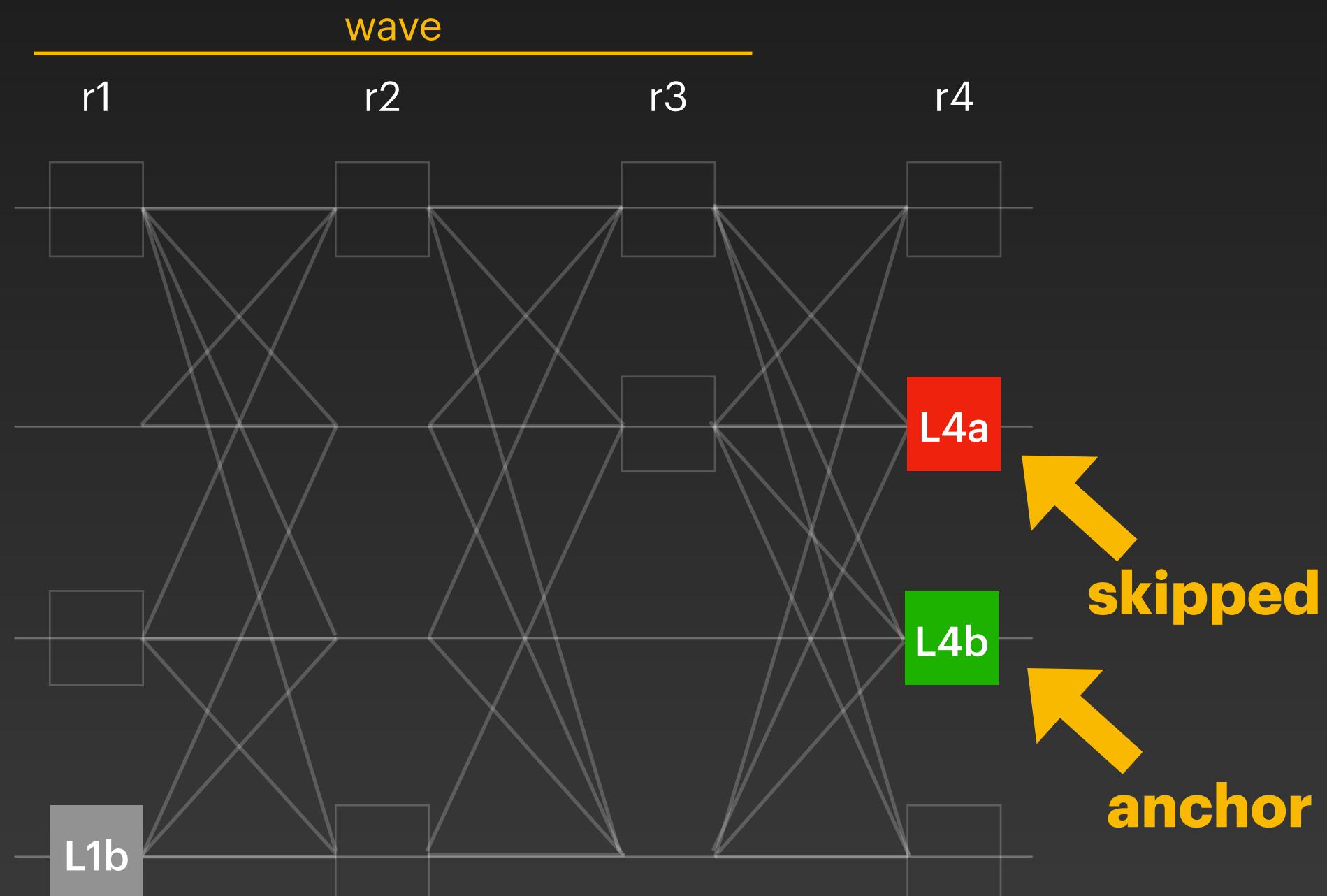
- First block with round $> r+2$ that is **Commit** or **Undecided**



Indirect Decision Rule

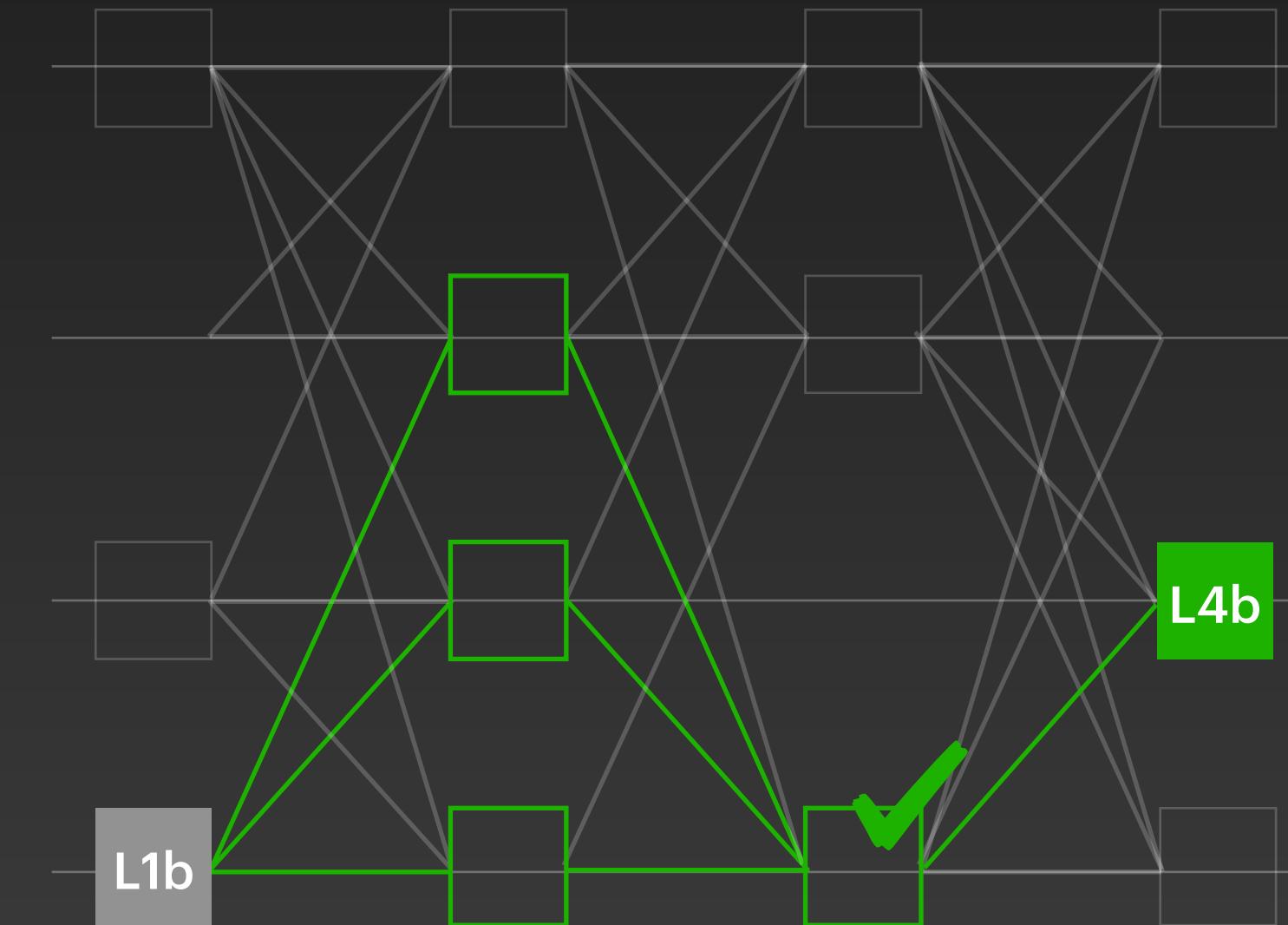
1. Find Anchor

- First block with round $> r+2$ that is **Commit** or **Undecided**

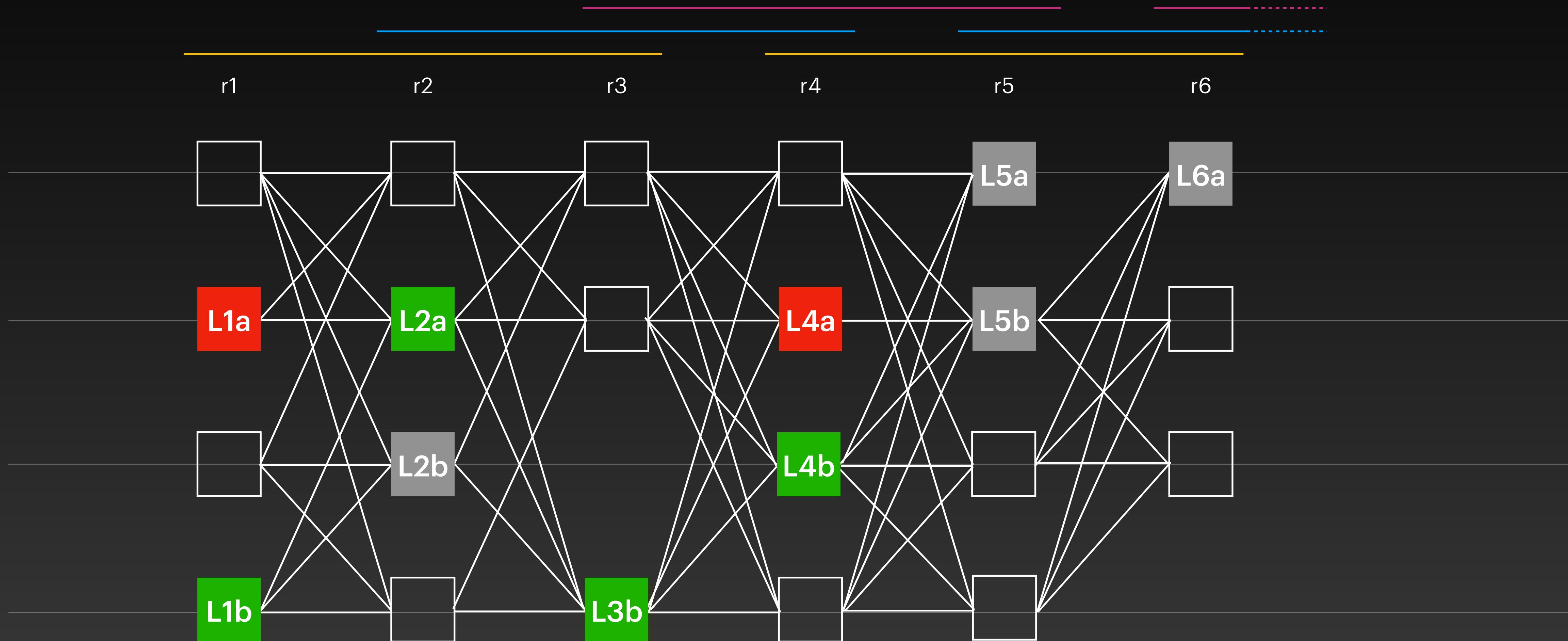


2. Certified link

- Commit** if $B <->$ certified link $<-> A$
otherwise **Skip**

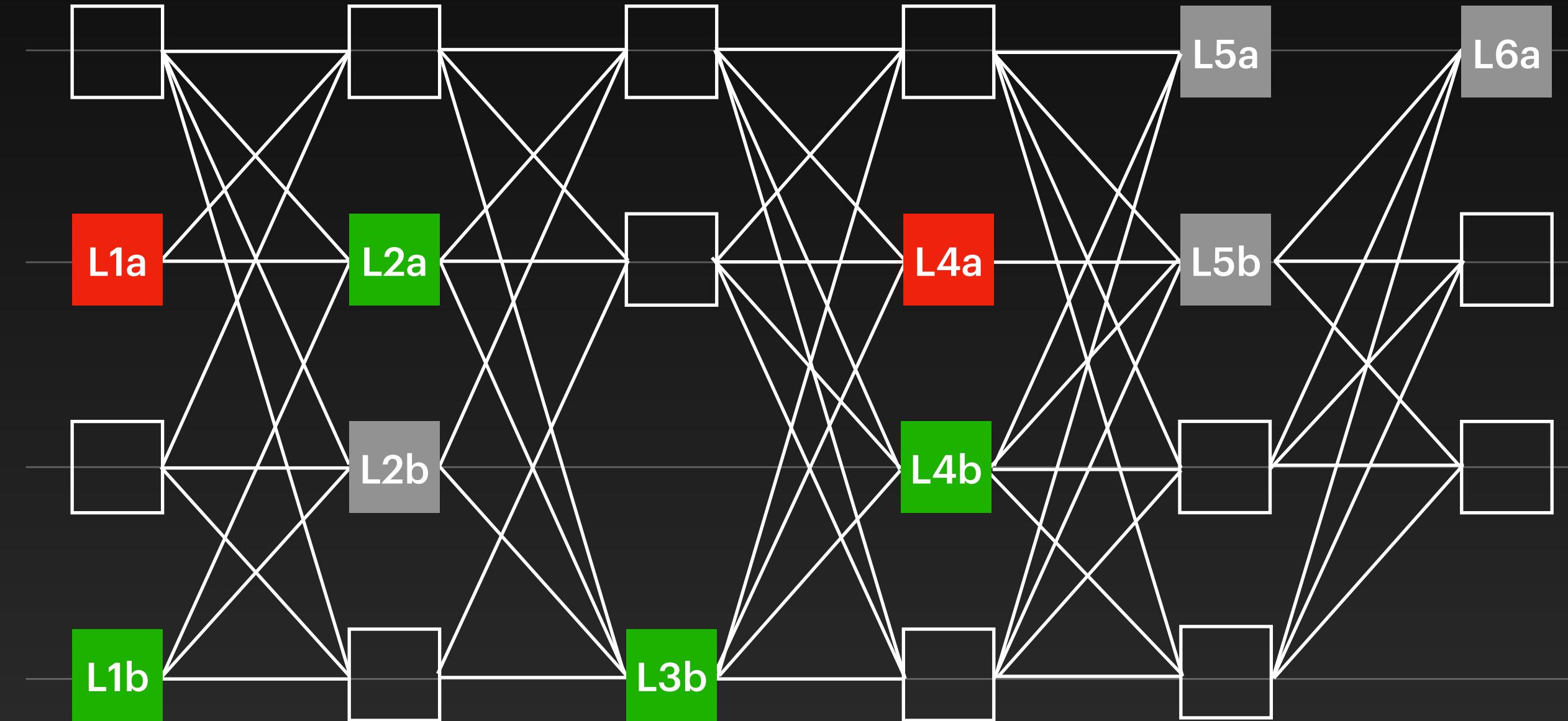


Current Status



Commit Sequence

Take all leaders in order

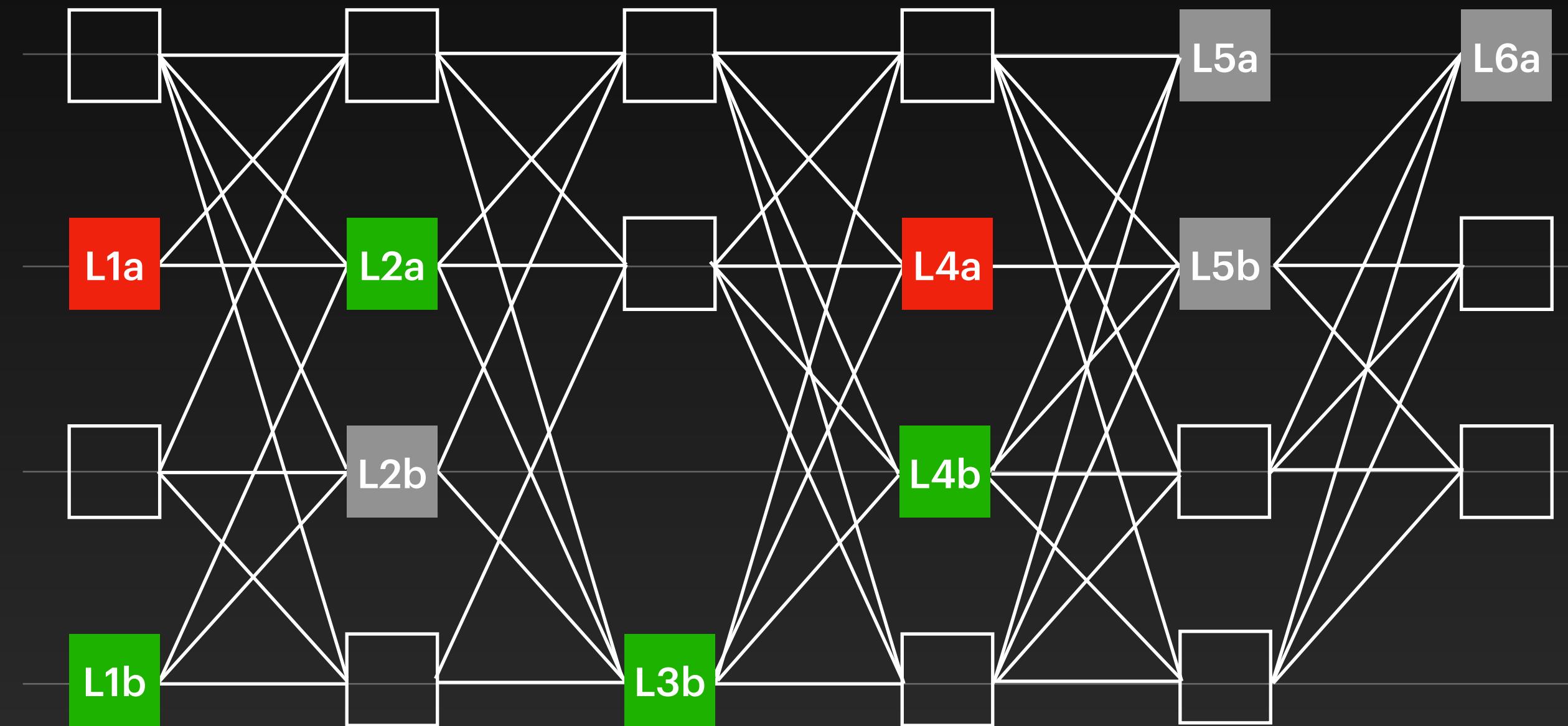


leaders sequence:

L1a L1b L2a L2b L3b L4a L4b

Commit Sequence

Stop at the first Undecided leader

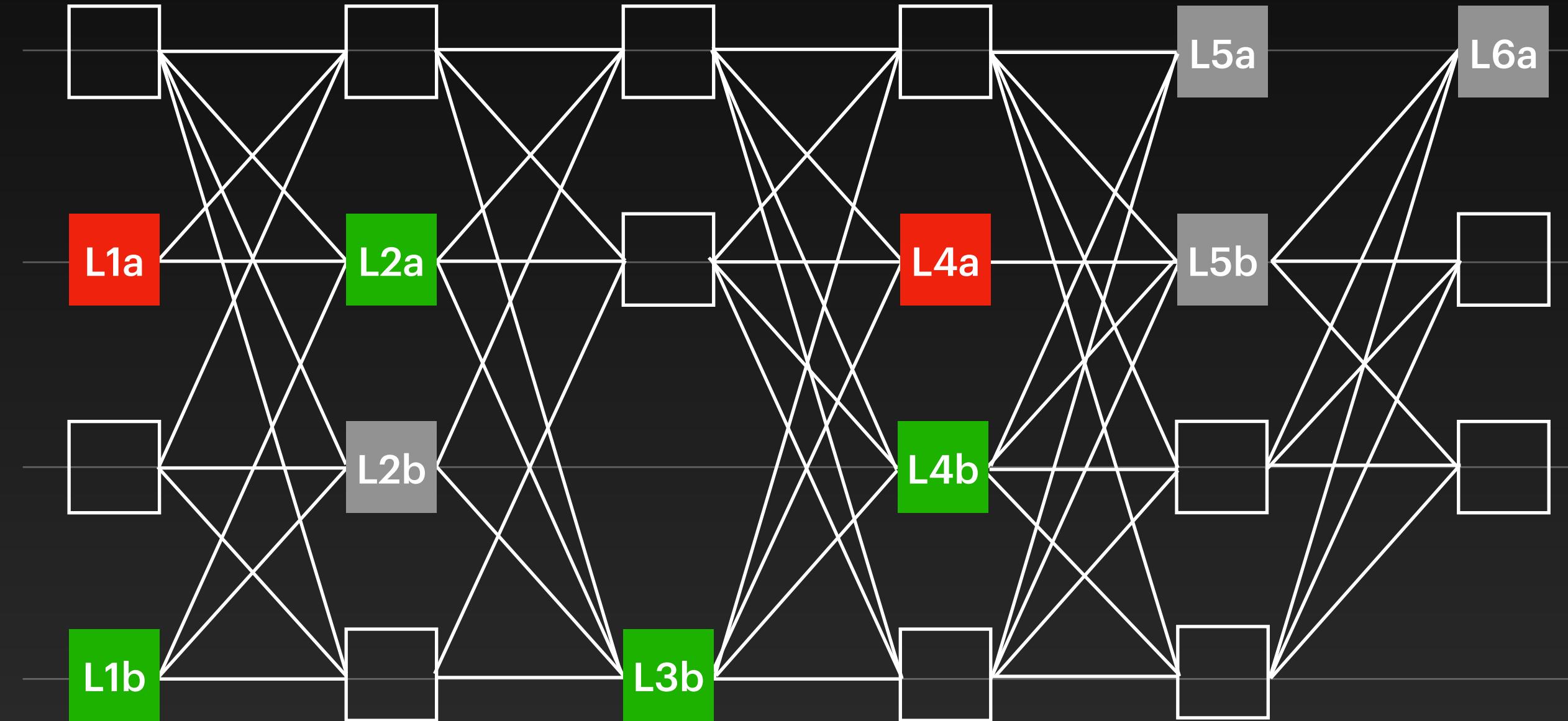


leaders sequence:

L1a L1b L2a |

Commit Sequence

Remove skipped leaders

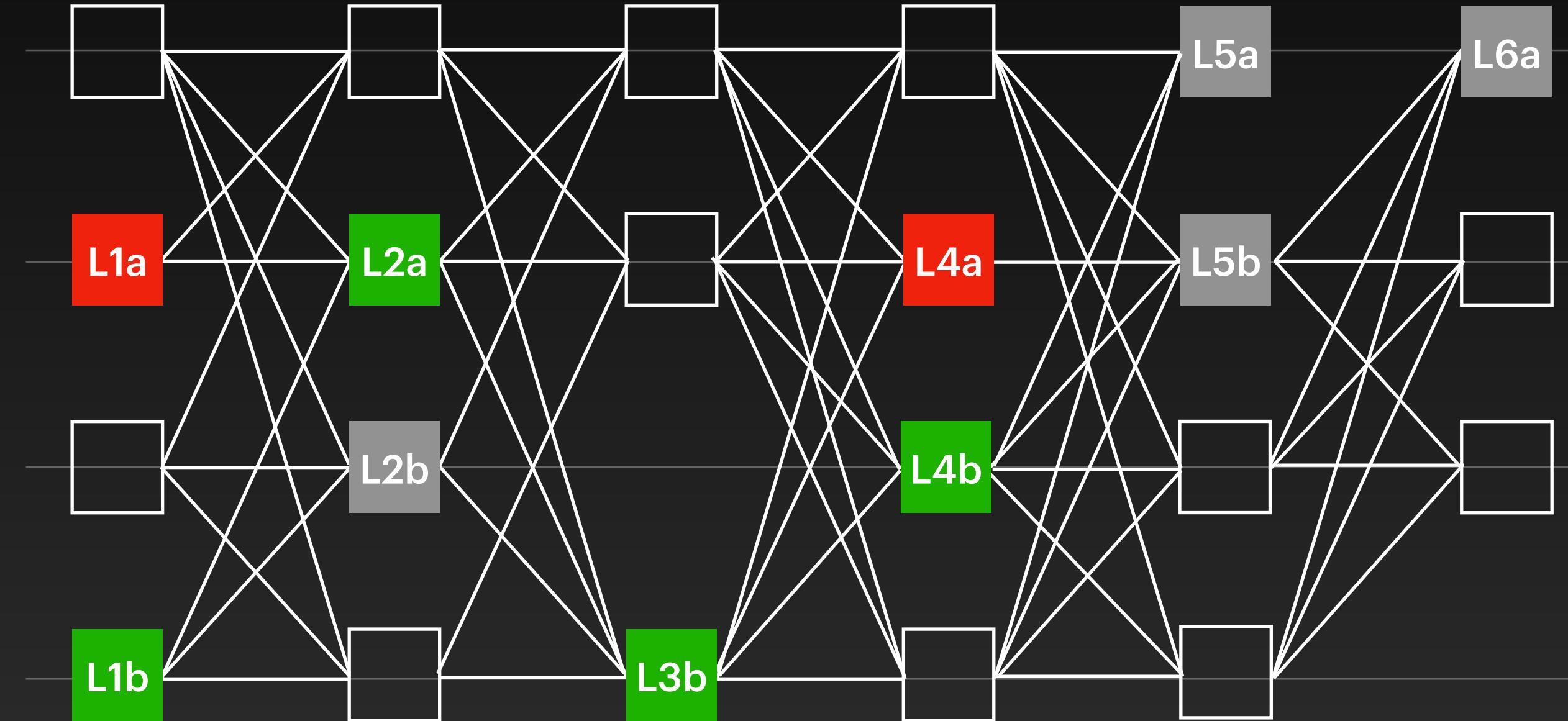


leaders sequence:

L1b L2a

Commit Sequence

Final leader sequence

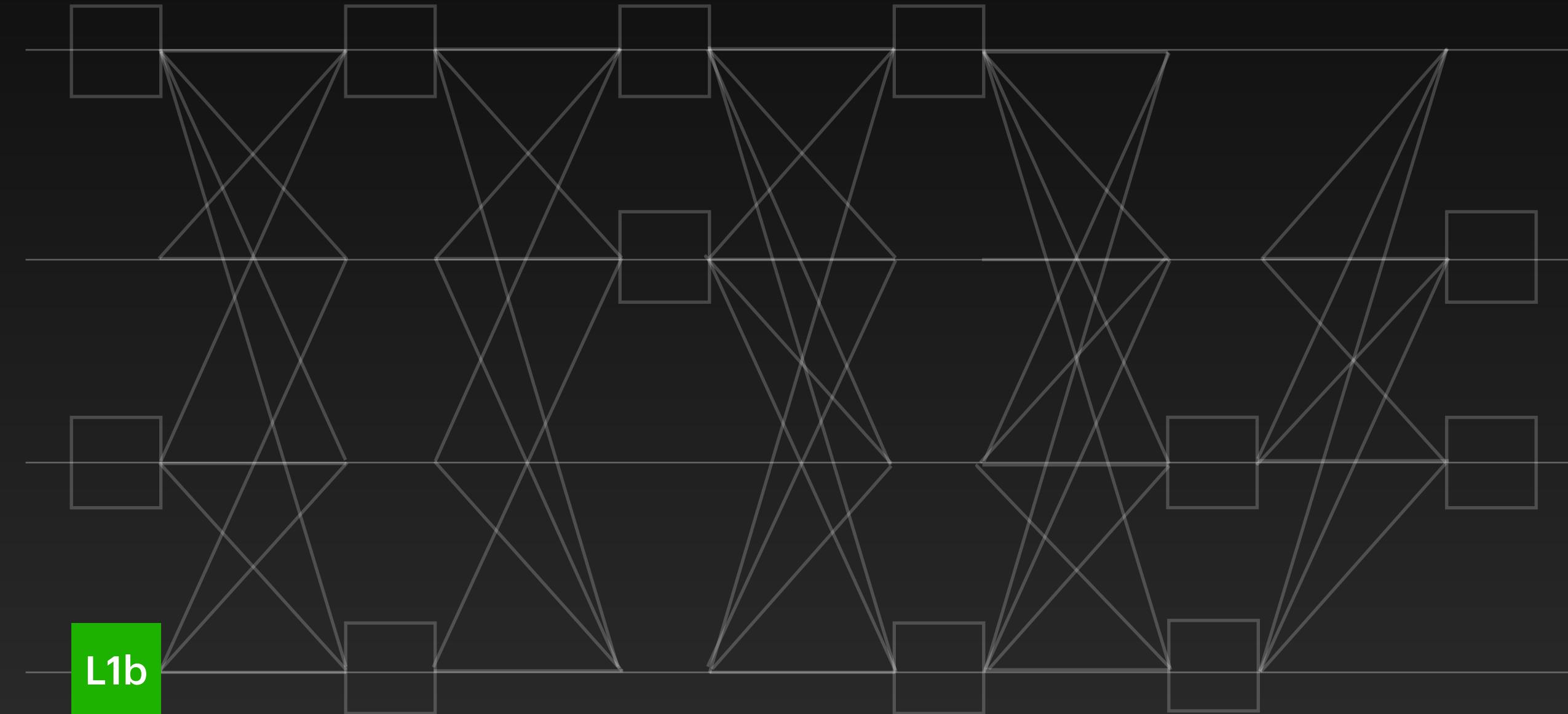


leaders sequence:

L1b L2a

Commit Sequence

Commit sub-dag



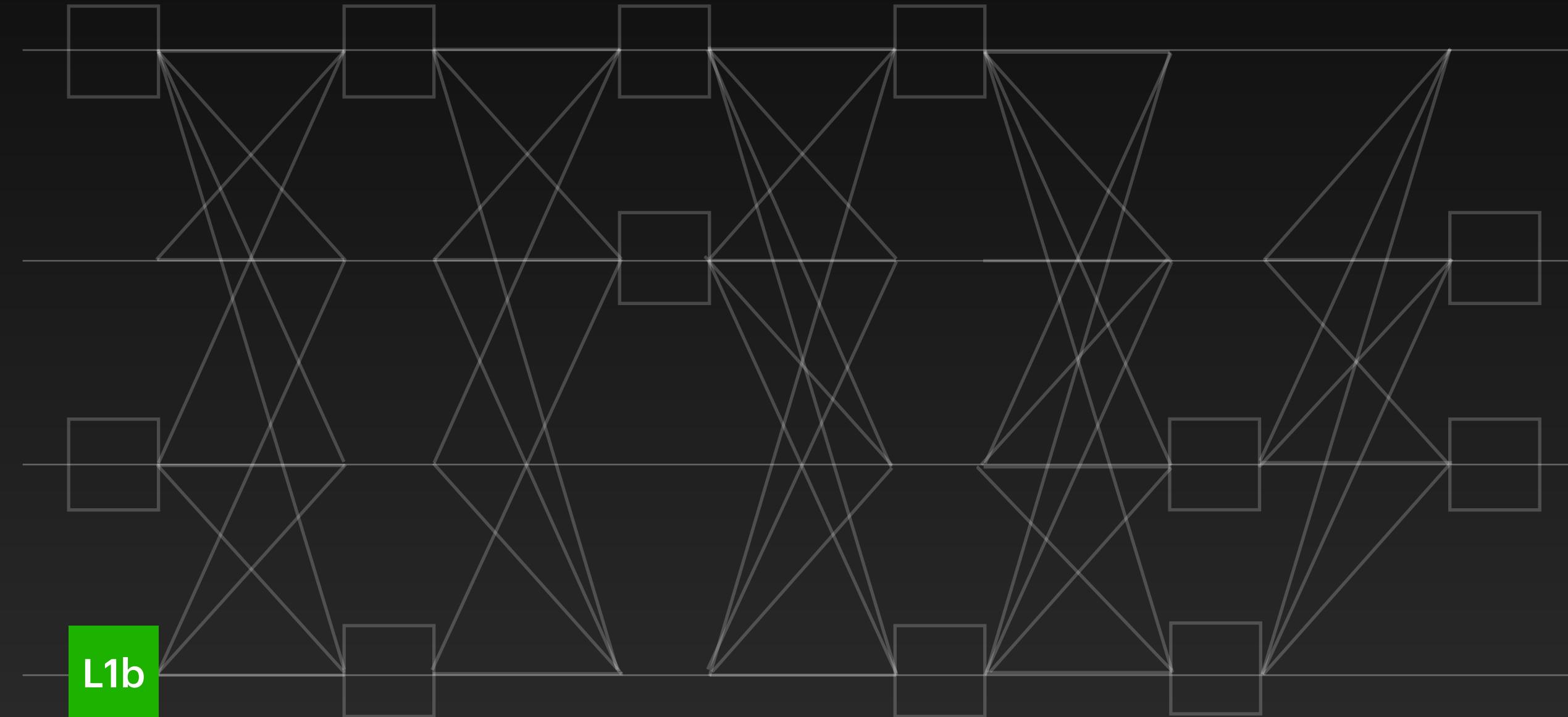
leaders sequence:

L1b

output sequence:

Commit Sequence

Commit sub-dag



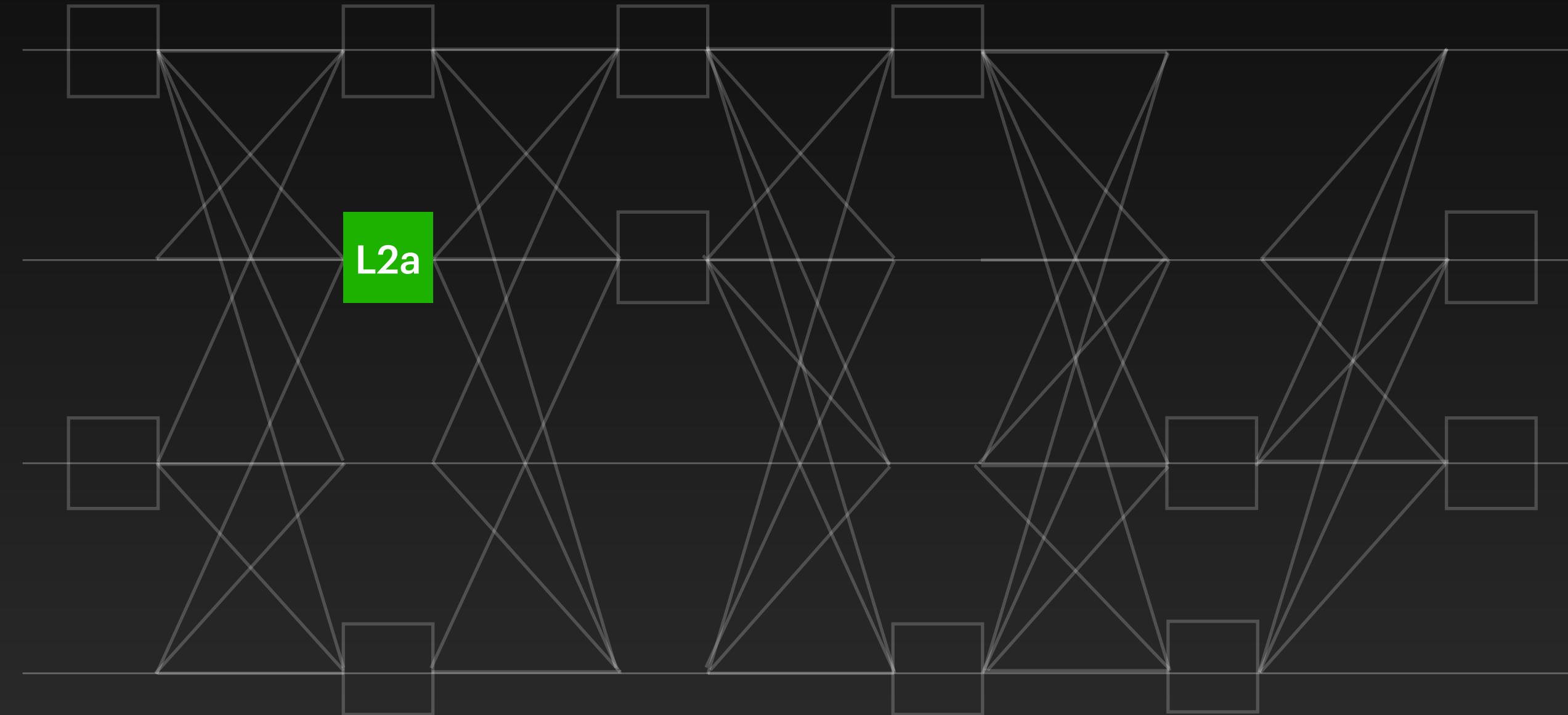
leaders sequence:

output sequence:

L1b

Commit Sequence

Commit sub-dag



leaders sequence:

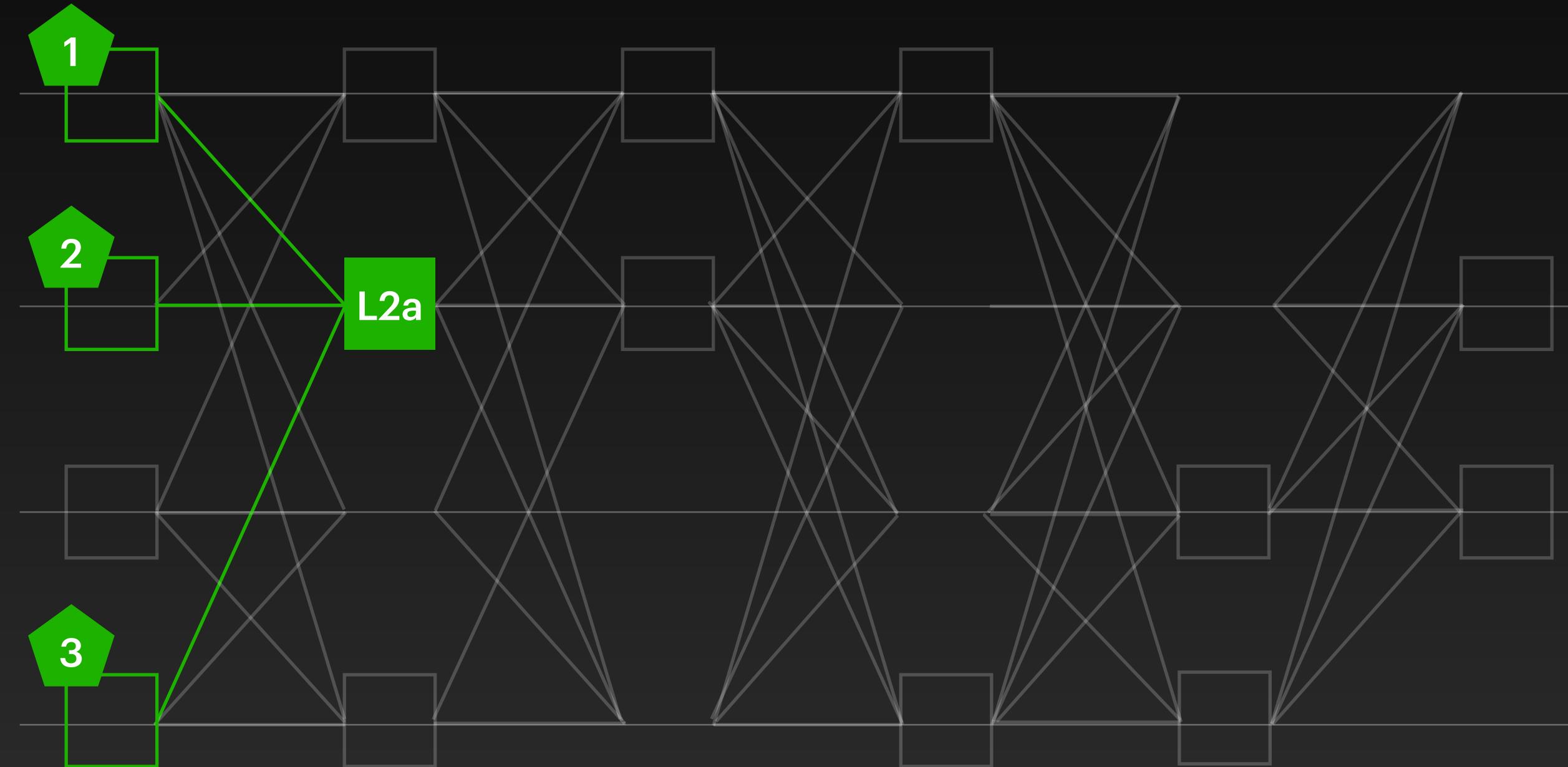
L2a

output sequence:

L1b

Commit Sequence

Commit sub-dag



leaders sequence:

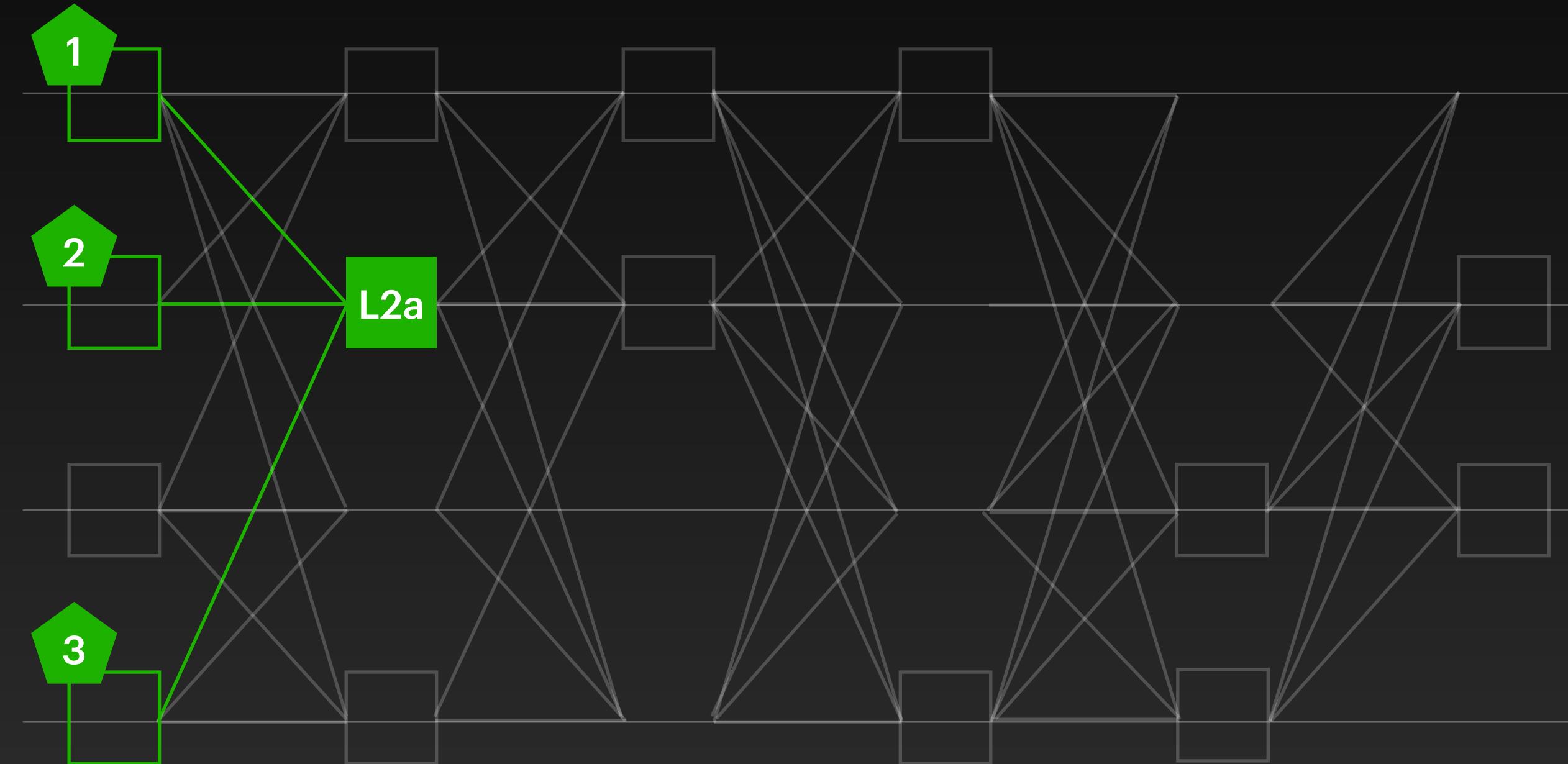
L2a

output sequence:

L1b

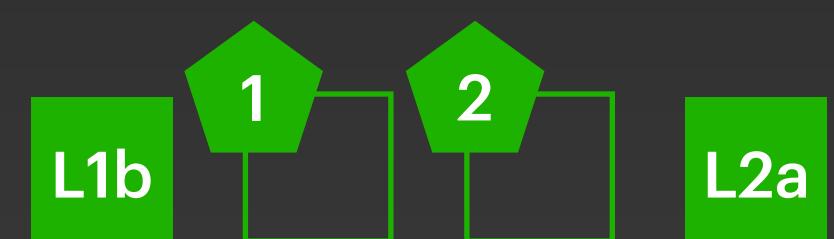
Commit Sequence

Commit sub-dag



leaders sequence:

output sequence:



Benchmarks

More to come soon

Implementation

- Written in Rust
- Networking: Tokio (TCP)
- Storage: custom WAL
- Cryptography: ed25519-consensus

<https://github.com/mystenlabs/mysticeti>

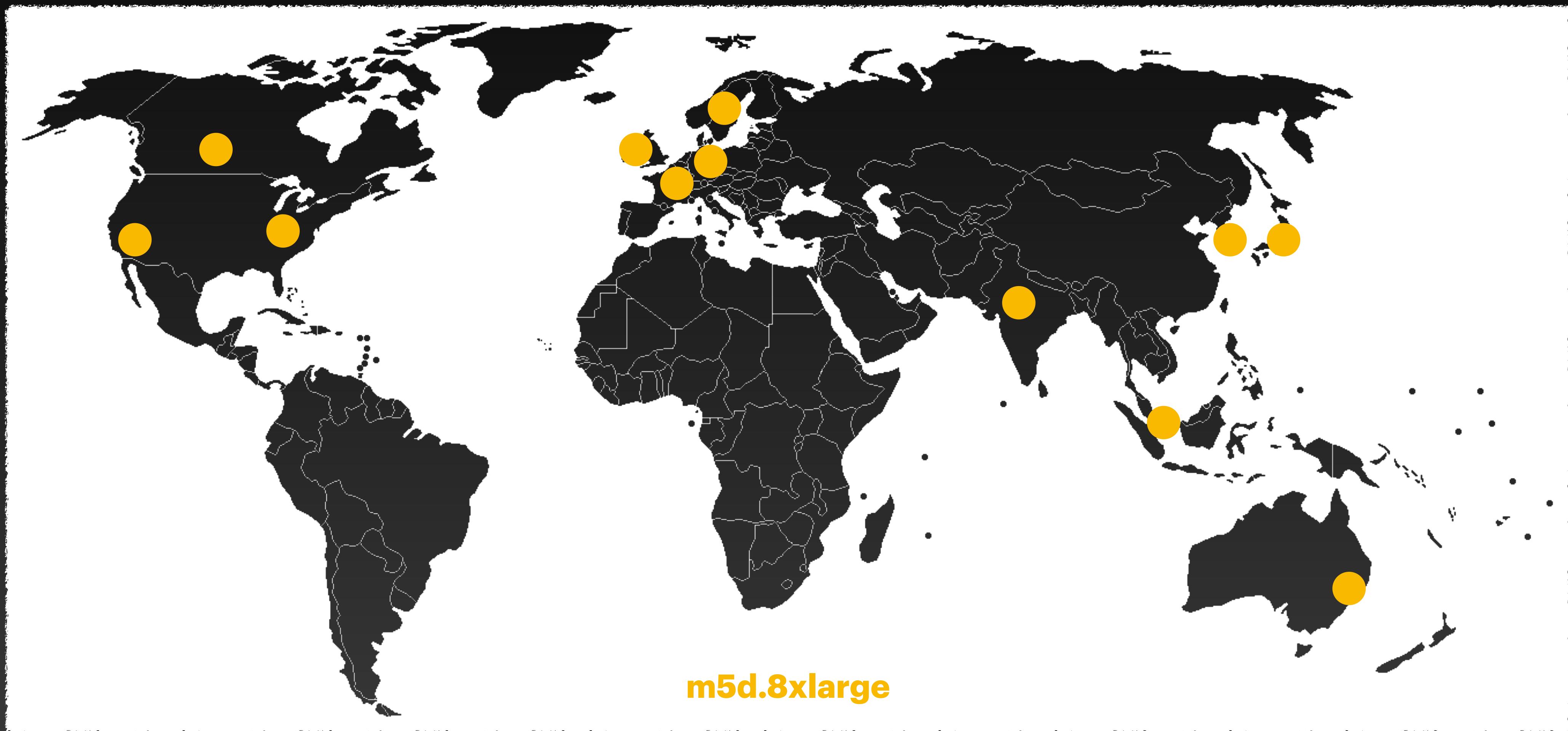
Implementation

- Synchronous core
- One Tokio task per peer (limiting resource usage)
- DTE simulator

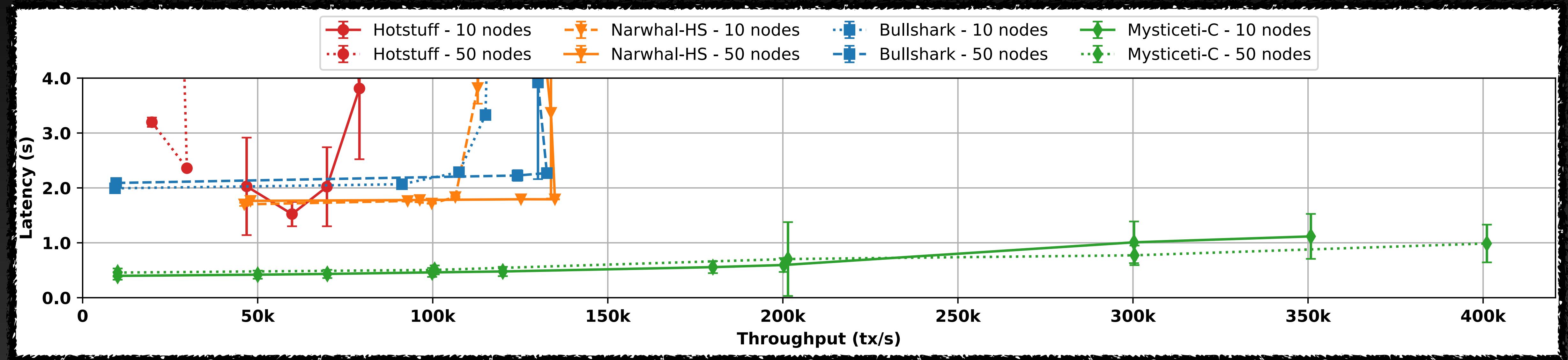
<https://github.com/mystenlabs/mysticeti>

Evaluation

Experimental setup on AWS



Preliminary Results



Testing Strategy



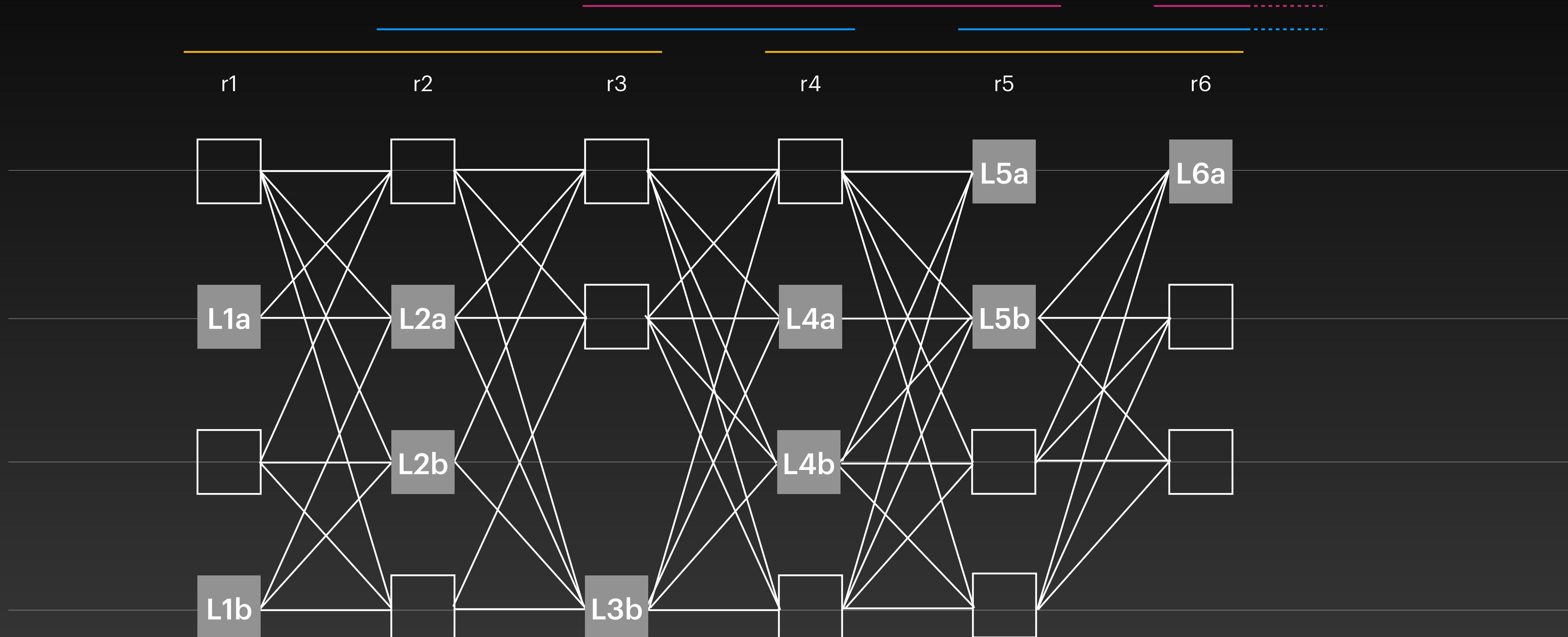
- Compare performance & robustness
- Test mainnet change bullshark -> mysticeti
- Prepare for the worst mysticeti -> bullshark

alberto@mystenlabs.com

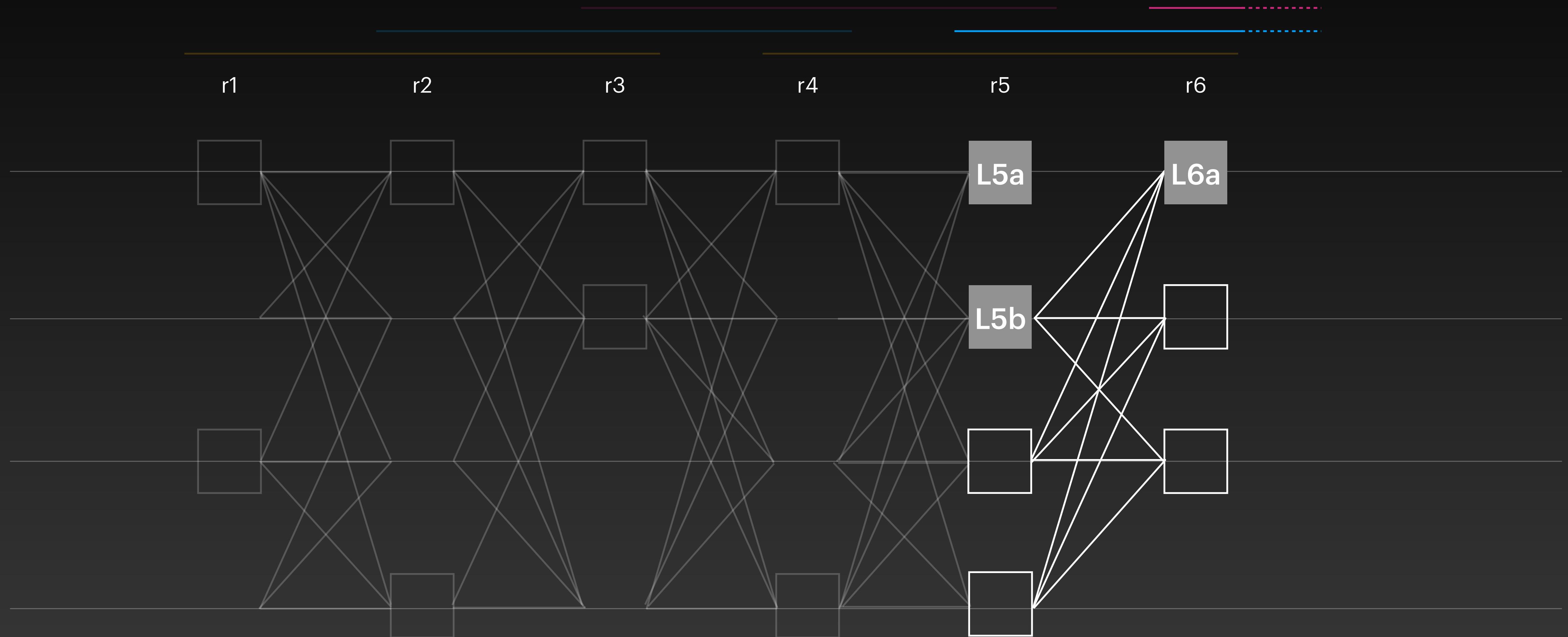
Example

Step-by-step

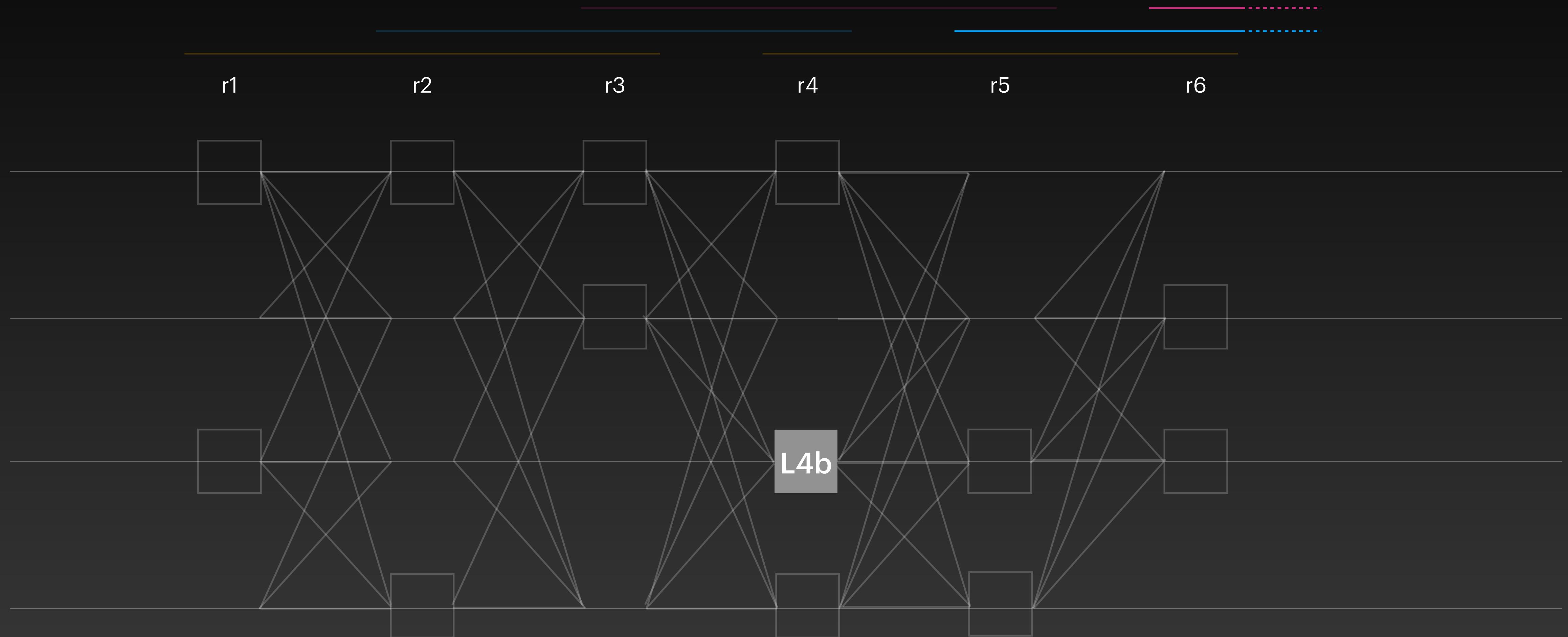
All Start at Undecided



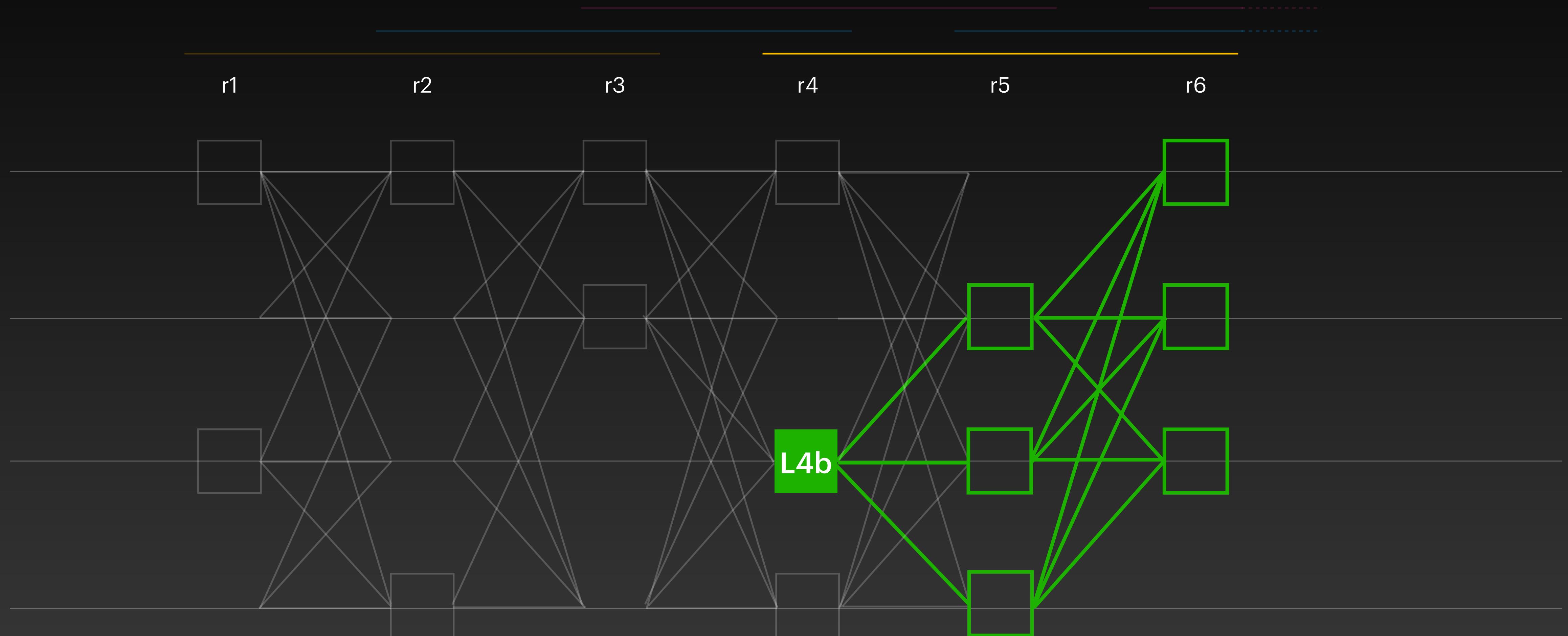
Ignore Incomplete Waves



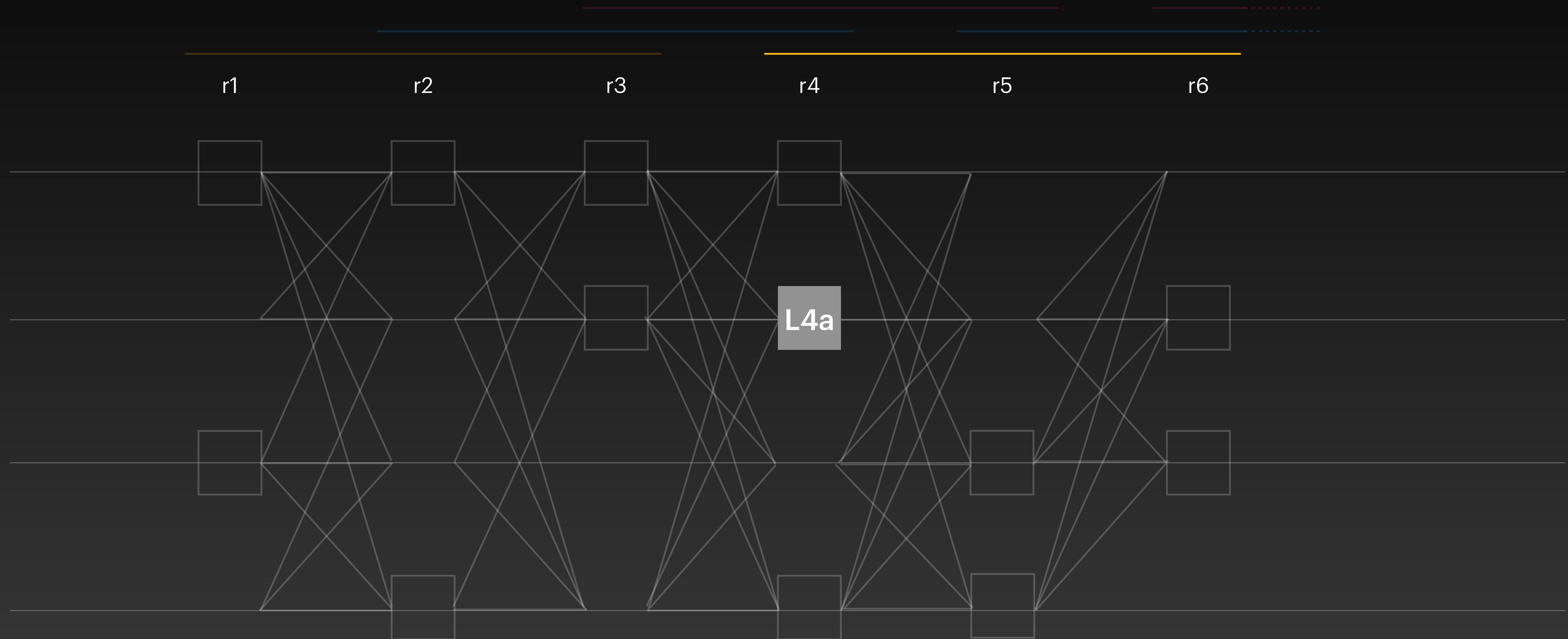
Apply Direct Rule



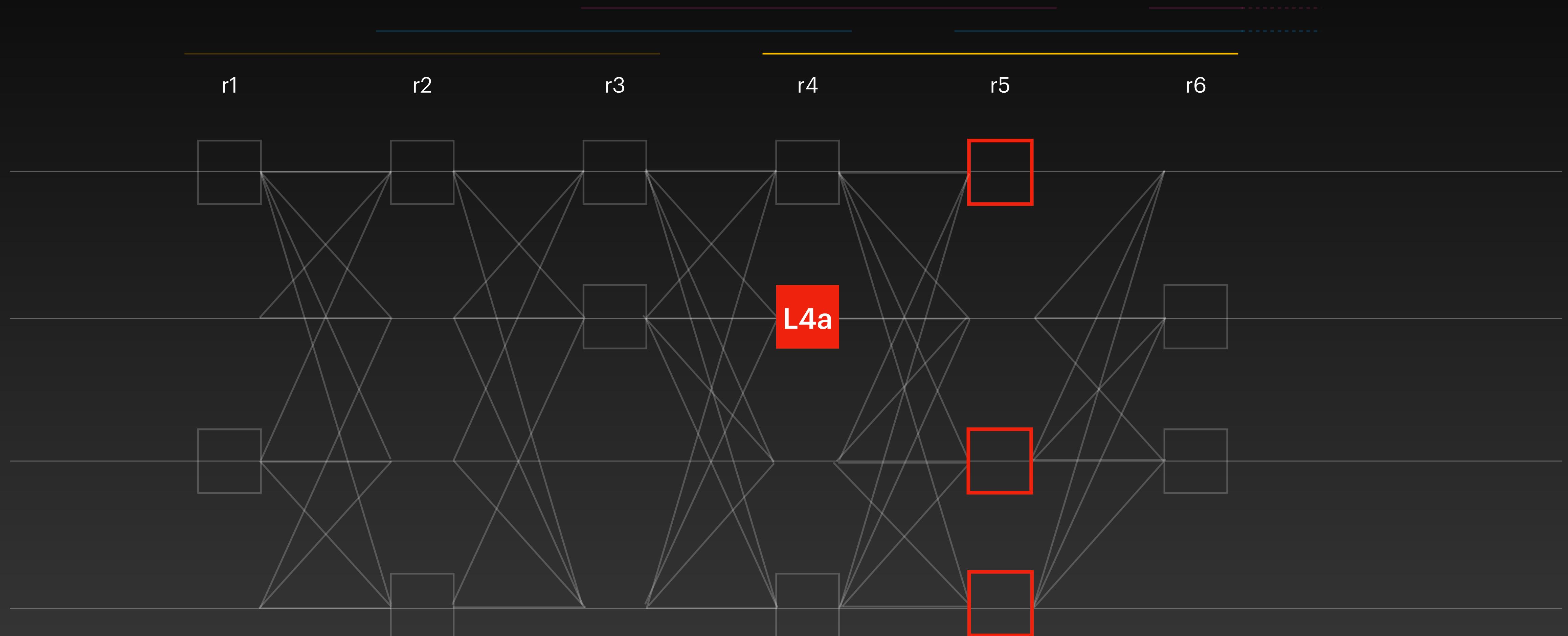
Apply Direct Rule



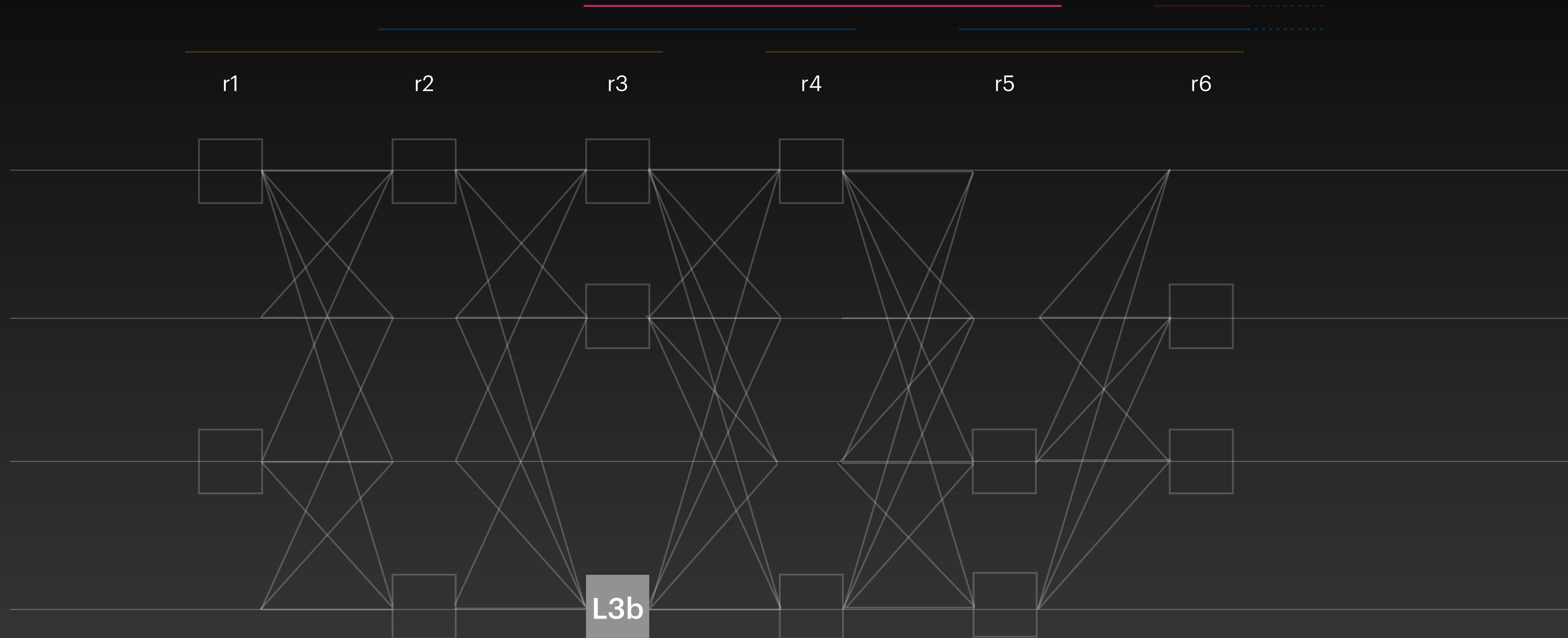
Apply Direct Rule



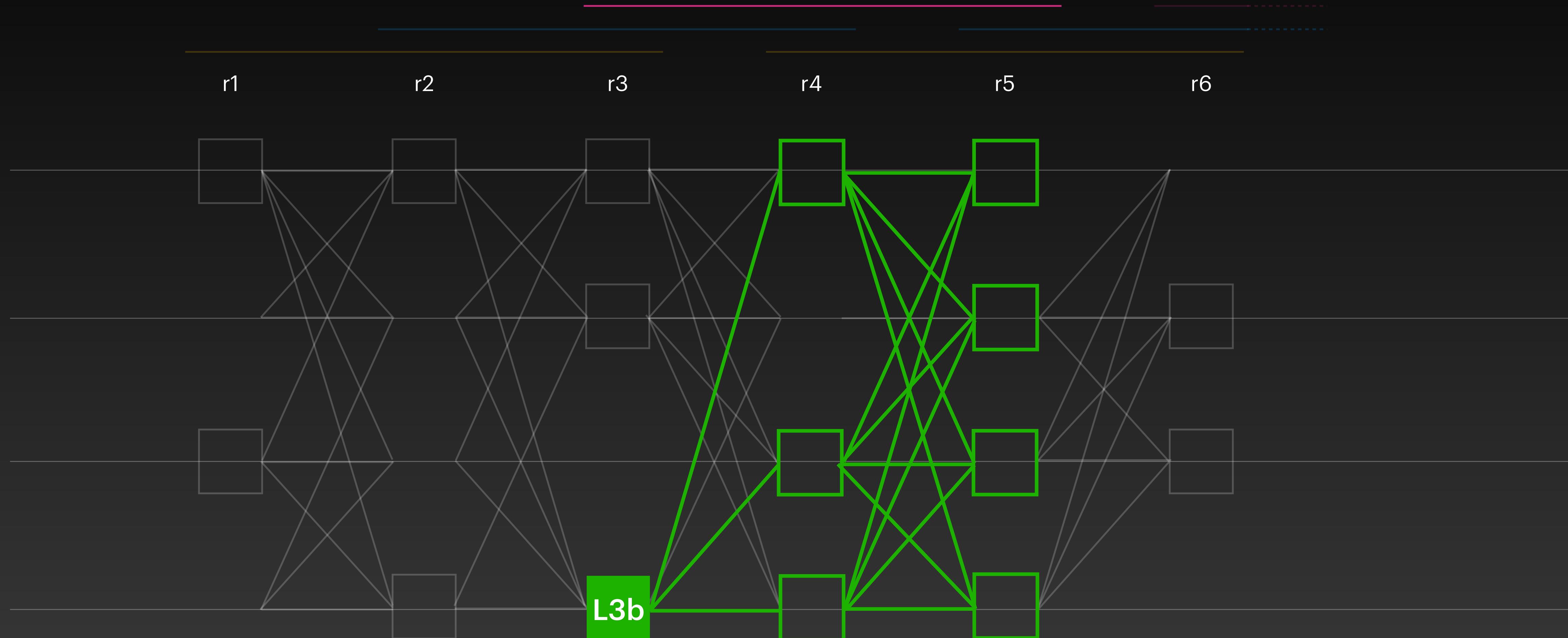
Apply Direct Rule



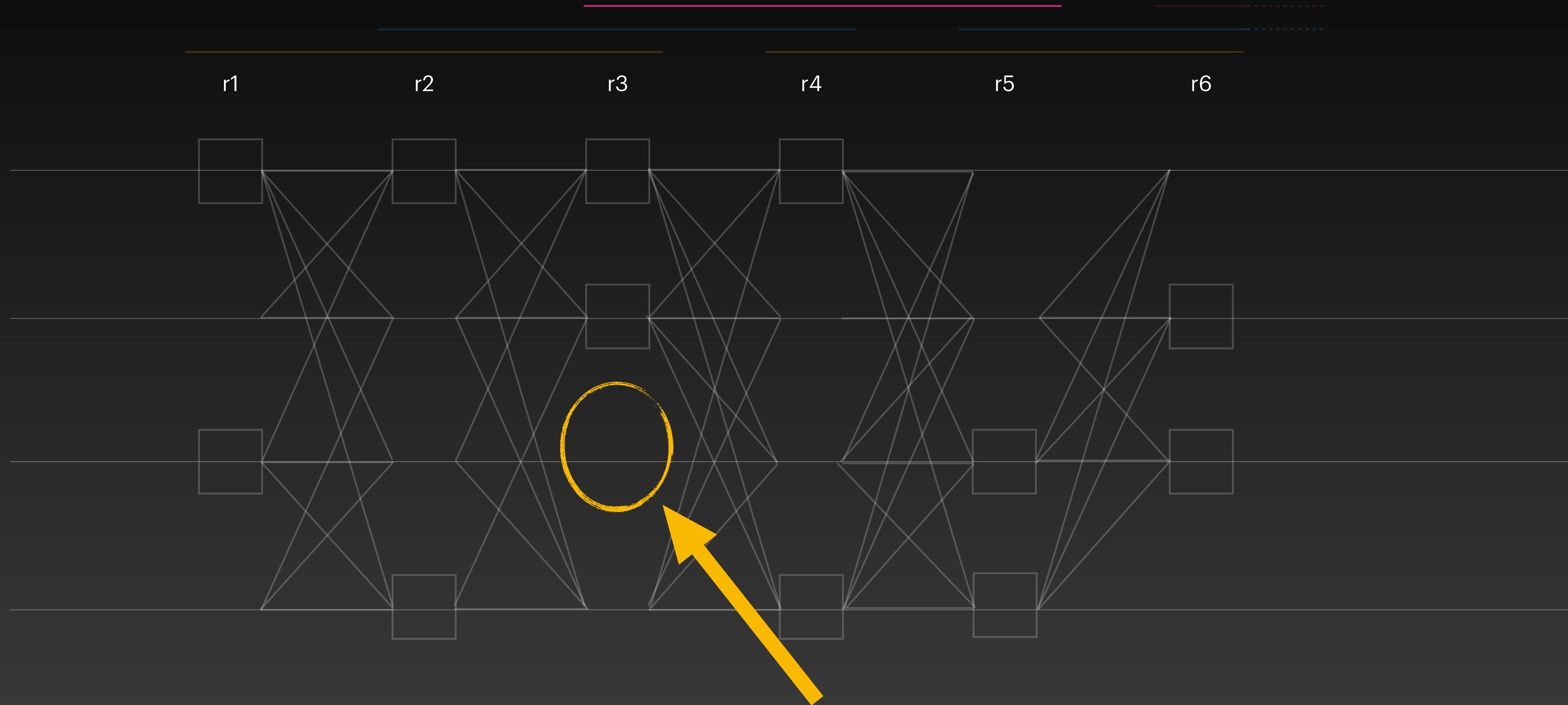
Apply Direct Rule



Apply Direct Rule



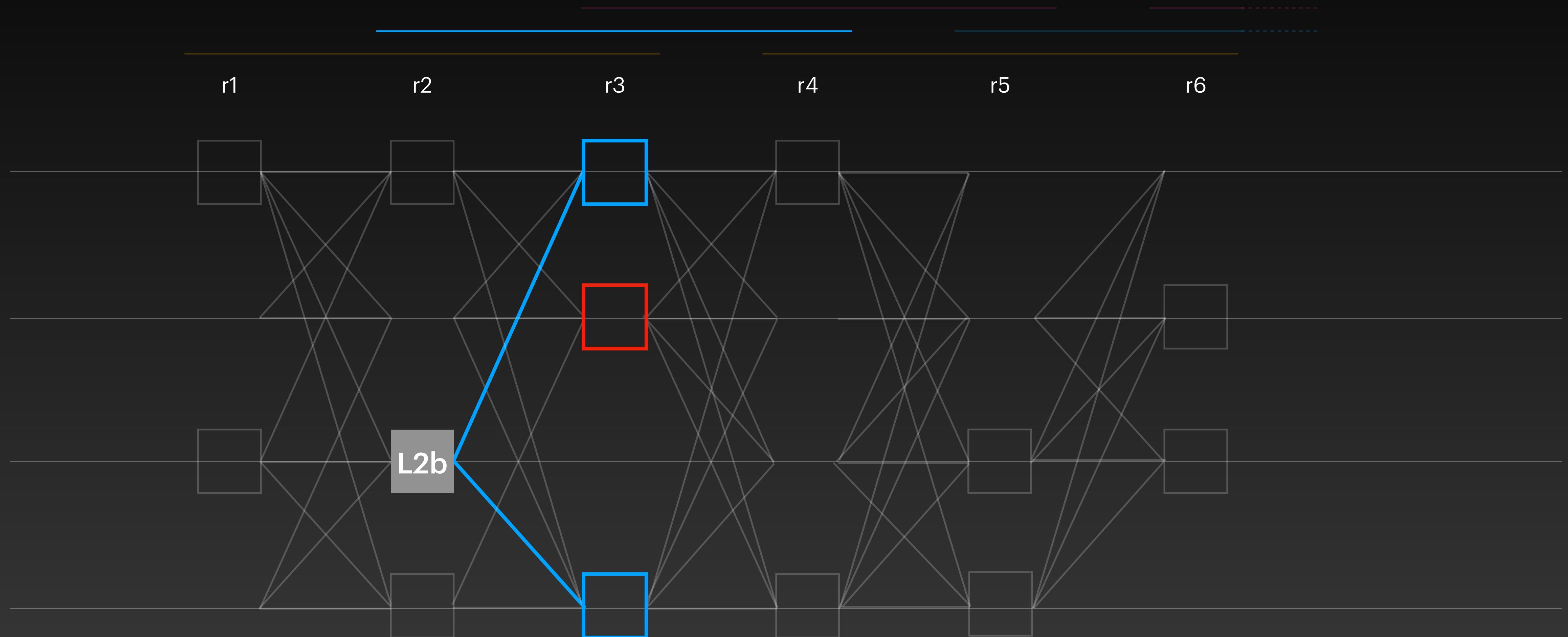
Ignore Missing Leader



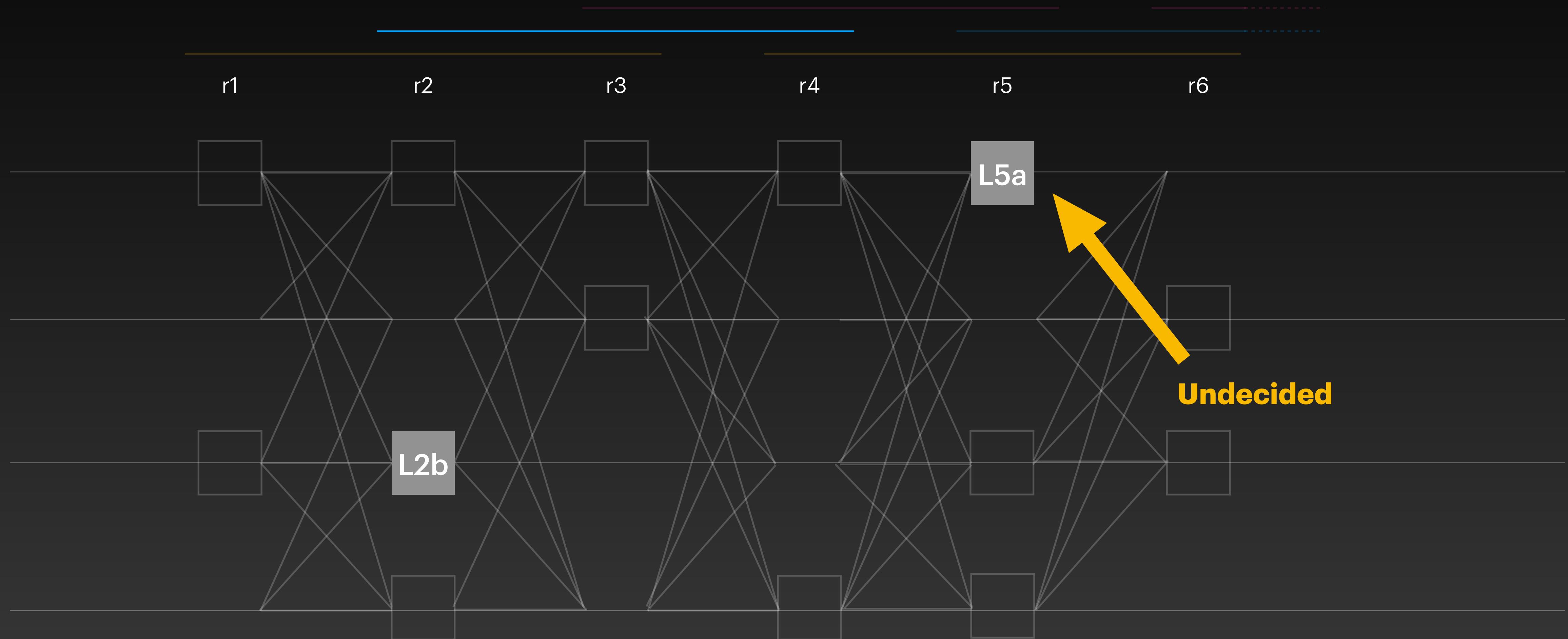
Apply Direct Rule



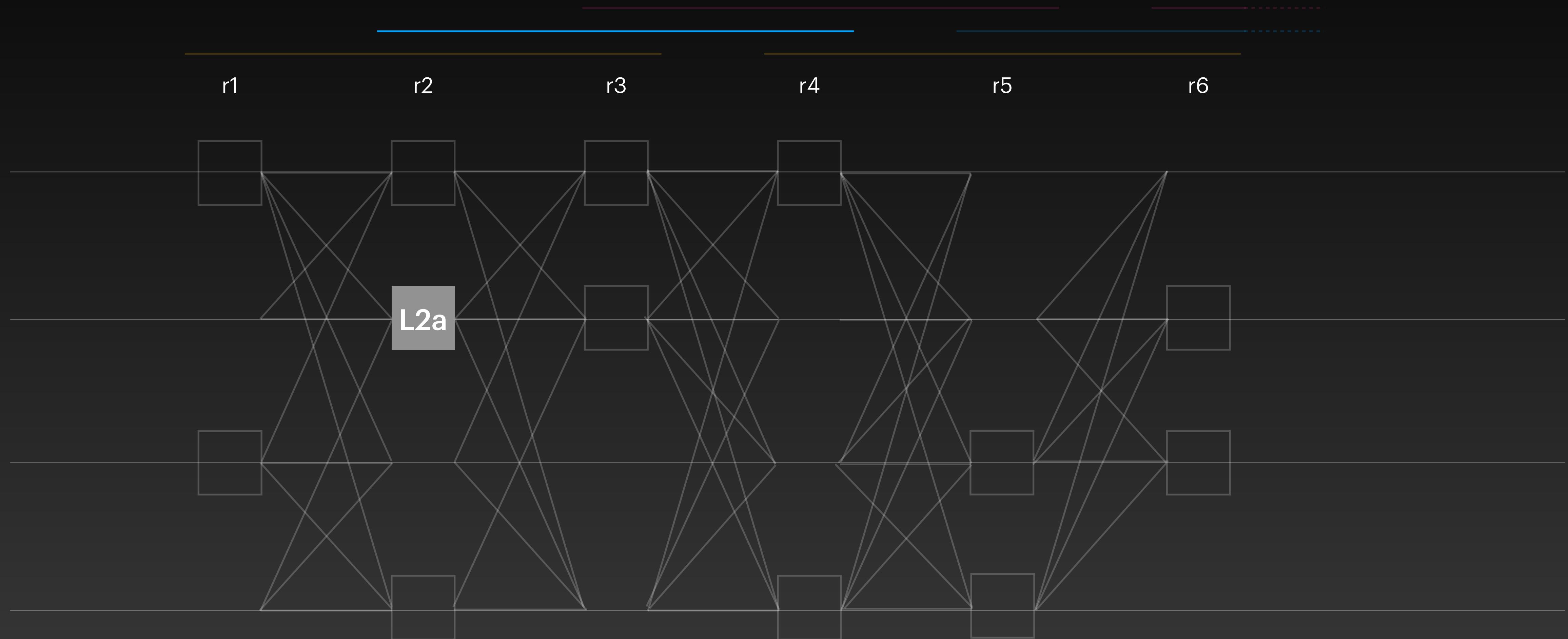
Apply Direct Rule



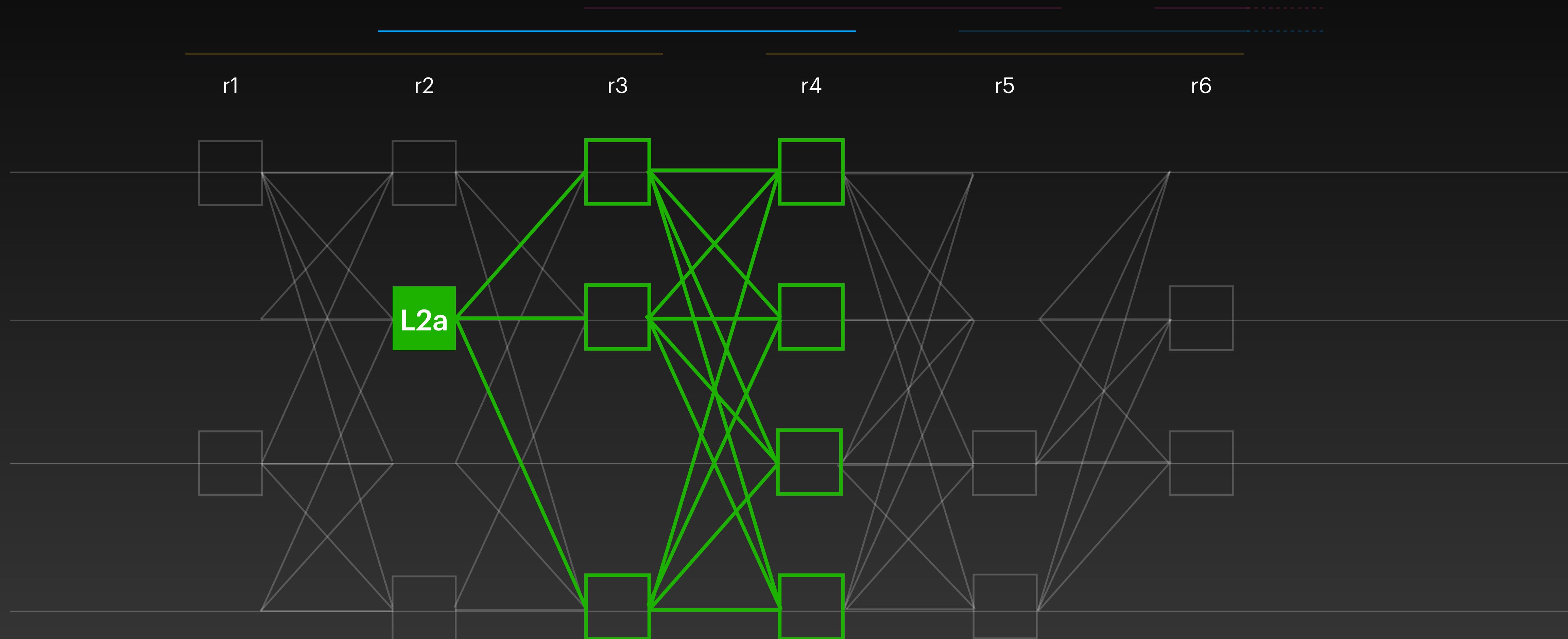
Apply Indirect Rule



Apply Direct Rule



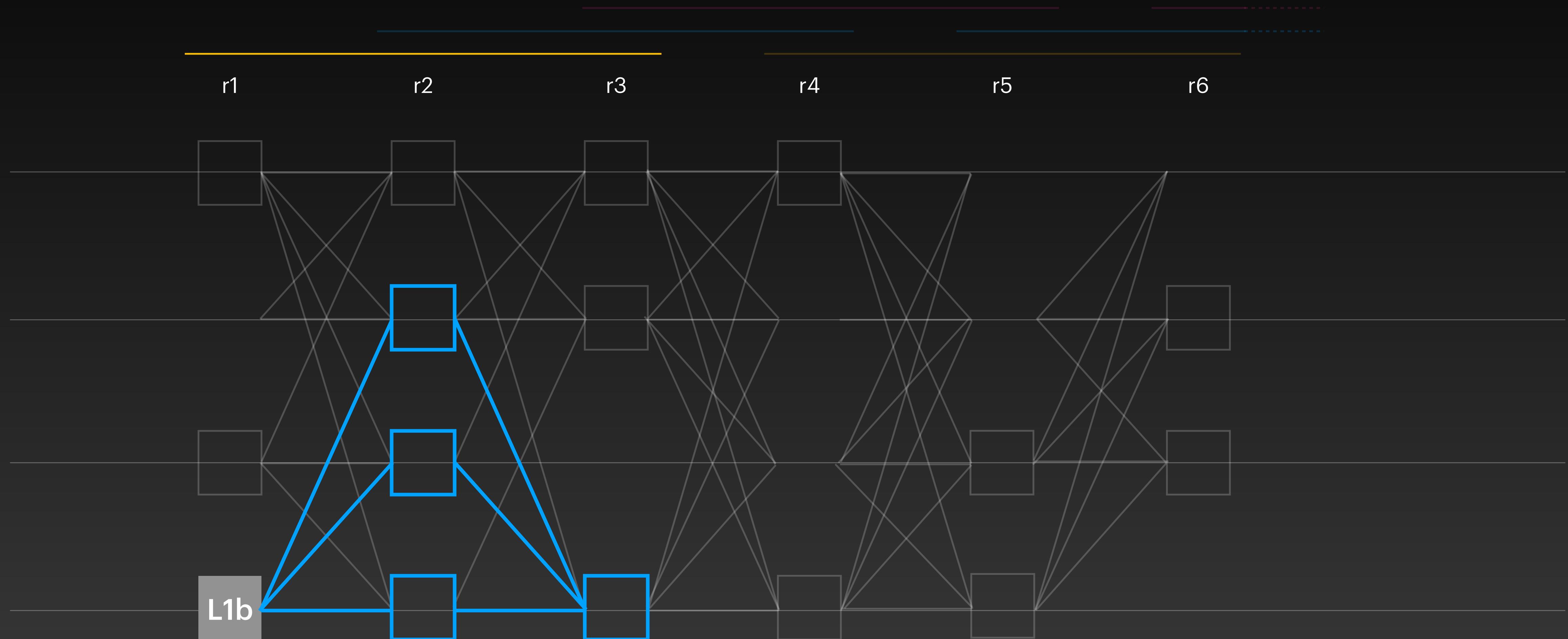
Apply Direct Rule



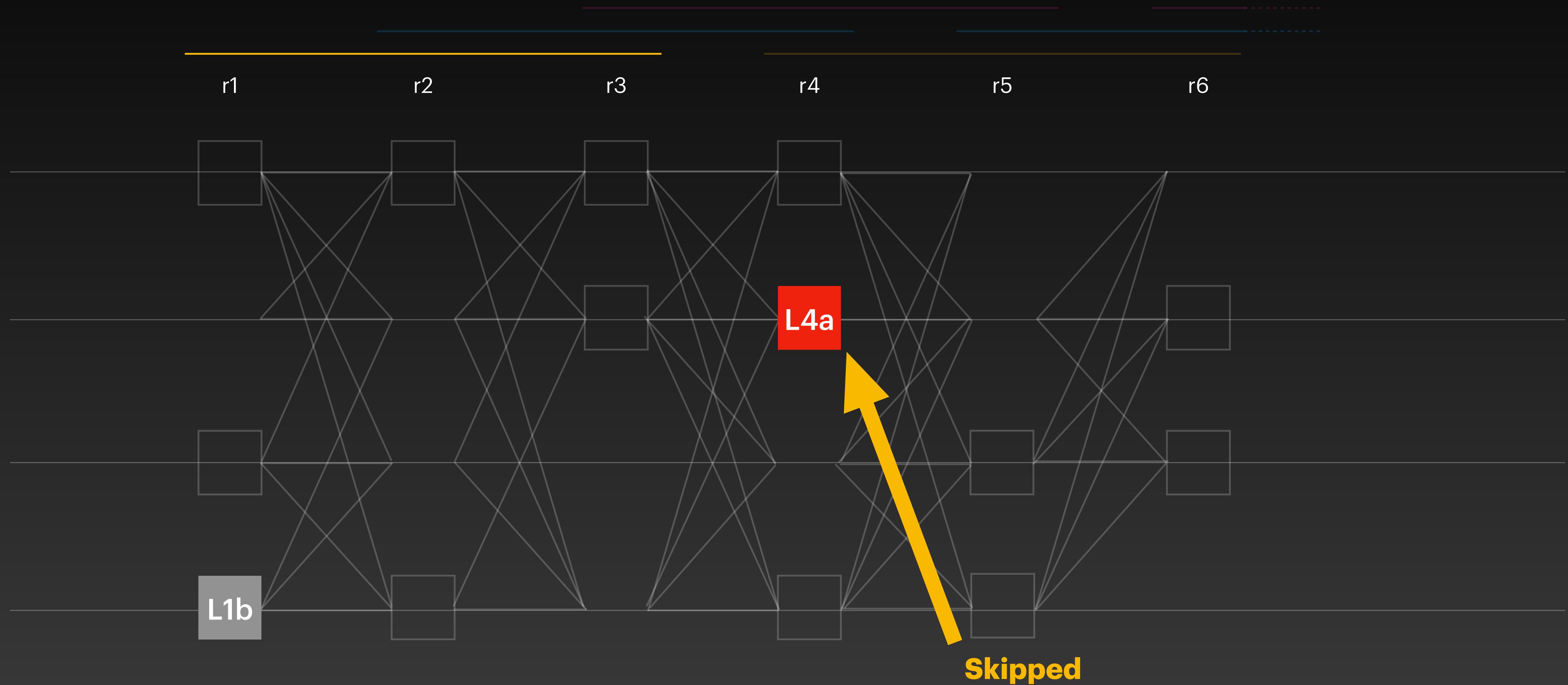
Apply Direct Rule



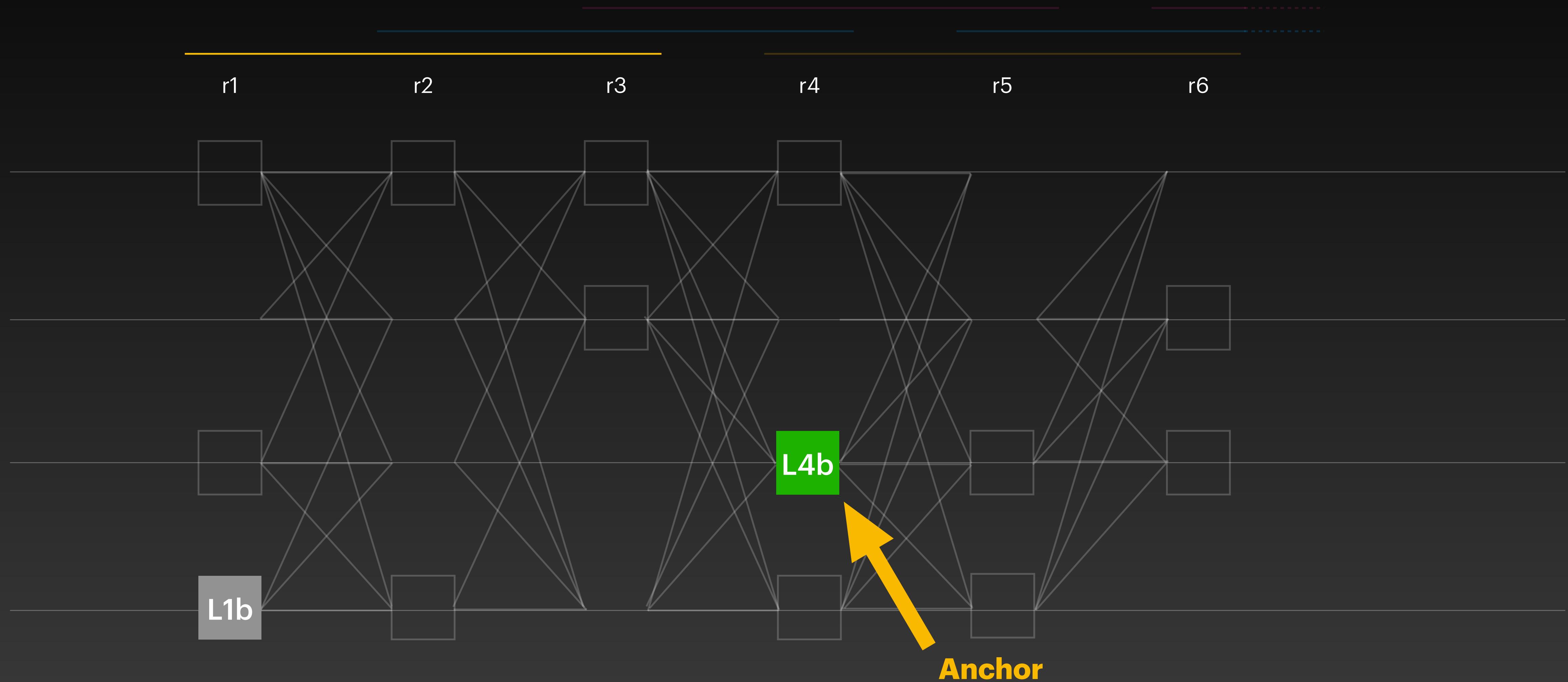
Apply Direct Rule



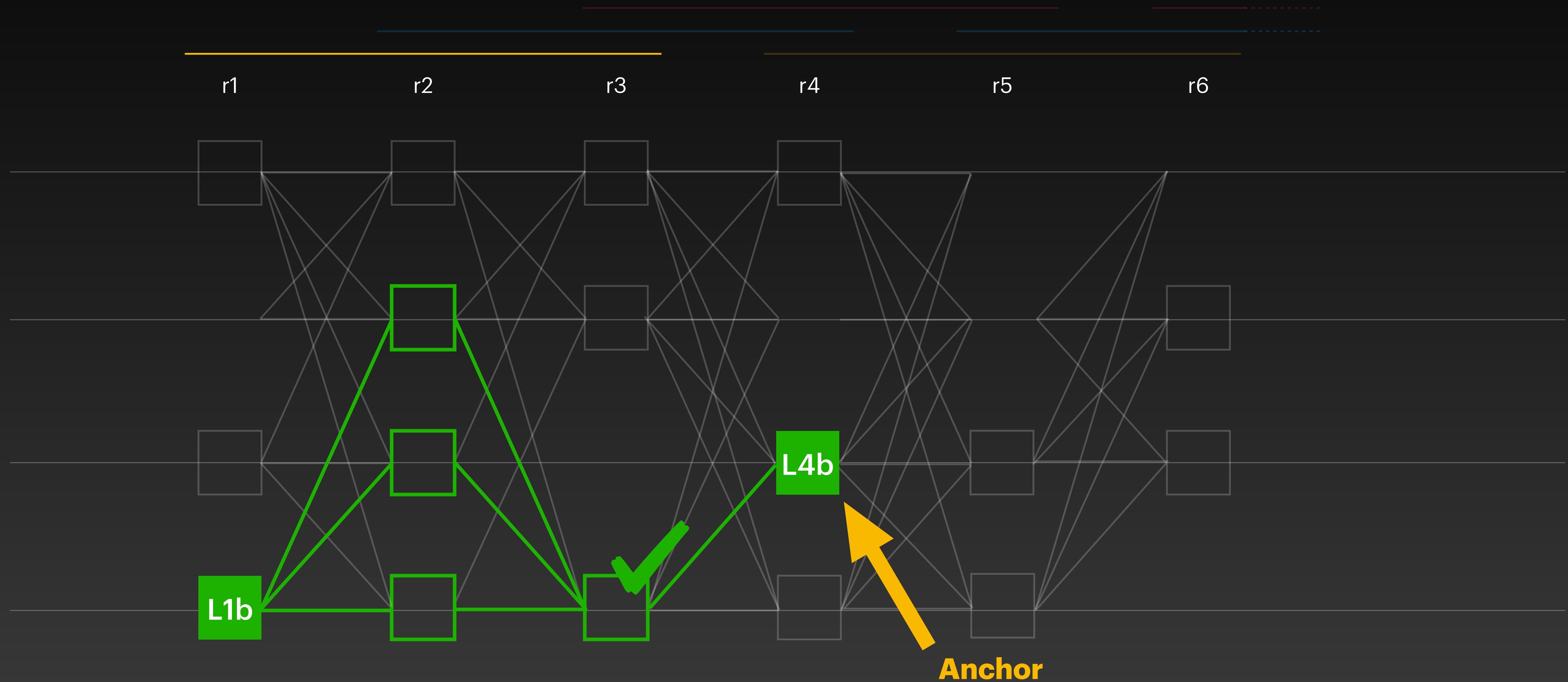
Apply Indirect Rule



Apply Indirect Rule



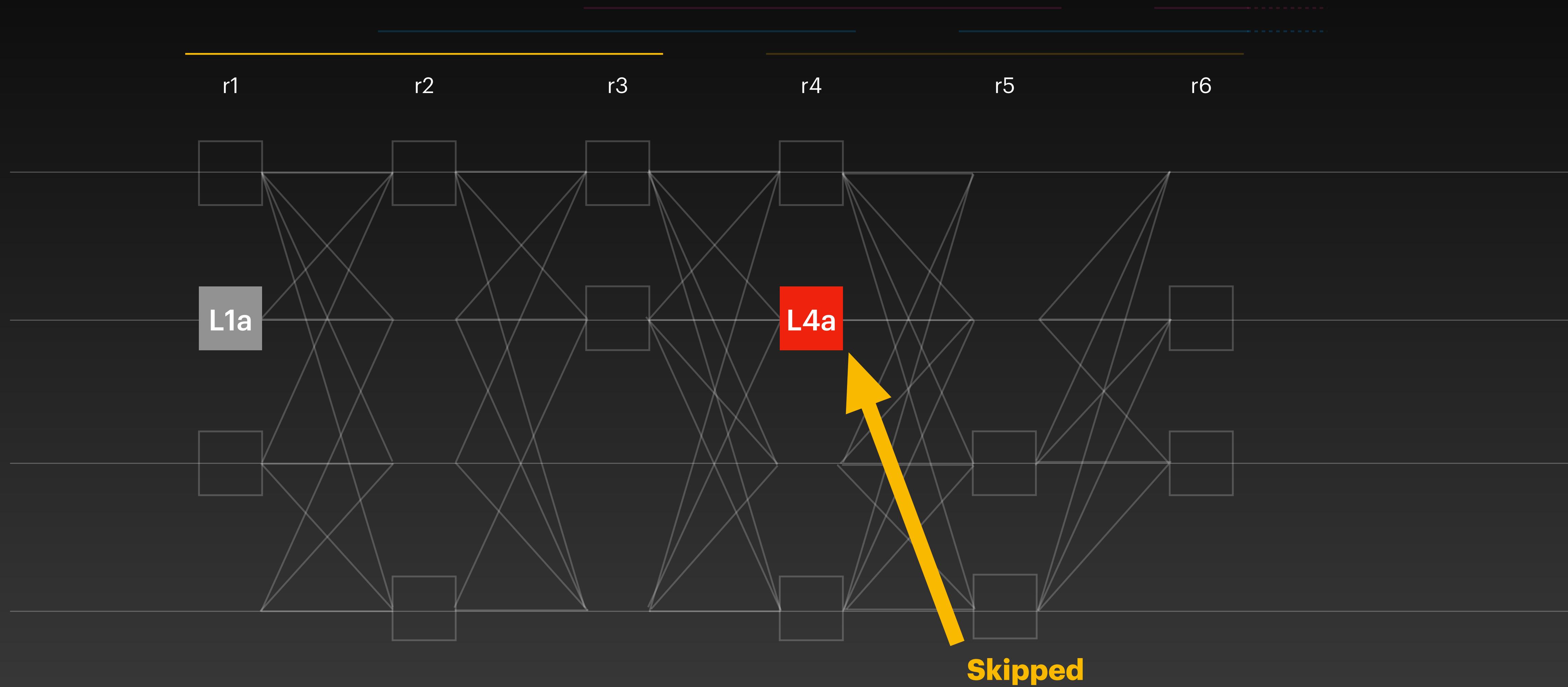
Apply Indirect Rule



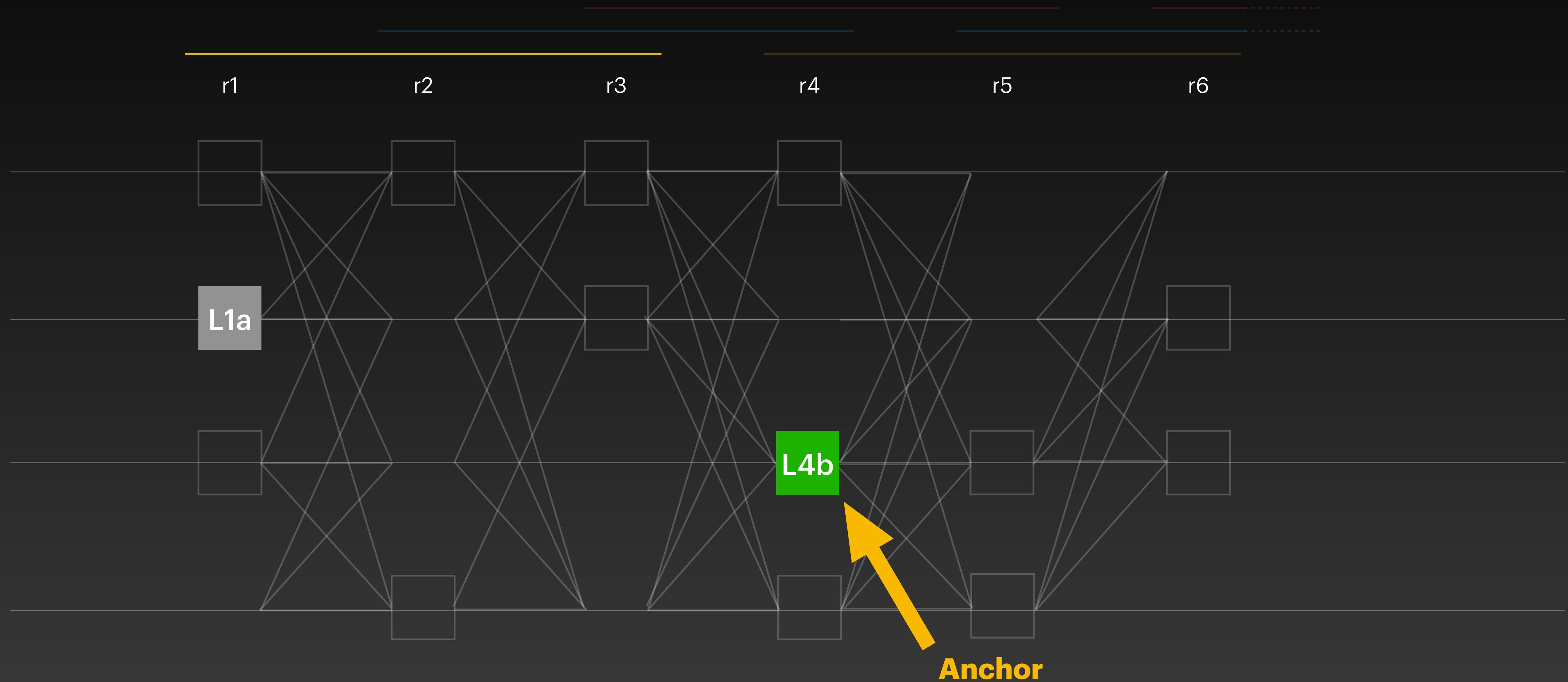
Apply Direct Rule



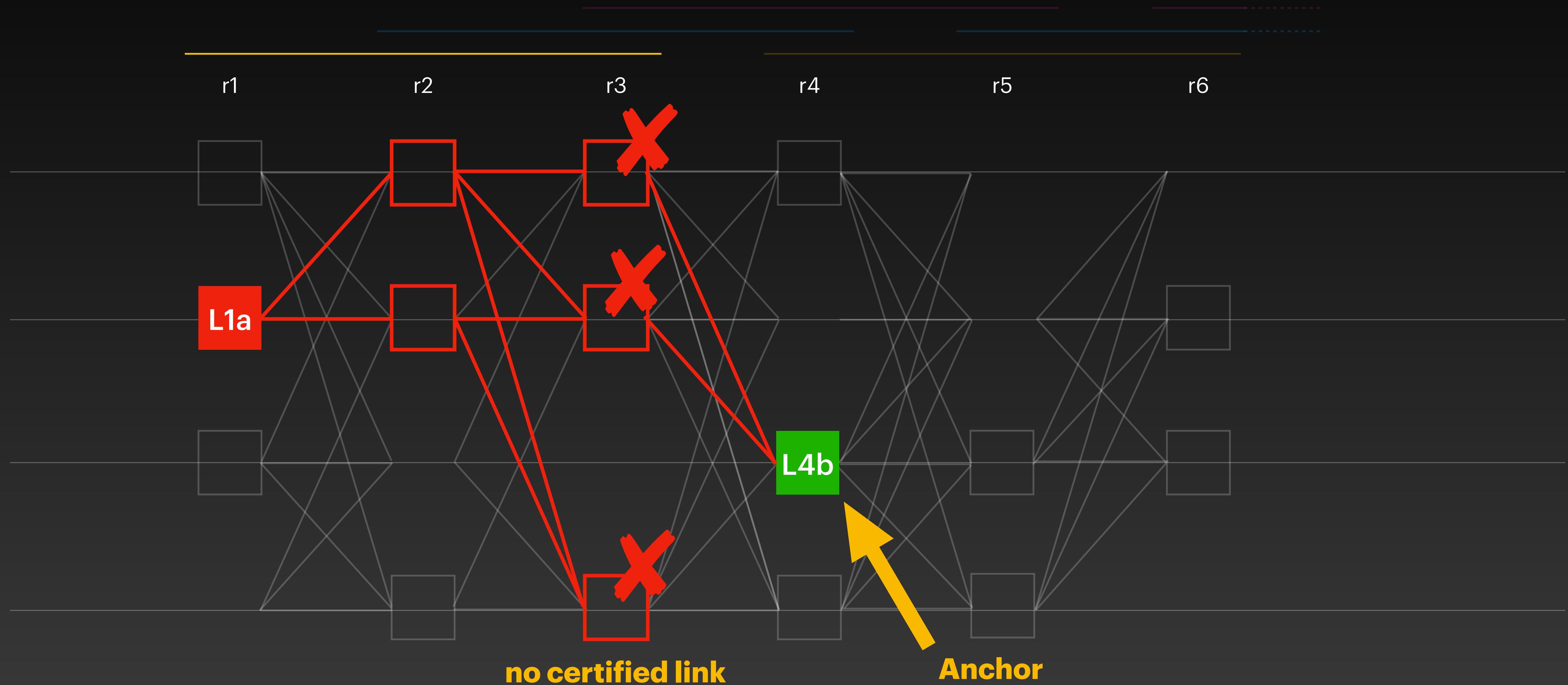
Apply Indirect Rule



Apply Indirect Rule



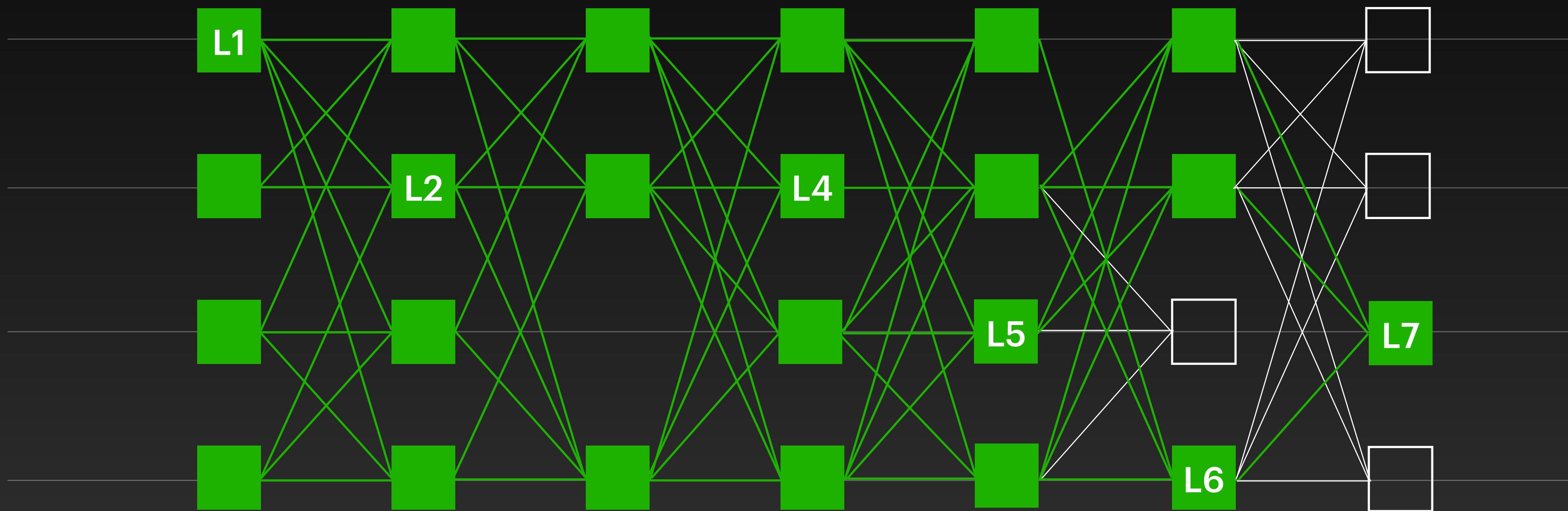
Apply Indirect Rule



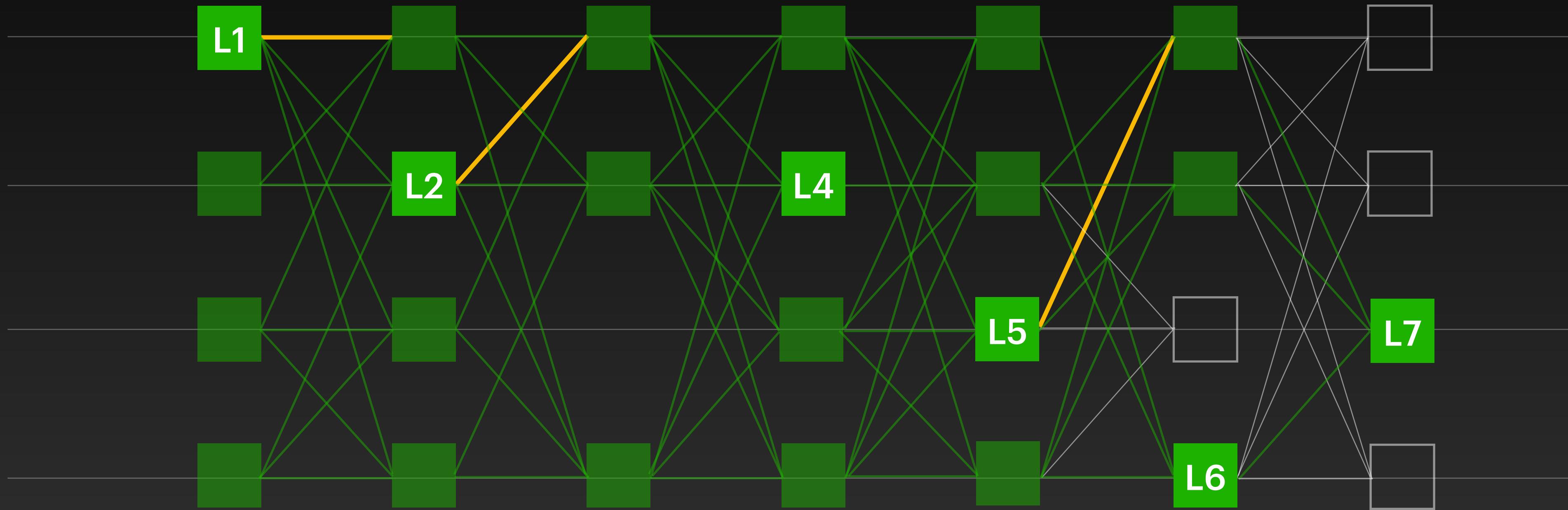
HammerHead

Mitigating slow leaders

Past Commits

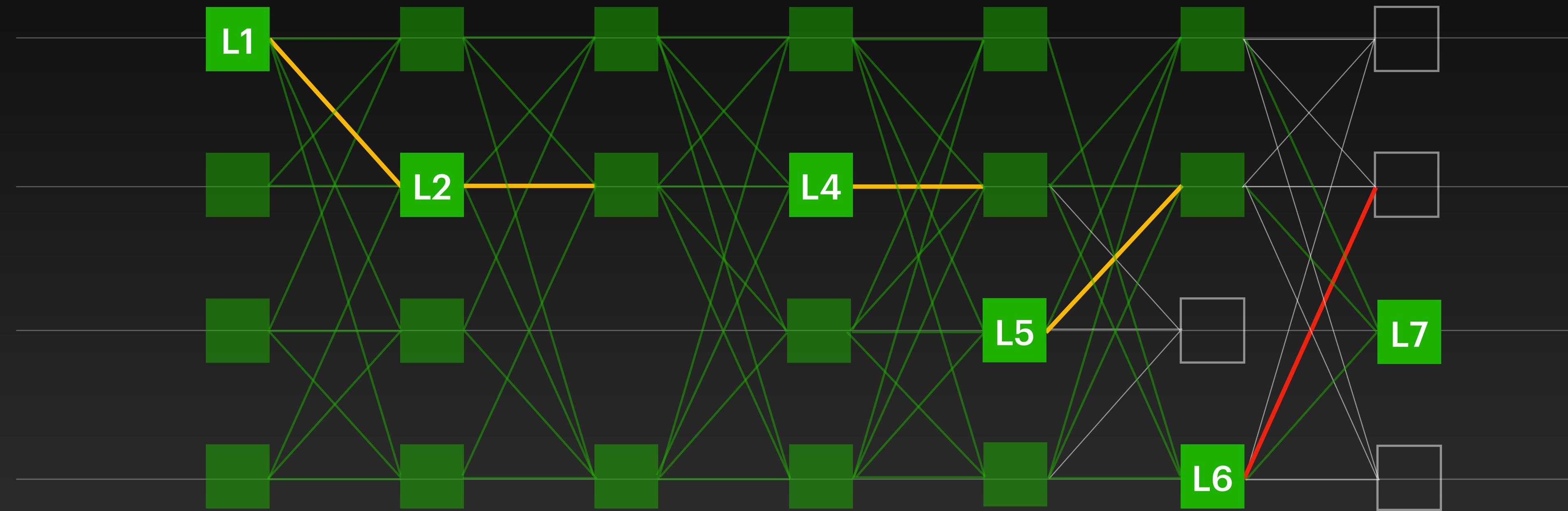


Compute Reputation Scores



node 1: 3

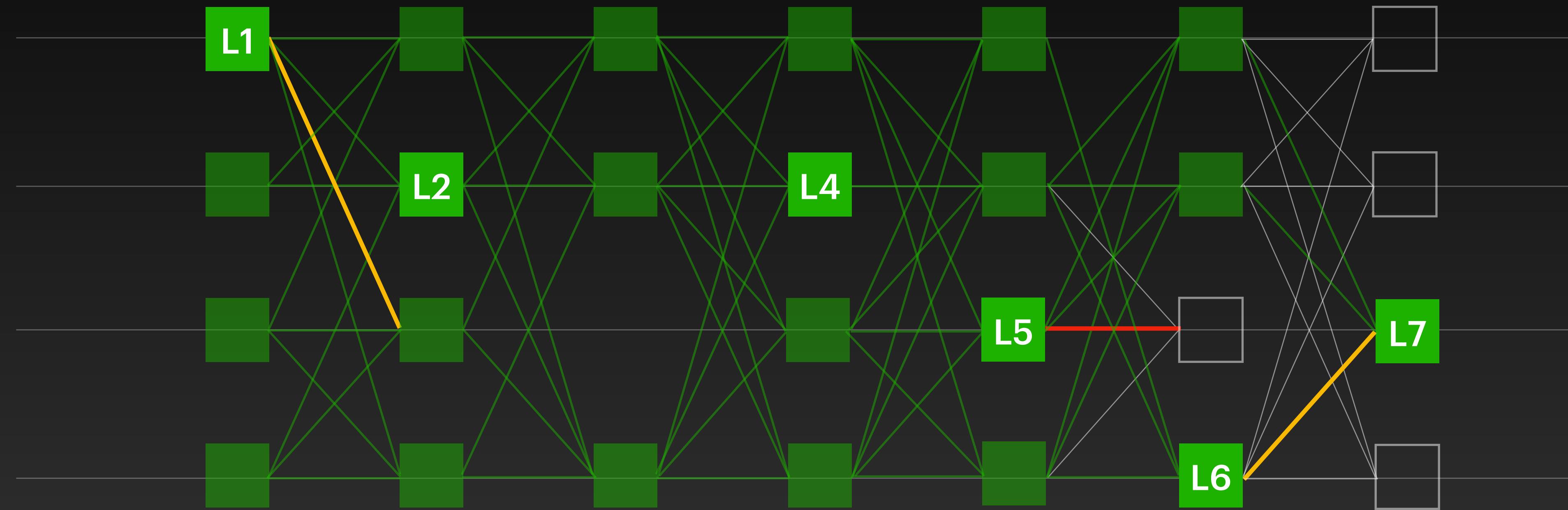
Compute Reputation Scores



node 1: 3

node 2: 4

Compute Reputation Scores

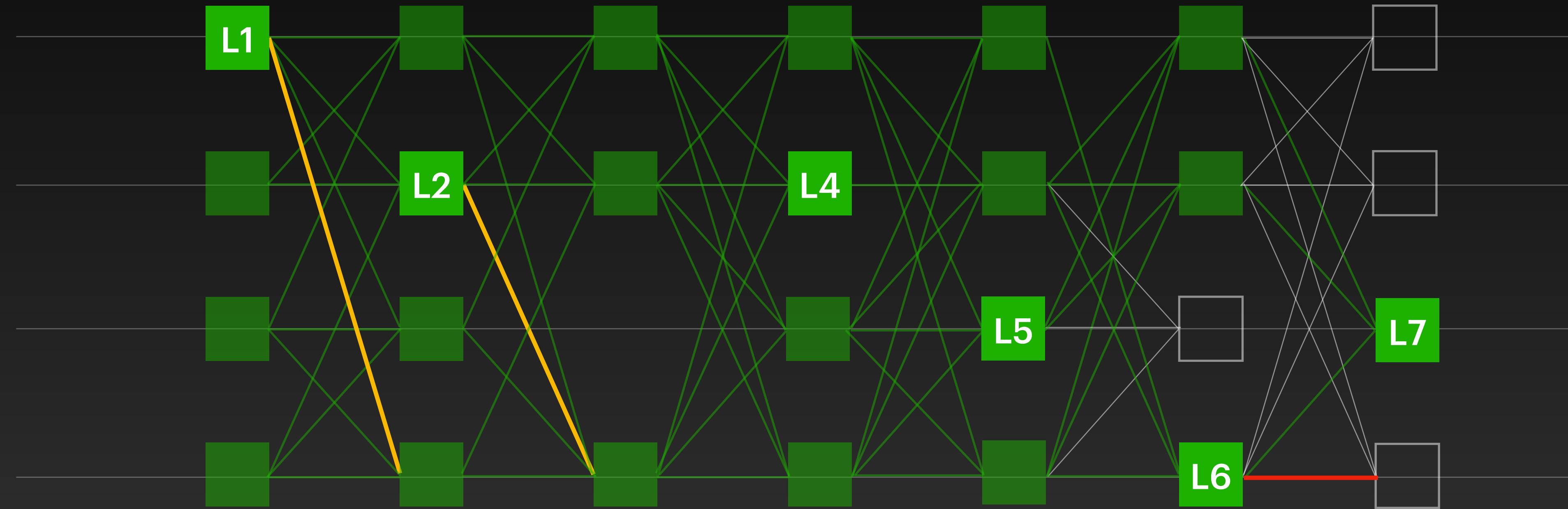


node 1: 3

node 2: 4

node 3: 2

Compute Reputation Scores



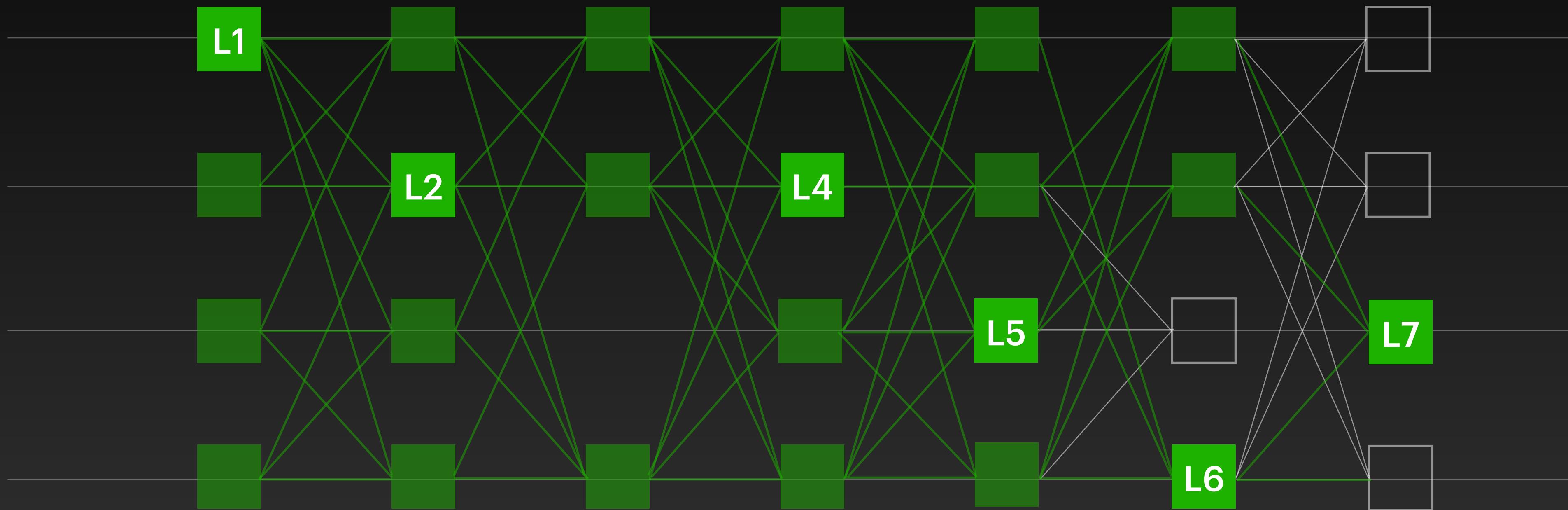
node 1: 3

node 2: 4

node 3: 2

node 4: 2

Future Leaders

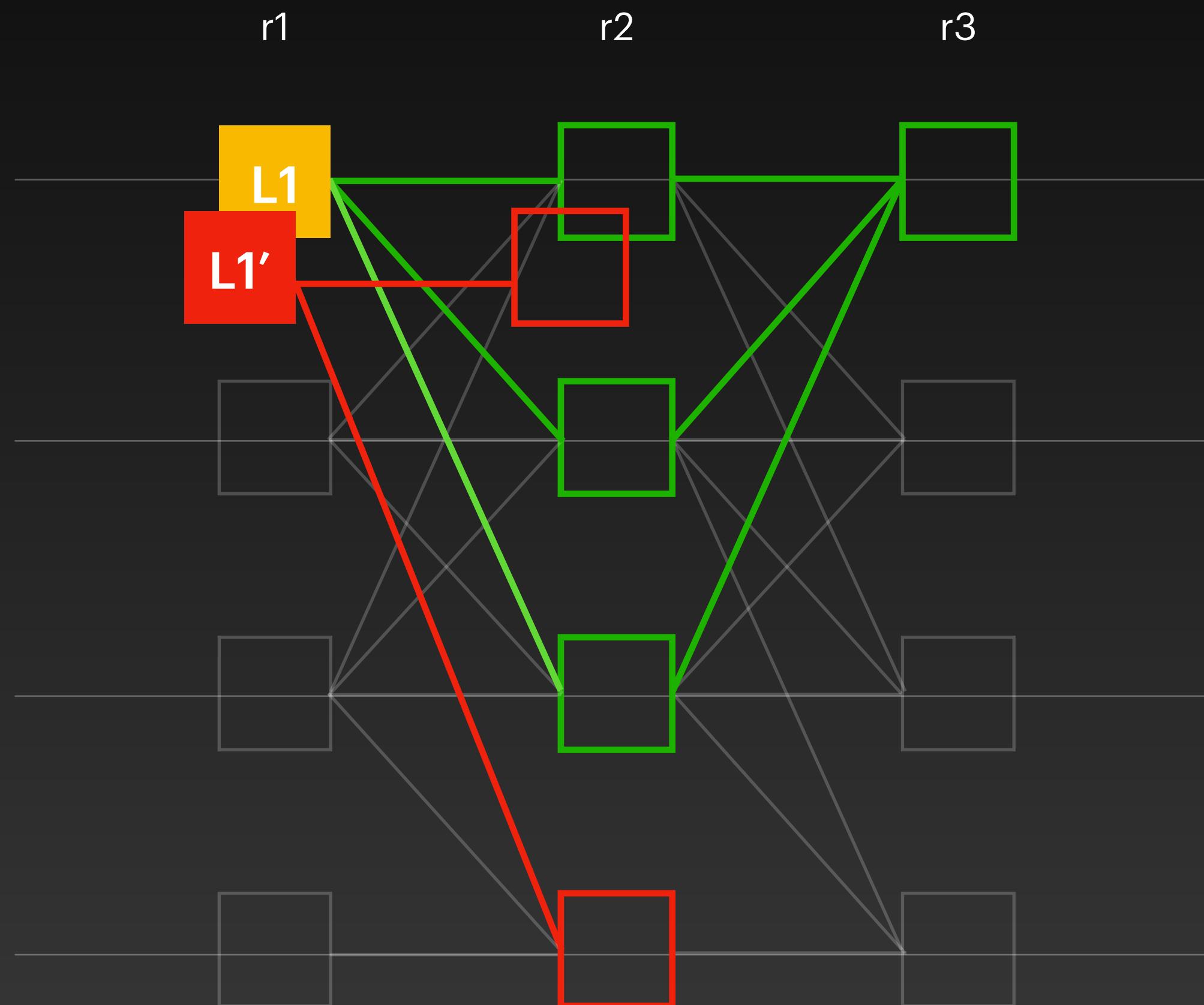


node 1: 3

node 2: 4

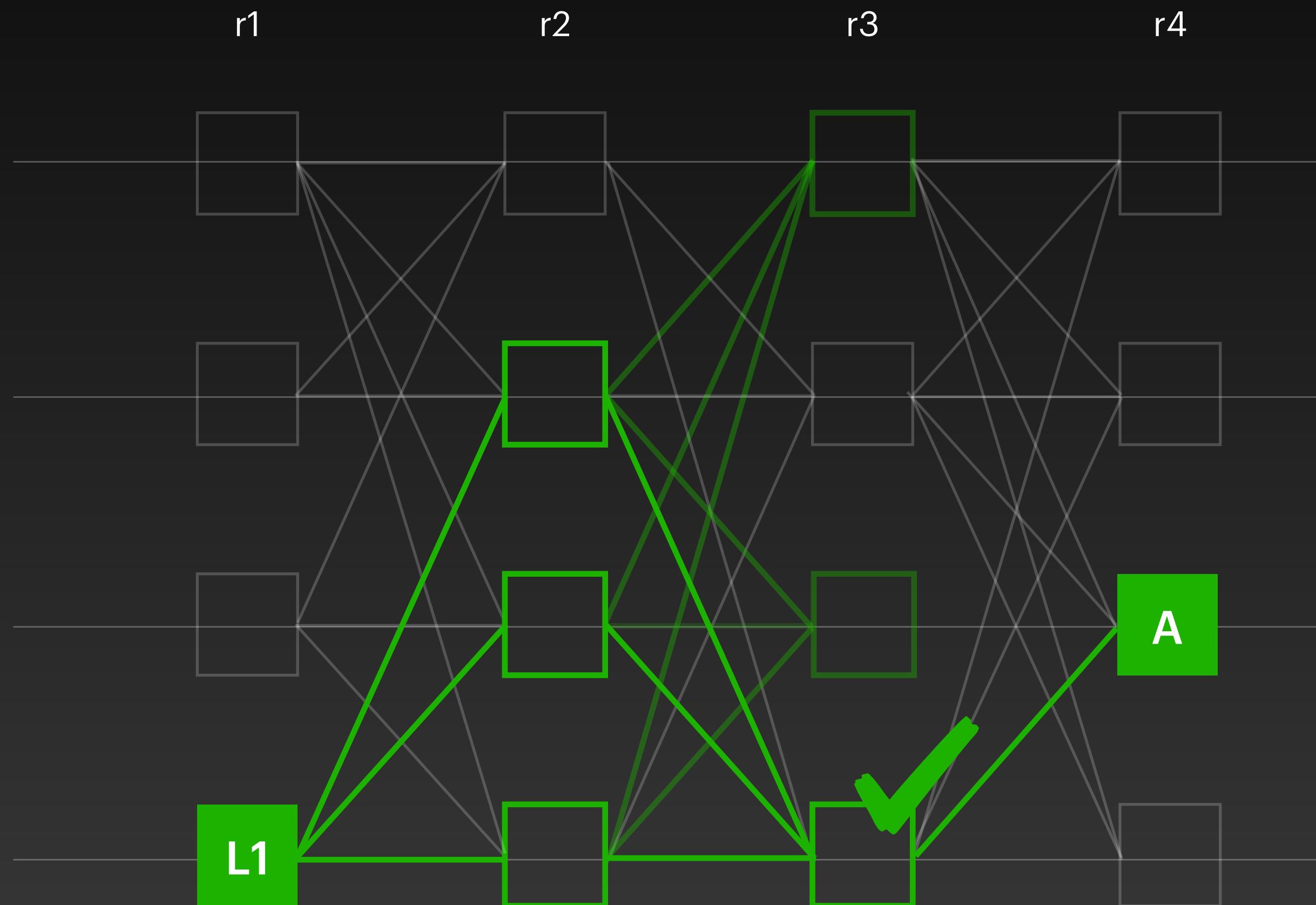
Security Intuition

Security Intuition



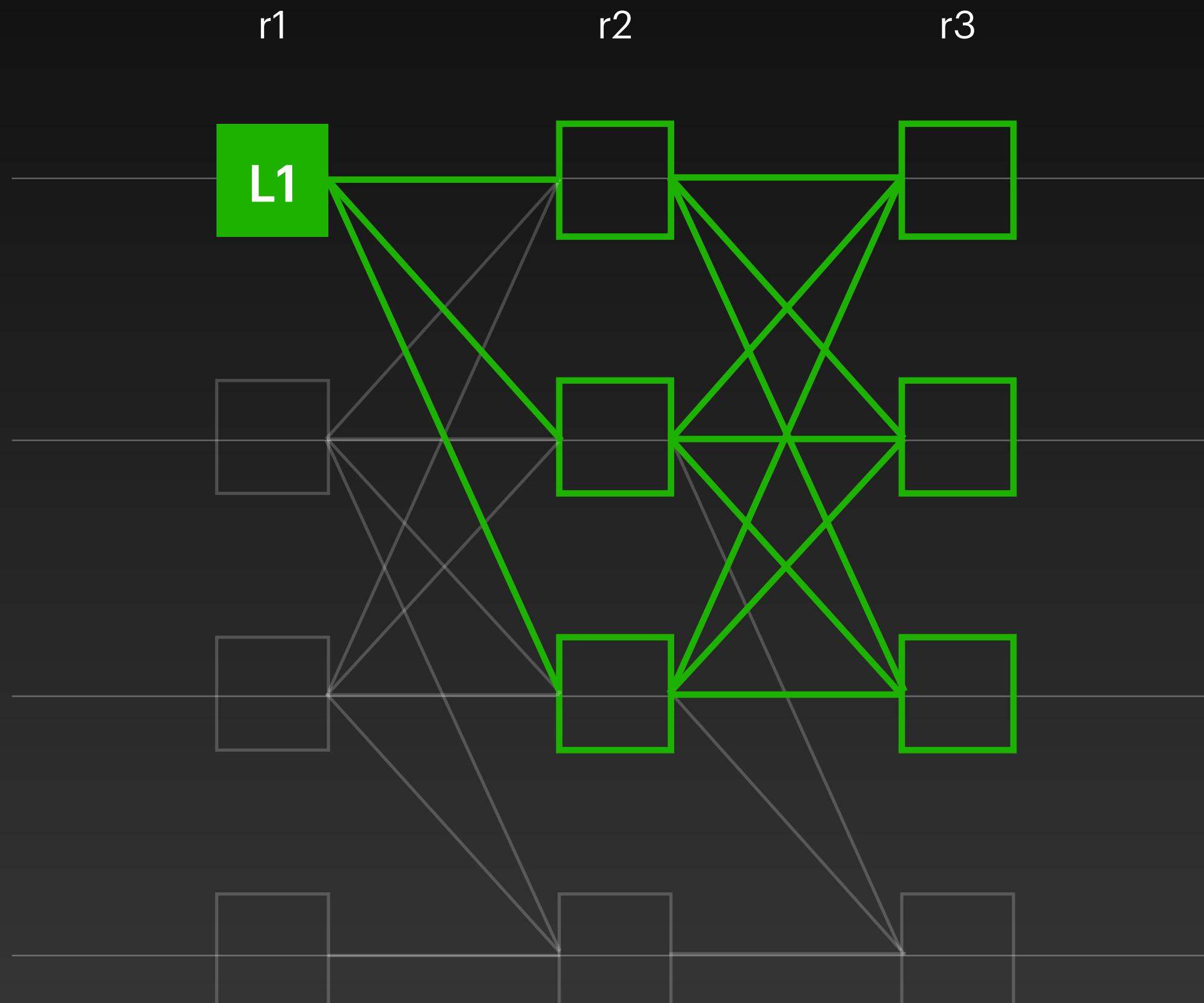
- At most **L1** or **L1'** can have a certificate pattern (quorum intersection)

Security Intuition



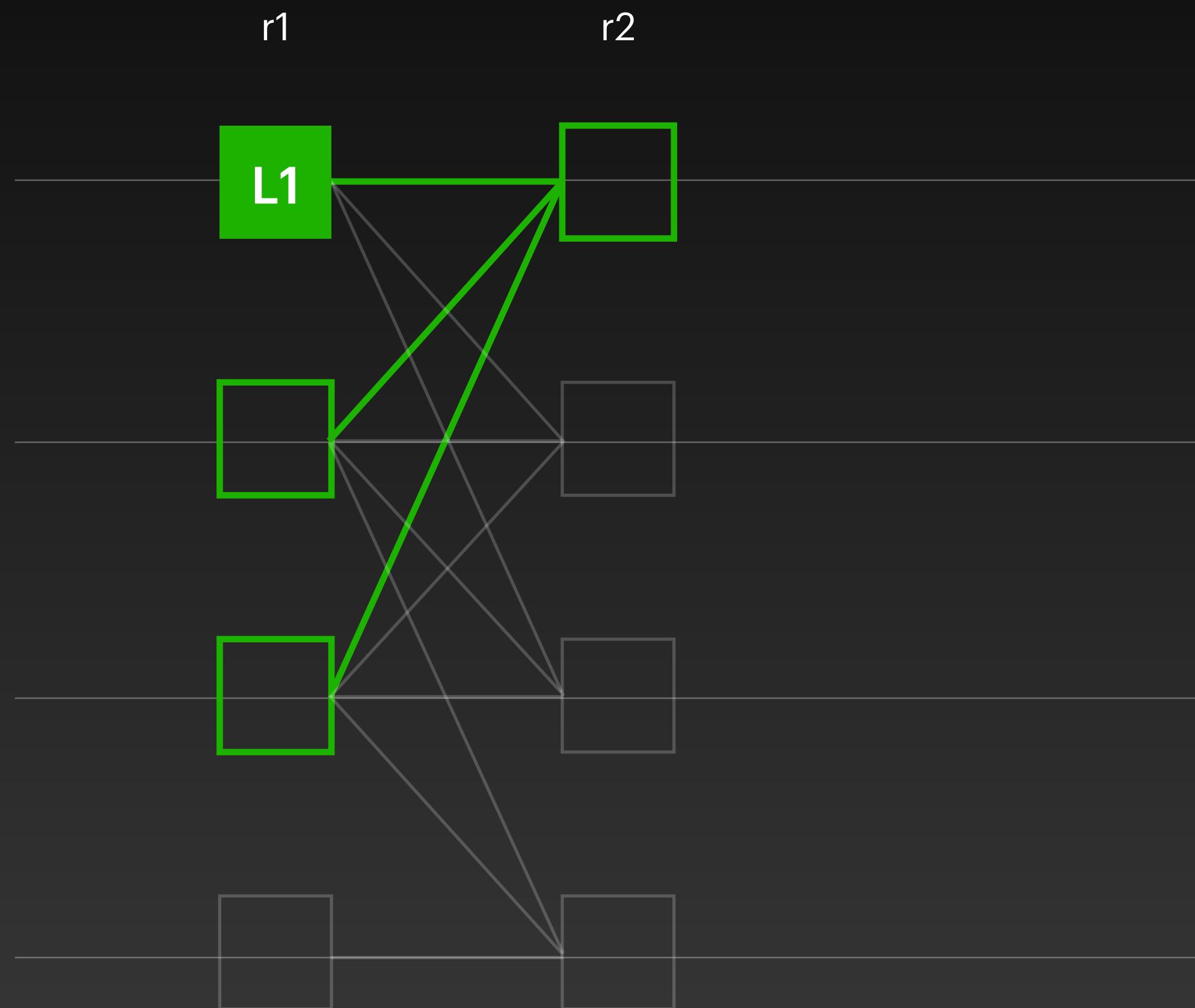
- At most **L1** or **L1'** can have a certificate pattern (quorum intersection)
- If **L1** has $2f+1$ certificate patterns, **A** always has a certified link to **L1**

Security Intuition



- At most **L1** or **L1'** can have a certificate pattern (quorum intersection)
- If **L1** has $2f+1$ certificate patterns, **A** always has a certified link to **L1**
- After GST, the direct decision rule **commits** a block

Security Intuition



Leader Timeout:

Wait for $2f+1$ parents + 250 ms

Mysticeti-FPC

Adding a fast commit path

Consensus Not Required

Coins, balances, and transfers

NFTs creation and transfers

Game logic allowing users to combine assets

Inventory management for games / metaverse

Auditable 3rd party services not trusted for safety

...

Consensus Required

Increment a publicly-accessible counter

Auctions

Market places

Collaborative in-game assets

...

Object Type

Owned Objects

- Objects that can be mutated by a single entity
- e.g., My bank account
- **Do not need consensus**

Shared Objects

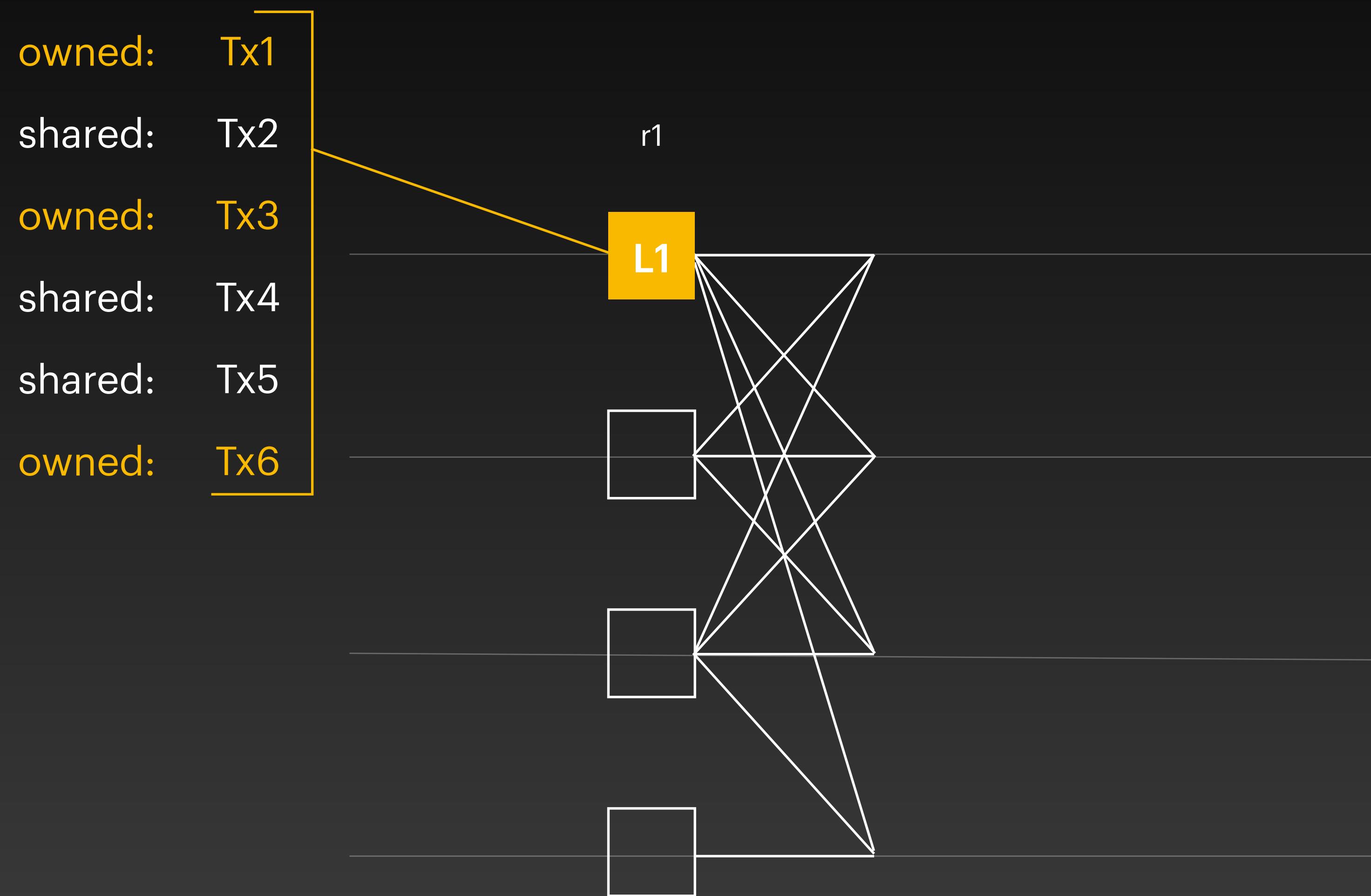
- Objects that can be mutated my multiple entities
- e.g., A global counter
- **Need consensus**

System State

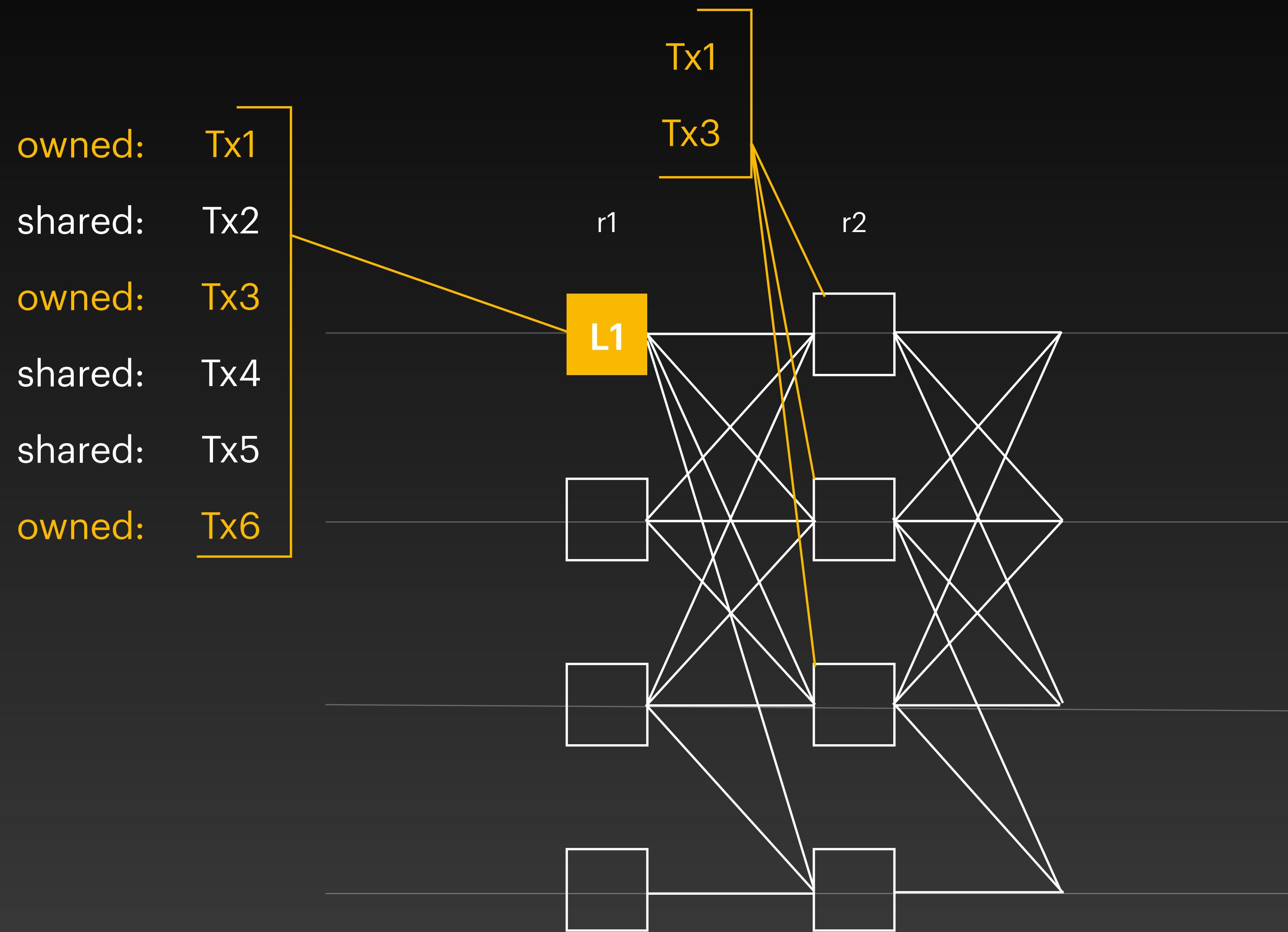
Objects:

- Unique ID
- Version number
- Ownership Information
- Type (shared, owned)

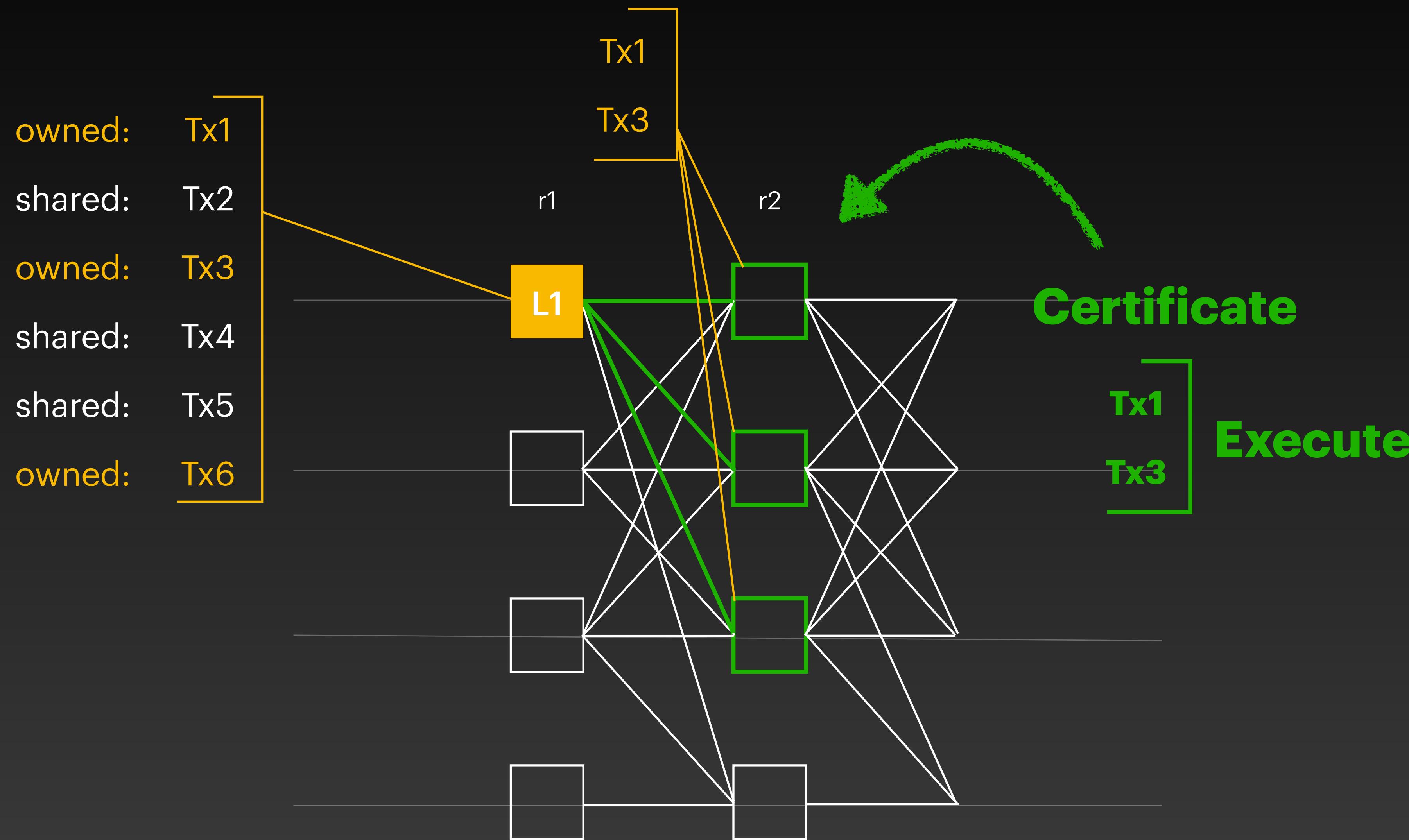
Fast Execution



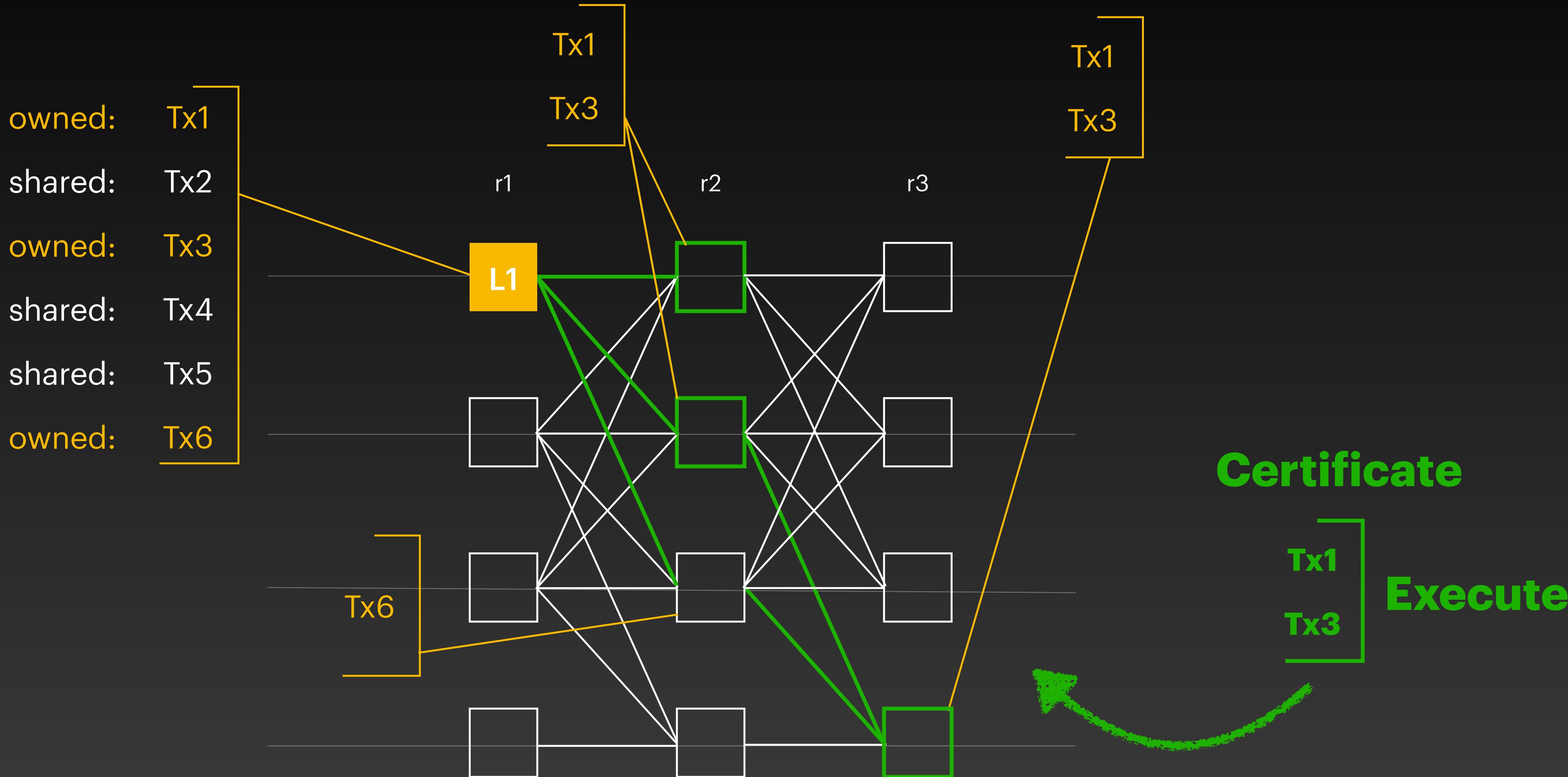
Fast Execution



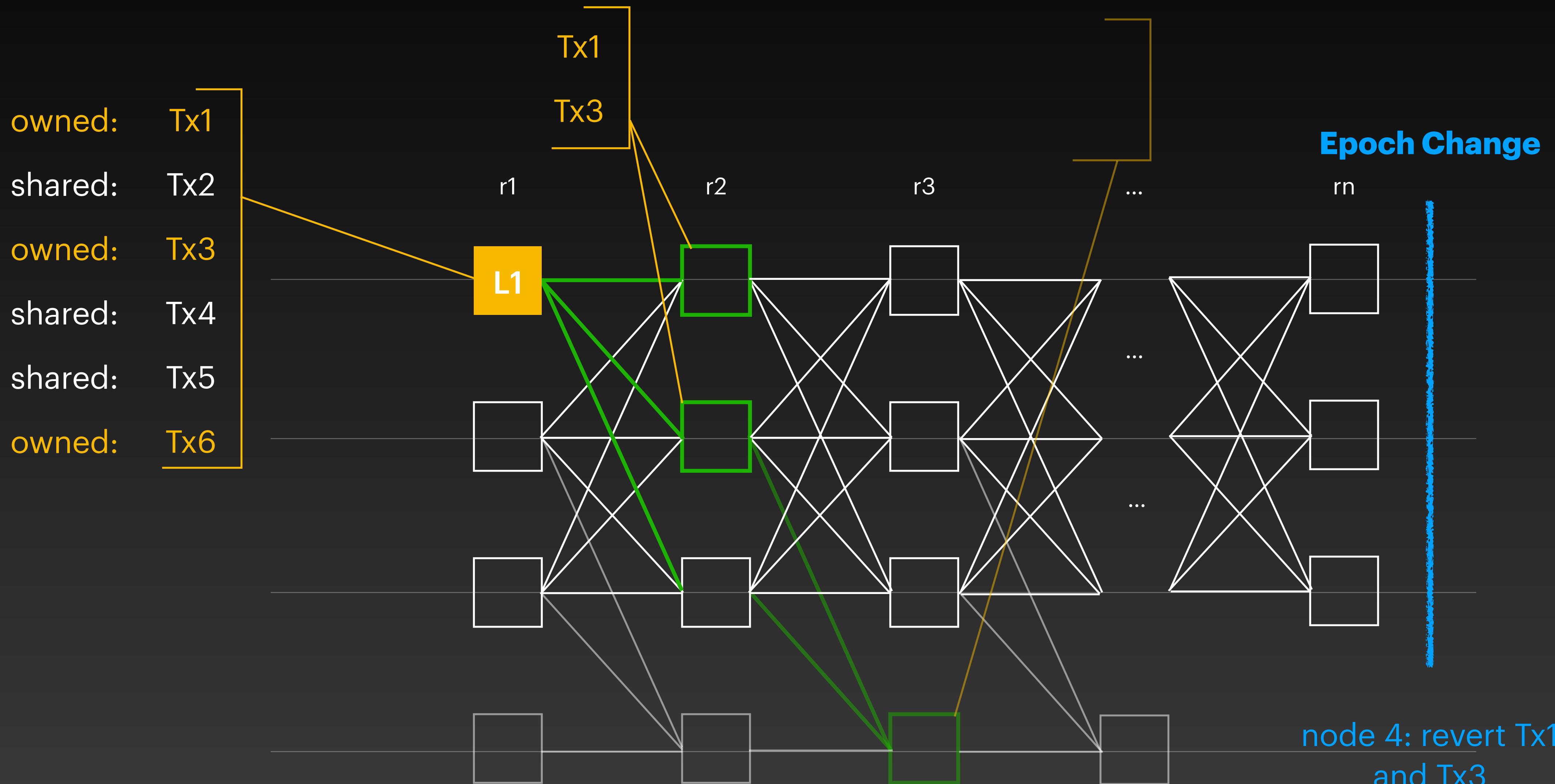
Fast Execution



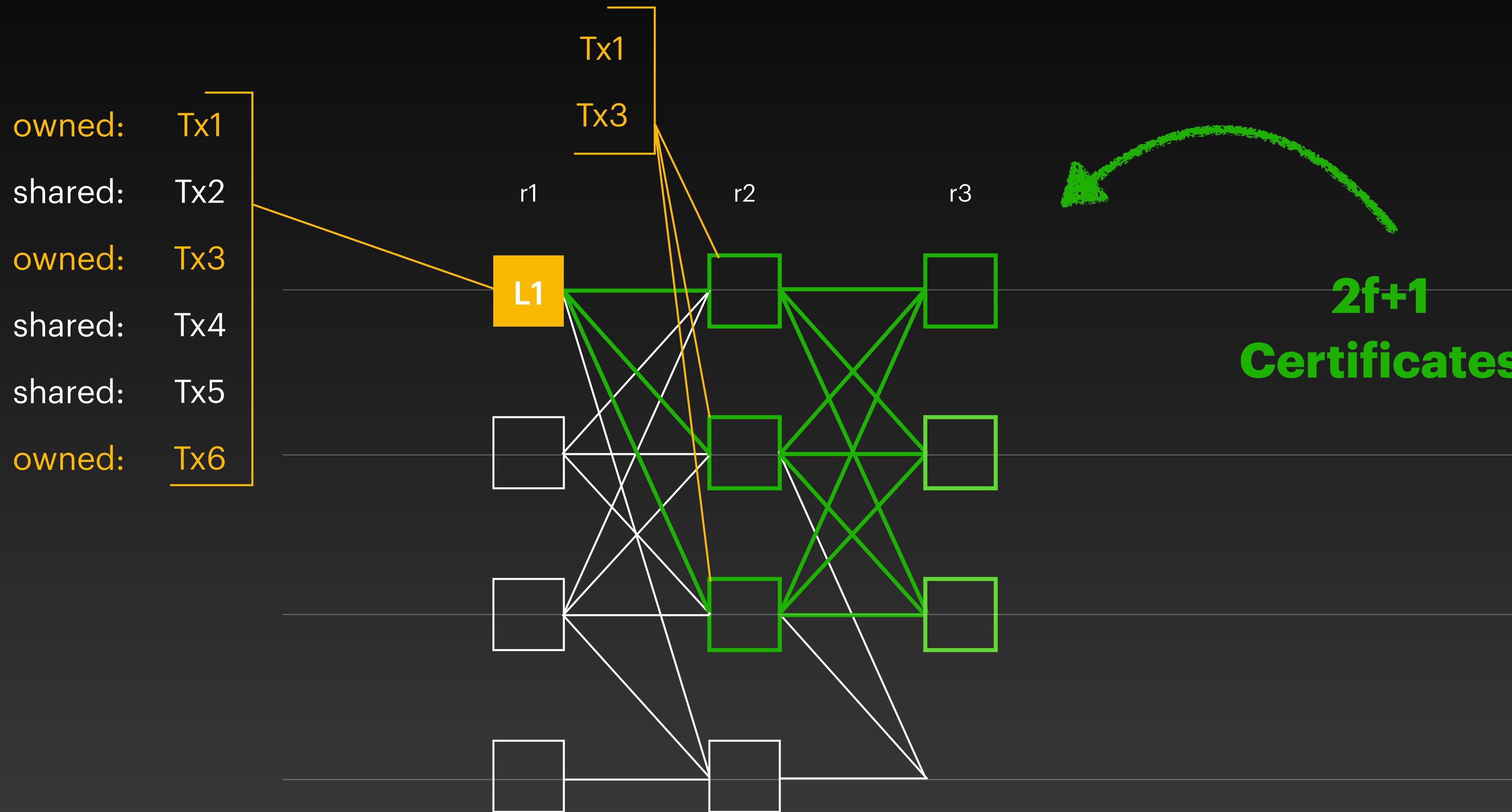
Fast Execution



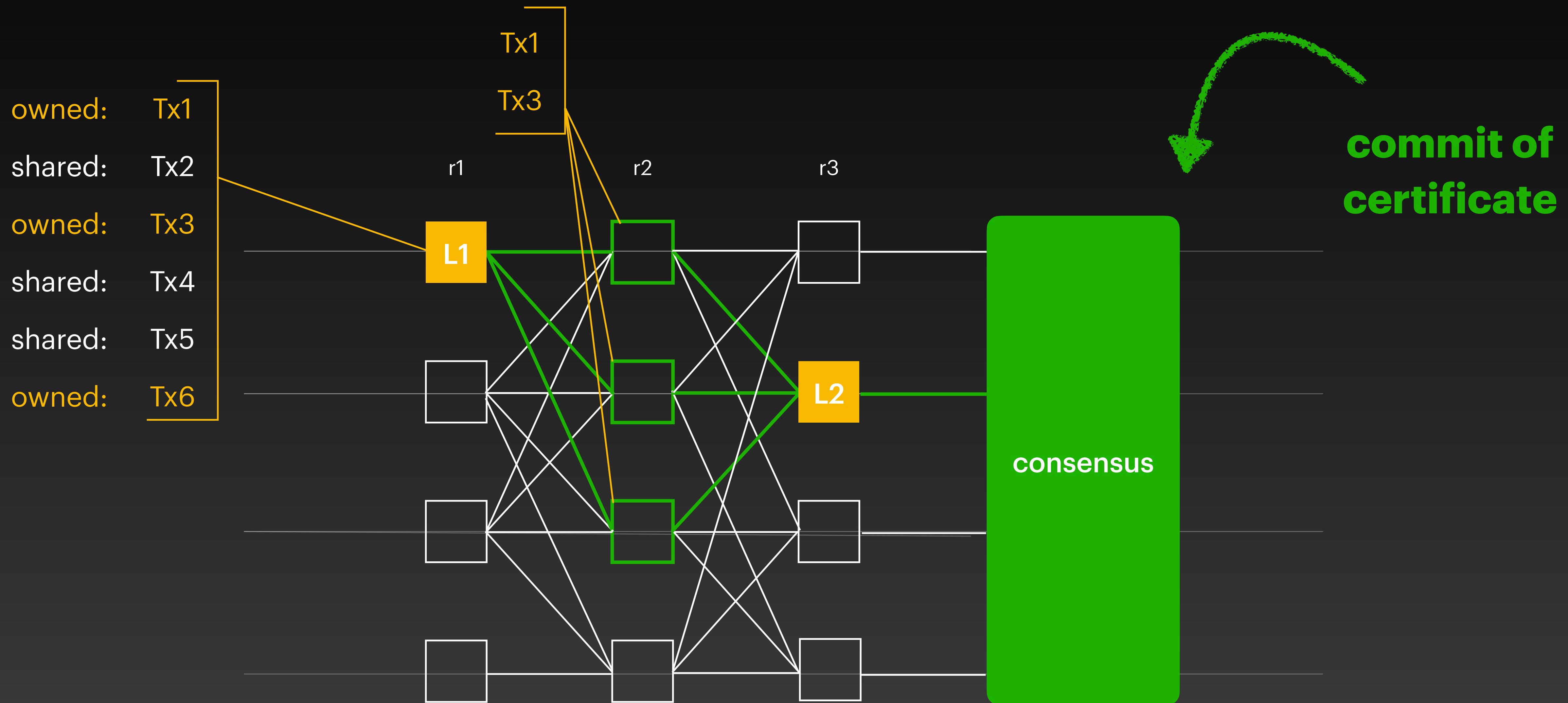
No Finality



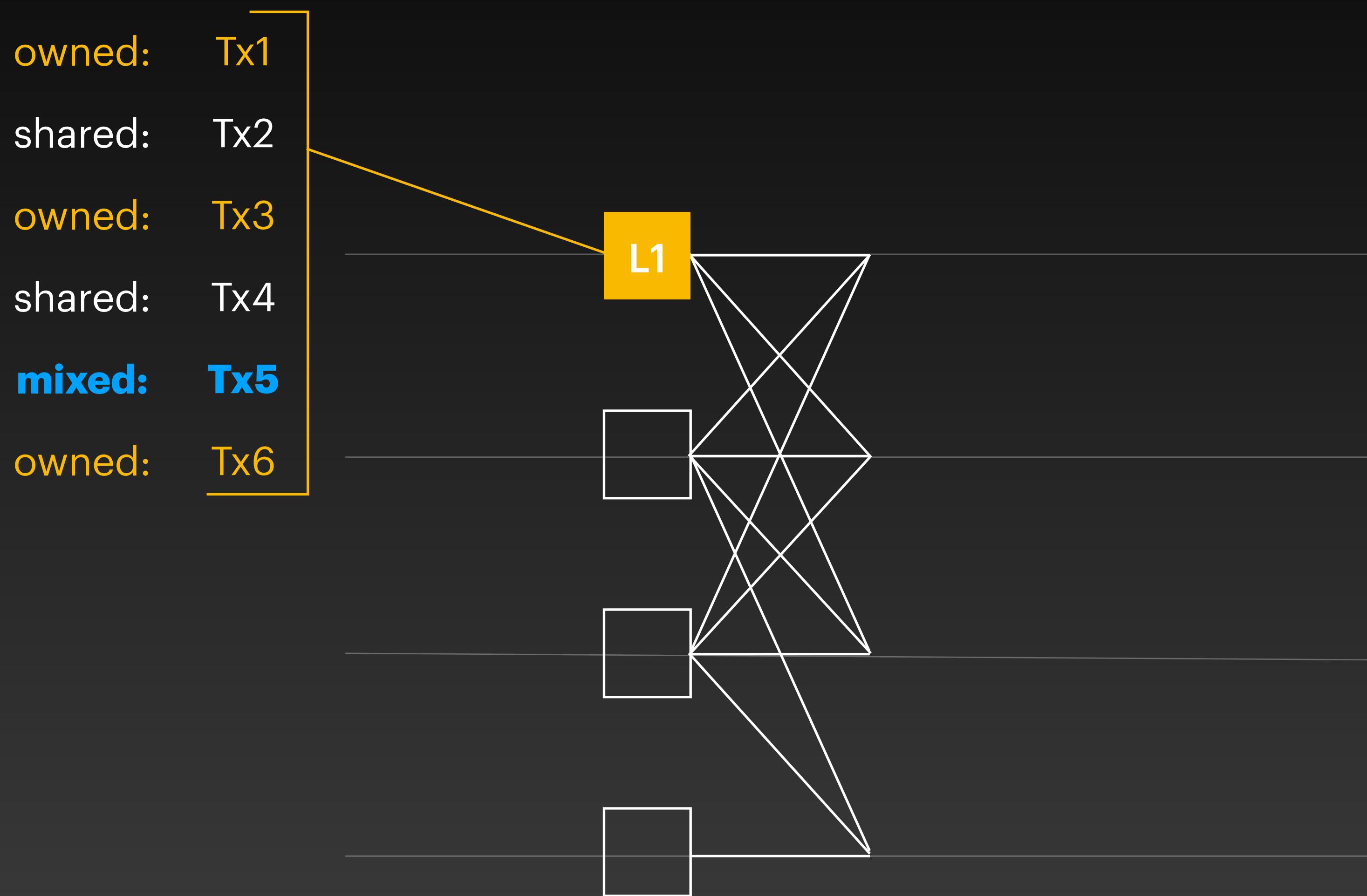
Fast Path Finality (1)



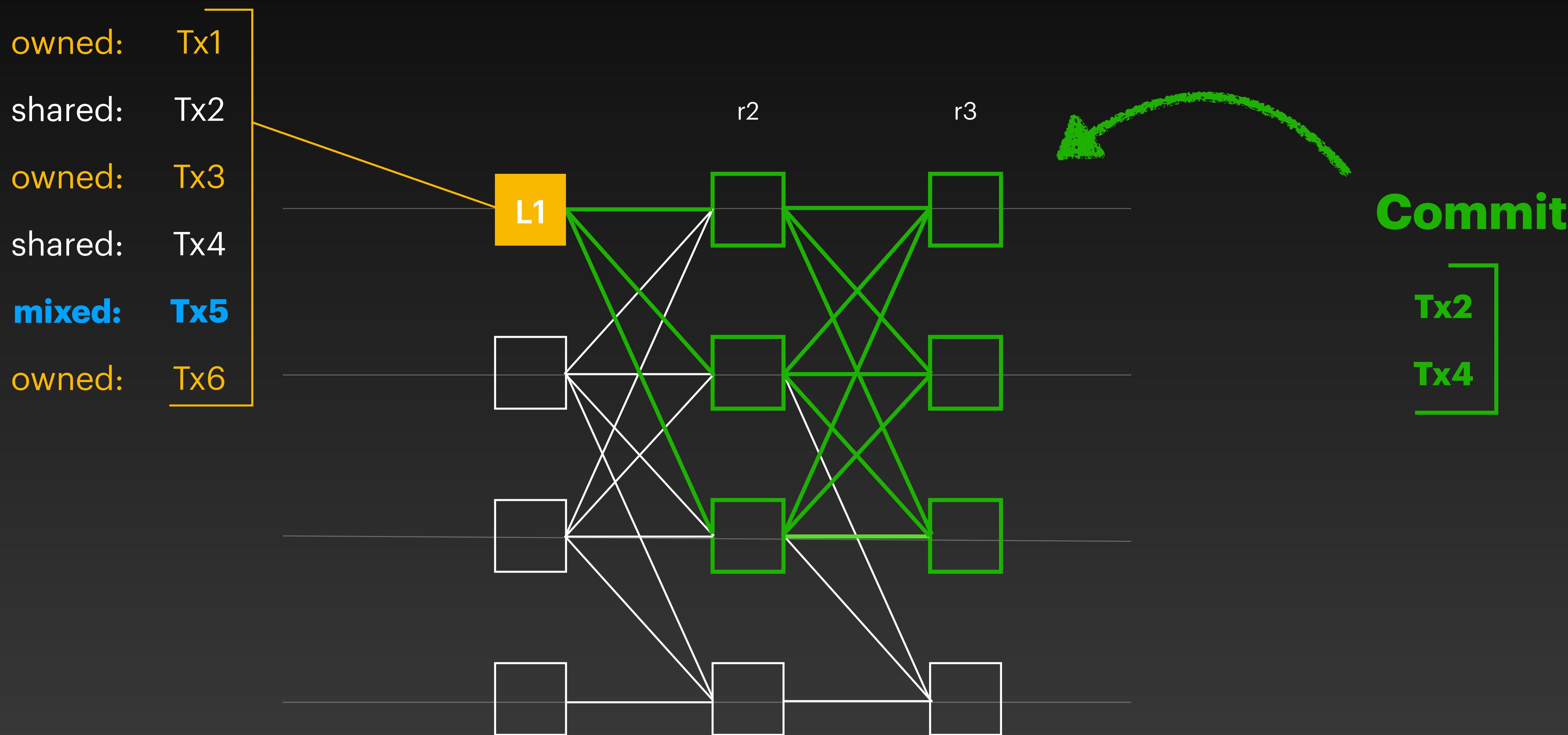
Fast Path Finality (2)



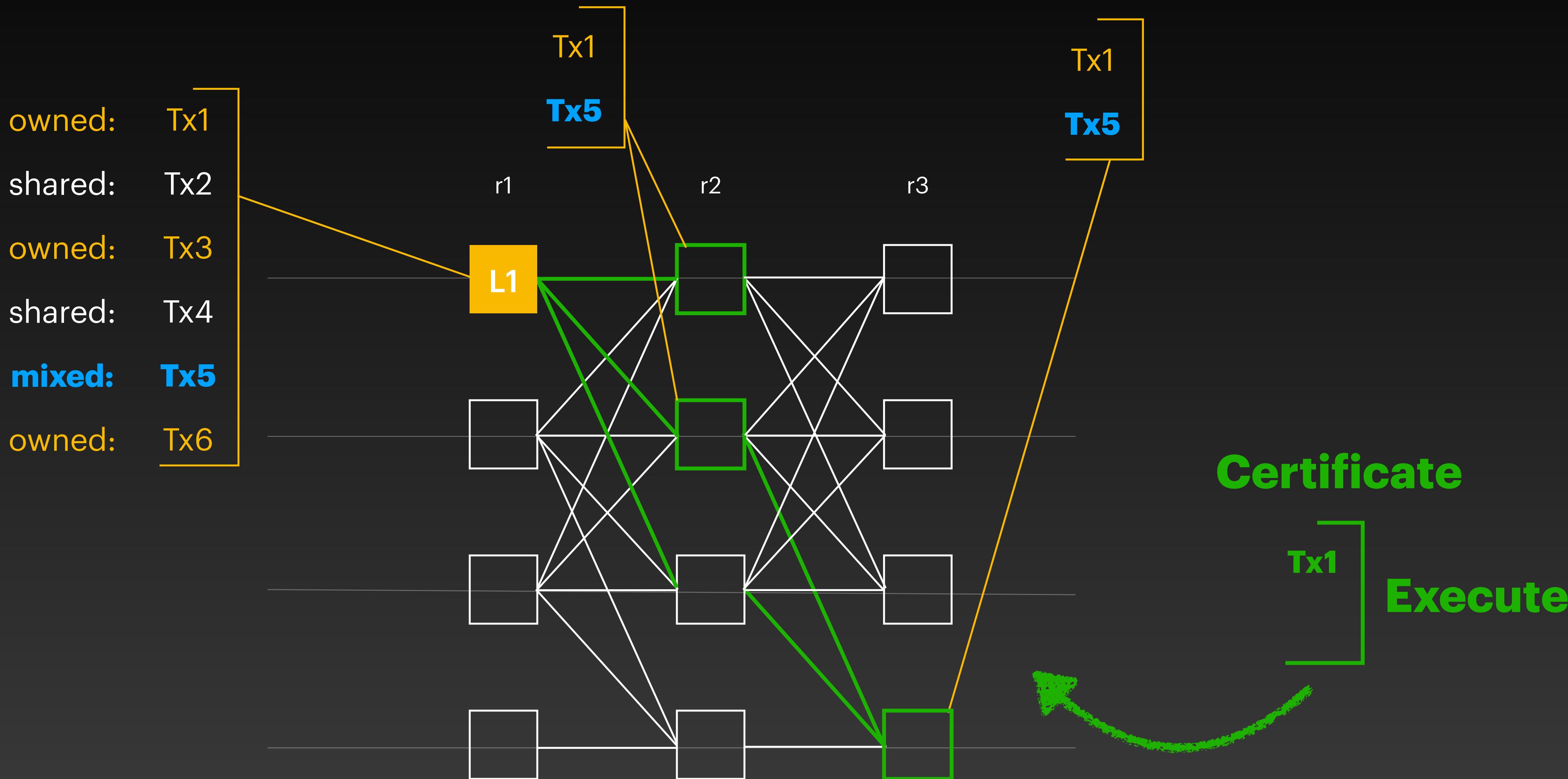
Mixed-Objects Transactions



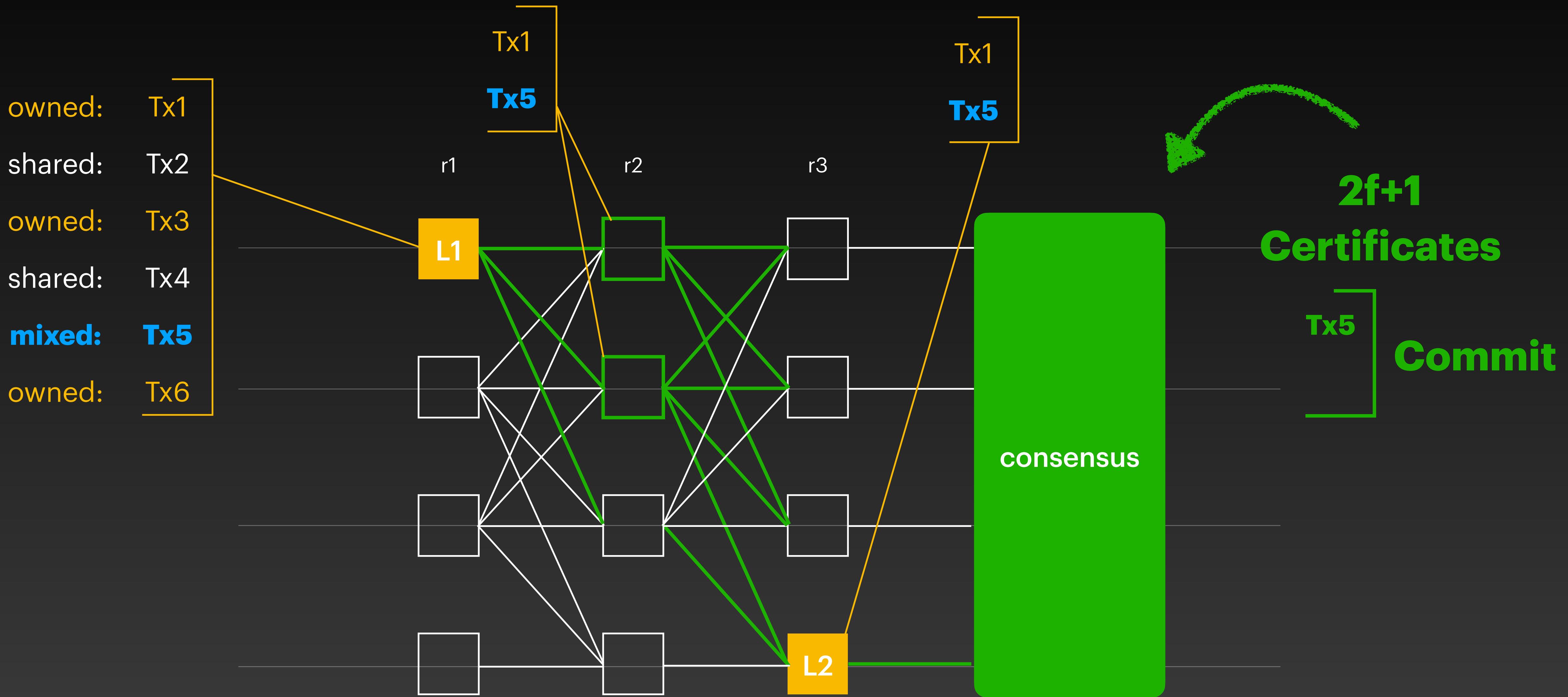
Mixed-Objects Transactions



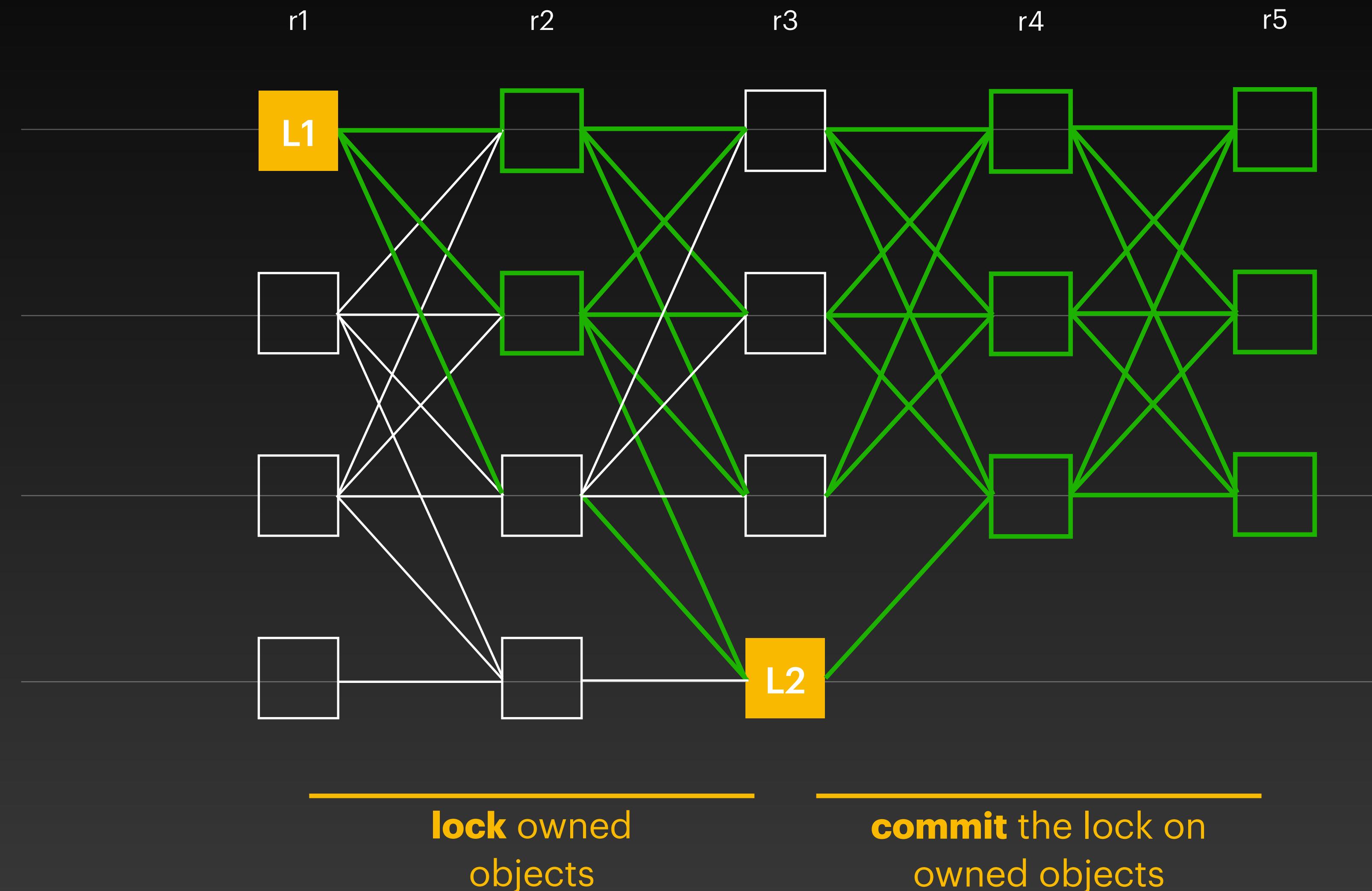
Mixed-Objects Transactions



Mixed-Objects Transactions



Mixed-Objects Transactions

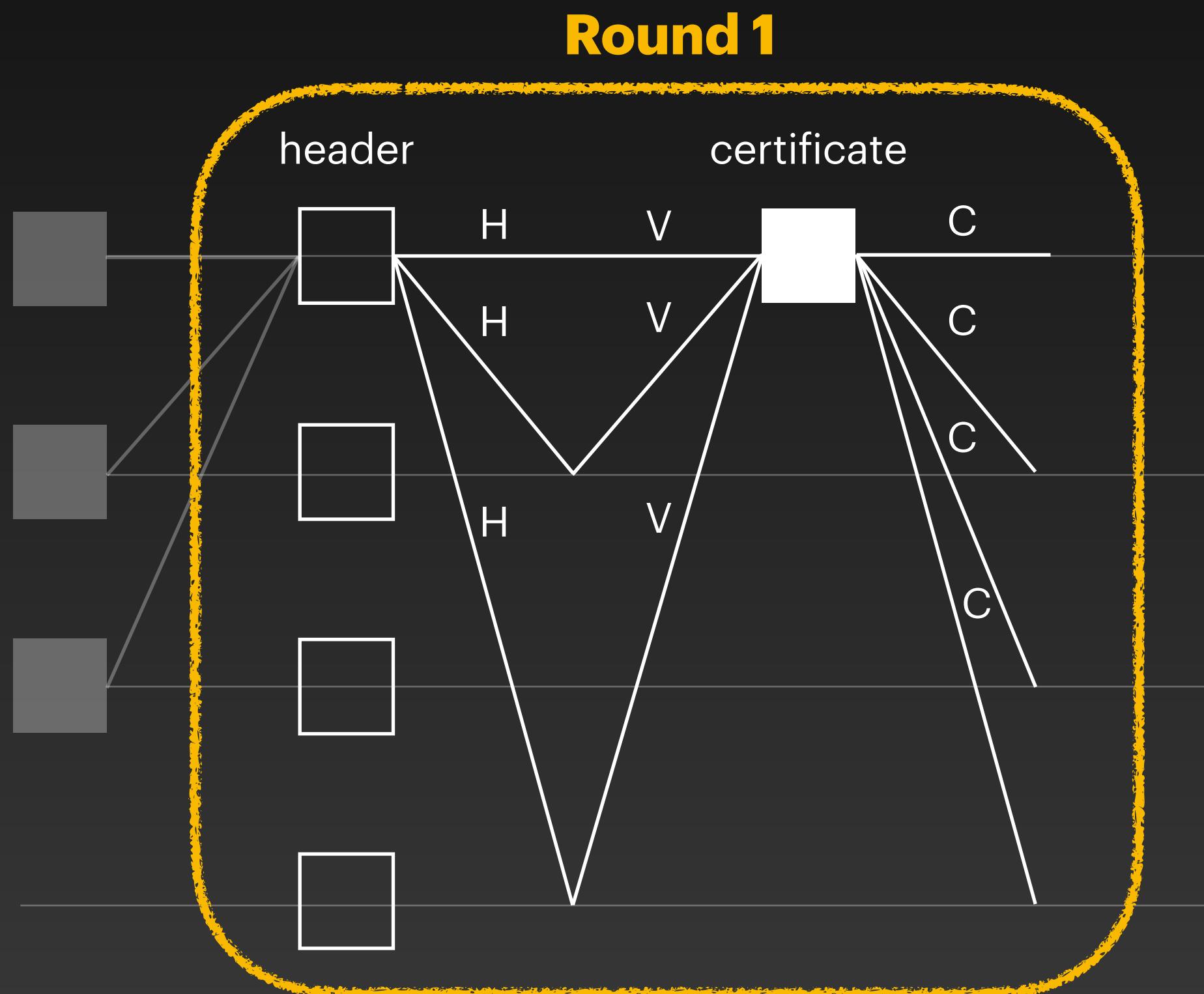


Narwhal vs Mysticeti

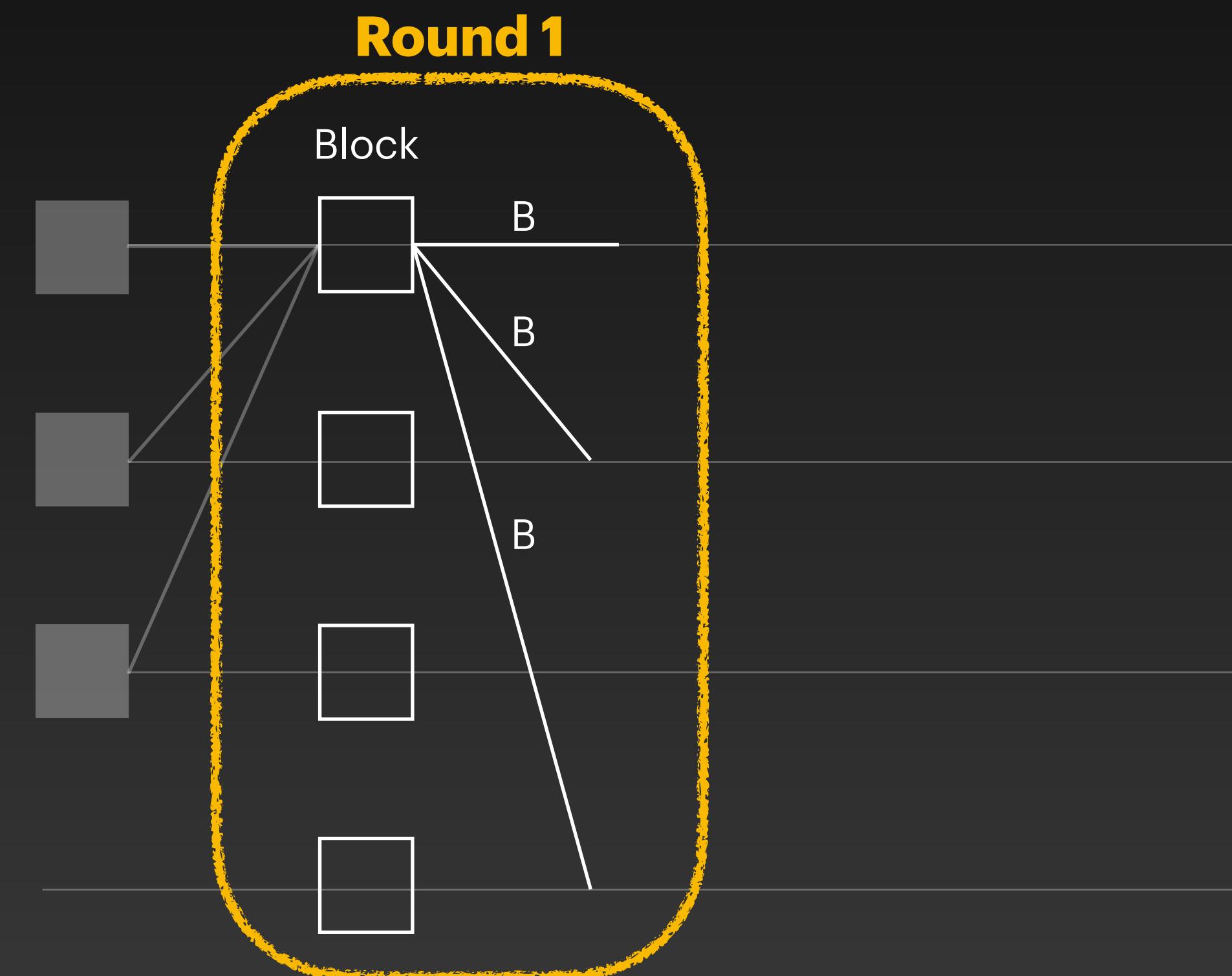
Key differences & Insight

Narwhal vs Mysticeti

Narwhal

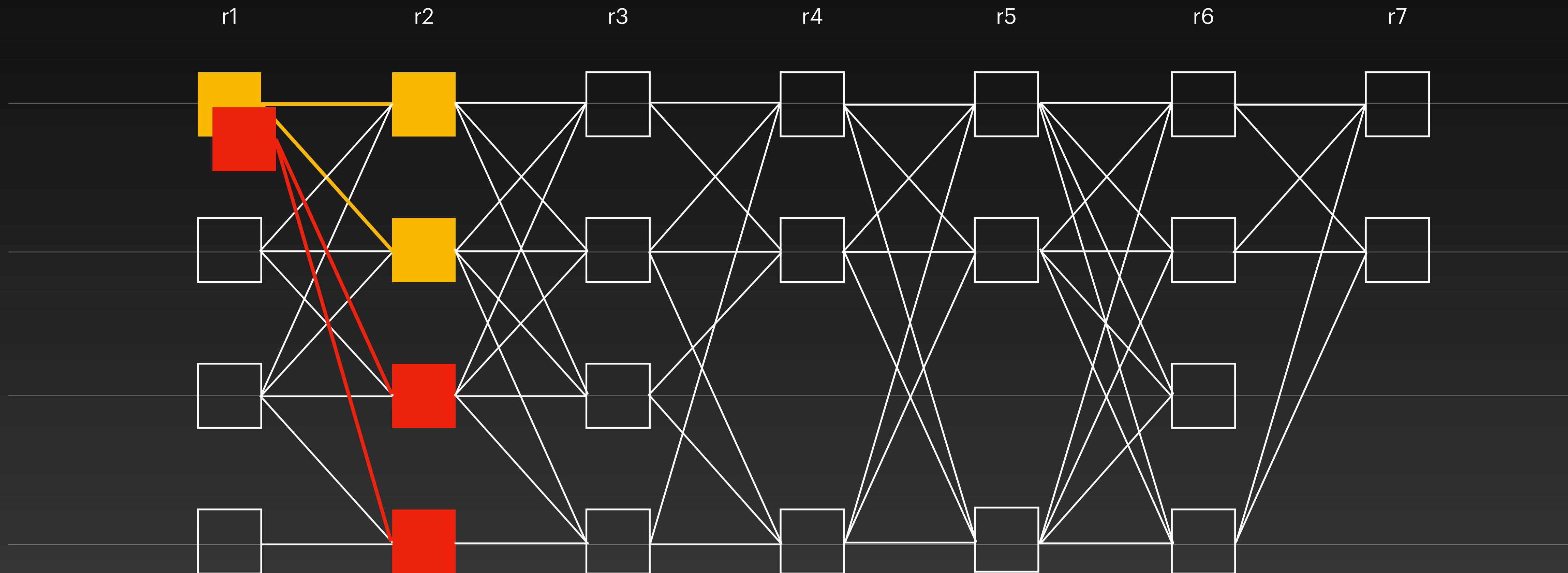


Mysticeti



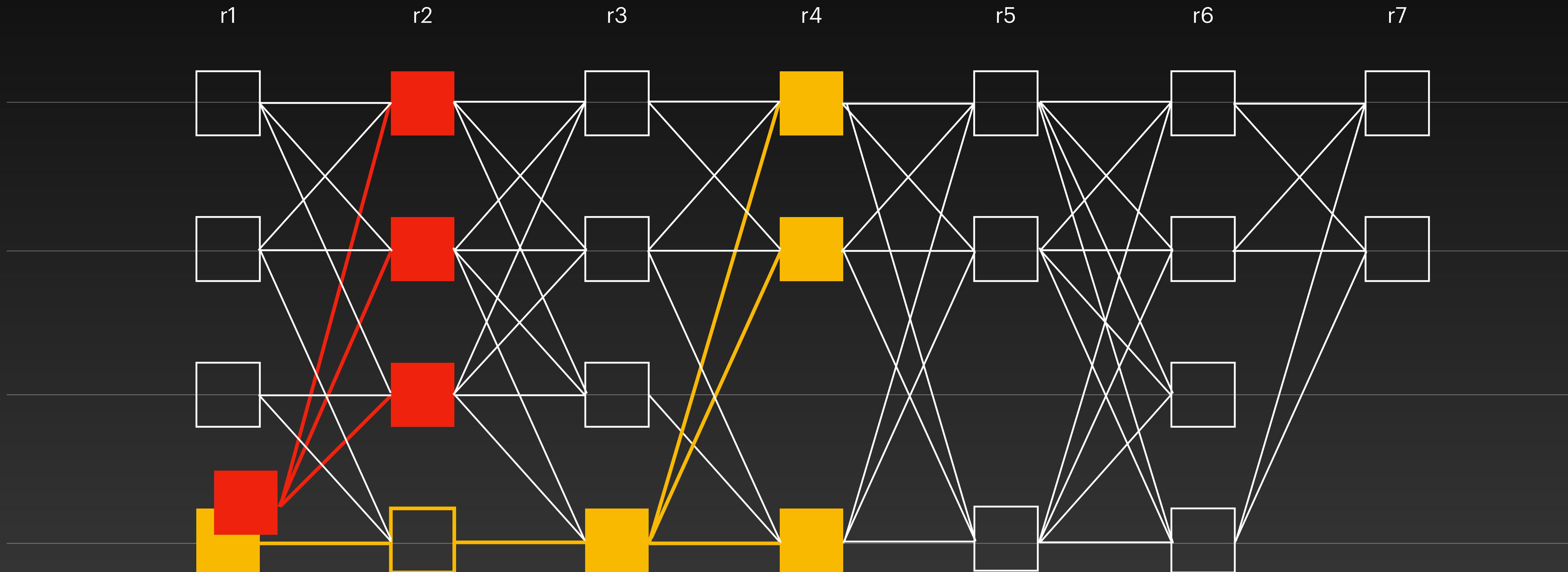
Main Challenge

Possible equivocations



Main Challenge

Possible equivocations (even with $2f+1$ support)



Decision Rules

Upon interpreting the DAG...

Bullshark

- A leader is **Commit** or not
- Either directly or indirectly (recursion)

Mysticeti

- A leader is **Commit**, **Skip**, or **Undecided**
- Either directly or indirectly (recursion)

Linear Chain vs DAG

Quorum-Based Consensus

Linear-Chain

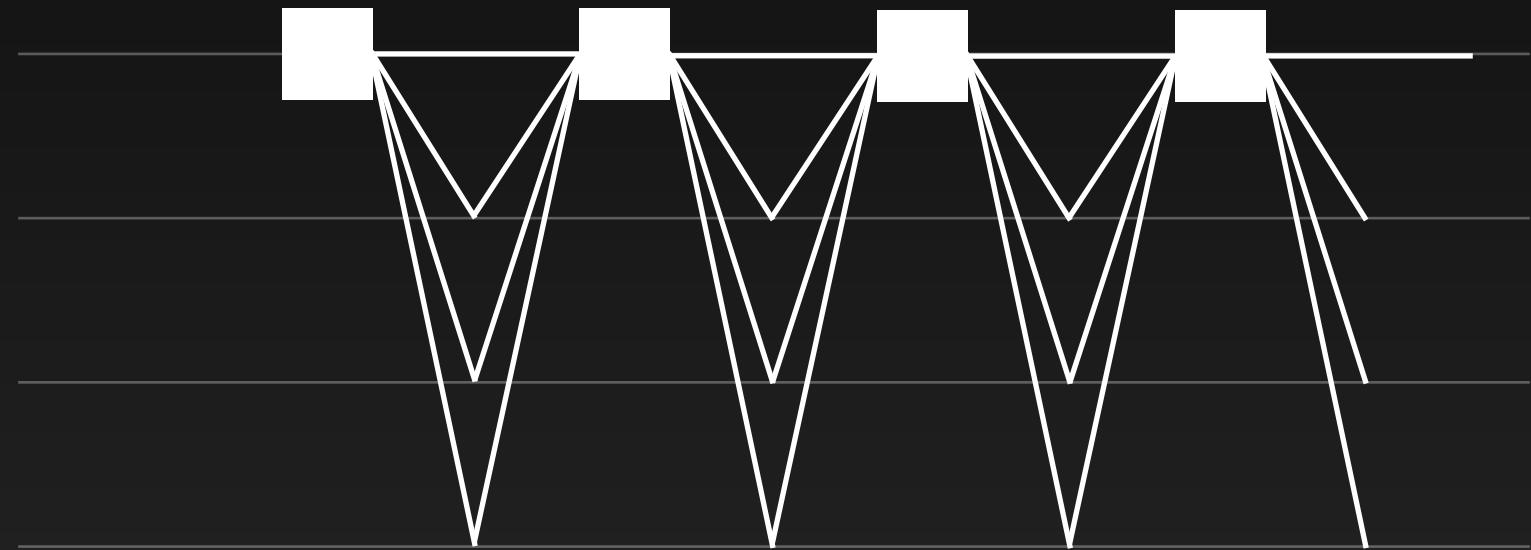
- Low latency
- Fragile to faults
- Complex leader-change

DAG-Based

- High latency
- Robust against faults
- No/Simple leader-change

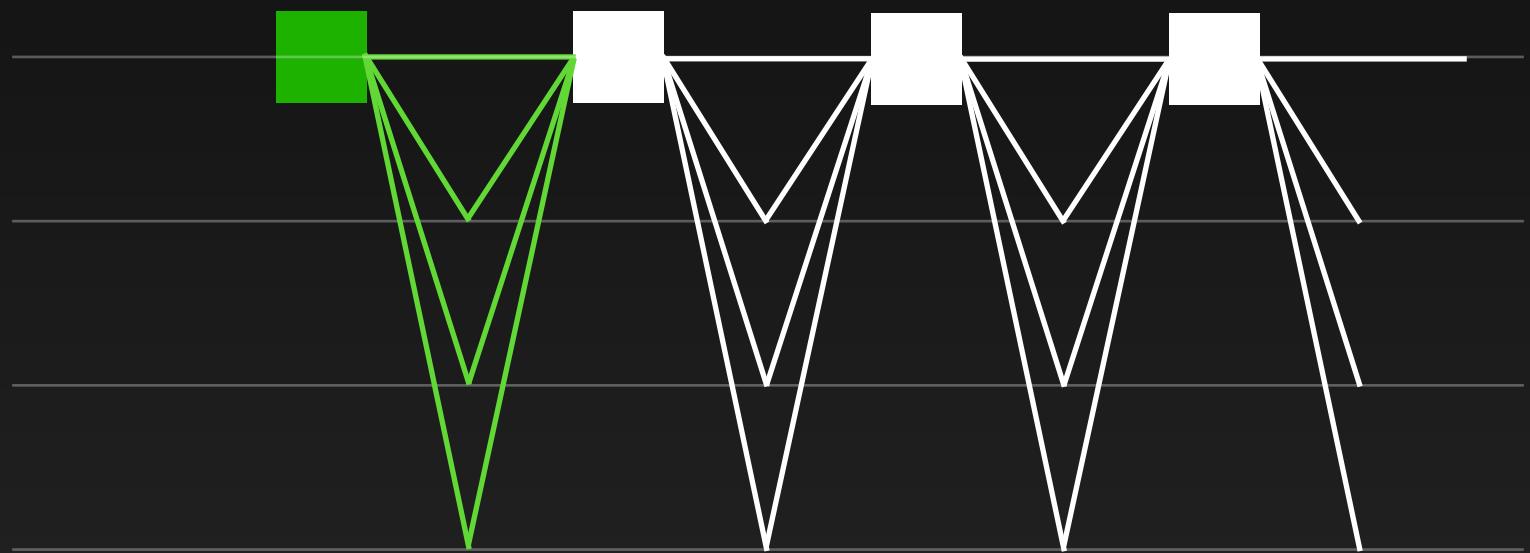
Linear-Chain Consensus

Rough overview



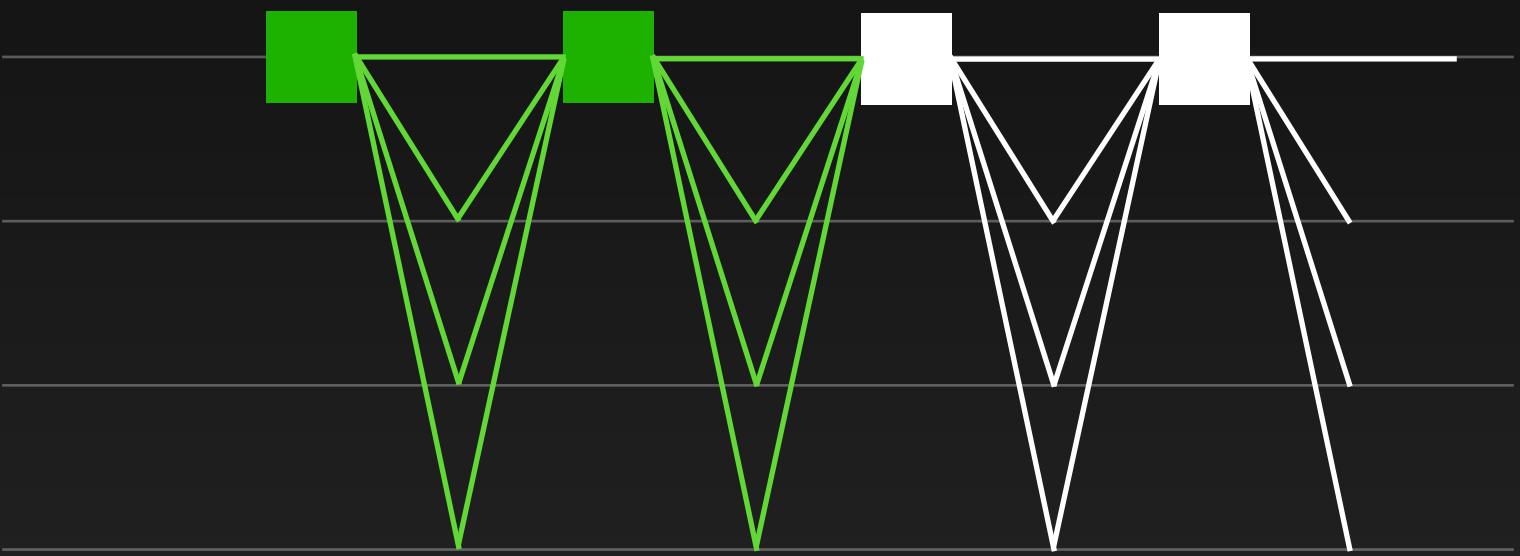
Linear-Chain Consensus

Rough overview



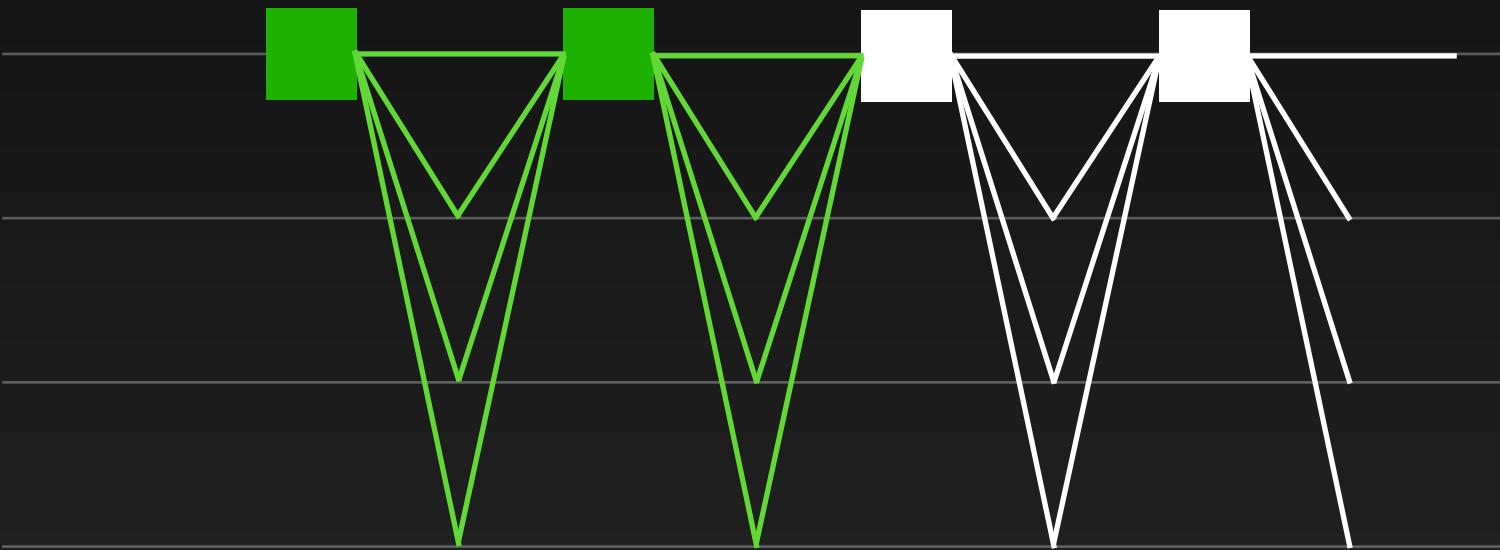
Linear-Chain Consensus

Rough overview



Linear-Chain Consensus

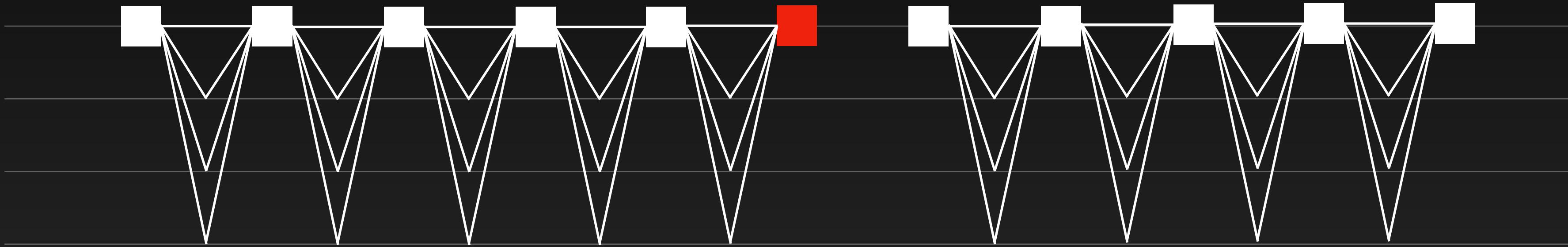
Rough overview



- The leader does all the work

Linear-Chain Consensus

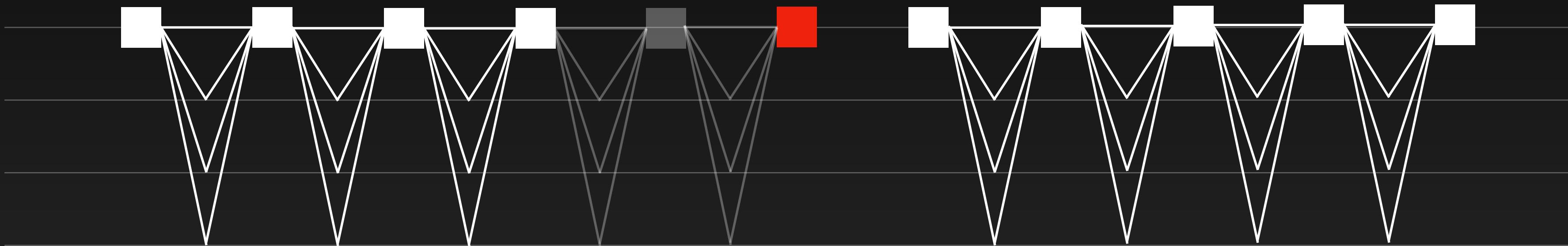
Rough overview



- The leader does all the work
- Complex leader-change

Linear-Chain Consensus

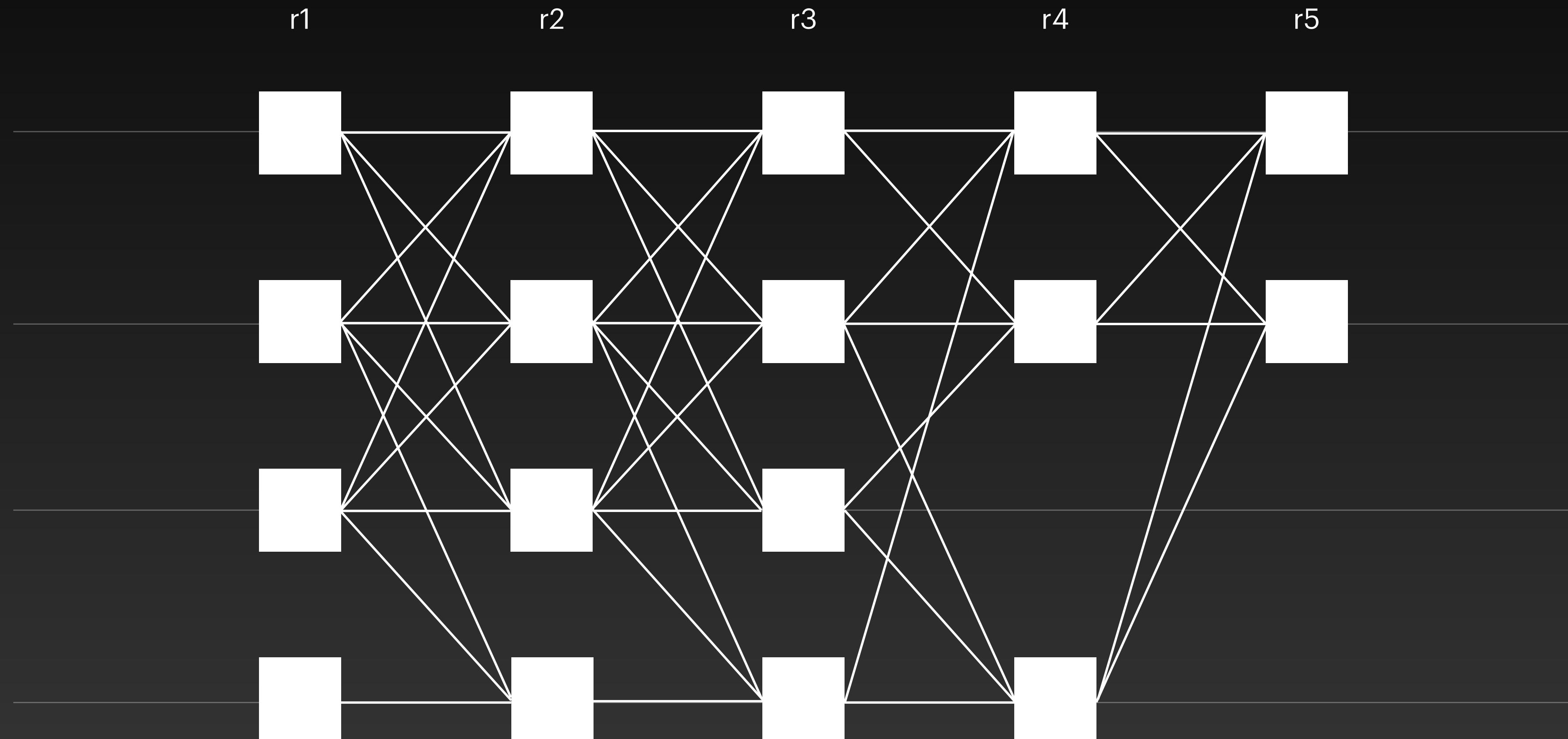
Rough overview



- The leader does all the work
- Complex leader-change

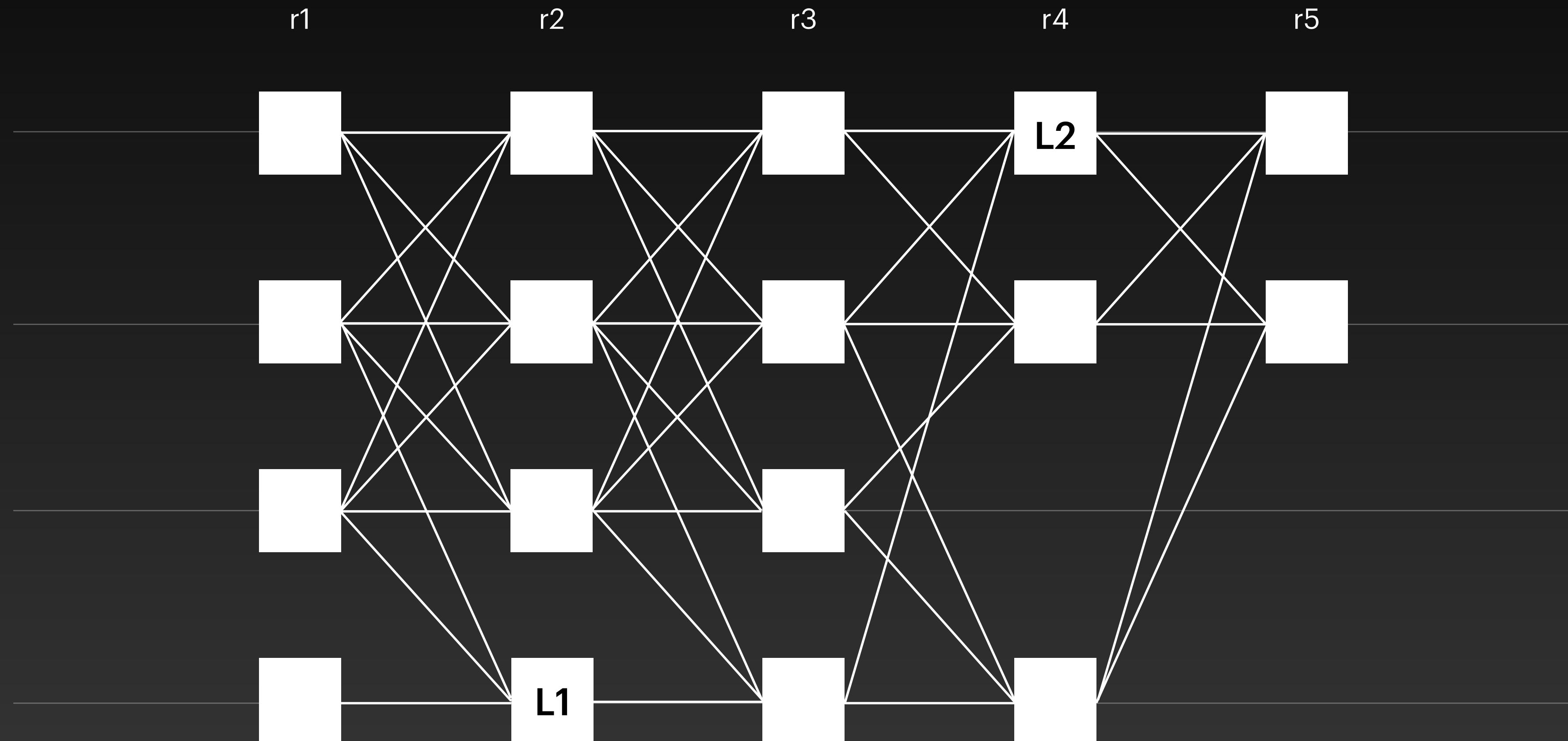
DAG-Based Consensus

Rough overview



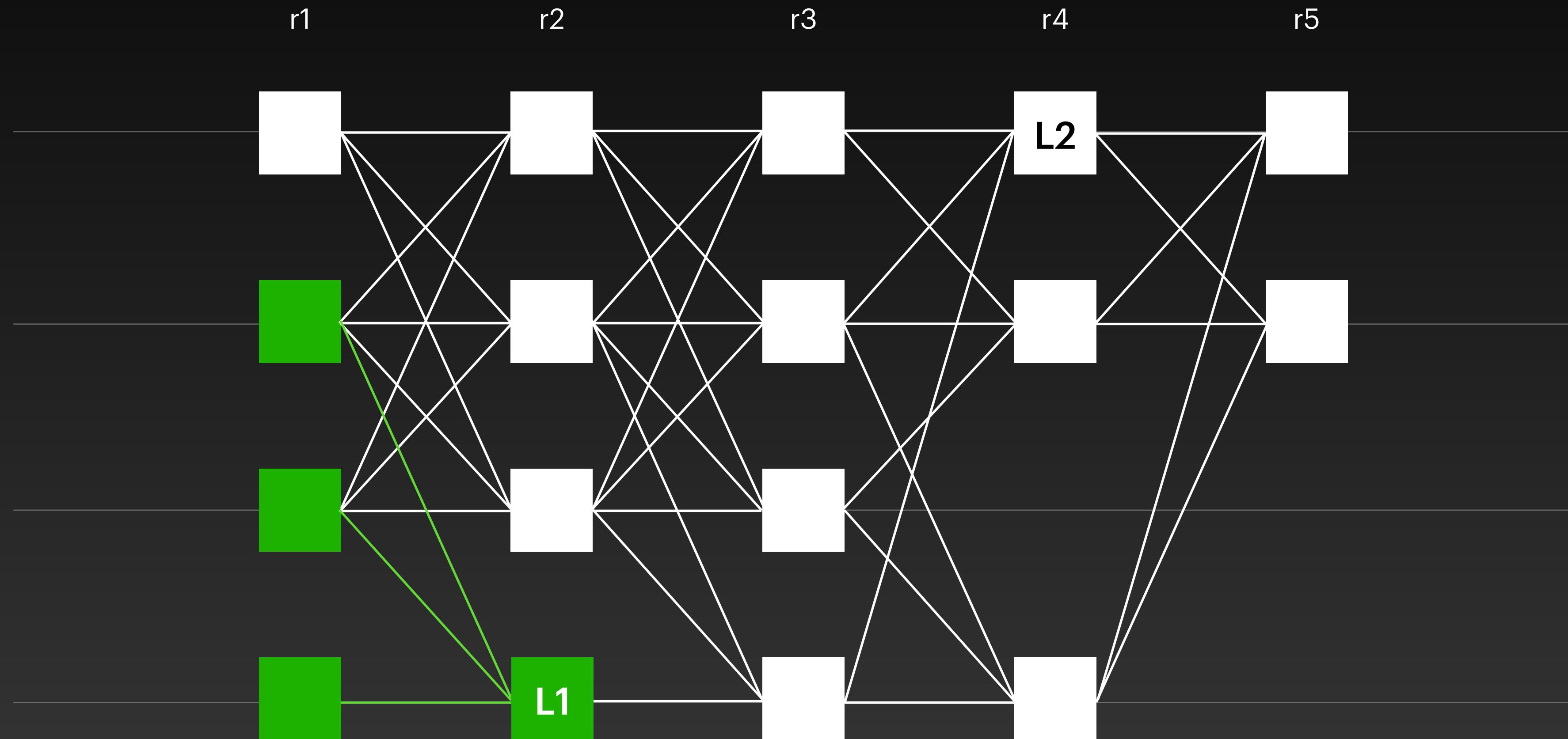
DAG-Based Consensus

Rough overview



DAG-Based Consensus

Rough overview



DAG-Based Consensus

Rough overview

