Mahi-Mahi

Low-Latency Asynchronous BFT DAG-based Consensus

Byzantine Fault Tolerance

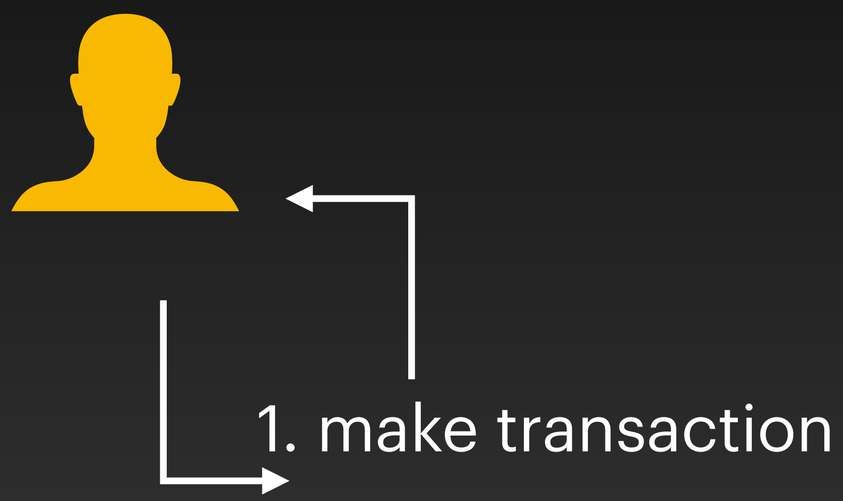
> 2/3

Byzantine Fault Tolerance

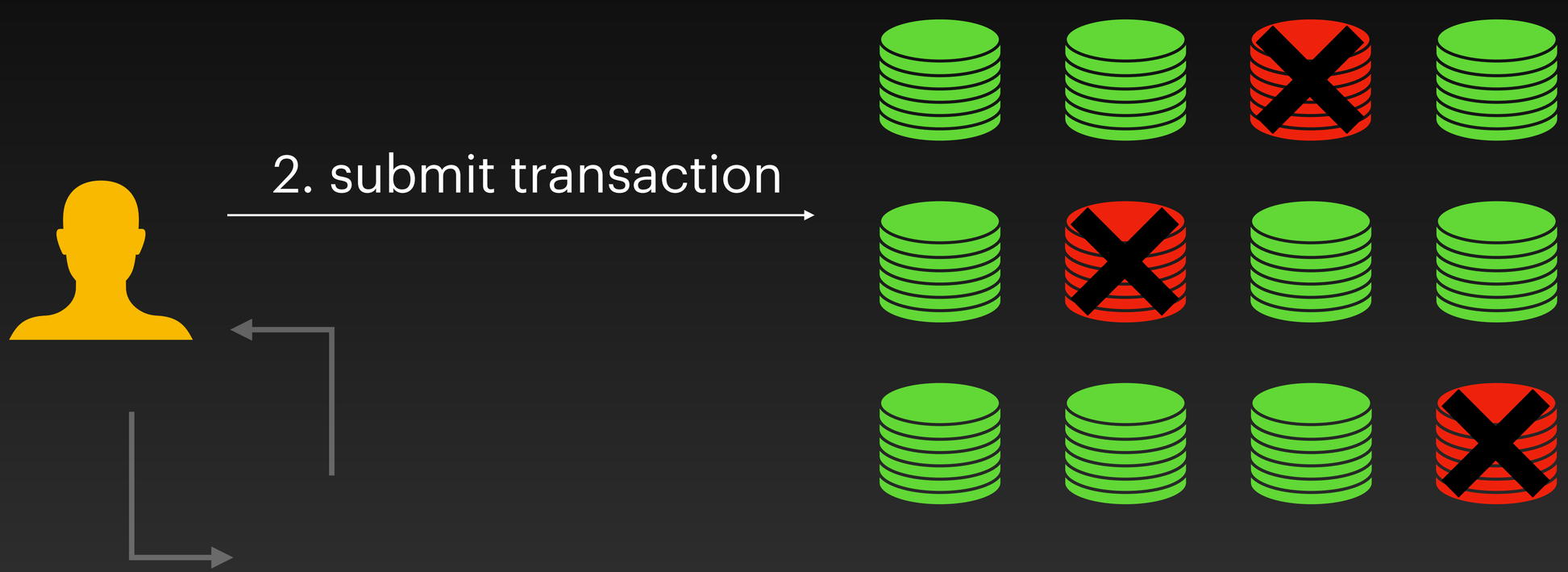
>= 2f+1

3f+1

Blockchains

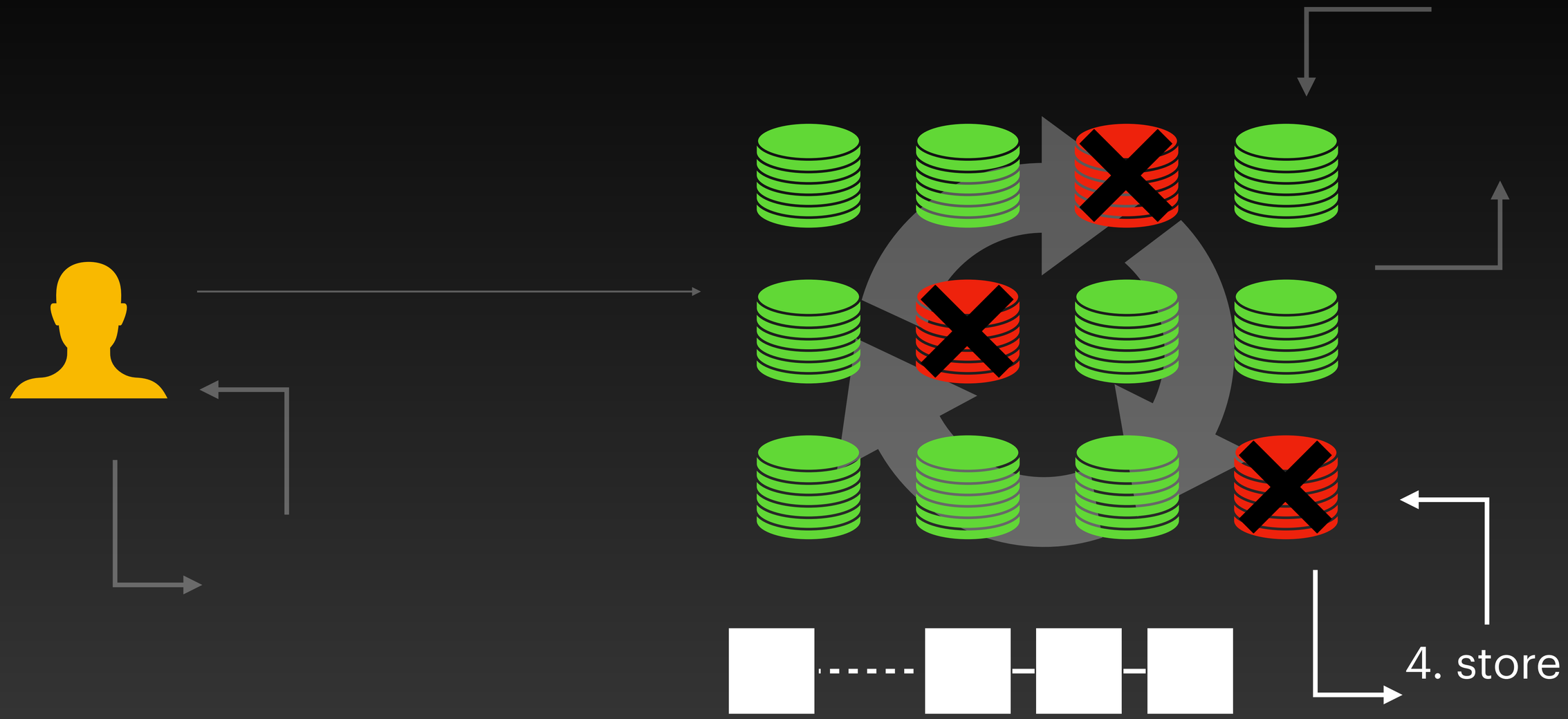


Blockchains

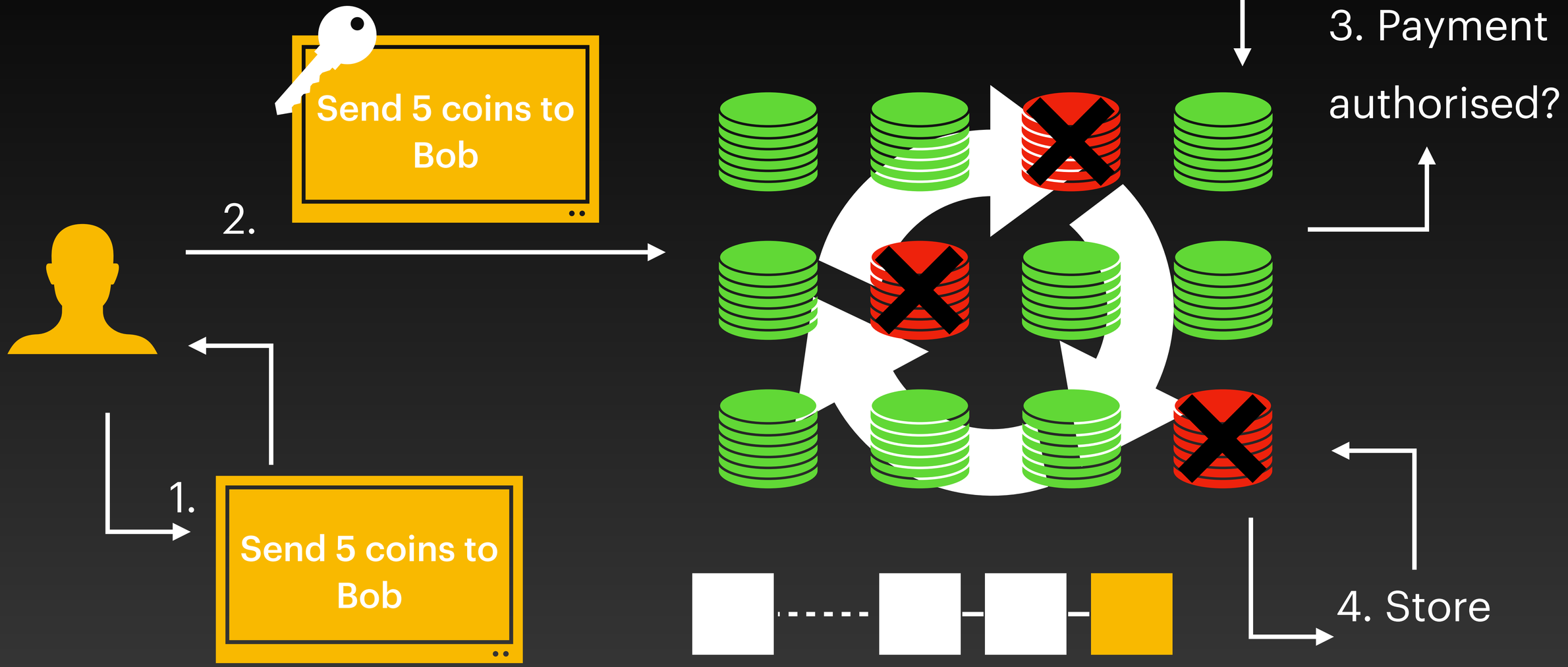


Blockchains

Blockchains



The Best Example



Keeping the Talk Short

In scope

- Ordering (quorum-based)

Not in scope

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

3 Timing Models

Synchronous

Partially Synchronous

Asynchronous

time

What do we want in a consensus protocol?

Simplicity

Dual mode is complicated

Performance

Can we improve throughput?

Resilience

Can we resist DDoS?

Can we have all three?

Recent Related Consensus Protocols

DAG-Rider: Asynchronous protocol needing 12 messages to commit

Tusk: Asynchronous protocol but needs 9 messages to commit

Mysticeti: Fast and simple protocol but only partially synchronous

FIN: Asynchronous protocol but low throughput

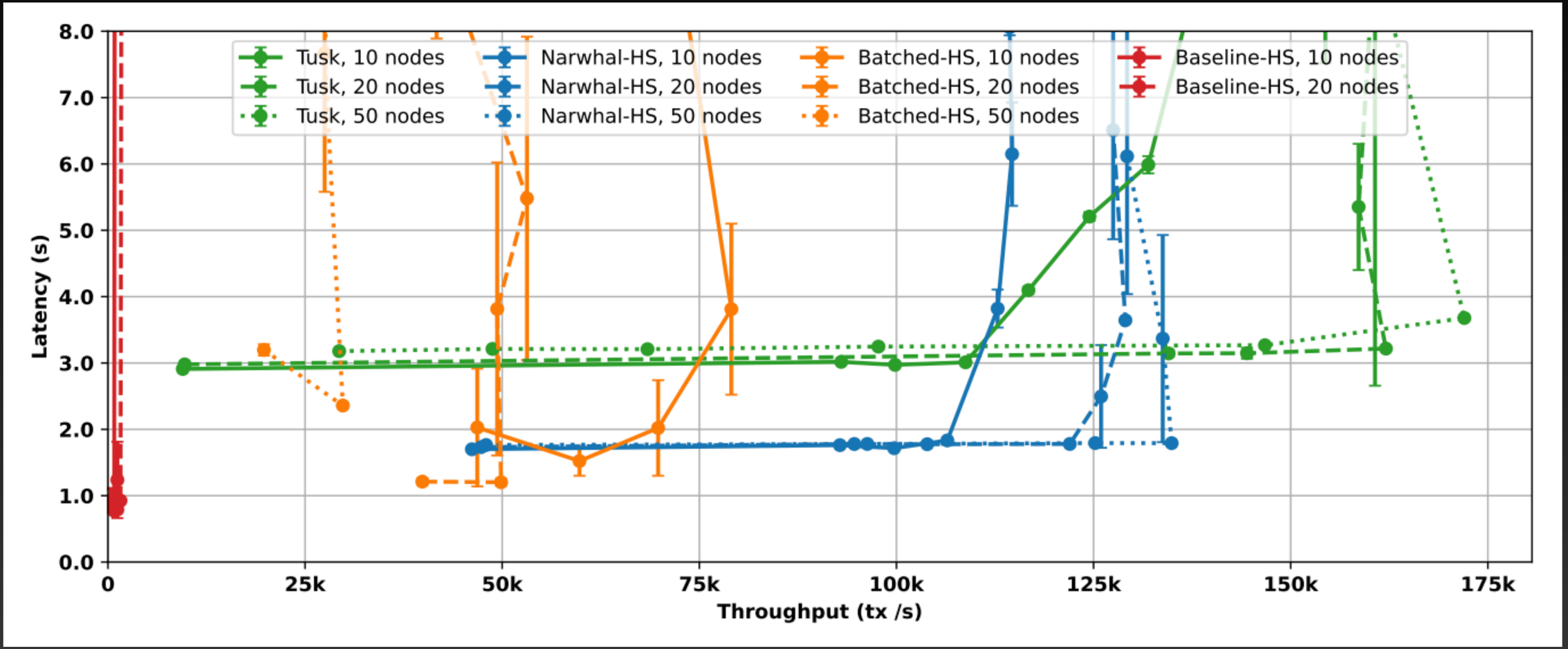
Cordial Miners: Asynchronous protocol but higher latency

Why Mahi-Mahi?

Fast, Simple, Resilient

- Is live and can progress under asynchronous conditions
- Uncertified DAG to reduce latency
- Sub-second latency with >100k throughput per second
- Simple to implement
- Can be parameterized

How did previous protocols perform?



How Fast is Mahi-Mahi?

Faster than any prior protocols!

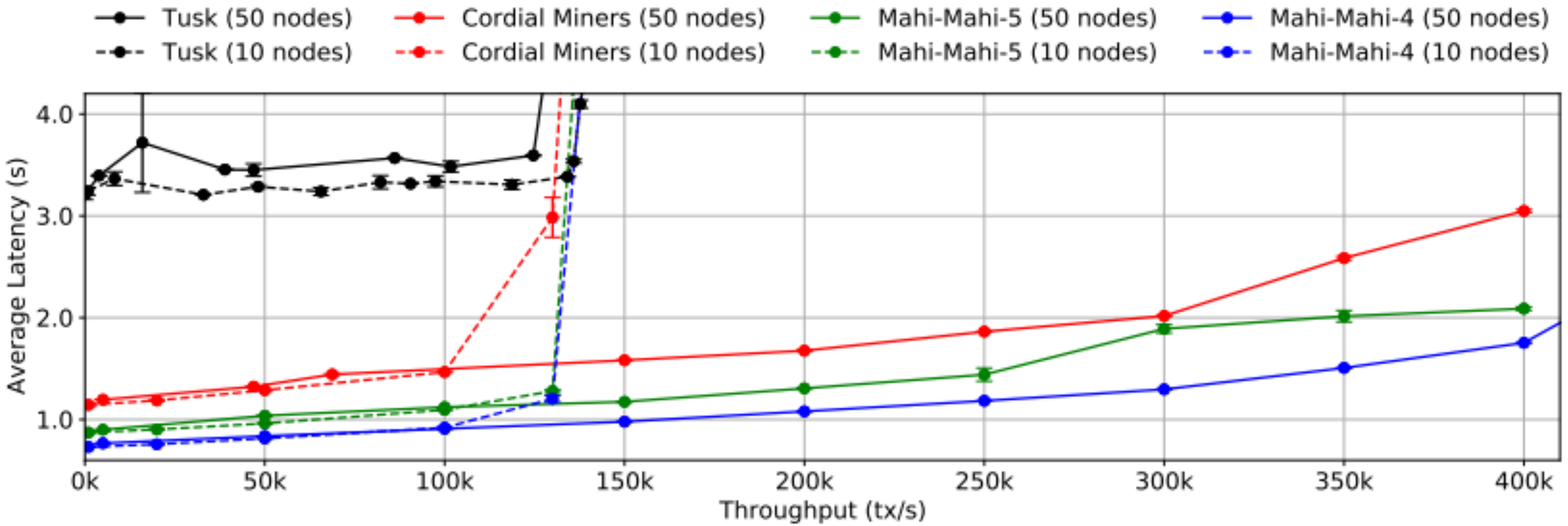


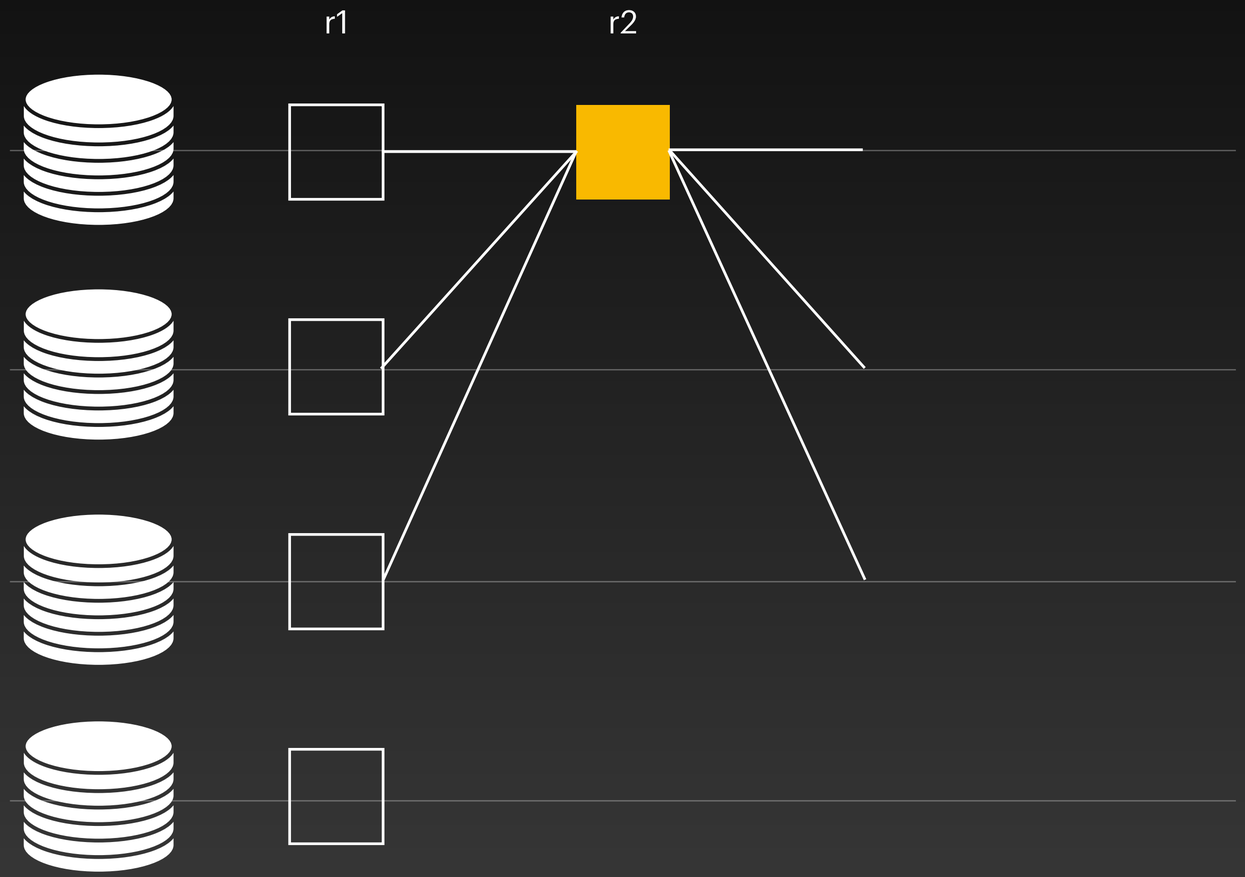
Fig. 3: Comparative throughput-latency performance of Mahi-Mahi, Tusk, and Cordial Miners. WAN measurements with 10 and 50 validators. No validator faults. 512B transaction size.

The Mahi-Mahi DAG

Implicit Certification Consensus

Uncertified DAGs

Block Creation

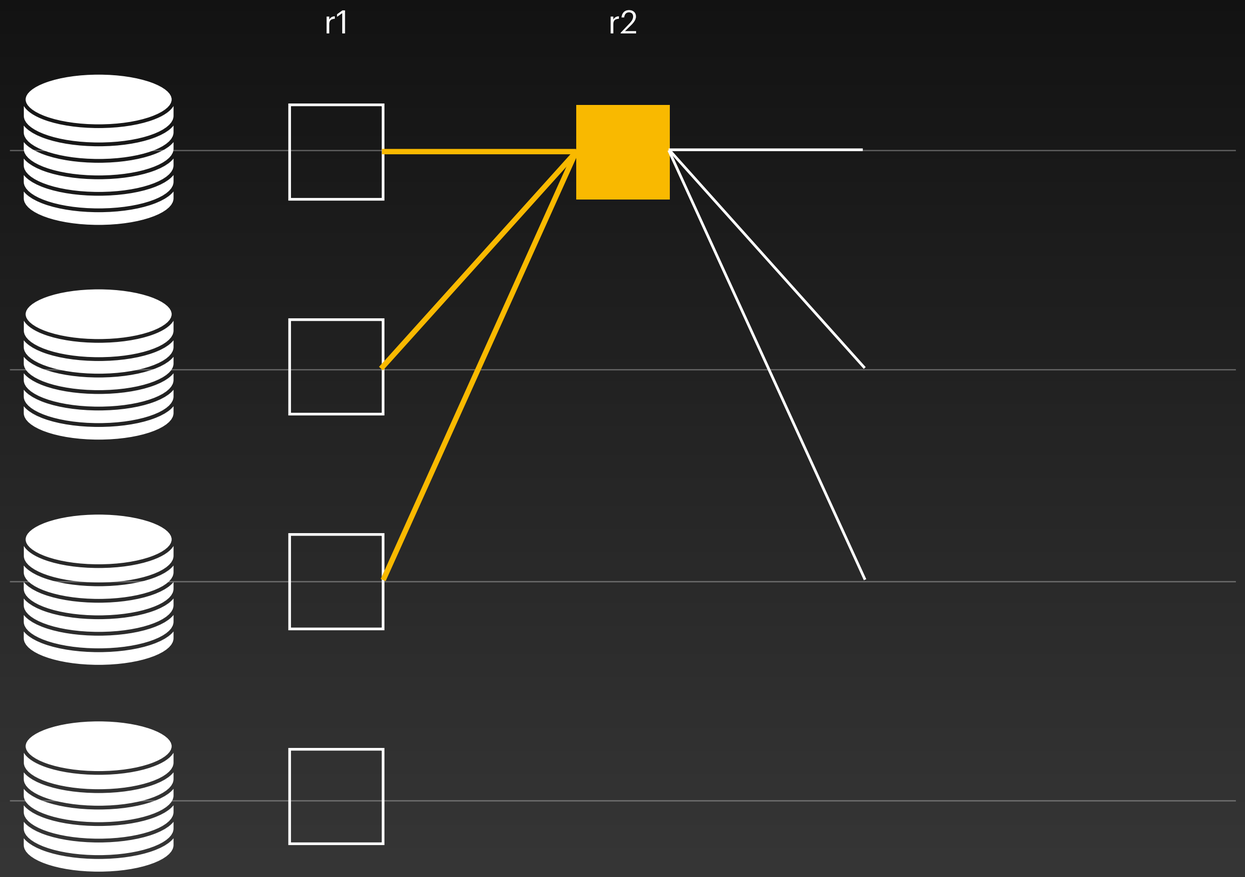


What’s in a block:

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

Uncertified DAGs

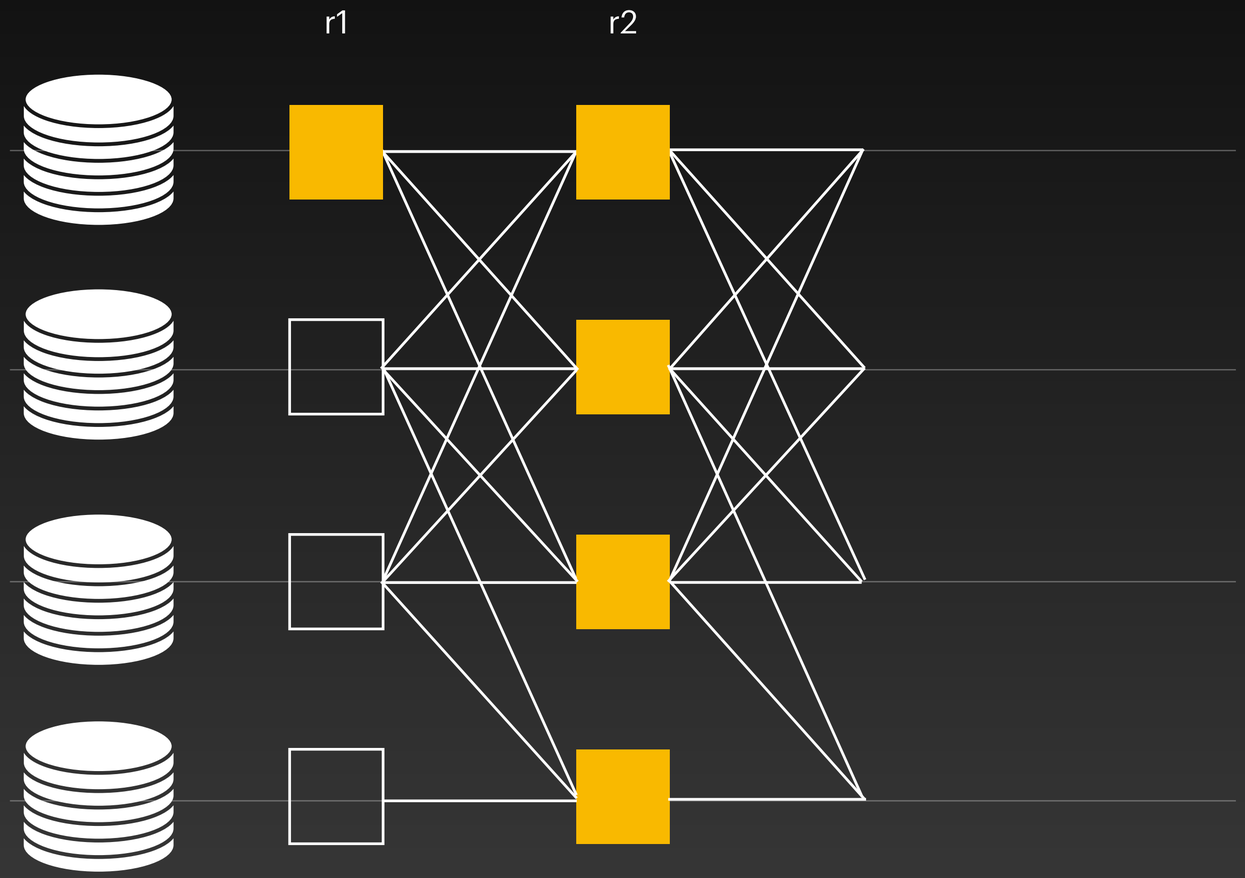
Rule 1: Link to 2f+1 parents



- Total nodes: **3f+1 = 4**
- Quorum: **2f+1 = 3**
- Generate randomness to retrospectively elect leaders with common coin

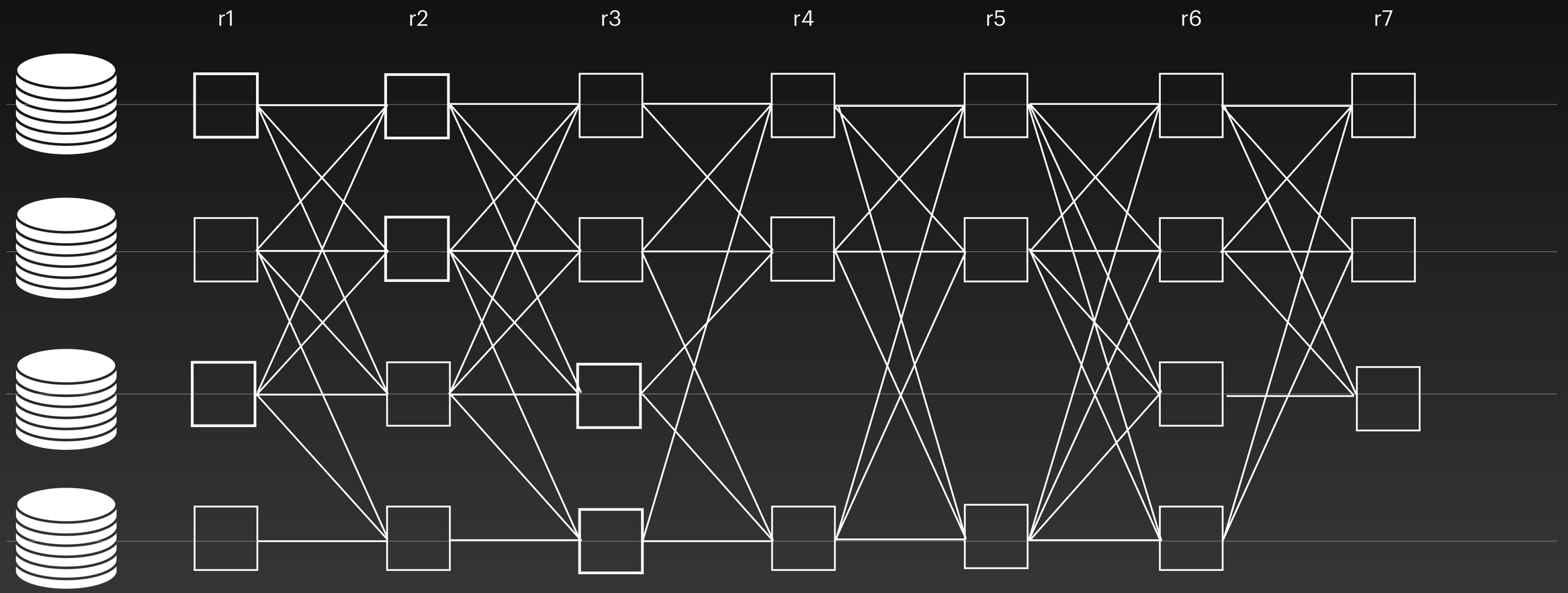
Uncertified DAGs

Rule 2: All nodes run in parallel



Uncertified DAGs

An Example



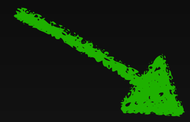
Main Ingredient:

All messages embedded in the DAG

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

Certified vs Uncertified DAG

Mahi-Mahi is an Uncertified DAG



Certified DAG

- Every round is a reliable broadcast
- 2f+1 quorum

Uncertified DAG

- Every round is a best effort dissemination
- 2f+1 quorum

How to we elect a Leader?

Blocks should be committed

Need Randomness

- Cannot reach consensus without randomness (FLP Theorem)
- Randomness retrospectively determines a leader

Use Common Coin

- Generates a random number for honest participants
- Require 2f+1 shares to execute coin toss.
- Can be implemented with threshold signatures

What do we want in a Consensus Protocol?

Validity

If an honest participant outputs a message, it is eventually delivered to every honest participant.

Total Order

All honest participants will output a consistent sequence of messages.

Agreement

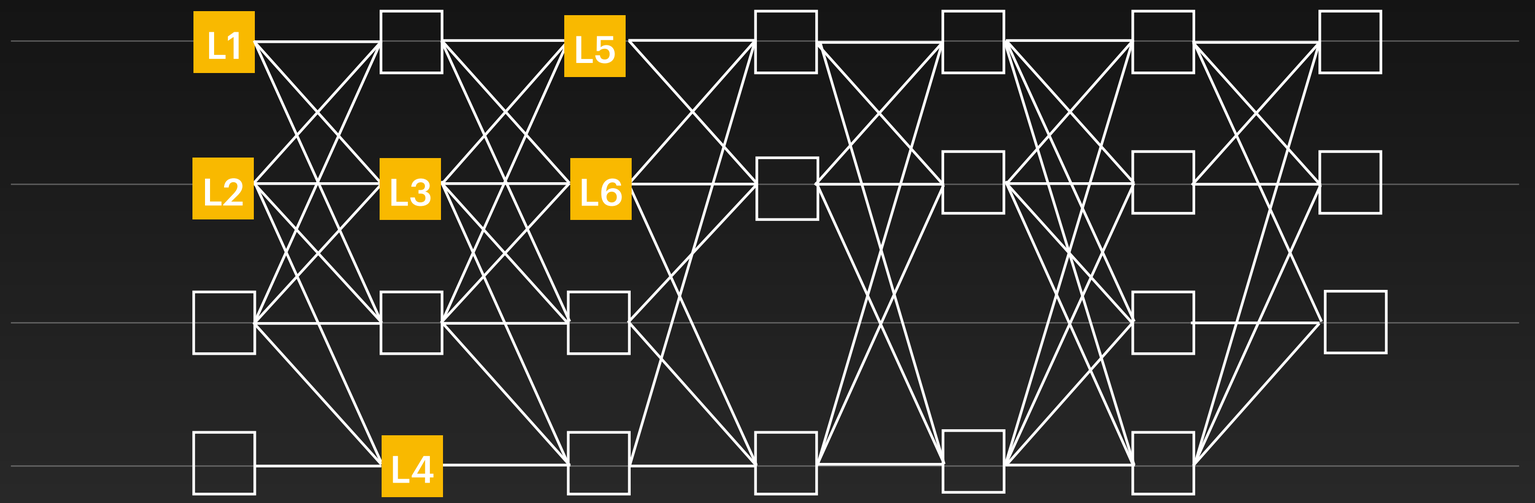
If an honest participant outputs a message, eventually every honest participant will output this message.

Integrity

Spurious messages cannot be attributed to honest participants.

End Goal

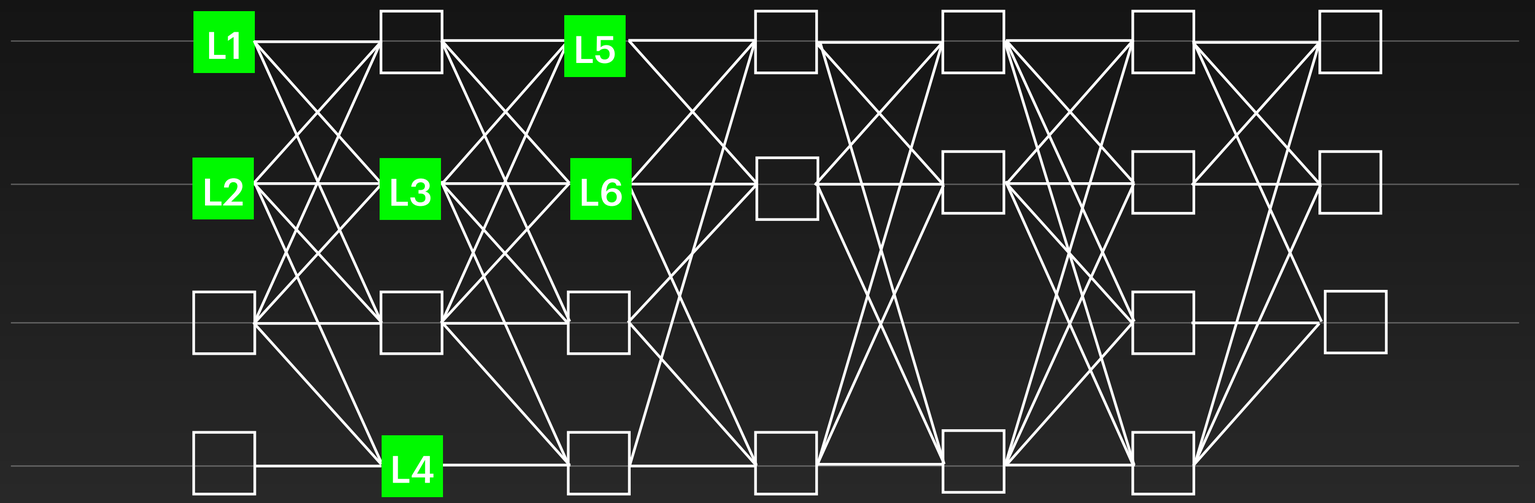
Ordering leaders



- We focus on ordering leaders: L1 L2 L3 L4 L5 L6

End Goal

Ordering leaders



- We focus on ordering leaders: L1 L2 L3 L4 L5 L6
- Linearising the sub-DAG is simple

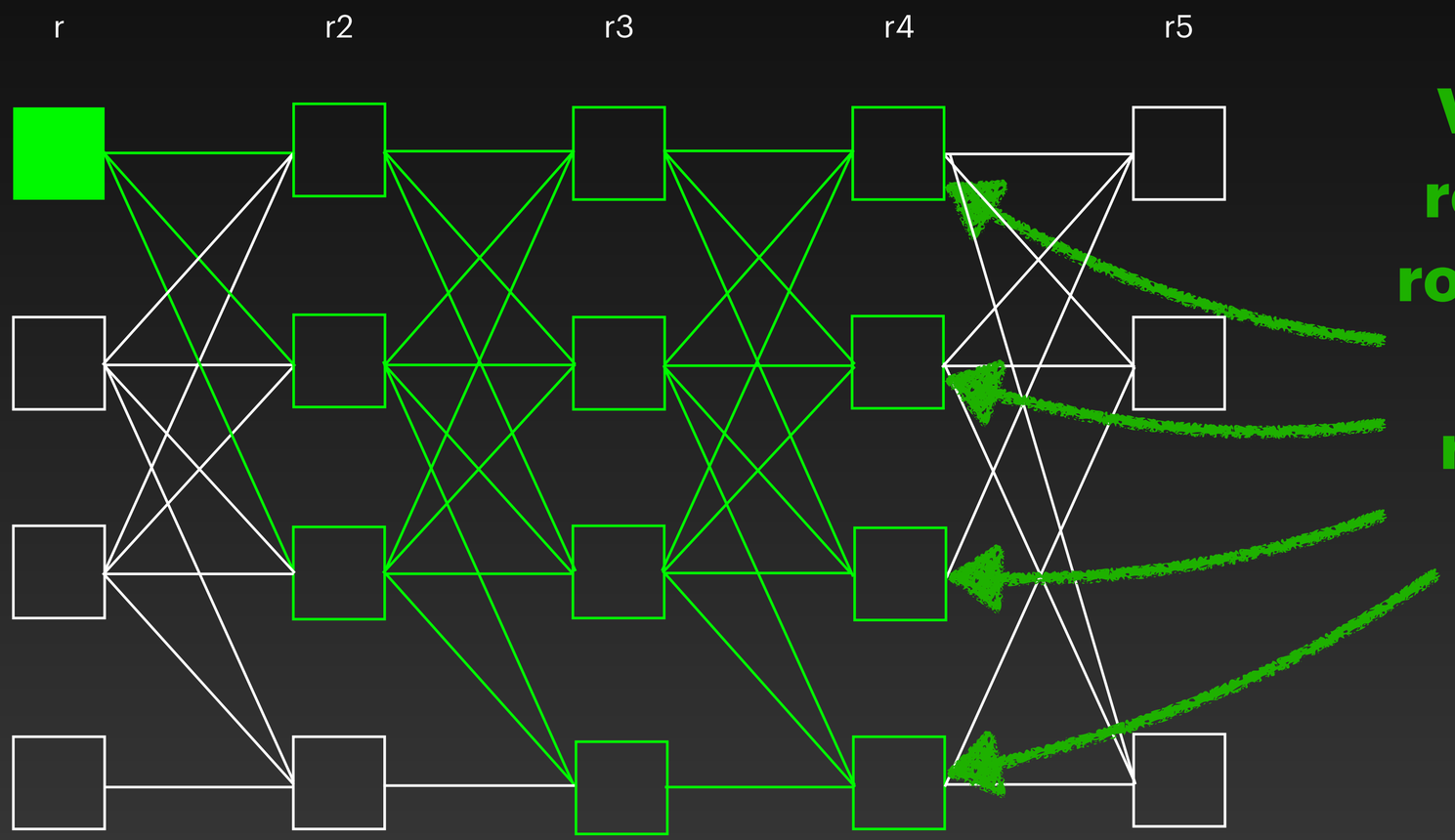
Components for Consensus

How to commit blocks?

Blocks can do the following:

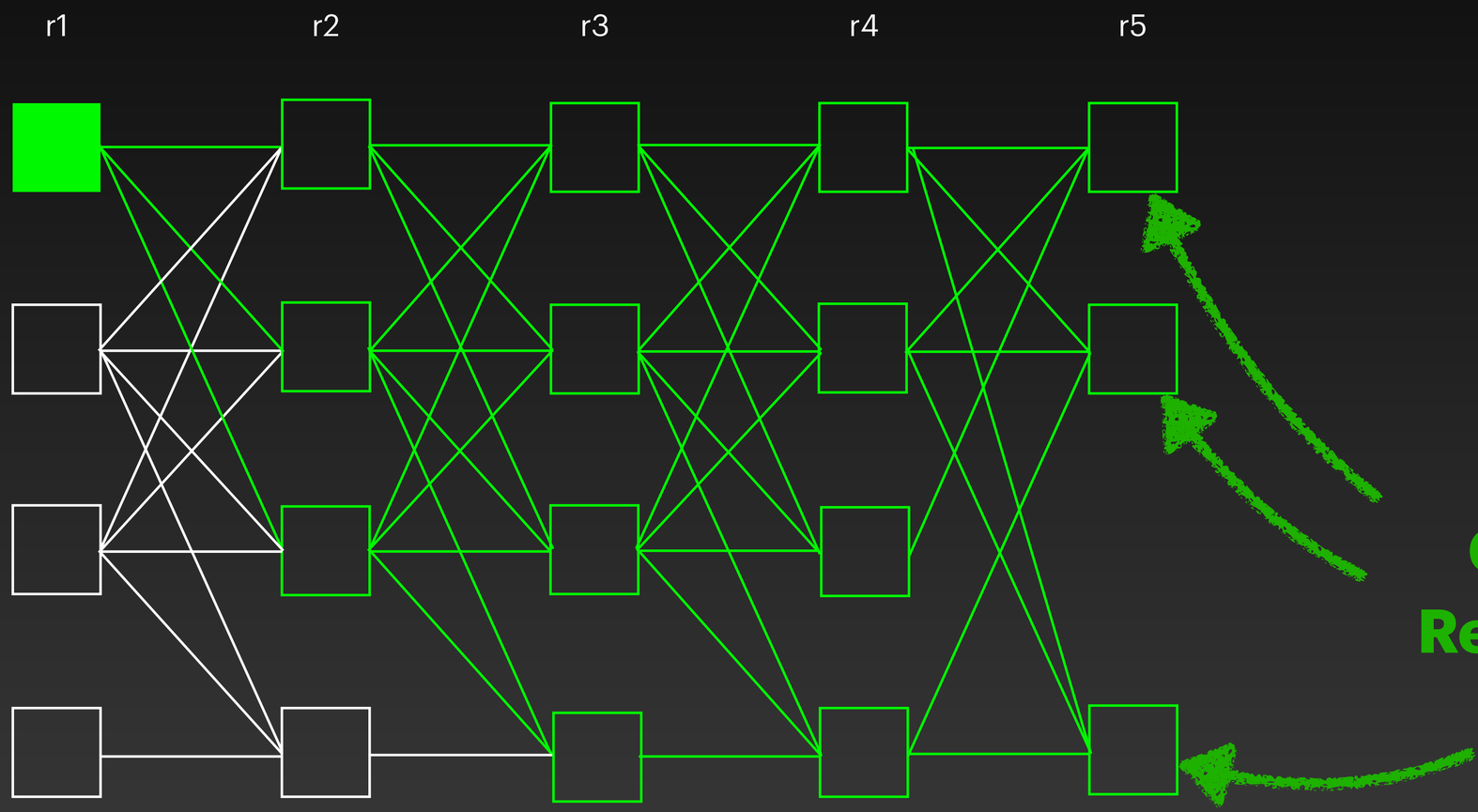
Vote	Certificate	Blame
Path from r+3 to r	Reference 2f+1 votes for a block	No path in r+4 to r

Voting for a block



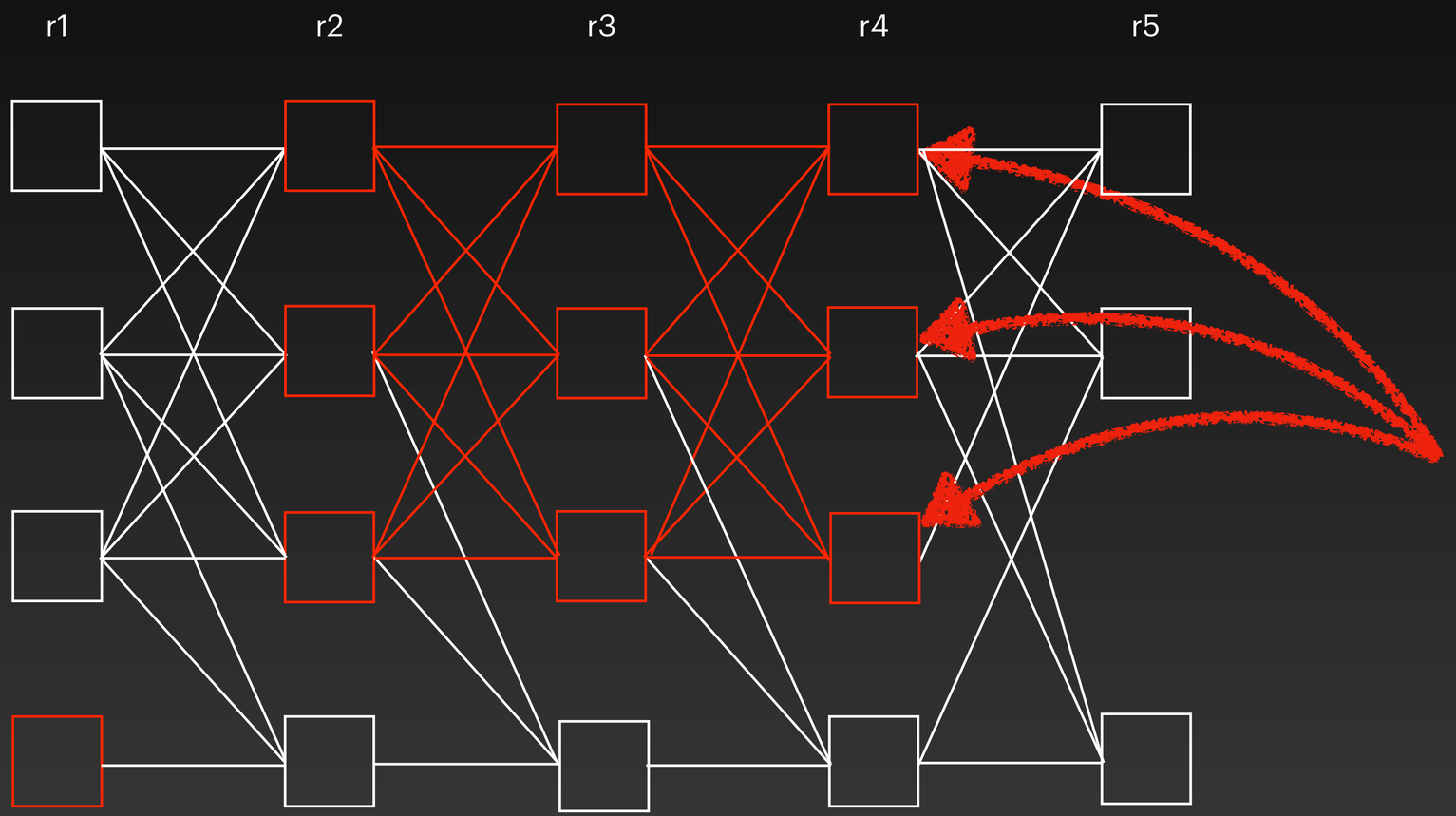
Vote: A path from round r+3 block to round r block *make consistent with round notation in dag

Certificate for a block



Certificate: Reference 2f+1 votes

Blames for a block

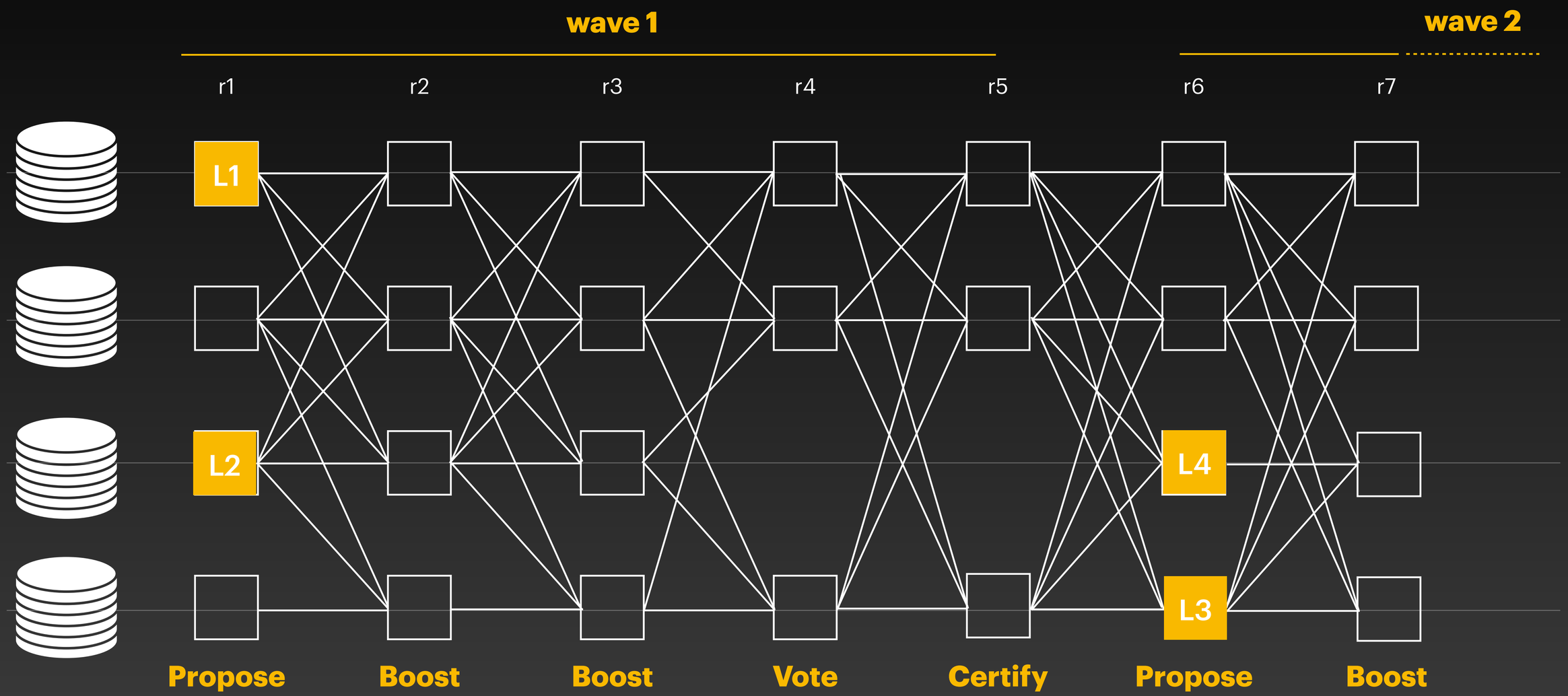


Blame: No path from r+3 block to r block

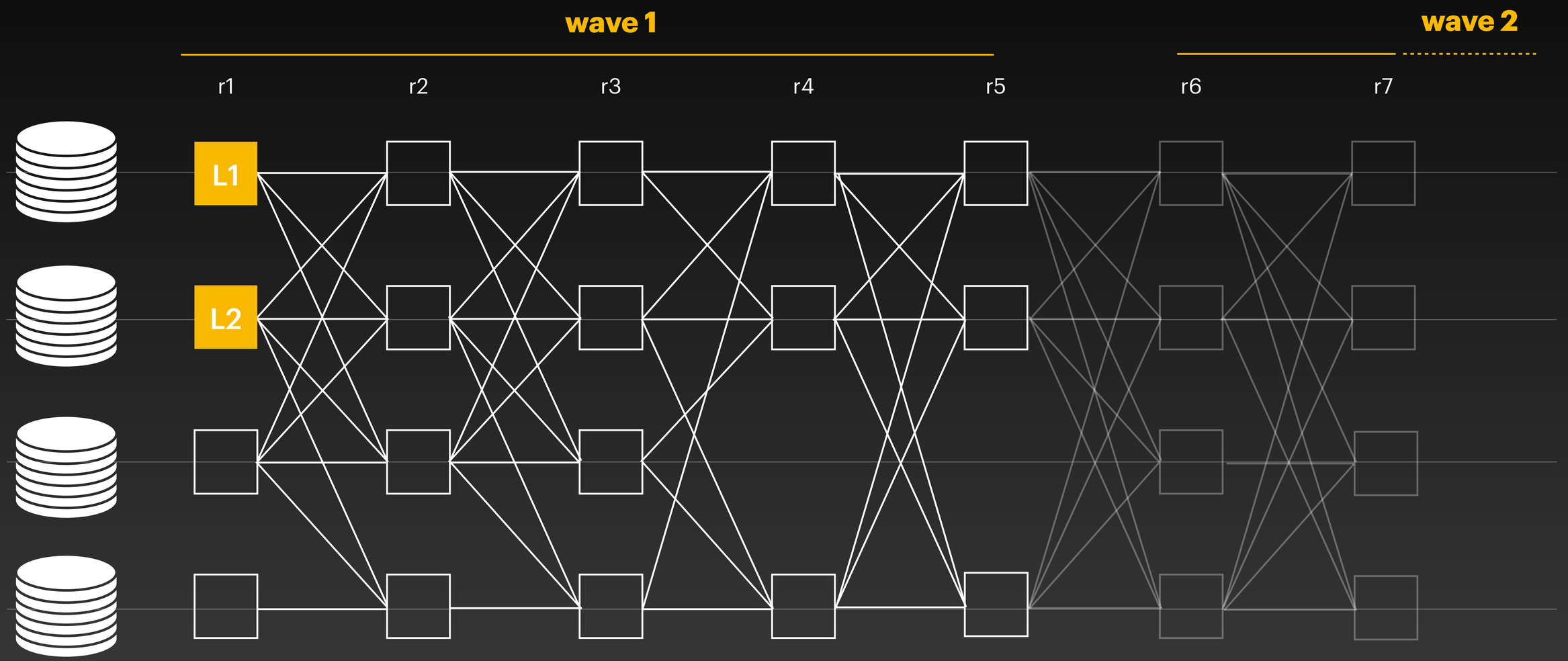
Mahi-Mahi Commit rule

How to reach consensus

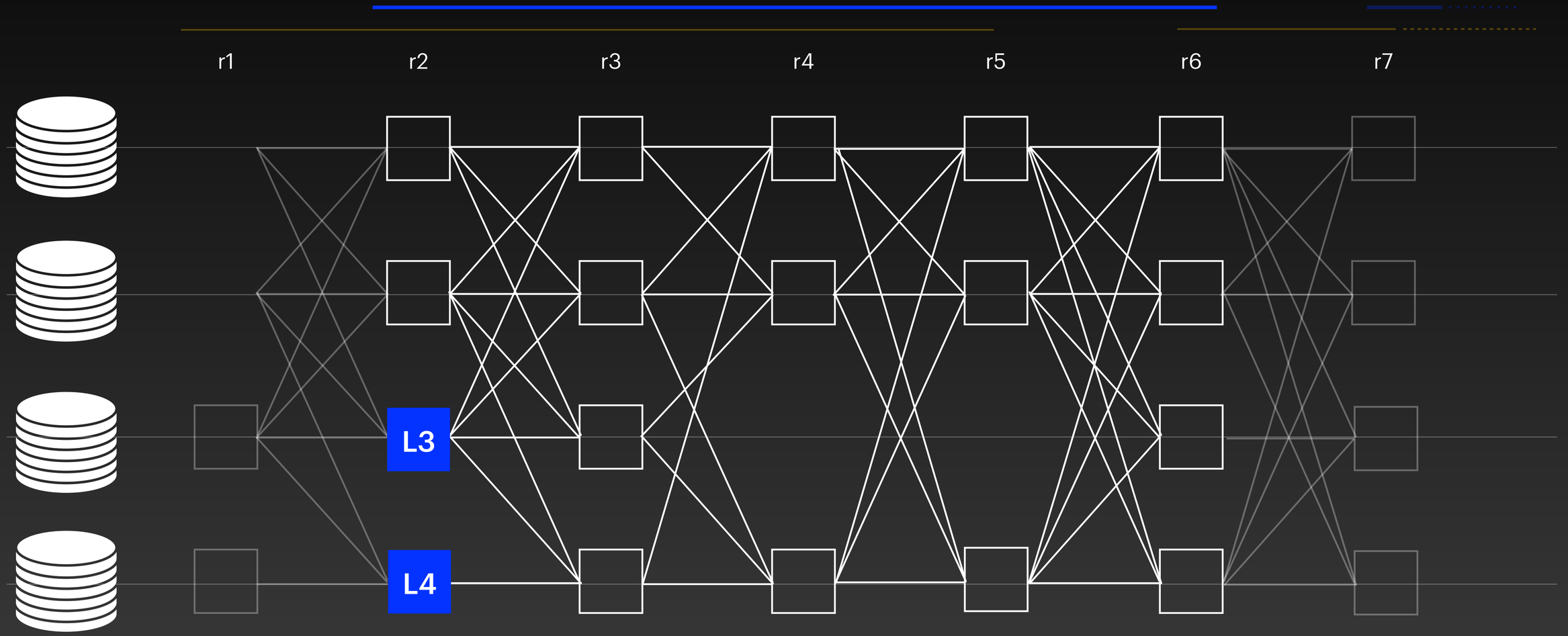
Mahi-Mahi: Outline of a wave



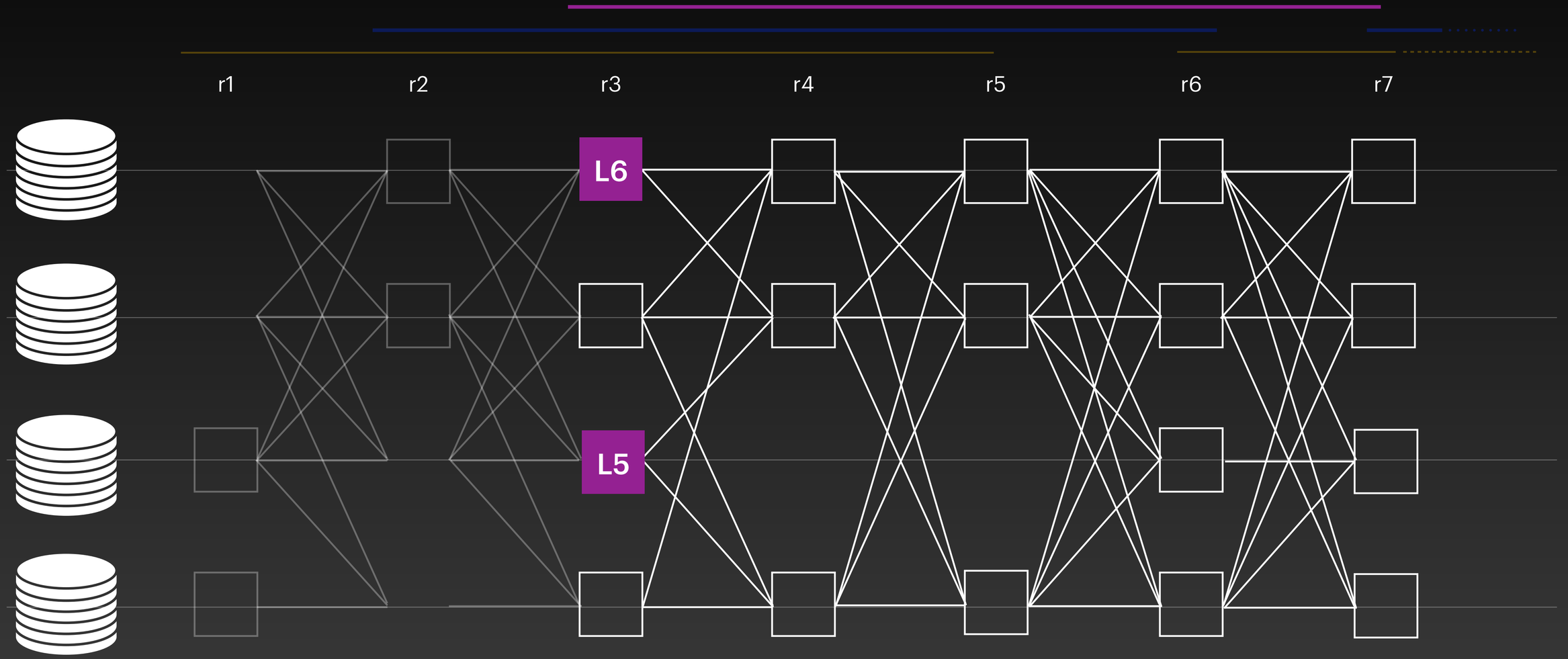
Pipelining in Mahi-Mahi



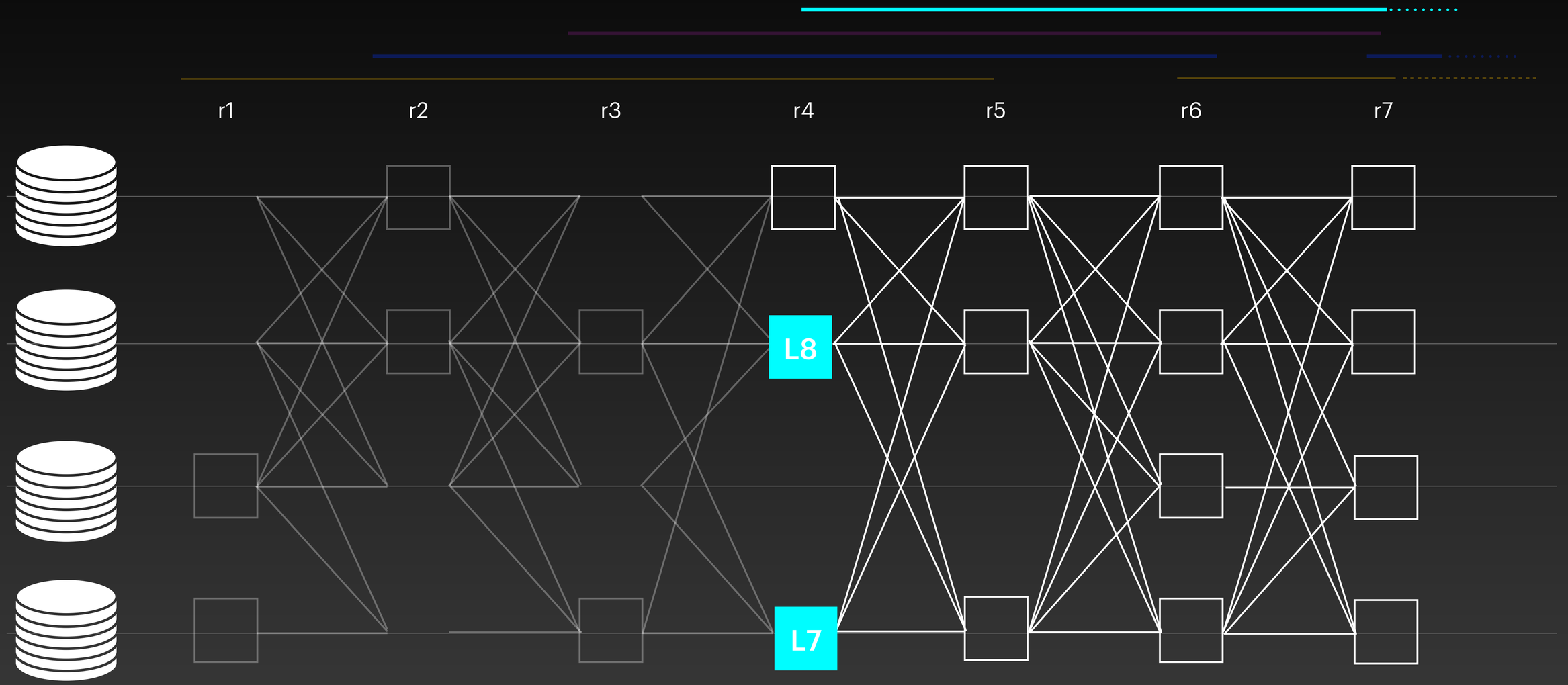
Pipelining in Mahi-Mahi



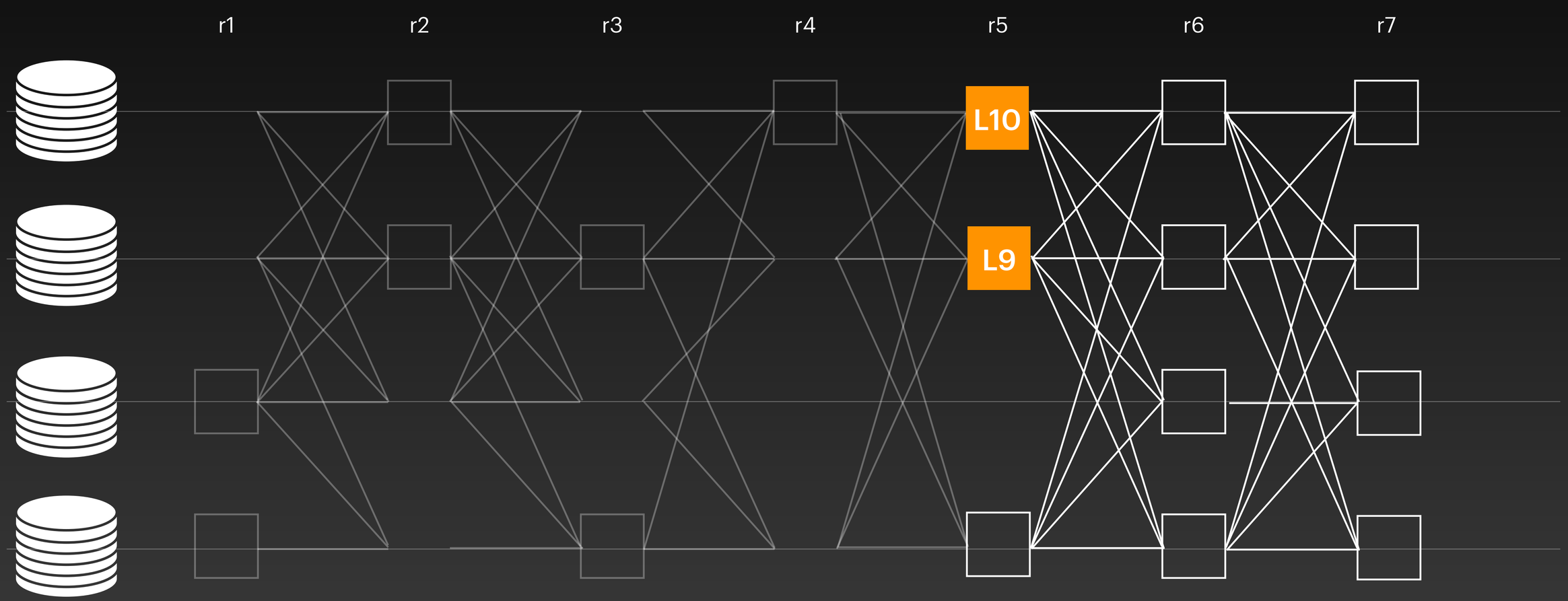
Pipelining in Mahi-Mahi



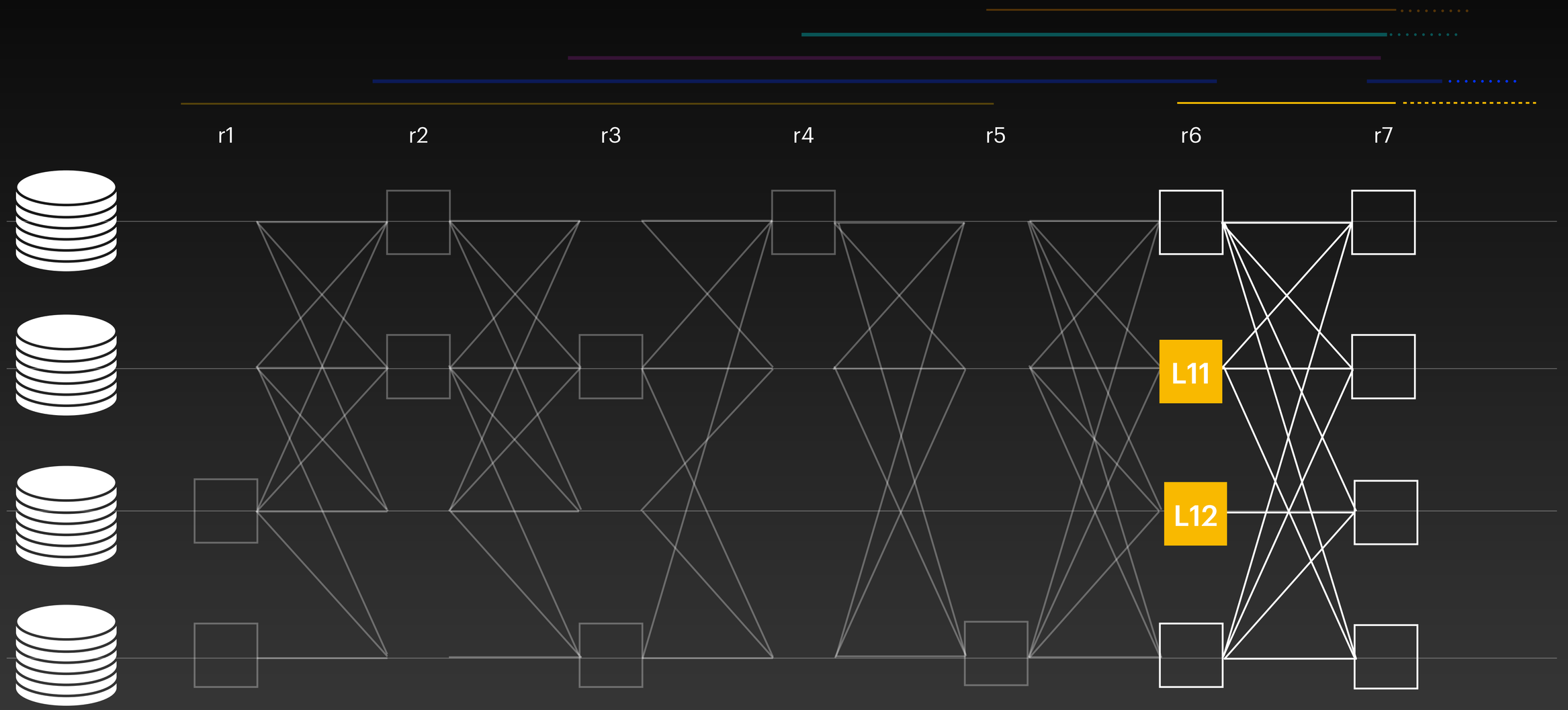
Pipelining in Mahi-Mahi



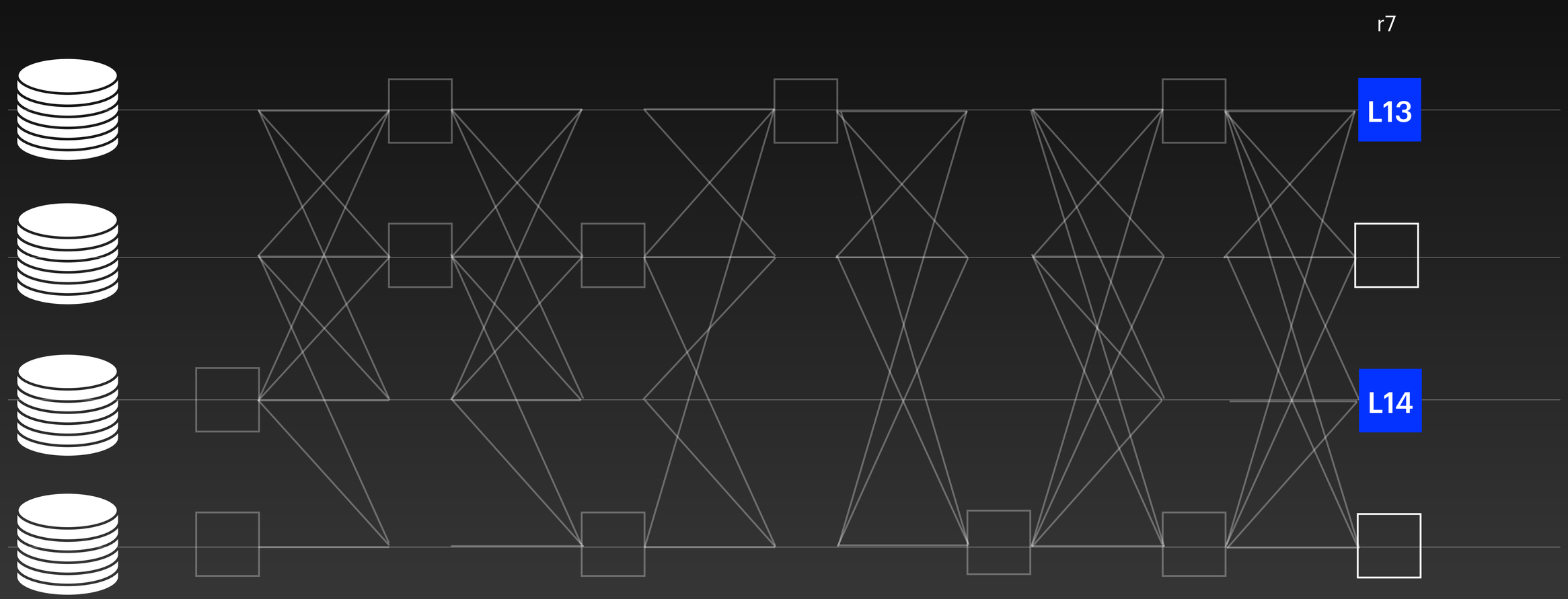
Pipelining in Mahi-Mahi



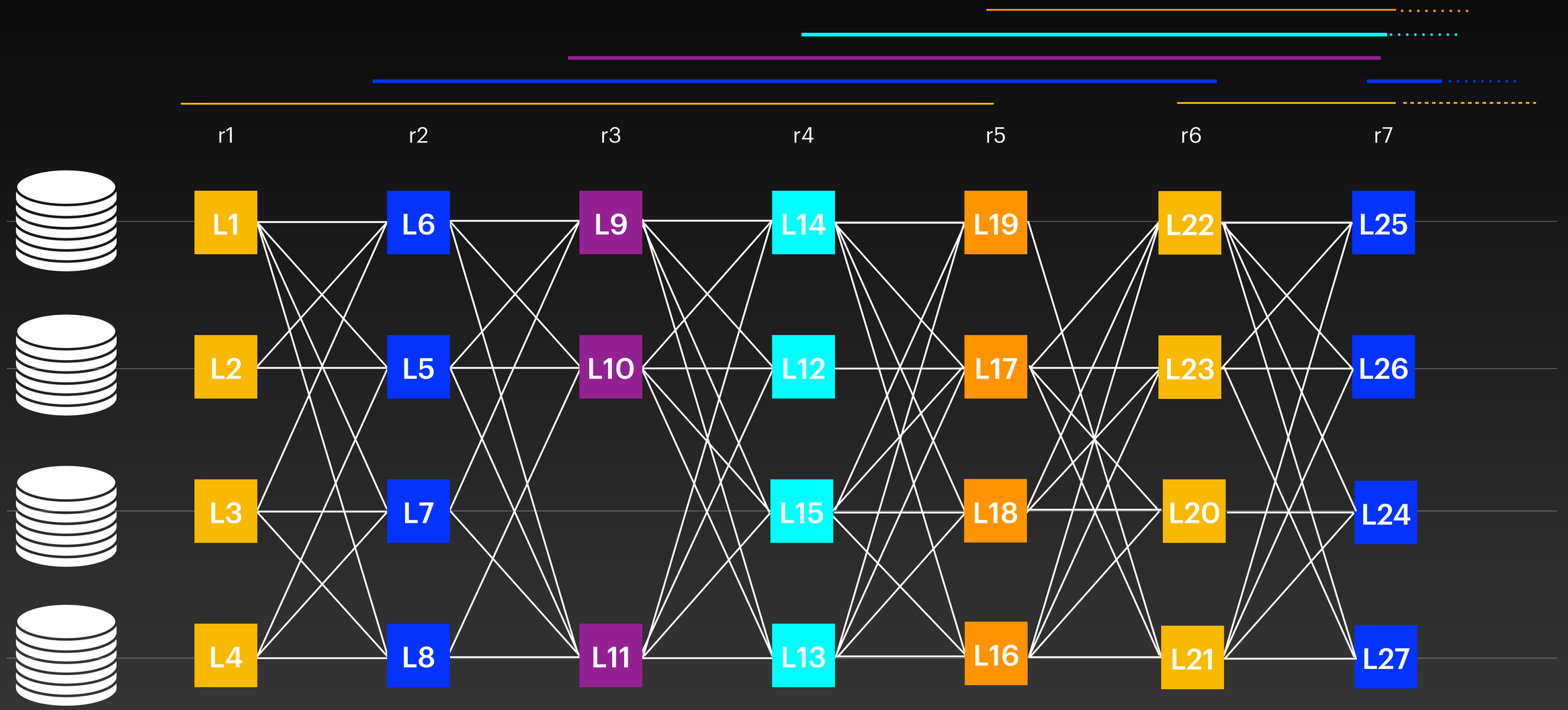
Pipelining in Mahi-Mahi



Pipelining in Mahi-Mahi

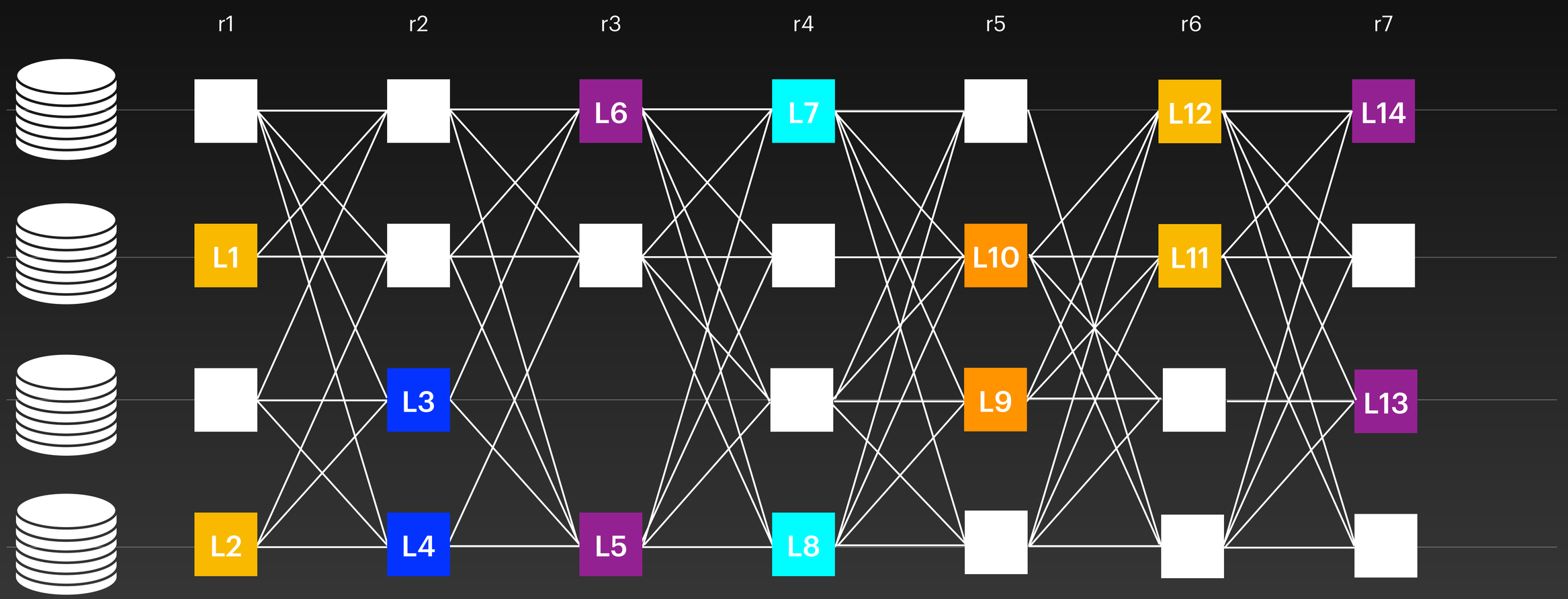


Theoretically, every validator can be a leader



Practical Implementation of Mahi-Mahi

2 leaders per round



Direct Decision Rule

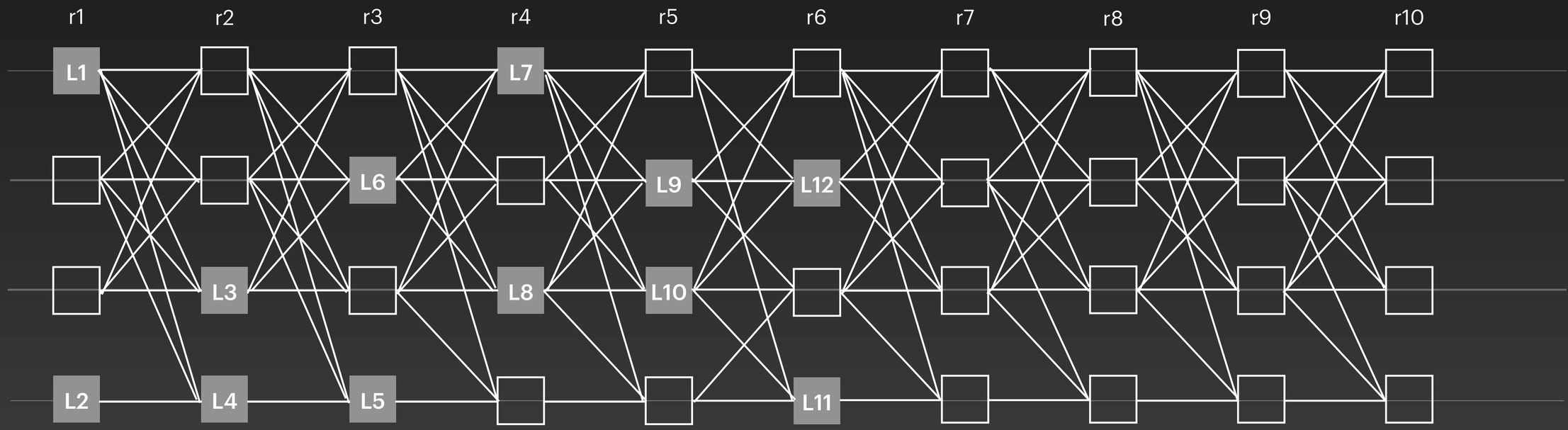
On each leader starting from highest round:

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

Direct Decision Rule

On each leader starting from highest round:

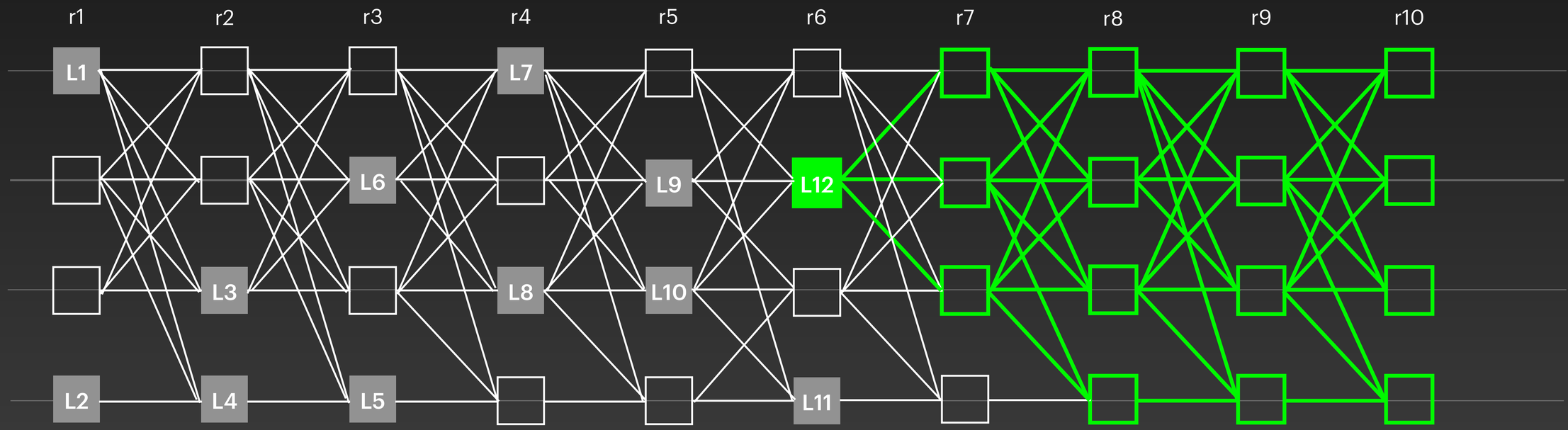
- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certificates
- **Undecided** otherwise



Direct Decision Rule

On each leader starting from highest round:

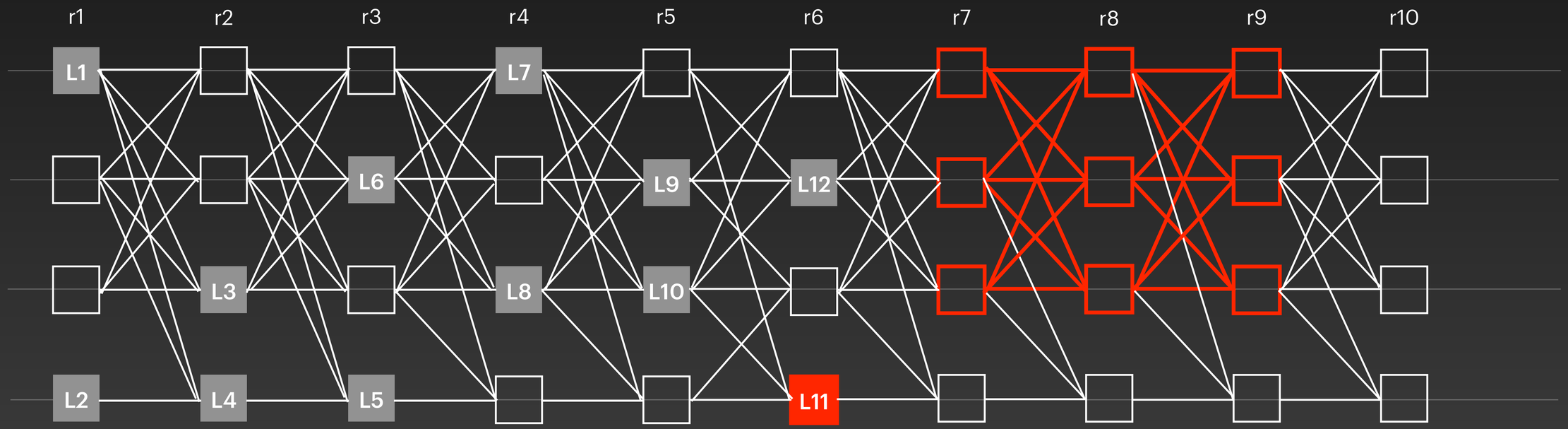
- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certificates
- **Undecided** otherwise



Direct Decision Rule

On each leader starting from highest round:

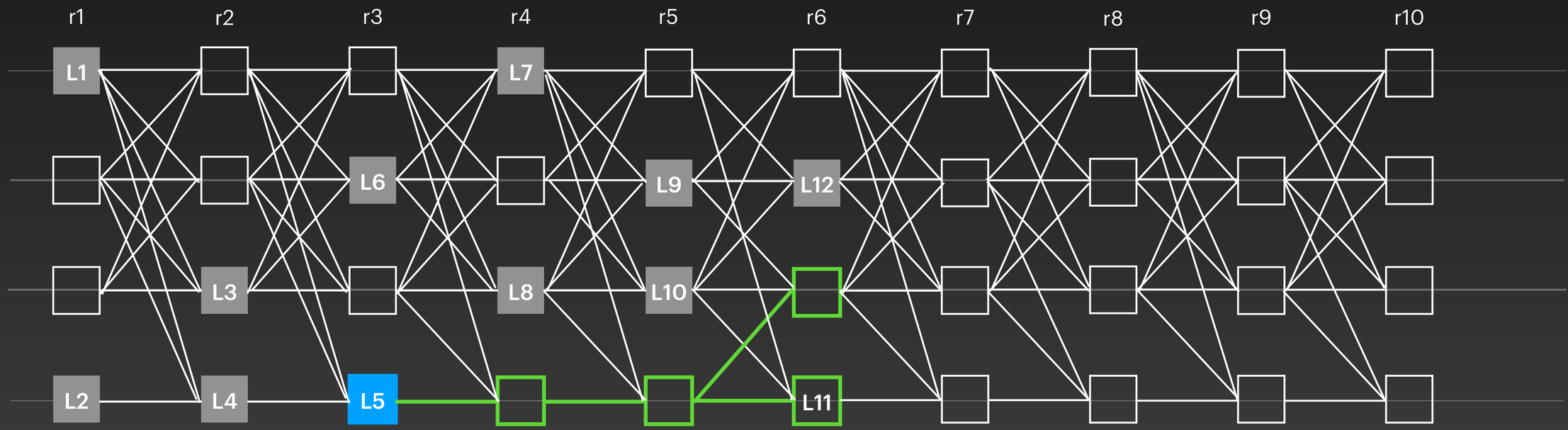
- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certificates
- **Undecided** otherwise



Direct Decision Rule

On each leader starting from highest round:

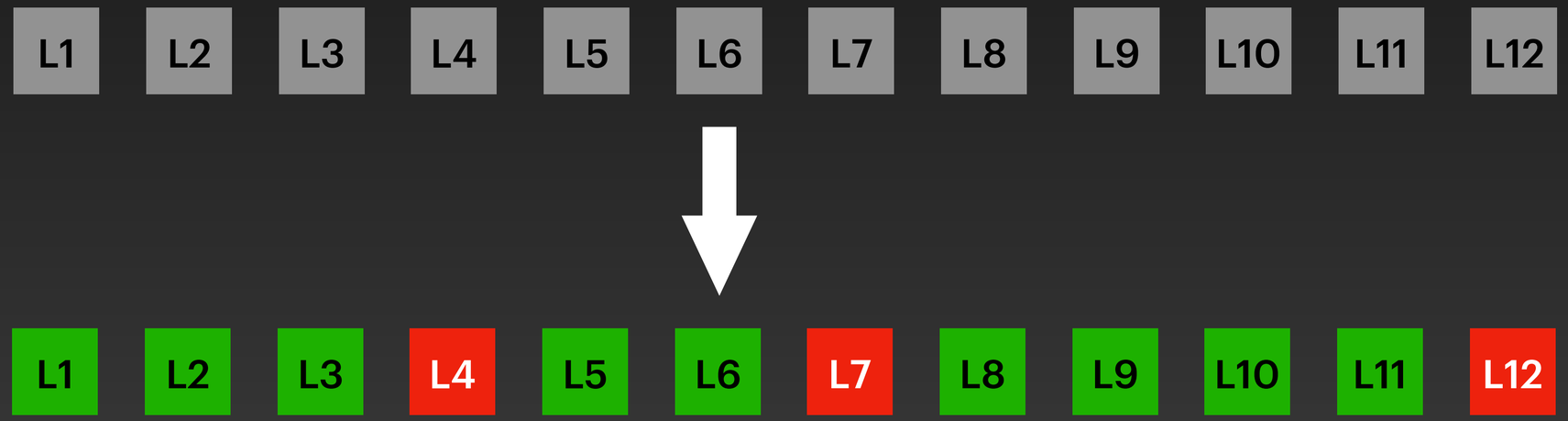
- **Skip** if 2f+1 blames
- **Commit** if 2f+1 certificates
- **Undecided** otherwise



End Goal

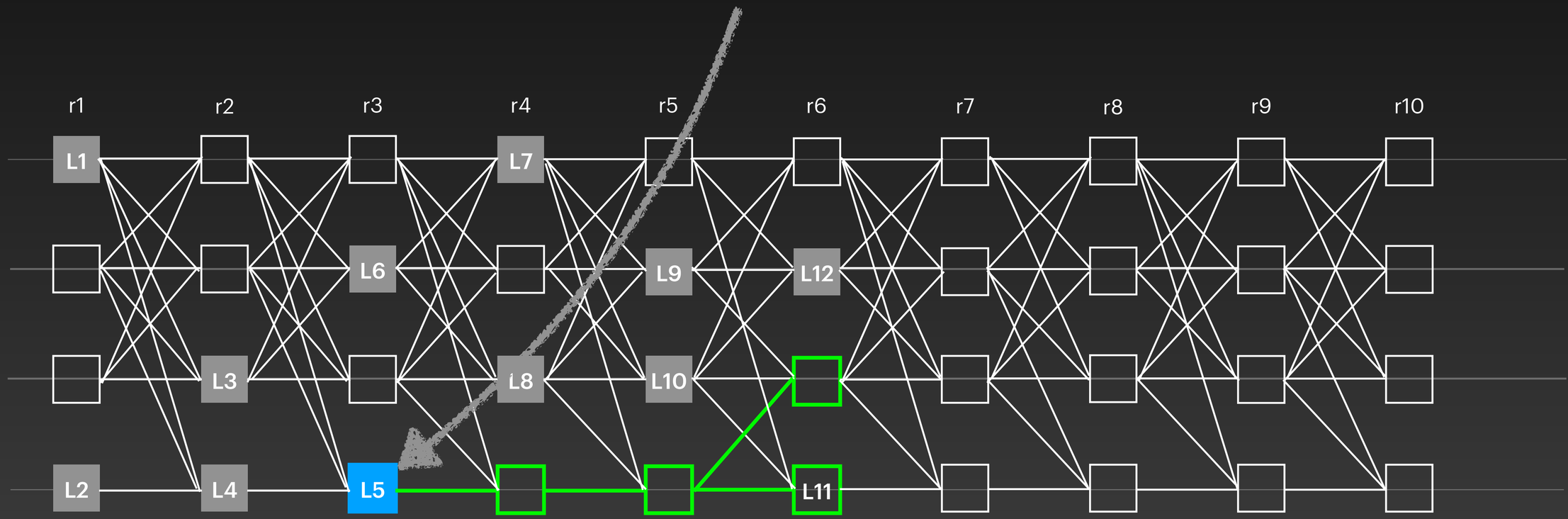
Output a sequence of leaders that are:

- Marked as **Commit** or **Skip** (not **Undecided**)
- Common to all nodes (despite Byzantine faults)



What to do about undecided blocks?

- **Undecided** if there's less than 2f+1 blames or less than 2f+1 certificates

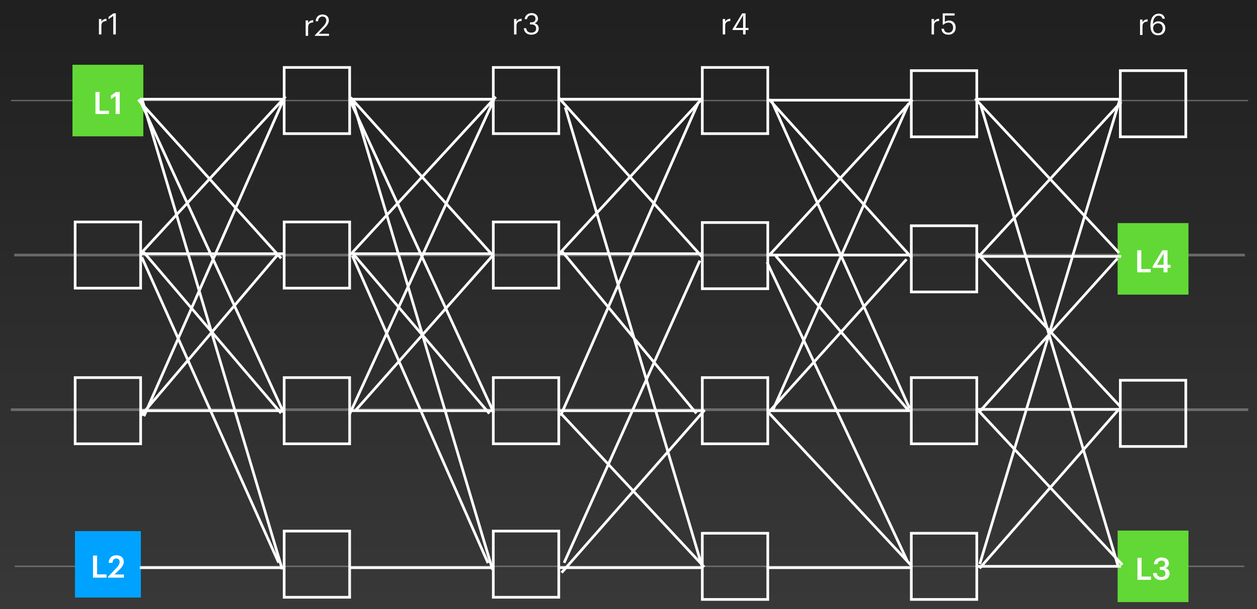


Indirect Decision Rule

Indirect Decision Rule

1. Find Anchor

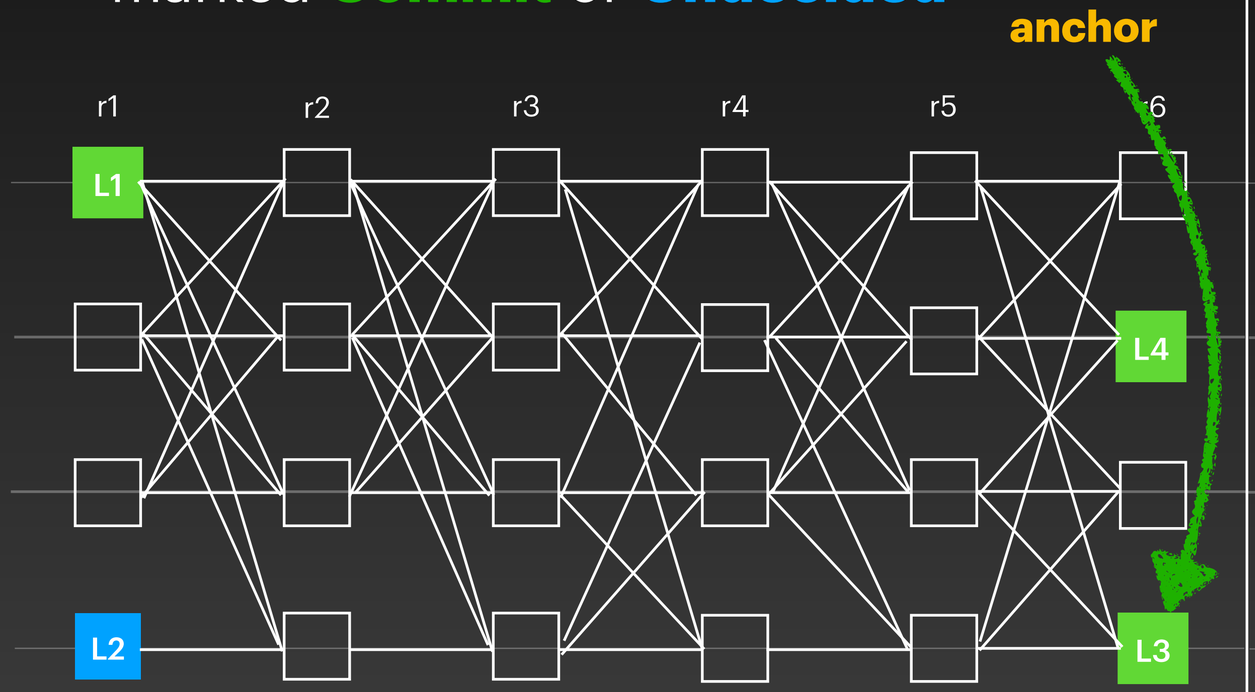
- First block with round > r+4 that is **Commit** or **Undecided**



Indirect Decision Rule

1. Find Anchor

- First block with round > r+4 that is marked **Commit** or **Undecided**

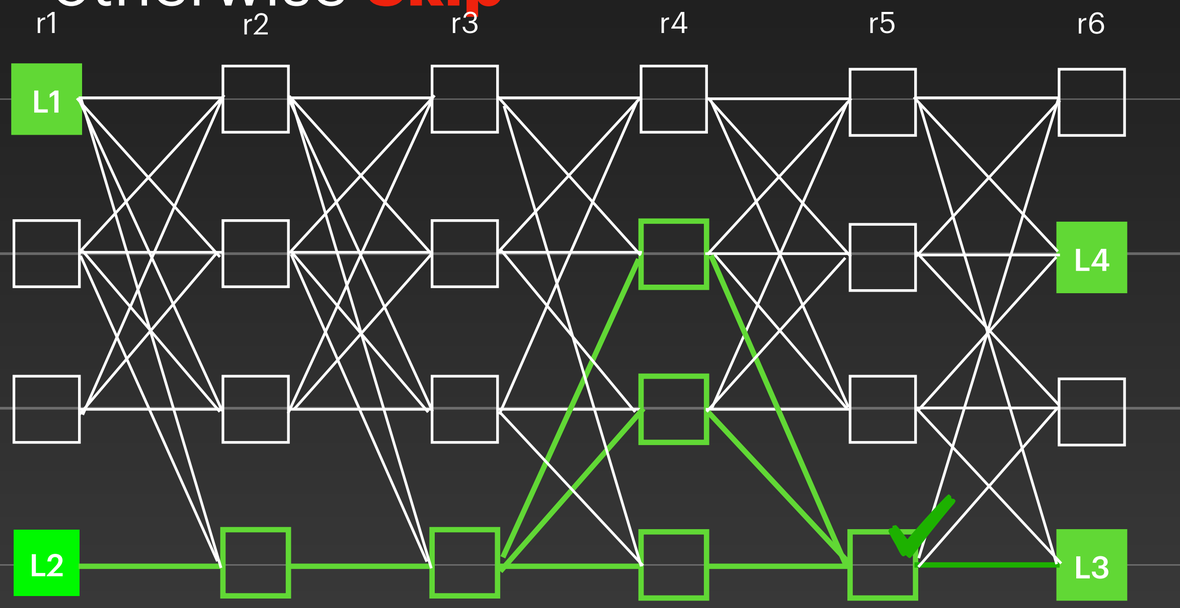


2. Certified link

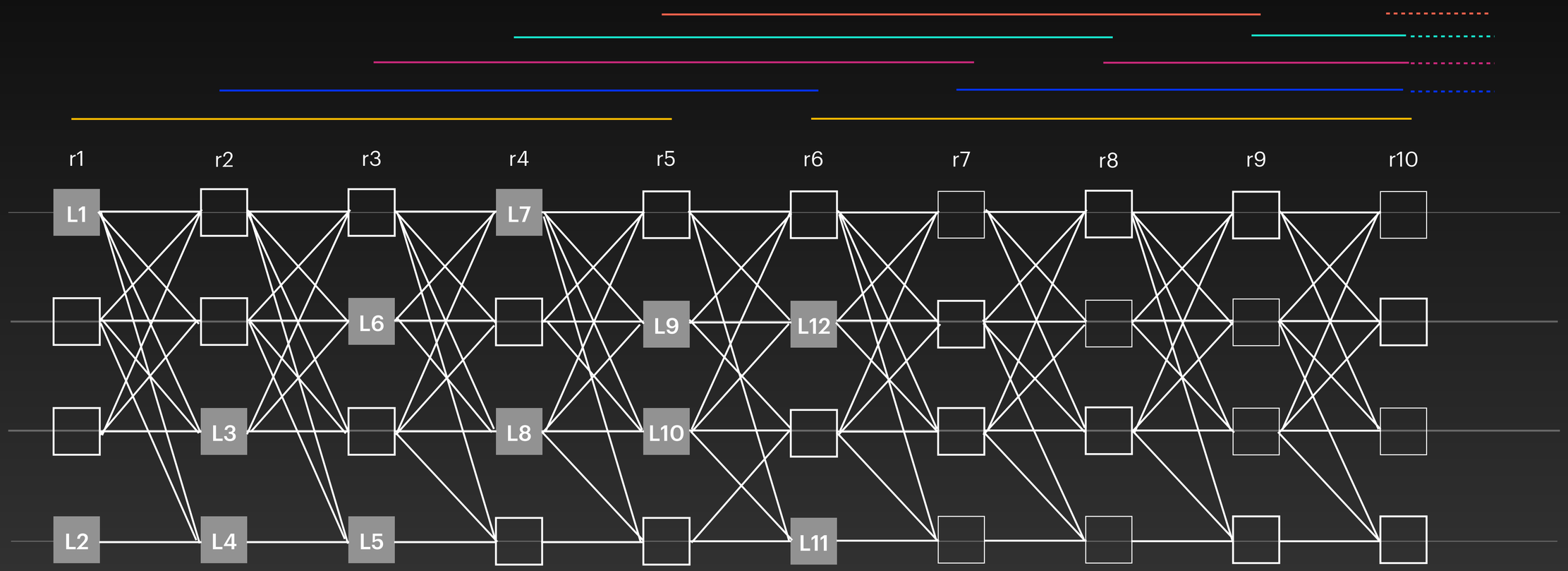
- **Commit** if

`B <-> certified link <-> A`

otherwise **Skip**

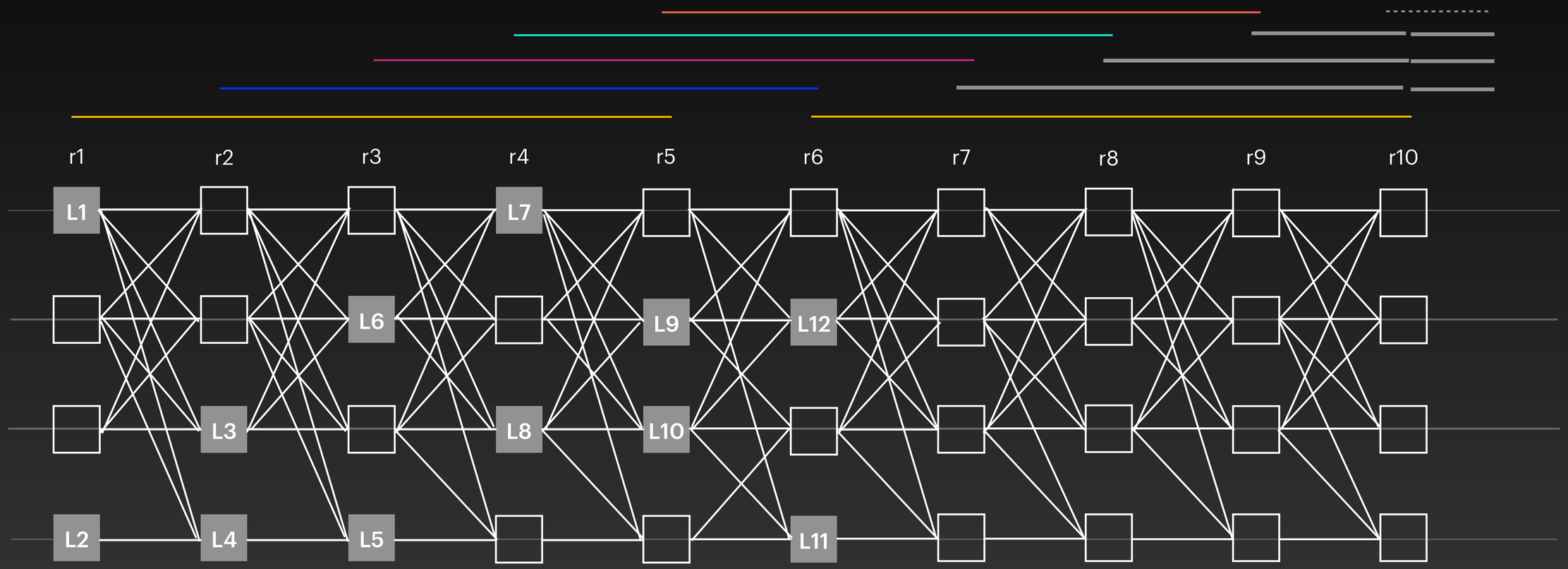


All Blocks Start As Undecided



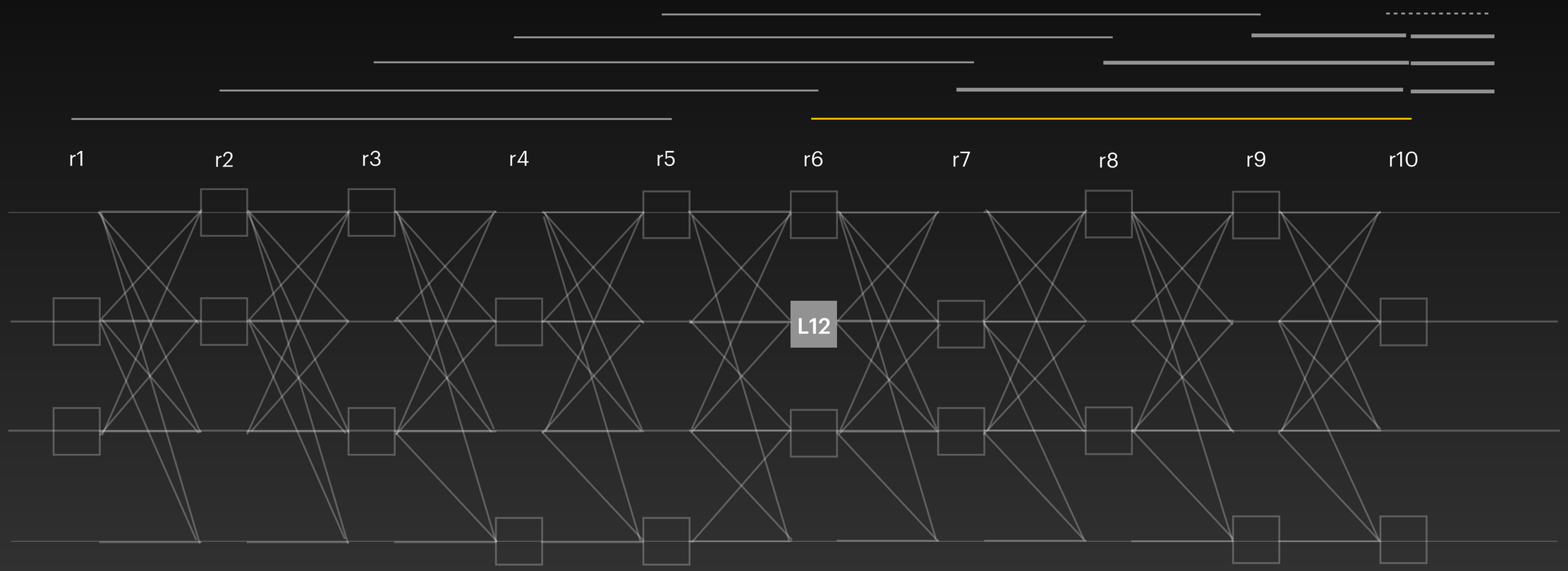
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Ignore Incomplete Waves



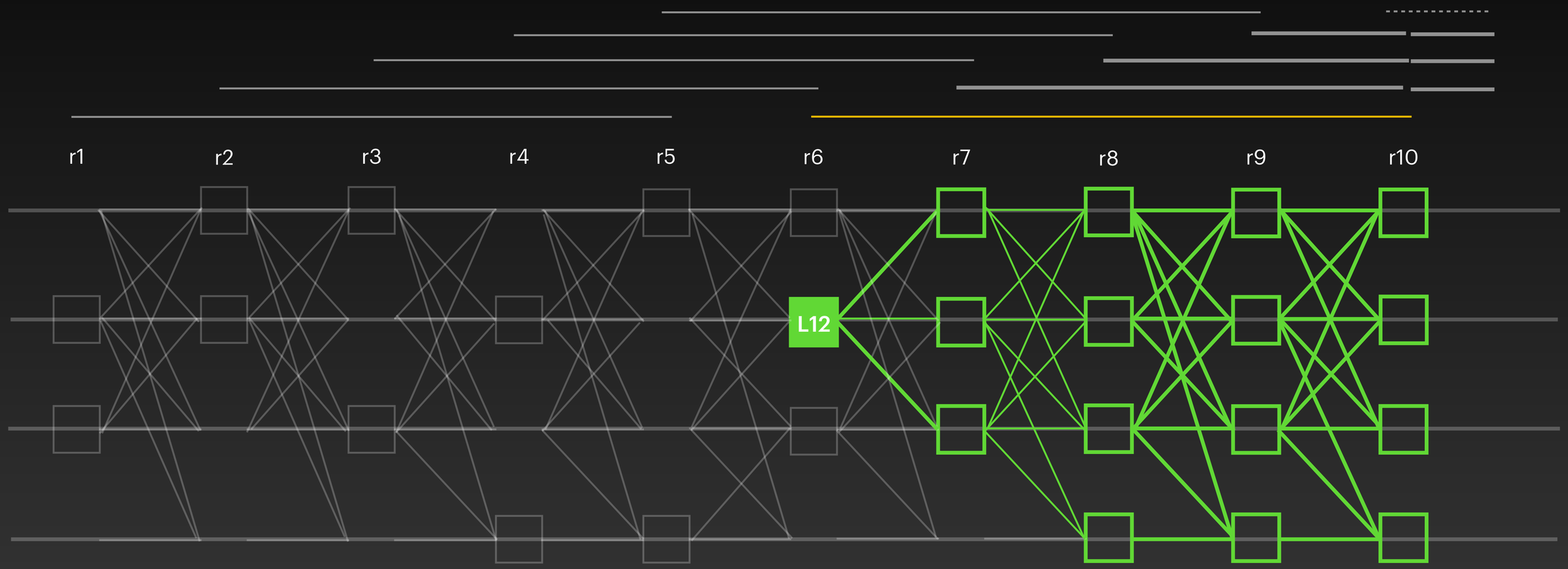
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



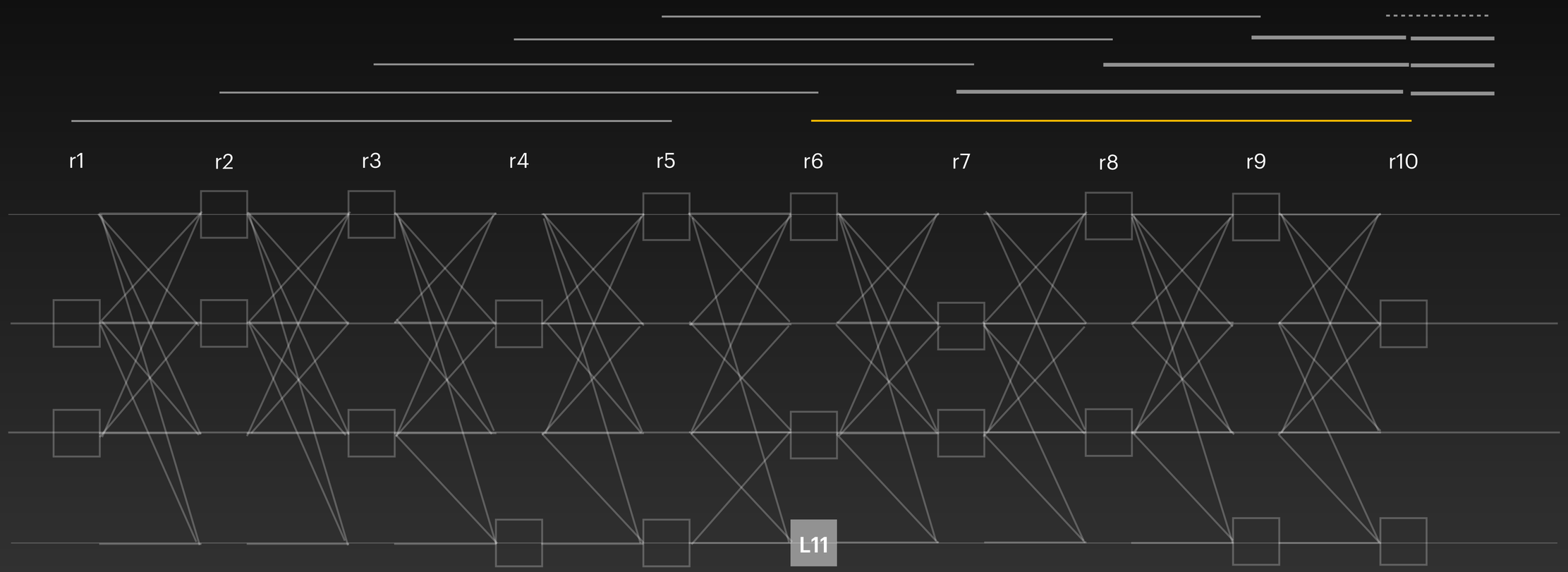
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



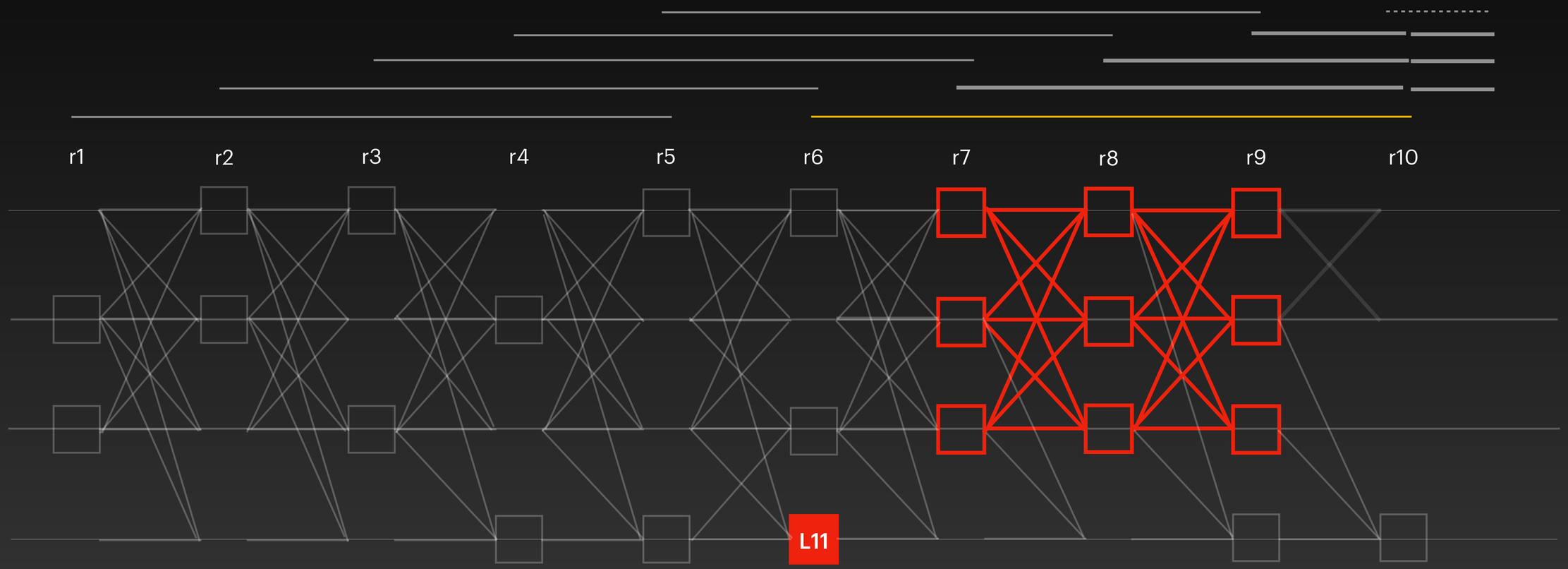
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



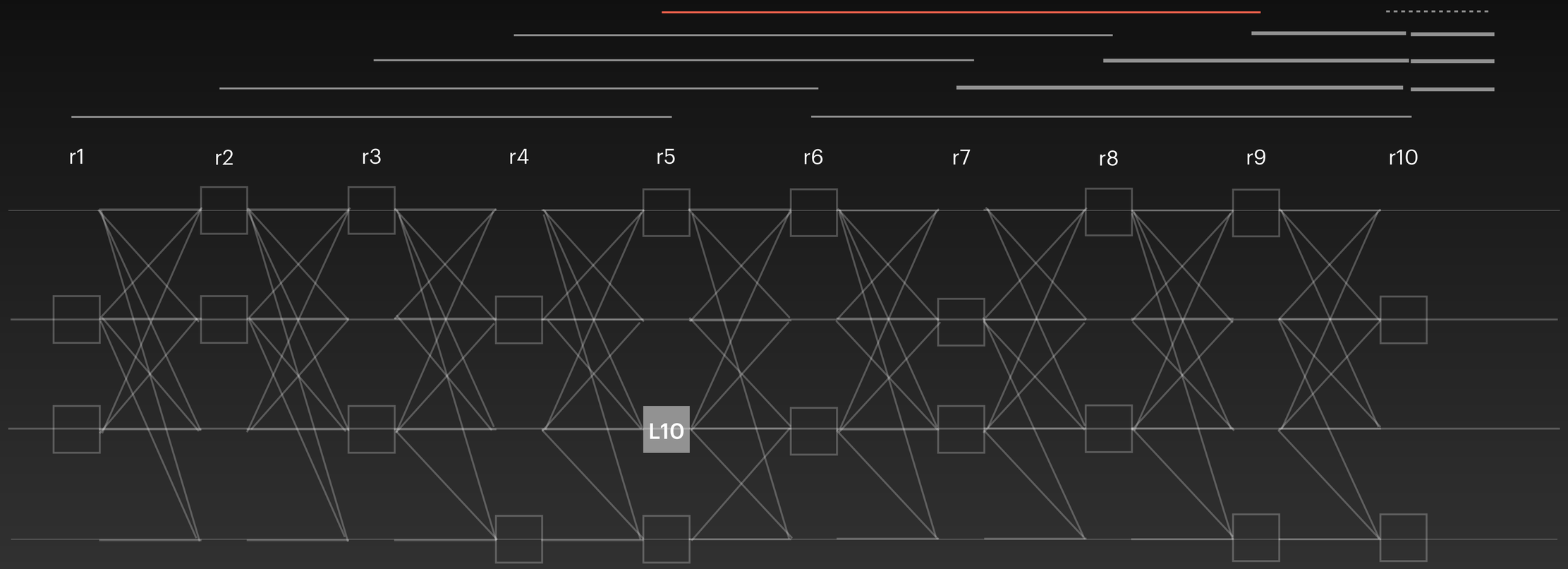
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



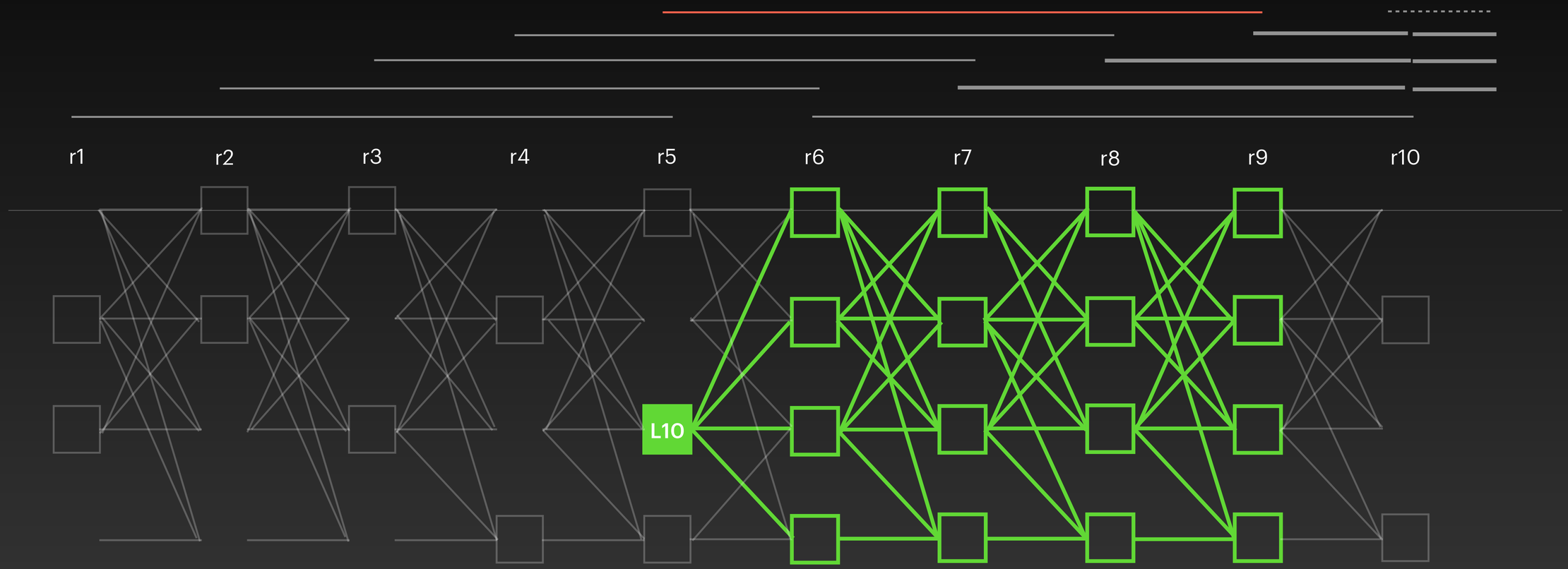
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



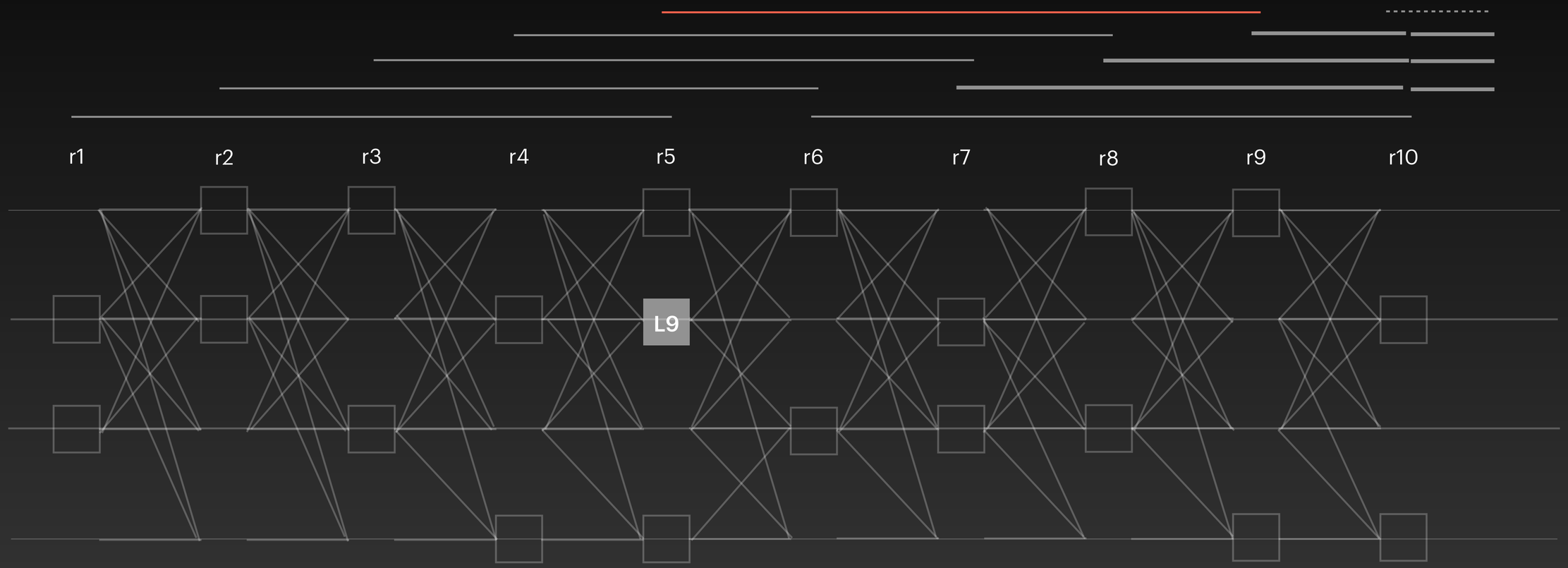
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



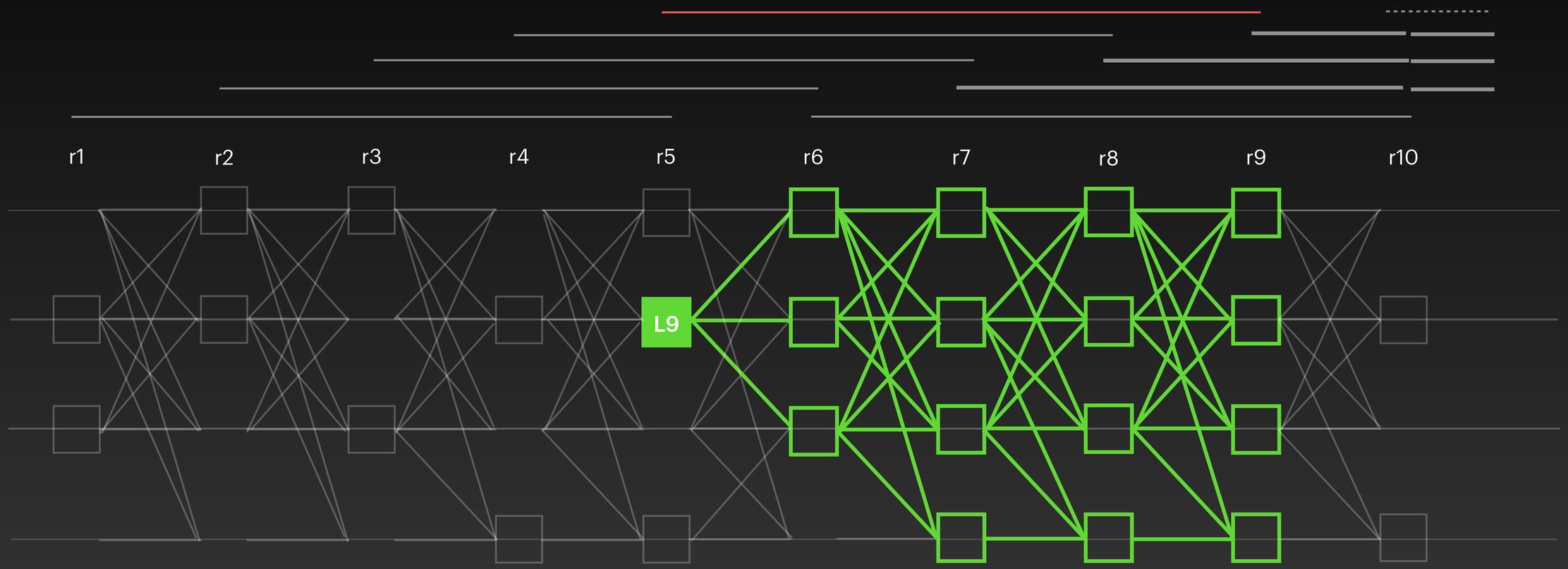
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



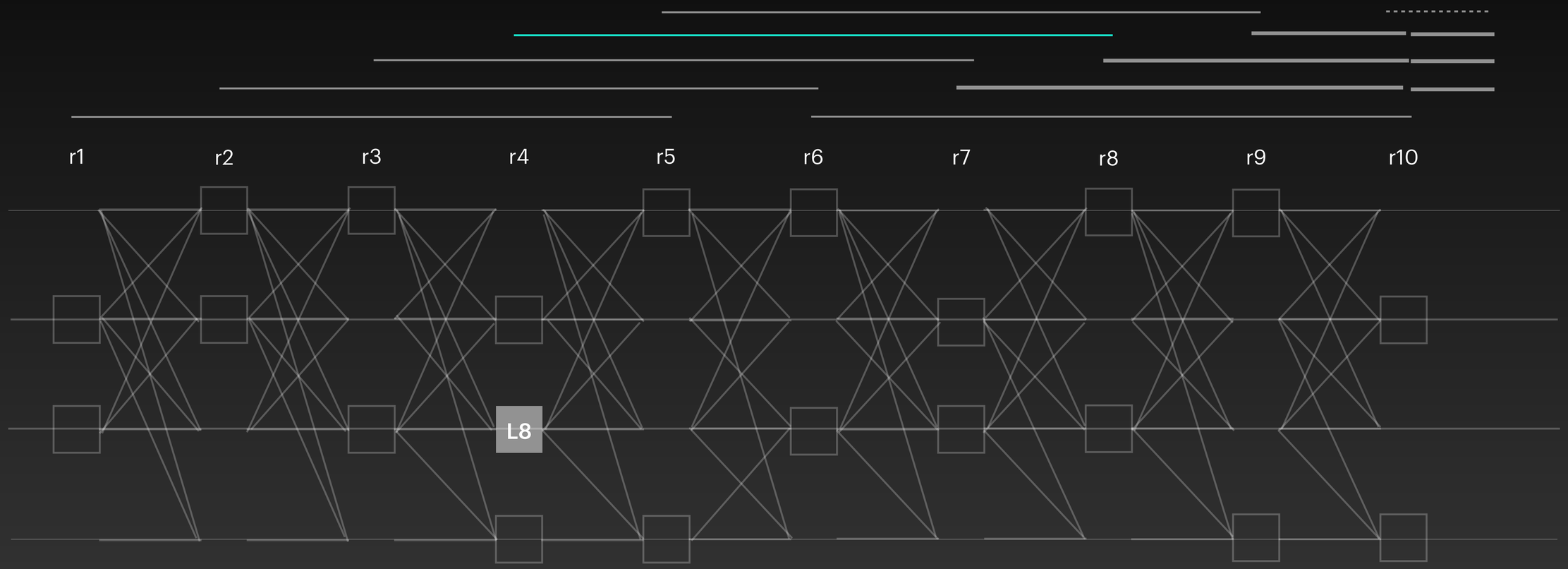
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



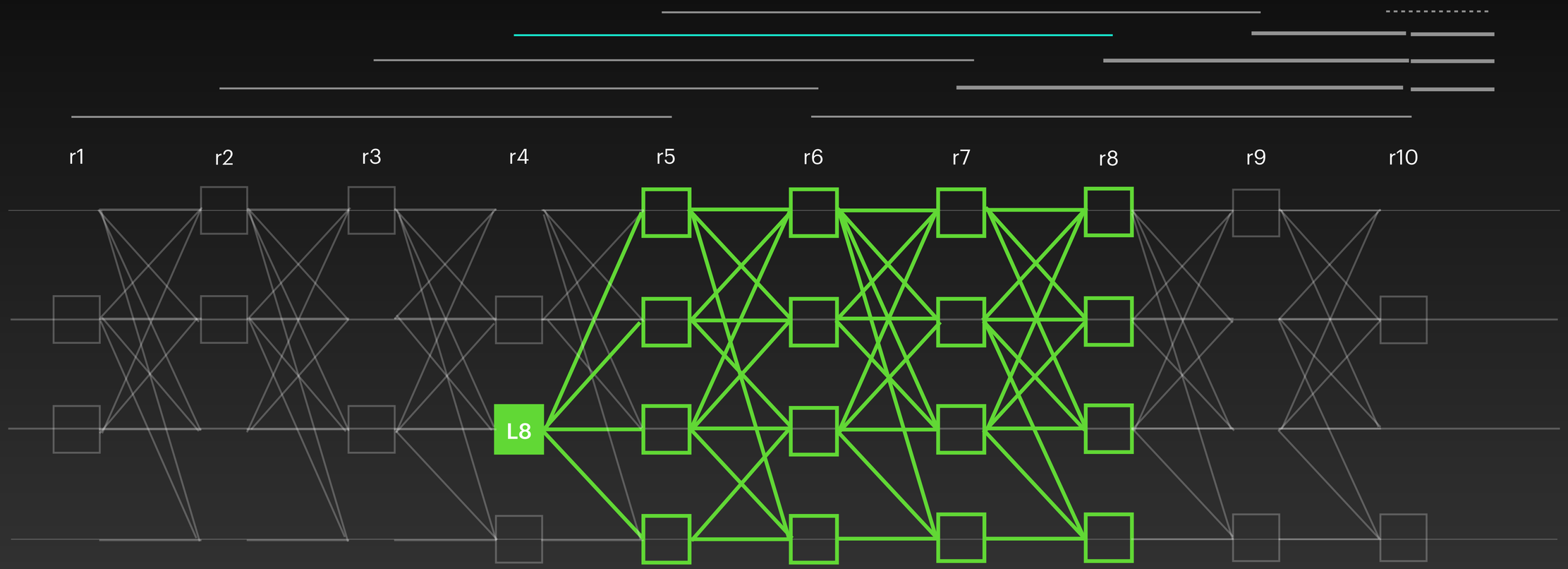
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



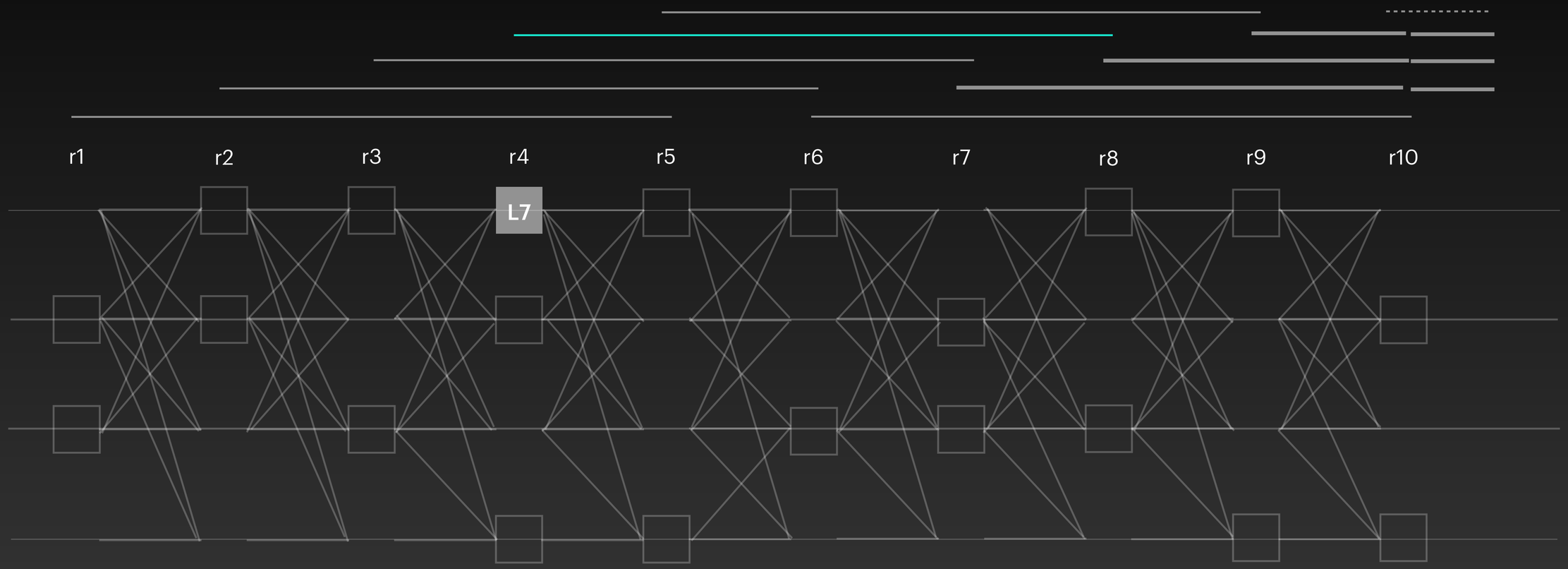
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



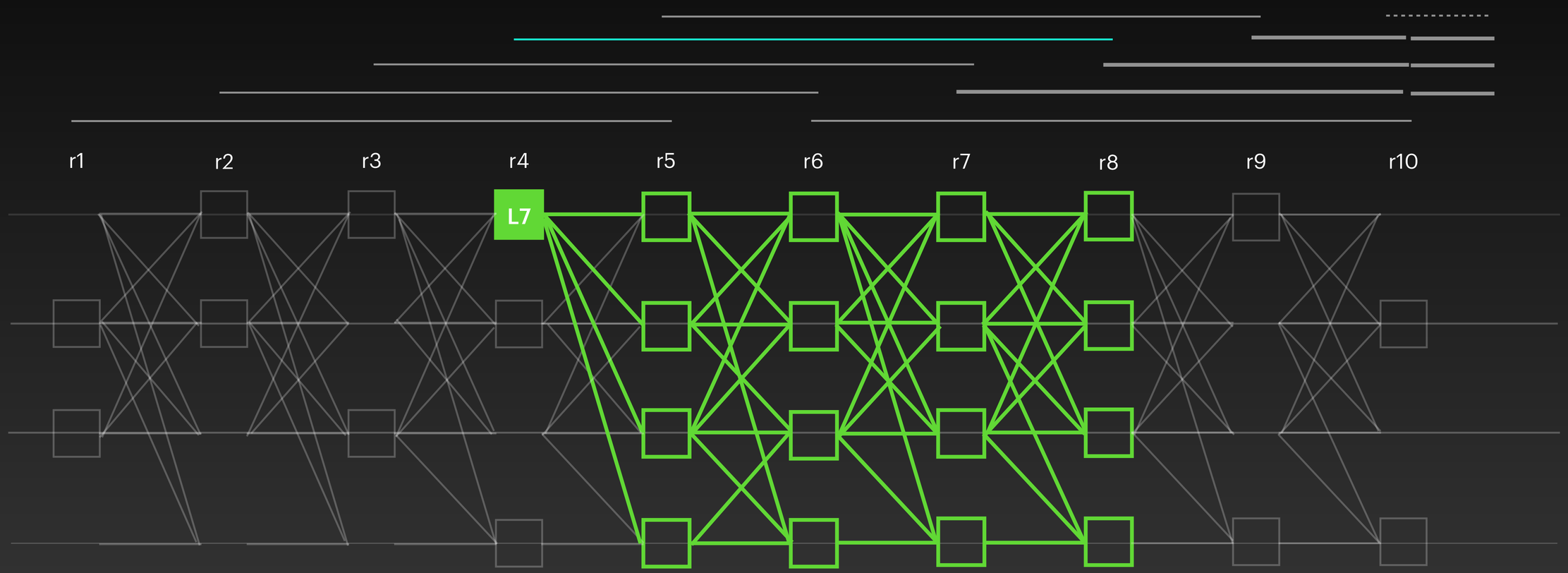
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



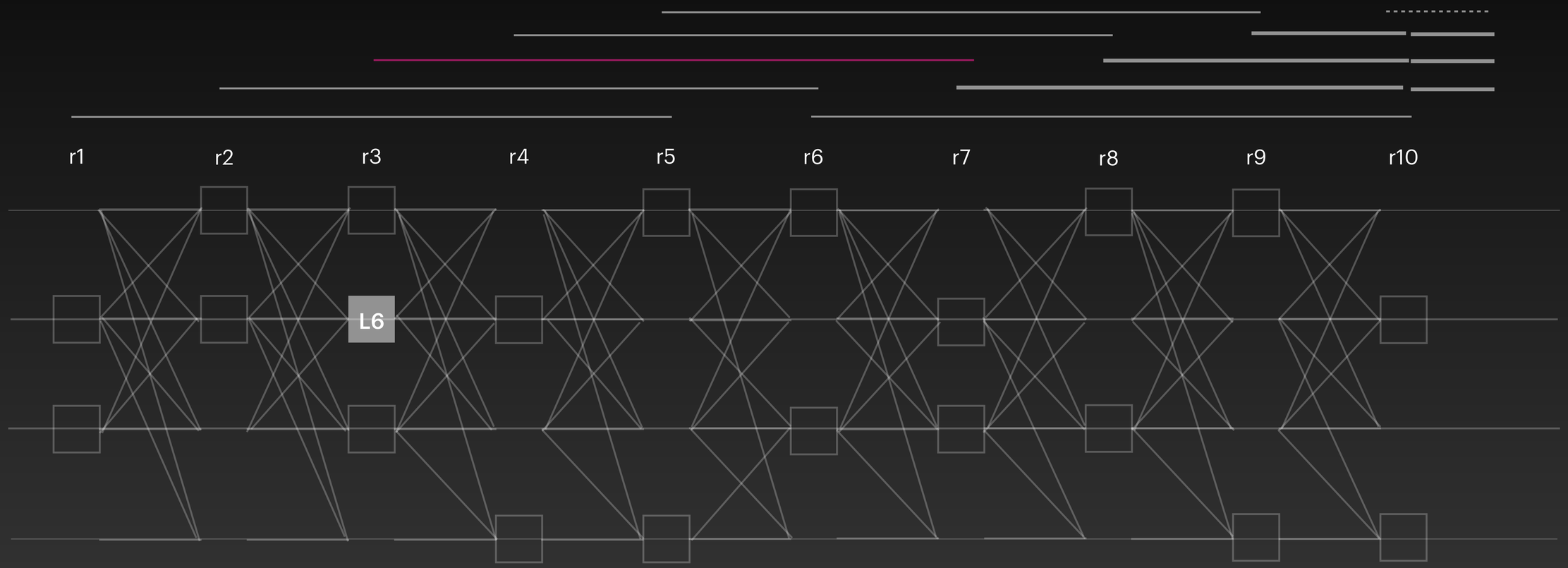
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



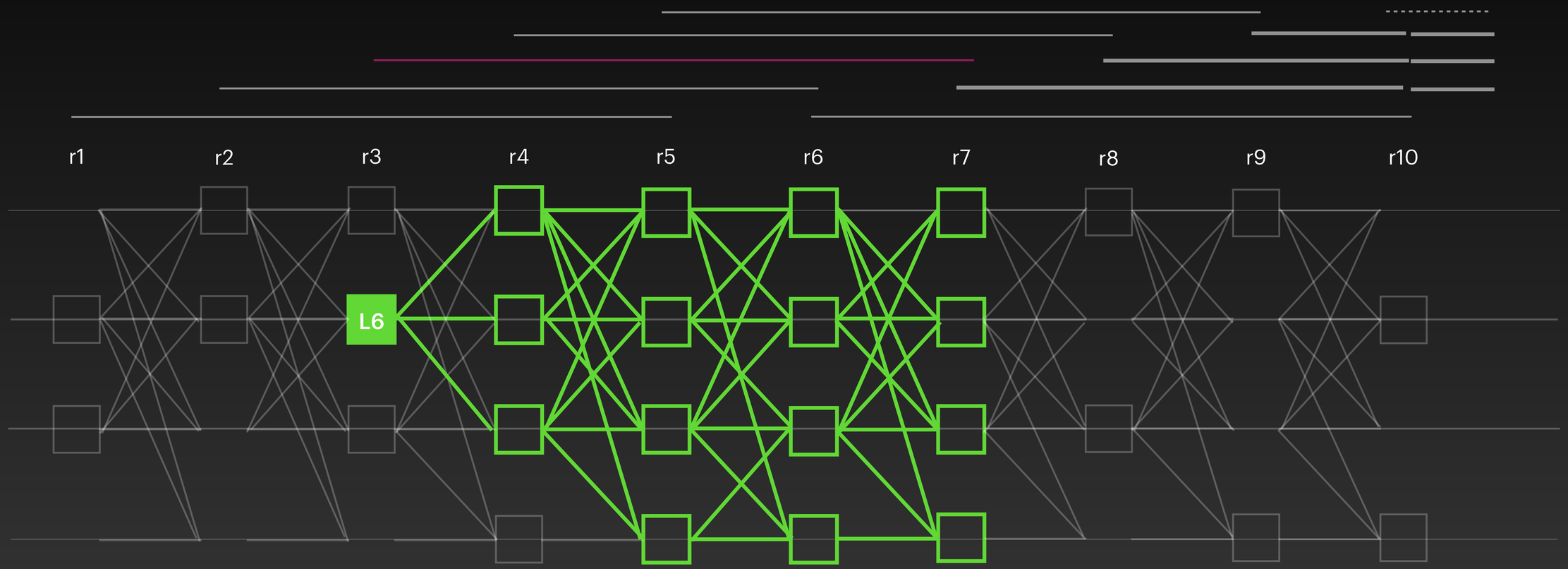
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



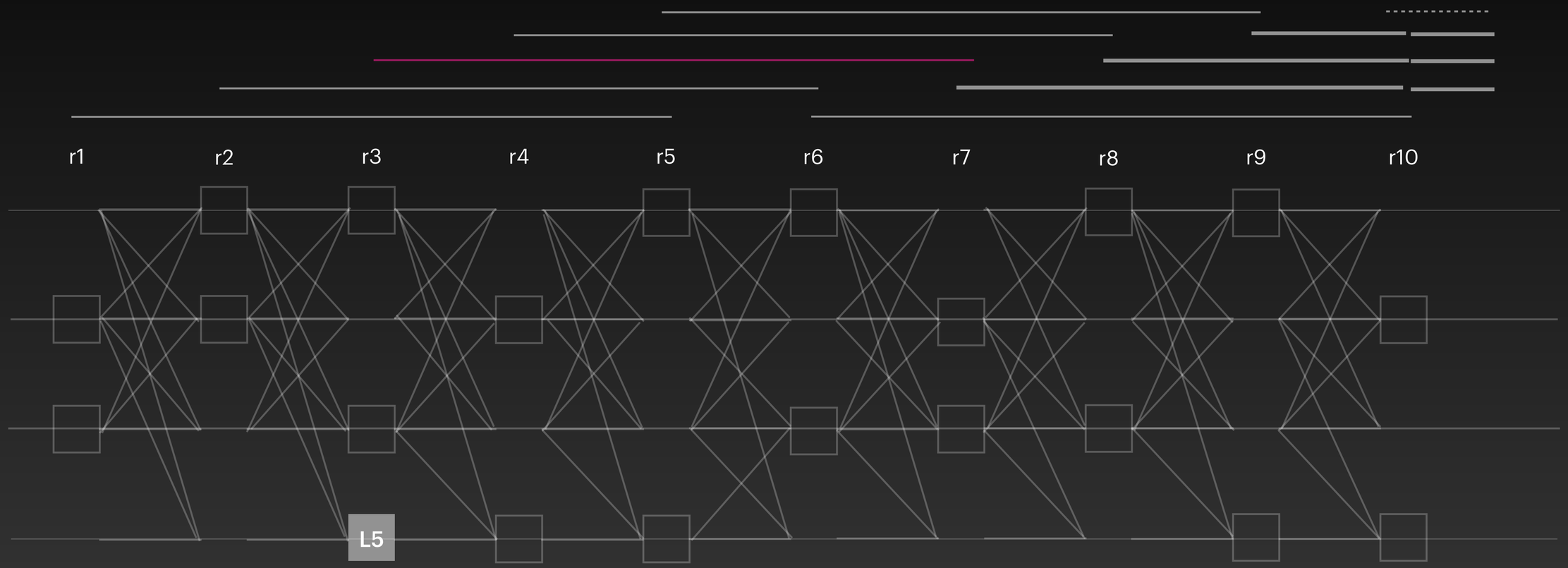
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



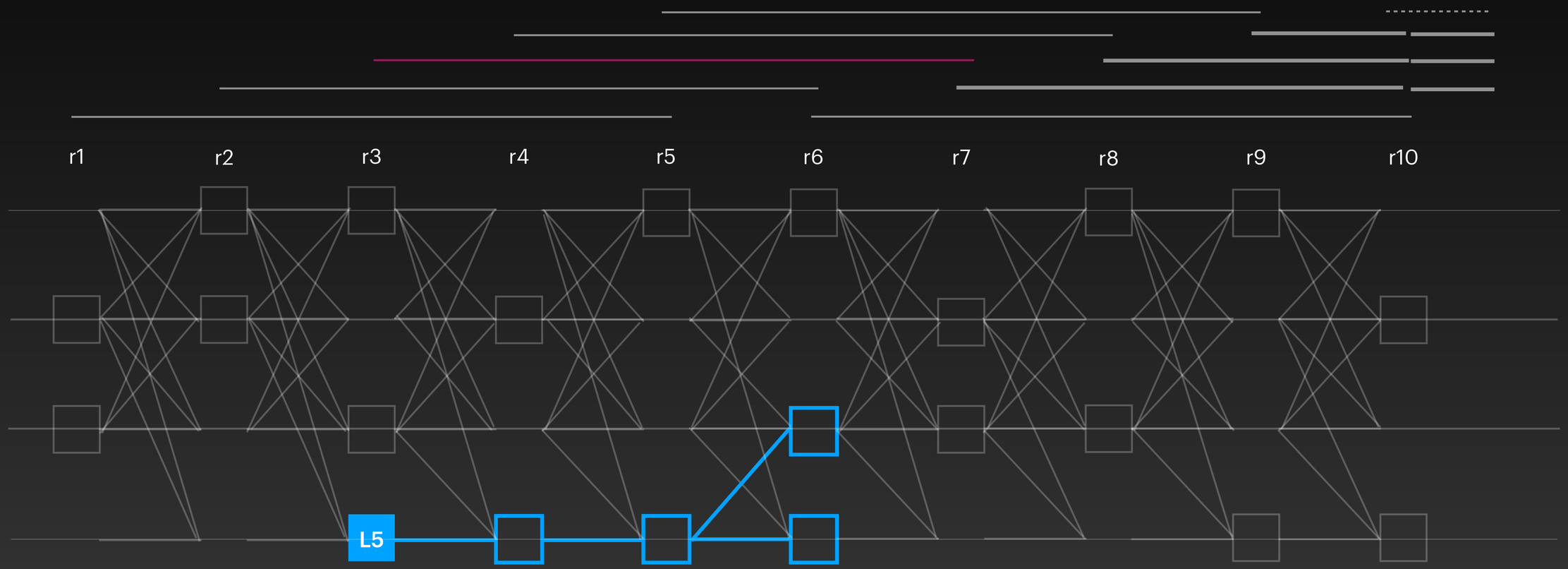
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



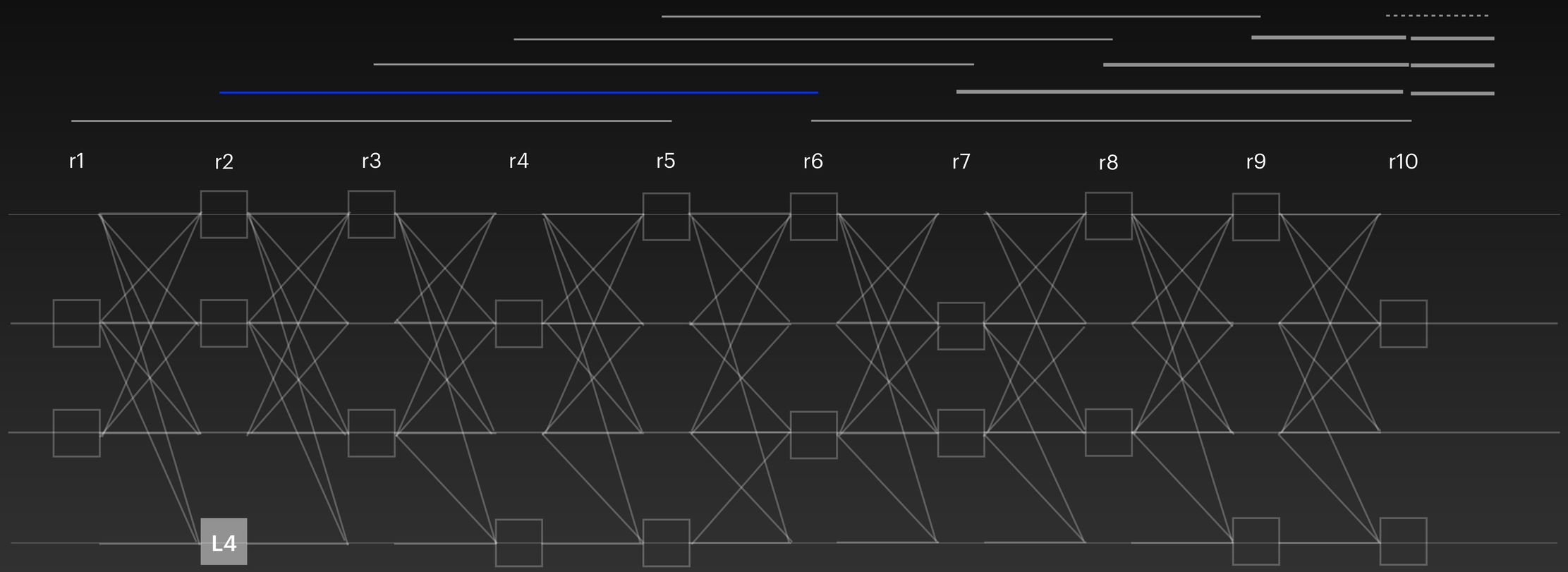
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



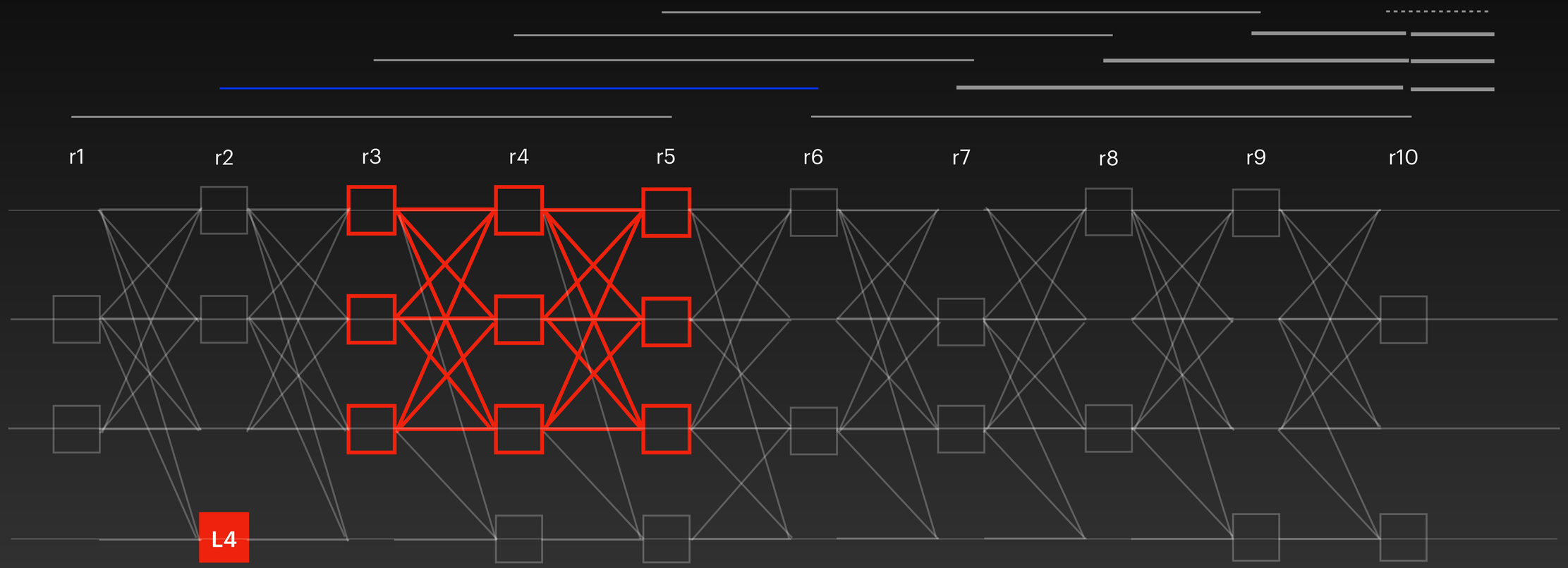
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



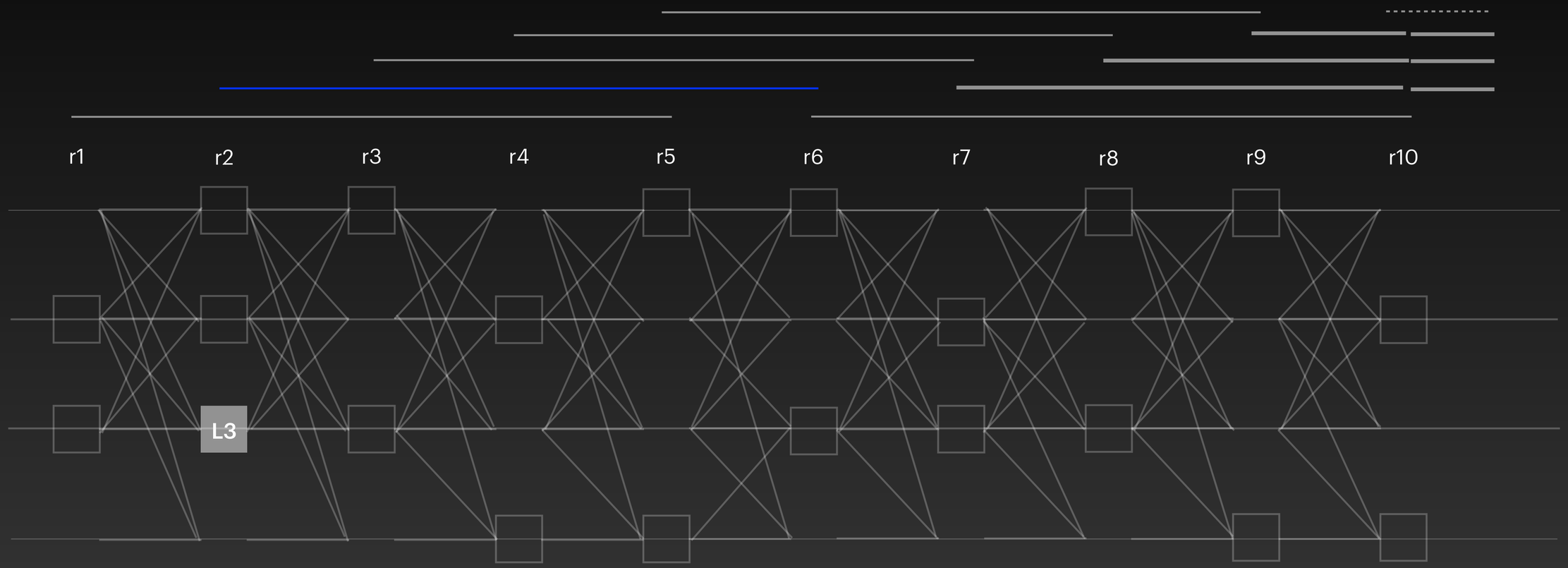
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



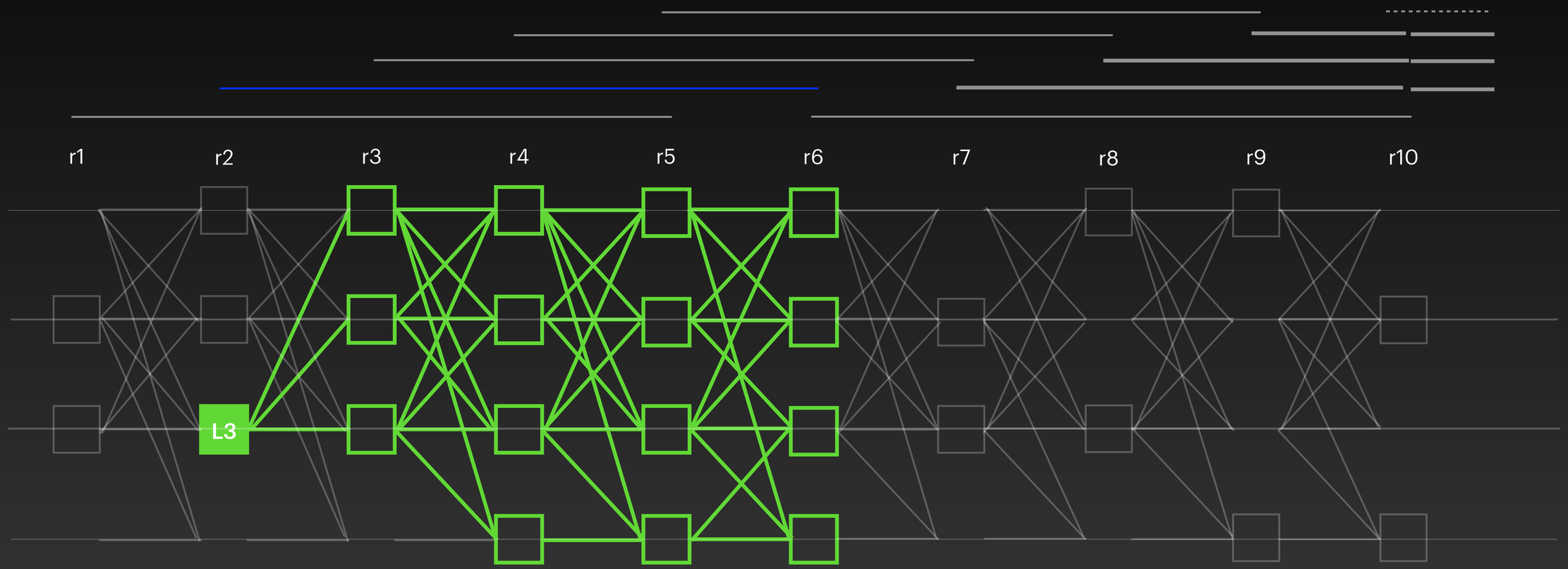
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



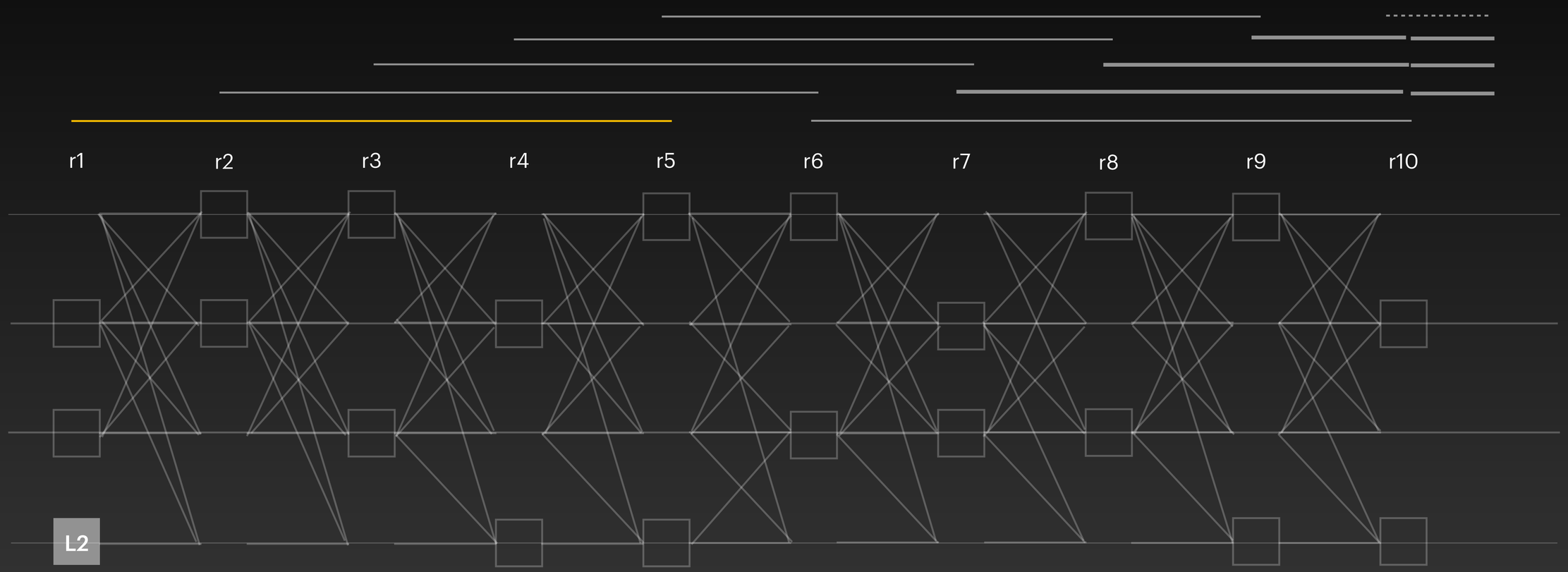
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



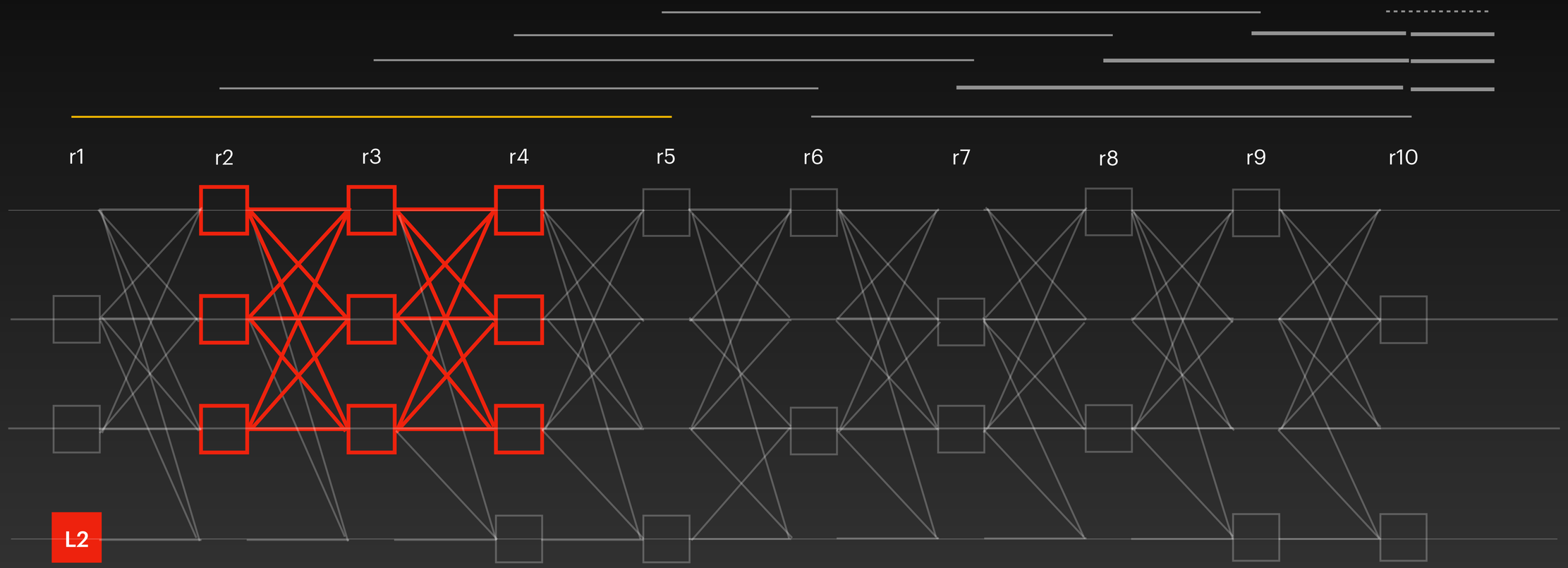
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



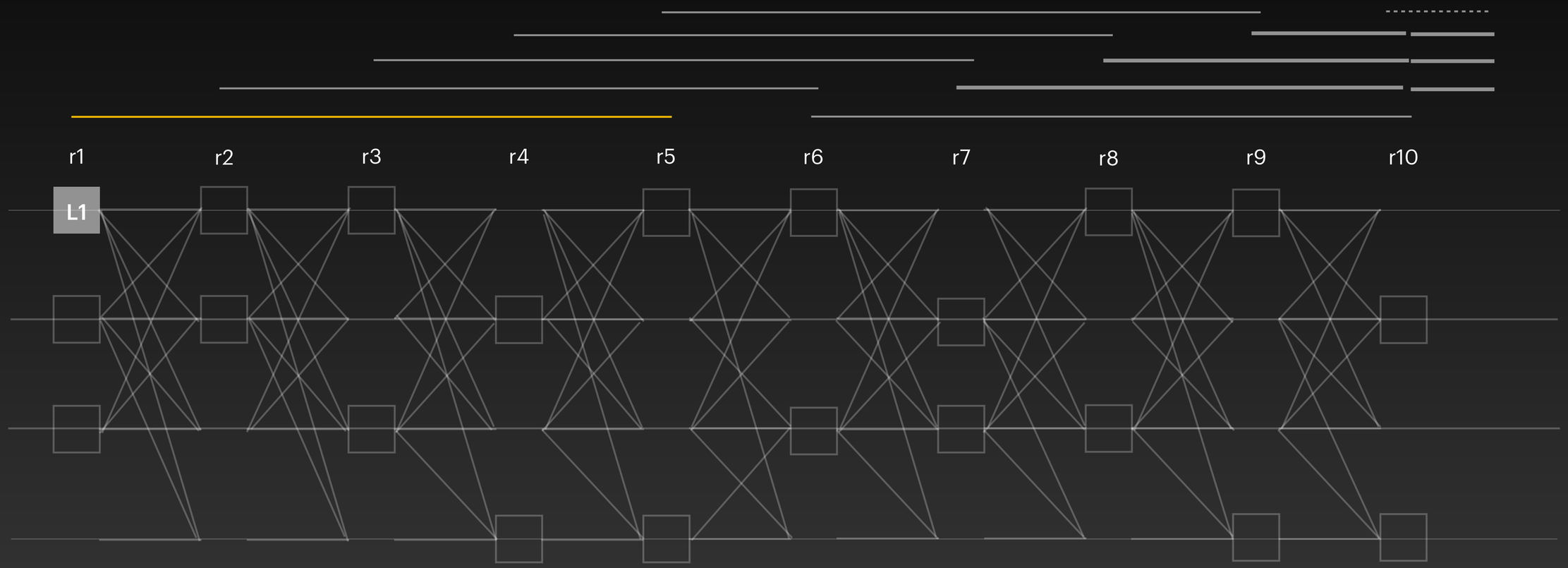
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



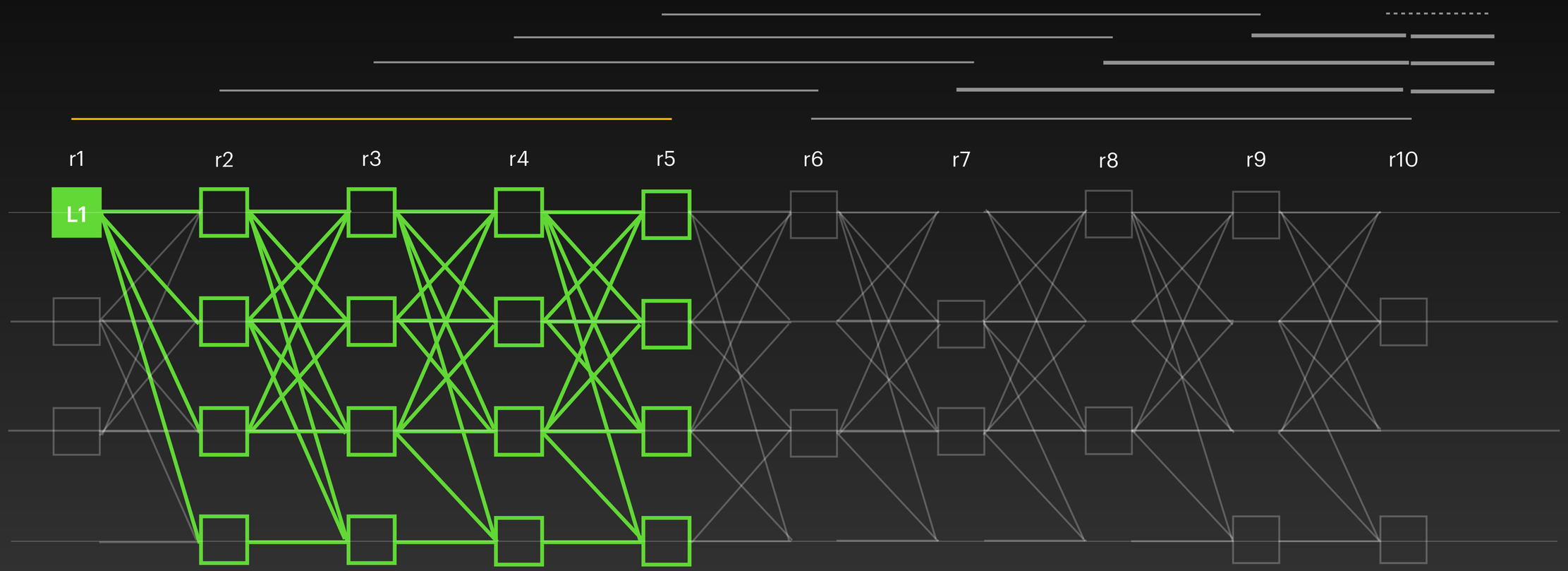
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



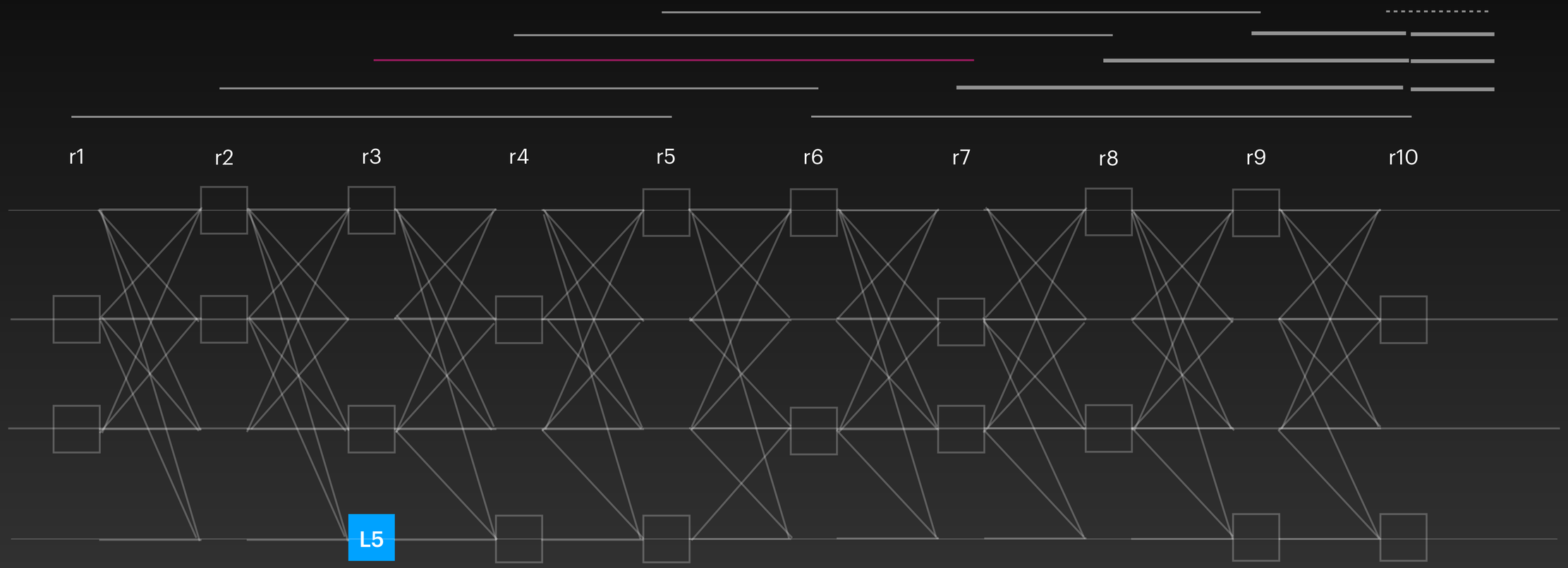
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Direct Rule



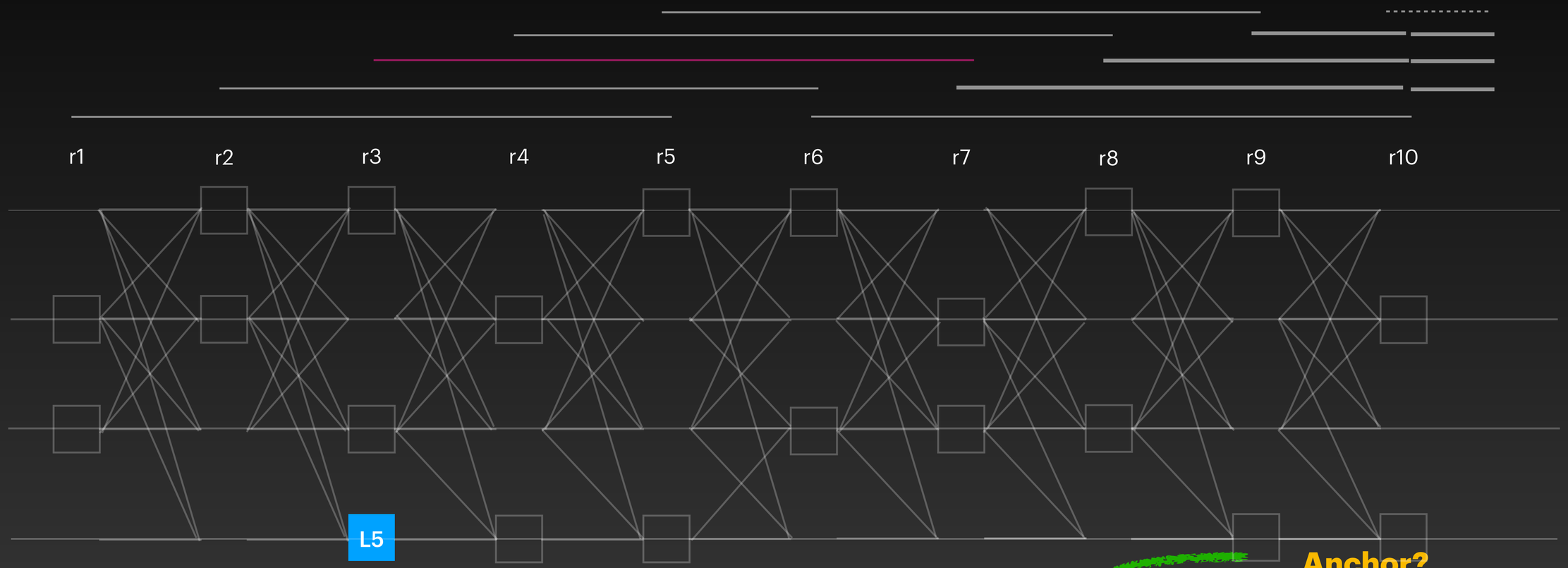
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Apply Indirect Rule



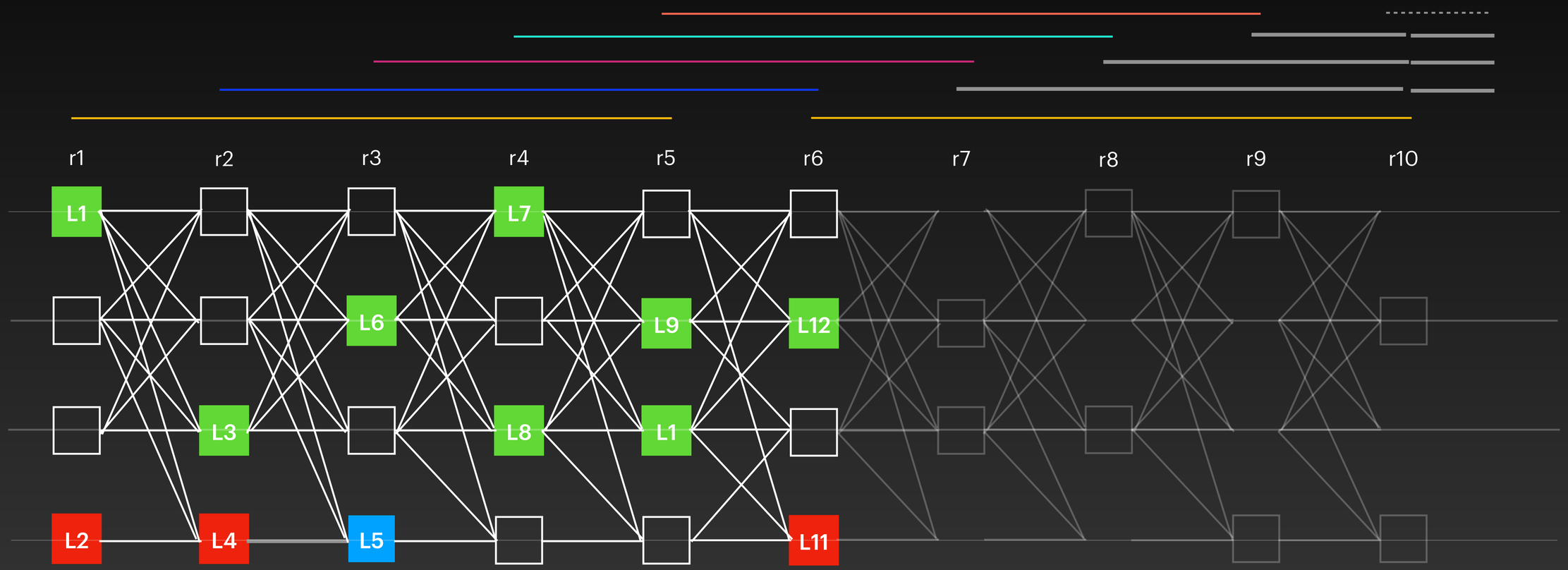
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Need to Wait for L13 and L14



L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

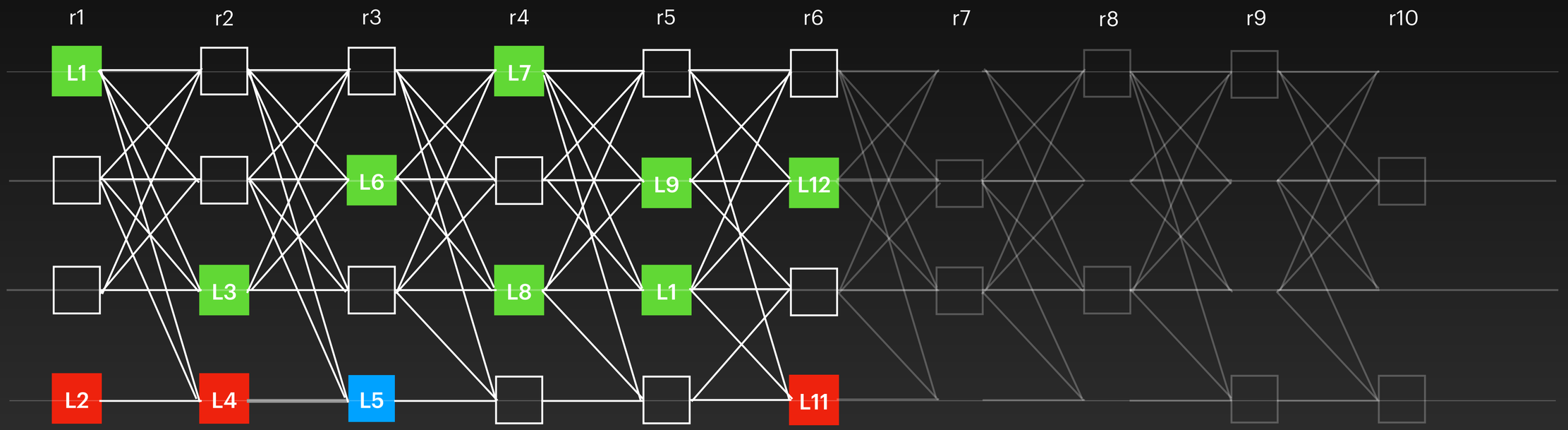
Current Status



L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Commit Sequence

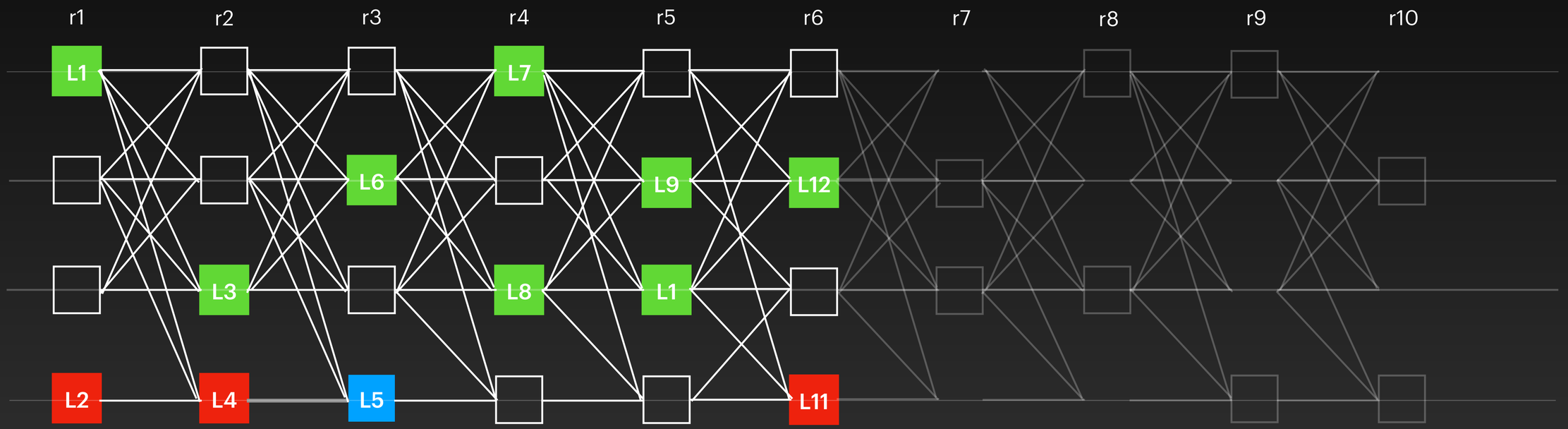
Take all leaders in order



leaders sequence: L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12

Commit Sequence

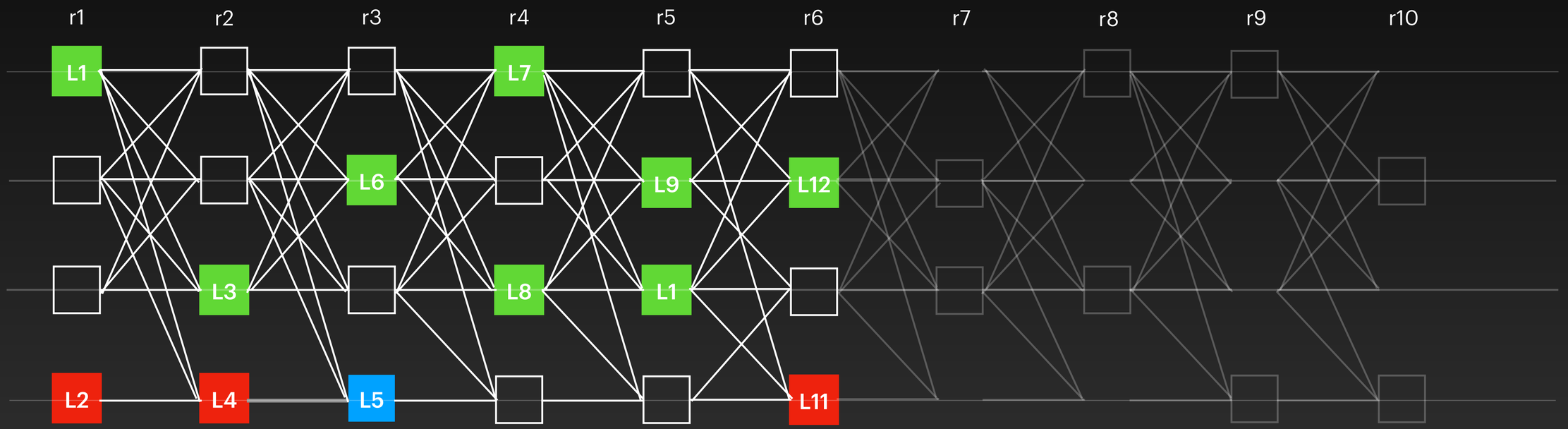
Stop at the first Undecided leader



leaders sequence: L1 L2 L3 L4

Commit Sequence

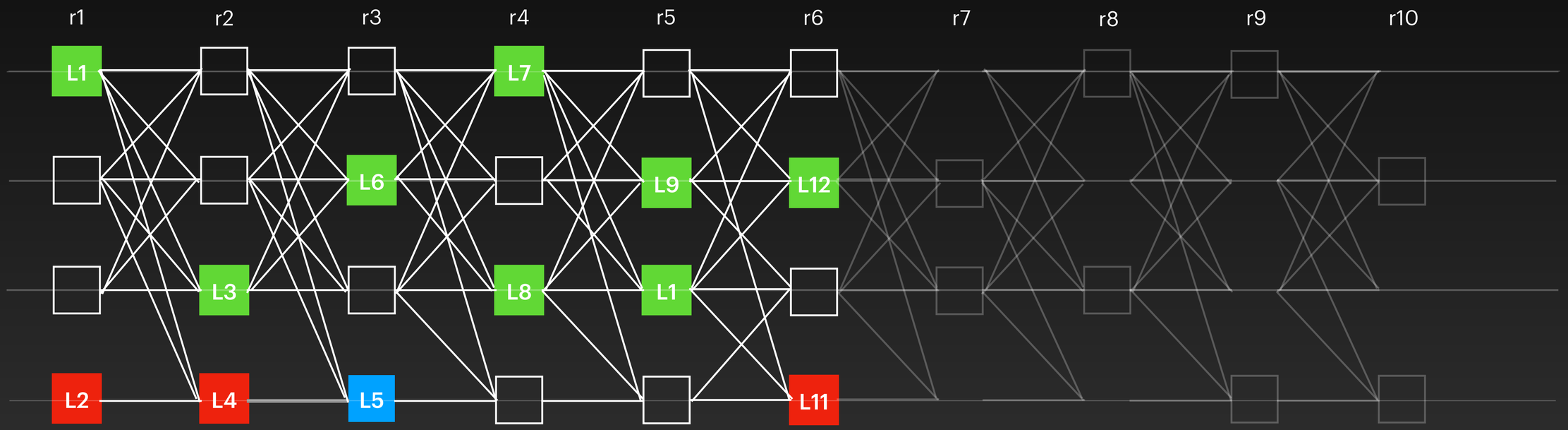
Remove skipped leaders



leaders sequence: L1 L2 L3 L4

Commit Sequence

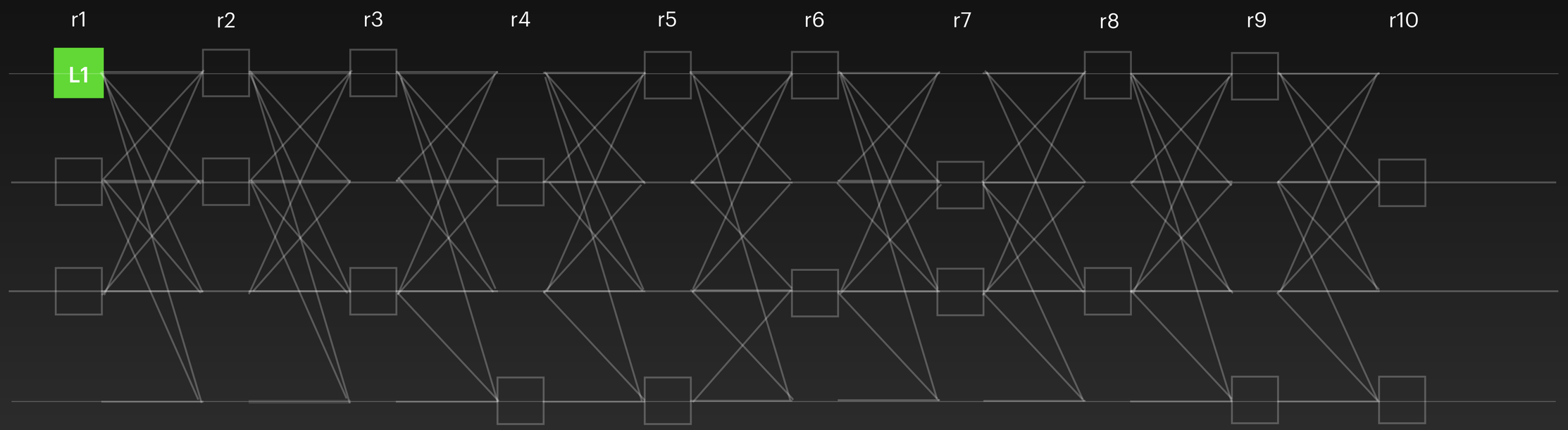
Final leader sequence



leaders sequence: L1 L3

Commit Sequence

Commit sub-dag

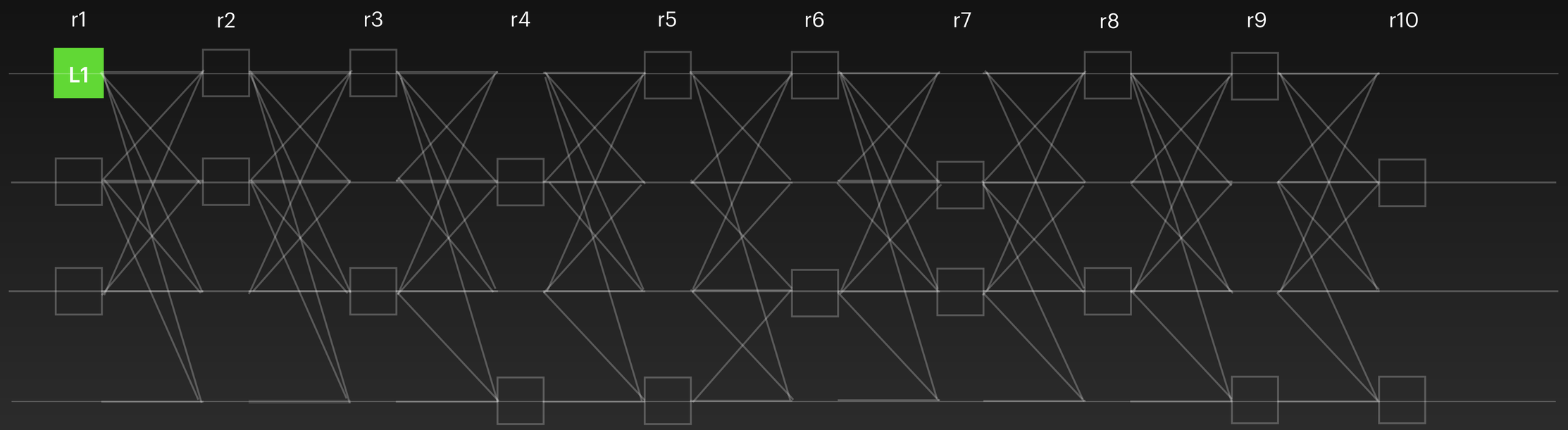


leaders sequence: L1 L3

output sequence:

Commit Sequence

Commit sub-dag

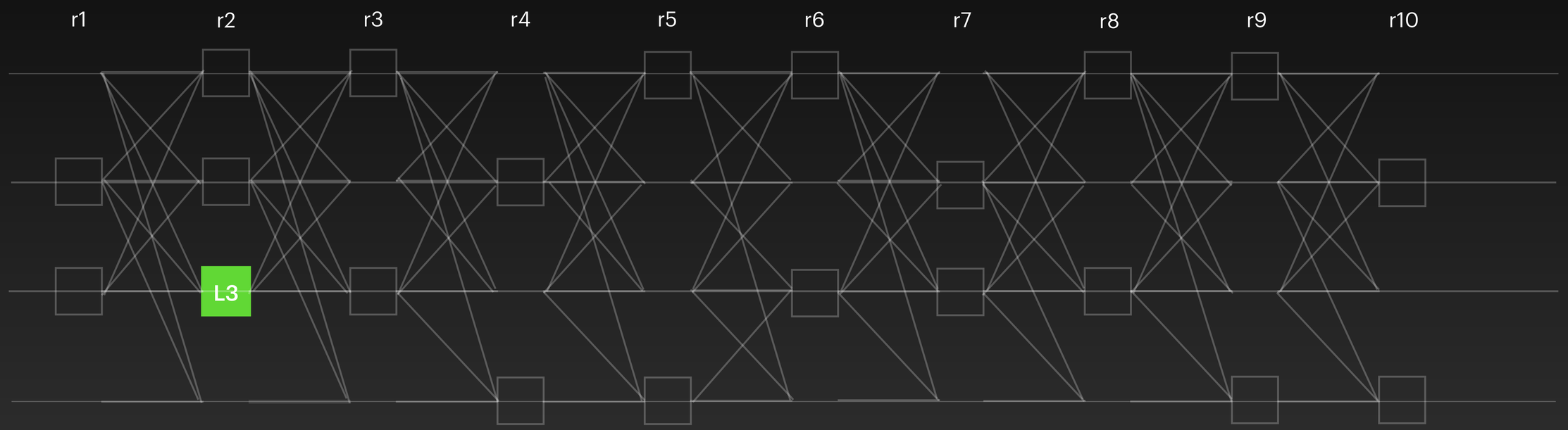


leaders sequence: L3

output sequence: L1

Commit Sequence

Commit sub-dag

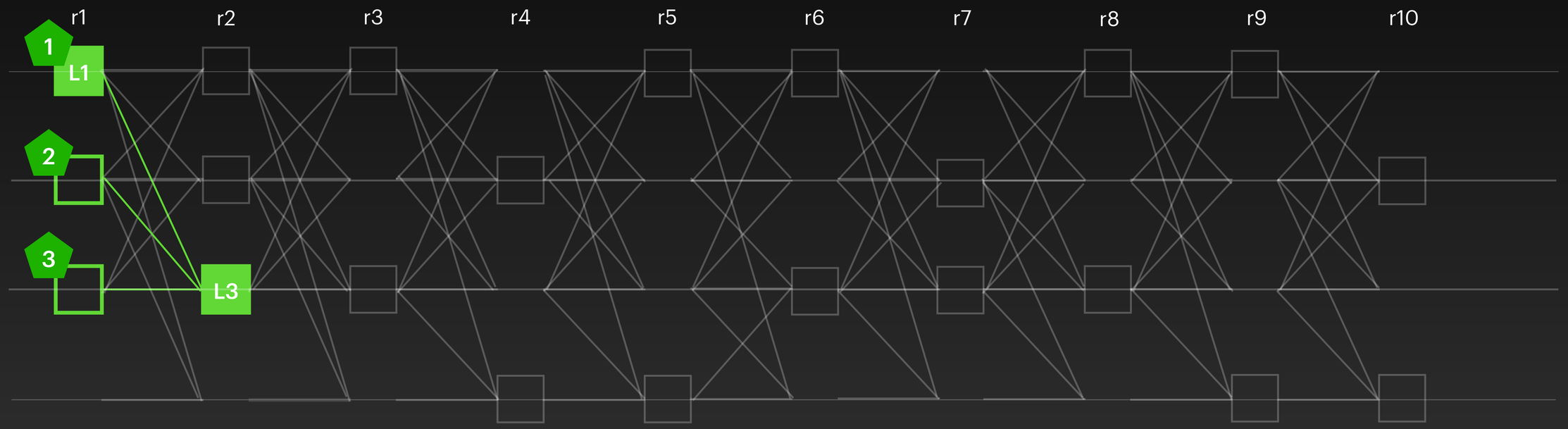


leaders sequence: L3

output sequence: L1

Commit Sequence

Commit sub-dag

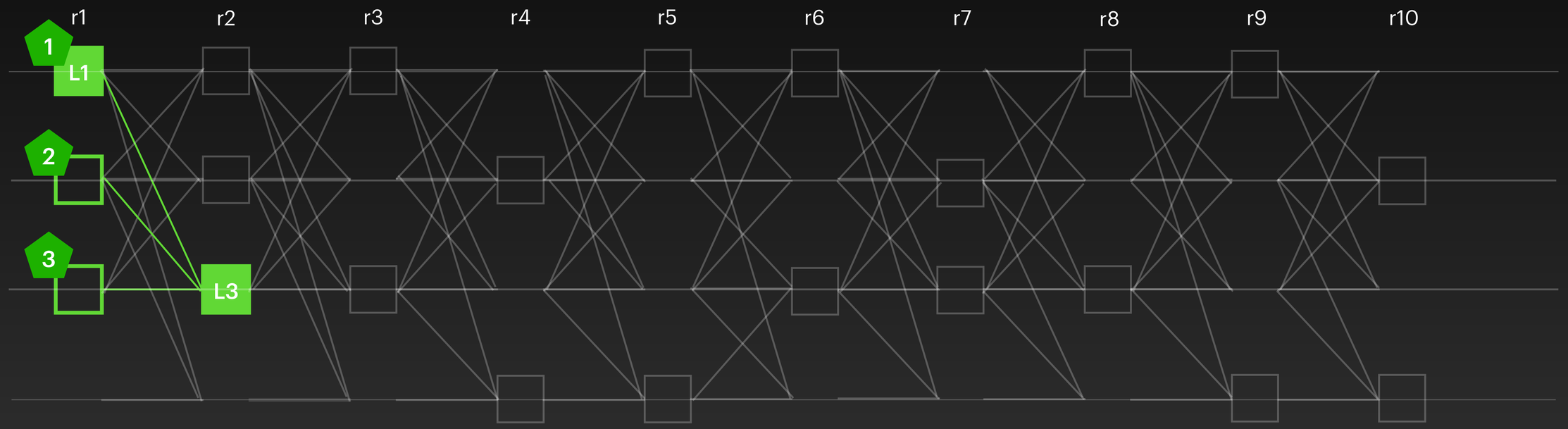


leaders sequence: L3

output sequence: L1

Commit Sequence

Commit sub-dag



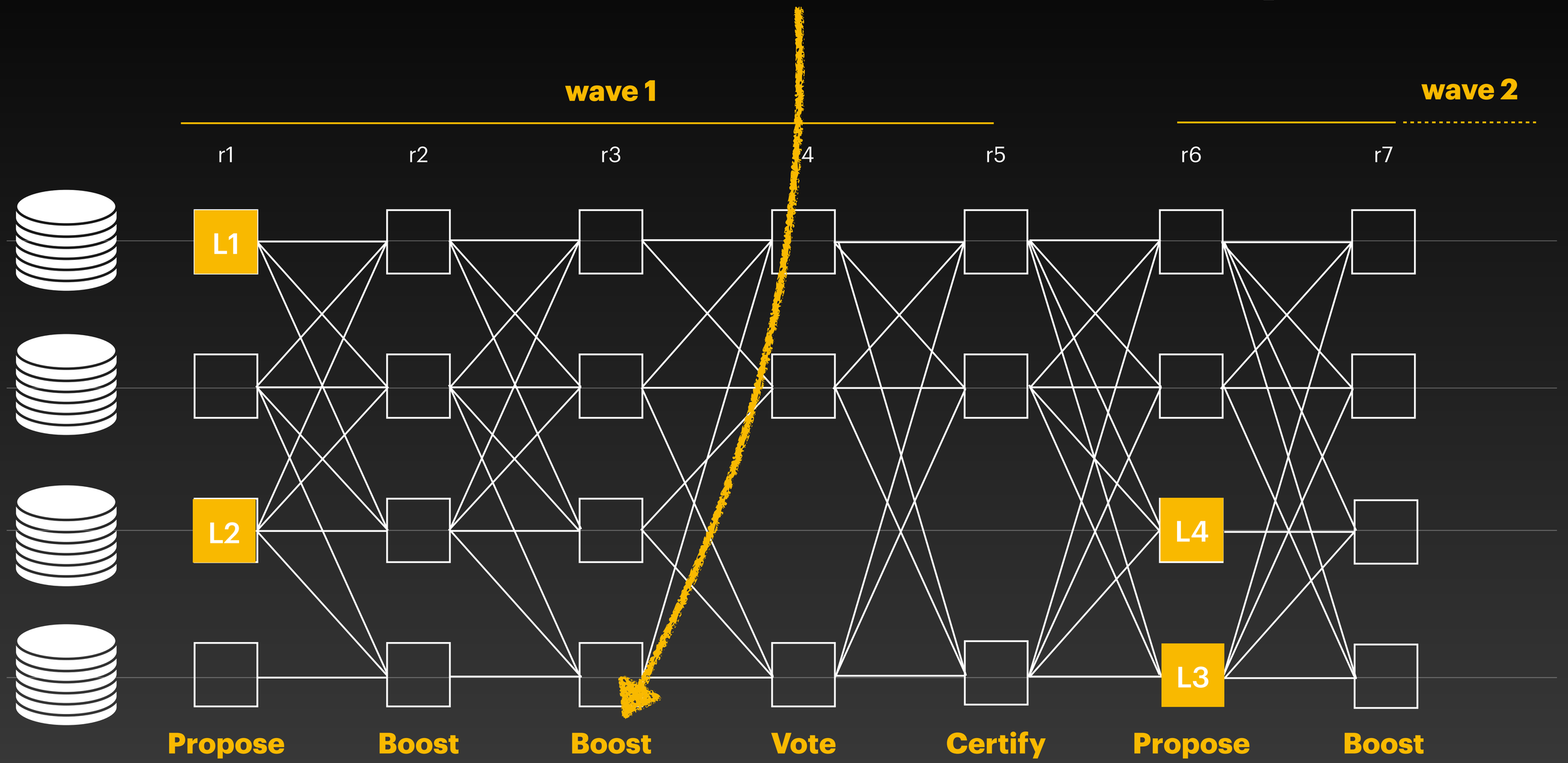
leaders sequence:

output sequence: L1 2 3 L3

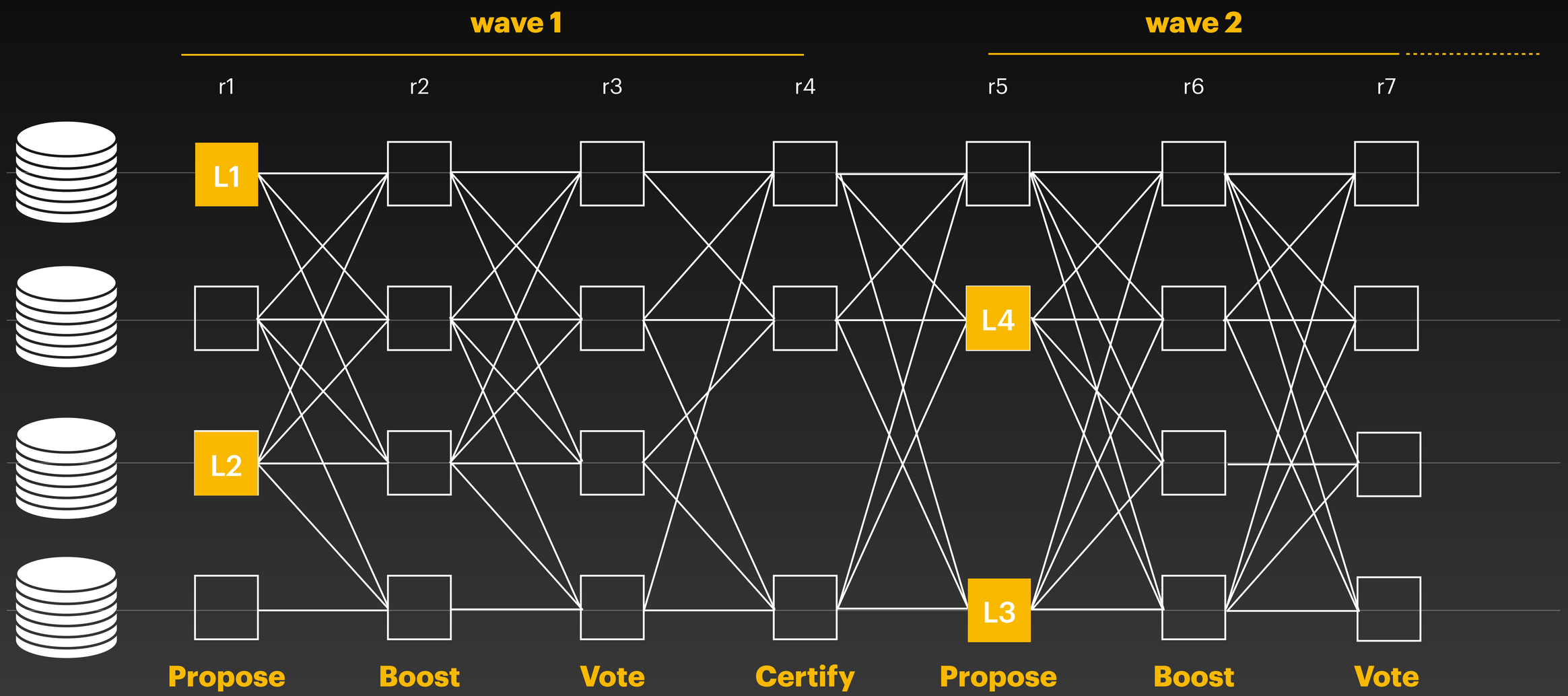
Parameterizing Mahi-Mahi

Number of Leaders + 4 or 5 Rounds per Wave

We can reduce the number of rounds per wave



Mahi-Mahi 4: 4 rounds per wave



Mahi-Mahi parameters

Select Number of Rounds and Number of Leaders per Round

5 rounds

- More round costs involved
- Higher probability of commits
- Maintains all security properties

4 rounds

- Less round costs involved
- Lower probability of commits
- Maintains all security properties

- Need to fix the size of leaders per round
- Both maintain liveness

Benchmarks

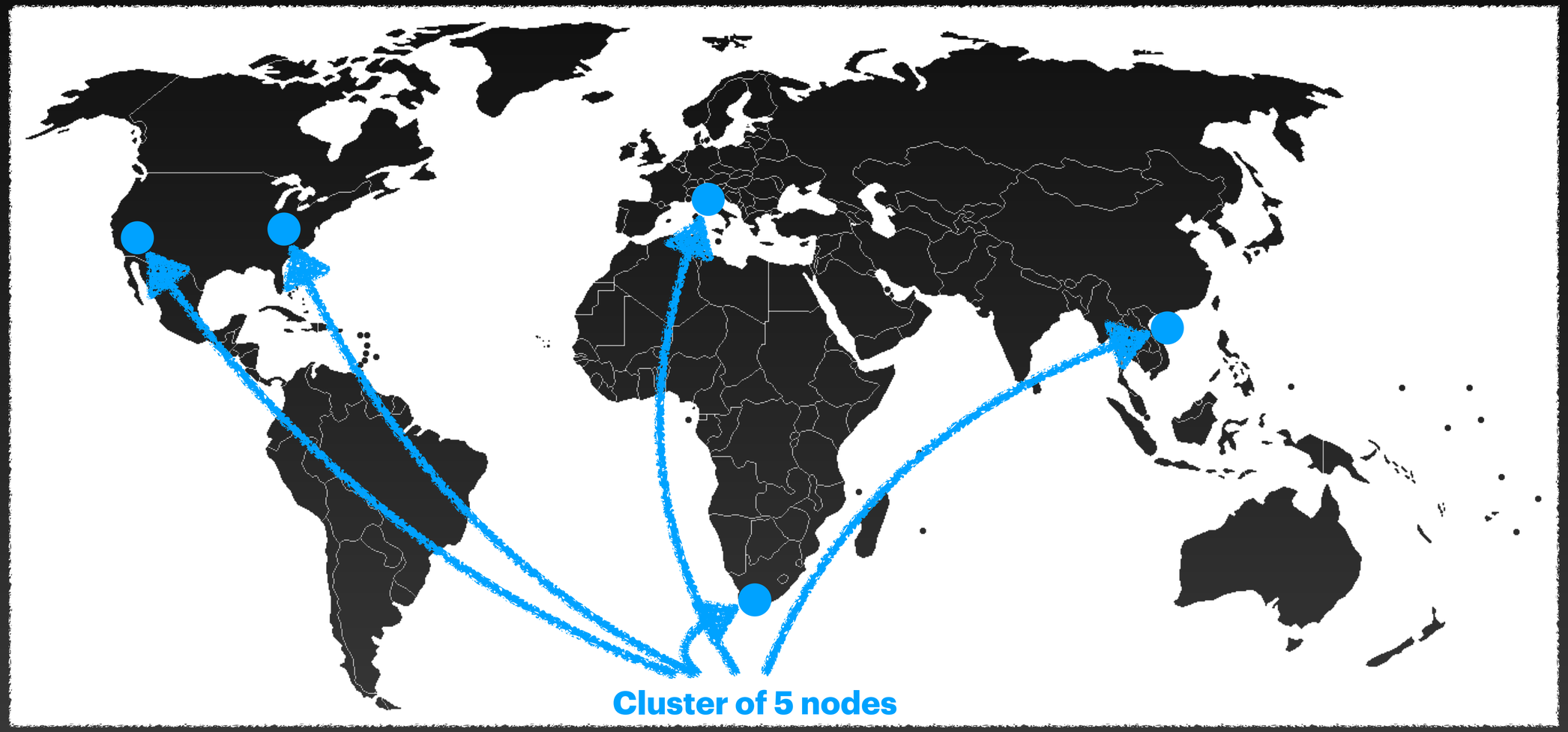
Implementation

- Written in Rust
- Networking: Tokio (TCP)
- Storage: custom WAL
- Cryptography: ed25519-consensus and blake2

https://github.com/PasinduTennage/mahi-mahi-consensus

Evaluation

Experimental setup on AWS



No Fault Performance

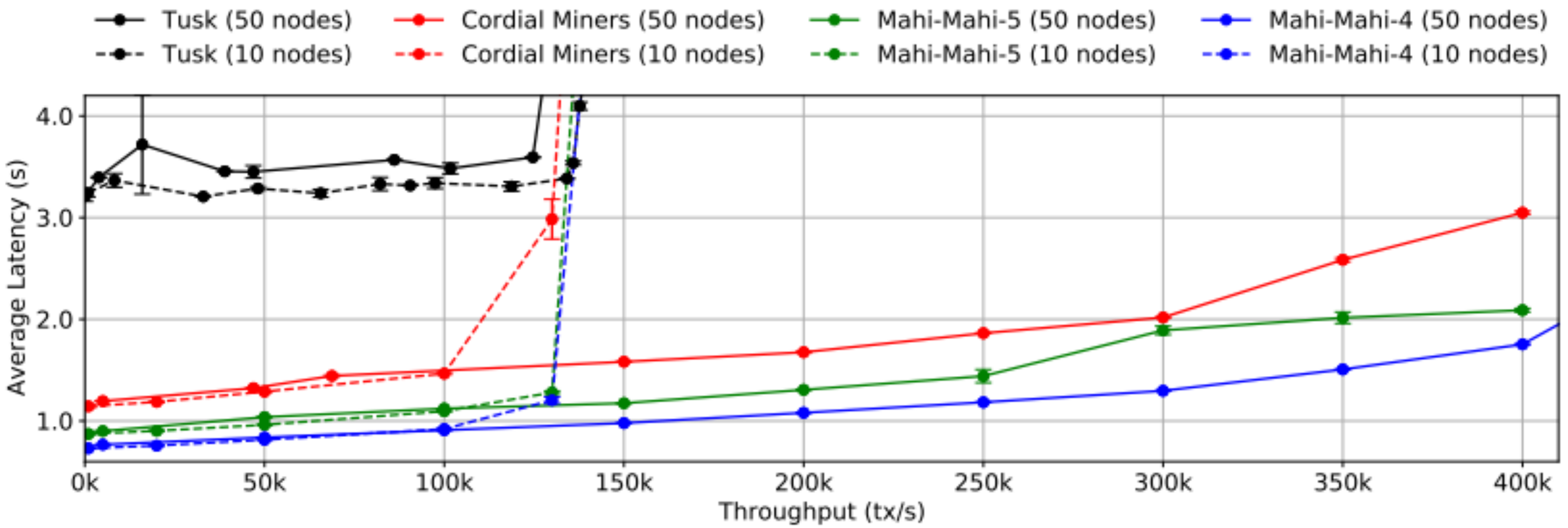


Fig. 3: Comparative throughput-latency performance of Mahi-Mahi, Tusk, and Cordial Miners. WAN measurements with 10 and 50 validators. No validator faults. 512B transaction size.

Faulty Nodes Performance

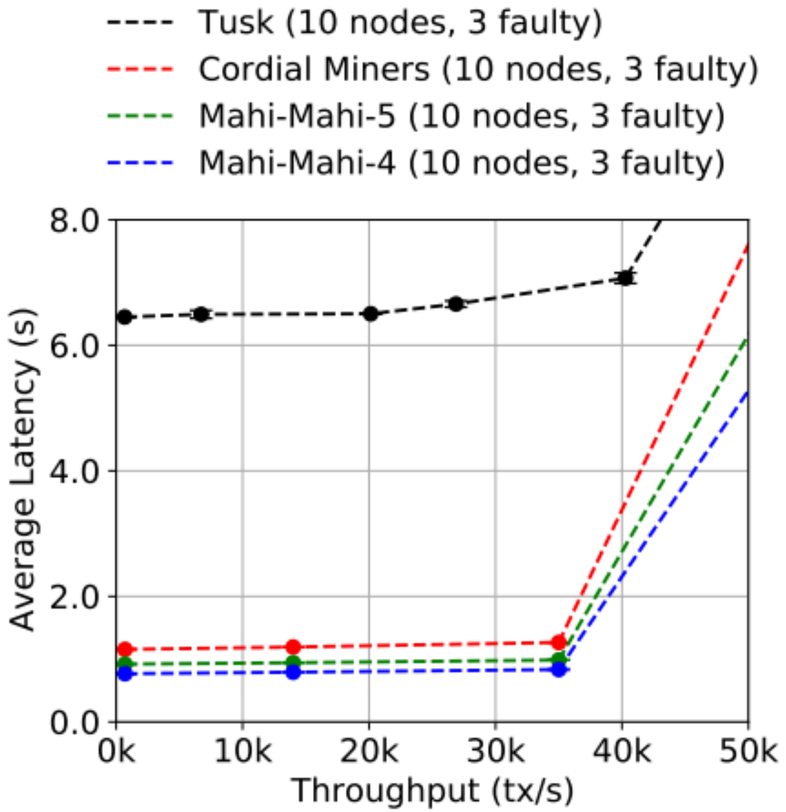


Fig. 4: Comparative throughput-latency of Mahi-Mahi, Tusk, and Cordial Miners. WAN measurements with 10 validators. Three faults. 512B transaction size.

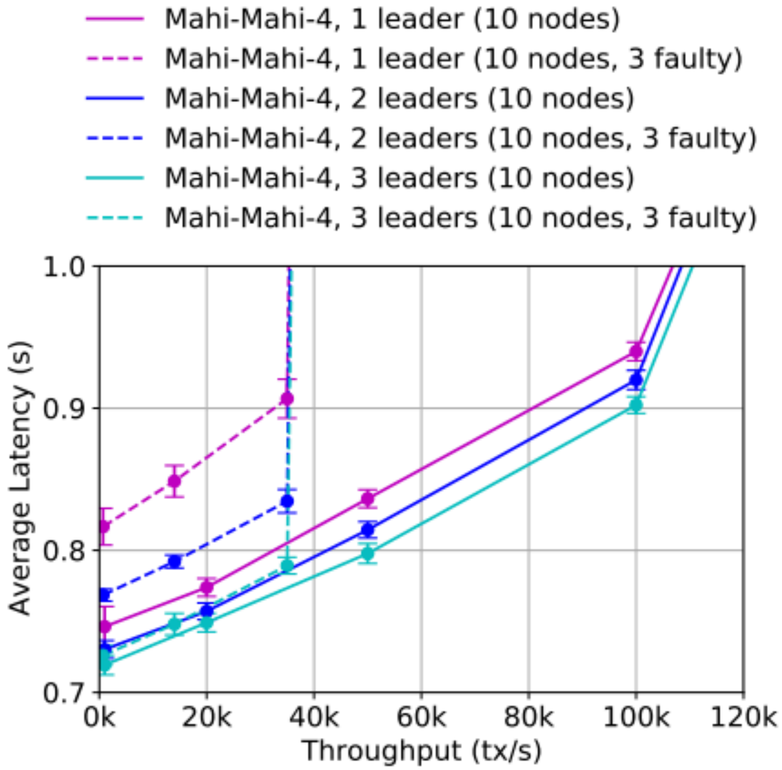


Fig. 5: Impact of the number of leaders per round in Mahi-Mahi. WAN measurements with 10 validators. Zero and three faults. 512B transaction size.

Various Mahi-Mahi Performance

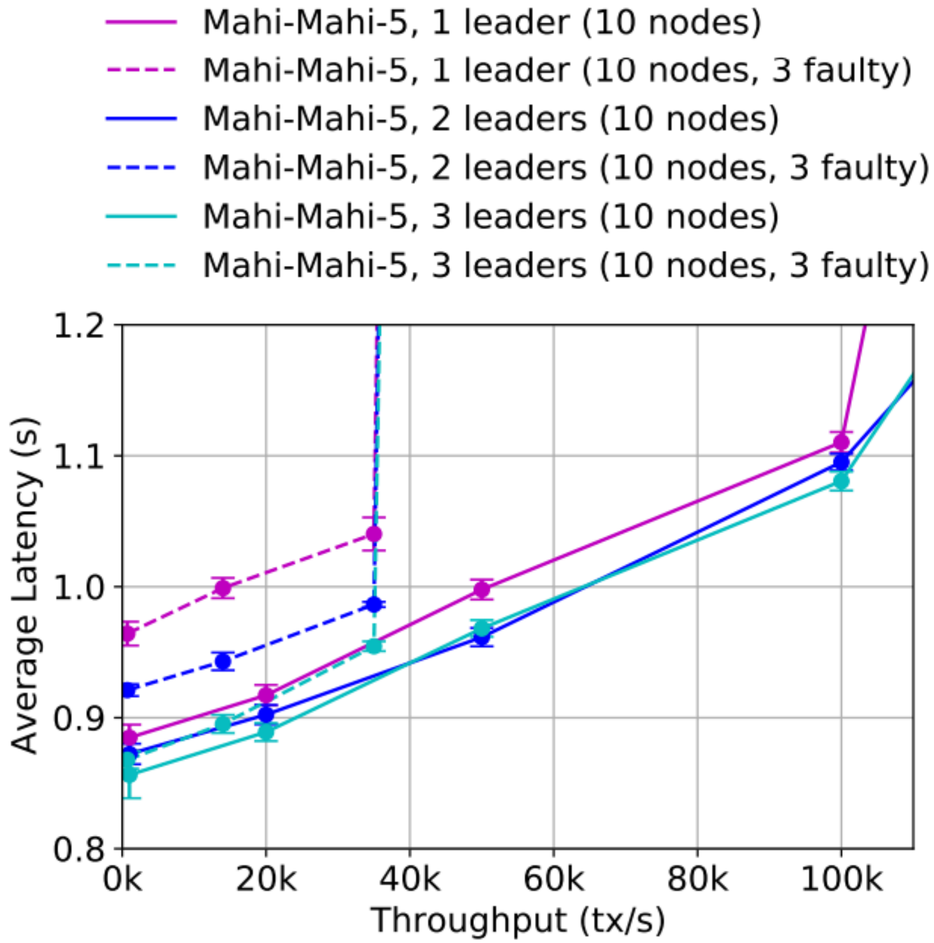


Fig. 7: Impact of the number of leaders per round in MAHI-MAHI. WAN measurements with 10 validators. Zero and three faults. 512B transaction size.

Summary

Mahi-Mahi

- Low-latency asynchronous protocol
- 100k tps with sub second latency

- **Paper:** https://www.arxiv.org/abs/2410.08670
- **Code:** https://github.com/PasinduTennage/mahi-mahi-consensus

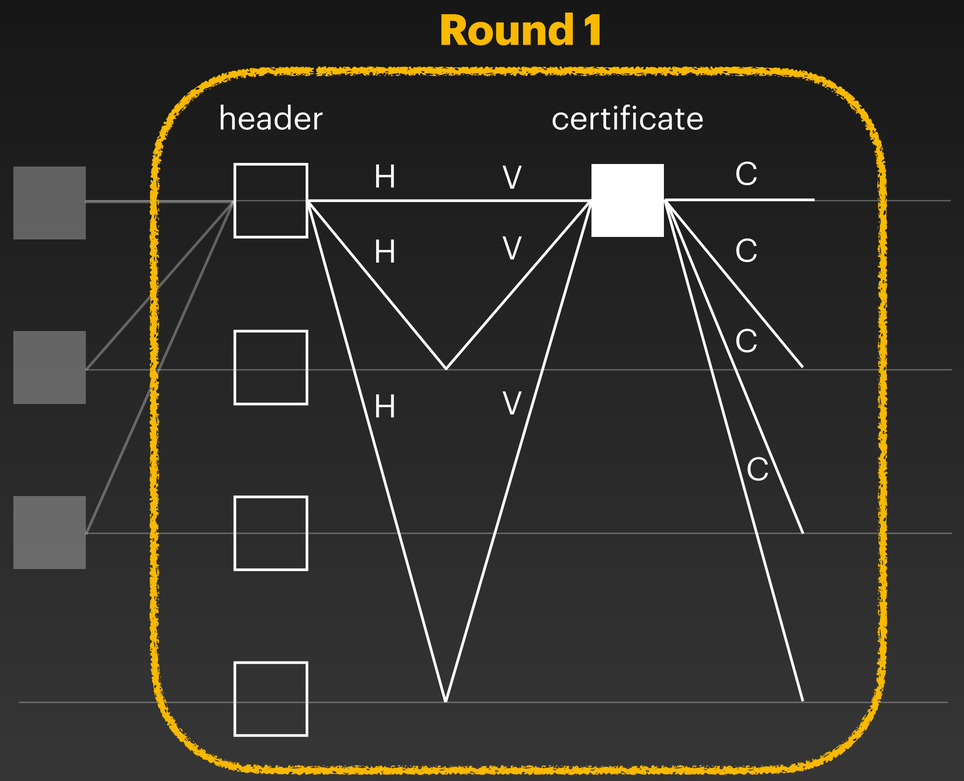
Narwhal vs Mahi-Mahi

Key differences & Insight

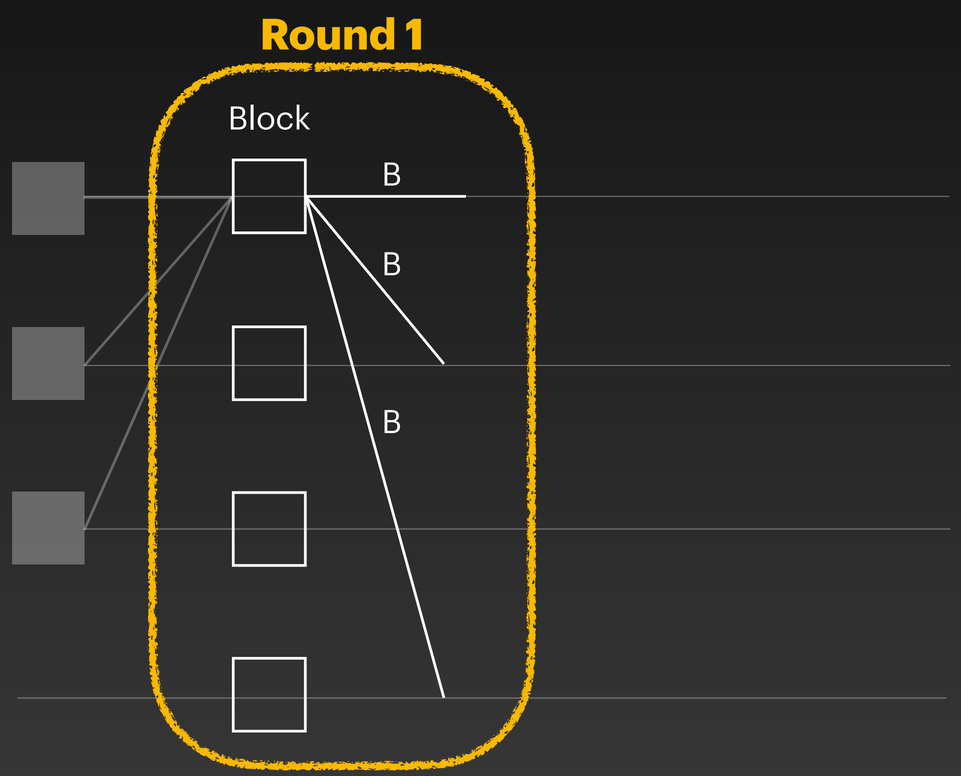
Generally kept as extra slides, anticipate CM being asked?

Narwhal vs Mysticeti

Narwhal

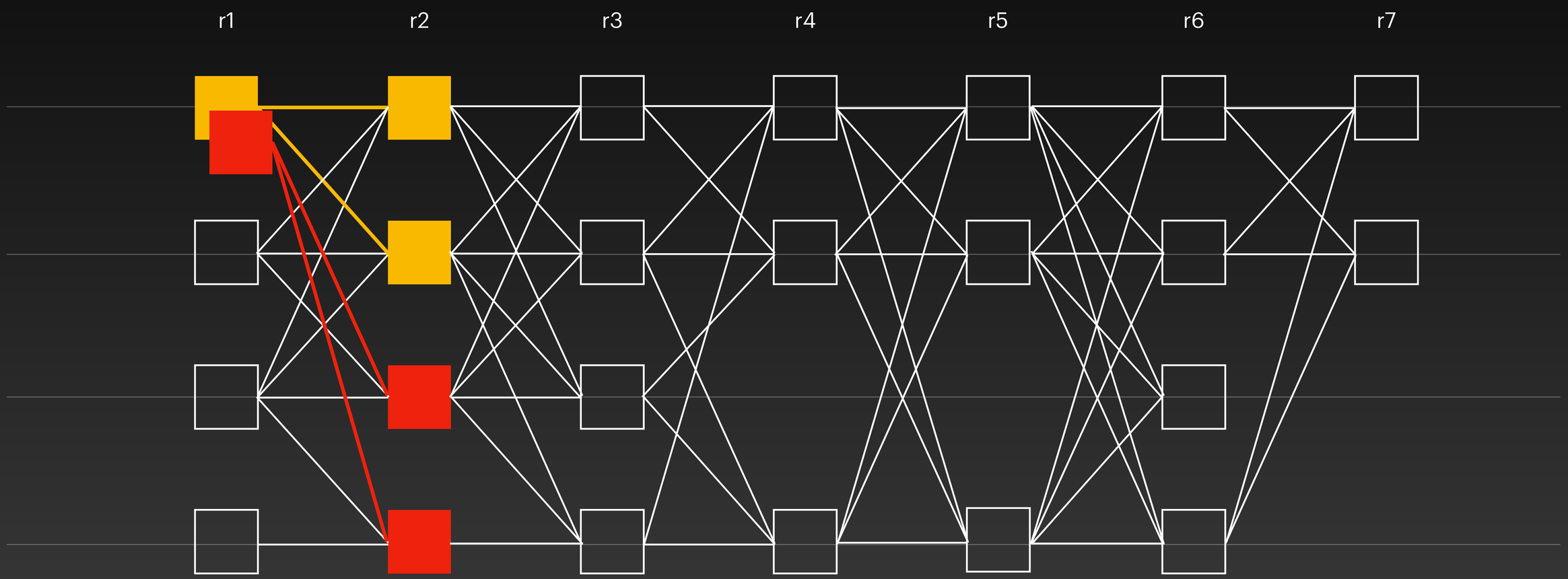


Mahi-Mahi



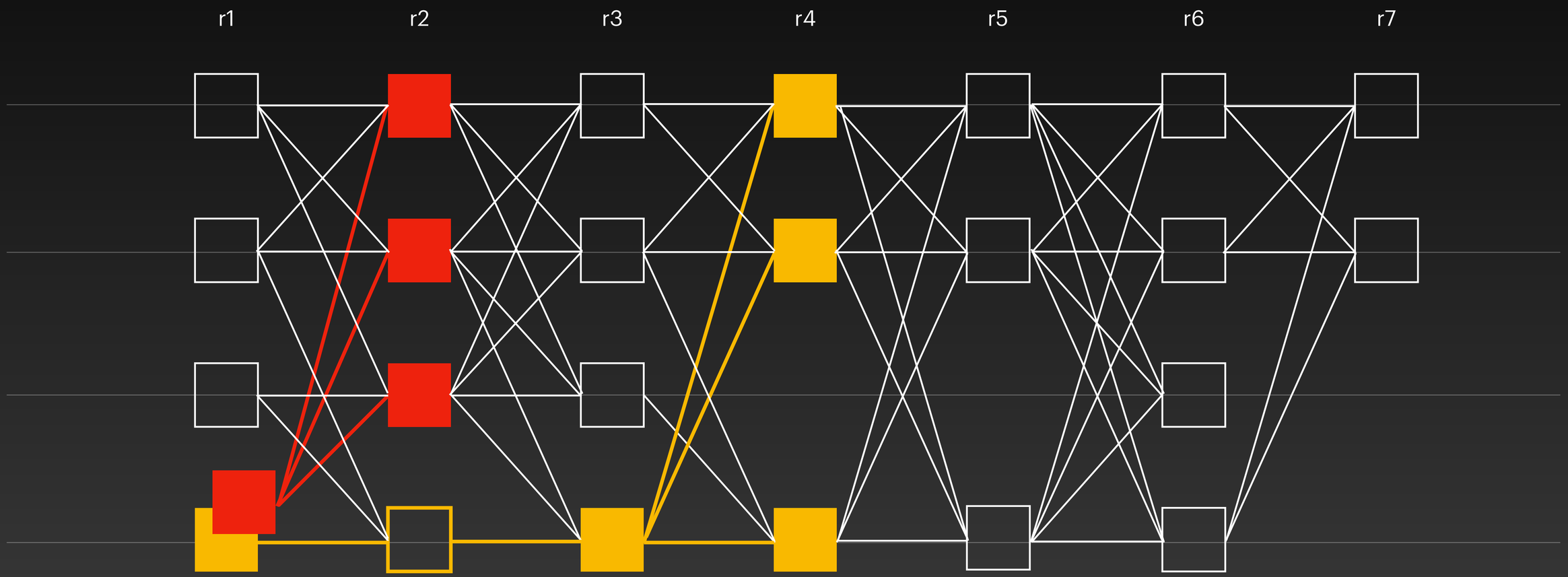
Main Challenge

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)

Mahi-Mahi

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