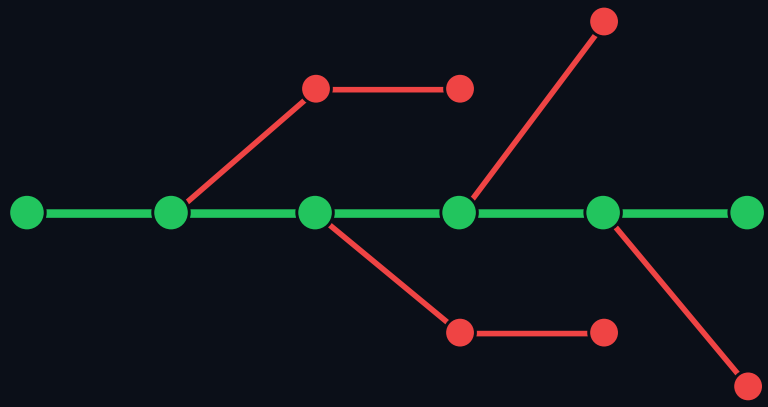
Distributed Mining Monitor

A controlled experimental platform to observe Proof-of-Work consensus, forks, stale branches, reorganisation events and difficulty adjustment.



PoW consensus lab

CPU/GPU miners

forks + reorgs

Tommaso Remedi · Giacomo Biagioni

Distributed Systems Course · A.Y. 2025–2026

What we built

The project in two minutes

Coordinator

Validates candidate blocks, stores the full DAG, selects the canonical branch and exposes APIs.

CPU/GPU miners

Independent processes searching for nonces. GPU miners use batched CUDA/CuPy search.

Dashboard

Observes the system in real time: forks, orphans, reorgs, difficulty and accepted blocks.

Controlled laboratory, not a cryptocurrency

No wallets, transactions, incentives, digital signatures or peer-to-peer propagation. The scope is didactic: make PoW branch competition observable and reproducible.

Core idea: allow temporary miner divergence, then observe how a canonical branch emerges again.

Why this is a distributed systems project

The system is distributed because independent processes compute, communicate, keep local state, and may temporarily disagree.

Key message The project intentionally trades immediate miner alignment for observability: miners can make local progress even when the global canonical branch has already advanced.

Overall architecture

Design choice The coordinator is centralised on purpose: it makes experiments reproducible, observable, and focused on PoW branch dynamics.

Mining interaction cycle

A miner does not ask for a full template at every attempt. It keeps a local branch and realigns only when the switch-lag rule says so.

Why this matters Local mining creates temporary divergence; head polling eventually brings miners back to the canonical branch.

Proof-of-Work and block template

Template returned by coordinator

- **height** — candidate block height
- **prev_hash** — parent block to extend
- **difficulty_bits** — PoW target for that height

Coordinator still validates everything again.

Local search condition

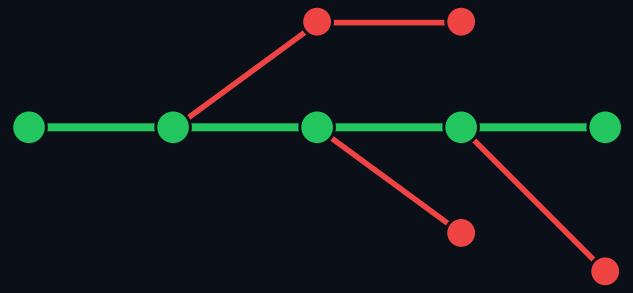
```
hash(height | prev_hash | nonce) < target(difficulty)
```

The nonce is the only value that the miner can freely modify. The difficulty determines how hard it is to find a hash that satisfies the required target.

Why pass difficulty to miners?

It avoids duplicating the difficulty-adjustment logic in CPU and GPU miners. The coordinator remains authoritative.

Forks, stale blocks and reorganisation



green = canonical branch

red = valid but non-canonical

Fork

two or more valid blocks share the same parent

Reject

invalid submission: wrong parent, height, PoW or duplicate

Stale / orphan

valid accepted block not in the current canonical branch

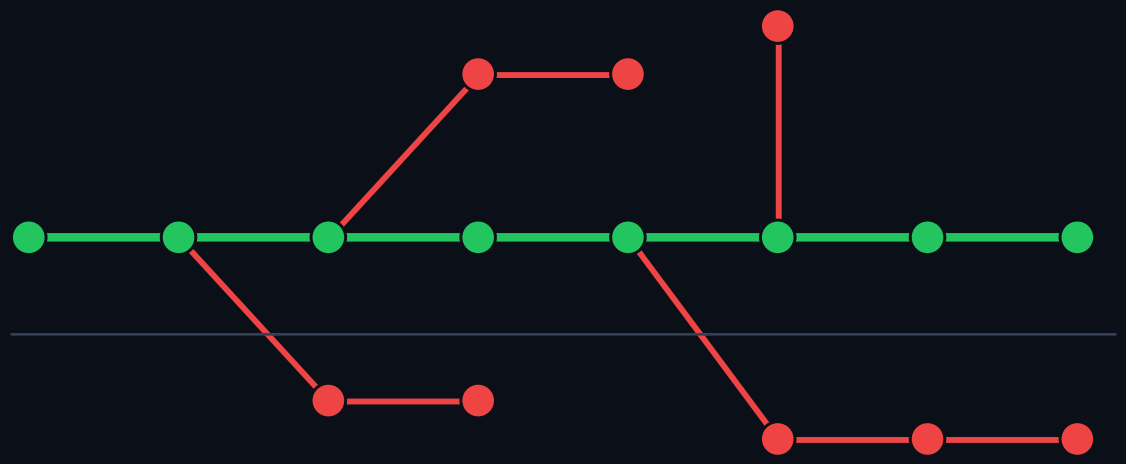
Reorg

canonical branch changes to a competing branch

Key distinction stale/orphan blocks are usually valid blocks kept in the DAG; rejects are validation failures.

How to read the block DAG

The dashboard graph is not just a sequence of dots: every block has a parent, and edges show how branches grow or die.



Canonical path

Green blocks and edges form the current main chain selected by the coordinator.

Stale/orphan branches

Red blocks are still valid and stored, but they are not part of the current canonical branch.

Connected edges make branch persistence and reorgs visible: a reorg changes which path becomes green.

Two parameters that control branch dynamics

They work at different levels: one controls miner realignment, the other controls coordinator reorganisation.

SwitchLagBlocks

Miner-side rule

How far behind a miner can be before it abandons its local branch and requests a new canonical template.

Lower value	faster realignment, shorter-lived local branches
Higher value	miners tolerate more lag, so competing branches can persist longer

ReorgThreshold

Coordinator-side rule

How much a competing branch must exceed the current canonical branch before the coordinator switches to it.

Lower value	easier reorgs, more reactive canonical choice
Higher value	harder reorgs, more stable canonical branch

Important reorgs also depend on delay, difficulty and miner speed.

Difficulty adjustment and target

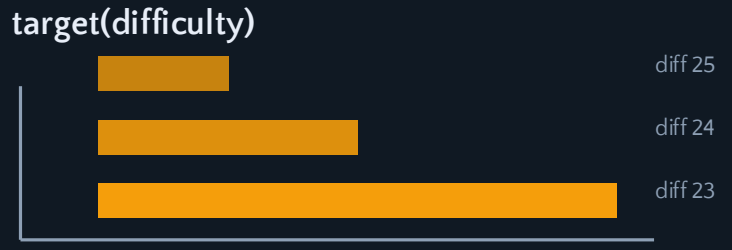
The comparison is always hash < target. When difficulty increases, the target becomes smaller, so fewer hashes are valid.

Height-based rule

1–99	base	diff 23
100–199	+1 bit	diff 24
200–299	+2 bits	diff 25
300+	+3 bits	diff 26

```
difficulty = base + floor(height / interval)
```

Why +1 bit matters



difficulty ↑ ⇒ target ↓ ⇒ fewer valid hashes

+1 bit approximately halves the probability of success, therefore the expected work roughly doubles.

Experiments: what changes across tests

Do not read this as a benchmark table only: each test isolates a different behaviour of the system.

T1

CPU baseline

linear chain

10

final height

0 forks / 0 reorgs

T2

GPU baseline

same logic, faster miner

422

final height

0 forks / 0 reorgs

T3

Mixed competition

CPU+GPU branch competition

261

final height

42 forks / 7 reorgs

T4

Switch lag sensitivity

same setup, lower SwitchLagBlocks

271

final height

40 forks / 15 reorgs

T5

Difficulty adjustment

long run with increasing difficulty

226

final height

34 forks / 12 reorgs

Interpretation

T1/T2 prove linear baselines. T3/T4 show competition and branch persistence. T5 proves that difficulty updates do not break fork handling or reorg logic.

Zero rejects is not a problem

Rejects count invalid submissions. Stale/orphan blocks are different: they are valid blocks accepted in the DAG but excluded from the current canonical branch.

Connection with course topics

Consensus

miners may diverge; canonical branch restores agreement

CAP

local mining favours availability; coordinator restores consistency later

DLT / Blockchain

PoW, forks, stale blocks, difficulty, branch selection

Replication & consistency

local views are temporarily inconsistent, then converge

Dependability

invalid inputs are tolerated, but coordinator is a SPOF

Logging & recovery

future work: persistent DAG, checkpoints, event logs

“We implemented a controlled lab for observing distributed consensus phenomena, not a production blockchain.”