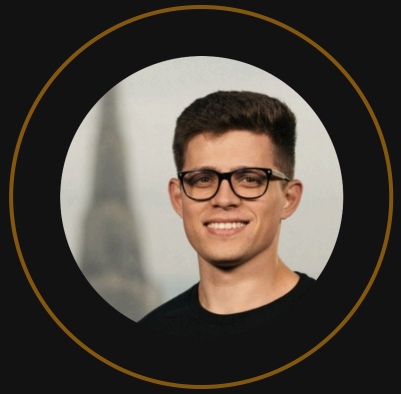
From Solo Mining to Poolless Mining

The Evolution of Bitcoin Mining Protocols

Johnny Santos `@johnnyasantoss`

TABConf 8 - Main Stage | Thursday, Oct 15, 2026

About Me



Johnny Santos

X: @johnnyasantos / GH: @johnnyasantoss

johnnyasantos.com

PGP ⇒ 76AC 83F1 AA14 9FF9 CB1E E8F8 F6B9 4BAC B8DF 2685

Software Engineer & Technical Leader

Over six years building Bitcoin tools, platforms, and infrastructure. Fourteen years of experience.

Focus & Passion

Bitcoin mining, hardware tinkerer, privacy advocate, open-source enthusiast and lately local AI inference.

vinteum Grantee

Contributor to decentralized mining protocol (**P2Poolv2**); studying the historical landscape of Bitcoin mining and Bitcoin Core IPC

Thank you for the support.

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.
2	`2010–2012`	Hashing leaves the node; the node still builds the block.

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.
2	`2010–2012`	Hashing leaves the node; the node still builds the block.
3	`2012`	GBT sends a full template, not a header.

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.
2	`2010–2012`	Hashing leaves the node; the node still builds the block.
3	`2012`	GBT sends a full template, not a header.
4	`2011–present`	Stratum v1 becomes the device-edge default and the industrial standard.

The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.
2	`2010–2012`	Hashing leaves the node; the node still builds the block.
3	`2012`	GBT sends a full template, not a header.
4	`2011–present`	Stratum v1 becomes the device-edge default and the industrial standard.
5	`2011–2018`	P2Pool and AuxPoW test alternatives.

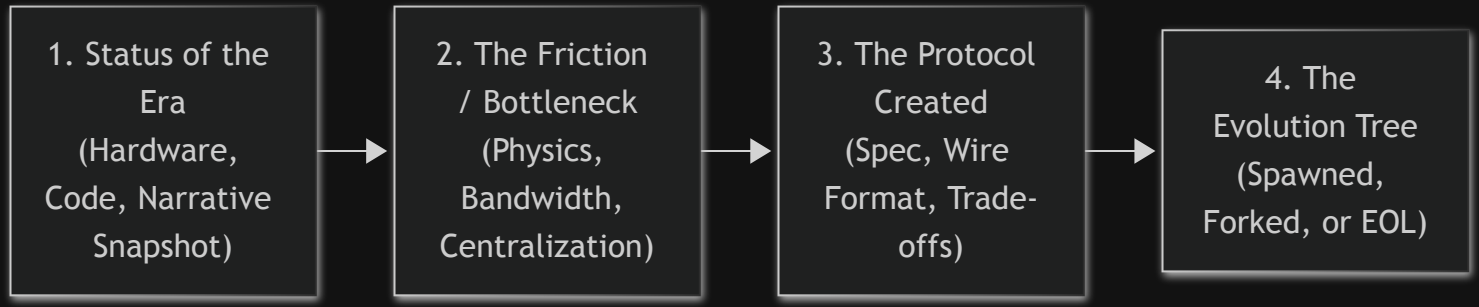
The Arc of This Tour

Every protocol changes the relationship between miner, pool, and node.

Era	Period	How it was/is
1	`2009–2010`	Miner and node are the same process.
2	`2010–2012`	Hashing leaves the node; the node still builds the block.
3	`2012`	GBT sends a full template, not a header.
4	`2011–present`	Stratum v1 becomes the device-edge default and the industrial standard.
5	`2011–2018`	P2Pool and AuxPoW test alternatives.
6	`2018–2026`	SV2, DATUM, and P2Poolv2 return template construction to miners.

Method: The Historical Loop

We will walk through each era using the same structure:



*At each transition, we will update our **Protocol Evolution Tree** so you can trace what survived, what died, and what was reborn.*

The Five Lenses Across History

Each era changes one or more of these boundaries:

1. **Who constructs the block?**

   Transaction selection, mempool filtering, fee calculation, and Merkle-root production.
2. **Who controls the work?**

   Search-space partitioning: nonce, extranonces, `nVersion` bits, and timestamps.
3. **How are shares accounted for?**

   Proof-of-work verification, share logs, PPLNS/PPS, and variance reduction.
4. **What must be trusted?**

   Custody of block rewards, transaction censorship resistance, and honest accounting.
5. **What resources does the miner need?**

   Compute, local node state, network bandwidth, and latency.

Era 1: The Mining is in the Client

2009 - 2010

Era 1: Mining Was a Function

`BitcoinMiner()` runs inside the node, in `src/main.cpp`.

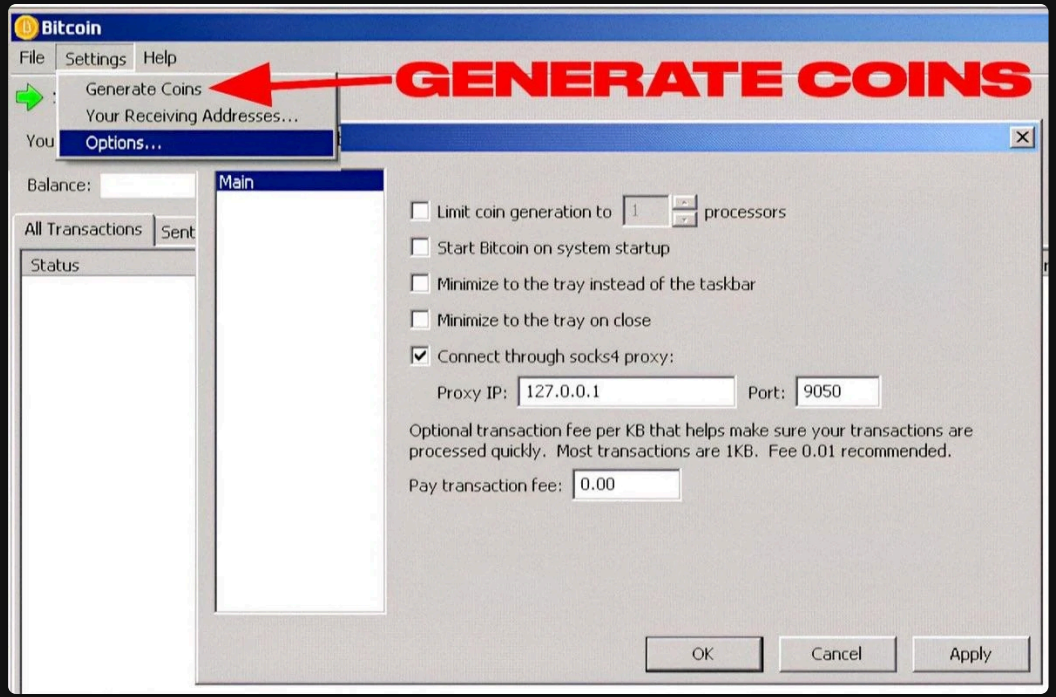
```
// BitCoin - v0.1.3
bool BitcoinMiner() {
    CBigNum bnExtraNonce = 0;
    while (fGenerateBitcoins) {
        // ... coinbase, tx selection, merkle ...
        txNew.vin[0].scriptSig << nBits << ++bnExtraNonce;
        // ... hashTarget, temp CBlock ...
        loop {
            if (hash <= hashTarget) {
                pblock->nNonce = tmp.block.nNonce;
                // proof-of-work found; repeat
```

– Bitcoin v0.1.3, `src/main.cpp`

- The miner generated its payout key, built the coinbase, selected transactions, and hashed the block.

Era 1: Mining Was a Function

`BitcoinMiner()` runs inside the node, in `src/main.cpp`.



- `-gen` / GUI `Generate Coins` set `fGenerateBitcoins`; `ui.cpp` threads `_beginthread(ThreadBitcoinMiner, ...)`.

Era 1: The 32-bit Nonce and `bnExtraNonce`

The header field scanned by the miner was a 32-bit `nNonce`.

```
txNew.vin[0].scriptSig << nBits << ++bnExtraNonce;
pblock->hashMerkleRoot = pblock->BuildMerkleTree();

tmp.block.nNonce = pblock->nNonce = 1;
FormatHashBlocks(&tmp.block, sizeof(tmp.block));

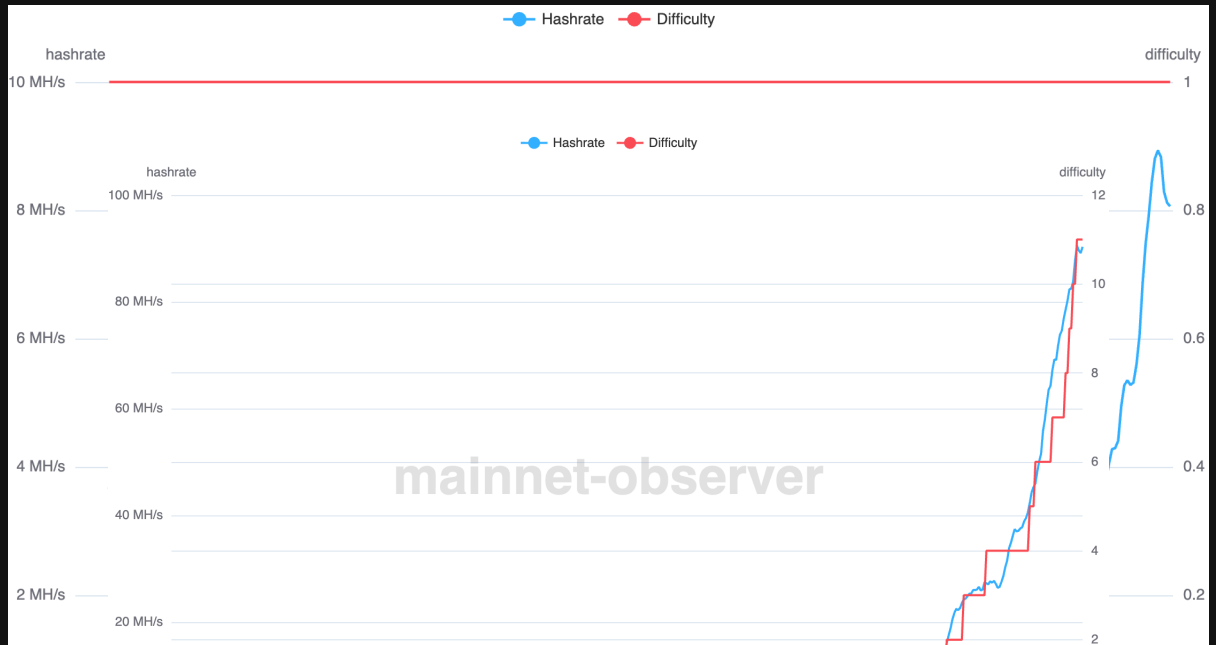
if ((++tmp.block.nNonce & 0x3ffff) == 0)
{
    if (tmp.block.nNonce == 0)
        break; // rebuild block with next bnExtraNonce
}
```

– Bitcoin v0.1.3, `src/main.cpp` (nonce wrap)

- `bnExtraNonce` entered the coinbase `scriptSig` before the Merkle root was built.
- Incrementing it changed the coinbase, the Merkle root, and therefore the whole header.
- This is the ancestor of later pool nonce-space partitioning, including Stratum's `extranonce1` and `extranonce2`.

Era 1: Friction & Collapse

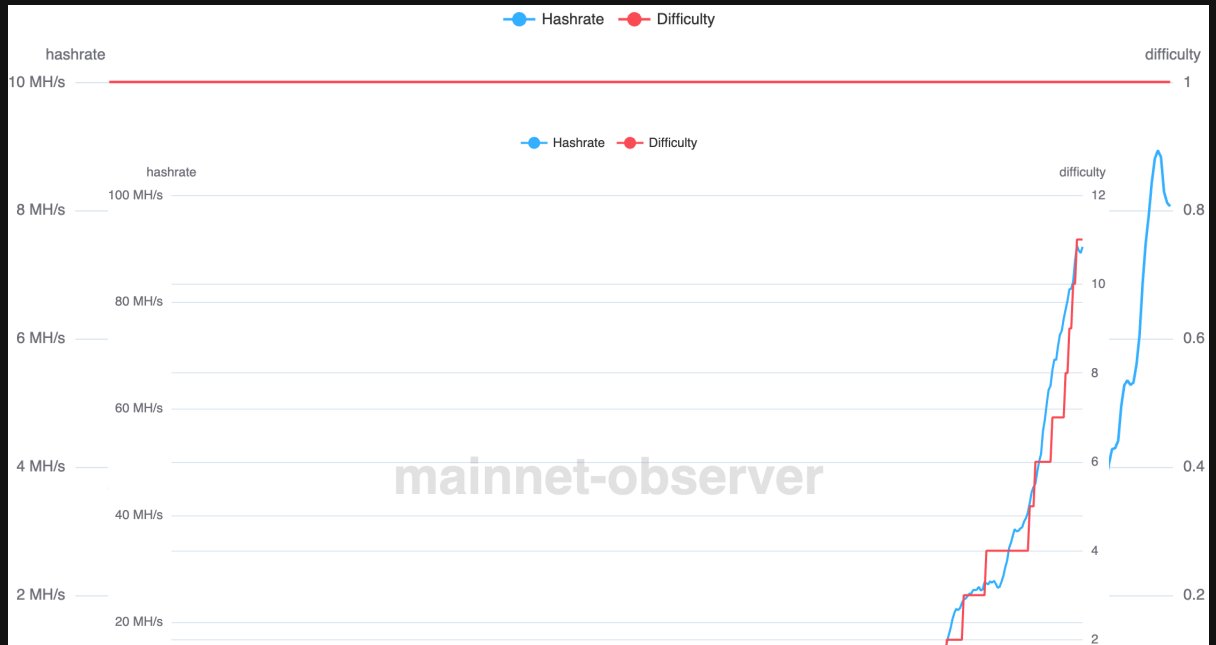
The era of client-embedded mining ended between mid-`2009` and late `2010`.



Era 1: Friction & Collapse

The era of client-embedded mining ended between mid-`2009` and late `2010`.

1. **Difficulty retargeted:** Sat at `1.0` until block `32256` (30 Dec `2009`), when it jumped to `1.18`.



Era 1: Friction & Collapse

The era of client-embedded mining ended between mid-`2009` and late `2010`.

1. **Difficulty retargeted:** Sat at `1.0` until block `32256` (30 Dec `2009`), when it jumped to `1.18`.
2. **The GPU Arms Race:** In July `2010`, *Laszlo Hanyecz* published an early **OpenCL** implementation. In October `2010`, *m0mchil* wrote a dedicated GPU hasher (`poclbm`).



Era 1: Friction & Collapse

The era of client-embedded mining ended between mid-`2009` and late `2010`.

1. **Difficulty retargeted:** Sat at `1.0` until block `32256` (30 Dec `2009`), when it jumped to `1.18`.
2. **The GPU Arms Race:** In July `2010`, *Laszlo Hanyecz* published an early **OpenCL** implementation. In October `2010`, *m0mchil* wrote a dedicated GPU hasher (`poclbm`).
3. **The hardware mismatch:**
   - A GPU could do tens of megahashes per second - exhausting the 32-bit nonce **in minutes**.
   - Running full Bitcoin GUI clients on mining rigs with a bunch of GPUs was unstable, bloated, and wasted memory.

Era 1: Friction & Collapse

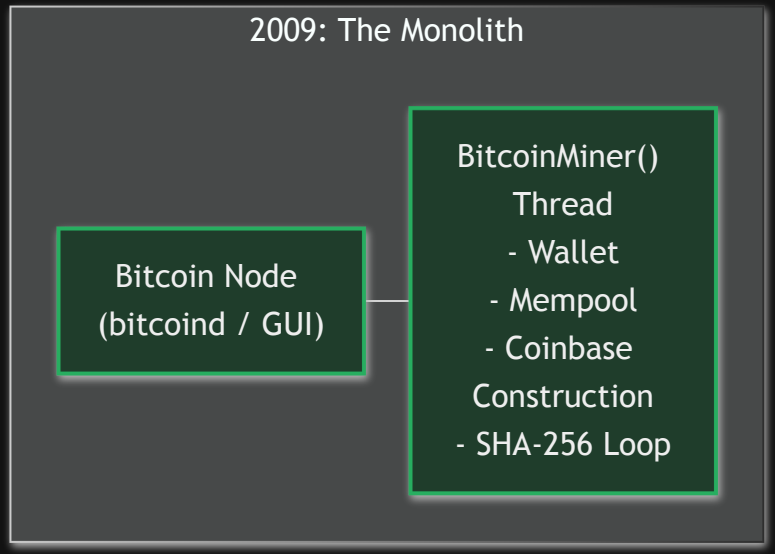
The era of client-embedded mining ended between mid-`2009` and late `2010`.

1. **Difficulty retargeted:** Sat at `1.0` until block `32256` (30 Dec `2009`), when it jumped to `1.18`.
2. **The GPU Arms Race:** In July `2010`, *Laszlo Hanyecz* published an early **OpenCL** implementation. In October `2010`, *m0mchil* wrote a dedicated GPU hasher (`poclbm`).
3. **The hardware mismatch:**
   - A GPU could do tens of megahashes per second - exhausting the 32-bit nonce **in minutes**.
   - Running full Bitcoin GUI clients on mining rigs with a bunch of GPUs was unstable, bloated, and wasted memory.
4. **Satoshi's plea on BitcoinTalk (Dec 2009):**

> "We should have a gentleman's agreement to postpone the GPU arms race as long as we can for the good of the network."
> – Satoshi Nakamoto

Evolution Tree: Stage 1 (2009)

The beginning was a monolith.



● active

There is no protocol. The miner and the node share memory, pointers, and disk.

Era 2: The Miner Leaves the Node

2010 - 2012

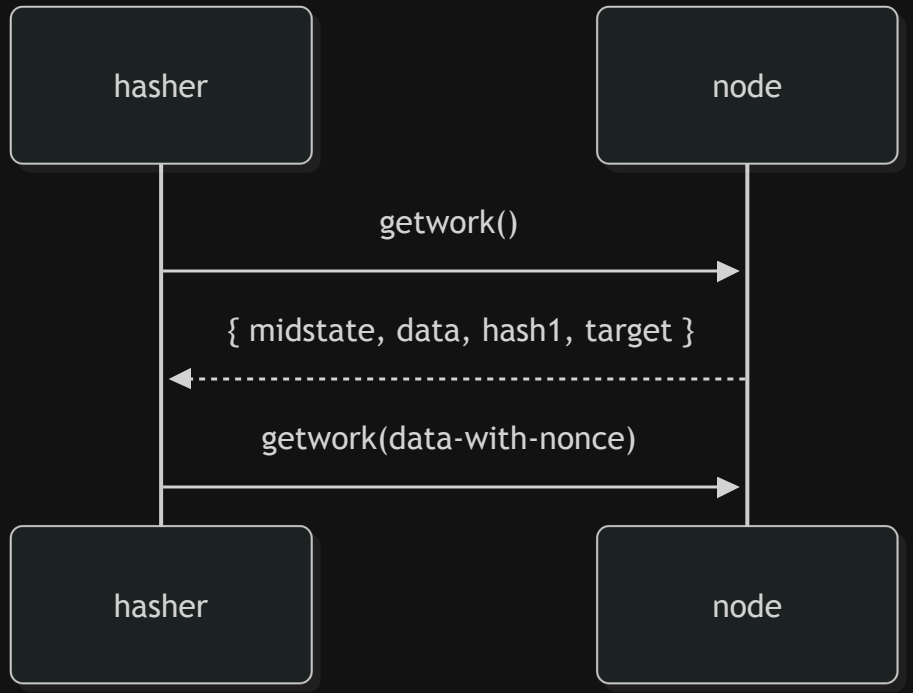
October 2010: `getwork` creates a wire boundary

On 1 October, *m0mchil* posted **"[PATCH] JSON-RPC getwork"**: external RPC clients ask for unsolved block data and submit solutions. *Satoshi* Nov 2010:

> "server farms can run a single Bitcoin node and the rest only run getwork clients."
> – Satoshi Nakamoto (Nov 2010)

- First tagged release with getwork: **Bitcoin v0.3.17** (25 Nov `2010`).
- External hashers follow: `poclbm`, `cpuminer`, `cgminer`, DiabloMiner.

`getwork`: a tiny job, over HTTP



`getwork`: a tiny job, over HTTP

Field	Purpose
`midstate`	SHA-256 state after the first 64-byte block
`data`	128 bytes: block header plus SHA-256 padding
`hash1`	second-hash buffer
`target`	share / network target

Pools pay for partial work

Slush's pool, Nov 2010: credit for partial work.

- **pushpool** backends handed out work at scale.
- **BitPenny** PPS: predictable payments, operator carries risk.
- Farm proxies point many GPUs at one endpoint.

The pool became both a coordinator of hashpower ***and an accountant for it***.

Friction: polling, stale work, and a silent authority shift

Early hashers polled HTTP every few seconds; long polling was added later.

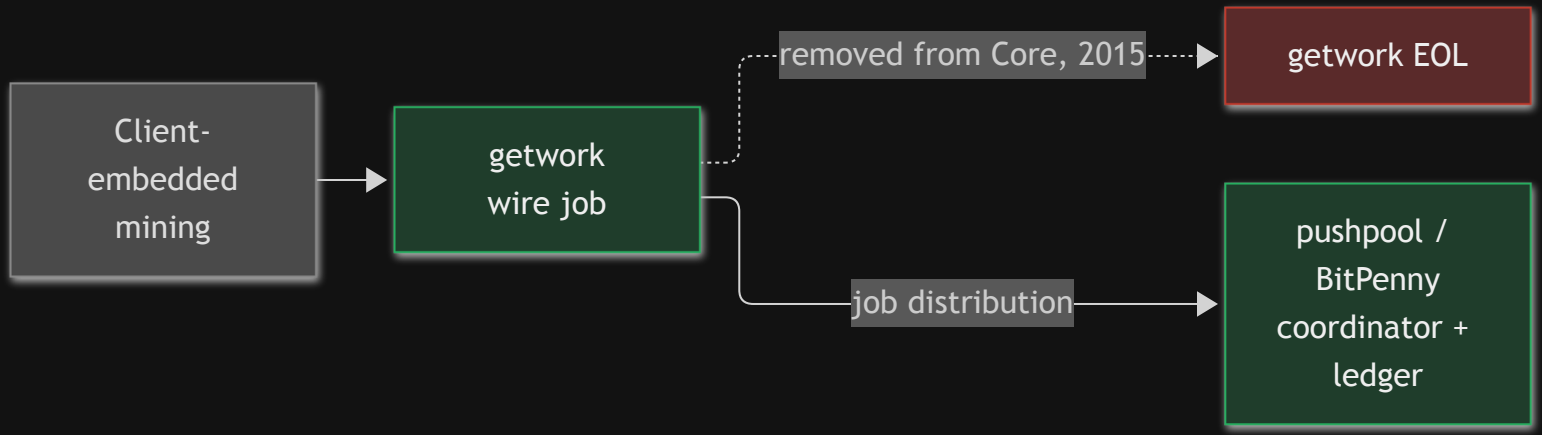
What a submitted share *did not* contain:

```
hasher submits: nTime + nNonce
node reconstructs: coinbase + Merkle root + full block
```

The node's stored extra nonce rebuilt the coinbase on submission.

– whoever operated the node, decided what was being mined

Tree update: `getwork` and its orbit



● active | ● historical / dormant | ● removed from Core

Era 3: Template construction back to the miner

2012

`getblocktemplate`: the node returns a block template

BIP 22/BIP 23:

> "Instead of sending a simple block header for hashing, the entire block structure is sent, and left to the miner to (optionally) customize and assemble."
> – BIP 22

1. The node could no longer efficiently generate individual getwork jobs at pool scale.
2. Pooled mining had become "too centralized on pools."
   (BIP 23 motivation)

Eloipool and libblkmaker were early implementations; Bitcoin Core had GBT by **v0.9.0**.

`getblocktemplate`: the node describes a candidate block

```
node -- template ⟶ work generator / pool
     previous block, target, time, version,
     transactions, fees, dependencies,
     coinbase value / auxiliary data, mutability rules

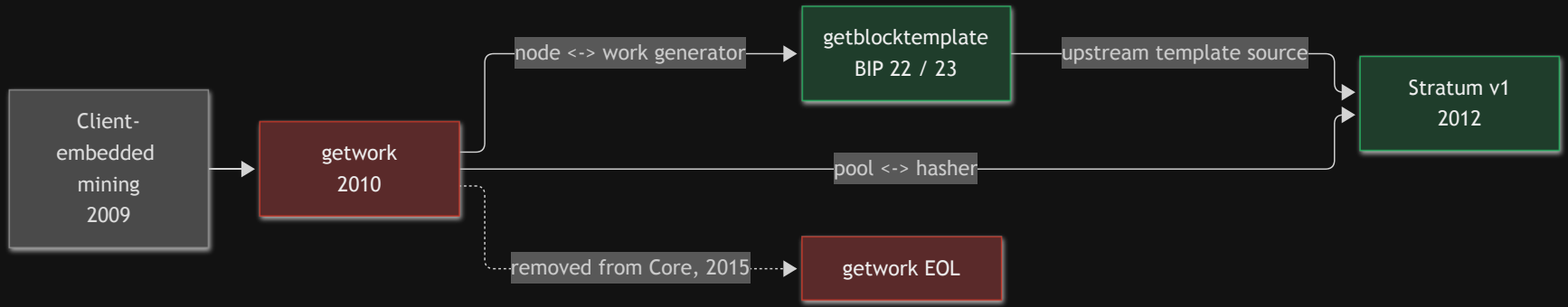
work generator -- submitblock(full block) ⟶ node
```

GBT: node-to-pool only, never device-edge

GBT made template construction possible for miners - but pools used it only **upstream**, between their node and their own server.

At the ASIC edge a full template was too costly; **Stratum v1** arrived for the pool-to-device path at the same moment.

Tree update: two descendants of `getwork`



● active | ● historical / dormant | ● removed from Core

Era 4: Stratum v1 becomes the device-edge default

2011 - present

Why Stratum v1

Slush's 2011 overlay proposal anticipated the ASIC problem:

> "ASIC miners are probably coming at the end of 2012… we definitely need some solution that scales to tera-hashes per second per pool user."
> – Slush (2012, stratum mining spec)

`getwork` : **hasher** drives HTTP requests. Stratum reversed it:

- one persistent TCP connection;
- server-pushed jobs;
- larger extranonce space.

Stratum v1: a pool constructs the job

```
hasher -- mining.subscribe / mining.authorize ⟶ pool
hasher ⟵——————— mining.set_difficulty ——————— pool
hasher ⟵————————— mining.notify ————————————— pool
hasher -- mining.submit(job, extranonce2, time, nonce) ⟶ pool
```

`mining.notify` : job ID, previous hash, version, bits, time, and the coinbase split:

```
coinbase = coinb1 | extranonce1 | extranonce2 | coinb2
                                  ^ device changes this
```

The nonce space grows again

```
coinbase = coinb1 | extranonce1 | extranonce2 | coinb2
                                  ^ device changes this

coinbase → txid → Merkle root → 80-byte header → 2^32 nonces
```

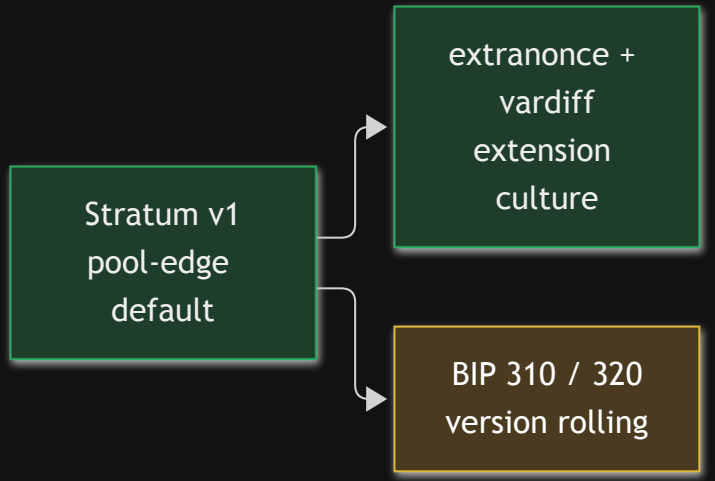
- Search space scales with `extranonce2` length.
- Per-worker share difficulty (**vardiff**), independent of network difficulty.
- Informal protocol, extension-driven; **BIP 310** later proposed version-rolling.

The cost of Stratum's efficiency

Restored the getwork authority model at the pool:

question	Stratum v1
Who builds the template?	Pool
Who chooses transactions?	Pool
Who constructs the coinbase?	Pool, with a worker-specific hole
Who accounts for shares?	Pool

Tree update: v1 becomes the device-edge default



● active | ● drafted, never (yet) standardized

Era 5: Pool alternatives and payouts

2011 - 2018

Payout schemes

Proportional payouts are vulnerable to **pool hopping** (Rosenfeld 2011).

scheme	trade-off	deployed
PPS	operator absorbs variance	**BitPenny** (first); **AntPool** (`2.5%` fee)
score-based	newer shares count more	*Slush's* pool (`score += exp(t/C)` , `C=300s`)
PPLNS	last *N* shares; variance on miners	**P2Pool** (by construction)
FPPS	pays the fee proxy for an average block	**BTC.com** (created it)

P2Pool: pool mining without a pool operator

P2Pool replaces the operator with a peer-to-peer **sharechain**.

```
Bitcoin blocks:  ————————o————————————————o————————
P2Pool shares:   o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o
                  ~30 seconds; about 20x Bitcoin's cadence
```

- A share at the Bitcoin target is a real block.
- The sharechain tracks recent work: PPLNS eligibility.
- Found blocks pay recent share owners directly in the coinbase.

Why P2Pool did not become the dominant pool

Same design, high cost:

- fast shares make stale shares economically significant;
- full node + P2Pool node per participant;
- many tiny coinbase payouts hit dust limits;
- miners must stay through the payout window.

– decentralized pool accounting was possible; not cheap or simple at scale

AuxPoW: one proof of work, more than one chain

Merged mining: a parent-chain proof of work validates an auxiliary-chain block.

```
parent coinbase commits aux-chain Merkle root
              v
parent block header meets aux-chain target
              v
AuxPoW carries header + Merkle proofs to the aux chain
```

Namecoin marker: `fa be 6d 6d` in the parent coinbase.

ASICs make protocol details physical

ASICs exhaust the `32`-bit header nonce in far below a second; the response is more mutable header space:

- **ASICBoost**: controlled variation of hashed work.
- **BIP 310**: version-rolling mask via `mining.configure`.
- **BIP 320**: version bits `13` - `28` (`0x1fffe000`) reserved.

ASIC firmware embedded Stratum v1 at the device edge; it persists as an installed base.

Tree update: the dominant wire and its side branches



● active | ● historical / dormant | ● pre-mainnet

Era 6: BetterHash, Stratum V2, DATUM, P2Poolv2

2018 - 2026

BetterHash and Stratum V2

Matt Corallo, 2018, BetterHash draft:

```
Work Protocol:   node ──────────→ miner / hardware
Pool Protocol:   miner ←────────→ pool
                 shares, identity, payout information
```

Never a numbered BIP or shipped implementation.

The work channel lets the miner build a template; the pool channels reduce variance **without** choosing block contents.

Stratum V2: a rewrite, not v1 with encryption

Stratum V2: binary framing, ordered transport, Noise-encrypted sessions.

1. **Mining Protocol** - job delivery to devices.
2. **Job Declaration Protocol (JDP)** - miner declares a template.
3. **Template Distribution Protocol (TDP)** - replaces `getblocktemplate`.

Standard jobs stay header-only; old hardware hashes them while a template provider runs a full node elsewhere.

Job Declaration Protocol (JDP)

```
job declarator -- DeclareMiningJob(txids, txdata?) ⟶ pool
pool -- SetCustomMiningJob ⟶ hasher / proxy / devices
```

Staged: disabled, client-push, client-declared. Optimistic mining avoids a new round-trip.

But payout trust stays with the pool:

– Job Declaration moves transaction selection toward miners (declarators); it does not remove the need to trust a pool for payouts (JDP spec summary)

DATUM: decentralized templates in production

OCEAN's DATUM gateway uses three separate links:

```
gateway -- RPC / GBT ⟶ miner's own Bitcoin node
gateway -- Stratum v1 + version rolling ⟶ ASICs
gateway -- encrypted DATUM protocol ⟶ OCEAN pool
```

> "The real miner is always whoever is running the Bitcoin node. With DATUM, that's not the pool."
>
> – DATUM, OCEAN

The DATUM specification is not public; blinding and validation should be treated as roadmap.

P2Poolv2: decentralized share accounting

P2Poolv2 keeps P2Pool's goal and rebuilds the sharechain:

- DAG of shares with uncles (reduces orphans);
- a share market for trading;
- large miners get direct coinbase outputs.

SV2 and DATUM handle work coordination; P2Poolv2 handles share accounting, so they share the same stack:

– both keep payout accounting with a centralised organisation; P2Poolv2 tracks accounting on the sharechain instead (p2poolv2 wiki, comparison summary)

Still in development: signet/testnet4.

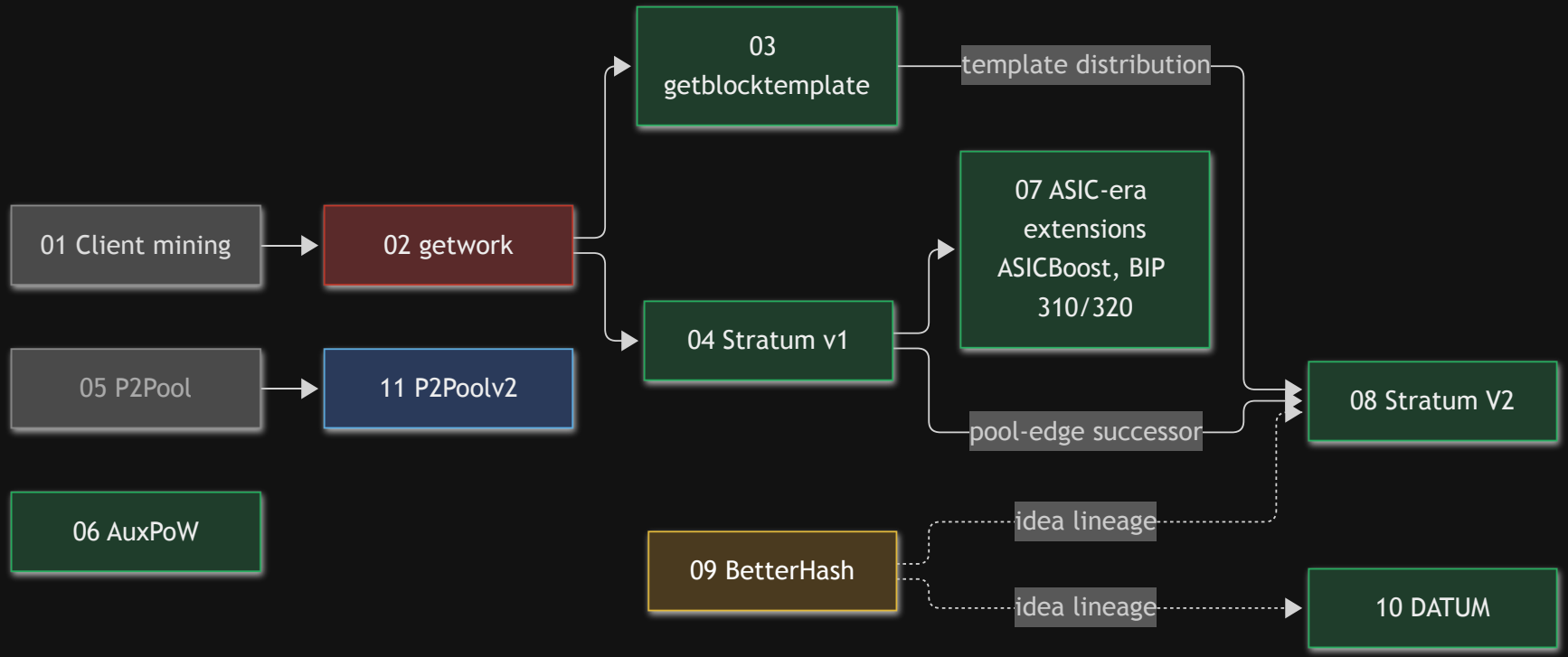
Three answers to the same 2012 question

strategy	who builds the template?	who accounts for shares?	spec status
Stratum V2 + JDP	miner / job declarator	pool	published spec
DATUM + TIDES	miner using own node	pool	spec unpublished
P2Poolv2	peer-to-peer miners	sharechain participants	in development

Two shifts:

1. From **who writes the block** to **who selects transactions**.
2. From template control to **who records and settles the work**.

Mining Protocols history tree



● active | ● historical / dormant | ● removed from Core | ● drafted, not (yet) standardized | ● pre-mainnet

Questions & Discussion

Thank you!

Presentation slides and research archive



https://johnnyasantos.com/presentations/2026-10-15-tabconf

Johnny Santos

Soli Deo Gloria