Why Nodes Get Stuck, and How to Tell If Yours Is

A node whose height stops moving is either perfectly healthy or badly stuck, and from the outside those look identical. Here is the number that separates them, and what to do about each. All times are UTC.

AUGUST 19, 2026 · 8 MIN READ · EASYBTX.COM/RESEARCH/WHY-NODES-GET-STUCK

ABSTRACT

A still block height means two opposite things on BTX, and telling them apart is one number. What we learned running an explorer through a week of network incidents, and the free tool we built from it for every operator.

All times in this article are UTC. Every number is a measurement from production nodes, and each one is timestamped in the text.

Two nodes, same symptom, opposite problems

Here are two BTX nodes. Both have stopped advancing. Both write the same warning to the log once a minute. Both show a healthy peer list.

The first one is fine. The network has not produced a signed block for an hour, so there is nothing for it to add. It is waiting, exactly as designed, along with everyone else.

The second one is in trouble. Signed blocks exist, other nodes have them, and this node is not collecting them. Left alone it will stay behind for hours or days.

From the outside those look the same. That is the whole problem, and getting it wrong is expensive in both directions. Treat waiting as a failure and you restart a healthy node for nothing, throwing away the peers that were about to feed it. Treat a real stall as waiting and you lose the afternoon.

The number that separates them

Your node already knows the answer and reports it. BTX nodes that follow the chain by signed confirmations, which is most nodes, track a **signed frontier**: the highest block the network's attestor has actually signed.

```
btx-cli getmatmulattestedtip
```

Inside the reply is `signed_frontier.blocks_behind`.

- `blocks_behind` **is 0.** You are at the frontier. Nothing newer has been signed, so there is nothing to fetch, and no amount of peer dialling will produce a block that does not exist yet. The correct action is none.
- `blocks_behind` **is above 0.** Signed history exists that you have not collected. This is your stall, and it is fixable.

That single number is the difference between "the network is quiet" and "my node is starving", and until this week nothing we shipped could read it.

How we found out, in public

On 19 August at about 11:50 the network's attestor went offline. Blocks kept being mined, because nodes with a qualified graphics chip can validate the proof themselves and do not need anyone's signature. Nodes that follow signatures, which is the majority, stopped where they were.

At 12:03 our explorer's own audit flagged it as a stall. It was not one. Our node sat at height 194,160 with `signed_frontier` also at 194,160 and `blocks_behind` at 0. It was exactly where the network was. The header for 194,161 had arrived and our node was correctly refusing to connect it, because nobody had signed it.

Meanwhile a pool operator with three graphics-card nodes read 194,203 and reported our explorer as stuck. Both readings were correct. The nodes were following different rules, and neither was broken.

Two hours later the attestor was still quiet. Five of our peers sat at exactly our height. Two were ahead on unsigned work. Every watchdog on the network that could not read the frontier had been reporting a stall that entire time, and dialling peers that had nothing to give.

That is the failure mode worth naming. A watchdog that warns wrongly does not just waste effort. It teaches its operator to ignore warnings, which is how the next real outage gets missed.

What a good watchdog does, and refuses to do

Running an explorer through a week of network incidents produced a short list of rules, each of them paid for.

Read the frontier before you speak. Everything above depends on it.

Dial peers. Never restart the node. When a node genuinely trails the frontier, the fix is attaching a peer that can serve it. Adding one over RPC works immediately and counts as a manual connection, which matters for reasons in the next section. On 16 August a three hour production stall cleared 21 seconds after that handshake. A restart, by contrast, throws away the peer set that is usually the only source of confirmations, and an unclean stop has damaged snapshot data before. Crash recovery is your service manager's job. Reacting to a stall is a different job.

Never touch the signer configuration. Changing which keys a node trusts, on a chain it has already validated, is a one way door. The node refuses to start, and the repair it suggests deletes state

before failing the same check again. On a pruned node there is no way back from that except starting over.

Respect the limits. A signer serves only the last 16 blocks and bans a peer for 24 hours after 32 ignored requests. Historical requests belong on archive nodes. A watchdog that hammers a signer turns a slow recovery into a banned one.

The trap that starves healthy looking nodes

This one is worth knowing even if you never run a watchdog.

A node following signed confirmations does not ask every peer for them. It only asks peers that are **manual**, meaning added with `addnode`, or **noban**. A peer that is merely connected is never asked for anything at all.

So a node can show twenty peers, several of them full archives, and still receive nothing. The peer list looks perfect. The node starves anyway.

Both lines are needed for each archive:

```
addnode=<archive>:19335
whitelist=in,out,noban=<archive-ip>
```

The `in,out` part matters too. A bare whitelist applies to incoming connections only, and the connection `addnode` creates is outgoing. This single detail was behind a week of stalls before anyone spotted it, ours included.

The kinds of node on this network, and which one you are

Much of the confusion in public discussion comes from people comparing readings taken by different kinds of node. There are four, and they are supposed to disagree.

Consensus nodes with a qualified graphics chip recompute the proof themselves. They need nobody's signature, so they keep advancing when the attestor is quiet. Their height is the proof of work tip.

Trusted mirrors, which is most nodes, follow the signed frontier and stop where the signatures stop. Their height is the attested tip. Explorers and wallet backends are built this way deliberately, because showing a transaction as confirmed and then having it vanish is the worst thing a wallet data source can do.

Archives are mirrors that also serve the signed history to others. They are the scarcest class on the network and the reason a node that falls behind can recover at all.

Signing nodes produce the signatures the other two depend on. Today there is one.

That last line is the whole story behind every network wide pause this month. On 19 August the proof of work tip kept climbing while the attested frontier stayed frozen at 194,160, and every mirror on the

network parked there, correctly. It was a one of one dependency doing what a one of one dependency does when it goes offline.

The fix is not a better watchdog. It is more signing nodes, so a quorum of several can attest instead of one. The network calls this multi signer, or M of N, attestation, and it is on the roadmap. Until it lands, a frozen attested tip beside a moving proof of work tip is expected behaviour rather than a fault in your node, and it is worth knowing that before you go looking for one.

Two practical consequences while the gap exists. If you run a mirror, a still height during a signer outage is not yours to fix and restarting will not help. If you mine, blocks built above a frozen frontier are unsigned, and the first signature after the signer returns decides whether they survive. Six such blocks were lost in one earlier episode. Nobody can promise you a different outcome, and anyone who does is guessing.

The Node Guardian

We packaged all of that into a small script, and it is free for anyone running a BTX node.

Every few minutes it reads your node and decides which of the two states you are in. When you are genuinely behind it dials the archive peers, at most once every ten minutes. When the network is merely waiting it says so and does nothing. It writes one JSON file with the full picture: heights, the signed frontier and your lag, your peer and archive census, which peers actually feed you confirmations, which ones you serve, disk, and how long each clock has been still.

It will never restart your node, never touch your data or keys, and never pester another operator's node past the published limits.

It runs on any machine with bash, python3 and your btx-cli. There is nothing to install and no account of any kind. The page at easybtx.com/guardian has the install line and the systemd timer, and the README covers the configuration.

If you run our software you already have this. The same logic ships inside the easyNode app and the BTX Keeper. The standalone script exists for everyone else: the pools, the explorers, the exchanges, the mining nodes, the people whose infrastructure we will never see.

Why we are giving it away

The honest reason is self interest, of the ordinary kind.

Our explorer serves wallet data, and it can only be as reliable as the network it reads. During last week's incidents an independent census found exactly one reachable node serving the full confirmation history. When that one node went quiet, everything downstream of it stopped. We could not fix that by making our own node better. The only fix is more nodes that stay healthy, run by people we have never met.

So every operator who catches a stall early, or correctly ignores a network pause instead of restarting a healthy node in frustration, makes the chain steadier for everyone including us. There is nothing to buy here, no token, and nothing earned by running it. That is not modesty, it is the design.

A week of incidents taught us things no documentation had. Keeping that to ourselves would have been the expensive choice.

Sources: production node RPC and logs from the api.btxscan.io explorer node, the public engineering thread on the BTX repository, and field reports from other operators running graphics card nodes and pools. Measurements timestamped in the text.

Frequently asked questions

My node's block height has not moved for an hour. Is it broken?

Probably not. Check how far you trail the signed frontier by running getmatmulattestedtip and reading signed_frontier.blocks_behind. If it is 0 you are exactly where the network is and there is nothing to fix. If it is above 0 there is signed history you have not collected yet, and dialling archive peers is the remedy.

Why does my log say 'matmul trusted mirror stall' every minute when nothing is wrong?

That line fires whenever the next block is not yet attested, which includes the completely normal case where the attestor has simply not signed anything newer. On its own it does not mean your node has a problem. The blocks_behind number is what tells you.

Should a watchdog restart a stuck node?

No. A restart discards the peer set that is usually your only source of signed confirmations, and an unclean stop has damaged node data before. Crash recovery belongs to your service manager. Reacting to a stall is a different job, and dialling peers is the action that actually helps.

My node has plenty of peers but still gets no confirmations. Why?

A trusted mirror only asks peers that are manual or noban for attestations and block downloads. A peer that is merely connected is never asked for anything. You need both an addnode line and a whitelist=in,out,noban line for each archive.

What is the BTX Node Guardian?

A small free script that makes this distinction for you every few minutes, dials archive peers when you are genuinely behind, stays quiet when the network is merely waiting, and writes one JSON file with your node's full health picture. It never restarts your node.

My node shows a different height than someone else's. Who is right?

Probably both. A node with a qualified graphics chip validates the proof itself and follows the proof of work tip. A trusted mirror follows the signed frontier and stops where the signatures stop. During a signer outage those two numbers diverge by design. Ask which kind of node produced each number before assuming either is broken.

Why is there only one signing node, and what happens when it goes offline?

Today the network has a single attestor, which is a one of one dependency: when it pauses, every node that follows signatures parks where the signatures ended while proof of work keeps moving.

Multi signer attestation, also called M of N, is the network's own planned fix. Until it ships, a frozen attested tip during a signer outage is expected behaviour rather than a fault in your node.

Are blocks mined above the frozen frontier safe?

They are unsigned until the attestor returns, and the first signature after it comes back decides which branch survives. In one earlier episode six such blocks were lost. Anyone promising you a particular outcome is guessing.

Do I need it if I use the easyBTX apps?

No. The same logic ships inside the easyNode app and the BTX Keeper. The standalone script is for nodes that are not ours: pools, explorers, exchanges, mining nodes.
