# The Day BTX Halted, a MacBook Kept the Chain's Memory. Field Notes.

*Most machines can no longer verify BTX's proof-of-work themselves. They follow the chain through small signed confirmations, and this August the entire network's supply of those confirmations rested on one reachable server. We put a second source on an ordinary MacBook, measured everything, and then the network had its worst day so far. The MacBook worked. What we learned shipped the same evening as easyNode 0.6.10 and Keeper mode.*

AUGUST 17, 2026 · 8 MIN READ · EASYBTX.COM/RESEARCH/ONE-MAC-ARCHIVE

ABSTRACT

On 17 August 2026 the BTX network stopped producing blocks for nine hours. Through that halt, a single MacBook served the signed confirmations the network depends on, answered 745 requests, found and fixed a bug that made every new node take hours to start, and crossed onto the emergency release within seconds. Here is the full story, with every number.

On the morning of 17 August 2026 we started an experiment: turn one ordinary MacBook (Apple silicon, M5 class) into a full trusted-mirror archive for the BTX network, measure everything, and see whether a laptop can matter to a chain. By that evening the network had halted for nine hours, recovered through an emergency release, and the answer was in: it can, and it did.

Everything below is measured, dated, and reproducible. The technical version of these findings is public on the BTX stability thread (PR #105).

## Why the network needed this

Since the MatMul v4.7 fork (our earlier article), verifying a block's proof-of-work takes a qualified GPU. Everyone else follows the chain through small signed confirmations, 208 bytes each. A signer only serves its most recent 16 blocks. Everything older must come from archive nodes that volunteer to keep and serve the history.

That morning we probed every archive address the network knew: 49 targets, one polite request each, checking the actual cryptographic signature in every reply. The result: 14 answered a connection, 7 completed a handshake, 6 advertised the archive service, 3 served genuine records, and exactly **one** served the deep history. The chain's entire memory, reachable through one machine. That is the gap the MacBook was built to fill.

## The build, and a surprise worth a hundredfold

The node was built from source in one morning, on a Mac with no Homebrew and no special setup. The surprise came during the first sync: the headers phase, which should take minutes, was projected to take four to five hours, with one CPU core pinned the whole time.

We profiled it. More than 99.9% of the time went into a single repeated ancestor walk inside the difficulty check, a walk whose length grows with chain height. We wrote a forty-line fix that caches exactly what the check needs, verified it produces bit-identical results, and measured again: the same phase finished in about one minute. Roughly a hundred times faster. This also explained a mystery in our own release notes, where an older version told users that headers "climbing and then dropping back is normal, it may do that several times." That was not normal. It was this bug resetting the sync. The fix has been offered upstream, and every fresh node on every platform gets faster if it lands.

## Serving through the halt

By mid-morning the node was synced, verified against two pinned block hashes, and serving. Demand found it immediately: within its first two minutes of life, before it held a single record, other nodes were already asking it for confirmations.

At 10:12 local time the network stopped. No new blocks for nine hours, the longest halt in the chain's life so far. The cause, in one sentence: the network's chain selection wedged on a self-signed losing branch, and the fix required an emergency core release, v0.33.3, with a hard line at block 191,714 that older software cannot cross.

Through all nine hours, the MacBook kept answering. By evening it had served **745 confirmation requests**, most of them during the halt, to seventeen peers including nodes running software from months ago. When v0.33.3 was tagged, we rebuilt at the release commit, stopped the node cleanly, swapped the binary, and started it again. The first new block connected within seconds. The Azure server that runs the network's explorer crossed the same line in about eighty seconds using the official build. The fix does what it says.

## The ten-gigabyte archive

One more result matters for everyone who wants to help without giving up a whole disk. We ran a second, pruned node, about 10 GB instead of the full chain, imported a deep-history confirmation into it, and asked for that record over the network. The pruned node served it, correctly signed, for a block whose data it had never held. The confirmation store is separate from block storage, and the whole chain's record set is about 40 MB.

That means a small node plus a small file equals a full-history archive. It is the entire technical basis of Keeper mode.

## What shipped the same evening

Everything above became product before midnight:

- **easyNode for BTX 0.6.10** (Mac, Apple silicon), carrying the official v0.33.3 engine. If you run 0.6.8 or 0.6.9, update now: older engines stop at block 191,713 and wait forever. First start takes minutes, not hours.
- **Keeper mode**, one switch in Settings: your Mac runs a small ~10 GB node that serves signed confirmations, pauses on battery, and stops cleanly if anything ever goes wrong. It never force-kills and never restarts itself.
- A one-command installer for the terminal crowd at easybtx.com/keeper, which now downloads a verified static engine in seconds. No Homebrew, no Xcode, no toolchain.
- The nodes directory gained an Archive column, so you can see who serves the chain's memory at any moment.

## The honest part

The confirmations counter resets to zero after the v0.33.3 update, for everyone, because the release changed the cryptographic context the records live under. Nothing about the chain itself was lost. The network's deep history now needs re-signing under the new context, a question we have put to the core team publicly.

And the same disclosure as always: there is no payment, no token, and no income anywhere in this. A Keeper gives the network uptime and gets recognition. On the day the network needed it most, one MacBook was enough to keep the chain's memory alive. Ten would make the next halt boring. That is the whole pitch.

---

## Frequently asked questions

### What is a signed confirmation (attestation)?

Since the MatMul v4.7 fork, verifying a block's proof-of-work takes a qualified GPU. Machines without one follow the chain by accepting a small signed record, 208 bytes, from the network's signing key instead of re-doing the work. Those records are what most nodes need to move forward at all. The whole chain's record set is only about 40 MB. The scarce thing is not storage, it is machines that are on and willing to serve.

### Does serving confirmations earn me anything?

No. There is no payment, no token, and no income here, and anyone who tells you otherwise is not speaking for this project. The reward is that the chain stays verifiable for people whose machines cannot verify it alone, and that this happened on your Mac.

### Is my Mac powerful enough to be a Keeper?

Any Apple-silicon Mac is far more than enough. Serving confirmations is a trickle: 208 bytes per record, rate-limited by the protocol. The one thing that matters is being on and reachable. Keeper mode pauses on battery and never blocks your display from sleeping.

### How much disk does Keeper mode use?

About 10 GB. A Keeper keeps recent blocks only, not the whole chain. We verified that a pruned node can still serve confirmations for old blocks it never held, because the confirmation store is separate from block storage.

### Why did my confirmations counter reset to zero after updating to 0.6.10?

The emergency v0.33.3 release changed the cryptographic context that confirmation records live under, so the network's record set started fresh. Nothing was lost from the chain itself. Counters begin again from zero for everyone.

### Can I stop being a Keeper whenever I want?

Yes. Flip the switch off, or uninstall with one command. The network simply has one fewer source until someone else turns one on. Nothing is locked in.

---