# **Factom Authority Node Operator Committees**

## **Core-, Technical and Code deployment Committee**

### Post-mortem report: September 2018 - [FIN-18-007](https://docs.google.com/spreadsheets/d/1xPfEjTETUulqO35aMjvEKMkGxroYwfatv6JYnCcuBMY/edit?usp=sharing)

##### Submitted date: 2018-10-02, V1.1

#### Description of events

##### Thursday (Time references in UTC -5)

This latest incident has exposed several failures that have cascaded in causing a multi-day outage. The incident began in the early morning hours (Texas time) of 27th September 2018. Multiple Factom Authority nodes experienced large memory spikes, and crashed as a result of this. In total there were held 6 Authority server elections spanning two blocks (160142, 160143); but the network seemed to recover after this.



Fig 1: The Initial 6 elections.

A host of unusual messages were observed propagating the network a while before the time the nodes started failing(Fig 2), and it is believed that nodes were not able to decode these properly, triggering garbage collection which increased the memory usage by a very high amount.

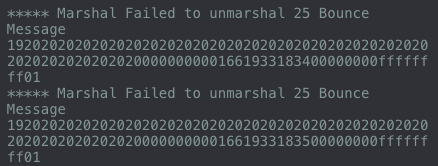


Fig 2: Test-messages associated with the failure.

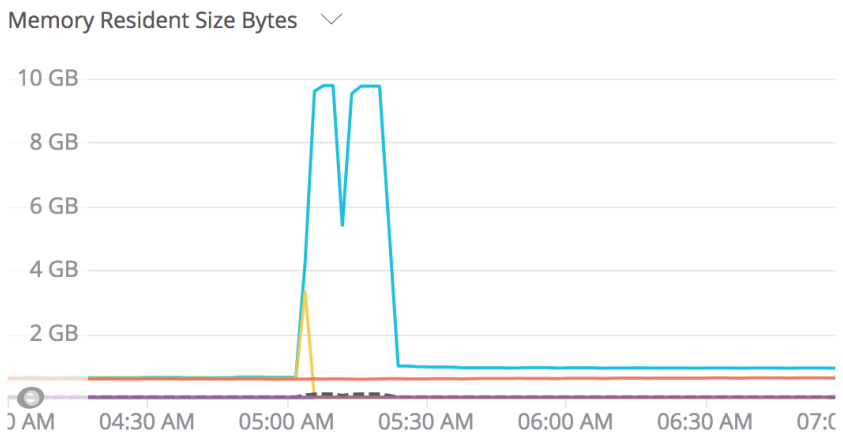


Fig 3: Memory spikes on two servers.

About 10 hours later the network began to falter again. It is unclear if these two events are related other than being close in time. New elections happened in starting at block 160177. The network stalled at block 160179. The network was restarted once or twice. Then we noticed the protocol had progressed without rebooting very slowly through blocks 160180 to 160184 where it stopped. During this time some of the blocks took longer than 1 hour for the federated servers to reach consensus and close the block.

While one of the blocks (160181) was progressing forward, a factoid transaction was included that had a time stamp roughly 88 minutes after the start of the block. Factom protects itself against duplicate transactions by two mechanisms. The first is to make sure no duplicate txids are included in the blockchain. This is only checked over a +- 1 hour period. Any transactions that are outside of this 2 hour window are considered invalid so they don’t need to be tracked. In block 160181 a FCT transaction was included in the blockchain when the transaction was outside the time window. The runtime checks did ensure no duplicate transaction was recorded.

The federated servers approved the transaction in real time as it passed all the rules. They approved it to be included in the block. When nodes who were not following by building the process list and instead were downloading blocks after they were created, the transaction appears to violate the 2 hour rule.

[This](https://explorer.factom.com/transactions/f96af063fffd90e261e45cbcdf34c3958cf54e2388fd3f8ef907dca903ea2f2e) was the transaction in question.

It is an otherwise valid transaction, but unpatched nodes will not consider it valid and reject block 160181. The patch developed early in the morning Friday Sep 28 labeled [**v5.4.3-fix1**](https://github.com/FactomProject/factomd/pull/584/files#diff-0934fa3059eb66c238a5b45a52edc5caR810) whitelisted this transaction.

A further complication is that even the loading of blocks from disk recheck the process rules. When reading 160181 from disk, nodes that had built the block and saved the block to disk still will not accept 160181, because the only context for validation is the block time on the block itself, and the transaction appears to be outside the allowed time window. Without the v5.4.3-fx1 patch, factomd will not process any block past 160180.

##### Friday

Overnight the Factom Authority Node Operators (ANOs) updated to **v5.4.3-fix1** and a restart of the network was attempted later on Friday. It succeeded in getting all upgraded nodes past the problematic block 160181, but the Federated servers did not begin building blocks.

Some of the federated servers had booted to block 160183 and some to 160184 (henceforth called 183 and 184). Almost all the Federated servers were at 183 and almost all the Audit servers would eventually get to 184.

Back to the two modes of operation mentioned before: process lists and block downloading. Followers and Audit servers will catch up with the latest blockchain by downloading blocks. Leaders do not do this. They will only catch up to the latest block by processing the constituent messages that create a process list which when iterated over will deterministically build a block.

The Federated servers needed to all get caught up to 184 in order to proceed forward with building the blockchain. The alternative was to back up to 183, however many nodes in the broader network would have progressed to 184. Any node that had progressed to 184 would have to delete their database and resync with the network. It was determined that this would have been a very difficult requirement to put on the exchanges and users. So a priority was placed on progressing to 184.

Friday was spent retrieving a complete process list from a follower node which had not been rebooted. These process list messages were attempted to be replayed over the network in order to get the Federated servers to move forward and save block 184. This would have not taken any effort on the part of the ANOs if it had worked.

The effort to simply update nodes with a “183” database to 184 did not work. Friday evening an announcement went out to the ANOs to use their best options to start their leaders with a database which contained block 184.

At this stage a [spreadsheet](https://docs.google.com/spreadsheets/d/1VSbmnW1ndmIdXHTyegFtC_SiIVj1CzFnxgURzj3tNCs/edit?usp=sharing) was created to detail the status of each Authority node. Specifically its blockheight and its perceived role (audit/federated server).

It is necessary for the Authority set to have a common blockheight in order for the network to restart. So as a next step, ANOs ensured that their blockheights were all at 160184 by either,

* Moving the authority identity to a backup-server that had reached 160184.
* Restarting the authority server without the authority identity, having it download block 160184 from the network as a follower, and then restarting it a second time with the authority identity in place.
* Copying the entire blockchain database from a node that included block 160184 (as far as we know this was not necessary as the above methods worked for everyone).

##### Saturday

After all the ANOs configured their Leaders with block 184, a network restart was attempted. It did not succeed. A curious effect was observed though. All the leaders had updated to 184, but after rebooting they were showing only block height 183 in their control panels, and were showing only booting to 183 in the log outputs. This was an unexpected bug, which was analyzed, and fixed on Saturday.

There were two aspects of the bug:

1. the part which changed the database
2. dealing with the corrupted database

1)

When booted without a fastboot file, the last block in the database is loaded, and is checked for signatures. There are no signatures for the last block (184), because signatures would naturally be kept in the next block. Since the node cannot find a signature for 184, it would ignore that block. The big bug was that the highest block counter gets overwritten to the 2nd highest block (183). The next time the node is booted it will act as if the highest saved block was 183, even though 184 was saved in the database.

The [fix](https://github.com/FactomProject/factomd/compare/d7da7c42fbc8a26b03c5194d020e0bd76d95e434...59de2fcf75fde92f0e65ef5fdb816c1b3a5e9199#diff-b97b7c8b73644eb78fd0ab2bc4b9edaaR198) for this was to consider blocks that are saved in the database as valid, as they wouldn’t have been saved if they were not validly signed.

When using the fastboot file, the time exposed to the network before finishing booting is much shorter. This explains the observation about the block rewinding happening far more often when booting without the fastboot file. When booting a node, it receives dbstates (completed blocks) from the network. These dbstates were used before the dbstates that were loading from disk. Since they originated from the network instead of the local disk, factomd thought they were “new”. Because of this, they were saved back to disk, setting the chainheads to 160181, 182, or 183. Dbstate 184 did not have signatures, as signatures are not saved to disk until the next block is saved, so requesting 184 would yield an unsigned block. This block was considered invalid and dropped. Subsequently, the block 184 loaded from disk (which skips validation), was dropped as 184 had already been examined and rejected. This left the last saved block as 183, setting the block counter to 183 after a reboot.

As an aside, this bug likely contributed to some of the chain head corruption problems that have been affecting the network for several months.

2)

The Federated servers had corrupted block height counters, and needed to be corrected. This patch checked if the height was set to 183 and checked if 184 had been previously saved to the database. If both of those were true, it would force the saved height to 184.

These two patches were released as [**v5.4.3-fix2**](https://github.com/FactomProject/factomd/compare/d7da7c42fbc8a26b03c5194d020e0bd76d95e434...59de2fcf75fde92f0e65ef5fdb816c1b3a5e9199#diff-f6be34387a141f652301c16cf7a5e36cR954). An alert to the ANOs to update the factomd containers went out Saturday night.

##### Sunday

The network was restarted and the patches fixed the identified bugs, but unfortunately the federated servers were unable to form a consensus to continue building blocks.

Further investigation commenced at this stage, and another related issue was identified; one ANO had an error in it’s nodes configuration files. This was rectified and deemed not a issue per se; as their federated servers should have been faulted out of the authority set if the network was able to progress and build blocks.

At this stage two additional things were being looked in to:

1. 6 federated servers were running in docker-containers not properly named (i.e had names other than “factomd”), which resulted in these not being restarted during the synchronized restart attempts, and;
2. Older process list items were identified to be interfering with new process lists being built.

Item 2 above turned out to be the main culprit at this stage; and after deploying a [fix](https://github.com/FactomProject/factomd/pull/588/files) to some nodes (Factom Inc.’s) the network was successfully restarted and new blocks started being built right away.

The Factom mainnet was stalled for **3 days and 15 hours;** which so far is the longest outage the network has experienced post M3.

Network stalled: 2018-09-27 - 16:34 UTC

Operations restored: 2018-10-01 - 07:43 UTC

As a consequence of the patching, nodes outside the authority set (“followers”) will need to update a patched version to get past block 160180. Release is pending.

#### Root causes:

The root cause that triggered the initial memory spike has been identified. The reason for the stall on Thursday is still unknown. The root causes of the subsequent issues were detailed above.

Origin of the bounce messages: bounce messages are messages that can be triggered by human intervention. It appears that garbage data from an overflow was being parsed and the garbage happened to decode to an invalid bounce message. The root cause is probably other malformed messages that happened to be displayed as bounce messages. More investigation is needed still.

Another takeaway is that if the network is struggling to form consensus, and is taking too long to build a block, the servers are in danger of building a block that includes an FCT or EC transaction that violates the rules the same way as the Thursday bug. In order to prevent another non-reversible release, the network should be rebooted if it is struggling along and appears to be taking more than an hour to finish the block. This is true if the network is being slow while running or during a reboot. A patch to prevent the exceptions from occurring is needed.

#### What could have been done to resolve the issue faster

Debugging of network issues is a time consuming and tedious process. Each network stall is accompanied by a (generally) multi-hour diagnosis process. Understanding why the network has stalled is important to prevent the same errors from occuring again. Sometimes the cause of the stall cannot be determined from examining the stalled network, and a reboot is necessary without discovering the cause of the stall. Attempts to discover the cause are generally made.

In the case of this series of problems, the network would not start, so debugging, discovering, correcting, and deploying of the successive problems provided for cumulative downtime.

There was no clear path how to create the v5.4.3-fix1 patch. We had to discuss which approach would be the best (cleanest, acceptable by exchanges...). Having a standard procedure for emergency patching would speed up the process and make it more robust.

#### Next Steps

A new version of factomd will be released which will keep up with the network. Since this upgrade is required by followers to keep up, it will [increment the major version number](https://factomize.com/forums/threads/versioning-scheme-used-in-factomd.591/).

The next release will be v6.0.0. It will be based on v5.4.3, the latest stable release, and include the patches needed to get the network running again.

Note, there was an aborted v5.4.4 which was partially released. Temporarily abandoning that release will cause some problems:

* golang 1.11 support was slated for 5.4.4, but was not supported in 5.4.3, so 6.0.0 will still require golang 1.10.x
* As part of the abandoning of 5.4.4, a force push was made to the master branch. If any developers or users have pulled master in the past 2 weeks, they may need to do a **git reset origin/master --hard** to get back on track.
* A non-reversible change to the testnet was made testing 5.4.4. Since 5.4.3 does not recognize that change, 6.0.0 is not compatible with the community testnet.

ANOs are already running compatible software. They are currently not running the last 5.4.3-fix3 patch to throw away old process list items, but upgrading to use that still requires some discussion.

#### Actions to be considered to improve the Factom network and incident processes:

* Parsing of malformed bounce messages (or any other kind) should not generate such high spike of memory consumption (Go garbage collection). Software should be fixed after emergency patch.
* Fix factomd to be able to smoothly handle transactions on long blocks.
* Create a standard procedure for emergency patching.
* Deploy drafted incident procedure to improve communication during stall, including a steady cadence of updates from the core-committee (4-6 hours?), which are also made available to the broader community.
* Further discussion of decentralization aspects related to having one team (Factom Inc.) able to debug and restart the network is necessary.
* Work with community to provide a status-page hosted on the protocol website in development.