**After-Action Report of 2019-04-12 Factom Network Pause**

On April 12, 2019 - the Factom network stopped forward progress of the blockchain. Factom being a distributed system that optimizes for safety over liveness, this behavior is the normal and expected when the system detects anomalies. The network was under diagnosis for about 6 hours. The new network\_utility for triggering a coordinated restart developed by Steven Masley, worked very well - and this being the first time it was used on mainnet proved itself reliable under pressure.

**Root cause:**

The root cause is still in the process of being fully debugged, but some promising leads have been found. The symptoms that were being observed during the incident have been replicated in a test environment, which is generally 80% of the battle in fixing a problem. There are two fixes that are being explored to remedy the problem.

**Environmental Conditions:**

There were two anomalous events that were happening simultaneously. It is still unclear if the root cause was related to one, or the other, or a combination of the two.

Anomaly 1: Network Upgrade

The Factom network was undergoing an upgrade. This was a significant upgrade, which had necessitated a slow, phased upgrade process. The upgrade was from 6.2.0 to 6.2.2. The upgrade is significant because it contained more updates than github is capable of displaying. <https://github.com/FactomProject/factomd/compare/FD-820_release_candidate_butter...FD-824_release_candidate_kraft>

The upgrade had been undergoing various tests for several weeks before the point of the liveness loss. The week prior, the community testnet was driven to a high transaction load and the network passed the testing.

Anomaly 2: Increased Mainnet Load

There was a number of spikes in incoming traffic coming from Factom, Inc customers on legacy infrastructure. This occurred a few times leading up the network pause a few days before, the day before, and the night of the occurrence.

The solution to restore forward progress, was to ask for the few ANOs who had upgraded to 6.2.2 to downgrade back to 6.2.0. The synchronized restart resolved the issue and allowed the blockchain to progress.

**Resolution Process:**

12:20 CDT(5:20 UTC)

The network appeared to be stalled at 187888, with some at nodes at 187887 still. There were 6 ANOs running 6.2.2, with the remaining running 6.2.0.

The stall happened during a spike of traffic from a single source. This traffic further complicated the restart since the network was under load.

The new network\_utility restart was used to restart the network, and it was ran from the older Docker Swarm Manager host(pre elastic IP).

1:55 CDT(6:55 UTC)

The first restart was attempted:

*Full restart in 28.190954482s*

*Reachable: 98*

*Unreachable: 62*

*84 Nodes stopped*

*84 Nodes started*

The restart reached enough nodes to be a network restart, but the network did not recover from the stall.

The nodes seemed to be taking extra time to sync after the restart, in general. It appeared ( and was rationalized) that it may be struggling due to the load it was under at the time of the stall.

The load was reduced on the net, to reduce the incoming entries. This was hoped that it would allow the network to successfully restart.

4:00 CDT, (9:00 UTC)

The changes were implemented and deployed to reduce the load, and a new restart was attempted:

*Full restart in 28.145576651s*

*Reachable: 98*

*Unreachable: 62*

*81 Nodes stopped*

*81 Nodes started*

After the ignore period, the nodes were syncing, and creating new minutes indicating forward progress. After some time, the 6.2.0 nodes seemed to be stuck at minute 5 for the next block. The 6.2.2 nodes were behind by varying amounts, and not syncing with the rest of the network. Although the network seemed to be progressing at an appropriate speed at initially, the network slowed to a point where it was not acceptable. Eventually it was determined that the network did not recover with this reduced load restart.

After some discussion it was decided to ask the few ANOs who had upgraded to 6.2.2 to downgrade to 6.2.0, and run another restart. The emergency alert system was employed to alert the upgraded ANOs who responded in an appropriately quick fashion.

6:00 AM CDT(11:00 UTC)

At this time, the downgrades were finished, and another restart was processed:

*Full restart in 27.505862699s*

*Reachable: 98*

*Unreachable: 62*

*83 Nodes stopped*

*83 Nodes started*

6:25 CDT(11:26 UTC)

After the ignore period, the nodes synced across the network and processed the DBSigs. The minutes progressed through without a halt, and it started to process blocks onto the chain. It was apparent within a few moments that the network had recovered from the stall on this restart.

**Replication Process:**

80% of the difficulty in debugging a problem is replicating the problem in a controlled environment. The symptoms that were observed during the attempted restarts were replicated in the simulator. This branch contains the test:

<https://github.com/FactomProject/factomd/commits/FD-957_induce_stall>

To replicate locally with a fresh **~/.factom/m2/local-database** folder

go test -v -run TestSetupBootWithoutDB

go test -v -run TestBootWithDB

===SummaryStart===

Time: 1555706105 2019-04-19 15:35:05 Elapsed time:6m50.145747097s

P= 0 PL= 2 US= 0 Z= 2 Ht: 4 New Chains: 0 New Entries: 0 sum: 0 Total Entries: 0 diff 0 Total EBs: 0 FCT: 0

Node ID Resets Drop Delay DB PL Min DBState(ask/rply/drop/apply) Msg Resend Expire Fct/EC/E API:Fct/EC/E tps t/i DBH-:-M BH

0 f FNode0[455b7b] L\_\_\_vm05 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 4/ 3 0/0/0/0 71/31/55/45 0 0 667/116/426 41/0/0 3.24/1.51 5-:-3 - 0004df

1 FNode01[1570f8] L\_\_\_vm06 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 3/ 3 0/0/0/0 79/59/43/14 0 0 668/121/436 90/0/0 3.31/0.55 5-:-3 - 0004df

2 FNode02[8da6ed] L\_\_\_vm01 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 4/ 3 0/0/0/0 84/54/56/66 0 0 667/116/425 59/0/0 3.25/3.02 5-:-3 - 0004df

3 FNode03[aeaac8] L\_\_\_vm03 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 4/ 3 0/0/0/0 72/97/73/29 0 0 667/119/426 15/0/0 3.27/7.99 5-:-3 - 0004df

4 FNode04[f0b7e3] L\_\_\_vm05 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 3/ 2 0/0/0/0 90/48/70/15 0 0 667/113/415 75/0/0 3.21/0.43 5-:-2 - 0004df

5 FNode05[702025] L\_\_\_vm00 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 4/ 3 0/0/0/0 87/70/52/25 0 0 667/121/434 45/0/0 3.30/2.03 5-:-3 - 0004df

6 w FNode06[67ee42] L\_\_\_vm07 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 3/ 3 0/0/0/0 86/43/66/49 0 0 667/114/419 15/0/0 3.24/1.74 5-:-3 - 0004df

7 FNode07[a5b597] L\_\_\_vm02 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 4/ 3 0/0/0/0 91/89/51/55 0 0 667/114/426 60/0/0 3.25/1.37 5-:-3 - 0004df

8 FNode08[d2bc4e] A\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 3 0/0/0/0 81/63/51/55 0 0 667/116/423 45/0/0 3.25/2.49 5-:-3 - 0004df

9 FNode09[7f03e5] A\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 3 0/0/0/0 87/36/62/46 0 0 667/116/426 60/0/0 3.26/4.25 5-:-3 - 0004df

10 FNode10[c0bc99] A\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 2 0/0/0/0 111/19/51/13 0 0 280/111/400 30/0/0 2.22/3.77 5-:-2 - 0004df

11 FNode11[0e03c9] A\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 3 0/0/0/0 85/92/53/49 0 0 667/118/419 60/0/0 3.25/3.45 5-:-3 - 0004df

12 FNode12[435ae7] \_\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 3 0/0/0/0 81/69/49/48 0 0 667/113/419 15/0/0 3.22/2.21 5-:-3 - 0004df

13 FNode13[9b844d] \_\_\_\_ 0/ 0 1.0% 0.500 4[584bfb] 4/5/6 -/ 2 0/0/0/0 106/12/51/13 0 0 160/112/408 65/0/0 1.95/0.00 5-:-2 - 0004df

EB Complete 0 EB Processing 0 Entries Complete 4 Faults 0

#Peers: 5 Avg/Total in Kbps: Out: 248.734/1243.672 In: 157.080/785.400

675/0/0

0 1 2 3 4 5 6 7 8 9 10 11 12 13

Review 0 0 0 0 0 0 240 0 527 0 0 0 0 0

Holding 843 945 838 691 1015 962 784 858 704 613 713 598 584 896

Commits 269 259 274 260 291 254 278 266 274 271 90 271 273 81

Pending EBs 118 122 118 123 113 124 116 117 116 117 111 118 114 112

Pending Entries 547 560 544 557 528 562 537 548 539 543 511 537 533 520

Acks 12 8 19 7 5 1 17 10 24 16 13 12 20 10

MsgQueue 13 10 39 19 0 39 0 37 0 0 0 3 0 0

InMsgQueue 2 0 0 4 4 0 22 1 3 8 584 5 4 703

InMsgQueue2 322 3 250 337 12 5 369 163 412 596 977 632 535 787

APIQueue 0 0 0 0 0 0 0 0 0 0 0 0 0 0

AckQueue 1 1 3 1 2 3 0 2 0 0 45 12 2 13

TimerMsgQueue 0 0 0 0 0 0 0 0 0 0 0 0 0 0

NetworkOutMsgQueue 0 0 0 0 0 0 0 0 0 0 0 0 0 0

NetworkInvalidMsgQueue 0 0 0 0 0 0 0 0 0 0 0 0 0 0

UpdateEntryHash 0 0 0 0 0 0 0 0 0 0 0 0 0 0

MissingEntries 0 0 0 0 0 0 0 0 0 0 0 0 0 0

WriteEntry 0 0 0 0 0 0 0 0 0 0 0 0 0 0

System List FNode0 Length: 0

===SummaryEnd===

The part which indicates the problem is in the column that has some simnodes leaders at 4/3, some at 3/3, and some at 3/2. This indicates that the leaders are in disagreement on their

**Future correction steps:**

* The particular customer on the legacy systems is planning on upgrading to the newer more robust version.
* Release a new version of factomd to the testnet and to the mainnet which resolves the stability regression that was revealed.
* Continue with the overall refactor of factomd to increase robustness and capacity

**Promising code updates:**

There are two pending fixes that likely solve the problems with the stall.

1. Redesign how End of Minute timers are handled. When a leader would get behind the others and it had backed up queues, it would throw away an internal message telling the node to advance to the next minute. The solution to this is to create a new highest priority queue to handle these types of messages (local EOM). With this update, the leader would still send out EOMs even when behind.
2. Redesign the entrysyncing code. This would vastly simplify getting entries which could have been part of what is stopping Federated servers from progressing. Federated Servers do not move forward if they do not think they have all the entries in the blockchain yet. Simplifying this code will reduce that risk. See progress in FD-967