Floating-Point Nakamoto Consensus

Abstract — It has been shown that Nakamoto Consensus is very useful in the formation of long-term global agreement — and has issues with short-term disagreement which can lead to re-organization ("or-org") of the blockchain. A malicious miner with knowledge of a specific kind of denial-of-service (DoS) vulnerability can gain an unfair advantage in the current Bitcoin network, and can be used to undermine the security guarantees that developers rely upon. Floating-Point Nakamoto Consensus resolves ambiguity in competing solutions and makes it more expensive to replace an already mined block vs. the creation of a new block. This proposed change will expedite the formation of a global consensus through the correct alignment of incentives, and prevents most disagreements from ever forming in the first place.

Introduction

The Bitcoin protocol was created to provide a decentralized consensus on a fully distributed p2p network. A problem arises when more than one proof-of-work is presented as the next solution block in the blockchain. Two solutions of the same height are seen as authoritative equals which is the basis of a growing disagreement. A node will adopt the first solution seen, as both solutions propagate across the network a race condition of disagreement is formed. This race condition can be controlled by byzentiene fault injection commonly referred to as an "eclipsing" attack. When two segments of the network disagree it creates a moment of weakness in which less than 51% of the network's computational resources are required to keep the network balanced against itself.

Nakamoto Consensus

Nakamoto Consensus is the process of proving computational resources in order to determine eligibility to participate in the decision making process. If the outcome of an election were based on one node (or one-IP-address-one-vote), then representation could be subverted by anyone able to allocate many IPs. A consensus is only formed when the prevailing decision has the greatest proof-of-work effort invested in it. In order for a Nakamoto Consensus to operate, the network must ensure that incentives are aligned such that the resources needed to subvert a proof-of-work based consensus outweigh the resources gained through its exploitation. In this consensus model, the proof-of-work requirements for the creation of the next valid solution has the exact same cost as replacing the current solution. There is no penalty for dishonesty, and this has worked well in practice because the majority of the nodes on the network are honest and transparent, which is a substantial barrier for a single dishonest node to overcome.

A minimal network peer-to-peer structure is required to support Nakamoto Conesus, and for our purposes this is entirely decentralized. Messages are broadcast on a best-effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone. This design makes no guarantees that the peers connected do not misrepresent the network or so called "dishonest nodes." Without a central authority or central view - all peers depend on the data provided by neighboring peers - therefore a dishonest node can continue until a peer is able to make contact an honest node.

Security

In this threat model let us assume a malicious miner possesses knowledge of an unpatched DoS vulnerability (“0-day”) which will strictly prevent honest nodes from communicating to new members of the network - a so-called “total eclipse.” The kind of DoS vulnerability needed to conduct an eclipse does not need to consume all CPU or computaitly ability of target nodes - but rather prevent target nodes from forming new connections that would undermine the eclipsing effect. These kinds of DoS vulnerabilities are somewhat less substional than actually knocking a powerful-mining node offline. This class of attacks are valuable to an adversary because in order for an honest node to prove that a dishonest node is lying - they would need to form a connection to a segment of the network that isn’t entirely suppressed. Let us assume a defense-in-depth strategy and plan on this kind of failure.

Let us now consider that the C++ Bitcoind has a finite number of worker threads and a finite number of connections that can be serviced by these workers. When a rude client occupies all connections - then a pidgin-hole principle comes into play. If a network's maximum capacity for connection handlers ‘k’, is the sum of all available worker threads for all nodes in the network, establishing ‘k+1’ connections by the pidgin-hole principle will prevent any new connections from being formed by honest nodes - thereby creating a perfect eclipse for any new miners joining the network would only be able to form connections with dishonest nodes.

Now let’s assume a dishonest node is modified in two ways - it increases the maximum connection handles to hundreds of thousands instead of the current value which is about 10. Then this node is modified to ignore any solution blocks found by honest nodes - thus forcing the dishonest side of the network to keep searching for a competitive-solution to split the network in two sides that disagree about which tip of the chain to use. Any new solution propagates through nodes one hop at a time. This propagation can be predicted and shaped by dishonest non-voting nodes that are being used to pass messages for honest nodes.

At this point an attacker can expedite the transmission of one solution, while slowing another. If ever a competing proof-of-work is broadcasted to the network, the adversary will use their network influence to split knowledge of the proof-of-work as close to ½ as possible. If the network eclipse is perfect then an adversary can leverage an eigen-vector of computational effort to keep the disagreement in balance for as long as it is needed. No mechanism is stopping the attacker from adding additional computation resources or adjusting the eclipsing effect to make sure the system is in balance. As long as two sides of the network are perfectly in disagreement and generating new blocks - the attacker has intentionally created a hard-fork against the will of the network architects and operators. The disagreement needs to be kept open until the adversary’s transactions have been validated on the honest chain - at which point the attacker will add more nodes to the dishonest chain to make sure it is the ultimate winner - thus replacing out the honest chain with the one generated by dishonest miners.

This attack is convenient from the adversary’s perspective, Bitcoin being a broadcast network advertises the IP addresses of all active nodes - and Shodan and the internet scanning project can find all passive nodes responding on TCP 8333. This should illuminate all honest nodes on the network, and even honest nodes that are trying to obscure themselves by not announcing their presence. This means that the attacker doesn’t need to know exactly which node is used by a targeted exchange - if the attacker has subdued all nodes then the targeted exchange must be operating a node within this set of targeted honest nodes.

During a split in the blockchain, each side of the network will honor a separate merkel-tree formation and therefore a separate ledger of transactions. An adversary will then broadcast currency deposits to public exchanges, but only on the weaker side, leaving the stronger side with no transaction from the adversary. Any exchange that confirms one of these deposits is relying upon nodes that have been entirely eclipsed so that they cannot see the competing chain - at this point anyone looking to confirm a transaction is vulnerable to a double-spend. With this currency deposited on a chain that will become ephemeral, the attacker can wire out the account balance on a different blockchain - such as Tether which is an erc20 token on the Ethereum network which would be unaffected by this attack. When the weaker chain collapses, the transaction that the exchange acted upon is no longer codified in Bitcoin blockchain's global ledger, and will be replaced with a version of the that did not contain these deposits.

Nakamoto Consensus holds no guarantees that it’s process is deterministic. In the short term, we can observe that the Nakamoto Consensus is empirically non-deterministic which is evident by re-organizations (re-org) as a method of resolving disagreements within the network.   During a reorganization a blockchain network is at its weakest point, and a 51% attack to take the network becomes unnecessary. An adversary who can eclipse honest hosts on the network can use this as a means of byzantine fault-injection to disrupt the normal flow of messages on the network which creates disagreement between miners.

DeFi (Decentralized Finance) and smart-contract obligations depend on network stability and determinism. Failure to pay contracts, such as what happened on “black thursday” resulted in secured loans accidentally falling into redemption. The transactions used by a smart contract are intended to be completed quickly and the outcome is irreversible. However, if the blockchain network has split then a contract may fire and have it’s side-effects execute only to have the transaction on the ledger to be replaced. Another example is that a hard-fork might cause the payer of a smart contract to default - as the transaction that they broadcasted ended up being on the weaker chain that lost. Some smart contracts, such as collateral backed loans have a redemption clause which would force the borrower on the loan to lose their deposit entirely.

With two sides of the network balanced against each other - an attacker has split the blockchain and this hard-fork can last for as long as the attacker is able to exert the computational power to ensure that proof-of-work blocks are regularly found on both sides of the network. The amount of resources needed to balance the network against itself is far less than a 51% attack - thereby undermining the security guarantees needed for a decentralized untrusted payment network to function. An adversary with a sufficiently large network of dishonest bots could use this to take a tally of which miners are participating in which side of the network split. This will create an attacker-controlled hard fork of the network with two mutually exclusive merkle trees. Whereby the duration of this split is arbitrary, and the decision in which chain to collapse is up to the individual with the most IP address, not the most computation.

In Satoshi Nakamoto's original paper it was stated that the electorate should be represented by computational effort in the form of a proof-of-work, and only these nodes can participate in the consues process. However, the electorate can be misled by non-voting nodes which can reshape the network to benefit an individual adversary.

Chain Fitness

Any solution to byzantine fault-injection or the intentional formation of disagreements must be fully decentralized. A blockchain is allowed to split because there is ambiguity in the Nakamoto proof-of-work, which creates the environment for a race-condition to form. To resolve this, Floating-Point Nakamoto Consensus makes it increasingly more expensive to replace the current winning block. This added cost comes from a method of disagreement resolution where not every solution block is the same value, and a more-fit solution is always chosen over a weaker solution. Any adversary attempting to have a weaker chain to win out would have to overcome a kind of relay-race, whereby the winning team's strength is carried forward and the loser will have to work harder and harder to maintain the disagreement. In most cases Floating-Point Nakamoto Consensus will prevent a re-org blockchain from ever going past a single block thereby expediting the formation of a global consensus. Floating-Point Nakamoto Consensus cements the lead of the winner and to greatly incentivize the network to adopt the dominant chain no matter how many valid solutions are advertised, or what order they arrive.

The first step in Floating-Point Nakamoto Consensus is that all nodes in the network should continue to conduct traditional Nakamoto Consensus and the formation of new blocks is dictated by the same zero-prefix proof-of-work requirements. If at any point there are two solution blocks advertised for the same height - then a floating-point fitness value is calculated and the solution with the higher fitness value is the winner which is then propagated to all neighbors. Any time two solutions are advertised then a re-org is inevitable and it is in the best interest of all miners to adopt the most-fit block, failing to do so risks wasting resources on a mining of a block that would be discarded. To make sure that incentives are aligned, any zero-prefix proof of work could be the next solution, but now in order to replace the current winning solution an adversary would need a zero-prefix block that is also more fit that the current solution - which is much more computationally expensive to produce.

Any changes to the current tip of the blockchain must be avoided as much as possible. To avoid thrashing between two or more competitive solutions, each replacement can only be done if it is more fit, thereby proving that it has an increased expense. If at any point two solutions of the same height are found it means that eventually some node will have to replace their tip - and it is better to have it done as quickly as possible so that consensus is maintained.

In order to have a purely decentralized solution, this kind of agreement must be empirically derived from the existing proof-of-work so that it is universally and identically verifiable by all nodes on the network. Additionally, this fitness-test evaluation needs to ensure that no two competing solutions can be numerically equivalent.

Let us suppose that two or more valid solutions will be proposed for the same block. To weigh the value of a given solution, let's consider a solution for block 639254, in which the following hash was proposed:

    0000000000000000008e33faa94d30cc73aa4fd819e58ce55970e7db82e10f8

There are 19 zeros, and the remaining hash in base 16 starts with 9e3 and ends with f8. This can value can be represented in floating point as:

    19.847052573336114130069196154809453027792121882588614904

To simplify further lets give this block a single whole number to represent one complete solution, and use a rounded floating-point value to represent some fraction of additional work exerted by the miner.

    1.847

Now let us suppose that a few minutes later another solution is advertised to the network shown in base16 below:

    000000000000000000028285ed9bd2c774136af8e8b90ca1bbb0caa36544fbc2

The solution above also has 19 prefixed zeros, and is being broadcast for the same blockheight value of 639254 - and a fitness score of 1.282. With Nakamoto Consensus both of these solutions would be equivalent and a given node would adopt the one that it received first. In Floating-Post Nakamoto Consensus, we compare the fitness scores and keep the highest. In this case no matter what happens - some nodes will have to change their tip and a fitness test makes sure this happens immediately.

With both solutions circulating in the network - any node who has received both proof-of-works should know 1.847 is the current highest value, and shouldn't need to validate any lower-valued solution. In fact this fitness value has a high degree of confidence that it won't be unseated by a larger value - being able to produce a proof-of-work with 19 0's and a decimal component greater than 0.847 is non-trivial. As time passes any nodes that received a proof-of-work with a value 1.204 - their view of the network should erode as these nodes adopt the 1.847 version of the blockchain.

All nodes are incentivized to support the solution with the highest fitness value - irregardless of which order these proof-of-work were validated. Miners are incentivized to support the dominant chain which helps preserve the global consensus.

Let us assume that the underlying cryptographic hash-function used to generate a proof-of-work is an ideal primitive, and therefore a node cannot force the outcome of the non-zero component of their proof-of-work. Additionally if we assume an ideal cipher then the fitness of all possible solutions is gaussian-random. With these assumptions then on average a new solution would split the keyspace of remaining solutions in half. Given that the work needed to form a new block remains a constant at 19 blocks for this period - it is cheaper to produce a N+1 block that has any floating point value as this is guaranteed to be adopted by all nodes if it is the first solution. To leverage a chain replacement on nodes conducting Floating-Point Nakamoto Consensus a malicious miner would have to expend significantly more resources.

Each successive n+1 solution variant of the same block-height must therefore on average consume half of the remaining finite keyspace. Resulting in a the n+1 value not only needed to overcome the 19 zero prefix, but also the non-zero fitness test.   It is possible for an adversary to waste their time making a 19 where n+1 was not greater, at which point the entire network will have had a chance to move on with the next solution. With inductive reasoning, we can see that a demissiniong keyspace increases the amount of work needed to find a solution that also meets this new criteria.

Now let us assume a heavily-fragmented network where some nodes have gotten one or both of the solutions. In the case of nodes that received the proof-of-work solution with a fitness of 1.847, they will be happily mining on this version of the blockchain. The nodes that have gotten both 1.847 and .240 will still be mining for the 1.847 domainite version, ensuring a dominant chain. However, we must assume some parts of the network never got the message about 1.847 proof of work, and instead continued to mine using a value of 1.240 as the previous block. Now, let’s say this group of isolated miners manages to present a new conflicting proof-of-work solution for 639255:

```
0000000000000000000058d8ebeb076584bb5853c80111bc06b5ada35463091a6
```

The above base16 block has a fitness score of 1.532 The fitness value for the previous block 639254 is added together:

$$2.772 = 1.240 + 1.532$$

In this specific case, no other solution has been broadcast for block height 639255 - putting the weaker branch in the lead. If the weaker branch is sufficiently lucky, and finds a solution before the dominant branch then this solution will have a higher overall fitness score, and this solution will propagate as it has the higher value. This is also important for transactions on the network as they benefit from using the most recently formed block - which will have the highest local fitness score at the time of its discovery. At this junction, the weaker branch has an opportunity to prevail enterally thus ending the split.

Now let us return to the DoS threat model and explore the worst-case scenario created by byzantine fault injection. Let us assume that both the weaker group and the dominant group have produced competing proof-of-work solutions for blocks 639254 and 639255 respectively. Let's assume that the dominant group that went with the 1.847 fitness score - also produces a solution with a similar fitness value and advertises the following solution to the network:

> 0000000000000000000455207e375bf1dac0d483a7442239f1ef2c70d050c113
> 19.414973649464574877549198290879237036867705594421756179
> or
> 3.262 = 1.847 + 1.415

A total of 3.262 is still dominant over the lesser 2.772 - in order to overcome this - the 2nd winning block needs to make up for all of the losses in the previous block. In this scenario, in order for the weaker chain to supplant the dominant chain it must overcome a -0.49 point deficit. In traditional Nakamoto Consensus the nodes would see both forks as authoritative equals which creates a divide in mining capacity while two groups of miners search for the next block. In Floating-Point Nakamoto Consensus any nodes receiving both forks, would prefer to mine on the chain with an overall fitness score of +3.262 - making it even harder for the weaker chain to find miners to compete in any future disagreement, thereby eroding support for the weaker chain. This kind of comparison requires an empirical method for determining fitness by miners following the same same system of rules will insure a self-fulfilled outcome. After all nodes adopt the dominant chain normal Nakamoto Consuess can resume without having to take into consideration block fitness. This example shows how disagreement can be resolved more quickly if the network has a mechanism to resolve ambiguity and de-incentivise dissent.

Soft Fork

Blockchain networks that would like to improve the consensus generation method by adding a fitness test should be able to do so using a “Soft Fork” otherwise known as a compatible software update. By contrast a “Hard-Fork” is a separate incompatible network that does not form the same consensus. Floating-Point Nakamoto Consensus can be implemented as a soft-fork because both patched, and non-patched nodes can co-exist and non-patched nodes will benefit from a kind of herd immunity in overall network stability. This is because once a small number of nodes start following the same rules then they will become the deciding factor in which chain is chosen. Clients that are using only traditional Nakamoto Consensus will still agree with new clients over the total chain length. Miners that adopt the new strategy early, will be less likely to lose out on mining invalid solutions.

Conclusion

Floating-Point Nakamoto consensus allows the network to form a consensus more quickly by avoiding ambiguity allowing for determinism to take hold. Bitcoin has become an essential utility, and attacks against our networks must be avoided and adapting, patching and protecting the network is a constant effort. An organized attack against a cryptocurrency network will undermine the guarantees that blockchain developers are depending on.

Any blockchain using Nakamoto Consensus can be modified to use a fitness constraint such as the one used by a Floating-Point Nakamoto Consensus. An example implementation has been written and submitted as a PR to the bitcoin core which is free to be adapted by other networks.

A complete implementation of Floating-Point Nakamoto consensus is in the following pull request:
https://github.com/bitcoin/bitcoin/pull/19665/files

Paper:
https://github.com/in-st/Floating-Point-Nakamoto-Consensus
https://in.st.capital