Avrio: A scalable, decentralised cash

Leo Cornelius
leocornelius019@gmail.com
www.avrio.tk

Abstract. A peer-to-peer scalable currency using a blockchain for every wallet, rather than one, validated by a network of economically incentivised "nodes". Empowered by sharding, the network's speed is linearly proportional to the power of the nodes and the number of committees. In this paper, we discuss the protocol behind this, as well as the economics of the coin.

0. Definitions

c = number of nodes per committee,
k = number of shards,
r = transactions per second per user,
n = number of nodes,
y = number of consensus nodes,
s = number of seed nodes,
PoN = Proof of Node,
PoW = Proof of Work,
Salt or *Nonce* = a random number/string

1. Introduction

In the year 2009, Bitcoin was released. It provided the first-ever decentralised method of monetary exchange. At first, it seemed like a solution to a major problem in modern finance - centralised control of money. However, as it gained more users, a major flaw emerged. Bitcoin could handle only ~7 transactions per second while centralised payment vendors such as Visa could handle up to 24,000 per second. This is largely down to Bitcoin's core design in which all transactions are stored on a public blockchain. Each transaction has to wait to be included in a block and blocks can only contain a few transactions meaning that transactions per second are severely capped. We now propose a new protocol, Avrio, which replaces the single public blockchain and PoW blocks with wallet chains and PoN.

2. Nodes

Nodes are core aspects of Avrio, they are servers running the Avrio software (however, in the context of this paper, we are using the word 'node' to replace fullnode). Anyone can run a node by simply running the software, however, fullnodes (nodes that are assigned committees and rewarded) must pay a registration fee to prevent a datacentre 51 attacking the network. Nodes are split into committees every epoch and each committee is responsible for verifying and propagating transactions from certain wallet chains (or shards).

All nodes run the same software but some nodes have different roles.

1) Seed nodes

These nodes are hardcoded into the source code. New nodes connect to these nodes on startup to get the peer list and synchronize from it. These nodes are in their own committee, the seed committee. They do not process transactions but maintain a full and up to date copy of the blockchain. Anyone can run a seed node as long as their node's IP address is added to the source code.

Seed nodes are also responsible for checking for updates. They do this by polling the git services and checking for new releases. The description of all releases contains a long and seemingly random string that can be processed by the node software to determine details about the release (e.g. release name, mandatory or not, etc.). Then the seed nodes make a block and release it into the blockchain. This block is an empty block other than the message section, which contains the release information. When nodes see this block (during synchronizing at the next epoch), they do the update.

2) Assessment nodes

At the beginning of each epoch, 5% of the committee nodes are selected to assess every node’s performance periodically. They do this by computing a vote based on a number of assessed parameters. These nodes are chosen based off a function of their ID, committee hash and epoch salt. The final value of the vote is a value ranging from 0 to 100, 100 being impeccable and 0 entirely deficient and awful. This vote on the performance of the node is based on a number of tested parameters:

1) Ping response time.

In this test, nodes ping the node they are testing and time the response. They then deduct 1 point from 20 for every 5 ms of latency over network average. To accommodate for natural network speed differences, the nodes ping three highest voted nodes, calculate the difference from what is considered a “normal time”, and factor this into the score.

2) Transaction validation time

In this test, nodes assess the time it take for a node to process, verify, sign and submit a block divided by the number of transactions in the block minus the assessed PRT (ping response time). This is accepted as the test of a nodes efficiency. The score is out of 50 and a point is lost for every x ms of delay.

3) Time to live (TTL) delay

In this test, a node is assessed based on the time it takes to complete all tasks needed to begin verifying transactions at maximum speed. It is scored out of 30 and 1 point is lost for every 5 seconds above the network average over the last 4 epochs. The assessment nodes contact the consensus committee at the end of the epoch to get this timestamp.

4) Blockchain Digest

This test does not score the node any extra points but is required for the node to be considered non-malicious and to allow any participation. The node creates a digest based on current top blocks (of the shards the committee is dealing with) and then sends it to the assessor node. The node must complete this within a certain period for it to be valid. If the node fails this test, it is informed that it is out-of-sync by the assessor node. The node then polls previous connections to their top blocks digest and if less than half+1 nodes respond with a different block, node resynchronizes. If not, then it informs the assessor node that *it* may be out-of-sync (the assessor node then double-checks that it is not out-of-sync or on the wrong chain).

At the end of the epoch, the assessment nodes create a final vote for each node, sign it, and publish it to the other assessment nodes. The assessment

nodes then check the signatures and create an average (for each node) via a mean of the values of each assessment node. Every assessment node then signs this and publishes it to the consensus committee.

3) Consensus nodes

These nodes, like assessment nodes, are selected via a function of the last block hash and their ID (this is described in detail in the committee section). However, unlike assessment nodes, consensus nodes have their own committee. This committee deals with:

1) Creating the new committees

Consensus nodes deal with the formation of committees and assign the shards and nodes (based off their IDs)

2) Creating epoch salt

Epoch salt is generated by the consensus committee at the beginning of each epoch. Each consensus node shares a self-generated random number with the other committee nodes, along with a signature. Then each node does the following on each number:

x = network_blockhash_digest.as_bytes()[random_number mod 64]

Then they take every number and do a random math operand to create a final number, this math operand is determined by taking the value of the first number of the ASCII code (mod 5) and selecting a math operand:

1 = +, 2 = -, 3 = *, 4 = /, 5 = %%;

The end value is then rounded and made positive, before being signed by at least y / 2 + 1 nodes (y = number of consensus nodes) and publishing it to the network.

3) Connect committee members together

The consensus committee from the previous epoch is used a 'yellow paper' as such - a node (after formulating an ID) contacts the consensus committee with its ID. Consensus nodes then determine the committee from the ID (as described in the committees section) before returning a

peer list (of the existing committee members) and a committee hash to the node. Consensus nodes then make a record in UNIX time) when the node submitted its ID (which is then used by the assessment nodes, as described above).

4) Defining the registration fee.

The registration fee is calculated by working out what £10 worth of Avrio is. It does this by polling every exchange's API, which is hardcoded into the source code to get the price, before making a mean of the values, and signing it. It then shares this with the rest of the consensus nodes. If there are different values shared, then the most common one is accepted. It then calculates the price of 10 AIO, signs it, and publishes it to the network as the new fee. Any attempted registrations in the next epoch with a fee under this value will be rejected.

4) 'Normal' nodes

These nodes make up the majority of the nodes and are the ones who validate transactions. Each node has a list of valid peers in its committee (which it regularly rechecks with the consensus committee) but does not handshake with all of them as this would be very expensive (broadband-wise) and would result in c^2 messages in each committee. Instead, it simply disallows connections from peers not in this list. Once a node has created the blocks, it sends it to all nodes which have messaged it that epoch or (if it is the first block of the epoch) all nodes in his list.

3. Wallet chains/shards

In Avrio, every wallet has its own chain. These are known as shards or wallet chains. This chain only has a block added to it when a transaction related to it occurs (e.g. when a wallet creates a send or receive transaction). This removes the bottleneck introduced by a central blockchain and allows the blockchain to scale linearly to the number of full nodes while also drastically reducing user storage requirements and TTL (due to synchronizing) times.

A transaction can add a byte of extra data; this can be used to set up user-based deposits. This allows services to integrate Avrio as a payment

method, then provide a unique address to each customer (which will send to their wallet chain but with the data automatically added) and wait for the transaction to their wallet with the correct extra data. This is similar to Cyptonote's payment ID but much more compact as it only takes up one extra byte of space as opposed to 64.

Multiple shards [x = (n - (s + y)] / c (x is equal to number of shards per committee) are processed by a single committee (other than the seed and consensus committees). The associated blockchains are stored only by the user, the fullnodes, seed nodes and other services, which have a full blockchain for speed (e.g. exchanges). Transactions can be validated (in 'light mode') using just the network balance sheet.

4. Transactions

There are two main types of transactions: send txns and reward txns. A sender creates send transaction, which includes recipient, amount, gas details (price, max, etc.) and a digital ECDSA signature to check integrity. A reward transaction is created by a node - it contains everything in a send txn but not the sender, it must also be broadcasted with a list of votes signed by at least half+1 of the assessment nodes to ensure reward amount is correct. There are a few other less common txns:

1) Username or node registration txn - this transaction is used to register a new username (or fullnode) for a wallet

2) Message txn - using the extra data field of the transaction a sender can send a message to any wallet by simply paying the gas fee

3) Fund lock txn - used by full nodes to lock some funds optionally in order to receive a larger reward. The lock time is set to 30 days at which point the full node automatically renews its certificate and relocks the funds

4) Burn txn - a user can choose to burn coins and allow them to be re-rewarded to full nodes (this will possibly be used to prevent nodes from misbehaving by burning some of their funds if they are defined as malicious)

A block can contain up to 100 transactions (though the receiving chain only stores the txns from that block that are relevant to it in a block) however they cannot be conflicting. A block is considered a state of the blockchain - each new block alters the old state of the blockchain (e.g. registering a new node or changing the balance of wallet X) and thus cannot contradict each other (e.g., send X funds to Alice then send these funds to Bob as well). Node first checks that the estimated work to complete the transaction(s) will be less than the max gas and then the transaction is processed by the node and it updates the state of their blockchains, node list, balance list or anything else that is changed by the txn. The gas is calculated based on the time the committee took to verify the txn, the number of operations needed to verify the txn and the size of the transaction. This is then multiplied by the gas price to get the fee in AIO. This fee is split between the committee based on the vote they get at the end of each epoch - this does not affect their daily (24 hrs of uptime) collection of funds

4.2 Validation

There are two methods of transaction validation: light and full. Light requires only a balance sheet (it only checks that the balance is sufficient) and does not guarantee that a transaction is fully valid and will be accepted. Full, on the other hand, requires an up-to-date record of the two involved wallet chains and does every check a fullnode would - guaranteeing that the transaction is valid and providing a strong indication that it will be accepted (not guaranteed due to the possibility of a network fork).

5. Usernames

Most cryptocurrencies utilise long and hard to recite wallet "addresses". This makes end-user adoption very difficult as communicating where to send money becomes extensively difficult. To solve this, Avrio will allow users to register a username tied to their public key, meaning you do not have to use a long and hard to remember address. Usernames are case sensitive and must be 10 or less characters long, allow all letters, numbers and special characters “_“, “.” and “!”, allowing total possible combinations of unimaginable magnitude. You can also create "sub-

usernames" for a low fee. This allows services to register a username (e.g. printersco) and register a sub-username for every user for deposits (e.g. maxcarter.printersco). Avrio Core wallet also allows tracking of each sub-username balance for easy deposit tracking. Sub usernames can also be granted varying levels of access to the wallet, ranging from deposit only, to usage of max of x funds per y, to full access (and everywhere in between).

6. Blocks

Each block has the current block version, the creator's chain key, the previous block hash, the timestamp the block was signed by the creator, the height of the block, the txns in the block, any extra data added to the block, a signature from the creator and a hash of the signatures of at least c / 2 + 1 nodes. The previous block's hash is included to link each block in the wallet's chain together.

Because a block cannot be altered or removed once it is shared across the committee (and later, on the network), forks will only happen if a node is disconnected from the network. If a conflicting block is presented to a node, it polls its committee for their top block, if over 50% of the committee have that block then it is accepted, otherwise it is rejected and all committee members accept the correct block.

7. IDs

IDs are a vital aspect of Avrio. They are used to form committees, choose assessment nodes and provide a unique identifier. They are formed as bellow:

ID = sign(private_key, hash(epochs_salt + public_key + nonce));

Additionally, a new hashing function is generated every epoch based of the last block of the previous epoch and the current epoch salt. A difficulty parameter is generated based off the parameters of this hash function to ensure IDs are generated no faster than 5 minutes on an average CPU - this 'average' is based off the mean of the time-to-live for the network (minus the outliers) over the last 3 epochs (with the newer epochs being weighted). A node keeps incrementing the nonce until its generated ID

fulfills the difficulty parameter (this is similar to Bitcoin's PoW system), however, generating an ID the fastest provides no extra reward or advantage and is, therefore, more comparable to Hashcash.

This difficulty requirement prevents a node generating multiple IDs and publishing them under different IPs (and node registrations which would be extremely expensive, via a VPN) until it either gets the committee it wants or causes a new epoch due to the network wide belief a huge number of nodes just came online.

The unpredictable hash function ensures the algorithm is CPU only and a rainbow table cannot be created in advance.

10. Committees

Committees are a vital aspect of Avrio - they permit the possibility of sharding and, hence, are vital to scalability. Committees are comprised of a set number of nodes though each committee can have 5% less or more nodes. What nodes are in each committee is defined by their ID. Each new epoch the rule set's logic-gate-parameters (which are used to determine what committee each node is in) are randomly generated by the consensus committee and kept private. This is to prevent selective withholding of IDs in order to get in the community of a (likely) malicious node's choice. To generate new parameters, each consensus node chooses a random number and mixes it with the network salt (via modulo) signs it and shares it with the rest of consensus nodes. It then adds all the numbers together and does mod 5. The final value is then used to select a logic gate operand using the following table (1 = and, 2 = nor, 3 = not, 4 = or, 5 = nand). Then the ID is recursively logic-gated (with the selected logic-gate) with itself until 1 to 4 hex values of 0x30 to 0x39 can be determined. The numeric value of the final hex values is added together and moduloed by 125% of the estimated number of nodes (based on previous epoch). This is then the position of the node on a list. Once there are enough nodes to fill each committee, the list is shuffled using the epoch salt via the following (where |x| = positive version of x; e.g. -9 becomes 9 and 2 stays as 2):

all positions ending with 0:

position change = current_position - (epoch_salt mod (1.25 * previous_epoch_node_count)) - last digit of current position)

all other positions:

position change = current_position + (epoch_salt mod (1.25 * previous_epoch_node_count)) - last digit of current position)

First c nodes on the list go into committee with index 0, next c nodes go into the committee with index of 1, etc. After all the nodes have been assigned to a committee, the number of shards assigned to each committee is calculated by [x = (n - (s + y)) / c]; x is equal to number of shards per committee. Then the wallet chains/shards are shuffled to the nodes and assigned to the committees in the same way as nodes were.

During this ID production, synchronization and committee formation, the nodes remain in their existing committees and continue to process transactions as usual (e.g., they function in exactly the same way they have throughout the epoch). As soon as every committee is set up, consensus nodes announce this and every node then swaps to their new committee. This way there is minimal transaction throughput reduction and near continuous transaction processing.

Nodes which had a vote of below a certain threshold (defined by a network average over the last 10 epochs) are denied access to a committee for one epoch on their first ban, 2 epochs on their second, 3 on their third and so on. If a node is temporarily banned 3 times in a row, it is then banned for 10 epochs, (approx. 48 hours). Currently, no permanent bans are implemented due to the fact that even exceptionally performing nodes may occasionally experience overnight failures and not be fixed straight away (due to the operator being asleep), which could result in an exceptional node, or even just an honest node, being banned unjustly. Nodes which do not meet the required version are not assigned a committee until they have updated (which *should* be done automatically by the software in each epoch as needed).

9. Epoch

An epoch is the target period between committee and ID regeneration. In Avrio, it is targeted to 5 hours - this equates to approximately 5 epochs per day. Nodes by default wait until near the end of the epoch to synchronize every other chain's blocks. This is to ensure their performance during the epoch (when they are being tested by the assessment nodes) is maximum (meaning they get a higher vote, and thus a higher reward). However, if a node wants to synchronize continually with the network (for whatever reason), it can choose to do so.

However, for the first few minutes of an epoch, a node will be generating an ID and awaiting a new committee placement (during which it continues everything it was doing in the previous epoch). This period is relatively similar epoch-to-epoch so the time spent in one committee is close to 5 hours.

Every epoch, a node also checks that there are no required software updates. If there are, it updates on another thread (while still functioning). Minor releases (e.g. non-protocol-breaking changes or ones, which, if not applied, will not negatively affect the network or enable security defects/severe bugs to put the network at risk) may be ignored until the node is restarted if this is specified in the node's personal config file (IGNORE_MINOR_UPDATES). Please note that any protocol-breaking or major change is mandatory and any node not updated will not be accepted into another committee until the next epoch (provided it updates by then).

10. Epoch salt

Epoch salt is generated by the consensus committee at the beginning of each epoch by the consensus committee. Each consensus node shares a self-generated random number with other committee nodes, along with a signature. Then each node does the following on each number:

x = network_blockhash_digest.as_bytes()[random_number mod 64]

Then it takes every number and performs a random math operand to create a final number; this math operand is determined by taking the value of the first number of the ASCII code (mod 5) and selecting a math operand.

1 = +, 2 = -, 3 = *, 4 = /, 5 = %%;

The final value is then rounded and made positive, before being signed by at least y / 2 + 1 nodes (y = number of consensus nodes) and published to the network.

11. Emission

Avrio is created with end-users in mind. This means that the coin price must not be too high or too low and most importantly, it has to be *stable*. If a user goes to sleep one day with £1000, they expect to wake up with that (or similar) value; if they wake up and they now have £730, they are going to stop using Avrio. Users are also used to paying a certain amount for each item and if you have a coin that is either very high or very low in price, the resulting cost of the item will seem alien to the users and obstruct adoption. For example, the average coffee (in the UK) costs ~£2.20 - a user will be taken back if that same coffee is 24506.4312 AIO or 0.022 AIO - therefore AIO should ideally remain as close to 1 GBP as possible. Both of these targets can be achieved by using a mixture of slow, steady emission and mass adoption. A coin with fewer coins emitted per day has less new coins being sold on the market and therefore increasing liquidity (reducing the buy-sell gap) and reducing new coin introduction - reducing the swings in a coin's price. Mass adoption will cause less people to want to sell, meaning there is a larger buyer price - pushing the price up to the natural intrinsic value until it is maintained by equal buy-sell power, and this reduces the volatility of a market.

11.2 Rewards

Every epoch, the cumulative value of the fees is split across the nodes based on their votes. This can only be claimed during the next epoch, any unclaimed funds are burnt. The amount that each node gets is calculated with the following formula:

fee_reward = vote * (fees_amount / cumulative_votes_of_committee)

The cumulative vote count includes *every* node in the committee. To collect this fee, the node creates a reward transaction of the correct amount and publishes it along with the vote it received, signed by at least half +1 of the assessment nodes of the relevant epoch. The committee that the node's linked account is processed by then validates the signatures of

the assessors, along with validating the amount, before adding it to the blockchain.

On top of the fees, a node collects a reward every 5 epochs (every ~24 hours). This reward is based on the mean of all votes earned during the 5 epochs and must be claimed after the 5th epoch has finished and the new committees have been adopted. It is calculated based on the following function:

reward = (2.5 / 100) * vote_mean

For example, if a node got a vote of 100/100 then it would receive 2.5 AIO as a reward, 50/100 = 1.25 AIO, etc. To claim the reward, a node takes at least half+1 of the signatures of each epoch's assessment nodes and the final vote, and attaches it to the reward transaction and then broadcasts it to the network.

Like with fee claiming, the committee that the node's linked account is processed by then validates the signatures of the assessors, along with validating the amount before adding it to the blockchain.

This slow and extremely small emission (a vote of 100/100 is practically impossible, an exceptionally performing node might be claiming 2.3 AIO but is unlikely to consistently get more) means it would be nearly impossible for a node to gain enough coins to dump enough to severely affect the coin's price.

12. Launch

The Avrio launch will be split into two parts. The first part is the cryptonote, PoW based coin and the second part is the swap to the sharding, PoN based codebase. The initial launch in a PoW codebase may seem counterintuitive for a coin that is aiming for fair distribution, however, below we describe our reasoning for doing so. The PoN swap will involve four major aspects.

1) Testnet

This is the first stage and will be open to anyone. Please note that all coins minted in the testnet stage will be destroyed, as the chain will be reset at

the end of the testnet stage, ready to progress to the mainnet stage.

2) Coin swap registration

At the end of the testnet, people will be able to apply for their coins to be swapped. The methods of doing so are yet to be finalised but are likely to be done via the uploading of your private key to a website, which will then create an account on the mainnet when launched with these funds.

3) Fullnode free registration period

Because PoN needs a large number of honest nodes running to be secure, anyone can register up to two nodes free of charge before the network launch. To do this, they will enter details online such as the account public key provided to them during the swap (if they do not have one, it will be created), their IP and they will be asked to complete a quiz about secure, efficient practises when running an Avrio node. They will then be asked to download a piece of software that will interact with Avrio's website and confirm the IP address is correct. It will also download and install the daemon code of Avrio when released. The website will then create a registration block and add it to the blockchain, ready for launch.

After this initial free launch, nodes will be required to pay a fee in Avrio, increasing from a predetermined low amount (based on number of nodes already registered) up to 10 GBP worth of AIO over time. This entry fee is discussed in the nodes section of the whitepaper.

We are launching with a PoW, cryptonote codebase initialy before moving to a brand new PoN codebase. We are doing this to increase circulating supply. Good supply is vital to liquidity and liquidity is needed to maintain a stable price. Once the codebase is finished and has been tested we will swap coins and move to the new codebase.

13. Speed

Due to Avrio's parallel blockchain technology and lack of mining, Avrio's only main bottleneck is the computational power of the nodes running Avrio. The number of users is presumed to increase every year. The nodes will most likely follow that path at a smaller rate and slow down once the

user count gets to a mass adoption level. Using current affordable technology, a node can validate roughly 50-150 transactions per second. Bitcoin has an average of 400,000 transactions confirmed per 24 hours, therefore to run a Bitcoin-sized network Avrio would need [((400,000 / 24) / 60) /60] / 100 = only 40 nodes. Moore's law states that computational power doubles every two years, therefore we can gain [(n * c)*100 / r] * 3 new users every two years (provided we gain no new nodes but every node updates to the newest hardware, and this is an unlikely situation).

14. Conclusion

In this paper, we have presented a P2P decentralised payment network allowing much higher amount of transaction per second over any other payment vendor on Earth (infinitely) and one that is capable of infinitely scaling without sacrificing decentralization, security or accessibility. We also discussed the emission schedule and our methods of maintaining a stable, medium price for the coin. We are in the process of creating an implementation of this coin in rust. We have chosen rust as it is as fast (if not faster in some cases) as c and c++, provides exceptional memory management, makes compiling incredibly simple, catches errors that would cause a crash during run time (unlike c/c++) and is much faster to learn, write in and read. Please note that while this aims to be the finalized release of Avrio's blockchain, there will be small changes made over time and a potential for larger changes. Please do not rely purely on this whitepaper for information about this coin; you should do your own research about the risks of investment before participating in the network in any way (be that via running a fullnode, investing or donating).

14. References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. bitcoin.org, 2009.
2. Blockchain stats. Bitcoin statistics. https://blockchain.info/stats, 2012.
3. Bitcoin wiki. Scalability. https://en.bitcoin.it/wiki/Scalability, 2015.
4. Visa. Visa's transactions per second. https://usa.visa.com/content_library/modal/benefits-accepting-visa.html, 2016.

5. Jeff Garzik. Making decentralized economic policy. http://gtf.org/garzik/bitcoin/BIP100-blocksizechangeproposal.pdf, 2015.
6. Gavin Andresen. Bitcoin improvement proposal 101. https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki, 2015.
7. Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol For Open Blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 17–30. DOI:https://doi.org/10.1145/2976749.2978389
8. Jeff Garzik. Bitcoin improvement proposal 102. https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki, 2015