Consensus Circle

An evolvable distributed shared ledger using trusted randomness to power its consensus protocol

Author : Greg Dickason with contributions from Nick Gatland and Cameron McEwan

Consensus Circle is a new blockchain consensus protocol that does not use proof of work (POW) or proof of stake (POS / DPOS) but is based on Byzantine Fault Tolerance (BFT). The consensus mechanism creates each global state change to the ledger through publishing a block of transactions (known as instructions). Each block is produced by a group of randomly selected network nodes working together, called a Consensus Circle, acting as a temporary BFT Quorum.

Nodes in the network are all identified and owned by entities (people or organisations) in a 1:1 model. With ongoing contribution to the network the owning entities can be elevated to higher status which increases the chance their owned node will participate in a Consensus Circle and earn mining fees.

Consensus Circle is evolveable through being able to change all parameters of the network in standard processes without any external forums or external agreements being required.

This paper describes some of the technical concepts and business use cases. A partial prototype written in Python and Lua and utilising Redis as the global ledger state machine is available at https://github.com/gregdickason/consensuscircle.

Terms used in this paper:

- **Owner.** A known person or organisation
- **Entity.** Normally the set of data associated to an owner. Can also be the set of data stored for a particular trusted purpose (for example against a long running workflow or to accumulate votes in an election).
    - **Attribute:** Data element stored against an Entity (eg someone's age, their wallet balance, the number of votes for / against a particular proposition, etc)
- Node or **Agent**. A software program that is owned by an Owner, with the set of all Agents being the entire network. An Owner can own at most 1 Agent. Agents participate in mining new blocks through creating Consensus Circles and propogate blocks and instructions through the network.
- **Consensus Circle**. A set of Agents that communicate to create the next block in the blockchain. Agents are randomly selected from the previous blocks random output and in turn generate a random output that determines the next Consensus Circle.
- **Instruction**. Similar to a Transaction or Smart contract, an Instruction is a script that updates the state of an entity or entities (eg to make a payment, add a data attribute, setup a long running process, etc). Instructions are simple Turing Incomplete scripts that can utilise the random number output of a block to produce non-deterministic blockchain outputs. Instruction Types may be created, modified or deleted if a Consensus Circle unanimously votes on the change, making the network evolveable. In the prototype Instruction Types are written in Lua.

Introduction

The two primary differentiators of Consensus Circle over currently available protocols are:

1. The block creation mechanism utilises **multiple nodes** (called agents) working together to generate a block, most other blockchains use a single node (miner).
2. Multiple nodes working together permit **trusted randomness** in the actual block and also force a level of decentralisation. The network can use trusted randomness to agree on non-deterministic outcomes and to inject this random trust into instructions and longer running workflows (For example to appoint a neutral mediator in a dispute or to randomly select someone from equally suitable candidates).

Other notes on the network:

- The network is a proof of contribution hierarchical network open to all. As it does not use Proof of Work, it does not require excessive power to operate while still remaining resistant to Sybil attacks
- The network allows for arms length interactions, guaranteeing trust in outcomes for independent third parties
- The network allows for participation of real world authorities such as government agencies without requiring them to move all their data and processes onto the blockchain
- The network permits computing at the edge, eliminating weaknesses with on-chain computation (such as requiring gas to execute a smart contract). This allows for general purpose programming languages to be used in workflows, and does not repeat the issues with stored procedures seen in relational databases
- The network allows humans to perform work, allowing for non deterministic workflows based on human judgement
- The network can carry 'proof of payment' in fiat currency or any other token (e.g. loyalty points). This is done through simple additional Instruction Types.
- The network can manage encrypted information while showing validations of that information (I can claim to be a British Citizen and submit my encrypted documents to the British Consulate who can publically verify my claim without revealing the documents to public access).

Components of the Network

The Network is composed of Owners (Normally people or organisations), Entities (data set normally associated to an Owner but can be independent), Instructions and their Instruction Types, and Agents (also called nodes).

Agents run the fabric of the network, executing instructions as part of blocks that update the global ledger. To recap:

- **Owner.** A known person or organisation

- **Entity.** Normally the set of data associated to an owner. Can also be the set of data stored for a particular trusted purpose (eg against a long running workflow or to accumulate votes in an election).
    - **Attribute:** Data element stored against an Entity (eg someone's age, their wallet balance, the number of votes for / against a particular proposition, etc)
- Node or **Agent**. A software program that is owned by an Owner, with the set of all Agents being the entire network.
- ***Instruction Types:*** A defined atomic script that has been accepted into the network as a mechanism to change its state. Takes a set of input parameters and updates entities. Examples include payments (decrement the payers account and increment the payee's account), attribute updates, and even an Instruction Type that allows the creation of new Instruction Types. In practical terms Instruction Types are defined in Lua scripts and are Turing Incomplete.
- ***Instructions*** : These are similar to transactions or smart contracts in other blockchains. An Instruction is an execution of an Instruction Type.

Consensus Circle process and agent hierarchy (how blocks are mined)

The ConsensusCircle network is composed of agents (nodes) in a hierarchy, with each agent randomly selected to participate in a Consensus Circle that generates a new block in the blockchain. This is designed to prevent sybil attacks while allowing all participants to have a stake in securing the network and earning rewards for their participation. The hierarchy is governed such that there are proportionally less agents in higher levels - so that higher level agents have a higher chance of being selected for each circle and hence opportunity to be rewarded with token payments. For example with a 5 circle consensus network (note the number 5 can vary as it is a network parameter):

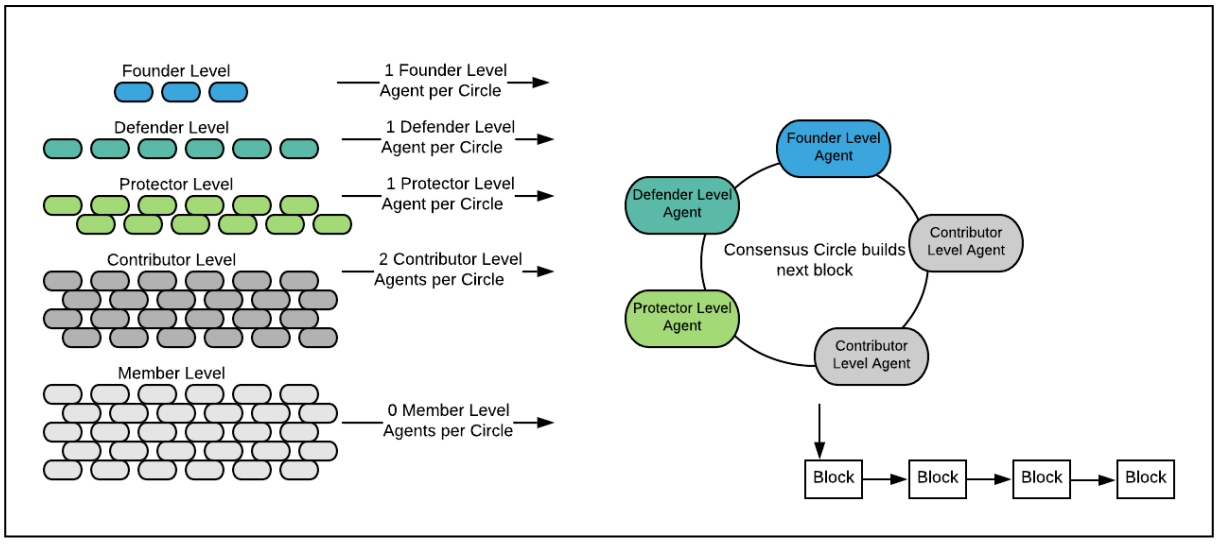


Figure 2: Consensus Circle Selection and Creation of each new Block in the Blockchain. The number of agents in the circle is 5 in the diagram but can be larger or smaller (as set and changed through an instruction executed by a consensus circle).

The circle is responsible for:

- Validating that instructions in the next block are valid

- Selecting which instructions are to be included in the next block in the blockchain
- Creating a set of (coinbase) payments to the previous circles Agents (creates new tokens)
- Participating in a random voting hash that generates random numbers to determine the next set of agents for the next consensus circle (for full description of the implementation of the *'convergence protocol'* see appendix)

All agents in the network maintain copies of the blockchain as a public record. These can then be accessed by client applications and other services (for example to see if someone has a particular attribute, to check account balances, or the status of a particular workflow).

All changes to the network, including software updates, are as a result of Instructions that are validated by one or more consensus circles. See "Appendix: Initial Instructions Types"

Owners and Agents

Owners are the people and organisations that use the network. Each owner can have at most 1 Agent working for them but they do not have to have one. The Agent is the computer software that runs the network (in bitcoin a node or ***miner***).

The following story illustrates how an owner is invited to the network and how they in turn can create an Agent:

"Bob signs up to the consensus circle network by downloading a mobile app and accepting an invite from his friend. He thinks up a complicated passphrase to create his private and public keys and uses this to add himself to the network.

What is added to the network when Bob signs up is an Entity of initial attributes that include his public key (which the network uses to identify him) and a wallet with starting balance of 0 coins. After signing up, Bob purchases some coins from a friend. This increases his balance to 25 coins. He then claims a skill and gets it verified by the network, which adds attributes to his Entity.
Bob does human work for the network by acting as a verifier for other network participants skills claims. During this period external bodies pay fees for the skills profile information and Bob receives fees of $500 for doing this work on the network.
Over time he builds up more and more attributes and history of work performed on the blockchain and in the process gains reputation from others.
Eventually Bob decides that he would like to have an Agent working for him on the network. He signs up to the 'Agent' website and deploys his agent, which costs $42 per month to run and is deployed on Amazon AWS. The Agent that is created works for Bob and any coins earned in Consensus Circles are for Bobs account. Bobs status and contributions to the network mean that other Owners see him as a valid contributor and they vote him to a 'Contributor' ranking - meaning his Agent is able to participate in consensus circles and earn coins.
In the first month the Agent earns $55 dollars worth of coins for Bob, which adds to his work payments of $500, helping him earn a living and helping to secure the network."

Instructions and Entities

Instructions are a single execution of an Instruction Type that normally updates an Entity (such as a payment), but can also change network settings (like the number of Agents in a Circle). In practical terms an instruction is an execution of a Lua script against a Redis instance that holds the state of the blockchain for an Agent.

As the number of Instruction Types is not fixed and can grow by adding new types, in effect the network can hold any type of state about any type of entity. New Instruction types are added by a consensus circle unanimously voting for their inclusion, thus providing a level of due diligence on what changes the network will accept.

Other Network Considerations

In order for the network to maximise its utility it needs to be resistant to attack, able to scale with increasing load, and be easy for anyone to join and to operate an agent. It also needs to have a stable money supply and token exchange rate (be resistant to speculation).

Preventing Attacks

The Consensus Circle network needs to be resistant to attacks that seek to either compromise its integrity or deny service to its users.

Mechanisms to resist attack are detailed at a high level in the below table and in more detail in the reference implementation.

Attack	Mitigation	Commentary
Flood network with multiple instructions	Cost for an instruction limits this attack vector. Owners have to be invited to join the network and can be removed from the network if they exhibit this behaviour.	Form of Sibil attack, may be initiated from groups of people co-operating but will be blocked by cost and ability of the network to remove owners.
DDOS the agents in a consensus circle (Know in advance who they will be so can take them down)	SSL connections between these agents. In worst cast a fork will allow the network to proceed through a different circle.	Although technically possible, consensus circles can fork and continue with different circles of different Agents.
Will the cost of running a agents be sufficiently inexpensive for the small chance of getting a block reward? Especially for agents at level Contributor and Member level.	Incentives will allow sufficient agents to operate to keep network robust (cost of operating an agent vs rewards for doing so). Number of agents will reach economic equilibrium	Reference implementation operates on cloud and could be deployed with largely serverless computing to minimise costs to operate and make it available to the most potential participants to operate an agent.

Consensus Scaling

The design of the consensus circle process allows for no minimum time between consensus circles, although in practice the process to startup, connect and converge on instructions for the next consensus circle will take a non trivial time. There will therefore be a limit to the network throughput (reference POC approximately 6000 workflow instructions per minute assuming 10 seconds per circle). Scaling the whole network can be achieved in 2 ways:

- Optimise the code and network connections. The reference POC is not performance optimised
- Execute an instruction to change network settings - eg the minimum number of instructions per block (initially 100) and the maximum (initially 1000).
- A special instruction could be constituted where the circle shards (splits) into 2^n consensus circles that are able to simultaneously process blocks (note the nodes software would need to be first upgraded - hard fork). Instructions based on their hash are assigned to different blocks, with each circle processing hashes in their assigned range (0 to $2^{256}/n$, $2^{256}/n$ to (2 X 2^{256})/n, ... , ((n-1) X 2^{256})/n).
    - If n is decreased the lower range consensus circle selects the next circle in the chain - so if n goes from 1 to 0 then the circle that produced the block for instructions in the range from 0 to $2^{256}/2$ constitutes the next consensus circle which will then process all instructions
    - If n is increased 2 outputs convergence vectors are produced per current circle, one for the lower range and one for the higher range.

Ease of Joining the Network

Joining the network requires someone who is already on the network to invite a new owner using a particular instruction type (note this has a small fee). This limits the attack potential for sybil attacks but makes joining a simple peer to peer operation (I can join you). In effect external organisations (such as 2-sided marketplaces) could create participants easily that mirror the human participants already onboarded onto those marketplaces.

Once on the network, participants can interact and build their reputation through attributes, and when appropriate launch their own agent.

The token economy

The POC produces coins per circle that are distributed to the agents in the previous circle. At this point there is no further work on how the money supply can either be limited or controlled, however as the number of coins produced is a network setting it can be changed with an instruction (and hence the total supply capped or not depending on network participants acceptance).

Conclusion

The consensus circle uses multiple nodes to create each block, and allows for trusted randomness. The network is designed to be:

- Easy to integrate with any external party
- Operated as a trust fabric for distributed decision making with provable arms length interactions (using the convergence protocol)
- Able to accept external authorities in a decision and workflow process (such as verifying citizenship)
- Simple and relatively cheap for anyone to run an agent (node) and share in the rewards of the network
- Resistant to Sybil attacks through having both an agent hierarchy and a proof of contribution approach

The network is built to be as simple as possible to prevent bugs through complexity. We believe it provides a foundation for robust interactions between real world processes and the trust fabric that is the blockchain, with a novel implementation of using trusted randomness. We welcome participation or simply taking the code and further developing it.

All code and all papers are open for others to build off using the MIT license.

Appendix: Consensus circle detailed specification and convergence protocol description

Determining the Input for the next block to form

Participants in the Consensus Circle are determined through an implementation of the convergence protocol (see diagram below and reference the code)

Each agent in the current circle is involved in the selection of the next consensus circle, without being able to game the system to select an agent (such as themselves).

The process involves:

- Each current member generates a vector of length equal to the size of the consensus circle, populated with random numbers of size between 0 and 2^{256} (the maximum size of a sha-256 hash)
- Each member creates a sha-256 hash of the random number vector as their 'hashed vector'
- Each member sends their hashed vector to the circle.
- Once a block is agreed the members broadcast their actual random numbers and these then are confirmed by the network (through comparing to the hashes). Random numbers are then modulus summed (added together and the 2^{256} modulus taken) - this output then becomes the agreed number for the next consensus circle selection.
- The agents in the next consensus circle are the agents whose hashed public key is the next largest in absolute distance from the vector slot in their level. For example if their hashed public key is ***n*** and they are a contributor level agent, they are eligible if they are the first agent in ascending order with a key greater than one of the scalar values in the vector at the contributor level slot.
- To prevent the outside possibility of a malicious attacker gaining control of a consensus circle and from then on being able to constitute all future consensus circles, or of a consensus circle being isolated and unable to complete processing, a few additional rules are encoded:
    - If the next consensus circle is constituted by less than the total number of agents allowed, or the consensus circle is unable to generate a block in a reasonable time, then the network will follow a fork which consists of an alternative consensus circle that is composed of the next nearest set of agents with public key hashes from the last valid block (see details below).
    - In general the network can fork with an alternative consensus circle competing to produce the next block. The Network will follow the longest chain with smallest average absolute difference between each circles agent hashed public keys and the nominated vector slots. Please see below for the full description of how each forks 'Depth Weighted Chain Distance' is calculated.

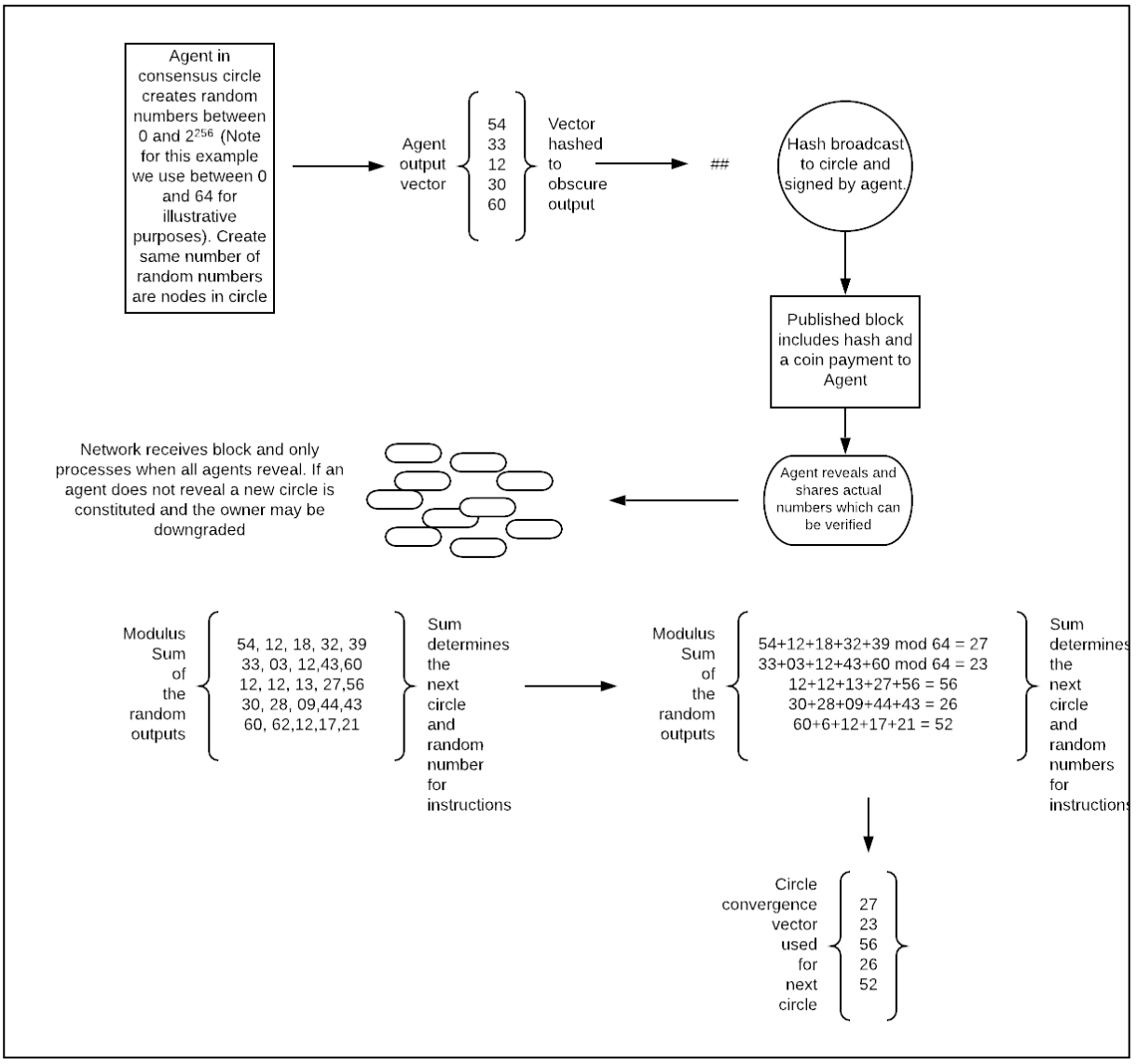


Figure A1.1: *The Convergence Protocol as used for creating the next Consensus Circle. For illustration purposes the example uses a 5 agent circle and random numbers below 64. In reality the number of agents is a network parameter and numbers below 2^{256} are used.*

A1.2 - Forming the next consensus circle using the output from the previous circle (and the concept of network forks)

Once the previous block is published, agents that are the closest to each vector slot in the output vector communicate with each other to constitute the next circle. They perform the following steps in order:

1. Establish connections between them by exchanging signatures. This validates that the correct agents are in the network and prevents some forms of DDOS attack. Most agents will start a separate software instance(eg Docker container) to run the consensus circle
2. Send their list of validated instructions to each other, sorted by order of hash. There is a maximum limit for the number to send and a minimum (to start, 0 Instructions as there are coinbase instructions).
    a. Collate from all inputs (the 16 other agents plus their own list of workflow instructions) which instructions are agreed by all (Unanimous requirement as some

instructions can change network settings so these are in effect votes). Create a new list and broadcast this. This is an iterative process until signed lists all converge to the minimum set all agreed. GREG here

i. Round 1: Each agent broadcasts their list to all other agents plus their hash of generated random numbers. They include a CoinBase instruction per agent, signed to give them the **BlockParticipantReward** (which is reduced for low instruction counts).
ii. Round 2 onwards: Each agent culls all instructions they have not got from every other agent, signs and rebroadcasts until all lists converge.
iii. Agents then broadcast the block: the converged instruction list with all signatures from other agents in the circle and their revealed random numbers
iv. All network agents (not just those in the circle) build the next block from the broadcast blocks from the circle, building up the revealed random numbers. If an agent in the circle does not reveal their random number the number is ignored and the agents coinbase instruction is discarded by the network. Also the network may fork and the whole circle could then be ignored.
v. If an agent behaves dishonestly through signing a list and then revoking Instructions already signed (i.e. not accepting them) then the agent is dropped from the circle and the remaining agents complete processing. This reduces the circle size and leaves the circle vulnerable to a network fork. The dishonest agent loses any coinbase rewards.

b. Note: A block produced by a circle with less than the optimum number of agents (i.e. where an Agent has cheated or otherwise become disconnected) can be replaced by a block built by another circle. (next nearest neighbours to the previous blocks random hash vector). This may also occur if a circle is unable to be constituted where agents are not able to join the network effectively. This is in effect a fork and the network can choose to follow this fork based on the depth (see figure A2 and A3)

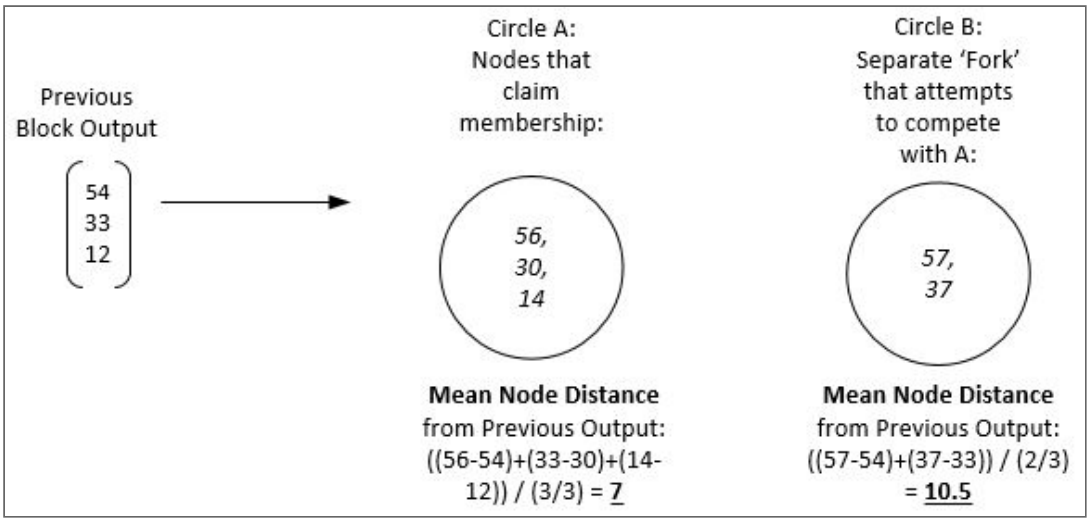


Figure A1.2.1 : Two competing circles for the next block. Circle A is chosen by the network if the block it produces is created in a similar time to Circle B (A has the lowest mean distance). Mean distance is a function of difference from previous block convergence protocol output with next circle agents, and number of agents in consensus circle. Example assumes 3 agents in a full consensus circle. Equation is Sum(abs distance of each agent from its slot value) / (number agents in circle/maximum number of agents in fully formed circle)

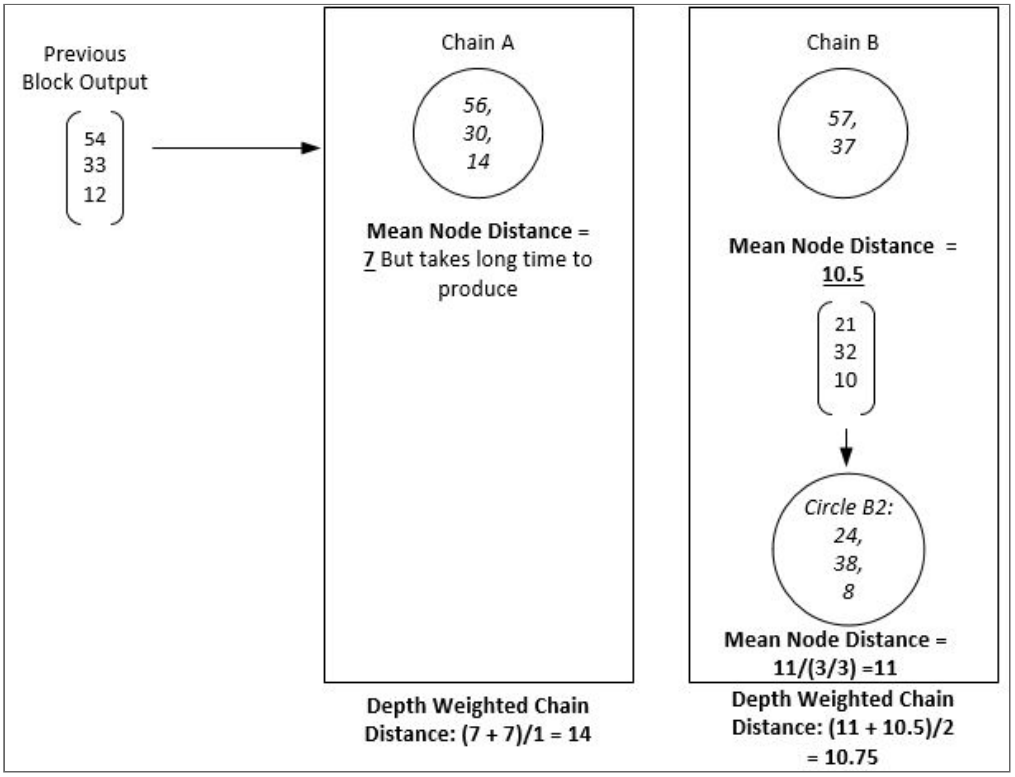


Figure A1.2.2 : Competing chains where the A chain has taken too long to produce a valid block. Network will follow chain B as weighting increases for every block missing in competing chains that are not the longest chain. Chain A is penalised as having only 1 block vs chain B's 2 blocks. Equation is:

Depth Weighted Chain Distance for Chain N = (Sum(Mean Agent Distance of each block since most recent common block) + (Length of longest known chain - length of chain N) X Mean Agent Distance for highest block in chain N) / (length of chain N).

Further example with longer chain depth. Chain 1 has depth since last common block of 3, with Distances of 7, 5 and 8 for these blocks in order; Chain 2 has depth since last common block of 5, with distances of 10, 12, 16, 20 and 10 for these blocks in order.

*Chain 1 Depth Weighted Chain Distance = ((7 + 5 + 8) + (5-3)*8) / 3 = 12*

Chain 2 Depth Weighted Chain Distance = (10 + 12 + 16 + 20 + 10) / 5 = 13.6

In this instance the network will follow chain 1 as it has a lower depth weighted chain distance.

A1.3 - Process for agent promotion/demotion of level

NOT IMPLEMENTED. Currently demotion / promotion is by voting in the consensus circle (unanimous)

Appendix: Initial Instructions Types

On initiation the following types of instructions are recognised. Note any number of new types are possible if Agents (really - Owners) accept them through unanimous voting to include the new

- CreateNewInstructionType. Creates a new type of instruction
- ChangeSetting. Changes the settings of the network (eg the number of agents in a circle, maximum number of Instructions in a block)
- AddOwner. Creates an Owner on the network with the initial set of attributes including a public key. Owners are added by other owners (so cannot self join the network). An Entity that holds extra data on the Owner is also created
- RegisterAgent. This registers an Agent for an owner (can only have at most 1 agent so will remove the previous agent details if present). Can only be registered by Owner
- PromoteAgent. This is an instruction that promotes an Agent up a Level.
- DemoteAgent. This is an instruction that demotes an Agent.
- DeregisterAgent. Remove an agent from the network.
- RemoveOwner. This removes an Owner from the network and deregisters their agent (if they have one).
- Payment. Pay funds from any wallet to any other wallet. Can use any currency
- UpdateAttribute. Updates an entities attributes. Must have permission to do so
- CreateEntity. A non Owner entity that can be used to track the state of long running workflows, accumulate votes in a ballot, etc)
- AddPermission. Allows a non owner to have permissions to update certain attributes. Usually used to allow verification of claimed skills, to mediate disputes, or for a third party to make a payment using another parties wallet. Permissions normally expire after a single update

Appendix: Block Structure

The block structure is similar to that of other public blockchains, but with a convergence header to expedite the convergence in a consensus circle:

{

"convergenceHeader" : convergence header (see below)

“blockHash”: hash of the convergence header

"blockSignatures" : signatures from all the circle agents for the convergence header

"instructions" : list of instructions

"broadcaster" : consensus circle member that broadcasted this block (not unique)

}

The convergence header:

{

"previousBlock" : hash of the previous block

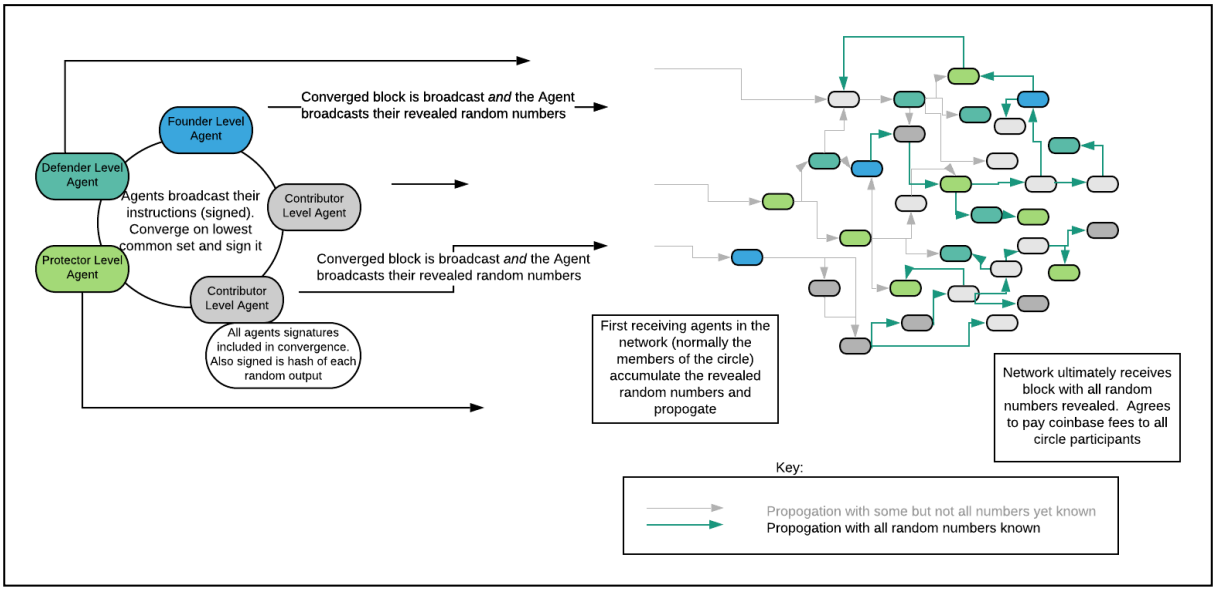
"instructionsMerkleRoot" : merkle root of the included instructions

"instructionCount" : number of instructions in the block (including coinbase)

"blockHeight" : height since epoch (blockheight = 0 for ‘genesis block’),

"randomNumberHashes": hashes of random numbers from all the agents.

}



Mining a block - random number reveals occur in the network post block publication