Intercon v 1.0

Alessandro Toumi

July 15, 2021

1. Introduction

All social structures strive to achieve two goals: that of functionality and that of coordination. "Functionality" in this context means the ability of a social structure to serve its purpose well. That purpose can be anything, *e.g.*, passing and enforcing a law or emitting and controlling a currency. Without this ability our lives would be chaotic and hardly endurable. "Coordination" in the same context is the ability to make the processes that serve social purposes concerted and predictable. This means aggregating the individual processes directed at serving said purposes of a social structure into complex, interdependent clusters that are hierarchical by nature. Think of individual, municipal governments being controlled by a central, national authority. Think of national central banks being controlled by the IMF or the World Bank. Both are examples of a social consensus that is structured hierarchically in order to provide coordination of individual processes, aimed at specific social goals, by an institution above them. This hierarchicity is necessarily involved.

However, it introduces a problem of its own. Any hierarchical consensus is by necessity centralised. All different processes finally must be supervised by one, central entity – otherwise the crucial property of coordination would be lost. And this central entity is simply a person or a group thereof; said person or group will always be guided by their own interests. This assumption is simply an assumption of a rational behaviour [1]. If and when this happens, the will of the few becomes the principle shaping the fortunes of many. We have always been struggling with this dilemma of either sacrificing social functionality for the sake of better coordination, thus changing the social aim from achieving goals that are truly benefiting the majority of us to the ones benefiting the minority, but allowing for coordination of efforts leading to those goals, or drifting towards anarchy, which gets rid of the problem of centralisation of power but at the cost of halted social coordination. Now more than ever, with the advent of the Internet, AI, Big Data, etc., the decisions of the ruling part of society can be made and enforced globally, almost instantaneously and in an automated manner, thus reducing the role of the individual in the process of social governance [2].

The invention of blockchain technology [3] was the first attempt to create a system facilitating social consensus that is both functional, which in this case means creating and operating a financial ledger, and coordinated, *i.e.*, every transaction is verifiably interdependent with the other ones (no double spending), and yet the system is decentralised. No central authority, no problem of individual rationality of behaviour or, simply put, selfishness, controlling the whole. This could have been, or at least it seemed to be, a solution to the dilemma mentioned before. A technology that distributes power in such a way that no centralisation of it is possible. But that experiment failed.

2. The drawbacks of blockchain technology

The two main types of consensus mechanisms in permissionless blockchain-based networks are the Proof-of-Work [4] and the Proof-of-Stake [5]. The Proof-of-Work at first seemed to, thanks to its "one CPU – one vote" rule, allow for decentralisation. But with the invention of the mining ASICs [6] this feature was lost [7]. When the value of the Bitcoin system grew, decentralisation was to a great extent replaced with the technological race to develop more and more efficient mining hardware. And every such race ultimately leads to either monopoly or oligopoly. If Bitcoin ever becomes the global world currency, then we will be ruled by a small group of tech companies, mostly those that monopolised the mining hardware manufacture. They will decide who can own the proper hardware and mine, which transactions can be confirmed in the blocks mined using said hardware and which not; in fact the fate of entire nations will depend on this new "crypto-elite" social class. A future reminiscent of the world that we live in now, the next incarnation of the same polarisation that we face today. The crux of the problem is that the means of reaching consensus in the Proof-of-Work protocol, *i.e.*, hashing power, has a tendency towards centralisation due to the natural technological monopolisation and economies of scale.

Even if a resource other than the hashing power would be used to distribute the coins and decentralise consensus, *e.g.*, storage capacity of hard drives (as in Proof-of-Space-and-Time [8]) or a special, purportedly ASIC-resistant kind of Proof-of-Work (*vide* Programmatic Proof-of-Work [9]), this would not get rid of the fundamental problem

– that as long as the competition is global and is based on some sort of hardware, even if not a specialised one, few hardware manufacturers will inevitably capture the market (by the force of economies of scale) with the products that provide the highest profit margin.

With the Proof-of-Stake protocol the problem gets even bigger. As the distribution of newly mined coins is governed by the current ownership thereof, there is no way around the "rich getting richer" problem [10]. Early backers and investors get to control the present and the future ownership of those coins, so no means of distributing wealth exist.

3. Further analysis of the problem and presentation of the solution

As we can see, even blockchain technology ultimately leads to centralisation of power and it does not solve the dilemma, presented above, between functionality and coordination in social systems. In contrast, probably the most widely used technology – the Internet, is decentralised and yet both fulfils its functional goals perfectly and does that in a concerted and predictable manner.

If we look at the Internet as a social system, we can see that it is composed of many nodes, none of which is a central one, and yet all of them cooperate according to a certain protocol. So the Internet is a system composed of many independent processes that run concurrently, while being interoperable with one another, and yet no central scheduler or managing element is introduced. What we need is a similar system for social and economic consensus. Instead of constantly creating systems that are either hierarchically structured from the onset or systems that will always lead to hierarchicity because of technological monopolisation and economies of scale, *i.e.*, blockchains, we need to create a system of individual, decentralised consensuses, each serving a specific part of society and/or designed for specific purposes, wherein no one single, global means of exercising control over those consensuses exists. If such means exists, it leads to centralisation, either through politics or technology, in the way it was previously described. The logical conclusion here is that the very means of reaching consensus must form a decentralised net of independent, yet connected by a common protocol, consensus-reaching means – rather than being a single, universal and geographically-and-functionally unbounded resource, like hashing power in the case of Bitcoin or network effects in the case of central banking or governments. A net of consensuses that are independent from each other, geographically-and-functionally bounded, non-hierarchically structured and yet interoperable.

4. The Intercon technology

Now that the limitations of other solutions have been presented, the Intercon technology will be described. The Intercon system is a network of fully interoperable [11] blockchains, where the interoperability does not require any central chain or other means of coordination. Each such blockchain facilitates consensus that is geographically bounded, *i.e.*, it is specific to nodes in a certain geography. That can be a region, a country, a city. If it is a country, then the nodes localised in this particular country earn a major share of the block rewards, thus forming a local economic system that is, through mentioned interoperability between such blockchains, fully connected to other similar local economic systems. Furthermore, each such blockchain network can reach consensus not only on its state and transactions, but also on any external (real-world) occurrence(s) of interest to it – even those relating to, *e.g.*, local politics or events, thus forming a local social system.

The description of the Intercon technology will be presented in the subsections that follow, each dealing with a specific feature. Said description, for the sake of simplicity and briefness, deals only with the most critical parts and features of the Intercon protocol, while the full description (which covers some features that are not covered here, e.g., creating algorithmic stablecoins and synthetic assets) is available in a more detailed patent documentation at <https://www.theintercon.org/assets/intercon_documentation.pdf>, hereinafter called the "Documentation". Specific terminology is used in the present document as it is used in the Documentation, so no separate section of definitions will be introduced. Furthermore, it is worth noting that the Intercon technology is meant to be fully open-sourced, both as a code and as an invention, upon release of its first implementation. Any patent rights, if granted, will be held only provisionally, to be abandoned later on, and only in the case they would be at some point necessary for accomplishing the process of building the actual implementation.

4.1. The consensus mechanism

First, the basic method for reaching consensus within the Intercon network will be presented in steps. The used Internet bandwidth is introduced as a real-world resource governing the distribution of coins minted as rewards for the production of blocks, to some extent analogously to how the used hashing power translates into the probability of becoming a block producer and earning a reward in the Bitcoin [3] network. Let us present the steps of reaching consensus on a new block of transactions in the Intercon network:

1. The chain of blocks is divided into sections, the "blockchain sections" (called "checkpoint-to-checkpoint blockchain extensions" in the Documentation) . Let us assume that each such blockchain section consists of 1000 blocks. A user possessing some amount of coins can use them to make a special transaction, called the "encryption keys declaration", that freezes the coins used to make said encryption keys declaration for some number of blocks. Let us assume that Alice uses her address with c coins to make the encryption keys declaration that, once included in one of the blocks, will both:

a.) stop Alice from spending her c coins until the end of the next blockchain section, freezing her address for this period of time

b.) include the root of the Merkle tree, wherein this Merkle tree stores 10,000 symmetric encryption keys, generated by Alice at will

Once included in the blockchain, this encryption keys declaration proves to other nodes that:

a.) Alice has frozen her c coins

b.) The root of the Merkle tree storing 10,000 symmetric encryption keys that Alice has generated is a hash h

This step is described in a more detailed way in section 2 of the Documentation.

2. The 10,000 symmetric encryption keys, declared by Alice in the form of the root of the Merkle tree that stores them in one of the blocks, will be able to be used to generate the so-called "ticket values", which allow to take part in the competition to produce blocks in the blockchain section that immediately follows the blockchain section in which the encryption keys declaration was included. Let us suppose that Alice made her encryption keys declaration in block 5250, belonging to the blockchain section of blocks 5001-6000. That means that her address used to make the encryption keys declaration is frozen until block 7000 (*i.e.*, until the end of the blockchain section immediately following the one in which she made her encryption keys declaration) and that she can participate in the competition to produce each of the blocks from block 6001 to block 7000 (those blocks form the blockchain section following the blockchain section in which the encryption keys declaration was made by Alice). Let us assume that each set of the symmetric encryption keys 1st - 10th, 11th - 20th, 21st - 30th, …, (numbered in accordance with how they are stored in the Merkle tree) declared by Alice can be used during the process of consensus on any single block. First 10 keys can be used to generate the ticket values that allow to take part in the process of consensus on the block 6001, next 10 keys can be used to generate the ticket values that allow to take part in the process of consensus on the block 6002, etc. Let us assume that Alice wants to take part in the process of consensus on the block 6010. That means she can use, in the process of generating the ticket values that allow to take part in the process of consensus on the block 6010, the keys belonging to the set S, where $S = \{K91, K92, ..., K100\}$.

In order to take part in the production of the block 6010, Alice would need to generate ticket values. Ticket values are generated through serial incrementation operations and concatenations and, following each iteration of incrementation and proper concatenations, encryption of the resulting number using appropriate symmetric encryption key from the set of keys that were declared in the form of the encryption keys declaration. This process is described in detail in section 3 of the Documentation; here it will only be covered briefly.

Alice can perform a number n of incrementation operations during the generation of the ticket values to be used in the process aimed at producing block 6010, where $n \propto c$.

Because the number of symmetric encryption keys belonging to S is 10, the number i of incrementation operations that can be performed during the generation of the ticket values that can be generated using each one of the keys belonging to S is $n : 10 = i$.

Alice can either generate the ticket values herself or allow another node to do it on her behalf. As we shall see, the competition to produce a block is based on generating ticket values and sending or receiving big amounts of data, consisting of ticket values generated by other nodes, for which a high-speed Internet connection is crucial. The amount of Internet bandwidth used over time determines the chance of successfully producing a block. Alice, having a low-bandwidth connection (but having made the encryption keys declaration), delegates the task of generating and sending/receiving ticket values to/from other nodes. The nodes that send/receive ticket values are called "satellite nodes". Satellite nodes do not necessarily need to be the nodes that made the encryption keys declarations. Let us suppose that Alice delegates this task to the satellite node David.

David uses encryption keys *K91, K92 , ..., K100* declared by Alice in order to generate the ticket values that will be used in the process aimed at producing block 6010. Let us remember that the symmetric encryption keys themselves are not visible on the blockchain, only the root h of the Merkle tree that stores them. This allows to prove, through the use of a Merkle proof, the inclusion of any particular symmetric key in the Merkle tree whose root was a part of a particular encryption keys declaration, without the need to disclose it at the time of the encryption keys declaration itself. A user that declared encryption keys, which can be used in the current competition aimed at producing a block, through an encryption keys declaration is called a “node-address”. Each time a node-address cooperates with a satellite node, the satellite node can generate ticket values using the encryption keys declared by said node-address.

Furthermore, the node-address can verify that the satellite node with which it cooperates is generating and sending/receiving ticket values to/from other satellite nodes, wherein the amount of ticket values thusly generated and sent/received is the main factor determining the odds of finding a correct pair of ticket values, which allows for the block production, as will be explained later on. The satellite node can verify that only it (from all the satellite nodes) has received specific symmetric encryption keys from any particular node-address. This mutual verification is described in notes 4 and 5 in section 5 of the Documentation.

The process of creating ticket values is described in section 3 of the Documentation. Two crucial features of it are as follows:

a.) a part of the process of creating each ticket value utilises a hash of the header of the previous block, which prohibits the generation of ticket values suitable for the competition aimed at the production of any single particular block before the previous one is produced and known

b.) a part of the process of creating each ticket value utilises a string whose length is readjusted at regular intervals similarly to how the difficulty level [12] is readjusted in the Bitcoin network. In our present example, let this string be α. The length of α depends on the frequency of block production in the Intercon network during a specific preceding period of time; the more Internet bandwidth was used by the satellite nodes during that period of time, the faster the ticket values generated by them were sent/received to/from each other and, as we shall see, the faster this happens, statistically the more frequent the block production. In turn, the more frequent the block production is, the larger the length of the string α. This is described in more detail in item 3 of section 3 of the Documentation.

3. David adds the hash of the header of the previous block 6009 in a decimal form to α, then increments the result of the addition n times. Each single outcome of incrementation is used in a series of specific operations, described in section 3 of the Documentation, which comprise encrypting using the appropriate symmetric encryption key declared by Alice in her encryption keys declaration. The larger n is, the more ticket values David can generate. Let us remember that $n \propto c$. The coins, temporarily frozen because of the encryption keys declaration made by Alice, allow David to generate a specific amount of ticket values. However, the amount itself is not the only factor determining the odds of success, consisting in the production of a block. As we will see, the speed at which the generated ticket values are exchanged between all satellite nodes is a factor as well.

4. David sends ticket values thus generated to other satellite nodes in the network, wherein each one of them has also generated a number of ticket values that depends on the amount of coins frozen because of the applicable encryption keys declaration(s) made by the node(s)-address(es) with which a given satellite node cooperates in a manner analogous to how David cooperates with Alice.

Because success, consisting in the production of a block, depends on finding a pair of ticket values such that, concatenated and (after concatenation) hashed, hashes to an output of value v or lower, the more ticket values a given satellite node sends/receives to/from other satellite nodes (and, if it receives them, concatenates and hashes) in a given interval of time, the larger the probability of finding at least one pair of concatenated ticket values that hashes to the value v or lower.

Let n be a number of trials, where each trial consists in the concatenation of a pair of ticket values and the subsequent hashing of the concatenated pair. Let p be the probability of success on a single trial and X be the number of sucesses consisting in finding a pair of ticket values that (concatenated) hashes to the value v or lower.

In order to calculate the probability of any particular number of successes, we can use the following binomial probability formula:

$$P(X)=\frac{n!}{X!(n-X)!}\times p^{X}\times(1-p)^{n-X}$$

Let us suppose that $p = 0.0001$. For 100 trials the probability of exactly one success is:

$$P(1)=\frac{100!}{1!(100-1)!}\times 0.001^1\times(1-0.001)^{100-1}$$

$$P(1)=0.090569784495867$$

To find the probability of at least one success, we need to find the sum of the binomial probabilities for each of the values of X from 1 to 100. Let the binomial probability for X=1 be S_1, S_2 for X=2, etc.

$$\sum_{i=1}^{100} Si=P(X\geq 1)=0.095207852886291$$

Mutatis mutandis, for 50 trials with the same value of p:

$$\sum_{i=1}^{50} Si=P(X\geq 1)=0.048794371802969$$

Mutatis mutandis, for 150 trials:

$$\sum_{i=1}^{150} Si=P(X\geq 1)=0.13935661731696$$

We can clearly see how the probability of finding at least one pair of concatenated ticket values that hashes to the value v or lower depends on the number of trials. In the calculations above we kept p constant to show how the number of trials translates into the probability of success. According to item 3 in section 4 of the Documentation, p would depend on how small v is, and this in turn would be calculated based on the total amount of possible combinations (pairs) of ticket values at a given time. This would ensure that the changes in the number of potential combinations of pairs of ticket values (which depends on the number of coins used to make encryption keys declarations) would not substantially affect the time it takes on average to find the proper pair.

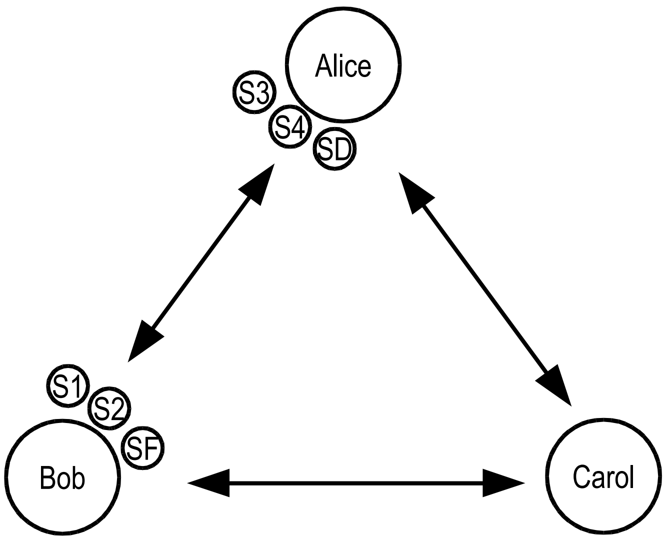

Figure 1. In Figure 1, there are three nodes-addresses: Alice, Bob and Carol, wherein each of them has a similar amount of Internet bandwidth available; and six satellite nodes: *S1*, *S2*, *S3*, *S4*, *SF*, *SD*. *SD* represents the satellite node David. *SF* represents the satellite node Frank. Alice cooperates with the satellite nodes *S3*, *S4*, *SD*. Bob cooperates with the satellite nodes *S1*, *S2*, *SF*. Note that one node-address can cooperate with many satellite nodes at the same time. The bilateral arrows represent sending/receiving ticket values. Carol does not cooperate with any satellite node. Ticket values generated using the symmetric encryption keys declared by Carol are therefore exchanged in the network at a slower pace, as only her Internet connection is used for that purpose.

5. If David finds, through the exchange of ticket values with another satellite node, a pair of ticket values that hashes to the value v or lower, then this allows for the block production. The block 6010 can be produced by David or Alice or any other node that knows the proper information. The exact procedure is described in section 5 of the Documentation.

Even if Alice produces block 6010 herself, David cannot be deprived of his part in the block production reward, according to the procedure described in item 4 of section 5 of the Documentation. Therefore, with the proper amount of the reward for the block production that goes to the satellite nodes vs to the nodes-addresses, the main factor determining the distribution of newly minted coins in the system is the amount of the used Internet bandwidth.

Let us assume that the pair of ticket values *u1* and *u2* that allows for the block production consists of two ticket values *u1* and *u2* and the first ticket value *u1* was generated by David, wherein said generation involved an encryption using the key *k1* declared (in the proper encryption keys declaration) by Alice, and the second ticket value *u2* was generated by the satellite node Frank, wherein said generation involved an encryption using the key *k2*, declared by the node-address Bob. The mentioned pair of ticket values was found through the exchange of ticket values between David and Frank. The produced block 6010 must comprise, *inter alia*, the pair of ticket values *u1* and *u2* that, after concatenation, as *u1 # u2* hashes to the value *v* or lower, which is the basis for the block production, both in the form of a plaintext and a ciphertext (as we mentioned earlier, a part of the process of generating ticket values consists of encrypting a plaintext number with an appropriate encryption key declared by an encryption keys declaration), the encryption keys *k1, k2* used in the process of generating, respectively, the ticket values *u1* and *u2* and Merkle proofs leading (in the sense of paths of hashes) from *k1* and from *k2* to the respective Merkle roots, each of which is a part of the encryption keys declaration made either by Alice or by Bob.

This way any node in the network will be able to verify (through the Merkle proofs) that:

a.) *k1* and *k2* were indeed declared by, respectively, Alice and Bob in the blocks belonging to the previous blockchain section

b.) *k1* and *k2* were used in the process of encryption that resulted in, respectively, the ticket values *u1* and *u2* (in the form of ciphertexts)

c.) the plaintext numbers that were encrypted using the symmetric keys *k1* and *k2* in order to generate the ticket values *u1* and *u2* are correct according to the protocol rules. This part of the verification includes, *inter alia*, verifying that in the process of the generation of the ticket values *u1* and *u2* the correct length of the string α was used, which translates into the total length of each of the ticket values generated by encrypting the proper plaintext number with the appropriate encryption key. The larger the plaintext is, the larger the ciphertext (in terms of the data size).

This is crucial as the larger this ciphertext, which constitutes each time a ticket value, the more data satellite nodes need to exchange (send/receive) to be able to perform the same number of concatenations of pairs of ticket values and subsequent hashing operations; so the more Internet bandwidth they use, the faster they send/receive ticket values to/from each other and statistically the sooner one of them finds the correct pair. The amount of Internet bandwidth used translates into the odds of production of the block, similarly to how hashing power fulfils its purpose in Bitcoin.

Every satellite node can generate a certain number of ticket values on its device. In order to do that, it uses some information that is public and the symmetric encryption keys that are not publically available on the blockchain (this procedure is described in section 3 of the Documentation). What makes the nodes-addresses delegate the task of generating and exchanging the ticket values instead of simply sharing the symmetric keys with each other, so that each node-address would be able to generate the ticket values on its own device? The answer to this is described in note 6 in section 5 of the Documentation. Basically, it relies on the assumption that the majority of the satellite nodes, which are invested in the protocol, will not collude to act in a dishonest way, as this would destroy trust in the protocol and the value thereof.

The method described in note 6 in section 5 of the Documentation is based on introducing "penalising transactions" into the system. Each such transaction: a) proves that a certain node knew of at least a certain percentage of the encryption keys declared by a particular node-address and b) can affect the penalised address by taking away their coins. If we assume that a sufficient majority of the satellite nodes do not collude in order to exchange encryption keys instead of sending ticket values (for the reasons mentioned above), then the amount of encryption keys, declared by a given node-address, that can be used to compose a penalising transaction cannot be (at the time the penalising transaction can be made) accessed by anyone except said node-address itself (as the encryption keys are widely distributed amongst the satellite nodes that use them to generate ticket values). Only if the node-address tries to dishonestly increase its chances of being involved in the block production (by sending its encryption keys instead of the generated ticket values) can it be penalised through a penalising transaction and loses its coins. Centralisation of the process of generating ticket values always comes with a disproportionately high risk of penalising transactions. For example, if we assume that a penalising transaction can be made using 20% of the encryption keys belonging to a particular node-address, then if this node-address sends 10% of its encryption keys to some other node, which owns 5% of all the coins frozen by all the valid encryption keys declarations in the network, then we can say that half of the required (to compose a valid penalising transaction) 20% of the encryption keys were leaked, whereas the advantage gained amounts to the possibility of generating only approximately 0.5% (10%×5%) of all the possible pairs of ticket values whose generation involves using the encryption keys of said particular node-address together with all the other encryption keys in the network.

According to note 4 in section 5 of the Documentation, satellite nodes exchange encryption keys and the Merkle proofs that prove their inclusion after a certain amount of blocks was produced since the production of the block for the production of which said encryption keys could have been used. In order for the nodes-addresses to know that the satellite nodes with which they cooperate work properly, they need to regularly request from their satellite nodes the encryption keys used to produce the pairs of ticket values that reached an output difficulty level (the term is defined in the Documentation) L or higher. L can be any suitable value that allows the nodes-addresses to verify their cooperation with the satellite nodes; however, the value of L should be set in the system so low that almost all encryption keys are disclosed (as used to produce the pairs of ticket values that have reached L) between the satellite nodes during the procedure described in note 4 in section 5 of the Documentation – as this is needed to allow for validating penalising transactions (as described in note 6 in section 5 of the Documentation). After getting to know the encryption keys of their peers, the satellite nodes can propagate the just obtained encryption keys in the network. Alternatively, simply all encryption keys are mutually disclosed by the satellite nodes during the procedure discussed here – this solution is proposed in note 4 in section 5 of the Documentation.

Each blockchain block produced has a certain "level of output difficulty". The lower the value of the hash that results from hashing the concatenated ticket values, which allowed for the block production, the higher the level of output difficulty of that block. The veridical blockchain is chosen by a node synchronising with the network using the levels of output difficulty obtained in the blocks – thus this is a source of consensus in the network. This is described, along with some calculations of the probability of a sucessful attack on the consensus, on pages 44-47 of the Documentation. Furthermore, note 12 in section 5 of the Documentation provides an explanation on how forks are resolved in the network.

The methods covered briefly in the present subsection are described in detail in sections 4 and 5 of the Documentation.

4.2. Geographically-and-functionally bounded blockchains

In the previous subsection we have described how, using the Intercon protocol, a blockchain that distributes coins (and, indirectly, power over the system) utilising the used Internet bandwidth as a factor governing the distribution can be achieved. However, such a system would in no way be less prone to centralisation than any other solution based on technological competition, *e.g.*, Proof-of-Work. Eventually, it would lead to the emergence of few key players (which benefited from the first-mover advantage, economies of scale and technological optimisation) producing blocks in the Intercon blockchain. This is precisely because this very blockchain would operate on a global scale. If we could create many blockchain networks, each one restricted to the nodes, which compete in terms of the Internet bandwidth used, localised in a certain geography, the very competition would inevitably be geographically restricted too. A node could compete on the micro-scale of the particular geography covered by a given blockchain, creating a local crypto-economy; however, all such micro-blockchains would be fully interoperable, thus they would function as a unified whole. No one local consensus could be overwhelmed by any other. Furthermore, as each such individual blockchain would be able to reach consensus not only on its own state and state transitions, but also on any occurrences/events that are, *e.g.*, pertinent to a given geography (as described in the following section), said unified whole would be a network of decentralised financial/social/political consensuses, possibly giving birth to a non-hierarchical global governance system.

A separate blockchain network that operates using the Intercon protocol is called a "sub-network." Let us suppose that a sub-network *A*, which operates in a given geography *G*, is to be created. The general design of the protocol will be the same as in the case of a geographically non-restricted (such as described in the previous subsection) sub-network. Some elements will, however, be dissimilar; those dissimilarities will be explained by way of examples:

1. The goal is to restrict the satellite nodes operating within the sub-network *A* to nodes that use only IP addresses localised within the geography *G* to send/receive ticket values to/from other satellite nodes. As the exchange of ticket values itself will always necessitate a bilateral exchange of data packets between the satellite nodes, an IP address used by a counterparty will be known by each participant of a particular exchange. Even though a given satellite node could use, *e.g.*, a proxy server, financially it would not make sense, as it would always be more expensive per the amount of data sent/received than to simply operate from within *G*.

The robustness of the solution presented here relies on two assumptions: that at least a certain share of the operators of the nodes-addresses within *A* also act in the sub-network as operators of satellite nodes; and that they, acting as the satellite nodes, send/receive a certain share of ticket values in the sub-network. It may be questioned whether such an assumption would hold true in any real implementation. For a node-address that possesses a small amount of coins (frozen in their encryption keys declaration) it may not make much sense to operate as a satellite node as well (because of, for example, barriers to entry). However, for a node-address that owns a large amount of coins, which are like a stake in the network, it makes perfect sense to invest in running a satellite node if this contributes to securing the unique property that gives the sub-network in question value – its geographical restriction. It contributes to securing the value of the sub-network that they are invested in. Furthermore, as the distribution of coins is related to the amount of Internet bandwidth used, the nodes that have a lot of coins will naturally be, in many cases, the ones that run the satellite nodes as well. For the reasons mentioned, let us assume that at least some share of the Internet bandwidth used to exchange ticket values in *A* is utilised by the satellite nodes operated by the entities that are also operators of nodes-addresses. Let us call such satellite nodes the "verifying satellite nodes."

Note 8 in section 5 of the Documentation specifies that, during each exchange of ticket values between two satellite nodes, one satellite node should send and another one should receive ticket values. This is to avoid concatenating and hashing the same pair of ticket values twice (once by each satellite node) and wasting Internet bandwidth. However, in the present context, it introduces a problem. As we want to restrict the satellite nodes in the network to specific IP addresses, we must face the fact that during any exchange of ticket values the source IP address can be spoofed – and the restriction thus circumvented; if circumventing said restriction gives a financial advantage over other satellite nodes, then no one will be incentivised to run a satellite node that receives ticket values. This can be easily fixed by splitting the task of sending/receiving ticket values between any two satellite nodes during any given exchange of ticket values; in this way, if, e.g., 1000 ticket values can be generated and sent from the satellite node *S1* to the satellite node *S2* or from *S2* to *S1*, then *S1* will send 500 ticket values to *S2* and *S2* will send 500 ticket values to *S1* – so that *S1* will send precisely the same amount of data to *S2* as *S2* to *S1* and no repetition of concatenating and hashing will occur. Each node (*S1* or *S2*) will concatenate and hash on its device different pairs of ticket values. Although the source IP address (of the sending satellite node) can be spoofed during each exchange, the destination IP address (of the receiving satellite node) cannot; as the two tasks of sending and of receiving are performed symmetrically, no special advantage is gained by any node.

2. Let us assume that, during the interval of time during which the blocks belonging to the blockchain section *B* are produced, the verifying satellite node *V* is exchanging ticket values with other satellite nodes. During each such exchange between *V* and any other satellite node (which possibly can be another verifying node; this is, however, unknown to *V*), *V* gets to know: a) the address that declared the encryption keys (through the encryption keys declaration) that are used by said other satellite node in the process of generating ticket values and b) particular numbers, called the "sequential numbers of ticket values" – those numbers, whose mode of operation is described in detail in section 3 of the Documentation, identify uniquely the encryption keys that were used by said other satellite node in the process of the generation of the ticket values that are now exchanged with *V*, but in a manner that does not disclose the symmetric encryption keys themselves (each specific range of those sequential numbers of ticket values is associated with a particular encryption key; as the number of encryption keys can be inferred from an encryption keys declaration, each range, together with the address used to make the encryption keys declaration, uniquely identifies a particular encryption key). This combination of data – the address using which a relevant encryption keys declaration was made and the sequential numbers of ticket values – are, under normal conditions, uniquely identifying any satellite node and, more importantly, any block or vote (the term "vote" is defined in the following subsection) produced using a ticket value generated utilising a specific encryption key, associated with that satellite node, and declared by a specific address.

3. *V* monitors which IP addresses the satellite nodes with which it cooperates are using during the data exchange aimed at finding pairs of ticket values allowing for the production of blocks in the blockchain section *B*. Shortly before the production of the last block in *B*, *V* creates a list *l* of all satellite nodes, identified by unique identifiers described in the previous item, that have used (at any time during the production of blocks in *B*) IP address(es) localised outside of the geography *G*. All the other verifying satellite nodes in the sub-network do the same as well. Using the lists in question, the verifying satellite nodes generate special messages that include them, so that, *e.g.*, *V* generates the message *m* with its list *l*, *V'* generates the message *m'* with its list *l'*, etc. Furthermore, if a given entity operating a node-address does not operate a verifying satellite node, it can ask one or more of its peers (the ones that it trusts) that do that for their list(s) and compose its own based on the responses. If, *e.g.*, *V'''* asks other nodes that it trusts (those may be the nodes that are generally trusted in the network and/or highly invested in it) for their lists, it can compose *l'''* (to be included in *m'''*) using the data received from them – this procedure is based on distributed trust.

4. Just before the production of the last block in *B*, the messages created by the nodes-addresses are propagated within the sub-network *A*. Each message is signed using a proper private key, belonging to a public key using

which an appropriate address (used by a given node-address to make an encryption keys declaration that allowed for the generation of ticket values during *B*) was derived. This allows every node that receives such a message to know the address it is associated with, verify the validity of the encryption keys declaration and the number of coins that were frozen through it.

5. When the messages have already been propagated, each node-address has received a similar (or the same) set of messages. Each message has a certain "weight" that depends on the number of coins that were frozen by the encryption keys declaration associated with the particular address that has originated the message. Every node-address can now compile a list of all uniquely-identified satellite nodes that were deemed to be dishonest (by not using an appropriate IP address, as described above) and compute an aggregated weight of messages pointing at each of them. If, for a given satellite node, that weight surpasses a certain threshold, a node-address compiling such a list identifies that satellite node as dishonest. This threshold must be low enough to cover dishonest satellite nodes, but high enough to allow for some margin of tolerance (as we must assume and take into account that some nodes-addresses, which created the messages described in item 4 above, can be malicious and point at honest satellite nodes). The more a given satellite node uses Internet bandwidth originating from outside of the geography *G* and the better works the mechanism of verifying satellite nodes together with distributed trust in them, the more likely said satellite node is to surpass the mentioned threshold.

6. Once the production of blocks in *B* finished, the production of blocks in *B′* has started. Every block in *B′* includes a special Merkle tree containing (as data blocks) a list, compiled on the basis of the list mentioned in the previous item, of the satellite nodes (identified using addresses/sequential numbers of ticket values, as described above) that: 1) acted dishonestly during the production of *B* and 2) were involved in the production of a block in *B* (this can be inferred by each node, which composed the list mentioned in the previous item, by comparing the list identifying the dishonest satellite nodes with the ticket values that were used to produce the blocks in *B*). Every data block in such Merkle tree contains unique identifiers, *i.e.*, the address/range of sequential numbers of ticket values, identifying the satellite node that acted dishonestly and the block that was produced using the ticket value associated with said address/range of sequential numbers of ticket values.

As described in section 5 of the Documentation, the production of every blockchain block involves signing it by a pair of nodes-addresses; those nodes-addresses must come to an agreement on the list in the form of a Merkle tree (described herein) to be included. Each such list in each block of *B′* represents a point of view, reached through an agreement by a particular pair of nodes-addresses, on which satellite nodes acted dishonestly (as an aggregated weight of messages pointing at them surpassed a certain threshold, as described in the previous item) during the production of *B* and generated at least one ticket value that was used to produce at least one block in *B* – according to the consensus of the whole network.

7. If a dishonest satellite node *M* is included in a list in the form of a Merkle tree (in the way described in the previous item) in more than a certain number of blocks forming *B′* (which represents a point of view of the entire network on the matter of dishonest satellite nodes), then (by this consensus reached in *B′* and because of it) if any block was produced in the previous blockchain section, *i.e.*, *B*, using a ticket value generated using the encryption key associated with the address and the sequential numbers of ticket values that uniquely identify *M*, then any rewards for the production of this block become nullified – not only the reward reaped by *M*. This is possible because the production of any block requires disclosing (in the block) the information needed to match the block to a particular satellite node in the way described here. In order to allow for said nullification, the rewards for the block production must be unspendable for a sufficiently large number of blocks (*e.g.*, until the end of the blockchain section that follows a coinbase transaction).

In general, it is rational for a satellite node not to act dishonestly by using an IP address localised outside of the geography allowed in a given blockchain network. If it does act dishonestly, odds are high that even if it succeeds in the competition to find a pair of ticket values that allows for block production, the reward for producing the block will never be spendable. Although the cost of exchanging data using a node from a prohibited geography may be lower, the risk of losing the reward should be a suitable deterring factor.

Even if some of the messages described above, consisting of the lists of dishonest satellite nodes and being propagated in sub-networks at regular intervals, are pointing at some honest satellite nodes (because of the malicious behaviour of some of the nodes-addresses), assuming only a sufficiently small share of the nodes-addresses is dishonest and the threshold described in item 5 above is sufficiently large, this should not affect the general consensus in a fatal way. Generally, the larger said threshold, the more of the nodes-addresses (as measured by the amount of coins they own) are assumed to be able to be malicious – but at the cost of the less efficient detection of dishonest satellite nodes (and vice versa). Combination of the use of the verifying satellite nodes and the distributed trust of the nodes-addresses that use the lists composed by them to generate and propagate the relevant messages should be enough to secure the protocol and bound it to a specific geography.

As the Intercon-based consensus achieves granularity through geographical boundedness, each single part (sub-network) serves a specific local or regional society; investments into any particular sub-network come from that geography and returns are distributed within it. It is not like in the case of Bitcoin; no special hardware can be designed by a particular manufacturer, which would centralise the distribution of the newly minted coins – no "Application Specific Internet Connection" exists; furthermore, as each node competes with the other ones that are localised in the same geography, no advantage specific to a certain geography, such as inexpensive hardware or electricity in some parts of the world, can become a factor. This allows for a more egalitarian distribution of wealth.

Presently it is believed that an ideal blockchain, which is yet to be invented, would solve the blockchain trilemma [13] of decentralisation, scalability and security through being resilient to centralisation, network congestion and security vulnerabilities. This resiliency means that even faced with the actors/events that would otherwise lead to centralisation, network congestion or security vulnerabilities, the perfect system would not break. Intercon not only solves this trilemma but completely transcends it by achieving decentralisation, scalability and security not through resiliency, but through antifragility [14].

Antifragility means that the more the system is faced with shocks, both of external and internal origin, not only does it withstand them, but also the better it becomes. In the case of the Intercon system, the more centralised consensus within a given sub-network becomes, the more likely the system is to upgrade by creating new sub-networks that cover the same geography, but with greater level of geographical granularity – to eventually outcompete the centralised one by partitioning the competition itself geographically, thus allowing for more entrants into the competition, which translates into accelerated mass adoption. If we apply this principle to scalability, we can see that the more congested the sub-network becomes, the more likely it is to be partitioned into smaller, interoperable sub-networks, thus increasing scalability. Same goes for security – with many subparts, if a vulnerability occurs within a subpart, it does not imply a central point of failure. This is impossible with modern blockchains, which are either completely unitary in nature or they introduce some central element in their architecture (*e.g.*, the Relay Chain [15] or similar).

Solving the trilemma through antifragility, coupled with the fact that the protocol can, as will be explained in the following subsection, facilitate consensus both on the occurrences that happen within it (state and state transitions) and without (external events, financial and political decisions), could possibly allow for the emergence of new kinds of social systems.

4.3. Interoperability between blockchains

In the previous subsections we have presented how, using the Intercon protocol, a blockchain-based consensus can be reached and how this consensus can be made geographically-and-functionally bounded. Let us assume that many such geographically-and-functionally bounded blockchains exist. How will they form a unified network of fully interoperable blockchains, without introducing any central, superior element into the architecture of the network?

Each sub-network can (in every one of its blocks) reach consensus on its state and state transitions. Additionally, it can reach consensus on any chosen occurrence/event, which is external to it, through a special voting procedure:

1. Let us assume that in the sub-network *E* satellite nodes are exchanging ticket values in order to produce block 1101. The pair of ticket values with a suitably high level of output difficulty is found and the block is produced. However, during the exchange of ticket values that led to this, the satellite nodes in the sub-network *E* have found some number of pairs of ticket values that only slightly missed the level of output difficulty required to produce block 1101. If this difference in the level of output difficulty is small enough, such a pair of ticket values can, instead of allowing for the production of the block, allow for the production of the so-called "vote". A vote is a special transaction that can be included in a block. This transaction must include a value related to a specific occurrence/event that the sub-network *E* reaches consensus on (concurrently with the basic consensus on the state and the state transitions within it). It can be almost anything that belongs to the domain of public knowledge and is mathematically expressible. Let this circumstance be *Cir*. Every vote is propagated in the network through a gossip protocol and is valid only for a specific block.

2. The producer of block 1101 includes the votes related to *Cir* in the block. He is incentivised to include as many votes as possible, as this increases his block reward. A median of the values related to *Cir* in all the votes in the block is included in it. Said median will be valid only if the number of votes is high enough. If the block producer would try to manipulate the median by calculating it based on an abnormally low amount of votes, that median would be invalid (for the purposes described below).

The producer of block 1101 in the sub-network *E* includes in said block the median *P1* related to the circumstance *Cir.*

3. In the sub-network *A* there is a transaction *X*. This transaction *X* somehow depends on the median *P1* reached in block 1101. In order to correctly process transaction *X*, the appropriate block producer in the sub-network *A* must obtain and provide *P1* in a secure and provable way. Let us assume that the sub-network *A* reaches consensus, in a similar way to how the sub-network *E* has reached consensus on the circumstance *Cir*, on the state and the state transitions occurring in the sub-networks *B* and *C*. It also reaches consensus on the values, related to some external circumstances, that the sub-network *B* and the sub-network *C* have reached consensus on. For example, *B* reaches consensus on the state and the state transitions in the sub-network *D*. So *A* reaches consensus on the values reached by consensus in *B*, said values being related to the state and the state transitions in *D*. All of this is described in detail in section 7 of the Documentation.

Using the method described in section 7 of the Documentation, the producer of block 1101 in the sub-network *A* can verifiably prove, through the use of paths of hashes consisting in Merkle proofs, what the consensus reached on any circumstance in any other sub-network is, no matter how many of them exist. Now he can prove *P1* and process the transaction *X*.

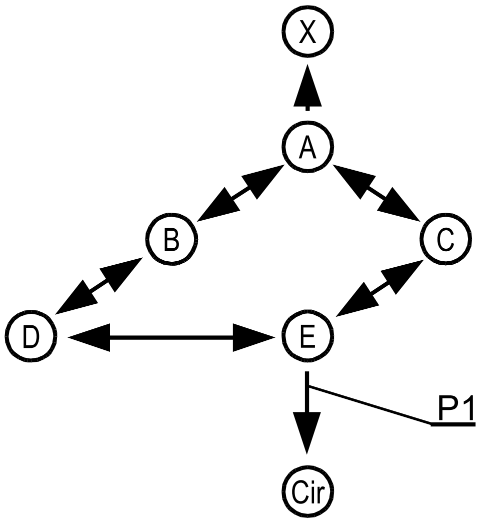

Figure 2. In Figure 2, *A*, *B*, *C*, *D*, *E* represent sub-networks. X represents the transaction *X* in the sub-network *A*. *Cir* represents a circumstance. Consensus on this circumstance is reached in the sub-network *E*. An arrow pointing from *E* to *Cir* represents the median *P1*. Bidirectional arrows represent consensuses reached by the sub-networks, in this case mutually (by *A* and *B*, *B* and *D*, *D* and *E*, *E* and *C*, *C* and *A*), about one another: about their states and state transitions, and the values related to the external circumstances on which they reach consensus.

The methods presented in this subsection are described in detail in section 7 of the Documentation. In combination with the previously covered methods, an infinitely scalable [16] network of geographically-and-functionally bounded blockchains can be created, where each blockchain can be a source of decentralised consensus on virtually any value or occurrence. This consensus, reached in any of the sub-networks, can be used to facilitate, in any other sub-network, transactions that somehow depend on it – without the need for unsecure “oracles” [17]. The whole network of sub-networks can serve both as a reliable source of truth and a facilitator of financial and social operations that rely on truthfulness of information. This translates into social and political systems that are less susceptible to corruption, misinformation and social engineering.

5. Conclusion

Implementing the Intercon system would translate into the possibility of creating a global network of interoperable, blockchain-based economic and social systems that are both in terms of the distribution of power and the distribution of wealth geographically-and-functionally bounded – thus solving the dilemma of social functionality vs coordination, which implies the concentration of power and the ensuing corruption and uncheckability thereof, described at the beginning of the present document. What the Internet is for sharing information, the Intercon could be for reaching consensus thereon.

References

[1] Askari, G., Gordji, M.E. & Park, C. “The behavioral model and game theory.” *Palgrave Commun* 5, 57. <https://doi.org/10.1057/s41599-019-0265-2>. May 2019.

[2] Yuval Noah Harari. “Why technology favors tyranny.” <https://www.theatlantic.com/magazine/archive/2018/10/yuval-noah-harari-technology-tyranny/568330/>. October 2018.

[3] Satoshi Nakamoto. “A Peer-to-Peer Electronic Cash System.” <https://bitcoin.org/bitcoin.pdf>. October 2008.

[4] https://en.bitcoin.it/wiki/Proof_of_work

[5] Sunny King, Scott Nadal. “PPcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.” <https://peercoin.net/whitepapers/peercoin-paper.pdf>. August 2012.

[6] https://en.wikipedia.org/wiki/Application-specific_integrated_circuit

[7] Alireza Beikverdi, JooSeok Song. “Trend of centralization in Bitcoin's distributed network.” <http://dx.doi.org/10.1109/SNPD.2015.7176229>. June 2015.

[8] Bram Cohen, Krzysztof Pietrzak. “The Chia Network Blockchain.” <https://www.chia.net/assets/ChiaGreenPaper.pdf>. July 2019.

[9] Greg Colvin, Andrea Lanfranchi, Michael Carter, IfDefElse. “EIP-1057: ProgPow, a Programmatic Proof-of-Work.” <https://eips.ethereum.org/EIPS/eip-1057>. May 2018.

[10] Yuming Huang, Jing Tang, Qianhao Cong, Andrew Lim, Jianliang Xu. “Do the Rich Get Richer? Fairness Analysis for Blockchain Incentives.” <https://doi.org/10.1145/3448016.3457285>. June 2021.

[11] Pascal Lafourcade, Marius Lombard-Platet. “About blockchain interoperability.” <http://dx.doi.org/10.1016/j.ipl.2020.105976>. September 2020.

[12] https://en.bitcoin.it/wiki/Difficulty

[13] Amani Altarawneh, Tom Herschberg, Sai Medury, Farah Kandah, Anthony Skjellum. “Buterin's Scalability Trilemma viewed through a State-change-based Classification for Common Consensus Algorithms.” <http://dx.doi.org/10.1109/CCWC47524.2020.9031204>. January 2020.

[14] https://en.wikipedia.org/wiki/Antifragility

[15] https://wiki.polkadot.network/docs/learn-architecture/

[16] Abdelatif Hafid, Abdelhakim Senhaji Hafid, Mustapha Samih. “Scaling Blockchains: A Comprehensive Survey.” <http://dx.doi.org/10.1109/ACCESS.2020.3007251>. July 2020.

[17] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, Davor Svetinovic. “Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges.” <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9086815>. May 2020.