TheChain: A Fast, Secure and Parallel Treatment of Transactions

Mohamed Ikbal Nacer
Bournemouth University
Bournemouth, UK
mnacer@bournemouth.ac.uk

Simant Prakoonwit
Bournemouth University
Bournemouth, UK
sprakoonwit@bournemouth.ac.uk

Ismail Alarab
Bournemouth University
Bournemouth, UK
ialarab@bournemouth.ac.uk

ABSTRACT

The Smart Distributed Ledger (aka blockchain) has attracted much attention in recent years. According to the European Parliament, this technology has the potential to change the lives of many people. The blockchain is a data structure built upon a hashed function in a distributed network, enabled by an incentive mechanism to discourage malicious nodes from participation. The consensus is at the core of the blockchain technology, and is driven by information embedded into a data structure that takes many forms such as linear, tree, and graph chains. The found related information will be subject to various validation incentives among the miners, such as proof of stake and proof of work. However, all the existing solutions suffer from a heavy state transition before dealing with the problem of a validation mechanism which suffers from resource consumption, monopoly or attacks. This work raises the following question: *"Why is there a need for consensus where all participants can make a quick and correct decision?"*, and underlines the fact that sometimes ledger is subject to maintenance from regional parties in the data that leads to partial territories and eliminates monopoly, which is the hurdle to eliminating the trusted party. The validity of the blockchain transaction comes from the related information scattered above the data structure, and the authenticity lies in the digital signature. The aim is to switch from a validator based on incentives to a broadcaster governed by an unsupervised clustering algorithm, and the integrity does lie in the intersection among regions. However, the data structure takes advantage of the Petri network regarding its suitability. Building the entire ledger in the Petri network model will allow parallel processing of the transactions and securing of the total order between the participants on the memory reference layer. Moreover, it takes account of validation criteria quickly and safely before adding the new transaction list using the graph reachability.

CCS Concepts

• **Computing methodologies ➝ Parallel computing methodologies➝ Parallel algorithms➝ Self-organization**

Keywords

Blockchain, consensus; Petri network; Transaction validation; banking; IoT

1. INTRODUCTION

Cryptopolises [28] is a world where the crypto citizen acts freely outside the bonds of the trusted authority. Blockchain has enabled cryptocurrency in real life, and it is the key to the world of cryptopolises. Blockchain is a data structure that is built upon a hashed function, then distributed among different nodes interested in its validity. The data structure is wrapped into a list of blocks linked together with sequential use of the hash function on the content, and each block uses the Merkle tree [21] to guarantee the order of transactions. Therefore, the ledger is immutable, and a minor change such as an order of two transactions requires readjusting all the hashed values. The received transactions are encapsulated into blocks and subjected to a consensus mechanism aiming to ensure that the entire network updates the distributed ledger with a valid transaction.

Primarily, the first implementation of the blockchain was a solution to process financial transactions without the participation of a trusted party. It was an integration of different techniques to secure a chain of blocks, and the first proposal for this type of chain was to guarantee the integrity of a document by keeping records of each access and providing a secure history. It was another approach to the digital safety-deposit box that suffers from a lack of privacy, bandwidth storage, incompetence, and trust [15]. Afterwards, optimisation is added to the next work by the usage of a Merkle tree [4]. Finally comes the adoption of a consensual mechanism that can eliminate the sibling attacks by reusing the HashCash proof of work (PoW) in a race setting. Applying the same technique in the other field has the potential to facilitate trade, identity verification, secure diamond grading, tracking the shipment around the world, and cross-boundary payment without fees [12]. However, the technique suffers from scalability problems, because a search over the data structure is costly, and the consensus with a permissionless network such as Bitcoin is limited to seven transactions per second [31].

This work aims to answer the question of why there is a need for consensus where all participants can make a quick and correct decision by taking advantage of the structure of the Petri net, leading to intersected regions of interest and increasing the importance of certain concepts within the network. The next section discusses the related work on consensus within blockchain technology. The third section is devoted to the introduction of the approach by discussing the proposed data structure, the validation layer and governance within the network. The fourth section compares our proposal with the works available in the literature and how it can show better performance, before finishing with a conclusion claiming the suitability for banking and micropayment for the Internet of Things.

2. RELATED WORK

The Bitcoin [22] proposal introduced the use of Hashcash [1] to deter a participant who attempts to attack the security or the liveliness of the system. The goal was to make it virtually impossible for them to invest IT resources before dealing with a massive number of nodes interested in the validity of the ledger for their financial benefits. Several works have studied the Bitcoin proposal and its vulnerability, including [18,13]. However, computer resources are the only condition for having a better probability of winning the race by solving the problem of the NP-complete puzzle box leading the consensus to rely on the honesty of 50% of the nodes [17]. This has led to the introduction of mining cartels and various selfish mining strategies related to it [7]. However, the dishonest nodes will invest in the longest chain vulnerability to alter the global belief in which version of the data structure is valid [16,31].

The proof of stake (PoS) was a solution to solve the problem of computational resources, inheriting from the PoW its randomness by implementing the Follow the Satoshi Algorithm [6]. It comes from the incentive that stakeholders such as miners within the Bitcoin network are very interested in keeping the ledger valid. However, the idea leads to a monopoly exhibited by 50% of stake value [31]. Moreover, the 'nothing at stake' attack from a random node can coordinate a long-range attack by investing in the vulnerability of following the longest chain and building side one [11].

The discussion on the adoption of the Byzantine fault tolerance BFT technique within the blockchain technology may open up a new possibility of solving it [14]. Lamport first proposed the problem on how to make the different processes reach a consensus on the order of an event. Castro et al. [9] proposed the Practical BFT that is considered the most widely used approach currently in the industry. Malkhi in [20] proposed the Flexible BFT and introduced the alive but corrupt attack, in which the attacker is interested in keeping the network alive but threatens its safety. Nevertheless, the epochs of messages that the community goes through with the elected leader have a high level of message complexity that makes it hard to implement for permissionless blockchain.

The IOTA foundation proposed the use of a dynamic acyclic graph (DAG) by removing the concept of a block and allowing a different search algorithm to find the associated information on the graph and the transaction finality does depend on a cumulative weight rule [31]. However, the splitting attack is discussed and addressed in G-IOTA [8] by proposing a new search algorithm. Moreover, the approach claims the zero-fee transaction, whereas it implements Hashcash PoW within each participant transaction. Wang et al. [30] proposed the use of ReRam, a non-volatile memory, and raised concerns about the computing resource, which can grow massively when DAG also grows.

The Petri net is a BI-Graph which has two different types of nodes. The network is constructed from a marking vector and two matrices, which are a Pre-Matrix that describes the outgoing value to the transition from the engaged places, and a Post-Matrix that describes the outgoing value from the transition to receivers' places. Also, the marking vector describes the different places with the number of the token included, and due to the network suitability for formal analyses within the real-time system, different work is built upon it to adjust it to particular use cases. The coloured Petri network is the technique of associating an identity to different values within the place. The object Petri network is an extension of the coloured network to give more formal descriptive implementation with more functionality such as abstraction and inheritance. Ramchandani in [25] proposed the timed Petri net, which is a time-oriented performance evaluation network that is defined with an association of firing duration linked to a transition.

Mathematically the Petri network is modelled as follows:

$$PN = (P, T, Pre, Post, M0)$$

P: *stands for a list of places*

T: *stands for a list of transactions*

Pre: *pre-transaction matrix*

$Post$: $[p\times T]$,*the post-transaction matrix*

M_0: *the initiation of the marking vector*

The calculation of the incidence matrix from the Pre-Matrix and Post-Matrix

$$C = post - pre \qquad (1)$$

The work in [19] discussed the different game theory analyses dedicated to the PoW; it concluded that the PoW is vulnerable to 50% attack and to various attacks which depend on the selection of the forks. Moreover, it can be subjected to latency due to the selfish behaviour of miners or pools. PoS suffers from various disadvantages such as monopoly, long-range attack, uncle's block and pool cartels [23]. IOTA approach suffers from centralisation and resource consumption that can grow massively [30]. Moreover, BFT suffers from a high message complexity toward the leader that makes it unsuitable for permissionless blockchain. Thus, all existing methods may suffer from either a heavy state transition, a resource consumption mechanism or vulnerability to attacks. According to the European Parliament, this technology has the potential to change the lives of many people. Consequently, this work tries to drop the consensus, aiming to look at the old problem from a different angle where intersected regions of interest are implemented by taking advantage of the Petri network structure and raise new possibilities of solving it.

3. THECHAIN

TheChain's objective is to develop a self-validation approach which quickly and correctly ensures the authenticity and the validity of each transaction within the chain before gossiping effectively in a network of regions. It takes advantage of the Petri network by imposing a tree structure in which the places dedicated to monitoring information linked to track balance growth of a public key, and the transition helps to apply rules leading to the construction of regions within the network. The use of graph reachability as a way of keeping track of balances is an effective way to validate transactions. The reachability concept is enabled by linking data structure to each other with memory references. Moreover, gossiping among the different nodes uses a clustering technique that identifies which nodes are more active in some regions. The intersection among regions is the key for keeping the coins counting in the network accurate. Consequently, regions with the most exchanges will always be interested in keeping each other regional ledger. The concept of leader election is abandoned for the concept of operating territories with broadcasters, and makes the reward a subject of reflection. In the case of rewards, the broadcasters will compete in the regions to create customers register and obtain a reward by guaranteeing to the customers that the transaction is public. In the absence of rewards, participants will be interested in keeping the network alive, and there will be no competition in the regions, but collaboration. However, each network broadcaster is assessed with the five measures that give the potential to be a regional coordinator, which are: confidence, solitude, rapidity, leadership of broadcasting, conscience and ensuring the concept of finality is the key for the ideal functioning of the system.

3.1 Data Structure

TheChain is a set of transactions kept in sequential order to generate a different balance account after the search. The marking vector of the Petri network, considered as an internal wallet, describes the current state or the difference, which gives an overview of the IoT data. The incidence matrix will describe the transition by translating the transaction into a credit rule to an account which is debited or credited from it. The transaction has reference variables which possess the memory addresses of the next transaction and previous types of participants' transactions to build a sequence of the linked related list. However, the system will always be searching to find the first unused transaction to calculate the related information from the attached coin objects. Secondly, it links the new block of transactions before setting the previous transactions variable with the memory reference value of the new one. *Figure 1* shows the structure of the block that contains two different types of components, which are wallets and transactions aggregated respectively into a vector of wallets named node, and a matrix of transactions named transition.

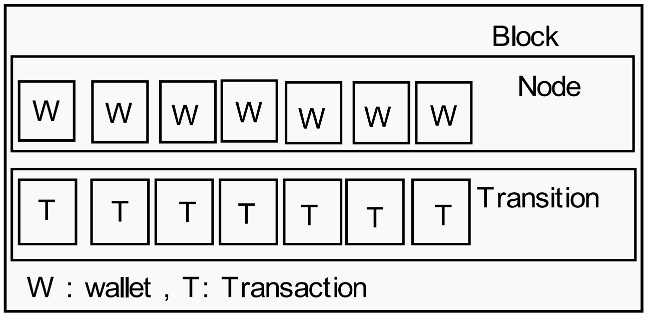

Figure 1. The Block Data Structure

A definition of the used data structure can be written as follows:

```
class Wallet:
    publicKey identity;
    double localSequentialNumber;
    Balance blnc;
class Coin:
    String identifier
    double value
    publicKey sender
    publicKey receiver
Class Transaction:
    Received timestamp
    reference sender
    reference receiver
    list<Coins> Values
```

The wallet class contains an identifier, a local sequential number that gives a precise number to how many transactions were applied by this identity, and a balance. The class coin contains a unique identifier attached to it besides its value, the last sender and receiver. A transaction is constructed from a list of coins, timestamps and memory references for management purposes.

3.2 Transaction Validation

Blockchain network produces a large number of partially independent transactions recorded together within blocks before making the whole chain the subject of a search for related information for validation. The process is burdensome, and the need to validate the whole chain to append a new block is inefficient. Therefore, a validation layer capable of handling parallel processing of transactions quickly and correctly to disseminate the validity of the block to peers is necessary for scalability purposes. *Algorithm 1* made use of the graph reachability to verify if the graph can reach that state with available criteria.

Algorithm 1 validation

```
input :listOfTrs
output :listOfvalid, VectorMarking, IncidenceMatrix
wallets ← [ wallets ×N];
Transactions ← $public_{key} \times Tranferred_{Value}$;
$Pre_{Matrix}$ ← keys× $Transferred_{Value}$;
$Post_{Matrix}$ ← keys× $Tranferred_{Value}$;
$vector_{Marking}$ ← findBalance(listpfTrs)
listOfTrs, listInvalid, c←verifyTrs(listOfTrs);
BlockIp(listInvalid)
$Pre_{Matrix}$, $Post_{Matrix}$ ← buildMatrices(listOfTrs)
$Incidence_{Matrix}$ ← ($Post_{Matrix}$ - $Pre_{Matrix}$)
While( i<columnSize($Incidence_{Matrix}$) )do
    $Vector_{Temp}$ ← $Vector_{Marking}$ + $Incidence_{Matrix}$ [i]
    if (NotAllPositive(temp)) then
        DropNegative($Incidence_{Matrix}$, listOfTrs, $Vector_{Temp}$)
    end
    $Vector_{Marking}$ ← $Vector_{Temp}$
    i + +
end
```

In *Algorithm 1*, the information from the list of received transactions is used to produce a vector and three matrices. It uses two defined data structures, which are the wallet and the transaction. The wallet must contain a balance, which represents the value of the aggregated coins created to the attached public key. The transaction contains the timestamp, the identity of the receiver, sender and the transferred value. The algorithm starts verifying the validity of the transactions, such as the digital signature, time attached and existence of the sender. It generates two lists of valid and invalid transactions and flags the IP addresses from which invalid transactions are received in addition to *c*, a confidence value discussed later. *Findbalances* is a method to return from the tree a partial marking vector related to the attached public keys.B*uildMatrices()* generates from the transactions two matrices used to calculate the incidence matrix. A matrix is a vector of vectors. Therefore, the loop continues to add each element to the temporary marking vector until the end. Within the loop, the checking of the temporary vector from negative value is held, and the function *DropNegative(,)* will drop the change and abandon the associated transaction to it. If all elements are positive, the temporary vector is affected to the marking vector. Moreover, for search limitation reasons, the public keys will be mapped into internal numbers. The process of validation of transaction is done through model checking by verifying the possibility of the graph to reach this state with the available criteria. However, each node nests a pool of transaction, with time processing that will vary with geographic distance dependability.

In [29] Nick Szabo discussed previous work by Wei Dai, with additional usage of cryptographic techniques which aim to automate the contractual relations because of the ability to virtualise the organisation, the intellectual and physical properties as entities within a distributed system. Ramchandani in [25] proposed the timed Petri network by attaching the value of time to model the temporal dynamic behaviour of a system. The contract proposed by Szabo and implemented by the Ethereum foundation functions as a proxy interface within the distributed system. This work injects sleeping Programming Threads within the validation layers and the associated time to apply the rule, which leads to the periodic application of the algorithm on the associated public keys.

Table 1. Validation Tree

Tree Trend	Transaction Matrix and wallet vector
0	VM=[[1,2,3,4,5, 6] [100,100,100,100,100,100]]
01	T M = [[1 -30 -20 5] [2 20 10 -5] [3 10 10 0]] VM =[[1,2,3] [55, 125, 120]]
00	T M = [[4 -20 0 10] [5 20 -30 0] [6 0 30 -10]] VM =[[4, 5 ,6] [90, 90, 120]]
011	T M = [[2 -30 0 0] [3 30 -20 -10] [7 0 20 10]] VM =[[2,3,7] [25, 95, 30]]
000	T M = [[5 -40] [7 40]] VM =[[5,7] [50, 70]]
0000	1. Sleep(1 Month) 2. Apply (TM = [[5 -40] [7 40]]) 3.back to 1

Table 1 shows the growth of the tree within the validation layer. Element *0* contains six wallets; each public key mapped to an internal identification number associated with the balance. Element *01* contains the application of the incidence matrix with the use of masking and indexing on the partial vector, which contains three wallets to generate a new vector for this state. The transition matrix is a vector of vectors that always contain as the first element the internal identifier, followed by the balance gain for each transaction, wherein the vertical side describes the transaction with gain to each identity. The same application is applied to element *00*, but in the element *011* the injection of a new public key is mapped to the value 7, and the system processes the new account by injecting it beside the most relevant balances, along with the sender. The goal is that applying the same philosophy leads to regions on the graph identified as a separate component. The earlier elements of the tree are periodically removed as they come with no use to validate the next transaction. A contract is a Thread embedded within the tree, and the element that falls under the branches *0000* is a contract that applies the transaction matrix every month.

3.3 Injection Layer

The validation layer wrapped the validated partial vector with the associated transaction in a block and distributed it to the responsible nodes, and internally to *Algorithm 2*. The injection layer is a persistence layer within the node, and it must secure the backup of transactions on the hard drive. Each transaction refers to the next transaction with the same identity, and the network declares the finality of the transaction when it converges on the total order for a certain global and local sequential number. Moreover, a hash table maps the identification number to a data structure defined in the class tracker, which contains the public key, the references to the first transaction, and the last injected transaction. The appending block algorithm begins by finding references to each identity on the list of transaction. In the case of the sender, it will change the first transaction reference in the tracker object, as the first coins will be used. Moreover, it will change for both the sender and the receiver the last reference as transactions added and update wallets.

```
Class tracker:
  public_key identity
  reference first
  reference last
```

Table 2. Reference HashTable

Internal identifier	trackers
158	Tracker1
25	Tracker2
856	Tracker3

Algorithm 2 Appending

```
input :listOfTrs, Graph
output : Graph
i←0;
while (i<columSize(IncidenceMatrix)) do
        $tracker_r, tracker_s$←getTrackers(t)
        UpdateFirst($tracker_s$)
        UpdateLast($tracker_s$,, $tracker_r$)
        list← [$tracker_s$, $tracker_r$]
        UpdateHashTable(list )
        i + +;
end
HashValue←MerkleTree(listofTrs, timestamp)
$block_{previous}$← ComponentPower(Graph)
B← Block(listOfTrs, $block_{HashValue}$, $block_{reference}$)
Graph.add(B)
```

The *Algorithm 2* will receive from the first layer an ensemble of information regarding the validity of transactions. It will go through each identity separately and get their tracker from the reference hash table demonstrated in *Table 2*. *UpdateFirst* is applied to the sender tracker to change the first reference variable as the first coins are used. The *UpdateLast* method is applied to both the receiver and sender to refer to the last appended transaction. The *UpdateHashTable,* updates references for the next search before building the block in *Block()* that calculates the most relevant previous block in the graph by ComponentPower before adding it to the graph. ComponentPower will not be expensive because of the assumption that each cartel will maintain a region. *Figure 2* is an architectural demonstration of the decision chain within the ledger, in which Mv stands for a partial marking vector, IM for incidence matrice.

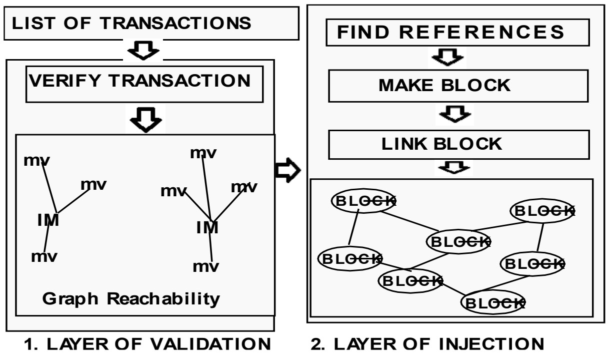

Figure 2. Decision Chain

3.4 Maintainers' Governance

Governance is the art of orchestrating nodes to work together to finalise transactions by identifying regions of exchange. However, the whole system is managed by how many coins with a unique identifier do exist. Consequently, adding to the fact that regions intersect makes it easier to track any fake coin. Moreover, a linear registry will work with DNS, in which it stores the history of IP addresses that have committed malicious behaviour in the past. Each block contains information about a node with proof of its previous behaviour. Consequently, it will derive a table for networking information. In addition, each node of the system is exceptional, with material resources and efficiency based on the following metrics:

$$solitude = \left(\frac{1}{devices}\right) \quad (2)$$

$$rapidity = Validation_{CPUPerCicle} \quad (3)$$

$$\text{power} = \frac{mi\,n(size(thp), Mspace)}{Rapidity(pr)} \quad (4)$$

$$conscience(node) = \frac{numberOfBlock(publicKey)}{numberOfBlocks(none)} \quad (5)$$

$$Confidence(block) = \frac{\sum_{i=0}^{N} verifySignature(t) \times validate(t)}{N} \quad (6)$$

thp: Throughput to the system

size: The memory space held by the throughput

Mspace: The Alive memory space

Pr: The peer identifier

numberOfBlocks: Number of a block with the associated public key

The confidence metric aims to evaluate the validity of the block, and for each transaction, it gives a boolean whether it succeeds or not. At the block level, it will produce a percentage *c* value disseminated in the event of fault behaviour to be recorded with the block in the DNS linear register. The solitude is a metric that evaluates the node by its reliability on the external computing machine. Moreover, the rapidity metric does depend on hardware devices available to the validator and the used programming language; however, in the initial stage, the value will be CPU clock dependent. The leadership of broadcasting (power) consists of assigning responsibilities and nodes with superior equipment to be candidates for the role of a star after having also evaluated their consciences. Conscience means the number of blocks processed and broadcasted by the node. The values will be calculated periodically and disseminated to the peers to readjust the network nodes by the governance algorithm described in *Algorithm 3.*

The intersection among regions is key for not exhibiting any elimination behaviour from one part of the cartel against another. The system grows gradually from a few maintainers that nested client directory and kept serving them by making their transactions public, to cartels that are responsible on a regional exchange, in which each maintainer is responsible on a partial part of the region. Maintainers will grow to understand that their advantage lies in cooperation with each other because each region is serving clients, but there is always a dependency on other regions.

Algorithm 3 AssignResponsability

```
input: chain
output: rankedResponsible
List←getList(chain,[blockid, recipient, sender, validator])
G←createGraph();
G.addEdges(TupleList(list[sender, recipient]));
Components←G.getConnectedComponents();
parse(DNSledger, List)
responsible ← Intersect(components, List);
rankedResponsible←rank(responsible, listOfProperties);
```

Algorithm 3 begins by filtering the data in the chain, which contains only the transactions of the last two months, and obtains the characteristics of the decision. Later, the penalty DNS registry will be continuously analysed in the data to generate a list that eliminates any previous malicious node from participation. *Intersect* will generate a list of maintainers that crosscheck the associated components before comparably classifying them into the three leaders in diffusion, consciousness, and solitude metrics. The region assumption is based on the centralisation of the graph on some data point. However, regions will be interested in maintaining other regions due to some exchange of values between them. In the case of reward, it will be set automatically by the governance algorithm or manually by the user who is responsible for validating the transaction.

3.5 Node Independency

TheChain objective is to build self-validating nodes that are enabled with a layer of validation for fast and parallel treatment of transactions. The network is governed by an algorithm that builds intersected regional maintainers. The proposal dropped consensus, which means the absence of convergence on a unique ledger. The normal function of the system is by setting a limited number of coins with unique identifiers that will be exchanged between the different users. Maintainers will operate in their region to make their customers' transactions public in exchange for a reward. It is in the interest of all the nodes to be up to date with the different exchanges to eliminate any fake coin generation. However, all the nodes will not be recording all regions' transactions due to the limited resources, but the closest regions keep up to date with the next regions' ledger, to build a complex sequence of regions that watch over the next to secure integrity, as illustrated in *Figure 3*.

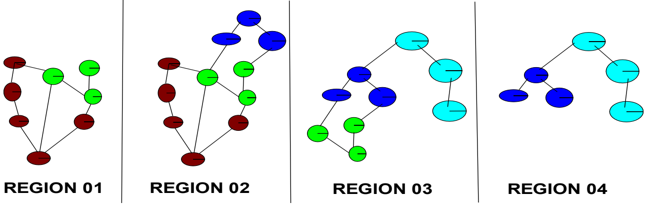

Figure 3. Regions

The nodes are independent of any exterior dictation of data, consequently eliminating any double-spend or fault injection of data that have a high impact on the network as a whole. Moreover, nodes operate in regions that lead to the elimination of any attacks that target the network liveliness.

4. DISCUSSION

4.1 Safety Attacks

This part discusses the type of attacks that intend to threaten the safety of the system, which means the robustness to secure regular operation. It focuses on a convergence of nodes on the same chain to finalise several transactions. The longest chain rule within the linear ledger approach leads a malicious node to invest in its vulnerability by aiming to create a side chain to double-spend. The attacker starts by sending a coin, treated and appended, while spending the same coin in a different place within a side chain. The attacker keeps working on the side chain to make it longer before making it public, consequently cancelling the transactions within the previous one. However, in TheChain nodes do not accept ledgers but a list of blocks that passes the same process of validation before appending it. The usage of referencing of the transaction into a linked series allows the different nodes to identify the double-spend while converging to finalise the last transactions, because it will yield to a unique local and global sequential number.

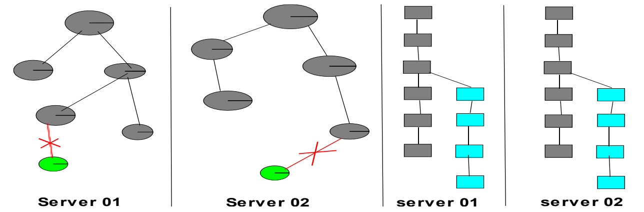

Figure 4. TheChain vs LongestChain

Figure 4 shows the difference between the two approaches. It shows as well that the graph could take many forms. However, the total order of the chain is secured on the memory reference layer in which all the transactions are appended sequentially and generate a precise local sequential number and a balance. Different kinds of attacks are investing in the longest chain vulnerability, such as a sibling attack that invests in creating many identities within the system and manipulates different peers' table to discover their neighbours. Nevertheless, the metric of confidence with a hard penalty makes this kind of attack inefficient.

Another type of attack tries to invest in the vulnerability of the search algorithm over DAG or a tree structure. The Tangle [24] runs over a DAG by adding a transaction called 'tip' before waiting for another user to append a transaction and validate two previous tips. The vulnerability is that if an attacker adds two conflicted tips in different leaves, this leads to what is called the splitting attack [8]. TheChain addresses this problem with two technical choices, firstly by searching for the latest transaction to link the new one to it, and secondly by the validation layer which builds a balance from the different coin objects and ensures the validity before appending it.

Figure 5 is for demonstration purposes, to define how the whole network is linked. Even the clients are connected via a gossip protocol to keep up to date with DNS ledger. Nodes have the capability of communicating out of their cluster. On the other hand, different selfish attacks in Bitcoin ideology will be advantageous among different nodes, such as Block withholding [2], pool hopping [5], and selfish miners [13]. However, this work is invested in such behaviour in order to build clients' directory for the cartel maintainer, in which each cartel must secure the finality of the transaction by making it public to get rewards, leading the system to function as a combination of many transfer companies.

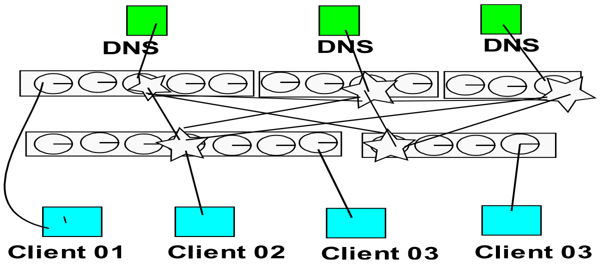

Figure 5. Network

4.2 Liveliness Attack

The work in [17] introduced the Goldfinger attack, where platform competitors try to fail the system by different means; the paper discussed the vulnerability of a 51% attack within Bitcoin. However, as discussed in the previous part, the mining cartels concept, combined with partial responsibility, leads to the assumption that validators must be cooperative in exchange for a periodic reward. The illustrated network in *Figure 6* shows how a malicious node must obtain the cooperation of all broadcasters to pass the fault transaction because of the inconsistent local and global sequential number that it will produce. However, if the number of nodes increases, the message complexity will increase and may lead to time delay due to difficulty to converge into a unique regional graph. Consequently, the resilience of such liveliness attacks will increase [10].

The attacker may choose to attack stars nodes in order to fail the system temporarily. However, the massive number of nodes will lead to a large number of star governments by an algorithm, and all the nodes in cartels will converge to the next in the waiting list if the main star fails. It must be clear that stars are not leaders of validity as used in BFT, but leaders of broadcasting.

4.3 Results

This section starts by giving proof of convergence to the solution. Secondly, it discusses the monopoly issue in previous works and how the proposed system has fixed this. It demonstrates as well how the system deals with a huge number of malicious nodes.

Based on Godel's completeness theorem [32], few axioms and rules are needed to prove a formula. Furthermore, based on the universal generalisation theorem (UG), if an element of the disclosure universe has proved an assumption with a chain of rules deduced from axioms, this means the proposition applies to all elements.

4.3.1 TheChain Proof of Convergence

If a node (e) receives an invalid block (b) it will flag the sender:

$$\forall e \in D, \forall\, b \in \; B\; invalid(e,b) \longrightarrow \; flagSender(e,b) \wedge updateDNS(e,b) \; (7)$$

Firstly, assume the network is with a small number of nodes. Consequently, the time to receive a block (b) is neglected between nodes ($e_{1,}\, e_2$) (9). Moreover, if a block is invalid in e_1 it will be invalid for all other nodes (8)

$$\forall e \in D, \exists e_1 \epsilon D, \exists b \in B, \quad invalid(e_1,b) \longrightarrow invalid(e,b) \; (8)$$

$\forall e_{1,}\, e_2 \in D, b \in B\,,\; TimeReceived(e_{1,},b) \Leftrightarrow TimeReceived(e_{2,},b)$
(9)

Based on UG, if an element reaches that assumption and all nodes share the same proprieties, then it is valid for all of them by converging on the same domain (D) (11):

$$D = D - r \qquad (10)$$

$$\exists e_1 \epsilon D\;, \forall e_2 \in D,\; updateDNS(e_1,\, r) \longrightarrow Conv(e_1,\, D_{E1}),\; Conv(e_2,\, D_{E2}) \qquad (11)$$

The network will grow massively, leading (9) to not hold any more. However, the governance algorithm will lead to direct contact between the star nodes (s), and by the addition of (12), the UG is valid again.

$$\forall e \in D, \exists\, s \;\in D,\, knowledgeable(s) \rightarrow knowledgeable(e) \; (12)$$

In *Algorithm 3*, a sorting list of nodes is returned, and each participant must pick the first *B* element as stars. If the probability of success in sending to the cluster is *P* and failure is *F*, picking *B* nodes receives the transaction to broadcast will make it with a probability of success as:

$$P \;=\; 1 \;-\; F^B \qquad (13)$$

Lastly, adding the following assumption that stated regions (r) would intersect and force nodes to be up to date with their regions:

$$\exists\; e_1, e_2 \;\in D \exists\; r_1, r_2 \in R, Interest(r_1, r_2) \longrightarrow regionUpdated(e_1, e_2) \quad (14)$$

$$\forall\; l \in L, \exists n_i D, ManyPossess(n_i, l) \longrightarrow \text{integrity(l)} \quad (15)$$

Based on (12) and (14), and by the addition of (15), which says if many possess the information, then there is integrity, TheChain will converge in a regional way, making every node possess a special graph that contains its region and the intersected with it showing high integrity.

4.3.2 Proof of No Monopoly in TheChain

Automatically, the client will be assigned to a cartel. However, it was stated that the role of the broadcaster is to make the transaction public in exchange for the reward, and a client (*c*) can also choose the validator (*e*) manually.

$$\forall c \in C, \forall e \in D, chose(c,e) \longrightarrow \; validate(e) \qquad (16)$$

Moreover, if a validator (e_1) is up to date then it will validate, get a reward and be interested in the ledger validity.

$$\exists l \in L, \forall e_1 \in D, \forall t_1 \in T, UpToDate(e_1) \longrightarrow \; Public(e_1, t_1) \wedge reward(e_1) \wedge maintain(e_1, l) \; (17)$$

In conclusion, the monopoly of the system by small nodes cannot exist, as their function is limited to make sure it is up to date and makes the transaction public in exchange for a reward

4.3.3 The Longest Chain Rule

1. If a node (n_2) has more means than any node (n_1), this means n_2 generates more blocks than n_1

$$\exists\; n_2, \forall n_1 \in D, means(n2, n_1) \longrightarrow generateMore(n2, n_1) \wedge leadMore(n2) \qquad (18)$$

2. A miner maintains the ledger for a reward.

$$\forall n_1 \in D, \exists b \in B, generateBlock(n_{1,}, b) \wedge AppendedInLedger(b) \longrightarrow getReward(n_{1,}) \quad (19)$$

3. The choice between two valid ledgers (L1, L2) is for the longest

$$\forall\; \mathrm{L_1, L_2 \in L, valid(L_1) \wedge valid(L_2) \wedge Bigger(L_1, L_2)} \longrightarrow \; \mathrm{choose(L_1)} \quad (20)$$

4. A transaction ($t_{1,}t_2$) can be valid in one ledger ($l_{1,}, l_2$) and invalid in another

$$\exists l_{1,}, l_2 \in L, \exists\; t_{1,}t_2 \in T,\; validTransaction(l_1, t_{1,}) \wedge validTransaction(l_2, t_{2,}) \wedge \neg validTransaction(l_1, t_{2,}) \wedge \neg validTransaction(l_2, t_{1,}) \; (21)$$

5. A node (n2) is interested in maintaining the version of the ledger where it gets the reward.

$$\forall n_2 \in D, \forall\; L_1, L_2, ContainReward(\; L_2, n_2) \wedge \neg containReward(L_1, n_2) \longrightarrow maintain(L_2) \quad (22)$$

Based on (18)(20)(22), the network can be monopolised by the nodes that have the majority resources because of the intention to keep getting rewards.

4.3.4 IOTA Tangle Approach

First, assume that the blockchain is a combination of miners and stakeholders.

$$\forall n_1 \in D\; l_1 \in L, network(L) \longrightarrow stakeholders(n_1) \vee miner(n_1) \; (23)$$

In the Tangle, there is no reward to the maintainer, but users validate transactions; consequently, maintainers are not interested in being part of the platform.

$$\forall\; n_1 \in D, \neg reward(n_1) \rightarrow \neg mining(n_1) \;\; (24)$$

The blockchain makes the transaction public among many maintainers to increase its integrity by eliminating the monopoly.

$$\forall\; l \in L, \exists n_i D, ManyPossess(n_i, l) \longrightarrow \neg\text{Monopoly(l)} \quad (25)$$

By the addition of the assumption that a SmallPossess is the negation of manyPossess (26).

$$\forall\; l \in L, \exists n_i D, \qquad SmallPossess(\mathrm{ni, Li}\;) \Leftrightarrow \; \neg\mathrm{manyPossess(ni, Li}\;) \; (26)$$

Based on (25) and (26)

$$\exists\; l \in L, \forall n_i D, SmallPossess\; (n_i, l) \longrightarrow \text{monopoly(l)} \quad (27)$$

In conclusion, IOTA tangle, by not providing a reward to a random validator, is conceptually vulnerable to monopoly, and from (15) and (27) there is no integrity either.

4.3.5 Scenario

The following scenario in *figure 6* was implemented and tested to demonstrate the case if all but one element of the regions start acting maliciously. It was created by running four nodes to communicate with a set of blocks where a transaction intended to double-spend, while the high probability of generating blocks on the right side due to the use of PoW has forced node 1 to settle for the red register. TheChain has dealt with the problem, as shown on the left after communicating the local and global sequential number.

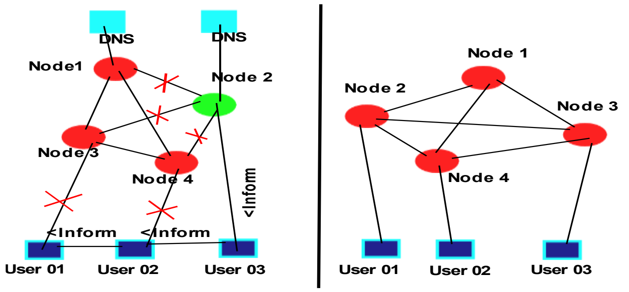

Figure 6. TheChain vs Longest Chain rule

Although the example was for the internal behaviour of a region, regions as a whole will grow to understand that any proof of malicious behaviour will cost them their customer to the intersected regions.

4.3.6 Comparison

Table 3. Comparison

	Proof of Work	**Proof of Stake**
Network failure	50% hash power [17]	50% stake value [31]
Fault data injection	50% hash power [27]	50% stake value[31]
Double spend	Longest Chain [16]	Long-Range [11]
Transaction finality	Longest Chain [31]	Longest chain [31]
	IOTA	**TheChain**
Network failure	1/3 of network hashing power	Nodes are independent
Fault data injection	Splitting attack [8]	Nodes are independent
Double spend	Splitting attack [8]	Nodes are independent
Transaction finality	cumulative weight rule [31]	Initially 100% of stars

Table 3 shows a comparison of TheChain with other consensus-based approaches. Consensual approaches aim to converge on one version of the general ledger at a time. These approaches are vulnerable to monopoly and manipulation due to dependence on resources, stakes or votes. Therefore, PoS and PoW are subject to fault data injection, network failure and double-spend by the monopolist due to the reliability on the longest chain rule, as explained in subsection 4.3.3. IOTA has a central feature, which makes it vulnerable to more than the third attack alongside the data structure choice which is vulnerable to double spending and the injection of fault data by the monopolist via the splitting attack. However, independence in TheChain is by stopping the convergence on one ledger to eliminate any dictation of data and investing in the broadcaster intention to make the transaction public to secure a reward.

The transaction finality depends on the longest chain convergence in PoW, PoS or the cumulative weighting rules in the IOTA approach. However, in TheChain is based on broadcasting transactions initially to all the stars before converging on the most relevant regions.

The integrity of TheChain and PoW is very high due to the openness to any participants and the motivating reward to maintain the ledger. Consequently, there is a high distribution among nodes. Privacy is the state of being free from public attention; the various consensual approaches, as well as TheChain depend on public and private keys to validate a transaction. On the other hand, the problems of the link between the public key and the real identity, as well as the right to be forgotten are confidentiality problems; these problems arise from the data structure and the networking choices. Therefore, *Table 4* shows high privacy of all techniques for validating a transaction within blockchain technology. Finally, IOTA tangle and TheChain operate over a graph that enables parallel treatment of transactions leading to high scalability.

Table 4. Criteria comparison

	Integrity	Privacy	Distribution	Scalability
Proof of work	High	High	High	Low
Proof of stake	Low	High	Low	Medium
IOTA approach	Low	High	Low	High
TheChain approach	High	High	High	High

4.4 Criticism

Although nodes must be up to date to attract more customers, some regions will act selfishly by abandoning the processing of some other regions' transactions, due to the zero exchange of money between the two parties. Consequently, it declares the lack of theory on the regions' behaviour. Thus, the next work will invest in the region's intersection to build a solid theory for the relationship between region size and the network, besides enabling the multi-label classification to attach one transaction to many validators. This ensures that the whole network is an intersection of many regions that are partially watching each other to increase integrity and get neighbours' customers in case of malicious behaviour.

5. CONCLUSION

This work is suitable to the IoT sector and banking systems due to the introduction of partial responsibility on the ledger that leads to territories, besides the zero computational fee invested on transaction validation. To sum up:

- Taking advantage of the Petri network structure to build the ledger and enable the total order among the participant on the memory reference layer, leading to the elimination of attacks based on forking.
- Validation layer that uses the graph reachability to enable fast and parallel treatment of the transaction.
- Introduction of the concept of region intersection to ensure the validity of the ledger.
- Definition of a governance algorithm that keeps clustering to leads to a rapid convergence within the network.
- Comparison of the proposal with previous work found in the literature.

6. REFERENCES

[1] Adam Back and others. 2002. Hashcash-a denial of service counter-measure. (2002).

[2] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. 2016. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security 12*, 8 (2016), 1967–1978.

[3] Alejandro Baldominos and Yago Saez. 2019. Coin. AI: A proof-of-useful-work scheme for blockchain-based distributed deep learning. Entropy 21, 8 (2019), 723.

[4] Dave Bayer, Stuart Haber, and W Scott Stornetta. 1993. Improving the efficiency and reliability of digital time-stamping. In *Sequences Ii. Springer*, 329–334.

[5] Marianna Belotti, Sofiane Kirati, and Stefano Secci. 2018. Bitcoin pool-hopping detection. In *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI).* IEEE, 1–6.

[6] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. 2014. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review* 42, 3 (2014), 34–37.

[7] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *In 2015 IEEE Symposium on Security and Privacy.* IEEE, 104–121.

[8] Gewu Bu, Önder Gürcan, and Maria Potop-Butucaru. 2019. G-IOTA: Fair and confidence aware tangle. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 644–649.

[9] Miguel Castro, Barbara Liskov, and others. 1999. A correctness proof for a practical Byzantine-fault-tolerant replication algorithm. Technical Report. Technical Memo MIT/LCS/TM-590, MIT Laboratory for Computer Science.

[10] Kaushal Chari. 2003. Model composition in a distributed environment. (2003), 399–413.

[11] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. 2019. A survey on long-range attacks for proof of stake protocols. IEEE Access 7 (2019), 28712–28725.

[12] DN Dillenberger, Petr Novotny, Qi Zhang, Praveen Jayachandran, Himanshu Gupta, S Hans, D Verma, Supriyo Chakraborty, JJ Thomas, MM Walli, and others. 2019. Blockchain analytics and artificial intelligence. *IBM Journal of Research and Development* 63, 2/3 (2019), 5–1.

[13] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security. Springer*, 436–454.

[14] Vincent Gramoli. 2017. From blockchain consensus back to byzantine consensus. Future Generation Computer Systems (2017).

[15] Stuart Haber and W Scott Stornetta. 1990. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography. Springer*, 437–455.

[16] Jehyuk Jang and Heung-No Lee. 2019. Profitable double-spending attacks. arXiv preprint arXiv:1903.01711 (2019).

[17] Joshua A Kroll, Ian C Davey, and Edward W Felten. 2013. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In Proceedings of *WEIS*, Vol. 2013. 11.

[18] Kevin Liao and Jonathan Katz. 2017. Incentivising blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security. Springer*, 264–279.

[19] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. 2019. A survey on applications of game theory in blockchain. arXiv preprint arXiv:1902.10865 (2019).

[20] Dahlia Malkhi, Kartik Nayak, and Ling Ren. 2019. Flexible byzantine fault tolerance. *In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 1041–1053.

[21] Ralph C Merkle. 1980. Protocols for public key cryptosystems. In 1980 *IEEE Symposium on Security and Privacy.* IEEE, 122–122.

[22] Satoshi Nakamoto. 2019. Bitcoin: A peer-to-peer electronic cash system. Technical Report. Manubot.

[23] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. 2019. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. IEEE Access 7 (2019), 85727–85745.

[24] Serguei Popov. 2016. The tangle. cit. on (2016), 131.

[25] Chander Ramchandani. 1973. Analysis of asynchronous concurrent systems by timed Petri nets. Ph.D. Dissertation. Massachusetts Institute of Technology.

[26] Elias Rohrer and Florian Tschorsch. 2019. Kadcast: A Structured Approach to Broadcast in Blockchain Networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 199–213.

[27] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, DaeHun Nyang, and Aziz Mohaisen. 2019. Exploring the attack surface of blockchain: A systematic overview. arXiv preprint arXiv:1904.03487 (2019).

[28] Melanie Swan. 2018. Blockchain enlightenment and smart city cryptopolis. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*. 48–53.

[29] Nick Szabo. 1997. Formalising and securing relationships on public networks. First Monday 2, 9 (1997).

[30] Qian Wang, Tianyu Wang, Zhaoyan Shen, Zhiping Jia, Mengying Zhao, and Zili Shao. 2019. Re-Tangle: A ReRAM-based Processing-in-Memory Architecture for Transaction-based Blockchain. In *2019 IEEE/ACM International Conference on Computer-Aided Design* (ICCAD). IEEE, 1–8.

[31] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. 2020. A survey of distributed consensus protocols for blockchain networks. IEEE Communications Surveys & Tutorials (2020).

[32] Stephen C Kleene. 1976. The work of Kurt Gödel. The Journal of Symbolic Logi