GuRuChain: Guarantee and Reputation-based Blockchain Service Trading Platform

Mouhamed Amine Bouchiha
L3i Laboratory
La Rochelle University
La Rochelle, France
mouhamed.bouchiha@univ-lr.fr

Yacine Ghamri-Doudane
L3i Laboratory
La Rochelle University
La Rochelle, France
yacine.ghamri@univ-lr.fr

Mourad Rabah
L3i Laboratory
La Rochelle University
La Rochelle, France
mourad.rabah@univ-lr.fr

Ronan Champagnat
L3i Laboratory
La Rochelle University
La Rochelle, France
ronan.champagnat@univ-lr.fr

***Abstract*—Blockchain (BC) is a promising Distributed Ledger Technology (DLT) that has attracted attention in recent years due to its characteristics of maintaining privacy and security. This technology, which removes the need for a trusted third party, can be applied in various trust-related domains, such as energy trading, crowdsourcing, and the Internet of Things. This paper presents GuRuChain, a platform framework for online service trading. It combines on-chain and off-chain trust management to provide a blockchain system suitable for trust-related applications. The platform does not only ensure the recording of online transactions; but also the deliverance of offline services. In GuRuChain, we propose a mathematical model to assess the trustworthiness of each participant. We also introduce an incentive mechanism based on reputation and guarantee to monitor the participants' behavior. Finally, we maintain the consistency of the BC through the proposed consensus scheme called *Proof Guarantee and Reputation (PoGR)*. PoGR selects nodes based on a scoring formula that uses three parameters, reputation score, guarantee balance, and risk taken to ensure fairness. The results of the performance evaluation demonstrate the feasibility, efficiency, and scalability of GuRuChain.**

***Index Terms*—Reputation, Distributed Ledger, Lightweight Consensus, Real-world Correlation, Fairness, and Reliability.**

I. Introduction

RECENTLY, Distributed Ledger Technologies (DLTs) have received a lot of attention, both from academic and business perspective. They have been adopted by various application domains such as supply chain [1], crowdsourcing [2], and Internet of Things [3]. Blockchain (BC), the widely used technology of DLTs., can be seen as a sequence of blocks chained together. Unlike the centralized network structure, there are no fixed central nodes (servers) in BC-based networks. The system is built on a typical peer-to-peer (P2P) network, allowing the distribution of data processing tasks among peers. Through a consensus mechanism, information stored and data generated by each node are synchronized [9]. The BC peers try to maintain an identical "Records Copy" locally, called **Ledger**. The records/blocks in the ledger are a set of causally related behaviors/transactions of all the participants in the system. The identical local ledgers held by all peers represent, therefore, a whole **Common Ledger(CL)**.

The system permanently stores the transactions that have been received and verified. The CL of the BC system thus keeps long-term records of all participants' actions. The stored data can be checked at any time and can therefore be used as eternal proof. Nevertheless, the BC itself has no concept of trust. All participants in the system are considered identical and trustless. Therefore, in order to securely and reliably update the CL, the system must employ a sophisticated consensus scheme to manage the addition of new blocks. Over the last few years, several consensus schemes have been proposed in the literature(PoW, PoS, PoA, BFT/PBFT...) [8]. They can be classified, according to the strategy they employ, into two main categories, competitive and cooperative. Competitive consensus like *Proof of Work (PoW)* force participants to perform useless amounts of computation to compete for the right of creating and adding new blocks. *Proof of Stake (PoS)* [9], also a competitive consensus, uses the auction results of virtual stakes held by participants to elect a winner who will have the right to create updates. However, PoS gives candidates with a large number of virtual stakes an advantage in winning, which paradoxically weakens the key concept of decentralization in the blockchain. *Byzantine Fault Tolerance/Practical Byzantine Fault Tolerance (BFT/PBFT)* [8], are cooperative consensus that aims to reach agreement in an asynchronous environment with bounded message delays and less than one-third (n/3) of Byzantine nodes. However, BFT/PBFT and its variants of consensus schemes support a small set of players known in advance and become very slow when the number of nodes exceeds a certain threshold.

To solve the problems of the above mentioned consensus schemes, new consensus mechanisms based on trust and reputation are emerging [2], [4], [5]. The term trust here refers to the mutual trust between two nodes, also called local reputation. It is established by evaluating and recording interactions between nodes. In contrast, the term reputation represents the overall opinion the system's nodes have regarding a specific node [22]. It is usually calculated by aggregating local reputation scores. A reputation-based approach can then be proposed to replace or improve existing approaches. Specifically, the global reputation score could be used to filter active nodes and thus optimize the selection of validators. As a result, the consensus scheme can be accelerated, which

ISBN 978-3-903176-57-7

certainly improves the scalability of the system. In addition, employing a strategy that uses a reputation score instead of computational power or staking amount alone will provide much greater fairness among BC peers and thus solve the problem of monopolizing the update. However, existing trust and reputation blockchain systems themselves have some problems. To begin, only few works have considered how to combine the offline trust of real-world participants (external trust) and the online trust of nodes running the BC system (internal trust) to satisfy the needs of various trust management applications and improve the scalability of the BC system itself. Moreover, there is a common problem with almost all existing solutions, which is the trust calculation method. Almost all reputation-based systems use trust models that estimate the trust of entities based on their previous behaviors or current reputation. We believe that trust should be related to real property i.e. physical or digital assets. Another point that almost all trust-related applications ignore is the real-world correlation. A well-designed BC system should not ignore offline tasks. It should not focus only on the recording of transactions. But it should also consider whether offline interactions are properly performed, e.g. whether goods after an online purchase are well delivered or not.

In this work we present "GuRuChain", a guarantee and reputation-based service trading platform framework. Its main contributions are summarized as follows:

1) A trading platform framework that combines guarantee and reputation to secure exchanges between participants.
2) An efficient trust/reputation model that allows assessing and evaluating trust for each participant in the system.
3) An incentive mechanism based on guarantee and reputation to control the behavior of participants. In GuRuChain, good behavior will be rewarded by improving the participant's reputation score while bad behavior will result in a considerable loss in reputation and property (deposit).
4) A green lightweight consensus called *Proof of Guarantee and Reputation (PoGR)* that requires less computation and can ensure fairness among nodes using both guarantee and reputation scores.
5) An initial implementation and evaluation of the proposed solution.

The remainder of this document is structured as follows: Section II describes some related works. The system architecture and the consensus scheme are presented in section III. Section IV presents the proposed consensus algorithm. Section V is devoted to the performance evaluation of the proposal. We conclude our work in section VI.

II. Related works

In this section, we review the solutions involved in previous works that introduce the concept of trust into Blockchain to optimize performance and efficiency.

[4] introduces *TrustChain*, a trust-based permissioned blockchain that replaces PoW consensus with a Proof of Trust (PoT) consensus. The proposed solution is based on the design of a comprehensive trust architecture that uses several modules to calculate the trust level and predict the behavior of each participant before interacting using smart contracts. However, the designed architecture requires additional complex modules such as the machine learning module and the artificial intelligence module that use many resources to predict the users' behavior. [5] presents a new consensus mechanism called *Delegated Proof of Reputation (DPoR)*, which is an improvement of *Delegated Proof of Stake (DPoS)*. The main idea of this system is to ensure fairness among peers, the authors propose to use a set of parameters to choose the next block producer. However, the proposed scheme requires more analysis and experimentation to assess the relevance of certain parameters(reputation ranking). [1] proposes a three-layered Blockchain-based framework for trust management in BC-IoT supported supply chains. The proposed solution represents a service platform implemented on a permission-based blockchain network that uses smart contracts to automate reputation calculations, along with a reward and punishment-based incentive mechanism to encourage users to behave honestly. However, the proposed solution does not address the consensus scheme, some entities are considered honest. Moreover the business network administrator has strong control over the network, which represents a central point of failure and a security issue. [2] combines crowdsourcing with the blockchain consensus process. It introduces a reputation algorithm adapted for crowdsourcing, proposes an improved version the PoT consensus [7], and employs an incentive mechanism based on game theory to ensure the honesty of nodes. However, there is a lack of determinism in the proposed consensus. The probabilistic algorithm does not guarantee the uniqueness of the elected leader. In addition, the authors did not discuss what protocol their system might use to solve the forks that may occur. [6], [12], and [13] apply explicit incentive mechanisms based on the reward and punishment approach, where participants will be rewarded for good behaviors, while bad behaviors will lead them to be punished or even removed from the system. Other use cases are proposed in [3], [14], [15]. The lack of scalability makes them unable to be used as a general framework in other scenarios.

Given the above issues, a blockchain system suitable for trust-related applications must be designed to effectively address the problem of mutual trust among participating members. The designed system should include a trust-based consensus scheme that establishes the best trade-off between scalability, security and decentralization. In addition, the consensus algorithm must ensure a certain degree of fairness among the operating nodes to avoid update monopolization. Furthermore, the business model of the platform must use a trust model that links the value of trust to real assets in order to secure the exchange process and guarantee efficiency.

III. System Architecture and Model Description

In this section, we propose GuRuChain a BC-based online service trading platform framework, which benefits from the trust model that relies on the market supply-demand and the guarantee mechanism. Our proposal has two main parts.

The first one is the trading logic implemented using smart contracts and designed to allow sellers to advertise their services with a guarantee that they are willing to provide. Sellers are free to set the price and guarantee for their services. On their side, buyers can also freely select one or more services among those advertised. A seller who has a high reputation score and offers a high guarantee is more likely to sell his service or product. The trading process involves both online and offline interactions. The online transactions are conducted on the BC system. The offline interaction is performed in the real world. Once the online transaction is completed, the seller begins to provide his service in the real world. This service can be an immediate transaction or a long-term interaction such as an online subscription. Buyers and sellers evaluate the previously provided service through feedback transactions. If the service provision is intact, the seller releases the guarantee, otherwise, it goes back to the buyer. The second part of the system is the proposed consensus scheme called *Proof Guarantee and Reputation (PoGR)* that enables the management of all transactions in GuRuChain. The PoGR consensus selects block producers based on a scoring formula that uses three parameters: reputation score, staked amount, and risk taken, to ensure reliability and fairness. The complete architecture of the system will be presented in detail in the following.

Two parameters have been considered in GuRuChain, Guarantee and Reputation, their definitions are the following:

- Guarantee: represents the amount G i.e "x of tokens" the participant is willing to lock in the BC as a deposit to secure a specific task, such as a trading interaction or BC updating operation.
- Reputation: refers to the opinion that BC participants have towards a specific participant. It is expressed as a score $R \in [0 - 1]$ and calculated by aggregating the results of past interactions.

A. System Architecture

Figure 1 shows the proposed layered architecture of our system. As mentioned before, our online service trading platform framework is composed of individual participants: each participant owns at least one End Device (ED) to interact with the system. Through the networking module, nodes are connected via a secure P2P protocol to form a well-connected network that supports the Distributed Ledger (DL) layer running above it. All nodes jointly maintain a Common Ledger (CL) by running the consensus algorithm to update their Local Ledger(CL). Another part of the system is the organization that is represented on the BC by participants who have the role of admin. Their main task is managing users as they enter and exit the system. Specifically, they can add or remove users or nodes from the system. The organization in our system can be a single organization or a consortium of organizations. It is considered the initiator of the trading system. The most important role of the organization is to ensure that each user can only have one digital identity in order to avoid sybial attacks. Therefore, users must be recognized with a ID, such as a Social Security Number (SSN). This information will be stored outside the ledger (off-chain) so that only the organization can access it. To enable that, another ledger, such as the Identity Management Ledger (IDML) [11], can be used alongside the main ledger to separate identity management from trading, thereby preserving privacy and protecting data. However, this part is beyond the scope of this work. The organization is also responsible for converting real-world currency into virtual funds. However, it does not get involved in the trading.

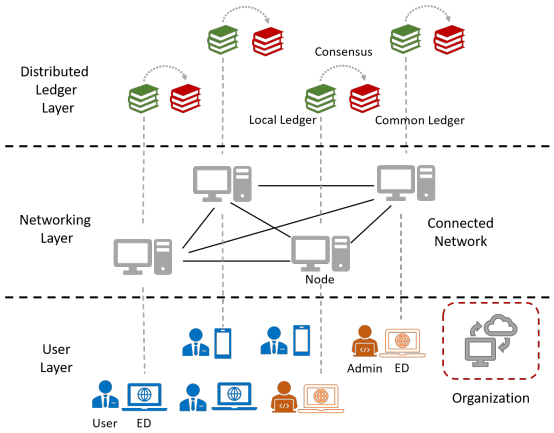


Fig. 1: System Architecture.

B. Node Structure

In GuRuChain, each user called participant is able to launch a node and join the BC network if and only if he or she meets the following two conditions: having a sufficient level of reputation, and depositing at least the minimum guarantee amount (more details are given in section IV). Figure 2 shows the complete structure of a node that can be launched by a participant. It contains the following functional modules:

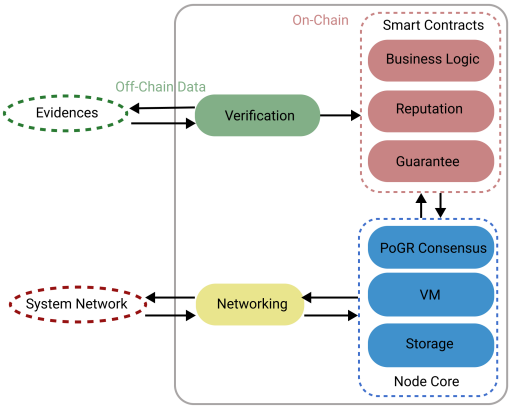


Fig. 2: Node structure.

- **Verification Module:** is used to check and evaluate whether the real-world service is properly provided. This module can verify immediate transactions and even track the quality of

services over time. It provides participants with a certificate to prove that the requested services have been correctly delivered. In order to deliver the service provision certificate, the verification module needs to collect trading information from the buyer, the seller, and an independent controller as proofs (a collection of real-world data and events), then verify them by corresponding measurements and finally produce a certificate that can be recognized by all BC nodes . Depending on the type of service provided, the controller in our system can be a standalone smart device with a dedicated interface to capture the required information or an independent physical entity equipped with a sophisticated end device to enter this information. Due to the variety of services provided in the system, the proofs collected by the verification module and the measures used may be different. The verification process and its output results should be considered as hands-off. We need to assume that when proofs are given as input, everyone's verification module will always generate the same certificate, which the participants cannot fake. The verification process can be handled using Decentralized Oracle Networks [10](DONs [1]) that enable the creation of hybrid smart contracts, where on-chain code and off-chain infrastructure are combined to support advanced decentralized applications (DApps) that react to real-world events and interoperate with traditional systems.

- **Smart Contract Module:** A smart contract is a deterministic and immutable program that includes an executable script and a data model stored in a BC in the form of a Merkle hash tree [8]. In addition, a smart contract can implement read-modify-write operations changing the data in the ledger and store the result of the processing in the blockchain itself. Smart contracts are triggered by addressing transactions to them. The smart contract module in our system is the module that implements the entire trading process. It contains three main sub-modules, Business logic, Reputation, and Guarantee submodule.
  1) **Business Logic Submodule:** or the Application submodule. It contains all the functions needed for a participant to interact with the system. This submodule is mainly responsible for the following tasks:
     - Manage participant login access and signing published information using a public and private key pair.
     - Managing the participant's independent wallet account.
     - Generate and process various business transactions such as service announcement and order placement sent to the system through its ED or the networking module.
  2) **Reputation Submodule:** it is responsible for trust and reputation management by implementing two main functions that allow first the calculation of the trust value of the last interaction and then the update of the global reputation score of the participants involved in this interaction.

Trust Value Calculation: We evaluate the trust of all participants in our system per interaction. More precisely, the system's reputation module assigns a trust value to each trader (seller or buyer) after each trading interaction. The trust value assigned to a seller T_s^i after a real-world interaction i is given by :

$$T_s^i = \theta.T_{b\to s}^i + (1-\theta).T_{v\to s}^i$$

Where $T_{b\to s}^i$ represent the subjective trust value, $T_{v\to s}^i$ is the verification (objective) trust value and θ is the weight that gives more relevance to the objective trust value. The trust value that represents the direct opinion of the buyer on the seller $T_{b\to s}$ is computed using the subjective trust logic [20], [21]. When a buyer b interacts with a seller s, an opinion denoted by $O_b^s = (t, d, u)$ is given to express b's belief in the trustworthiness of s. Where, t, d, and u represent trust, distrust, and uncertainty, respectively; $t + d + u = 1$ and t, d, $u \in [0, 1]$.

$$\begin{cases} t = (1-u)\frac{m}{m+n} \\ d = (1-u)\frac{n}{m+n} \\ u = 1 - I_f \end{cases} \quad (1)$$

The value of $T_{b\to s}$ after the i^{th} interaction,

$$T_{b\to s}^i = t + \psi u \quad (2)$$

Where, m and n are the number of valid and invalid interations between b and s respectively, I_f denotes the interaction frequency. The higher interaction frequency results in a lower uncertainty. ψ is the uncertainty weight [21].

The objective trust value $T_{v\leftarrow s}$ depends on the results of the verification process and the measures used in that process. As mentioned before, according to the types of services or products provided on the platform, different metrics are required to verify and evaluate the provision of these services, such as compliance, quality, delivery time, etc. Our solution is a general platform framework that we can adapt to many types of applications. Therefore, the objective trust formula may differ from one service to another. However, the calculation method is almost the same. It depends mainly on the presence of the certificate, the amount of the transaction and the time between the last two interactions. It is clear that it is not expensive for a buyer to collude with other sellers to improve their reputation. He or she can buy many cheap products from the same seller (or different sellers) to improve their reputation together in a short time. Thus, to mitigate this type of collusion attack, objective trust should be tied to the value and timing of T_x. For example, for an ongoing interaction, if one of the traders has already exchanged a product a short time ago, $T_{v\leftarrow s}$ should be set to a relatively low value to avoid collusion attacks. The calculation formula is :

$$T_{v\to s}^i = C\ [\sigma_c + \sigma_t F_t + \sigma_a F_a] \quad (3)$$

[1]https://chain.link/education/blockchain-oracles

$$\sigma_c, \sigma_t, \sigma_a \in [0,1] \ ; \ \sigma_c + \sigma_t + \sigma_a = 1 \tag{4}$$

Where, C is a boolean that refers to the presence of the certificate '1' or not '0', σ_c is the weight of the certificate itself. σ_t and σ_a are the weights of the time t and the amount a of the interaction, respectively. F_t and F_a are the functions that normalize t and a, respectively (F_a, $F_t \in [0,1]$). They both have a positive correlation with t and a.

The combination of objective and subjective trust results in the overall trust value of the ith interaction. The same model will be used to calculate the trust value of the buyer T_b^i.

Overall Reputation Update: In our system, an initial reputation value R_{init} is assigned to each new participant. This value is assumed to be the critical line of trust, so that all participants whose reputation is lower than R_{init} are considered untrusted. We update the overall reputation score of a trader after each real-world interaction. We believe that the only way to show good intentions is through proper real-world interaction. This is because online interactions are governed by the platform itself, so participants do not need to trust each other. Unlike real-world interactions where buyers must trust sellers to provide services that have already been paid for. To summarize, the only way for a trader to improve his reputation is by conducting correct interactions in the real world. We now explain the process, the reputation model is triggered after each offline interaction to update the overall reputation score of both parties. First, we compute the trust value T_i as described in 2, then we use it to update the overall reputation score of the participants involved in this interaction as follows:

$$R_i = \begin{cases} (1-\omega)R_{i-1} + \omega.T_i & T_i \geq R_{init} \\ \omega R_{i-1} + (1-\omega)T_i & T_i < R_{init} \end{cases} \tag{5}$$

$$\omega = F(T_i, N_b) = \kappa T_i \frac{1 - e^{-\lambda.N_b}}{1 + e^{-\lambda.N_b}} \tag{6}$$

Where R_{i-1} represents the current reputation value, i.e. before the last interaction. ω is a weighting function, N_b refers to the number of blocks added to the ledger since the trader joined the network. We use a variable weighting factor ω (a Hyperbolic tangent function that normalize N_b) instead of a static factor to ensure that older participants have more opportunity to improve their reputation without giving them any liberty to make mistakes. Thus, the longer the users stay in the system, the more likely they are to improve their reputation, provided they maintain correct behavior. The maximum value of the weighting, which can be reached when N_b becomes significantly high, is $\kappa \in [0,1]$. Moreover, the weighting function does not depend only on N_b but mainly on the trust score T_i computed during the last interaction. Therefore, a bad action represented by a low T_i ($T_i < R_{init}$) will have a huge impact on the overall reputation score whatever the value of N_b.

3) **Guarantee Submodule:** In our system, buyers can reject the delivered service after verification. To protect sellers from such behavior, we allow them to set the proportion they want to get as service delivery fee in case of rejected service. They are free to set this value, which we call the guarantee proportion P_g. Let V be the value of the service, the amount transferred to the seller in case of a rejected delivery is $G = P_g V$. Buyers, on the other hand, are free to choose the services offered by different sellers. To encourage buyers to choose its service, a seller must choose an appropriate P_g. Services with low P_g will be preferred, while those with high P_g will be avoided.

- **Node Core Module:** It includes all the components of an Ethereum[2] BC client, from the networking and storage modules to the execution engine. It also includes the consensus algorithm that enables transaction verification and block validation.

The communication among these modules ensures the functionality of the node. The verification module has a specific external interface to collect the proofs related to service provision. The result of the verification process, the certificate, will be sent to the smart contracts module for further processing. Except for the verification module, which has a specific interface, the network module is the only interface between the system and the node itself.

IV. PoGR Consensus Scheme

In this section, we discuss the consensus scheme in GuRuChain. Specifically, the mechanism used to select the block to be added to the BC. A reliable consensus mechanism must ensure validity and consistency to operate properly. Meanwhile, efficiency and cost-effectiveness must be maintained to achieve high performance.

We present **Proof of Guarantee** & **Reputation (PoGR)** a lightweight consensus mechanism that replaces the need for unnecessary mining of PoW-like consensus with a scoring method to enable **Block Producer (BP)** selection. PoGR reduces the use of processing power by each running node and therefore decreases its energy consumption. The scoring method ensures fairness among nodes and thus, solves the update monopolization problem that often occurs in PoS-like consensus schemes.

A participant who wants to be involved in the BC management must run a node that will execute the consensus algorithm and thus be able to produce blocks. To do this, he or she must generate two transactions, TX_{G_B} to pay a deposit $G_B \in [G_{min} G_{max}]$ as a guarantee balance and TX_{G_p} to specify the effort or risk he or she is willing to take $G_p \in [G_{min} G_B]$ to get the updating right. Both transactions TX_{G_B} and TX_{G_p} are sent to the BC system for verification and storage. The participant will not be able to launch its node to run the consensus algorithm and create updates unless these two conditions are satisfied: both TX_{G_B} and TX_{G_p} transactions are valid, and its reputation is higher than R_{min}.

[2]https://besu.hyperledger.org/en/stable/private-networks/

In POGR, each node maintains a list of operating nodes and their scores locally. This list is updated after adding a new block to the BC. In GuRuChain, participants who run nodes are rewarded for doing so. The G_{min} and G_{max} values are the minimum and maximum amounts that the system should pay to the participants who run nodes as a reward. Their initial values are set by the system administrators and can be adjusted based on available information. The idea behind this is that when a malicious block producer generates an invalid block, it will eventually lose a portion of its deposit G_p ($G_p \geq G_{min}$). The lost amount G_p is then transferred to the other participants managing the nodes as compensation.

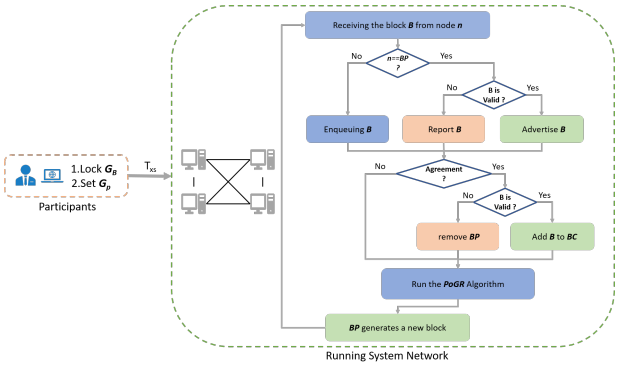


Fig. 3: PoGR consensus scheme.

Note that nodes whose remaining guarantee G_B is below G_{min} or whose reputation is lower than R_{min} are automatically removed. We examine all requests using a scoring method. Each node sort the list of running nodes using the following formula:

$$S_n = (\alpha.G + \beta.E + \gamma.R).\tau_n \ ; \ \alpha + \beta + \gamma = 1 \qquad (7)$$

Where, S_n is the score of the node n, $G = G_B/G_{max}$ is the proportion to the maximum amount the participant wants to lock as a guarantee balance, $E = G_p/G_B$ is the proportion to the balance that represents the Effort/Risk taken by the participant, R is the reputation score of the participant who manages the node. Finally, $\tau_n = F(N_{gb})$ is the available generation rate initially equals to 1. It decreases exponentially with the number of blocks N_{gb} generated during a block cycle B_c, which refers to the period in blocks needed for a block producer to recover 100% of its producing right, i.e., $\tau_n = 1$. For example, let's take $B_c = 5$ and a node n that generates a new block ($B = 10$ is its first block in the last five blocks). n needs to wait for at least five blocks to recover 100% of its available generation rate τ_n (until $B = 15$). In other words, τ_n will be reset to 1 only if n does not generate any of the following five blocks. Moreover, τ_n will continue to decrease exponentially according to the number of blocks N_{gb} that n generates during this period. In this example, $\tau_n = 1/exp(2,1) = 1/2$, with a $base = 2$ and $N_{gb} = 1$, $B = 10$ is its first block. The complete sheme is described in detail in Fig.3 and Algo 1

Data: B_c, $listNodes$, G_{min} , G_{max} , CL
Result: selection of the next BP
begin
  BP $\leftarrow$ $receivedBlock$.`getBP()`
  if *!* $receivedBlock.isValid$ **then**
    Remove <BP , S_{BP} > from $listBP$
  else
    S_{BP} = `computeScore(`*BP*`)`
    Call `updateBPList(`*BP*,S_{BP}`)`
    foreach *node n in* $listNodes$ **do**
      if *exists a* T_x *of n in* $receivedBlock$ **then**
        S_n = `computeScore(`*n*`)`
        Call `updateBPList(`*n*,S_n`)`
      end
    end
  end
  Set next BP the node with the highest score
end
Function `computeScore(`*n*`)`**:**
  Read (R, G_p, and G_B) of n from CL
  if $R < R_{min}$ *or* $G_p \notin [G_{min}G_B]$ **then**
  Remove <n , S_n > from $listBP$
  else
    $N_{gb} \leftarrow 0$, $i \leftarrow 0$
    $currentHeader \leftarrow CL.blockHeader$
    while $i < B_c$ *and !* $currentHeader.isNil$ **do**
      if $currentHeader$.`getBP()` == *n.adr*
      then
        $N_{gb} = N_{gb} + 1$
      end
      $currentHeader \leftarrow$
      CL.`getHeader(`$currentHeader$.PARENTHASH`)`
      $i \leftarrow i + 1$
    end
    $\tau_n = 1/\exp(base, N_{gb})$
    $S_n = [\alpha.G_B/G_{max} + \beta.G_p/G_B + \gamma.R].\tau_n$
  end
  return S_n
Function `updateBPList(`*n*,S_n`)`**:**
  if $n \notin listBP$ **then**
  Insert <n , S_n > in $listBP$
  else
    Update <n , S_n > in $listBP$
  end

Algorithm 1: Proof of Guarantee & Reputation.

We now examine how PoGR can guarantee fairness among nodes analytically. The scoring formula (eq. 7) used to evaluate node candidates must satisfy the following two conditions:

- **Reputation should be given more relevance than guarantee** for two reasons; First, it explicitly reflects the level of trust the system has in the participant; Second, the participant locks his guarantee balance and sets the proportion that he is willing to use freely, unlike the reputation score, which is governed by the system itself. In other words,

the participant has no intervention in the calculation of his reputation.

$$\gamma \geq \alpha, \beta \tag{8}$$

- **The effort E importance over guarantee balance** G. eq.7 must meet this condition because, in PoGR, more effort ($G_p \approxeq G_B$) means more risk of leaving the consensus. Let P_1 and P_2 be two participants who have the same reputation R and both of them did not generate any block yet. However, P_1 has a low guarantee balance $G_{B_1} \approxeq G_{min}$, so he or she has to fixe $E_1 \approxeq 1$ i.e. a high risk. On the other side, P_2 is a participant who has a very high balance $G_{B_2} \approxeq G_{max}$, but decides to not take a high risk $E_2 = G_{p_2}/G_{B_2} \leq \frac{G_{B_2}-G_{min}}{G_{B_2}} = \frac{G_{max}-G_{min}}{G_{max}}$. P_1 should get at least the same score as P_2, because if P_1 produces a malicious block, his or her node will be automatically deleted (the remaining G_{B_1} is less than G_{min}).

$$\begin{aligned} S_{P_1} \geq S_{P_2} &\Rightarrow \alpha.G_1 + \beta.E_1 \geq \alpha.G_2 + \beta.E_2 \\ &\Rightarrow \alpha.\frac{G_{min}}{G_{max}} + \beta.1 \geq \alpha.1 + \beta.\frac{G_{max}-G_{min}}{G_{max}} \\ &\Rightarrow \alpha.\frac{G_{min}}{G_{max}} \geq \alpha - \beta.\frac{G_{min}}{G_{max}} \\ &\Rightarrow \frac{G_{min}}{G_{max}} \geq \frac{\alpha}{\alpha+\beta} \end{aligned} \tag{9}$$

The designed scheme aims to improve the reliability of the updating process without forcing participants to pay more, which is ensured by the condition 9 implicitly. It encourages participants to take more risks than investing more funds. Since the consequence of a malicious update with higher risk is node expulsion, all malicious nodes are likely to be removed, which improves the system's efficiency and reliability.

V. Evaluation and Results

In this section, we first present the business model formulation of the proposed platform in the context of Hyperledger Besu [3] followed by the experimental setup. Next, we discuss the evaluation results of the platform in terms of the effectiveness of the trust and reputation model, the overall system scalability, and performance.

A. Business Model

We implement the proposed palatform framework on Hyperledger Besu which is an open-source Ethereum client developed under the Apache 2.0 license and written in Java. Besu includes a command line interface and a JSON-RPC API to run, maintain, debug and monitor nodes in an Ethereum network. The API can be used via RPC over HTTP or WebSockets. The API supports typical Ethereum features such as: Smart contract and Decentralized application development (Dapp). For the evaluation, we define a trading logic that includes:

- Participants: Traders (Sellers and/or Buyers) and Admins.
- Assets: we define a data structure that could represent any real-world service or product.
- Smart Contracts: There are three types of smart contracts used to develop the business model: the Permission Smart Contract (on-chain permissioning [4]) ; it provides the functionality of adding and removing participants and nodes by administrators. The second SC is the Trading Smart Contract which handles all the trading operations, from service announcement to deposit revocation. The last one is the reputation and guarantees smart contract; it maintains the trust calculation and the reputation update. It also manages the deposit submission and revocation.

B. Experimental Setup

The deployment of the overall system platform and the evaluation tests are carried out on a cluster of eight PCs whose characteristics are given in Table I. We use Hyperledger Caliper [5] to run the evaluation experiments. Caliper is a blockchain performance evaluation framework that allows testing and measuring the performance of different blockchain solutions using metrics such as latency and throughput. In addition, based on the logs of running nodes, we analyze other measures such as the fairness of the consensus scheme and the effectiveness of the trust model. The platform's business model configuration involves the following setups: Permissioning Smart Contract for rules management, Hyperledger Besu for trade management, and Web3js [6] for client application interaction with the smart contract.

TABLE I: Computing environment.

Machine	**#**
Intel® Core^{TM} i7-6820HQ CPU @ 2.70GHz × 8 ; 16GiB	3
Intel® Core^{TM} i7-8700 CPU @ 3.2GHz × 12 ; 32GiB	1
Intel® Core^{TM} i7-4710MQ CPU @ 2.50GHz × 8 ; 16GiB	1
Intel® Core^{TM} i7-4800MQ CPU @ 2.70GHz × 8 ; 32GiB	1
Intel® Core^{TM} i7-6500U CPU @ 2.50GHz × 4 ; 16GiB	1
Intel® Core^{TM} i7-6700HQ CPU @ 2.60GHz × 8 ; 16GiB	1

C. Trust & Reputation Model Effectivness

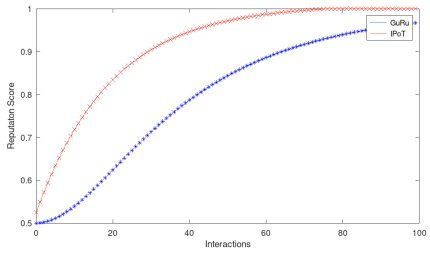
We begin the discussion with results that demonstrate the effectiveness of our trust and reputation model. Figure 4a shows the ideal reputation growth of a participant in our system (**GuRu**) compared to the referenced one (**IpoT**) [2] which uses the same logic to compute trust (interactions) and provides better results than the traditional reputation models. We observe that the reputation score R increases as the number of interactions increases in both models. However, we note that the growth of R in our model is slower than in the other. Therefore, the participant in our system needs to perform more interactions to reach the maximum score. The reason behind this is that, in our reputation model, we select the update formula based on the trust value of the last interaction. We give less relevance to T_i if its value is above the critical trust

[3] https://besu.hyperledger.org

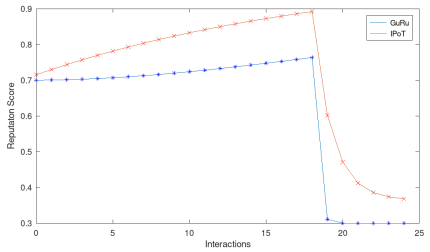
[4] https://github.com/ConsenSys/permissioning-smart-contracts

[5] https://github.com/hyperledger/caliper-benchmarks

[6] https://web3js.readthedocs.io



(a) Ideal reputation growth of a normal participant.



(b) Reputation growth of a malicious participant.

Fig. 4: Effectivness of the Trust and Reputation models.

line R_{init} because it reflects the expected behavior. On the other hand, if the trust value is below the threshold, it will be given significant relevance relative to the current reputation value R_{i_1} because it represents inappropriate behavior. Figure 4b shows the response of both systems to improper behaviors. To trick the system, a malicious participant will continue to behave correctly for a while and then try to do something wrong. As the figure shows, after producing a bad action, the reputation score decreases rapidly in both systems. However, what we notice is that the score drops faster in (**GuRu**). This is because our system has a high sensitivity to improper behaviors. We can conclude that a participant in our system needs more time and interactions to achieve a high reputation. Moreover, as soon as he misbehaves, he will get a reputational hit, which drops his score below the critical line. As a result, the system will consider him as an untrusted participant, and it will be difficult for him to get back his reputation.

D. Performance & Scalability Evaluation

Now, we present results quantifying the performance of our system for relevant benchmarks using Hyperledger Caliper. We consider three metrics for GuRuChain performance evaluation as described below:

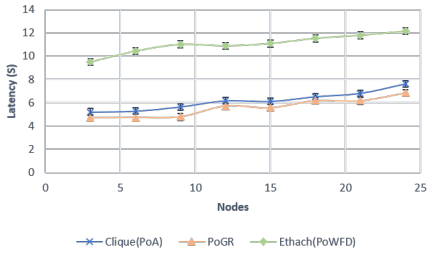
- Throughput: the number of successful transactions per second (TPS).
- Latency: refers to the time difference in seconds between the submission and completion of a transaction.
- Scalability: Changes in throughput and latency when altering a configuration parameter, such as network size or node configuration.

Increasing the size of the network may be a feasible approach to improve the performance of some P2P systems. However, in the context of blockchains, additional factors such as block propagation time and consensus costs may have a direct impact on system scalability. In this section, we present scalability evaluation results obtained from several comparison tests between our proposed protocol (PoGR), the standard PoA protocol (Clique), and the Ethach (Ethereum's PoW). In our experiments, we collected several data points, each corresponding to the average of several runs with a specific network configuration. Each run consisted of 3000-5000 transactions. To fairly compare the protocols, we applied the same settings for both PoGR and PoA consensus, i.e., the same: block period (5-10s), node configuration, block size (gas limit), and workload configuration. In other words, we only varied the network size at a time while other parameters were set to the same values. For the PoW settings, we used a low fixed difficulty to adjust the block frequency and get an average block time between 5 and 10s. The other parameters are the same as for PoGR and PoA.

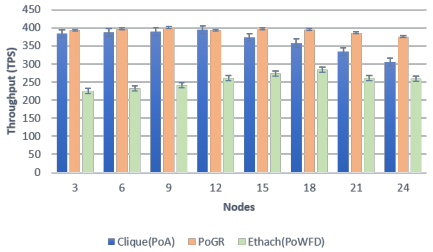
From Fig.5, we can see the gap between the consensus protocols PoGR, PoA (Clique), and PoW (fixed difficulty). PoGR and PoA outperform PoW in terms of both latency and throughput. Furthermore, when adding nodes to the network, the results show that the PoW's throughput increases at the beginning but starts to decrease once the peak is reached. Contrary to the PoGR's throughput, which remains stable as the block producer selection in our protocol does not depend on the number of participating nodes. The slight overhead is related to the network's communication and consensus costs. Furthermore, compared to the standard PoA protocol, no significant performance difference has been observed for small network ($N < 12$). This is due to the experimental setup where nodes in both protocols seal blocks in the same fixed period. However, when the network size exceeds a certain threshold ($N > 12$), the throughput of Clique starts to decrease while we continue to get a higher throughput in our protocol. This is justified by the fact that in Clique, the probability of getting forks increases with the number of validators. Therefore, the system will take extra time to resolve those forks, which will obviously generate an additional time cost.

VI. Conclusion

In this paper, we proposed a platform framework for online/offline service trading. The proposed solution combines on-chain and off-chain trust management to provide a blockchain system suitable for trust-related trading systems. GuRuChain supports both the recording of online transactions and the provision of offline services. Our solution relies on market supply and demand to ensure free exchange between participants. Our goal was to enable effective trust management for all participants. To achieve this, we complemented the GuRuChain trust model with a guarantee and reputation-based reward and punishment mechanism. The introduced trust and reputation system helps not only buyers to select the services provided by sellers, but allows also reliable participants to manage nodes and create updates. The core part of GuRuChain is the proposed Lightwgheit consensus scheme, called PoGR, which selects block producers based on guarantee and reputa-



(a) Latency comparison



(b) Throughput comparison

Fig. 5: Latency and Throughput comparison between Ethach(PoW) and clique(PoA) and PoGR.

tion scores. We implemented and deployed a prototype of the proposed framework using Hyperledger Besu. Accordingly, several test scenarios were performed to evaluate the effectiveness of the proposed trust model and the scalability of the overall system. The results of this evaluation demonstrate the feasibility, efficiency, and scalability of GuRuChain.

ACKNOWLEDGMENT

We acknowledge the support of the Nouvelle Aquitaine region for this research. We thank the FUI23 PARFAIT and B4IoT projects for funding and supporting this work.

REFERENCES

[1] Sidra Malik, Volkan Dedeoglu, Salil S. Kanhere and Raja Jurdak, "TrustChain: Trust Management in Blockchain and IoT Supported Supply Chains," 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, pp. 184-193, doi: 10.1109/Blockchain.2019.00032

[2] Xiaoyu Zhu, Yi Li, Li Fang, and Ping Chen. An improved proof-of-trust consensus algorithm for credible crowdsourcing blockchain services. IEEE Access, 8:102177-102187, 2020.

[3] Volkan Dedeoglu, Raja Jurdak, Guntur D. Putra, Ali Dorri, and Salil S.Kanhere. 2019. A trust architecture for blockchain in IoT. In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '19). Association for Computing Machinery, New York, NY, USA, 190–199. DOI:https://doi.org/10.1145/3360774.3360822.

[4] Jayasinghe Upul, Lee Gyu Myoung, MacDermott Áine, Rhee Woo Seop. (2019). TrustChain: A Privacy Preserving Blockchain with Edge Computing. Wireless Communications and Mobile Computing. 2019. 1-17. 10.1155/2019/2014697.

[5] Thuat Do, Thao Nguyen, and Hung Pham. Delegated Proof of Reputation: a Novel Blockchain Consensus. In Proceedings of the 2019 International Electronics Communication Conference (IECC '19). Association for Computing Machinery, New York, NY, USA, 90–98. DOI:https://doi.org/10.1145/3343147.3343160.

[6] Eric Ke Wang, Zuodong Liang, Chien-Ming Chen, Saru Kumari, and Muhammad Khurram Khan. Porx: A reputation incentive scheme for blockchain consensus of iiot. Future Generation Computer Systems, 102:140-151, 2020.

[7] Jun Zou, Bin Ye, Lie Qu, Yan Wang, Mehmet A. Orgun, Lei Li (2019). A Proof-of-Trust consensus protocol for enhancing accountability in crowdsourcing services. IEEE Transactions on Services Computing, 12(3), 429-445. https://doi.org/10.1109/TSC.2018.2823705

[8] Emanuele Bellini, Youssef Iraqi and Ernseto Damiani, "Blockchain-Based Distributed Trust and Reputation Management Systems: A Survey" in IEEE Access, vol. 8, pp. 21127-21151, 2020, doi: 10.1109/ACCESS.2020.2969820.

[9] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, Dong In Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks" 2018, arXiv:1805.02707. [Online]. Available: http://arxiv.org/abs/1805.02707

[10] De Pedro, Adán Sánchez, Levi Daniele, et Cuende Luis Iván. Witnet: A decentralized oracle network protocol. arXiv preprint arXiv:1711.09756, 2017.

[11] Malik Sidra, Gupta Naman, Dedeoglu Volkan, Kanhere Salil, Jurdak Raja. (2021). TradeChain: Decoupling Traceability and Identity in Blockchain enabled Supply Chains. 1141-1152. 10.1109/TrustCom53373.2021.00155.

[12] Changbing Tang, Luya Wu, Guanghui Wen, Zhonglong Zheng, "Incentivizing Honest Mining in Blockchain Networks: A Reputation Approach," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 1, pp. 117-121, Jan. 2020, doi: 10.1109/TCSII.2019.2901746.

[13] Nojoumian Mehrdad, Golchubian Arash, L. Njilla , K. Kwiat, C. Kamhoua. (2019) Incentivizing Blockchain Miners to Avoid Dishonest Mining Strategies by a Reputation-Based Paradigm. In: Arai K., Kapoor S., Bhatia R. (eds) Intelligent Computing. SAI 2018. Advances in Intelligent Systems and Computing, vol 857. Springer, Cham. https://doi.org/10.1007/978-3-030-01177-2-81.

[14] Singh Nikita, Tarun Kumar and Manu Vardhan. "Blockchain-based e-cheque clearing framework with trust based consensus mechanism." Cluster Computing (2020): 1 - 15.

[15] Cai Wenjun, Jiang Wei, Xie Ke, Zhu Yan, Liu Yingli, Shen Tao. (2020). Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain. International Journal of Distributed Sensor Networks. 16. 155014772090733. 10.1177/1550147720907335.

[16] Ahmet Bugday, Adnan Ozsoy, Serdar Murat Oztaner, and Hayri Sever. Creating consensus group using online learning based reputation in blockchain networks. Pervasive and Mobile Computing, 59:101056, 2019

[17] Jingyu Feng, Xinyu Zhao, Guangyue Lu, and Feng Zhao. 2019. PoTN: A Novel Blockchain Consensus Protocol with Proof-of-Trust Negotiation in Distributed IoT Networks. In Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things (IoT S&P'19). Association for Computing Machinery, New York, NY, USA, 32–37. DOI:https://doi.org/10.1145/3338507.3358613

[18] Leonard Kleinrock, Rafail Ostrovsky, and Vassilis Zikas. A por/pos-hybrid blockchain: Proof of reputation with nakamoto fallback. IACR Cryptol. ePrint Arch., 2020:381, 2020.

[19] You Sun, Rui Zhang, Rui Xue, Qianqian Su, and Pengchao Li. A reputation based hybrid consensus for e-commerce blockchain. In International Conference on Web Services, pages 1-16. Springer, 2020.

[20] Josang Audun, Hayward Ross, Pope Simon (2006)"Trust Network Analysis with Subjective Logic" . In Dobbie, G. Estivill-Castro, V (Eds.) Conference Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSW 2006). Australian Computer Society, CD Rom, pp. 85-94.

[21] Jiawen Kang, Zehui Xiong, Dusit Niyato, Dongdong Ye, Dong In Kim, Jun Zhao, "Towards secure blockchain-enabled Internet of vehicles: Optimizing consensus management using reputation and contract theory," 2018, arXiv:1809.08387. [Online]. Available: http://arxiv.org/abs/1809.08387

[22] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, Eric Friedman Communications of the ACM, December 2000, Vol. 43 No. 12, Pages 45-48 10.1145/355112.355122

[23] Zheng, P., Zheng, Z., Luo, X., Chen, X., Liu, X.: A detailed and real-time performance monitoring framework for blockchain systems. In: 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2018, pp. 134–143. ACM (2018)