A Technical Comparison Of Smart Contract Platforms

Cryptocurrencies and decentralised ledgers (COMP90088)

Research Project

Semester 1 2022

Aditya Desu (1000447)

Kelvin Lim Wan (929715)

Chan Jie Ho (961948)

1. Introduction

1.1. Blockchain

For over a decade, blockchain technology has emerged as a distributed computing paradigm that successfully overcomes the problem of trusting a centralised party. Blockchain is a type of a distributed ledger technology that allows online payments to be transferred from one party to another in a peer-to-peer network through the use of cryptographic signatures (Suvitha and Subha, 2021). These payments are recorded in the form of transactions, and to be added onto the blockchain, it must first be approved by the majority of the parties (nodes) within the network where it would then be combined into blocks. These transactions are publicly available, but each transaction is encrypted by the owner's secret key (Suvitha and Subha, 2021). Once the proof-of-work has been completed, it is connected to the end of the longest blockchain via hashes and it can't be altered or removed. This forms the basis of blockchain technology that allows it to securely store information without using a central authority (Makarov, 2021). With blockchain, we are able to achieve decentralisation, tamper-resistance, anonymity, and auditability (Wang et al., 2018).

1.2. Smart Contracts

Despite its advantages, the base blockchain technology – represented by Bitcoin and cryptocurrencies and coined as blockchain 1.0 (Wang et al., 2018) – on its own does not allow for writing contracts with complex logic due to the shortcomings of the scripting language.

Hence, smart contracts, first proposed in 1994 by Nick Szabo, were brought into the spotlight. Smart contracts are computer protocols that run on top of the blockchain and are designed to autonomously aid, verify and enforce the negotiation and agreement among distinct parties in a distributed environment (Vigliotti, 2021) independent of third party central authorities as these transactions are trackable and impossible to be tampered with (Suvitha and Subha, 2021).

Written in languages such as Solidity, Java, Python, etc., the way smart contracts work is that after the contract signed by the involved parties, it is attached to and recorded in the blockchain after the verification process of the blockchain. The smart contract dictates pre-defined states and transition rules, and when its encoded trigger conditions are met, are executed by the blockchain (Wang et al., 2018).

Smart contracts are applicable to a wide-range of purposes such as: financial transactions; prediction markets; Internet of Things (IOT); and many more (Wang et al., 2018).

1.3. Current Challenges and Related Work

Despite its significant progress in recent years, smart contracts still face many challenges such as the re-entrancy attack on Ethereum's DAO (Decentralised Autonomous Organisation) in 2016 (Morrison, Mazey and Wingreen, 2020). Aspects concerning the definition of standards and quality evaluation have not been thoroughly researched and the existing approaches mainly focus on performance and security features (Vigliotti, 2021). As a result, some platforms tailor their protocols for different applications such as providing financial services and binding legal contracts.

Another challenge surfaces when multiple dependent transactions in the same block call the same contract. Hostile miners can then take advantage of this by tampering with the ordering of the transactions. This can alter the final state of the contract. One solution proposed by Natoli et al. uses Ethereum-based functions to enforce the order of these transactions. (Wang et al., 2018)

As with any technology that stores user information, security bugs and privacy issues can lead to issues with severe consequences. Oyente (Luu et al., 2016) and Hawk (Kosba et al., 2016) seeks to help fix these issues. Oyente is a symbolic execution tool that can be used to search for the security bugs, while Hawk is a blockchain model that tries to protect user privacy by not storing the financial transactions clearly on the blockchain. (Wang et al., 2018)

1.4. Methodology

Focusing on Ethereum, Hyperledger Fabric, Corda, and NEM, this paper presents a comprehensive analysis of Smart Contract platforms from a technical point of view. In particular, we aim to evaluate each platform in terms of several key features, namely *programming language*, *execution environment*, *consensus mechanism*, *scalability*, *data storage* and *permission type*.

2. Feature Comparison And Analysis

2.1. Programming Language and Execution Environment

Ethereum is a public blockchain platform that uses its own native cryptocurrency called Ether (ETH) (Wang et al., 2018). One of the oldest smart contract platforms, it is also the most popular blockchain platform as of today that supports many high-level programming languages, namely Solidity, Serpent, and LLL (Ahmed, 2021). These smart contracts are then compiled into a bytecode language that's executed on a Turing-complete virtual machine called Ethereum Virtual Machine (EVM) (Parizi, Amritraj and Dehghantanha, 2022). Ethereum is also one of the most robust platforms such that EVMs have been implemented in many languages both declarative, such as Haskell, eand imperative, such as JavaScript, Java, Python, C++, etc. (Ahmed, 2021). Thus, allowing developers to more efficiently and easily develop smart contracts.

Created by the Linux Foundation, *Hyperledger Fabric* is an open-source blockchain platform that develops smart contracts using Java and Go, with three key functions in mind: Init, to implement contract deployment; Invoke, for transaction processing; and Query, for querying transactions (Wang et al., 2018). These contracts, known as "Chaincode" in the Hyperledger Fabric world, are executed in a Docker container (Harz and Knottenbelt, 2022). As with Ethereum, the developers of Hyperledger Fabric also developed JavaScript-based tools to promote efficiency and ease in development of smart contracts (Suvitha and Subha, 2018).

Written in Java, the *NEM*, which stands for New Economy Movement, is a blockchain platform that uses its own native cryptocurrency called XEM and revolves around API calls (Julian, 2018). On top of fewer security vulnerabilities, using Java – one of the most used programming languages in the world – provides ease of use and familiarity to developers as they would not need to learn new platform-specific languages such as Solidity (Davies, 2022). Simulating the current software development world, NEM seeks to allow developers to focus on the business portion of their own application and when needed, they can easily interact with the features on the platform through its API interface, thus allowing for development of code in languages independent of the smart contract language (Tam, 2018).

R3 *Corda* is an open-source blockchain platform written in Kotlin. Smart contracts on this platform are executed and validated on the Java Virtual Machine (JVM) and thus allows developers to use any related programming paradigm, which makes it easier for developers to use code from internal applications (Suvitha and Subha, 2021). According to the Corda whitepaper (Brown, 2018), the reason for using JVM for contract execution and validation is to make full use of the wide-spread familiarity and abundance of existing libraries available to the JVM. The main difference is that R3 Corda uses a custom, more restrictive, sandbox as compared to the usual JVM sandbox to allow for deterministic execution as well as tighter security requirements.

2.2. Consensus Mechanism

A consensus algorithm is defined as the predefined specific validation process used to assign the consensus to the miners (Di Lucca and Tortorella, 2021). This means that the consensus algorithm is applied when a new block is added to the blockchain, to ensure that the block is valid, a correct chaining of transactions and to provide a guaranteed unique order. This section analyses the different consensus mechanisms used in each smart contract platform and discusses their strengths and weaknesses.

Even though an upgrade to Proof-of-Stake (PoS) is being deployed for *Ethereum 2.0*, *Ethereum* currently uses the Proof-of-Work (PoW) consensus algorithm (Di Lucca and Tortorella, 2021), which is discussed in this report. PoW is a decentralised consensus mechanism that requires members of a network to expend effort solving an arbitrary mathematical puzzle to prevent anybody from gaming the system (Di Lucca and Tortorella, 2021). PoW at scale requires huge amounts of energy, which only increases as more miners join the network. Therefore there is an issue with scalability for platforms employing this consensus algorithm.

Hyperledger Fabric adopts a democratic system and applies the Byzantine Fault Tolerance (BFT) consensus mechanism, on the basis of the Byzantine Generals' Problem (Khan et al., 2021). BFT derives from the qualified majority (50% +1) of the miners. The major advantage of BFT is that it is able to continue operating even if some of the nodes fail or act maliciously (Khan et al., 2021).

Corda's consensus mechanism works by proving that a transaction is both valid and unique (Di Lucca and Tortorella, 2021). The validity consensus checks that, for the proposed transaction and for every transaction in the backchain that generated the inputs of the proposed transaction, the transaction is accepted by the contracts of every input and output state and has all the required signatures. A backchain is a link, specific to *Corda*, that connects an input back to its issuance transaction, which does not require any input and is signed by the initial state participants (Brown, Carlyle and Hearn, 2016). The uniqueness consensus is when a notary checks that a node has not used the same input for multiple transactions (Di Lucca and Tortorella, 2021).

NEM employs the Proof of Importance (PoI) consensus algorithm, a process similar to PoS. PoS requires nodes to prove the ownership of a certain amount of token to participate in the validation of transactions (Khan et al., 2021). The nodes with more tokens will have a higher chance to be the validator of the next block. PoI builds on top of PoS by adding some new ranking criteria and features.

2.3. Permission Type

The permission type of a smart contract platform is the policy for defining user roles and permissions within the blockchain. A blockchain can be Permissionless or Permissioned. Permissionless blockchains allow anyone to have access to the blockchain and no approval from any authority is required; there is no central owner/control of the network and software (Di Lucca and Tortorella, 2021). Permissioned blockchains require that an administrator will approve the access to the network, thus transaction validators are pre-selected by blockchain administrators. A permissioned blockchain can be a public, or open, one (i.e., a blockchain which anyone can access and view, but where only authorised participants are permissioned to generate transactions and/or update its state), or closed one (usually with restricted access and only the administrator is permissioned to generate transactions and/or update the state) (Di Lucca and Tortorella, 2021).

Ethereum is a public, permissionless blockchain (Khan et al., 2021). *Hyperledger Fabric* has a permissioned architecture, with an open smart contract model allowing to implement different solution models, such as account model, UTXO model, structured/unstructured data (Khan et al., 2021). *Corda* is a permissioned and highly secured blockchain platform (Di Lucca and Tortorella, 2021). *NEM* can support both a permissioned or permissionless blockchain, depending on the purpose of the platform (Khan et al., 2021).

2.4. Scalability

Scalability is the ability to increase performance of the blockchain by adding resources such as sharding, side chains. Prior to assessing the scalability of different smart contract platforms, we review the existing performance of smart contracts platforms in terms of throughput - Transactions Per Second (TPS) and some proposed solutions to scale them.

Smart Contract Platform	Consensus Protocol	Permission Type	Transactions Per Second	Reference
Ethereum	Proof of Stake	Permissionless	15-20	(Buterin 2013)
Hyperledger Fabric	Practical Byzantine Fault Tolerance	Permissioned	3500	(Androulaki et al. 2018)

NEM	Proof of Importance	Permissioned	100	(Deval et al. 2019)
Corda	Raft	Permissioned or Permissionless	170	(*Transactions Per Second (TPS)* 2018)

Table 1. Performance comparison of Smart Contract Platform

While *Ethereum* can process up to 15-20 TPS which is more than Bitcoin (7 TPS), it does not fare well compared to permissioned smart contract platforms such as Hyperledger Fabric, NEM and Corda, as presented in ***Table 1***. *Hyperledger Fabric*, which uses PBFT as consensus mechanism works on small sets compared to PoS, to achieve high performance in terms of transactions per second as 3500 TPS, but unpredictable state and space scalability has remained an issue (Androulaki et al., 2018). *NEM* project which uses PoI as a consensus algorithm builds on top of PoS to account for interest of nodes while transacting, which is the value of frequency and value of transacting of a node. *Corda* has a “pluggable” uniqueness service that enables validation of transactions for states optional, thereby removing the need to see the entire content of transactions and improving scalability.

In public networks such as Ethereum, a large number of nodes verify the transaction whereas in permissioned networks, only a limited number of nodes verify the transactions. A small set of nodes complete the transaction verification process faster compared to a large set of nodes, but this also means more centralization. Degree of centrality measures how connections between are distributed across nodes in a network and smaller set of nodes implies lesser number of hubs and more centralization. Hence, these permission-based platforms (Hyperledger Fabric and Corda) achieve a better performance than Ethereum but do not currently scale enough to attain the performance of Visa which can achieve 24000 TPS.

Sharding is a solution that divides the network into small groups, called shards to participate each shard simultaneously to take part in the validation of the transactions. Sharding eliminates the need of the entire blockchain to process the transaction thus increasing the TPS on the blockchain. Ethereum have plans and implementations to implement sharding, which can improve Ethereum network’s TPS by 100x and Hyperledger Fabric has had applications of sharding to improve its performance (Luu et al. 2016, Dang et al. 2019, Androulaki et al. 2018a). However, there are security concerns of implementing sharding as if an attacker can gain a majority on any shard they comprise the whole system. By design, validators on one shard accept the majority decision of other shards.

2.5. Data Storage

HyperLedger Fabric's world state is implemented as a database. It provides efficient storage and retrieval of ledger states as the database implementation can vary accordingly. CouchDB and LevelDB are options for its world state, it could be graph store, or a relational data store or a temporal database (*Ledger — hyperledger-fabric docs master documentation* n.d.). Data storage is pluggable in *Corda*, only data relevant to the node's owner is contained by each node. Community Corda has the support of H2 and Postgres. Corda Enterprise supports even more databases: Azure SQL, SQL Server, Oracle and PostgreSQL. It means that it's possible to store ledger data in an existing database and use all the capabilities of relational databases to query and join the data from both the ledger and the application (Derecha 2021). Corda allows the application data to combine with ledger data. *Ethereum* on the other hand, manages and stores data using tries- storage trie, state trie and transaction trie. The state trie consists of a key and value for every account which is present on the Ethereum account. Storage trie is where the contract data is present. Every ethereum account has its storage trie. 256-bit hash of the storage trie's root node is stored in the global state trie as storageRoot.

3. Conclusion

As shown above, every platform (including those not mentioned in this paper) is unique in a multitude of ways that suit different application scenarios. Example application fields include banking and finance, Internet of Things, healthcare, etc (Wang et al., 2018). As the world transitions into a digital world with priorities in user privacy and integrity, blockchains and smart contracts are also getting significantly more popular, especially with the surge of attention towards cryptocurrencies.

3.1. Future Directions

That said, this technology is still fairly new and thus still has numerous issues pertaining to data security, computational and network costs, administration, and storage expenses, to name a few (Suvitha and Subha, 2018). These issues could provide researchers with potential directions for research to better optimise the blockchain and smart contract technology.

Apart from that, as the digitisation of the financial world progresses and with more people hopping onto the blockchain and smart contract trend, integration with other technologies such as parallel computing (Wang et al, 2018) could also provide better scalability and allow for much more efficient use of these technologies.

Inversely, blockchain could also be further used to improve other existing technologies such as Artificial Intelligence (Salah, Rehman, Nizamuddin and Al-Fuqaha, 2019) where blockchain technologies can help to improve reliability, security, transparency, trust, and management.

4. References

Ahmed, M., 2021. Ethereum High-Level Languages. [online] Medium. Available at: <https://medium.com/@bloggingtech260/ethereum-high-level-languages-e2752718cd91>.

Androulaki, E, Cachin, C, De Caro, A & Kokoris-Kogias, E 2018a, ‘Channels: Horizontal Scaling and Confidentiality on Permissioned Blockchains’, *Computer Security*, pp. 111–131.

Androulaki, E, Manevich, Y, Muralidharan, S, Murthy, C, Nguyen, B, Sethi, M, Singh, G, Smith, K, Sorniotti, A, Stathakopoulou, C, Vukolić, M, Barger, A, Cocco, SW, Yellick, J, Bortnikov, V, Cachin, C, Christidis, K, De Caro, A, Enyeart, D & Ferris, C 2018b, ‘Hyperledger fabric’, *Proceedings of the Thirteenth EuroSys Conference on - EuroSys ’18*.

Brown, R 2018, *The Corda Platform: An Introduction*.

Brown, R, Carlyle, J, Grigg, I & Hearn, M 2016, *Corda: An Introduction*.

Buterin, V 2013, *Ethereum Whitepaper*, ethereum.org.

Dang, H, Dinh, TTA, Loghin, D, Chang, E-C, Lin, Q & Ooi, BC 2019, ‘Towards Scaling Blockchain Systems via Sharding’, *Proceedings of the 2019 International Conference on Management of Data*.

Davies, A 2022, *What are the 5 Best Smart Contract Platforms for 2022?I DevTeam.Space*, DevTeam.Space, viewed 26 May 2022, <https://www.devteam.space/blog/what-are-the-5-best-smart-contract-platforms-for-2022/>.

Derecha, V 2021, *Distributed Ledger Frameworks Comparison: Corda vs Hyperledger Fabric*, Eleks Labs.

Deval, V., Dwivedi, V., Dixit, A. and Norta, A., Consensus Mechanism and Scalability Issues for Mobile Smart Contracts: A Systematic.

Di Lucca, G. and Tortorella, M., 2021. Comparing Blockchain Platforms for Smart Contracts: A Preliminary Framework. Proceedings of ISCA 30th International Conference on Software Engineering and Data Engineering, 77, pp.104-114.

Harz, D. and Knottenbelt, W., 2022. *Towards Safer Smart Contracts: A Survey of Languages and Verification Methods*. [online] Available at: <https://arxiv.org/pdf/1809.09805.pdf>.

Julian 2018, *Decentralization: NEM vs Ethereum*, NEM News Website, viewed 26 May 2022, <https://nemflash.io/decentralizationnem-ethereum>.

Khan, SN, Loukil, F, Ghedira-Guegan, C, Benkhelifa, E & Bani-Hani, A 2021, 'Blockchain smart contracts: Applications, challenges, and future trends', *Peer-to-Peer Networking and Applications*, vol. 14.

Kosba, A., Miller, A., Shi, E., Wen, Z. and Papamanthou, C., 2016. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/7546538> .

Ledger — hyperledger-fabricdocs master documentation n.d., hyperledger-fabric.readthedocs.io, viewed 23 May 2022, <https://hyperledger-fabric.readthedocs.io/en/release-1.3/ledger/ledger.html>.

Luu, L, Narayanan, V, Zheng, C, Baweja, K, Gilbert, S & Saxena, P 2016, 'A Secure Sharding Protocol For Open Blockchains', *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

Luu, L., Chu, D., Olickel, H., Saxena, P. and Hobor, A., 2016. Making Smart Contracts Smarter | Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. [online] ACM Conferences. Available at: <https://dl.acm.org/doi/abs/10.1145/2976749.2978309>.

Morrison, R, Mazey, NCHL & Wingreen, SC 2020, 'The DAO Controversy: The Case for a New Species of Corporate Governance?', *Frontiers in Blockchain*, vol. 3.

Parizi, RM, Amritraj & Dehghantanha, A 2018, ‘Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security’, *Lecture Notes in Computer Science*, pp. 75–91.

Salah, K, Rehman, MHU, Nizamuddin, N & Al-Fuqaha, A 2019, ‘Blockchain for AI: Review and Open Research Challenges’, *IEEE Access*, vol. 7, pp. 10127–10149.

Suvitha, M & Subha, R 2021, *A Survey on Smart Contract Platforms and Features*, IEEE Xplore, vol. 1, pp. 1536–1539, viewed 26 May 2022, <https://ieeexplore.ieee.org/document/9441970>.

Tam, KC 2018, *Ethereum and NEM: Two Different Approaches for Blockchain Application Development*, Medium, viewed 26 May 2022, <https://kctheservant.medium.com/ethereum-and-nem-two-different-approaches-for-blockchain-application-development-c36135df21cb>.

Transactions Per Second (TPS) 2018, Corda, viewed 22 May 2022, <https://www.corda.net/blog/transactions-per-second-tps/>.

Valenta, M & Sandner, P 2017, *Comparison of Ethereum, Hyperledger Fabric and Corda*.

Vigliotti, MG 2021, ‘What Do I Talk About When I Talk About Smart Contracts? Open Challenges in Smart Contracts’, *Frontiers in Blockchain*, vol. 3.

Wang, S, Yuan, Y, Wang, X, Li, J, Qin, R & Wang, F-Y 2018, ‘An Overview of Smart Contract: Architecture, Applications, and Future Trends’, *2018 IEEE Intelligent Vehicles Symposium (IV)*.