Analysis of a Non-Fungible Token centric Blockchain Architecture and Comparison with a General Purpose Blockchain

1st Ricardo Lopes Almeida
Università di Camerino
Università di Pisa
Camerino and Pisa, Italia
ricardo.almeida@unicam.it

2nd Fabrizio Baiardi
Dipartimento di Informatica
Università di Pisa
Pisa, Italia
fabrizio.baiardi@unipi.it

3rd Damiano Di Francesco Maesa
Dipartimento di Informatica
Università di Pisa
Pisa, Italia
damiano.difrancesco@unipi.it

4th Laura Ricci
Dipartimento di Informatica
Università di Pisa
Pisa, Italia
laura.ricci@unipi.it

***Abstract*—Blockchain is the revolutionary technology that enabled the first true digital currency, i.e., cryptocurrency, a concept that has forced changes in the world. Cryptocurrencies are no longer amateur experiments and became valid investments that compose an increasingly larger share of investment portfolios. Cryptocurrencies are based in *Fungible Token (FT)*, a type of blockchain tokens. But these tokens can be programmed differently and transformed into *Non Fungible Tokens (NFTs)* instead. NFTs are among the most recent and promising additions to the blockchain universe. While FTs are used to implement digital currencies, where these tokens emulate coins and bank notes used in normal, fiat currencies, NFTs are used to establish ownership of objects, digital or otherwise. Non-fungible counterparts can be used to store data directly in the blockchain. This is how they establish the ownership relations and how the digital or physical object whose ownership is established by the token are identified. This operational aspect provide NFTs with a larger landscape of possible implementation architectures than the fungible version. This paper focuses on an alternative NFT architecture introduced by Flow, the first blockchain created with NFT support as its central tenet, and it works and how it compares to Ethereum, one of the most popular and NFT supporting public blockchains.**

***Index Terms*—Blockchain, Non-Fungible Tokens, Ethereum, Flow, Solidity, Cadence**

I. Introduction

Non Fungible Tokens (NFTs) are one of the latest major features introduced in the decentralised blockchain universe. NFTs follow the tokenisation trend introduced by Bitcoin [1], the first example of a public blockchain and cryptocurrency, and establish an almost antagonist concept to the latter. Cryptocurrencies are established in blockchains with *Fungible Tokens (FT)*, a concept akin to coins and notes used with fiat currencies. As such, FTs are interchangeable and can be divided in subunits without loss of value, just like regular money. NFTs on the other hand, as the name implies, cannot be subdivided or exchanged by another token with similar value. A good analogy for NFTs are works of art, such as painting and sculptures, that is, unique objects valued individually, irregardless if all were produced by the same artist, just like each NFT has its own characteristics and value independent of the contract that issued it.

NFTs are implemented through smart contracts, which are software scripts that can be deployed, i.e., stored, in a blockchain block and executed distributively by the active nodes in the network. In a smart contract enabled blockchain, smart contract instructions are sent to execute in a subset of active nodes and their results are then confirmed and validated by the rest of the network towards achieving a computational consensus. The aggregated resources of a blockchain network, e.g., computational power, memory, storage, etc. are often referred as a *Blockchain Virtual Machine*. Most modern blockchains implement Turing-complete virtual machines used to run smart contracts deterministically and implement non-fungible tokens.

The first example of an NFT was introduced by a pair of digital artists through the Quantum project [2]. They used Namecoin, a blockchain forked from Bitcoin, but with the addition of a *blockchain transaction database* that could be used to store non-transactional data [3]. This allowed the artists to store a digital produced image directly in the blockchain and use the transactional data to establish a unique ownership relation between the record and an account address, thus creating the first real world instance of a *Non-Fungible Token*. The NFT concept was known at the time and, as such, this project was soon

TODO: ADD THE CAMERINO INFO HERE

followed by other, more technologically sound, projects. The majority took advantage of Ethereum and the smart contract support introduced when this blockchain, which promised a easier method to create NFTs than the one used in Quantum. It did not took much for Ethereum to become the most popular public blockchain for NFT-based projects. Currently, the top 10 NFT projects ranked by market cap are all Ethereum based [4].

Event though Ethereum and smart contracts added significant programmatic flexibility to the universe of blockchain solutions, it is still a notoriously difficult network to scale and it tries to establish a rate of one block per 12 15 seconds, which can limit significantly the throughput of the chain, as well as the scope of applications supported. One of such applications was the *CryptoKitties* project launched back in 2017 [5], one of the earliest NFT contracts deployed in Ethereum. This project was created by *Dapper Labs* (known previously as *Axiom Labs*) and extended the NFT concept with a new usability layer that was absent from other similar projects. The contract minted a *CryptoKitty*, a NFT representing a digital cat-like creature, and each kitty token was characterised by a unique genome parameter, an internal 256-byte string from which the visual characteristics of the token were derived from. Parameters as eye color, skin color, ear type, etc were encoded in portions of the genome string. The innovative aspect of this project was that two CryptoKitties could be "bred" to generate a new one with a genome string that derived from the parent's genome. Dapper Labs established the genome mechanics such that new traits were acquired somewhat randomly (pure randomness is hard to achieve in a purely deterministic blockchain environment) and some traits were rare and hard to obtain than others. This new approach translated into a peak of popularity and a surge in Ethereum transactions, which were used to acquire new kitties and breed new pairs, and that end up exposing the scalability limitations of Ethereum [6].

Dapper Labs initially tried to solve these issues from within the Ethereum blockchain, but at some point it became clear that the blockchain needed architectural modifications to be able to overcome these throughput limitations. As such, instead of trying to "fix" Ethereum, Dapper Labs launched Flow in 2020 instead [7], a new blockchain solution developed from scratch centered around supporting NFTs and related mechanics. Flow presents several key differences from Ethereum, from how nodes behave in the network, the consensus algorithm used, to how data is stored and accessed in the chain, as well as similarities, such as defining and using a native cryptocurrency token to regulate blockchain operations (*gas* in Ethereum) and smart contract support. Yet, Flow claims to present the same level of NFT functionalities as Ethereum and other similar NFT ready blockchains, but approaching the concept from a fundamental different point.

This article presents an introduction to the Flow blockchain and how it organises itself from the architecture standpoint, followed by the introduction and analysis of a pair of simple implementations of a NFT minter smart contract, one in Cadence, Flow's smart contract programming language, and another in Solidity, Ethereum's equivalent. We finish this publication by comparing both architectures and implementations towards determining the merits of Flow's claims be a viable and optimised alternative from NFT-based projects.

The rest of this article is structured as follows: Section II provides an overview of publications relevant to this work. In Section III we provide an introduction to Flow, the blockchain central to this exercise. Section IV provides a comparison between implementations of NFT smart contracts in both Cadence and Solidity, as well as the supporting blockchain architectures, namely Flow and Ethereum. This article concludes with Section V.

II. Related Works

Academic research using Non-Fungible Tokens is as recent as the concept itself. The Quantum project that produced the first NFT happened a decade ago, therefore NFT-based research is necessarily younger. But even considering such small temporal window, the research community did produce a significant number of relevant publication that used NFTs in some capacity. The potential of NFT technology is evident, so it did not took long before researchers begun publishing NFT-based solution for known problems. The authors in [8], [9], and [10] explore a tokenisation approach to manage real estate, where houses, apartments, land plots, etc. are abstracted by NFTs since the mechanics used to operate with NFTs in a blockchain are quite similar to how real estate markets work, and real estate properties share the uniqueness and individuality of NFTs as well. A similar approach is followed by [11] where they use NFTs to abstract pharmaceutical products and use a *digital twin* approach to ensure the traceability of a given product by mirroring the lifecycle of a NFTs within a blockchain with a series of checkpoints, as the product goes from the production line to where it is going to be distributed. This tokenisation trend continues with [12], where a similar strategy is used to propose a energy management system for microgeneration cases. The authors developed a blockchain-based environment where they used NFTs to abstract actors in the system, i.e., solar panels, battery packs, wind turbines, consumers, utility companies, etc. and the values exchanged in the system, i.e., electric energy and money, are abstracted with cryptocurrencies. Another similar example using the same strategy is found in [13] where NFT tokenisation happens in a event management system where these are used to abstract event tickets, taking advantage of the same uniqueness and individual elements, such as allocated seat, event name, id number, etc, that characterise these tickets and how they can be encoded into the metadata of an NFT.

Other opted for a higher lever approach and presented an analysis based on the architectural aspects of NFTs rather than use them as a simple building block in a solution. [14] explores the architectural aspects of specific NFT implementations in the Hyperledger environment, a framework to develop private custom blockchains. [15] and [16] present a similar high level architectural approach but with a specific scope in mind, namely, tracing and value transfer applications. Even though NFT is a recent technology, [17] and [18] presented *systematisation of knowledge (SoK)* articles about this technology, but they did not approached any architectural aspects of the technology. All publication mentioned thus far used Ethereum and Hyperledger for the examples and prototypes developed, and none even mentioned Flow as an alternative. In that regard, to the time of this writing, we found no academic publications using Flow as a development platform, let alone exploring if the new architectural approach could benefit their solutions. The few mentions to Flow in academic examples came from review style papers, namely [19], [20], and [21], the latter providing the most extensive explanation.

A. Our Contribution

This paper presents a detailed exploration of the Flow blockchain as an architectural alternative to implement NFTs. To illustrate the differences, we also use Ethereum as an example of a general-purpose blockchain for comparison. We also present a concrete example of a simple NFT contract using Cadence, the smart contract programming language used in Flow, and compare it to a Solidity version of a functionally equivalent NFT contract. This exercise finalises with the analysis of the results obtained.

III. Flow Blockchain

This section goes into detail about the functional aspects of the Flow blockchain and how they differentiate this blockchain from others that support NFTs. This assumes a general knowledge of blockchain technology from the reader and therefore, basic details about blockchain workings are going to be omitted.

A. Consensus Protocol

The first public blockchains in existence implemented *Proof-of-Work (PoW)* consensus protocol, where nodes solve cryptographic puzzles towards getting the privilege of publishing the next block and getting any rewards that usually follow. In the years that preceded the release of Flow, this protocol fell out of favor among the blockchain universe due to the high levels of energy waste that it entitles. Computations executed in the pursuit of solving these puzzles have no use whatsoever and the popularity of PoW blockchains such as Bitcoin exacerbated this issue.

The blockchain community reacted to this by proposing alternative consensus mechanisms, such as *Proof-of-Authority (PoA)* where the nodes maintain a reputation system in the network and associating the odds of being selected to insert the next block proportional to the reputation value, *Proof-of-Elapsed Time (PoET)*, where the odds of round selection are proportional to the time the node has been waiting for selection, or *Proof-of-Stake (PoS)*, where the probability of a node being selected in a round is proportional to the amount of native cryptocurrency staked, among other less known [22].

PoS was one of the first alternatives to PoW proposed around the time when Flow was being developed. Also, around that time Ethereum announced a future fork of its chain to switch the consensus protocol to PoS in the new stream. As such, Flow was created and made available from the beginning with a PoS consensus mechanism.

B. Flow Node Roles

The scalability and efficiency claims of Flow derive from an innovative four-node type architecture used to pipeline executions in the network. The increase in role types, as opposed to the 2-node architecture used by Ethereum, sacrifices some redundancy and adds a small increase in complexity in return for gains in speed, throughput and scalability, while maintaining minimal operational costs.

Flow differentiate its nodes into: *consensus nodes* that decide the presence and order of transactions in the blockchain, *collection nodes* to enhance network connectivity and data availability for applications, *execution nodes* that perform the computations required in transactions, and *verification nodes* to validate the computations returned from the execution nodes [23]. Delegated computations are validated with *Specialised Proofs of Confidential Knowledge (SPoCK)*, a type of non-interactive zero-knowledge proofs based on the *Boneh-Lynn-Shacham (BLS)* signature scheme. These were developed by the Flow creators for this specific purpose [24].

C. Cadence Language

Flow's version of Ethereum's Solidity is called Cadence. It is a programming language used to write smart contracts in Flow, as well as the scripts and transactions used to interact with the blockchain and deployed smart contracts. Ethereum uses Solidity only for smart contract development. To interact with the blockchain, i.e., invoke a function, access a public parameter, etc, from a deployed contract, Ethereum API is compatible with several general purpose languages, such as Python and Javascript, for that purpose. Flow integrates all these operations in its own programming language. Cadence implements a *Resource Oriented Programming Paradigm* through a strongly static type system, with built in access control mechanics that can be further specialised, i.e., to narrow the scope of allowed users, through the utilisation of *capability-based* security. Cadence syntax was inspired by modern general-purpose programming languages such as *Swift*, *Kotlin*, and *Rust* [25] [26]. Files written in Cadence, namely smart contracts, transactions and scripts, have the `.cdc`

extension. The following sections go into greater detail about the concepts introduced thus far.

1) Smart Contracts: Smart contracts in Flow serve the same purpose as in Ethereum and other blockchains with similar support. Syntactically these are quite different from Solidity contracts, but functionally they are very similar: both use the **contract** keyword to define the main contract structure, define constructor functions that executed automatically once during deployment and can extend their functionalities by importing external contracts and interfaces. They do have differences as well, such as, for example, Cadence does not require the implementation of default destructors in their contracts, unlike Solidity, because Flow blockchain has storage management functionalities that automatically manage blockchain storage when resources or other contracts are deleted/destroyed. Smart contracts deployed in Flow stay initially in an "updatable" phase. During it, the code can be changed in the block storing it. Once the developer(s) are satisfied with the contract's performance, they can *lock* it, thus preventing any future changes. This allows for cleaner deploys and optimise blockchain storage, unlike Ethereum that simply saves any updated versions of a contract in a new block. If the developers do not care to delete the old contract, the blockchain simply keeps these older versions in storage.

2) Interacting with Flow Smart Contracts: Flow differentiates the two types of blockchain interactions with different file types: if an interaction is limited to read operations, i.e., executing the instructions in the file **does not changes** the state of the Flow blockchain, then a *script* should be used. If the set of instructions to execute **change the state** of the Flow blockchain (by saving, modifying or deleting data), a *transaction* should be used instead. The main difference between these files is that *transactions* need a valid digital signature to execute and usually require funds to be used as gas as well since Flow, just like Ethereum and others, restrict modifications to digital objects in storage to only the owners of such object, hence it is critical that the owner signs the transactions first. *Scripts* in Flow are "free" to execute, i.e., they do not consume gas, and therefore they do not require any signatures and can be executed by anyone.

3) Resource Oriented Paradigm: Cadence establishes its programming paradigm through a digital object that has a special status among others named *Resource* which was inspired by Rust's *linear types*. From a functional point of view, Cadence Resources are similar to Objects in any Object-Oriented programming language. The main difference resides in the control exercised by the language to ensure that Resources are unique in the blockchain environment. Once created, a resource cannot be copied, only moved or destroyed. Since Resources cannot be copied, they also can only exist in one location at a time, often saved in an account's storage. Resources can only be created through a smart contract function. Cadence uses the **create** keyword to create an Resource and this keyword can only be used in a smart contract. A smart contract creates a resource by first establishing a function that does creates and returns a given resource, and using a transaction to invoke such function. Creating a Resource changes the state of the blockchain, therefore they requires transactions to execute. Currently only Flow uses this paradigm and it is the main method used to create NFTs. As such, Flow NFTs are not yet compatible with other blockchain due to this fundamental architectural difference.

4) Cadence Types: Cadence was developed as a type-safe language and this is part of the strategy to individualise and uniquely identify each digital object in the blockchain. Cadence uses basic types, similar to Solidity, for basic data elements, such as integers, strings, floats, etc. The nomenclature is slightly different but consistent with Solidity. An unsigned 256 bit integer in Solidity is preceded by **uint256**, but in Cadence the equivalent type is **UInt256**. Complex types, e.g, resources, events, structs,etc., have complex types resulting from the concatenation between the name of the implementing contract and the name of the element. Since elements such as resources, structs and events can only be created through a contract that contains their implementation details, Flow uses this to its advantage. Pairing the contract name with the resource name creates a unique identifier by default. This ensures type uniqueness withing a Flow projects since contracts saved in the same account storage require different names. To uniquely identify a specific resource in the blockchain, this nomenclature is extended to include the address of the deployed contract prefixed to this identifier. This strategy can uniquely identify every complex type in the blockchain. For example event emitted by transactions are displayed as `A.0ae53cb6e3f42a79.FlowToken.TokensWithdrawn`. This is a complex type concatenating the address of the deployed contract (0x0ae53cb6e3f42a79), the name of the contract (FlowToken) and the name of the event (TokensWithdrawn).

5) Access Control: Cadence implements access control in two levels of granularity. At a coarser level, contracts and objects can limit the access to their inner constituents, e.g., functions, parameters, structs, etc., using the **access** keyword and the scope of access inside parenthesis after. **access(all)** grants public access, i.e., anyone can execute a function or access a parameter preceded by this access modifier. **access(E)**, where E is an entitlement restricts access to only users with such entitlement. Entitlements are custom declarations that can be used to fine tune access control in Cadence. For example, a contract can define a function with an *Admin* entitlement level (**access (Admin)**) restrict its use to only users able to provide the *Admin* entitlement. Moving deeper into Cadence access control, the next level is **access(account)**. This limits the access of the element to other elements existing within

the same account, like for example another smart contract saved in the same account storage. **Access(contract)** narrows the access further by limiting the element to be accessible only to other elements in the same contract. A function labeled with this modifier can only be invoked by other functions and resources defined in the same contract. The last and most restrictive modifier is **access(self)** which narrows the scope of access to the element and its children. Elements at the same level in the contract cannot "see it" even. For a more fine grained access control, Cadence has developed a *Capability* system in a similar fashion as other legacy capability based programming languages. Sec. III-C6 goes into greater detail about it.

6) Capabilities:

D. Token Standards

E. Account-based Storage Model

1) Storage Domains and Paths:

2) References:

IV. Architecture Comparison

ARCHITECTURE COMPARISON [27]

V. Conclusion

CONCLUSION [28]

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *www.bitcoin.org*, pp. 1–9, 2008.

[2] J. Exmundo. (2023, 3) Quantum, the story behind the world's first nft. [Online]. Available: https://nftnow.com/art/quantum-the-first-piece-of-nft-art-ever-created/

[3] A. Loibl, "Namecoin," in *Seminar Innovative Internettechnologien und Mobilkommunikation SS2014*, 2014, pp. 107–113.

[4] CoinGecko.com. Top nft collection prices ranked by marked cap. [Online]. Available: https://www.coingecko.com/en/nft

[5] D. Labs, "Crytokitties: Collectible and breedable cats empowered by blockchain technology," Dapper Labs, Tech. Rep., 2017.

[6] bbc.com. Cryptokitties craze slows down transactions on ethereum. [Online]. Available: https://www.bbc.com/news/technology-42237162

[7] R. Gharegozlou. Introducing flow, a new blockchain from the creators of cryptokitties. [Online]. Available: https://medium.com/dapperlabs/introducing-flow-a-new-blockchain-from-the-creators-of-cryptokitties-d291282732f5

[8] N. N. Hung, K. T. Dang, M. N. Triet, K. V. Hong, B. Q. Tran, H. G. Khiem, N. T. Phuc, M. D. Hieu, V. C. Loc, T. L. Quy, N. T. Anh, Q. N. Hien, L. K. ṅd Bang, D. P. Nguyen, N. T. Ngan, X. H. Son, and H. L. Huong, "Revolutionizing real estate: A blockchain, nft, and ipfs multi-platform approach," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14416 LNCS. Springer Nature Switzerland, 2023, pp. 68–73. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-48316-5_10

[9] D.-E. Bărbuță and A. Alexandrescu, "A secure real estate transaction framework based on blockchain technology and dynamic non-fungible tokens," in *2024 28th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2024, pp. 558–563.

[10] A. Sharma, A. Sharma, A. Tripathi, and A. Chaudhary, "Real estate registry platform through nft tokenization using blockchain," in *2024 2nd International Conference on Disruptive Technologies, ICDT 2024*. IEEE, 2024, pp. 335–340.

[11] F. Chiacchio, D. D'urso, L. M. Oliveri, A. Spitaleri, C. Spampinato, and D. Giordano, "A non fungible token solution for the track and trace of pharmaceutical supply chain," *Applied Sciences (Switzerland)*, vol. 12, pp. 1–23, 2022.

[12] N. Karandikar, A. Chakravorty, and C. Rong, "Blockchain based transaction system with fungible and non-fungible tokens for a community-based energy infrastructure," *Sensors*, vol. 21, p. 32, 2021.

[13] F. Regner, A. Schweizer, and N. Urbach, "Nfts in practice-non-fungible tokens as core component of a blockchain-based event ticketing application completed research paper," in *Proceedings of the 40th International Conference on Information Systems*, 2019, pp. 1–17.

[14] S. Hong, Y. Noh, and C. Park, "Design of extensible non-fungible token model in hyperledger fabric," in *SERIAL 2019 - Proceedings of the 2019 3rd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2019, pp. 1–2.

[15] L. Yang, X. Dong, Y. Zhang, Q. Qu, and Y. Shen, "Generic-nft : A generic non-fungible token architecture for flexible value transfer in web3," *TechRxiv*, pp. 1–7, 2023.

[16] "Nftracer: A non-fungible token tracking proof-of-concept using hyperledger fabric," *arXiv: 1905.04795v1*, pp. 1–9, 2019. [Online]. Available: http://arxiv.org/abs/1905.04795

[17] G. Wang and M. Nixon, "Sok: Tokenization on blockchain," in *Proceedings for the 2021 IEEE/ACM14th International Conference on Utility andCloud Computing (UCC '21) Companion (UCC '21 Companion)*. ACM, 2021, pp. 1–9.

[18] K. Ma, J. Huang, N. He, Z. Wang, and H. Wang, "Sok: On the security of non-fungible tokens," *arXiv:2312.0800v1*, vol. 1, pp. 1–15, 2023. [Online]. Available: http://arxiv.org/abs/2312.08000

[19] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token: Overview, evaluation, opportunities and challenges," *arXiv*. [Online]. Available: http://arxiv.org/abs/2105.07447

[20] Q. Razi, A. Devrani, H. Abhyankar, G. S. Chalapathi, V. Hassija, and M. Guizani, "Non-fungible tokens (nfts) - survey of current applications, evolution, and future directions," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2765–2791, 2024.

[21] B. Guidi and A. Michienzi, "From nft 1.0 to nft 2.0: A review of the evolution of non-fungible tokens," *Future Internet*, vol. 15, 6 2023.

[22] S. Bouraga, "A taxonomy of blockchain consensus protocols: A survey and classification framework," *Expert Systems with Applications*, vol. 168, pp. 1–17, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.114384

[23] A. Hentschel, Y. Hassanzadeh-Nazarabadi, R. Seraj, D. Shirley, and L. Lafrance. Flow: Separating consensus and compute - block formation and execution. [Online]. Available: http://arxiv.org/abs/2002.07403

[24] T. Ben, Y. Riad, and S. Wahby, "Flow: Specialized proof of confidential knowledge (spock)," 2020. [Online]. Available: https://eprint.iacr.org/2023/082

[25] F. blockchain development team, "Flow prime," Flow, technical report, 2020. [Online]. Available: https://flow.com/primer

[26] (2023) The cadence programing language. [Online]. Available: https://cadence-lang.org/docs/language/

[27] K. Sadia, M. Masuduzzaman, R. K. Paul, and A. Islam, "Blockchain-based secure e-voting with the assistance of smart contract," in *IC-BCT*, 2020, pp. 161–176.

[28] W.-J. Lai, Y.-C. Hsieh, C.-W. Hsueh, and J.-L. Wu, "Date: A decentralized, anonymous, and transparent e-voting system," in *Proceedings of the 2018 1st IEEE International Conference in Hot Information-Centric Networking*, 2018, pp. 24–29.