Contract-based Storage versus Account-based Storage: a Comparison among Implementation Strategies for Non-Fungible Tokens

Ricardo Lopes Almeida[1], Fabrizio Baiardi[2], Damiano Di Francesco Maesa[3], and Laura Ricci[4]

[1, 2, 3, 4]Dipartimento di Informatica, Università di Pisa, Italia
[1]Università di Camerino, Italia

November 1, 2024

Abstract

Blockchain and Distributed Ledger Technology (DLT) has been on the forefront of the latest advancements in commercial cryptographic applications. These technologies produced the first instances of decentralised digital currencies, i.e., cryptocurrencies, but these are but one of many potential applications of this technology. Non-Fungible Tokens alongside with the decentralised Virtual Machine idea increased considerably the scope of possibilities offered by blockchain and DLT.

Non-Fungible Tokens (NFTs) surge as a functional inversion of "normal" cryptocurrency tokens and thus also provide a new spectrum of utilization that is still being explored in research. The non-fungibility aspect alongside with a blockchain implementation gives NFTs a "digital uniqueness" that was impossible to achieve with centralised approaches and the potential uses for such characteristics are still being investigated. NFTs can provide a clear and simpler mechanism to establish ownership of objects, digital and otherwise, in a blockchain. The potential for this technology warrants a proper investigation of these tokens.

This paper selected two commercial and public blockchains with well known NFT capabilities and use them to create, deploy and compare NFT implementations. For this purpose Ethereum, one of the most experienced, popular and general-purpose blockchains for research; and Flow, a newer, smaller, less popular but highly NFT-specialized blockchain, were chosen for this exercise.

1 Introduction

For the last 15 years, blockchain has been one of the most disruptive fields in academic research. The first blockchain applications made available to the

wider public were cryptocurrencies, fully digitalized currencies that could be used to abstract products and services since they had a fiat price associated to it and, therefore, value. But a new way to conduct finances was just the beginning. In parallel with cryptocurrency projects, researchers and enthusiasts begun to experiment with a similar but functionally different token-based concept. Shortly after the release of Bitcoin, the first blockchain cryptocurrency in history, the first Non-Fungible Token project actually was based in the Namecoin blockchain, an offshoot project that sat somewhere between Bitcoin and Ethereum. The Quantum project [8] was started by digital artists Jennifer and Kevin McKoy. After creating the image depicted in Fig. 1, they wanted to be able to sell this artwork in its digital form. The main problem was to find a verifiable way to establish the provenance of that digital piece of art. After collaborating with technology experts, they decided to register the work in the Namecoin blockchain, but using a different approach than the one used to establish ownership of cryptocurrencies, which was by far the main application of the early public blockchains of the 2010s. The ownership relations established by registering the image in the blockchain and "locking" its owner to a single account address at all times prove to be transformative. In some aspects, they simply adapted the ownership mechanics of cryptocurrencies, also known as Fungible Tokens, to another type of token that could be used to represent an actual unique object, which can be digital or physical. Given the contradictory operations of these tokens when compared to the Fungible ones, it was only natural to name these as *Non-Fungible Tokens (NFTs)*.

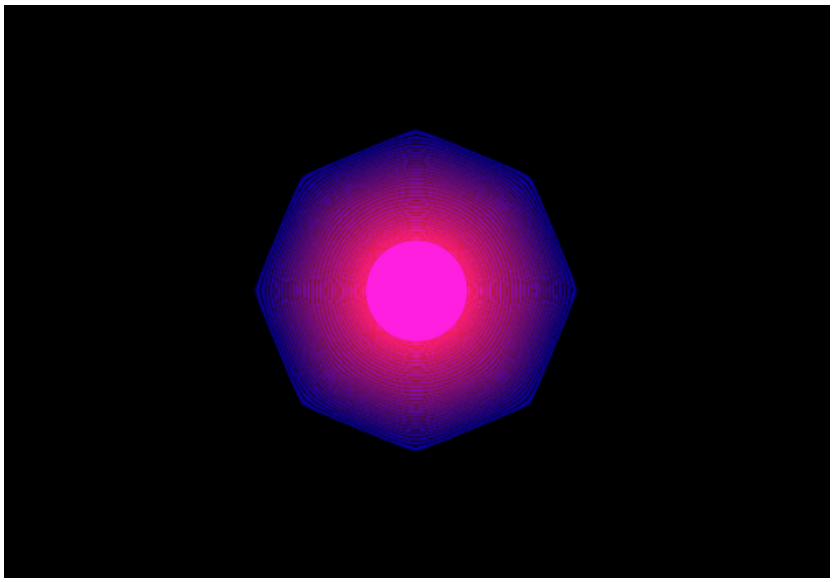

Figure 1: Quantum, the first work of art to be encoded into the metadata of an NFT [8]

Quantum was followed by other, arguably more successful, NFT projects, mostly establishing ownership of digital artworks. Popular projects such as *CryptoPunks* [24] and the *Bored Ape Yatch Club (BAYC)*, followed a similar trend as in Quantum and created finite sets of artistic NFTs that were sold directly to users, or even given away to whomever requested them, as it happen during the early days of *CryptoPunks* due to how unusual it was to buy something with crypto. Both these projects are deployed in Ethereum and transactions with these NFTs use Ethers (ETH), Ethereum's native cryptocurrency.

Both projects minted "static" NFTs in which their metadata, which is used to encode an image, either directly in the blockchain (costly) or indirectly by saving the image in a distributed repository and save its url in the NFT metadata instead (cheaper). The *CryptoKitties* NFT project [18] followed the previous two but with a slight difference. Each *CryptoKitty* was similar to a *CryptoPunk* in the sense that the image encoded in the NFT was created from an internal parameter of the NFT, and not just some random image file. The characteristics of each *CryptoKitty* were derived from an internal "genome", a 256-bit string from which substrings encoding the token characteristics (eye color, fur color, ear type, tail shape, etc.) were derived. But unlike *CryptoPunks*, *CryptoKitties* could be "bred" to generate new ones with characteristics derived semi-randomly from the parents genome. This dynamic, almost gaming, aspect of *CryptoKitties* proved to be quite attractive to collectors and enthusiast, which translated in a peak of NFT activity in Ethereum that pushed the network to and above its limits [3].

The *CryptoKitties* project exposed the limitations of the Ethereum network, particularly regarding scalability. Initially, the team behind the project (initially named *Axiom Zen* but later renamed to *Dapper Labs*) tried to address these problems from withing the Ethereum framework, but with little success. After a while, it became clear that a new blockchain architecture was needed to fully solve the issues encountered and this is how the *Flow* blockchain came to existence.

The approach taken with Flow is quite different than Ethereum from an internal perspective, but for a regular user, interacting with a Flow smart contract or NFT is not much different than with any other NFT. Just like Ethereum introduced *Solidity* as the smart contract programming language for Ethereum contracts, Flow developed *Cadence* with the exact same purpose, albeit with a significant change in how programming principles are approached. The most glaring difference between these two blockchains is how they approach data storage and ownership within the chain. Ethereum stores NFT metadata in a semi-centralised, contract-based fashion, it the sense that the central address is the one where the smart contract that regulates the NFT behavior was deployed, and all storage derives from this point. Flow has a radical approach in this sense, extending the concept of an account from just an address, as it goes in Ethereum, to use this address as a preamble to access a storage area that is unique and individual to each account, i.e., only the owner of the account has access to digital objects stored in the account's storage area. Flow also uses a similar strategy to *gas* to protect and limit the storage space associated to each

account, in which the storage space available is proportional to the amount of cryptocurrency staked in that account. Just as Ethereum uses Ether (ETH), *Flow* uses the *FLOW* token as its cryptocurrency used to pay gas fees and to sustain the account's storage area.

These differences, as well as the fact that the Flow blockchain was created precisely to address the limitations of the Ethereum blockchain regarding NFT interoperability, justify this article. In this work we detail how NFTs are implemented in each of the blockchains considered - Flow and Ethereum - in order to provide an objective comparison between them, as well as infer on how well the Ethereum limitations identified by the *CryptoKitties* project were improved in Flow.

The rest of this article is structured as follows: Section 2 provides an overview of publications relevant to our work in this article. In Section 3 we provide an introduction to concepts that are key in this work that some readers might not be as familiar as needed to fully understand this publication. Section 5 details the construction and deployment of an *ERC-721* compliant NFT Solidity smart contract in the Ethereum network, while Section 6 repeats the same process but for the Flow blockchain, which uses Cadence as the programming language to construct smart contracts. Section 7 provides a summary of our findings, as well as a comparison between the two implementations. This article concludes with Section 8.

2 Related Works

The body of research work around blockchain is quite limited and with NFTs it becomes even smaller due to how recent these technologies are. In some aspects, the technological framework of blockchain as a whole is being developed as we speak. Nonetheless, research in NFT technology and applications is an active field and has produced a significant body of work but centered on specific applications of NFTs. So much so that this in itself as justified the publication of several literature survey studies to make sense of the development of the field of NFT applications.

As example, [9], [23], and [12] provide somewhat extensive surveys of NFT applications, but they do so in a more generalised fashion, with a focus on the potential applications and expected results from the introduction of NFT-based logistics as a new approach to solve existing problems. They do a good job in enumerating the main challenges preventing a wider adoption of the technology and do present a through analysis of the main standards that promote the required interoperability in such applications.

Additionally, [29], [2], and [1] presented similar surveys but with more emphasis in the challenges and opportunities of the technology, but with lesser technical detail. The more detailed surveys are also the more recent, namely [22], and [11]. These provide a thorough synthesis of the evolution of this field, with plenty of references to concrete proposals identifying a specific instance where NFTs where used to achieve a solution.

In all of the cases considered, all emphasised the use of existing NFT standards but their mentions were limited the to the ERC standards from the Ethereum network, which is understandable given that most projects referenced in this articles are based in Solidity smart contracts deployed in the Ethereum network.Though some mention the existence of other blockchains with NFT capabilities, only [29], [22], and [11] mention the Flow blockchain specifically, with [11] providing the most detailed introduction.

2.1 Our Contribution

None of the cases considered made any mentions to the architecture behind how NFTs are implemented, either in Ethereum or any of the alternative blockchains mentioned, nor did any inference on the mechanics of data storage, specifically NFT metadata storage. As far as this writing, no other article to our knowledge was published centered on NFT storage mechanics nor considered other blockchain architectures with sufficient detail to enact a fair comparison. As such, we propose the following contributions to this research field:

1. Compared to other publications in this area, we provide a more focused and detailed introduction to the architectural aspects used to implement a smart contract capable of minting and operating with NFTs, with emphasis on the storage mechanism and control access structures used.

2. We also provide an introduction to a less popular but more application specific blockchain towards establishing a reference point for future comparisons.

3. Given that the Flow blockchain was created towards solving the technological issues that their creators encountered when implementing of the first NFTs projects in the Ethereum network, we also provide an objective comparison between the process and end result of implementing basic NFT projects in both blockchains.

3 Background

The following section provides general definitions to the main concepts that this proposal is built upon. Blockchain technology is still recent but it has matured enough and is sufficiently popular currently to a point that detailed information about it inner workings is actually quite abundant. This section intends to provide sufficient background knowledge for a reader that is not familiar with blockchain technology and related applications.

3.1 Blockchain

A blockchain is, essentially, a distributed database. A blockchain is a common data structure to a set of active nodes that compose a *peer-to-peer (P2P)*

network. Each of these active nodes has a copy of the common image of the blockchain in their internal storage. The common blockchain image refers to the data structure in which all active nodes of the P2P network agree on. Modification to this image, from here on referred to simply as blockchain, cannot be done outside of the protocol that regulate network operations and a node can only participate in that P2P network if it follows every rule established by the blockchain protocol. Protocol violators are punished, often by mechanisms baked into the protocol itself, and often removed from the network itself.

This allows blockchain to have a somewhat paradoxical nature: digital data uniqueness accomplished through data replication. The security and guarantees of uniqueness in blockchain data are proportional to the size of the network. The more active machines exist replicating **the same data structure**, the safer the data is because the higher number of potential subversions an adversary is required to do to modify blockchain data outside of the protocol.

Blockchains are append-only databases. This means that the only way to change the blockchain is to add new data to the existing common image. Deletions and modifications are nor allowed by the protocol. As a consequence, blockchains provide levels of transaction transparency that no other tool of that kind ever did: every transaction gets recorded in the blockchain in perpetuity and it is quite trivial to retrieve the historic of operations that affect a given piece of data.

Users interact with blockchains through transactions and transactions themselves are only executed if they display the correct digital signature of the "owner" of the data that can be affected by it. A simple example is a cryptocurrency transfer. Transferring any amount of cryptocurrency between two blockchain accounts changes the state of the blockchain by changing the current balance of each of the accounts affected. Or more simply, adding the transaction that moves funds from account A to account B to the blockchain is, in itself, a change of the state of the blockchain before anything else. It is important to note that this does not violate the "append-only" nature of blockchains: a blockchain account balance is the result of every transaction that ever affected it. New transactions "pile" on top of old ones but these never get deleted or modified. To allow a cryptocurrency quantity to be transferred, blockchains require only a valid digital signature for the "from" account, since it is the one that is going to get its balance diminished. Blockchain accounts are characterized by a pair of asymmetrical encryption keys, with the private one being used to digitally sign transactions in a similar fashion as with regular documents: the digital signature is the transaction text encrypted with the user's private key.

Data is appended to a blockchain with blocks and each block is a collection of transactions. A transaction is considered executed/validated once a block containing it is added to the head of the blockchain. Some applications only consider a transaction as valid once a certain number of blocks have been added on top of the block with the transaction to be validated. Consecutive blocks are connected, i.e., the "chain" in blockchain, using cryptographic hashes. Every subsequent block displays an hash digest that is calculated by concatenating the data in the current block with the hash digest of the previous block and

applying the hash function to the combination. Fig 2 illustrates this mechanic. A blockchain starts with a genesis block, which is the only one that does not contain a reference to a previous one.

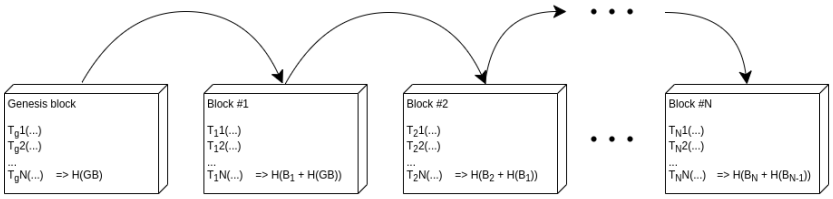

Figure 2: Example of how a generic blockchain is composed of individual blocks cryptographically connected with hash digests.

3.2 Consensus Protocols

Adding a new block is a critical step of the protocol. Only one active node can do it in each round and different protocols have different strategies to select the node allowed to publish the new block. After that, the remaining nodes update their state of the database with the new block added on top, and that becomes the latest common image until the next round. As an example, Bitcoin uses *Proof-of-Work (PoW)* to select which node gets to add the new block. New blocks are added every 10 minutes and active nodes, also known as miner nodes, try to solve a cryptographic puzzle during that period. This is a computationally intensive operation and the first node to find a solution gets to publish the new block. In most public blockchains that support cryptocurrencies, adding a new block also includes a monetary reward in the form of a newly minted cryptocurrency value. With the popularity of Bitcoin and other PoW based cryptocurrencies, this consensus protocol became progressively problematic due to the high energy cost associated to the enormous amount (and ultimately useless) number of computations required to solve these cryptographic puzzles. In part inspired by this, the blockchain community suggested and adapted more efficient consensus protocols, such as *Proof-of-Stake (PoS)*, where the new block is awarded to a node selected from a group that had previously staked some amount of cryptocurrency. The probability of getting a block is proportional to the amount of cryptocurrency locked in a stake account.

Ethereum was created initially as a PoW blockchain but it switched to a PoS consensus protocol in 2022 due to energy efficiency reasons, but also because it is a more secure and scalable protocol than PoW [7]. Another popular consensus protocol is *Proof-of-Authority (PoA)*, where nodes stake their reputation in the blockchain as leverage. Honest behaviour is rewarded and vice-versa. Higher reputations result in higher changes of publishing a new block.

3.3 Smart Contracts

The concept of smart contract was developed initially by [25] but as a reference to the automation of general-purpose legal contracts. In the blockchain context, this term refers to a script of code, witten in the specific programming language supported by the underlying blockchain, and whose execution runs synchronously on multiple nodes of the network [30]. The aggregate of computational resources of a network can be organized by a blockchain protocol running in each member, effectively creating a *Distributed Virtual Machine* and this is the main computation platform that executes transactions that can trigger smart contract functions to execute. Due to the lack of a central organizing entity, and to prevent malicious code that can damage this machine, smart contract computations require *gas*, a usually small but significant monetary value in the form of a cryptocurrency supported by the blockchain. The gas required by a smart contract execution is paid when the transaction is submitted, which means that every smart contract execution has a finite amount of gas to execute. This prevents malicious actors from running code that could exhaust the available resources of the virtual machine (by running a script with an infinite loop, for example). When the gas allocated to a transactions runs out, the transaction reverts, i.e., the blockchain recovers the state it had before attempting to run the transaction.

3.4 Non-Fungible Tokens

Non-Fungible Tokens (NFTs) work paradoxically to regular cryptocurrency tokens. Cryptocurrencies tie an address to a single variable representing the amount of that cryptocurrency the controller of that address owns. NFTs invert this relationship in the sense that they connect the token to an address instead. This clever aspect is part of what guarantees their digital uniqueness: an NFT can have one and only one owner at any point in time. NFTs essentially abstract a piece of data stored distributively in a blockchain and associate it to a single account address that represents the owner of that data. Ownership is ensured by the assumption that the private encryption key from which the account address was derived is under the control of the account owner and no one else.

The non-fungible characteristic of this tokens derives from their inability to be interchanged. With fungible tokens, as with in any cryptocurrencies, each token has a predefined value, can be traded by another one with the same denomination without any loss in value, and can be split and combined into and with fractions of other tokens. Non-fungible tokens cannot be split or combined with others, since each NFT is unique, and implies that there's no direct and fixed correspondence between any two or more NFTs. Fungible tokens are like bank notes: each note has a fixed value, can be exchanged with another with the same value or by a combination of others of smaller values that add to it without any loss in value. Non-Fungible Tokens are akin to paintings or trading cards. Each one is unique, indivisible and usually their value is highly

affected from speculation. In the public realm, most NFT applications thus far seem to emulate the analogies considered: they either represent a unique object, such as a painting or any other unique object, or they are used in a collectible setting, in which multiple NFTs are published within the context of a collection. Each token retains its uniqueness because, like physical trading cards, each has a unique identifier that differentiates it from others, even if the remaining metadata is identical to all the other tokens in that collection or series.

The NFTs ability to abstract ownership does not limit itself to digital objects. In fact, a curious argument can be made to use digital NFTs to represent physical, real "ownable objects", which includes everything from a car and a house, to less tangible goods such as ownership of land or intellectual property rights.

NFTs are implemented and can be created (minted) in a smart contract. Blockchains that support NFTs also implement a programming language that allows the creation and deployment of smart contracts that can be used to define and mint these tokens. The actual mechanics behind this process differs significantly per blockchain. This fundamental difference from the cryptographic tokens used as currency expands the scope of applications of blockchain considerably.

NFTs achieve *interoperability* through the implementation of standards. In the Ethereum network, these standards comprise of contract interfaces which define a set of requirements, namely parameters and function signatures, that a NFT token must implement. NFTs that conform to these standards indicate it so in the smart contract that governs them and the implementation of these standards guarantees to other uses a minimum of operability that ensure a secure interface with such token. In the Ethereum network, NFTs are governed by two *Ethereum Improvement Proposals (EIP)*, namely *EIP-721* and *EIP-1155*. These proposals are abstracted into the respective *Ethereum Request for Comments (ERC)* standards *ERC-721* [6] and *ERC-1155* [21], which in turn are written as smart contract interfaces that, when implemented in another contract, ensure that the any NFTs minted with that contract contain all the internal mappings required to establish ownership relationships, as well as functions required to transfer the token or to determine its internal parameters, for example.

3.4.1 Ethereum blockchain

Ethereum is one of the more popular public blockchains today. It was announced in 2013 through the publication of a whitepaper [5], followed by an *Initial Coin Offering (ICO)* in 2014, and culminating with the official launch of the new blockchain during 2015, through the release of *Frontier*, a early, bare-bone but functional version of the project. Ethereum was launched as a *Proof-of-Work* blockchain, similarly to the only existing public blockchain at that time, Bitcoin, and implemented *Ether (ETH)* as the cryptocurrency of the blockchain. The innovation of Ethereum was its smart contract support, which created a whole new logical and programmable layer on top of it. The community reacted very

positively and soon after the first Ethereum smart contract projects begun being deployed in it. ETH is used to pay the *gas* required for their executions, as well as rewarding with newly minted ETH tokens the miner node that solved the cryptographic puzzle for a given round. But like with any other cryptocurrencies with fiat value, ETH can be used as a "normal" currency. By 2022, Ethereum, still in its version 1.0, had grew considerably, and the drawbacks of using a PoW consensus protocol is such large network were becoming quite apparent and prejudicial. As such, this network switched to a PoS consensus protocol during its upgrade to version 2.0.

Ethereum was the first public blockchain to offer smart contract support though its *Ethereum Virtual Machine (EVM)*, and it was also the blockchain where the first NFTs projects were deployed. Ethereum uses Solidity to write smart contracts which in turn are used to define the rules that implement a NFT. In Ethereum NFTs are essentially defined as a combination of synchronised records kept by the implementing contract and that establish a unique relationship between a NFT, identified uniquely by id value, traditionally named *tokenId* in Solidity smart contracts (but not mandatory), and an account address that owns that token. NFT ownership is thus defined as the ability of a user to produce a correctly digitally signed transaction that can change the state of the Ethereum blockchain. This change is mostly restricted to transferring that NFT to a different address, which changes the internal mappings that establish the chain of NFT ownership, therefore triggering the blockchain to change its state. Early NFT projects limit NFT operations to only transfers, but more recent projects take advantage of *dynamic or mutable NFTs*, which are NFTs that allow the owner to change their metadata by exposing functions that allow it [11]. For an Ethereum NFT, all records that establish its ownership are centralised in the minting smart contract. Though the actual data is replicated through a series of active nodes, conceptually, all the relationships used for ownership purposes are stored in contract itself as mappings, i.e, key-value structures (known also as *dictionaries* in other programming languages). This means that there is a central point of failure in the Ethereum architecture. If the owner of the contract, or an adversary for that matter, is able to delete the contract, the ownership information is lost irreparably. This type of conceptual centralisation, allied with a fairly high block rate (15 seconds per block in average), make Ethereum difficult to scale, a fact that became evident with the sudden surge in popularity of the *CryptoKitties* project in 2017 [3].

3.4.2 Flow blockchain

The Flow blockchain was formally launched in October 2020 through the usual *Initial Coin Offering (ICO)* that helped launch most public blockchains in recent years. Flow followed a trend that was addressing the high energy consumption associated to earlier, PoW based blockchains, and established itself with the more efficient and energy friendly PoS consensus mechanism [15].

In a PoS based blockchain, new blocks are "mined" by active nodes based on their *stake* on the network, namely, the volume of the blockchain's native

cryptocurrency the node currently holds in its account. Higher stakes mean higher probability of being awarded a block publication by the protocol that regulates the network, with the token incentives that such entitles, and vice versa. This process is more energy efficient and faster than the "classical" PoW consensus, where nodes pointlessly spend computer cycles solving cryptographic puzzles only to win a race towards its completion. Given the significant energy waste due to the popularity of PoW blockchains, such as Bitcoin, a blockchain that stays clear of such nefarious protocol is not just a smart choice but a conscientious one as well [16].

Flow was created by the same team that created the *CryptoKitties* project in Ethereum, one of the first NFTs based blockchain games in this network to adopt the ERC721 standard for Non-Fungible Tokens. Even though Ethereum was the first one to offer NFT support, it did so on top of a chain that did not envision such applications at the time of its conception. This was evident by the lack of flexibility, inherent security flaws in the code implementing the tokens and its mechanics, as well as a general complexity in writing code and operating with these constructs in the Ethereum network. Ethereum was so ill prepared to deal with the popularity of this initiative that its network went down briefly due to its inability to cope with the added network traffic [3]. Flow was developed with NFTs support as its main feature, prioritizing NFT creation and mechanics over other applications, a clear contrast over blockchains focused in cryptocurrency transactions [10].

Flow establishes a new computational paradigm in this context, namely a *Resource Oriented Paradigm (ROP)*. It does so through Cadence, a smart contract programming language developed for Flow. Cadence establishes ROP through a special type of object, named *Resource*. From a technological point of view, a Resource is akin to a structure or an object from an Object-Oriented context, offering a significant degree of freedom on the type of data that it can encode. But Cadence regulates operations through the notion that every Resource is unique in Flow. This means that Resources need to be accounted at all times, and they cannot be copied, only moved. They can be stored in decentralized storage accounts and loaded from them but, at all times, the Resource is either in storage or out of storage, never in the same state at the same time. These Resources can be created only through smart contracts functions and are exclusive to the Flow blockchain, i.e., not interchangeable with other NFTs from other blockchains.

Flow implements a four-node type architecture that implements pipelining to achieve gains in speed, throughput and scalability while keeping minimal costs at a minimum sacrifice in network redundancy. Active nodes in Flow are divided into *Consensus, Collection, Execution* and *Verification* nodes that are organise in a pipeline to optimise computations. Fig. 3 provides an overview of the architecture of Flow.

- Consensus Nodes Decide the presence and order of transactions on the blockchain.
- Collection Nodes Enhance network connectivity and data availability for

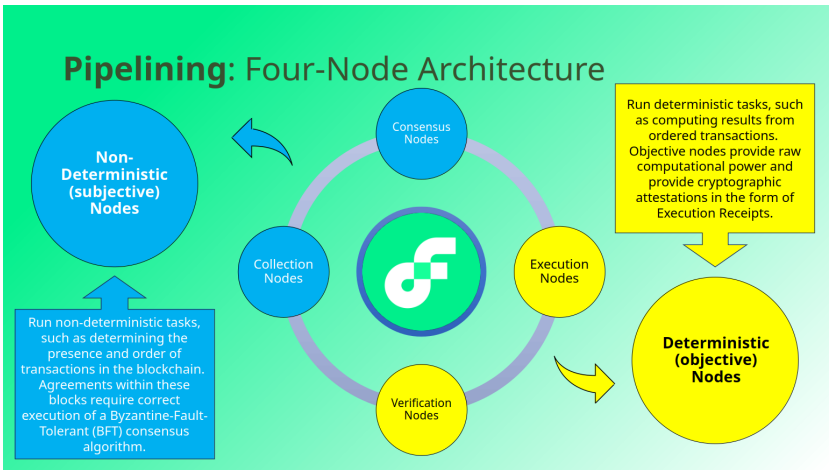

Figure 3: The four-node type architecture of Flow [17]

applications.

- Execution Nodes Perform the computation associated with each transaction.
- Verification Nodes Responsible for keeping the Execution nodes in check.

To validate computations, the development team behind Flow developed *Specialised Proofs of Confidential Knowledge (SPoCK)*, a type of non-interactive zero-knowledge proofs based on the *Boneh-Lynn-Shacham (BLS)* signature scheme. These proofs address the *Verifier's Dilemma*, a situation where rational miners are well incentivised to accept unvalidated blocks. Miner nodes in Flow get their block reward by providing a SPoCK that can only be obtained by correctly executing all the transactions that were assigned to it [4].

3.4.3 Storage in Ethereum vs. Flow

One of the most interesting features that distinguishes Flow from other blockchains, like Ethereum, is its account-based storage model, as opposed to the contract-based one used in Ethereum, for example. In Flow, each user account has its own storage space in the blockchain. Data stored in that space is protected by the same mechanics that prevent someone from transferring a NFT that he/she does not own, transferring cryptocurrencies from another account into his/her own etc. By default only the account controller can access and operate over its storage space. This type of storage decentralisation makes Flow more robust and potentially more scalable, since NFT transactions occur between user accounts and do not have a centralising element that needs to be changed with each transaction that affects a NFT.

Flow employs an *Account-Centric Storage Model* in which storage is the blockchain is based in an account address and the available storage space is proportional to the balance of FLOW token in that account. Due to this, Flow requires a minimum of 0.001 FLOW (around 0.0005$ at the time of this writing) in the account balance. If an account balance drops bellow this value, the account becomes limited to receive deposits and delete data until the minimum value is restored. Currently, 1 FLOW (0.5$) allows for 100 MB of on chain storage.

3.4.4 Token standards

Token standards, for both Fungible and Non-Fungible Token, were created to establish interoperability standards so that different projects in the same network can interact among themselves, namely, cryptocurrencies and NFTs can be traded with minimal effort if users "expect" those tokens to have certain properties and functions. This enforcement is accomplished through token standards, which functionally are contract interfaces. These work very similarly to normal programming interfaces from general purpose languages, like Java for example [20], that define internal parameters and function signatures that are "forced" to be implemented if a class declares the implementation of such interface. If an NFT contract implements the ERC-721 token standard, for example, a user interacting with tokens minted by this contract can access the public internal parameter *tokenId* without having to check the actual contract to see if this parameter was implemented or not. The ERC-721 compliance of the NFT contract guarantees this, along with the remaining parameters and functions.

Ethereum Token Standards Ethereum was the first blockchain to welcome and popularize both the NFT concept, but cryptocurrencies in general through the definition of standards. Ethereum projects that followed these token standards created a interoperable ecosystem, where token, but fungible and otherwise, could be traded freely among different contracts within the Ethereum network.

- ERC-20 Fungible Token Standard The ERC-20 standard regulates the fungible tokens architecture and behaviour in the Ethereum blockchain. These tokens are mostly used as cryptocurrencies. Blockchain projects advertise their ERC-20 compliance as a statement of compatibility with other projects in the Ethereum ecosystem. The standard itself consist in a `.sol` contract interface file written in Solidity [28].

- ERC-721 Non-Fungible Token Standard ERC-721 establishes a similar set of rule for smart contracts implementing NFTs. Implementing this interface in a contract forces it to inherit a series of key-value mappings that are used internally to implement the NFT itself, both in terms of ownership record and token metadata [6].

- ERC-1155 Multi Token Standard The non-fungible standard in Ethereum was later amended with the ERC-1155 standard to extend the ecosystem to include "semi-fungible tokens". Regular, ERC-721 tokens associate one id to one token. ERC-1155 allows for an id to be associated to a finite set of tokens. Each token with the same id is, effectively, fungible within the same set, a behaviour emulating trading card game, for example.

Flow Token Standards Flow was created with a different architecture from Ethereum and Cadence is quite different than Solidity, though both achieve similar results. But the similarities in operation with the Ethereum blockchain require that Flow established a similar set of standard for its tokens.

- Fungible Token Standard Flow's version of the ERC-20 standard is simply named *Fungible Token Standard* [26], since it was created with the express purpose of regulating fungible tokens in the network, and not as a result for a *request for comments* action. It operates very similarly to the ERC-20 counterpart: the standard itself is an inheritable contract interface in the form of a `.cdc` file [13], the extension used by the Cadence language, defining a set of parameters and functions that require implementation.

- Non-Fungible Token Standard For NFTs, Flow implements a single standard, named simply *Non-Fungible Token* [27], but it does encompass both functionalities from the ERC-721 and ERC-1155 Ethereum standards. It is, in itself, a `.cdc` [14] contract interface file that is imported from implementing contracts. NFTs have a different behaviour in Flow, mainly because of the storage architecture implemented by this blockchain. Flow NFTs are digital object stored in a specific storage path while Ethereum NFTs are, essentially, a series of synchronized key-value mappings that establish a unique relation between an address an token id. Because of this storage difference, Flow NFTs are usually stored inside collections, which are a different, non-unique, type of resource that has the capacity of "holding" multiple NFTs from the same type, i.e., minted from the same contract. This is used primarily to simplify the handling of storage paths, but it essentially mimics the behaviour from the ERC-1155 without an additional standard required.

4 Non-Fungible Tokens

NFTs here!

5 Solidity-Ethereum NFT Implementation Details

5.1 General Architecture of a Solidity NFT Smart Contract

Constructing a ERC721 compatible smart contract in Solidity begins with importing the contract interface standards, specifically the `.sol` file implementing the contract interface. These can be imported from a local path, but in recent years, informal online repositories such as *OpenZeppelin* [19] simplify this process greatly by making these standard contract interfaces available for installation as if they were normal npm modules. As such, "installing" these contracts is as easy as installing an npm module. The `.sol` files are copied to an internal *node_modules* folder and, from there, are imported into the contract, so in a way they are still being imported locally.

We begun this project by implemented a ERC-721 compliant NFT minting contract, restricting it to the minimal functionalities required by the standards imported. The idea is to restrict its functionalities to the most basic ones since these are also reflected in other architectures, specifically in the Flow blockchain. which would allow us to to an objective comparison between smart contracts written in different languages and deployed in architecturally different blockchain but offering similar capabilities. Though architecturally, Flow and Ethereum are quite different blockchains, both NFT implementations using the official standards create common elements to both implementations, such as token unique identifiers and token balances as internal parameters, *balanceOf* and *ownerOf* functions, and *Transfer* events.

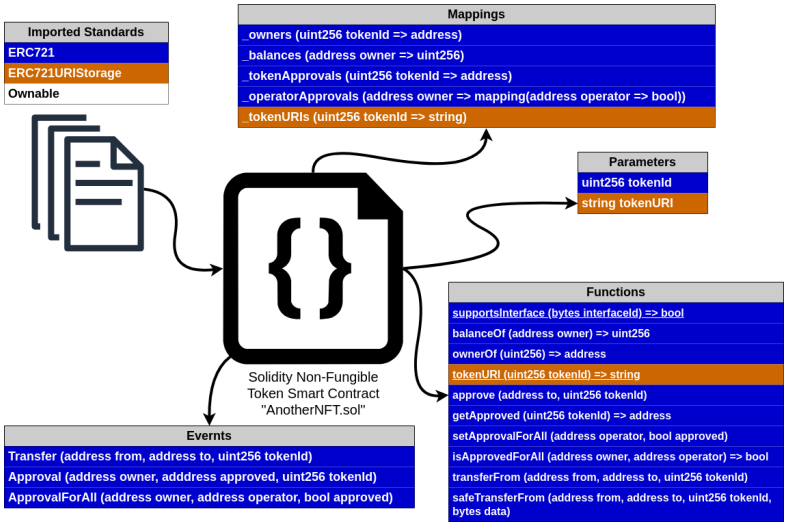

Figure 4: General organization of a Solidity NFT minting contract

Fig. 4 references the general architecture followed in the implementation of the example NFT contract. We use a color scheme to identify the parameters, functions, event and mappings (exclusive to Solidity contracts) that are inherited from the standards. In Solidity, importing these interfaces in another contract makes the structures and functions indicated in Fig. 4 available without having to explicitly declare them in the contract. Section 5.1.1 goes into more detail about this.

For this purpose, we imported three Solidity interfaces:

1. **ERC721** This is the most important standard to implement an Ethereum NFT and that is visible in the amount of functions, parameters, mappings and events inherited from this standard alone.

2. **ERC721URIStorage** This contract interface is used to establish a more "standardised" way to deal with NFT metadata. It does so by establishing a mapping specific to store that metadata (*_tokenURIs* mapping), as well as function to retrieve the NFTs metadata.

3. **Ownable** This interface is used to simplify the establishment and verification of ownership for the contract itself only, hence why no other structures or functions get inherited from this standard. The ownership mechanics for any NFT minted are processed by functions and mappings inherited from the ERC-721 standard. It is possible to write the contract without it, but it makes it unnecessarily complicated.

5.1.1 Interoperability derived from Standards

If a user interacts with an ERC-721 compliant Solidity NFT contract, the user can invoke the *balanceOf* function without any need for him/her to check the contract code first to ensure that the function is implemented. In fact, the majority of the cases, this function and others are simply omitted because the interface does not has them marked as **override**, which means that the implementing contract does not need to have them explicit in its code to have them available for usage. Others, such as `supportsInterface` from the ERC721 standard and `tokenURIs` from the ERC721URIStorage one are marked for overwrite, and as such they need to be explicit in the implementing contract. Overwritten functions are underlined in Fig. 4. Interface functions marked as **virtual** can be overwritten.

5.1.2 Mapping-based Ownership System

Token ownership in the Ethereum network is established by the `\_owners` mapping. When a NFT is minted, a new entry is added to this mapping creating a one-to-one relationship between an account address and a unique `tokenId`. The `tokenId` is unique within the ecosystem defined by the NFT contract. In other words, the NFT is defined by the pair contractName-tokenId.

The NFT metadata is defined through the `_tokenURIs` mapping, but this one is the one inherited by default from the ERC721Storage standard. This

standard is not mandatory to implement a NFT in the Ethereum network. Realistically, NFTs can be created without any metadata set, though the usefulness of such construct is limited. Also, a mappings of this type can be added to the NFT contract without importing the ERC721URIStorage, and they can be used to store any kind of string, just like the `_tokenURIs` mapping. But using standards simplifies development and introduces a degree of certainty to other users familiar with them.

The `balances` mapping is self explanatory and the mappings suffixed with "Approvals" are used to delegate token access to other users other than the user. If the user wishes, he/she can use the "approval" functions to set permissions to allow other users to transfer the NFT and execute other owner-exclusive functions. Approval functions change the state of the blockchain, therefore require signed transactions to be executed. Each approval function is owner protected, i.e., they all begun by testing if the address requesting changes to the approval mappings is the owner of the token referred. If that is not the case, the transaction reverts.

5.1.3 Internal Parameters

In its most basic form, a standard Ethereum token, i.e., ERC721 compliant token, is defined solely by a `tokenId`, an unsigned 256-bit integer. As mentioned in Sec. 5.1.2, NFT metadata is not strictly required and a NFT contract can be implemented without one. The ERC721 standard guarantees solely the existence of the unique token id and, therefore, the uniqueness of the token in the network.

5.1.4 Events

Blockchain events are the main information mechanism to detect transaction results and contract changes in the network. Obtaining an output from functions than do not return any values, such as nft mints and token transfers, from any blockchain is quite difficult because one has to look for the correct output from the agglomerate of all the outputs of the various distributed accesses that the blockchain is attending. For a popular and sizeable blockchain, such as Ethereum, that task becomes prohibitive. Events solve this limitation. Event logging is processed differently and these can be indexed, which allows for their fast retrieval and for software tool to pool for these at intervals (event listeners). The ERC721 defines three events by default, one to be emitted whenever a token is transferred(*Transfer*) and two for whenever the approval mappings are modified. These events require arguments to be emitted, which allows for the transmission of specific information with more generalist events. For example, the *Transfer* event, when captured, also indicated the two addresses involved in the transfer and the `tokenId` of the token transferred. The emission of these events is triggered by ERC721 functions, so the whole process is automated and inherited from the contract interface.

5.2 Functions

Functions such *balanceOf* and *ownerOf* are self-explanatory and the approval related functions were covered in Sec. 5.1.2. The only function inherited from the ERC721URIStorage standard, *tokenURIs* is also quite simple: it receives a `tokenId` as argument and returns the string defined as metadata for the token identified. The *supportsInterface* is used to determine if the contract in question does conform to the ERC721 standard (or any other standards indicated by the contract), and it is not just something the developer is labeling the contract with, but the expected contract interface was not correctly implemented.

Finally, the last functions indicated in Fig. 4 are used to transfer the token to a `to` address. Both functions do what their name suggests, but the "safe" version does additional checks, namely if the receiving address (parameter `to`) is a contract address and, if so, test if the correct NFT receipt was returned from the contract.

6 Cadence-Flow NFT Implementation Details

ADD NEW SECTION 05 HERE

7 Results and Comparison

8 Conclusion

ADD CONCLUSION HERE

References

[1] Omar Ali et al. "A Review of the Key Challenges of Non-Fungible Tokens". In: *Technological Forecasting and Social Change* 187 (2023), pp. 1–13. ISSN: 00401625. DOI: 10.1016/j.techfore.2022.122248.

[2] Hong Bao and David Roubaud. "Non-Fungible Tokens: A Systematic Review and Research Agenda". In: *Journal of Risk and Financial Management* 15 (5 May 8, 2022), pp. 1–9. ISSN: 1911-8074. DOI: 10.3390/jrfm15050215.

[3] bbc.com. *CryptoKitties craze slows down transactions on Ethereum*. URL: https://www.bbc.com/news/technology-42237162 (visited on 06/22/2022).

[4] Tarak Ben, Youssef Riad, and S Wahby. *Flow: Specialized Proof of Confidential Knowledge (SPoCK)*. 2020. URL: https://eprint.iacr.org/2023/082.

[5] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. 2014.

[6] William Entriken et al. *ERC-721: Non-Fungible Token Standard*. URL: https://eips.ethereum.org/EIPS/eip-721 (visited on 11/29/2023).

[7] Ethereum.org. *Proof-of-Stake (PoS)*. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ (visited on 10/24/2024).

[8] Jex Exmundo. *Quantum, The Story Behind the World's First NFT*. Mar. 2023. URL: https://nftnow.com/art/quantum-the-first-piece-of-nft-art-ever-created/.

[9] Saeed Banaeian Far et al. "A Review of Non-fungible Tokens Applications in the Real-world and Metaverse". In: *Proceedings of the 9th International Conference on Information Technology and Quantitative Management*. Vol. 214. Elsevier B.V., 2022, pp. 755–762. DOI: 10.1016/j.procs.2022.11.238.

[10] Roham Gharegozlou. *Introducing Flow, a new blockchain from the creators of CryptoKitties*. URL: https://medium.com/dapperlabs/introducing-flow-a-new-blockchain-from-the-creators-of-cryptokitties-d291282732f5 (visited on 2022-06-20).

[11] Barbara Guidi and Andrea Michienzi. "From NFT 1.0 to NFT 2.0: A Review of the Evolution of Non-Fungible Tokens". In: *Future Internet* 15 (6 June 2023). ISSN: 19995903. DOI: 10.3390fi15060189.

[12] Badis Hammi, Sherali Zeadally, and Alfredo J. Perez. "Non-Fungible Tokens: A Review". In: *IEEE Internet of Things Magazine* 6 (1 Mar. 2023), pp. 46–50. ISSN: 2576-3180. DOI: 10.1109/iotm.001.2200244.

[13] Joshua Hannan et al. *Flow Fungible Token standard Interface contract*. URL: https://github.com/onflow/flow-ft/blob/master/contracts/FungibleToken.cdc (visited on 10/28/2024).

[14] Joshua Hannan et al. *The Flow Non-Fungible Token standard Interface contract*. URL: https://github.com/onflow/flow-nft/blob/master/contracts/NonFungibleToken.cdc (visited on 10/28/2024).

[15] Alexander Hentschel, Dieter Shirley, and Layne Lafrance. *Flow: Separating Consensus and Compute*. URL: http://arxiv.org/abs/1909.05821 (visited on 06/22/2022).

[16] Alexander Hentschel et al. *Flow: Separating Consensus and Compute - Block Formation and Execution*. URL: http://arxiv.org/abs/2002.07403 (visited on 06/22/2022).

[17] Alexander Hentschel et al. *Flow: Separating Consensus and Compute - Execution Verification*. URL: http://arxiv.org/abs/1909.05832 (visited on 06/23/2022).

[18] Dapper Labs. *CrytoKitties: Collectible and Breedable Cats Empowered by Blockchain Technology*. Tech. rep. Dapper Labs, 2017, pp. 1–9.

[19] openzeppelin.com. *Build Secure Smart Contracts in Solidity*. URL: https://www.openzeppelin.com/solidity-contracts (visited on 10/29/2024).

[20] Oracle. *The Java Tutorials: What is an Interface?* URL: https://docs.oracle.com/javase/tutorial/java/concepts/interface.html (visited on 10/28/2024).

[21] Witek Radomski et al. *ERC-1155 Multi-Token Standard*. URL: https://eips.ethereum.org/EIPS/eip-1155 (visited on 11/29/2023).

[22] Qaiser Razi et al. "Non-Fungible Tokens (NFTs) - Survey of Current Applications, Evolution, and Future Directions". In: *IEEE Open Journal of the Communications Society* 5 (2024), pp. 2765–2791. ISSN: 2644125X. DOI: 10.1109/OJCOMS.2023.3343926.

[23] Wajiha Rehman et al. "NFTS: Applications and challenges". In: *2021 22nd International Arab Conference on Information Technology, ACIT 2021*. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN: 9781665419956. DOI: 10.1109/ACIT53391.2021.9677260.

[24] NFTnow.com staff. *A Guide to CryptoPunks NFTs: Pricing, How to Buy, and More*. Feb. 2024. URL: https://nftnow.com/guides/cryptopunks-guide/.

[25] Nick Szabo. *Formalizing and Securing Relationships on Public Networks*. 1997. URL: https://firstmonday.org/ojs/index.php/fm/article/download/548/469.

[26] Onflow Developer Team. *Flow Fungible Standard*. URL: https://github.com/onflow/flow-FT (visited on 10/29/2024).

[27] Onflow Developer Team. *Flow Non-Fungible Standard*. URL: https://github.com/onflow/flow-nft (visited on 10/29/2024).

[28] Fabian Vogelsteller and Vitalik Buterin. *ERC-20: Token Standard*. URL: https://eips.ethereum.org/EIPS/eip-20 (visited on 10/29/2024).

[29] Qin Wang et al. "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges". In: *arXiv* (Oct. 24, 2021). URL: http://arxiv.org/abs/2105.07447.

[30] Weiqin Zou et al. "Smart Contract Development: Challenges and Opportunities". In: *IEEE Transactions on Software Engineering* 47 (10 2021), pp. 2084–2106. ISSN: 19393520. DOI: 10.1109/TSE.2019.2942301.

Todo list