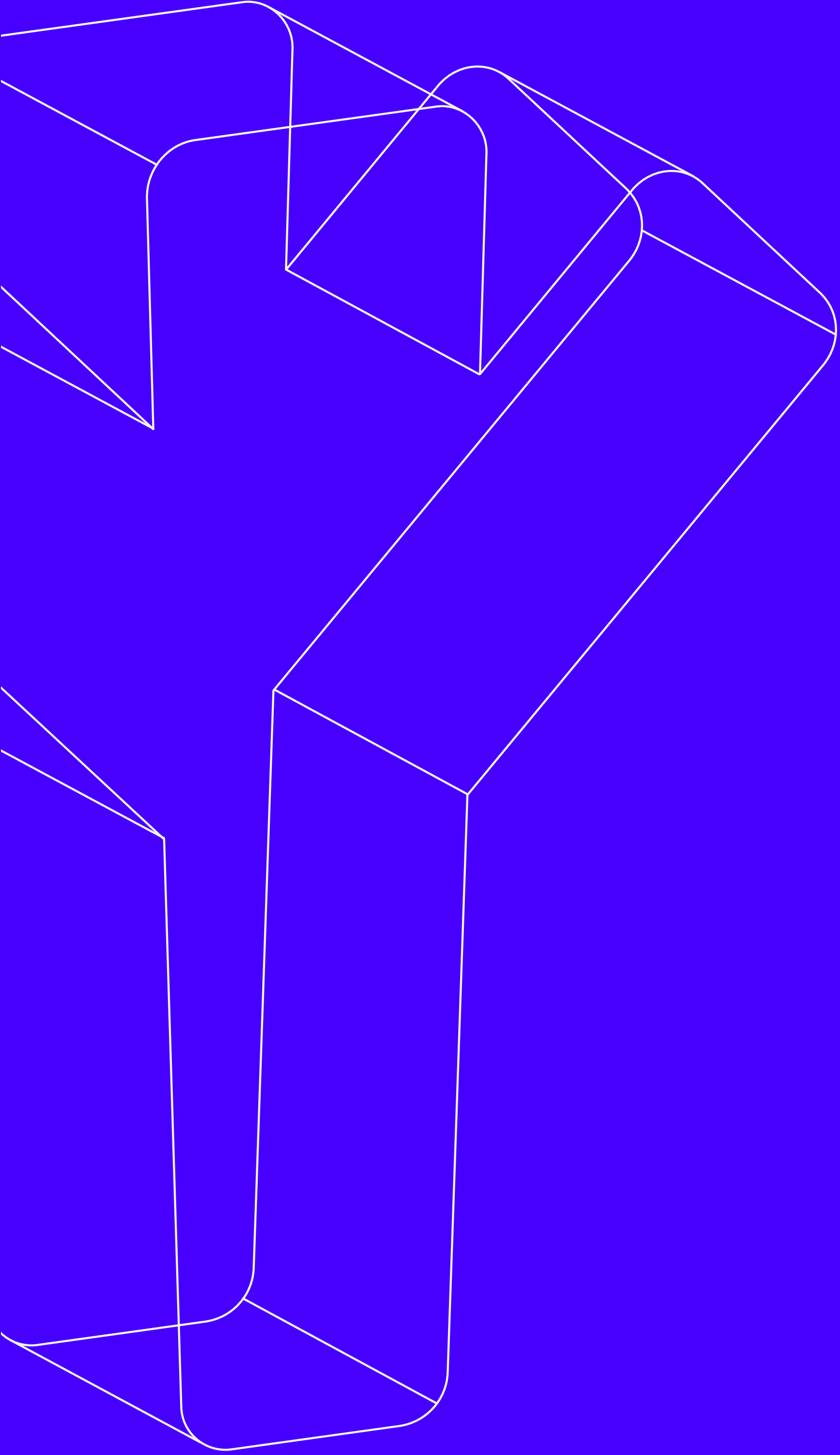


WWW.Naduceus.FOUNDATION

Naduceus

WHITE PAPER

The world’s first blockchain protocol

powered by decentralised edge rendering

June 2024

1. Abstract

Since blockchain came into existence, several distributed ledger technologies have been developed for various purposes. Starting with the basic architecture, several modified versions such as Ethereum Virtual Machine (EVM), distributed hash table (DHT), Time To Live (TTL), decentralised applications (dApps) etc have entered the blockchain ecosystem. However, as with every new technology, blockchain comes with unique challenges and technical issues. Mining is a widespread challenge limiting the usage of blockchain technology and posing a problem in the way of decentralisation because it heavily consumes both time and energy.

In this paper, we introduce an improved approach that solves multiple issues and eliminates the need for mining while maintaining all the benefits of blockchain technology, and also makes the development and usage of blockchain easier, more convenient and more effective, especially for AI, XR, and Metaverse applications.
That's what NaduceusProtocol is all about.

2. Introduction

Naduceu sevolves the open source software paradigm by implementing blockchain alongside its rich library of components - this facilitates the building of highly performant infrastructure that is trustworthy, efficient, secure and dependable. Naduceu s provides numerous essential infrastructures for blockchain and AI adoption.

- Naduceusprovides fund security, NFT asset security, user 5 privacy security, trusted transactions, automatic settlement, token ecological incentives, and DAO user autonomy through a high-speed, scalable blockchain network.

- Naduceusprovides decentralised cloud computing support for applications through a massive distributed IPFS storage/GPU cloud rendering/SWAN custom network.
- Naduceus provides the metaverse with a blossoming device ecology through the advanced XR Glass open-source hardware solution, lowering the hardware threshold and increasing device penetration.
- Naduceus connects blockchain seamlessly through SDK, providing interfaces and tools for NFT art creators and application developers.
- Naduceus enhances user experience and lowers user threshold through a 3D OSUI system, providing a user-friendly interaction portal.

3. Overall Structure

3.1 Logical Structure

The Naduceus Protocol ecosystem main elements are:

- **Management Platform:** a management platform for blockchain, including chain management, block information retrieval, visual monitoring and other functions.
- **Smart Contract Development IDE:** an online development environment for smart contracts. All contract languages supported by the NaduceusProtocol can be developed, compiled, and debugged on this IDE.
- **Command Line Tools:** Naduceus Protocol provides a wide range of tools to facilitate user command line deployment and management operations on the chain, such as certificate generation, chain configuration, rapid deployment.

- **SDK:** helps users communicate with the chain through RPC and complete functions such as contract creation, invocation and chain management.
- **Gateway Node:** also known as the Protocol Converter, used to handle the communication protocol between external services and the blockchain, data format or language differences.
- **Verification Node:** performs transaction preprocessing, verifies the signature and legitimacy of the transaction, and converts the transaction into a read-write set.
- **Consensus Node:** participates in consensus voting, transaction execution, event verification and accounting.
- **Sync Node:** also called Witness Node, the job of this node is to synchronise and verify blocks, execute transactions, and record complete ledger data; it does not participate in consensus voting.
- **Light Node:** synchronises data from Consensus Nodes, verifies data validation, filters and stores transactions belonging to the same organisation. It does not have the function of receiving transaction requests or broadcasting transactions.
- **Storage Node:** Distributed file storage node. For some special business needs, each file will be cut into 4K file fragments, and then stored in different nodes.
- **Storage Service:** NaduceusProtocol supports the storage of large files such as videos, audio, pictures in the blockchain network.

Normally when building a chain, the number of consensus nodes to be deployed should fulfil the consensus algorithm requirements, as for whether to add more consensus nodes or synchronisation nodes, this should be decided according to specific business needs.

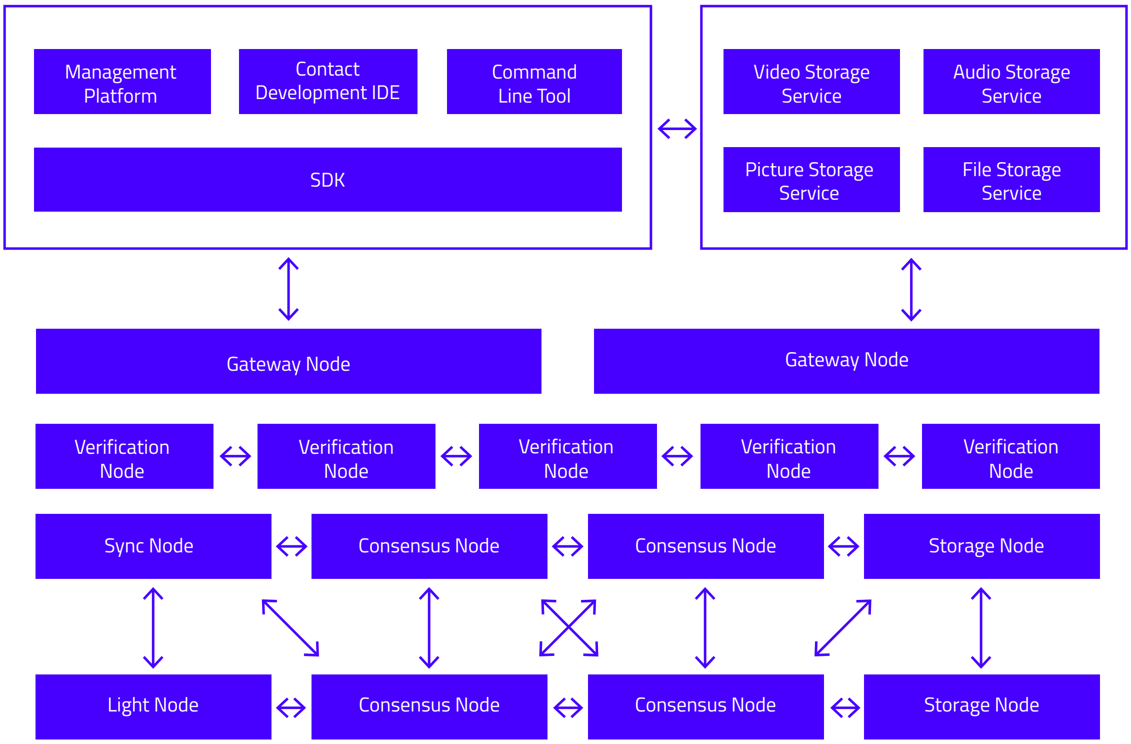


Figure 1: Logical Structure

3.2 Core Process

The event processing flow is as follows:

1. **Receiving Transaction:** The source of transactions in the transaction pool is either:
   a. Clients (submitted by a client), or
   b. Other node(s) (broadcasted by another node). Transactions submitted by clients need to be simulated by the verification nodes.

   Then all transactions (from both client and other nodes) and result sets get stored in the transaction pool.

2. **Proposing Event:** The proposing node then uses the core engine to select a batch of transactions, asynchronously verifies the transactions and result sets in batches, generates an event, then sends the event to the consensus module.

3. **Broadcasting Event:** The consensus module then broadcasts the proposed event through the P2P protocol to other consensus nodes.
4. **Receiving Event:** Every node that receives the broadcast event then executes the witness algorithm on the event; every witness in Round R + 2 will collect votes from Round R + 1; witnesses in Round R + 2 that strongly see witnesses (Events) of Round R + 1 can also strongly see events that are witnesses of Round R + 1. If the number reaches the requirement of an absolute majority, the event is confirmed immediately, reaching the consensus of the entire network and cannot be changed.
5. **Forming/Shaping Block:** Based on the transactions information in the event, a block will be formed.
6. **Performing transactions in batches:** Based on the transactions in the block, verify the transactions and result sets in batches asynchronously, and analyze the transactions Directed Acyclic Graph (DAG); asynchronously write read-write sets into the global database.

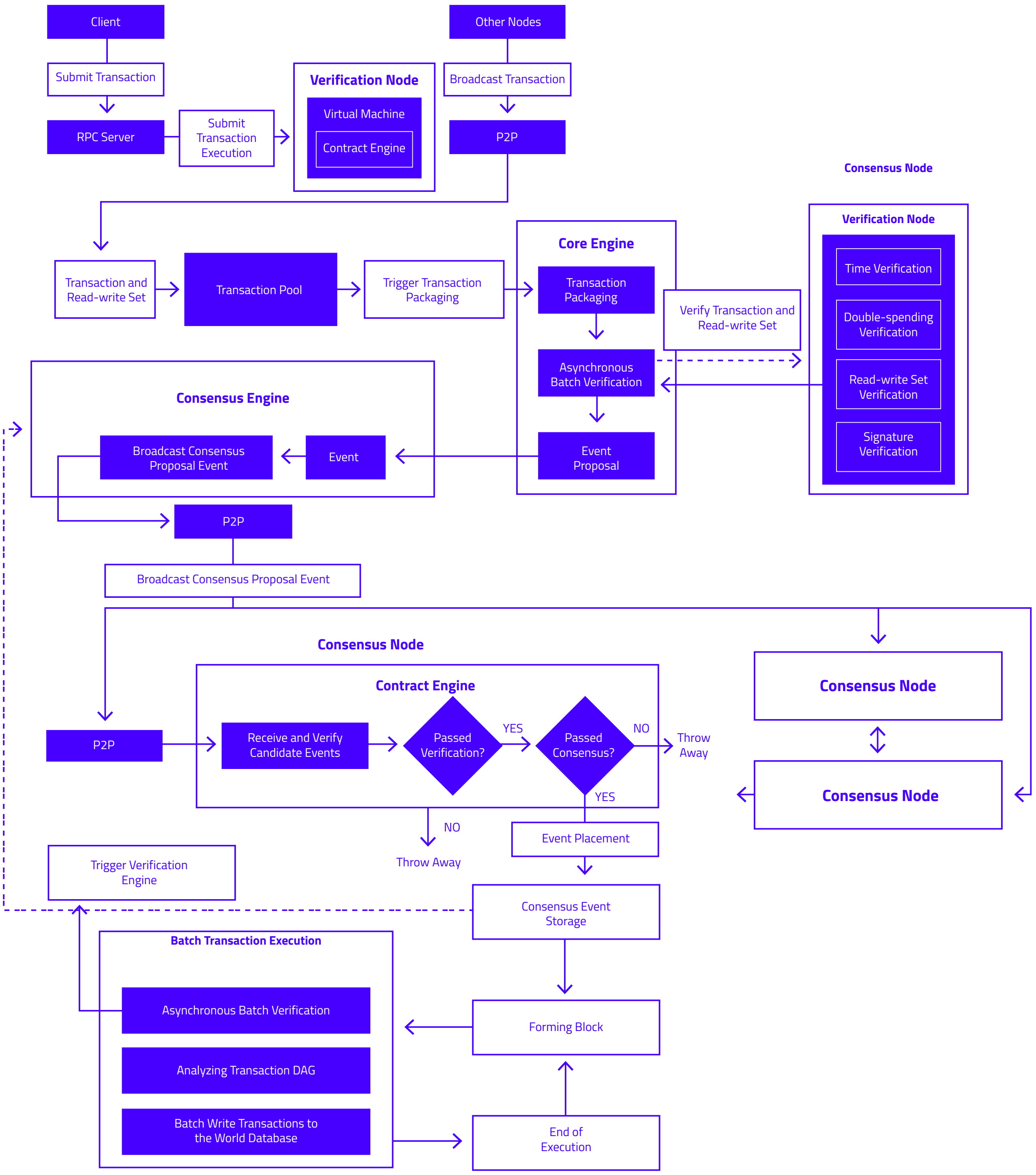

Client
Submit Transaction
RPC Server
Submit Transaction Execution
Verification Node
Virtual Machine
Contract Engine
Other Nodes
Broadcast Transaction
P2P
Consensus Node
Transaction and Read-write Set
Transaction Pool
Trigger Transaction Packaging
Core Engine
Transaction Packaging
Verify Transaction and Read-write Set
Asynchronous Batch Verification
Event Proposal
Verification Node
Time Verification
Double-spending Verification
Read-write Set Verification
Signature Verification
Consensus Engine
Broadcast Consensus Proposal Event
Event
P2P
Broadcast Consensus Proposal Event
Consensus Node
Contract Engine
P2P
Receive and Verify Candidate Events
Passed Verification?
YES
Passed Consensus?
NO
Throw Away
YES
NO
Throw Away
Consensus Node
Consensus Node
Event Placement
Trigger Verification Engine
Consensus Event Storage
Batch Transaction Execution
Asynchronous Batch Verification
Analyzing Transaction DAG
Batch Write Transactions to the World Database
Forming Block
End of Execution

Figure 2: Core Process

4.Core Features

4.1 Autonomous and Controllable Underlying Platform

Original blockchain underlying technology architecture with deep modularity, assemblability and high-performance parallel execution.

4.2 Flexible and Efficient Assembly Mode

Deep modularity. Different modules and components can be selected according to the user's needs to quickly assemble a customised blockchain system. Pluggable, detachable and autonomous core framework that has the ability to quickly access the underlying modules/single customised development modules.

4.3 International Advanced Processing Performance

Transaction processing is parallelized to the fullest extent, and the peak transaction processing speed of a single chain can reach over 100,000 transactions per second.

Memorybased data systems are supported, improving processing performance for transactions.

4.4 Standardised Open Ecosystem

- Adopting user-friendly open-source agreements and open software source code.
- Promoting the standardisation of multiple technical systems, establishing a standardised development ecosystem.

4.5 Complete Integrated Set of Convenient Support Tools

- Support for large screens, charts, various forms of interface interaction management, monitoring, operations and maintenance.
- Support for Java, Golang and JavaScript blockchain SDKs.
- Support for various implementation methods such as customized deployment and BaaS.

- Friendly and convenient online smart contract development environment.
- Supports rich set of mechanisms for processing blocks, transactions, subscriptions and event monitoring

4.6 Abstract and Unified Execution Flow

The overall processing of every blockchain implementation varies greatly. In order to assemble all kinds of blockchains' protocol requirements, NaduceusProtocol needs to reasonably abstract the overall execution process of the blockchain and combine modules based on this general process. The NaduceusProtocol will also consider increasing the flexibility of the overall process to support more abundant blockchain settings.

4.7 Deep modularity

NaduceusProtocol not only requires complete independence of blockchain module functions, clear interface definitions, pluggable replacements, but also complete virtualization of communication between modules, which can support calls from functions, inter-process communication (IPC) to various network communication protocols and other different implementation modes. This makes it possible to assemble and combine convenient and free modules.

4.8 Comprehensive Supporting Protocol and Assets

NaduceusProtocol fully supports the asset agreement of the existin g public chain, such as:

- **ERC20:** (A Fungible Token or FT) The unit value of all tokens is the same and can be divided.
- **ERC721:** (A Non Fungible Token or NFT) Each token is different and has its own unique properties and value. Of course, this also means that they are both

indivisible and traceable. NaduceusProtocol Supports the creation of new native NFTs or imports existing NFTs from external chains. Governance enables platform stakeholders to vote on the direction of the chain and the infrastructure. NaduceusProtocol will be bound by a governance plan specifically designed to recognize its nuances.

- **ERC1155:** Uses a new method to define tokens. Items are stored in a central smart contract, which takes up very little space and is only used to distinguish each other. It supports batch transmission of multiple token IDs in a single transaction. Any token can be merged into a single 'Token Package', which also has its own independent ID.
- **ERC998:** (A Composable NFT, or CNFT) It is an extension to the ERC-721 standard that adds the ability for non-fungible tokens to own other non-fungible tokens and ERC-20 tokens. Non-fungible tokens that implement ERC998 also implement the ERC-721 standard.

4.9 The Core Application:

4.9.1 Decentralised Storage

NaduceusProtocol is committed to developing new underlying technologies to help provide solutions for data storage (Distributed Storage). This solves issues raised by existing centralised storage. With distributed storage systems, there is no need to reply to a large number of centralised data islands; nor do these data islands undermine important values such as privacy and freedom of information.

The application is a decentralised storage network based on an InterPlanetary File System (IPFS). It is the only incentive layer on IPFS. A token is issued, based on blockchain technology. Miners can obtain tokens by providing storage and retrieval services for customers, and customers can compensate miners by spending tokens on the storage or distribution of their data.

The IPFS is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses Content-Addressing to uniquely identify each file in a global namespace connecting all computing devices. Content-based addressing allows customers to access specific data based on its unique fingerprint. No matter where this data is stored, if the customer has a unique fingerprint of it, they can get its content. In the content-based addressing mode (within the concept of IPFS), content is no longer obtained through a single address on the Internet. Instead, it can be obtained from any specific IPFS network node that stores the content requested by the customer. Fragments of the content are shared by many participants and can always be obtained from a single node as a whole – such as through the pinning service node, or collected and compiled from multiple nodes.

NaduceusProtocol implements a new type of storage proof. Storage 13 miners need to prove to the verifier that they store the corresponding data on a specific device, instead of storing multiple copies of data on one device, effectively preventing Sybil attacks, Outsourcing attacks and Generation attacks. With the combination of timestamp technique, proof that the miner stored data for a given period of time is obtained. Even if, at some point in the future, the user is not online, the time and space proof can be used to verify the data stored by the miners during that period.

The fastest growing content category in the crypto world is that of NFTs. However, NFTs have begun to be used as a research object for usability and sustainability issues, all of which can ultimately be attributed to the field of content addressing and sustainability. When we discuss the minting and trading of NFTs, we actually mean the changes in the records behind the artwork (or other asset). The content and metadata of this artwork (colour, shape, sound, etc) does not automatically exist on the blockchain. 'Content' refers to the picture itself. 'Metadata' refers to the CID identification number that describes the text/artist information/real content, etc. If the content and metadata of the NFT is not reliably stored, it will expose many problems in the addressing and sustainability of the

NFT. The decentralised storage of NaduceusProtocol can solve the addressing problem of NFTs, and the incentive mechanism of Naduceus Protocol encourages global storage space providers to continue to store NFT content and related metadata for extended periods.

4.9.2 Analysis of Naduceus Decentralized Edge Rendering Technology

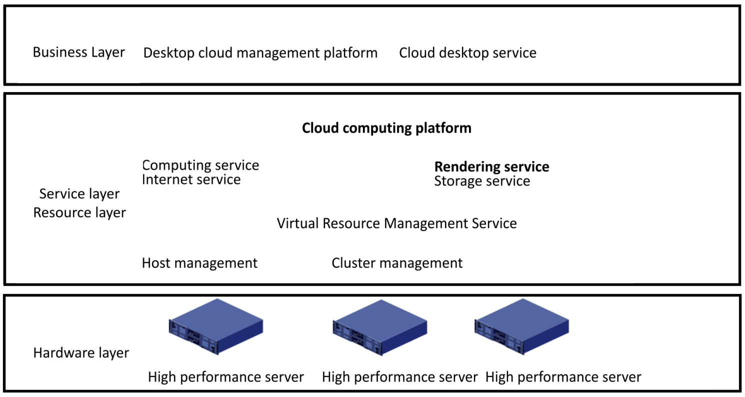
Naduceus decentralized Edge Rendering is a remote service that provides computational, rendering, application, and display capabilities for AI, XR, Metaverse and Web3 in the decentralized cloud.

Decentralized Edge Rendering technology is still an emerging sector and rapidly growing, most widely used in the fields of gaming, teaching, medicine, and defense. There is an ongoing trend to replace traditional rendering technology with Decentralized Edge Rendering because of its multiple advantages, including fast deployment, easy management, high maintenance efficiency, safety and reliability, and lower energy consumption than the traditional rendering technologies.

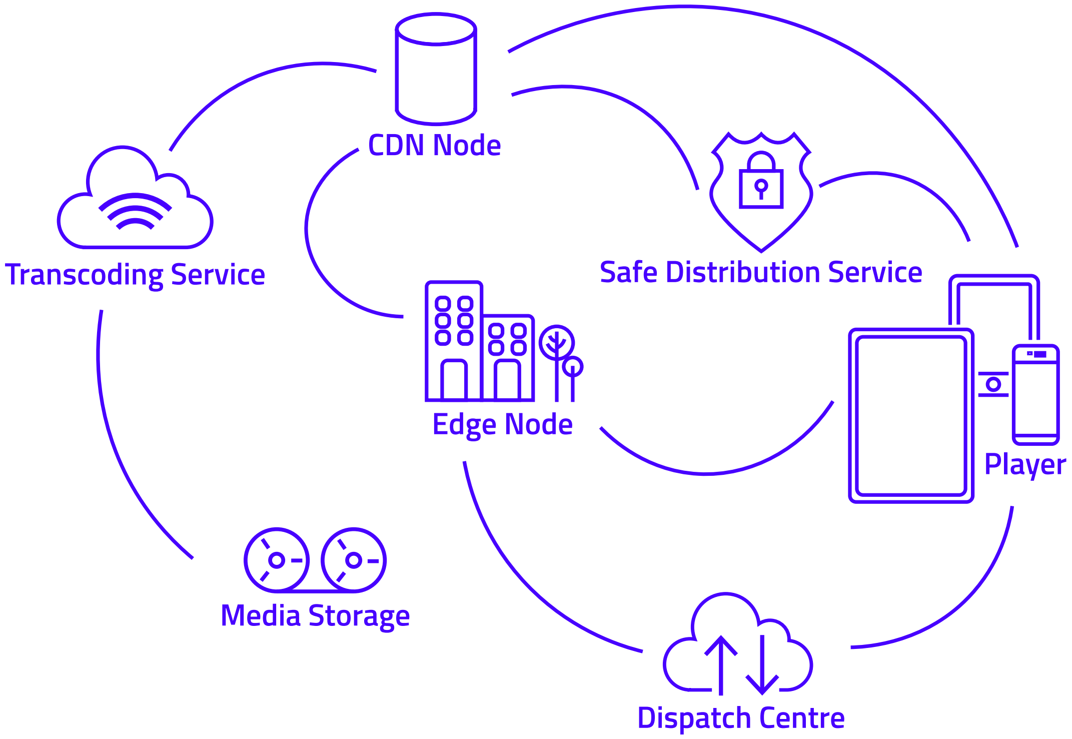
Decentralization itself is a structure that occurs only in systems with many users or nodes dedicated to similar tasks. Technically speaking, every user or node is at the center, and everyone can connect and influence other nodes within the group. This technology system or structure of flattening, open-source, and equalization is called decentralization. It allows for the formation and relationship of content and data processing, and brings with it a new type of network content production innovatively and complimentary to traditional centralization.

Naduceus combines the core technologies of cloud computing, cloud rendering, and cloud storage, with decentralization technology that tackles high-performance demands. Naduceus' cloud edge rendering transmits video back to users and leverages streaming techniques that create new visual experiences and business opportunities from gaming through to video editing.

The edge cloud servers bring unparalleled scaled computing and image processing, which allocate and optimize application resources through virtualization technology. They also provide the application programs and rendering results required by users in the form of PaaS and SaaS. The PaaS provides the cloud function of customer applications, which can offer online rendering and experience services for more users, while the SaaS can render the output online for the final customer.



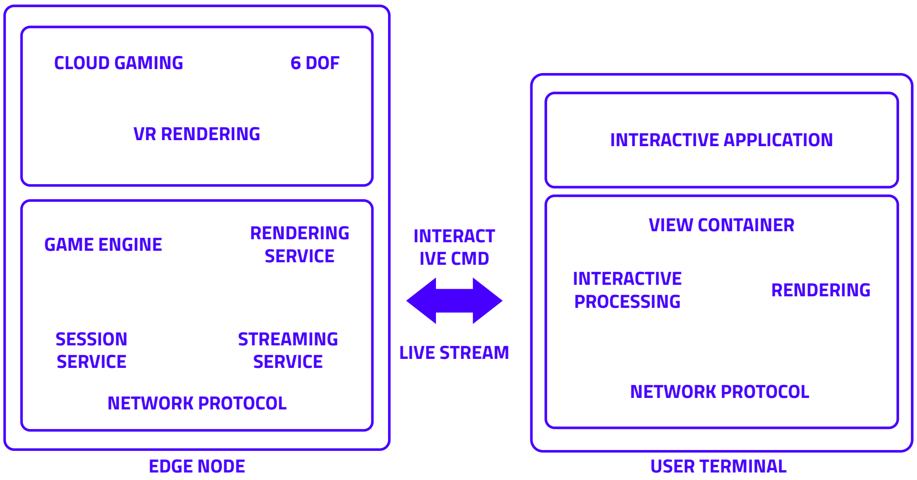
The following diagram is a schema of computing and distributed rendering that represents the bespoke development Naduceus has designed. This solves interactions with real-time data flows, including large quantities of calculations, which are some of the challenges in the gaming space:



The design of Naduceus' Decentralized Edge Rendering Technology is divided into five parts:

- Edge service framework
- Network protocol
- End-to-end interaction engine
- End-to-end scheduling system
- Application development tool chain

The edge service framework, network protocol, and interaction engine are shown in the figure below. These are responsible for the frame service capabilities of edge nodes, protocol processing of network communication, and terminal interaction and the rendering engine itself. The edge scheduling system intelligently balances edge node computing power with the user's local rendering capabilities in order to schedule rendering service processing between the user terminal or at the edge node.



Naduceus has self-developed a secure and efficient image transmission protocol to enhance the network optimization usage of cloud edge rendering. These enhancements include image information cache matching, adaptive video compression, intelligent transmission flow control, zero bus peripheral mapping and fine-grained transmission channels.

Here is the performance comparison between decentralized edge rendering and traditional rendering:

Performance	**Traditional Rendering**	**Decentralized Edge Rendering**
Security	Poor anti-DDOS attack capability and incompatibility.	All data is transmitted back to a large number of edge servers with high security.
Data storage	There are barriers in the cloud system, and data migration is difficult.	There is no need to establish their own private or centralized data center for their data needs.
Maintenance	The maintenance cost and time are directly proportional to the number of hosts.	The possibility of network interruption is very small.
Efficiency	Service is expensive.	Processing power depends on the intensity of processing tasks as configured, and depends on server configuration.
Scenes	**Existing Problems**	**Solution**
Gaming	Computers have high energy consumption, many failures, and are difficult to maintain hardware and software (drivers) without expert knowledge.	Simplification of operation and maintenance through decentralized management of edge rendering sets.
Teaching	Due to the frequent reinstallation of a large number of operating systems for different courses, the security management of the system in the campus network is difficult and the teaching management efficiency is low.	Adopt 3D edge rendering protocol, realize screen broadcasting, student presentation, file distribution, screen monitoring and other functions on the basis of simplifying the operation and maintenance process.
Medical	There are many businesses, and different scenarios such as nurse station, registration room, and imaging room require different systems such as HIS, LIS, PACS, RIS, CIS, 0A, etc., and maintenance is difficult.	Push relevant desktops adapted to different business systems according to different medical scenarios; implement template management for business system upgrades, software installation, system security reinforcement and other operations. The intelligent hardware driver separation technology enables one image to start multiple models, which can be fully compatible with HIS, PACS, LIS, various printers, scanners, ID card reader and other hardware peripherals and business systems without changing the original end-users' habits.
Defense industry	There are problems of military information leakage, data security and terminal supervision.	Realize physical or logical isolation of multiple networks through edge rendering, intranet office, extranet security access and other functions.

At present, the advantages of Decentralized Edge Rendering are:

1. Comprehensive enterprise end experience

- A digital graphical workspace that's available anywhere to enhance mobile productivity
- Enhance the experience, whether visual or digital, with zero downloading and high rendering quality
- Protect critical assets, secure access, developer, creator and user can fully control the ownership of their assets
- Simplify IT - centralized operation and maintenance of hardware assets
- More environmentally friendly.

2. Ability to quickly build basic services

- Cloud 3D/VR streaming for gaming or metaverse
- Cloud host video and video editing
- Cloud distribution
- Cloud rendering(such as: 3D-design, industrial-design, physical-simulation)
- Cloud Personal Computer
- Publish dynamic content(such as:providing graphic and video services) for Web3.0

3. Application opportunities

- There is a huge amount of data transmission and processing in scenarios such as industrial internet, autonomous driving, smart transportation, cloud gaming, and VR/AR.
- Decentralized Edge Rendering brings together ultra-low latency, massive data processing, edge intelligence, data security, and cloud collaboration. These are just some of the factors that drive businesses to enter the edge rendering space.
- If decentralized edge rendering technology is not adopted, not only does the bandwidth cost remain high, but also the extremely high demands for ever-increasing GPU scaling continue to be difficult to meet.

5.Core Engine

5.1 A Brief Description

The core module (core engine) is responsible for event creation, verification and 14 15 submission processing.

- **Event Packaging** needs to interact with the consensus module and the TxPool module, and trigger operations based on the current latest event and the preconditions (current) of the packaging.
- **Transaction Scheduling:** Responsible for calling the smart contract execution of the transactions which are waiting for Event Packaging. If a transaction is found to have a read-write set conflict, it will be rescheduled according to the rules, returning the transaction execution result and transaction execution order (DAG). At the same time, the module supports the specified DAG, executes the smart contract and obtains the transaction execution result, which is used for validity checks.
- **Event Verification:** after the node receives the Event that was received by the consensus or synchronisation module, the engine verifies the legitimacy of the Event. For details of the verification, please refer to the Process Description section.
- **Event Submission:** After the consensus or synchronisation module confirms that the Event is legitimate and has completed the voting process (required for consensus), the Event and the read-write set of the Event are recorded in the ledger database. Once the submission is successful, the latest Event information is updated to the ledger cache (ledger module), then the event of the new Event (the one just recorded to the ledger) is notified to the consensus, synchronisation and message subscription modules.

5.2 Process Description

5.2.1 Event proposal

1. Determine whether the current node is the proposer node (by obtaining notification 16 of the consensus module through the OnReceiveProposeStatusChange method and change the proposer status of the core engine).
2. Trigger Event Packaging
    a. Based on Reentrant Lock concurrency control, only one thread performs event packaging task execution.
    b. Triggering mechanism includes: the proposer's timeout or the size of the TxPool module exceeding the threshold.
    c. Determine whether the event can be packaged, if it passes, go to step 3:
        i. Confirm again that this node is the proposer.
        ii. Determine that at this stage the block is not being packaged and the proposal block is not yet in the settled status.
3. Event Packaging
    a. Determine whether this node has already proposed the event at the height of this event. If the event has already been proposed, it won't be packaged again, but it will be voted on for consensus repeatedly.
    b. Get a batch of transactions from the transaction pool.

c. Determine whether the ledger already exists or the obtained batch contains duplicate transactions and de-duplicate them if there are any. Following deduplication, if the transaction set is empty, stop packaging.

d. The verification node executes the transaction through the TxScheduler smart contract, and obtains the execution result of each transaction (including the read-write set).

e. Based on the result of the smart contract execution, the event gets packaged 17 along with the event transaction number, the event TxRoot, the event DAG hash and the event read-write set hash.

f. If the previous event is a configuration block, modify PreConfHeight to the height of the previous event and associate the configuration event.

g. When the event packaging has completed, the current proposed event is cached and the consensus module is notified.

h. If, while packaging, the proposer module is notified by the consensus module that it is no longer a proposer module, the Halt method of TxScheduler will be called to stop the scheduling execution of the virtual machine and the packaging event will be terminated.

5.2.2 Transaction Scheduling (the TxScheduler)

1. Execution Simulation (Schedule)

   a. Schedule the VM contract execution corresponding to each transaction request and collect the result. If there is a conflict between read-write sets, reschedule the execution (parallel execution).

   b. If it times out or the contract scheduling ends, it jumps out of the parallel execution logic.

   c. Build a DAG.

   d. Generate a read-write set map based on TxId, and return the map.

transaction execution algorithm to solve the problem of long transaction processing time when the volume of concurrency is high. The verification node performs transaction processing in batches according to the account relevance of the 18 transactions. For example, when the transaction chain comprises a -> b -> c and x -> y -> z, the root accounts a and x of the transaction have relative independence and can be processed in batches in the same batch. Meanwhile b and y need to be processed based on the result sets of the parent account’s transaction.

2. Verify transaction and read-write set according to DAG (SimulateWithDAG)
    a. Concurrently verify read and write sets based on the order of DAG.
    b. If timed out or the contract scheduling ended, it jumps out of the parallel execution logic.
    c. Generate a read-write set map based on TxId and return the map.

5.2.3 Event Verification

1. In the MetaverseGraph(mGraph), we use Event to record transaction information. Event is a data structure similar to the block in the traditional blockchain world. Usually, each node will create Events that contain four elements: transaction set, timestamp and two parent hashes. Every node that receives the Event will sign it, repackage its hash into a new Event and resend it to the network. When a node receives data containing new transaction information, it will combine them and possibly add transactions that it knows to the new Event.
2. Every node in the MetaverseGraph(mGraph) will try to sort out the order of the blocks. At this time a node will initiate a consensus proposal for itself; for example: ‘check whether block a is earlier than block b’, then the node will follow the hardcoded logic and rules according to the hashmap saved by the node itself to

perform vote-counting and consensus calculations from the perspective of each known node. The consensus here is asynchronous, which means that each node will 19 initiate a virtual vote at different points in time, make a decision, and assume that this proposal will obtain the same decision (consensus) in the entire network.

a. Accept and process the received event information.
b. Create a new event and point to its last event and the last event of the node that sent the event to this node (gossip node).
c. Assign all known events to create a round and determine whether the block is a witness in the round.
d. Vote for all known witnesses blocks and calculate results to decide whether they are famous witnesses.
e. Determine the consensus sequence of all blocks through famous witnesses.

5.2.4 Transaction Verification

1. The DAG blockchain technology supports high concurrency, combined with a two layer consensus mechanism and uses a proof-of-work consensus algorithm. In addition, it can also prevent 'The Double-Spending Problem'.
    a. Data is not strongly synchronised as with Bitcoin and Ethereum, but weakly synchronised, allowing nodes to have different data at the same time (the data can have some slight differences).
    b. Transactions can confirm the reference between data units, that is, the unit that occurs later refers to the previous unit. This way there is no need to pass the data to miners. The whole process is completed automatically, and quickly..

    c. Let's look at how DAG prevents 'the double-spending problem'. In Directed Acyclic Graph (or DAG), if a mainchain can be selected, all the nodes in the 20 DAG can sort this mainchain by connecting the serial numbers in a row. This will make this graph similar to blockchain in structure (sequential structure), that is: sorted events and each event is a transaction, not a block. Therefore, the mainchain is determined and the total sequence can be formed through the mainchain. The result is, in some logical states, the transactions are still sorted, which is the most critical core of DAG.

2. High-speed writes to the blockchain asynchronously in units of transactions in the traditional blockchain. The problem when making consensus, is that it needs to be built block by block. Before generating a block, all transactions need to be put in a Transaction Pool; the miners will pick transactions from the Transaction Pool to be packaged and then be able to create a block and add it to the blockchain. When generating blocks from transactions in the Transaction Pool, if no new blocks are mined and broadcast, these transactions are unconfirmed. This is a blocking write queue, which is a blocking issue.

    a. The book-keeping unit became more fine-grained, the book-keeping unit is not the block, it is the transaction. A transaction will be written immediately as soon as it happens, which is much faster than the traditional model of waiting for the block to be mined and then written.

    b. DAG brings into play the ability to use P2P Mutual Verification in wallet clients – this verification is parallel. Hypothetically, if, there were 10,000 transactions happening at the same time, they can go through 10,000 relational verifications between them. If this number is producing forks, there will be, at the same time, different wallets book-keeping different transactions into different forks. This leads to a problem – it can only be a partial order, not a total order. To obtain a total order, the mainchain needs to be determined. 21 Currently DAG-based projects such as IOTA and Byteball have their own way of determining the mainchain. Therefore, DAG book-keeping writes down the data, without checking for double-spending. In fact, a certain percentage eg: 1% of double-spending can be known, enduring double-spending for 2-3 seconds, then determining the mainchain, as soon as the mainchain is established, the double spends can be removed. Before the mainchain is determined, it is a parallel verification operation and it is placed on the data structure in parallel and then the data structure is checked to extract bad transactions. So DAG writes to the blockchain asynchronously in units of transactions.

5.2.5 Event Submission

1. Obtain the proposed event from the cache. If it does not exist, it means that the event has not passed strict verification and cannot be submitted.
2. Perform simple verification on the event to be submitted, including pre-hash and the hash of the event itself.
3. Call the storage module interface, save events and read-write set data, if it fails, panic.
4. Clear cache data, transaction pool data and snapshot data.
5. If it is a configuration block, notify the chain configuration (chainconf) module.
6. Synchronise the latest block to the consensus, synchronisation and message subscription modules through msgBus.

6. Smart Contracts

In order to build a DAPP ecosystem based on the NaduceusProtocol's leadin g high availability, high security and high TPS system, NaduceusProtocol wil l build the DAPP ecosystem based on the side chain, lowering the threshold of the DAPP construction through its own side chain and significantly lower the product development threshold through the provided open smart contract templates, the open source front-end module and the Bass/Fass platform. In the DAPP ecology, the XYZ chain is committed to building a three-layer ecosystem, which includes: the underlying P2P+ network, MetaverseGraph(mGraph), and the phenomenon DAPP side chain. Naduceus Protocol does not encourage the development of a large number of projects that cannot demonstrate real applications for blockchain, but it does encourage three types of project:

1) Projects that aim to transform the centralised internet products and services into decentralised DAPPs;

2) Projects that demand the transformation of their corresponding decentralised token economies to DAPPs;

3) Projects that aim to innovate and replace the traditional 5G, VR, AR and AI applications. NaduceusProtocol is committed to building a universal and practical token economic and commercial platform. The vision for Naduceus Protocol is to facilitate the development of DAPPs which reform the internet's pre-existing product models and rethink user needs and experience.

6.1 Contract Classification and Execution Process

NaduceusProtocol can run smart contracts based on WASM and EVM, in addition to multiple built-in system contracts. Smart contracts support user-oriented programming capabilities on the blockchain, while system contracts provide the necessary 22 23 conditions for the management and configuration of Naduceus Protocol's blockchain.

- When a transaction is executed in the contract module, first decide whether to hand it over to the smart contract or the system contract for execution based on the name of the contract.
- Before the smart contract is handed over to the smart contract engine for execution, it will go through a series of parameter verifications. These verifications include: bytecode, version, contract calling method name, contract calling parameters, and contract engine type.
- When the smart contract execution engine is started, the bytecode, version, contract call method name, contract call parameters will be parsed and serialised into the data required by the smart contract execution engine and the data will be copied to the smart contract engine.
- During the execution of the smart contract execution engine, it will execute according to the above information and return the contract execution result. Finally, the contract execution result is handed over to the storage module.

6.2 Introduction to Contract Engine

Naduceus Protocol currently supports four types of smart contract executi o n engine:

- **WASMER:** Supports the wasm bytecode of smart contracts generated by Rust language, which is executed by AOT technology at runtime.
- **GASM:** Supports the use of Go language to write contracts, uses the smart contract wasm bytecode generated by the TinyGo compiler and uses interpretation 24 technology to execute at runtime.
- **WXVM:** Supports the wasm bytecode of the smart contract generated by C++ language and uses localised compilation technology to execute at runtime.

- **EVM:** Supports the use of Solidity language to write contracts, uses the smart contract bytecode generated by the solc compiler and uses interpretation technology to execute at runtime.

6.3 System Contract

The current system contract includes:

- **SYSTEM** CONTRACTCHAINCONFIG: add, delete and modify chain configuration
- **SYSTEM** CONTRACTQUERY: query the configuration on the chain
- **SYSTEM** CONTRACTGOVERNMENT: on-chain governance
- **SYSTEM** CONTRACTMULTSIGN: multi-signature on the chain

6.4 Contract SDK

Naduceus Protocol provides an SDK for writing smart contracts in different languages and interacting on the chain. The main interface functions provided by the SDK include:

- Read the data on the blockchain database
- Write data to the blockchain database
- Obtain the current transaction ID and block height
- Obtain the identity information (public key, organisation, role) of the person who created the contract 25
- Obtain the identity information (public key, organisation, role) of the caller (ie the sender of the transaction)

7. Consensus Algorithm

7.1 MetaverseGraph(mGraph)

7.1.1. Algorithm Introduction

The consensus algorithm is based on the core concepts of MetaverseGraph(mGraph), which are as follows:

MetaverseGraph is a 3D bilayer graph structure consisting of two parallel 2-dimensional planes.

1. **MetaNebula** - The transaction issuance graph.
2. **MetaNeuron** - The transaction validation graph. Each transaction is sorted by consensus in MetaNebula and propagated, executed, verified and stored in MetaNeuron. As with quantum entanglement, MetaNebula and MetaNeuron are linked by the transaction queue number; MetaQuantum.

7.1.2. MetaQuantum

Also known as the transaction reservation number, generated by MetaNebula, which must be non-repeatable, non-missing, non-falsifiable and non-replayable.

- **MetaTxNo:** incremented continuously from 1. The upper limit is 2^64. MetaTxNo cannot be repeated. For example - the MetaTxNo of two transactions cannot be the same.MetaTxNo cannot be omitted so there cannot be other MetaTxNo between the MetaTxNo values of two similar transactions.
- **MetaTxTime:** The current system time (in nanoseconds) of the consensus node when MetaTxNo is generated. MetaNebula consensus nodes need to synchronise the time through the NTP (Network Time Protocol) to minimize the time error. Note: This time only serves as a reference. The system is sorting the transactions in MetaTxNo order.

- **MetaTxMD5:** To minimise the amount of network data and storage data, MetaQuantum does not save the original transaction content but only the 26 transaction summary. The MD5 Message-Digest Algorithm, a widely used cryptographic hash function, generates a 128-bit (16-byte) hash value to ensure that the message transmission is complete and consistent. The use of MD5 has two advantages: Firstly, it can be used by MetaNeuron to verify whether the transaction content has been tampered with. Secondly, the transaction content can be kept private, the MetaNebula network cannot identify the transaction content.
- **MetaTxSigner:** MetaQuantum's issuer address on MetaNebula MetaTxSign: MetaTxSigner uses its private key to sign the following: List{MetaTxNo+MetaTxTime+MetaTxMD5} Since MetaTxNo is continual, incremental, non-repeatable and non-missable and the content of MetaQuantum needs to go through MetaNebula for consensus, it cannot be tampered with nor forged. Furthermore, a MetaTxNo can only correspond to one unique transaction,so it is also non-replayable.

7.1.3. MetaNebula

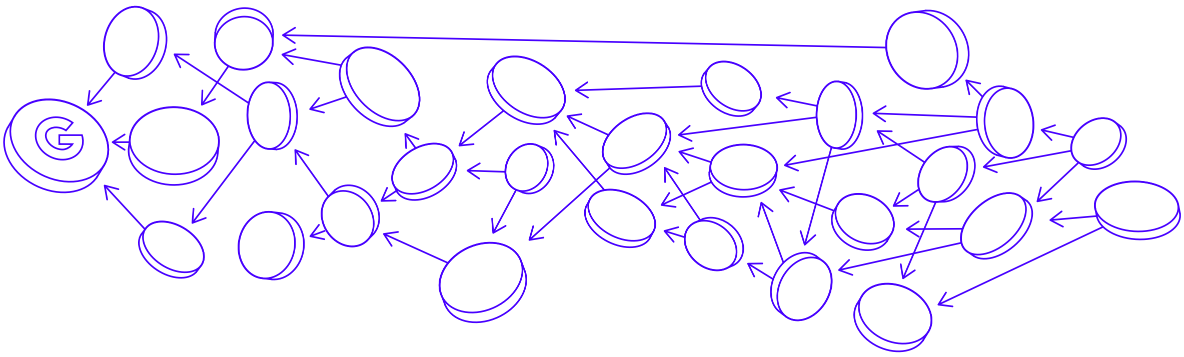


Figure 3: DAG

MetaNebula is a DAG consensus network that aims to maintain MetaQuantum in a decentralised way (As shown in Figure 3: DAG). Since the content of MetaQuantum is straightforward and uniform, unlike blockchain smart contracts with complex data and operations, the MetaNebula network can be made very simple and efficient. In addition, to further improve efficiency, the Tx transaction of MetaNebula network supports batch processing of MetaQuantum, such as generating 1000 MetaQuantum at one time, which can increase the sorting efficiency by 2 to 3 orders of magnitude and easily reach one million TPS.

- Issuance process: The requestor generates the transaction summary MetaTxMD5 in bulk based on the transaction content + random seeds and then send it to MetaNebula which generates List {MetaTxNo + MetaTxTime} + MetaTxSigner + MetaTxSign back to the requestor by consensus, while saving it in the distributed ledger with MetaTxNo as the Key and MetaTxTime + MetaTxMD5 as the Value.
- Verification process: The requesting party verifies the original transaction content and the content returned by MetaNebula with MetaTxSigner's public key and MetaTxSigner is within the MetaNebula consensus node list. The verifier can also verify by querying the MetaNebula transaction block content.
- Network bandwidth estimation. MetaTxMD5 occupies 16 bytes, 1000 or 16KB. MetaTxNo+MetaTxTime occupies 16 bytes, 1000 or 16KB. MetaTxSigner occupies 40 bytes. MetaTxSign occupies 64 bytes. Therefore, according to 1M TPS and single-link transmission, the upstream bandwidth needs 16MB/s and the downstream bandwidth needs 16.1MB/s.
- Estimation of storage space. MetaTxMD5 occupies 16 bytes, 1000 or 16KB. MetaTxNo+MetaTxTime occupies 16 bytes, 1000 or 16KB. Therefore 1M TPS storage performance requirement is 32MB/s. Performance Scalability: To further improve the TPS of the MetaNebula system, it can be implemented by the

following measures:1. Sharding Consensus. 2. Equalising Loading. 3. Cluster Nodes. 4.

7.1.4. MetaNeuron

MetaNeuron is an ultra-high bandwidth P2P network with the primary goal of broadcasting, validating, executing and storing transactions. MetaNeuron consists of two parts: MetaAxon and MetaDendrite（As shown in Figure 2:MetaAxo and Meta). MetaAxon is located on the backbone of the global submarine fibre optic cable and MetaDendrite is located in the data centre.

- **Broadcast:** MetaNeuron gets the transaction data sent by the requesting party and broadcasts it to other MetaAxons worldwide through the neighbouring MetaAxon. 29 All MetaAxons then broadcast to MetaDendrite in the region.
- **Receive:** The validation node pulls the transaction list from MetaDendrite periodically according to MetaTxNo by the maximum number of transactions or by the minimum time out of the block and packages them into blocks.
- **Execution:** The validation node executes the following processes in parallel for unrelated transactions from the block through the transaction parallel execution engine. - Verify that the specified MetaTxSigner issues the MetaTxSign signature. - Verify that the MetaTxMD5 transaction digest exactly matches the transaction content. - Verify that the transaction initiator's signature is correct. - Verify that the transaction initiator's NonceID is correct. - Verify whether the read/write set conditions of the transaction have changed. - If it has changed, re-execute the native transaction operation or call the virtual machine to execute the smart contract and generate a new read/write. - If not changed, write the write-set to the state database.

- **Verification process:** To prevent the nodes of the verifier from falsifying data and to tamper with transaction execution results. The explorer or wallet client collects feedback from the validation nodes of the transaction and uses the number of validation nodes that pass the validation and have the same TxHash as the number of confirmations. The higher the number of confirmations, the more validator nodes have validated the transaction correctly. Conversely, nodes that intentionally validate incorrectly can be identified and rejected by the wallet node.（As shown in Figure 3:Core Process)

Votes to confirm the state

8. P2P Network

8.1 Networking Method

- The P2P network of NaduceusProtocol is implemented and improved based on libp2p and the network address of the node follows the libp2p address format protocol.
- Automatic node discovery and automatic connection functions can be realised through the seed node settings. By default, each online node can serve as the seed node of other nodes to provide network discovery services. This way, the automatic networking mechanism of NaduceusProtocol is realised.
- NaduceusProtocol uses the message broadcast / subscribe function implemented by the improved libp2p-gossip-pubsub. It can ensure that the broadcast message can finally reach all nodes online. In the multi-chain scenario, each chain on the node enjoys an independent GossipPubSub service and through precise control of each Gossip routing table, the broadcast data isolation between multiple chains can be realized, ensuring that the broadcast data is only in the

”

”

- **First segment:** IP protocol version or DNS resolution protocol version. ip4 stands for IPv4, ip6 stands for IPv6; dns4 corresponds to IPv4 version DNS service, dns6 corresponds to IPv6 version DNS service
- **Second segment:** IP address or domain name, need to correspond to the first segment

8.3 Network Message Data Format Description (Before Encryption)

The message data before encryption is composed of 8-bits byte indicating data length + 1- bit byte as data compression flag + actual data, for example:

[00000007801057105......80858372]

Suppose this is a network message data to be sent, where:

- The first 8-bits [000000078] indicate the length of the data to be sent. When the receiver receives the data, if the received data length is less than this value, it will try to continue to read the data until the full length of data is received or the reception fails.
- The 9th bit, [0] or [1] is the data compression flag bit. If it is 1, the receiver will decompress the received data after receiving the complete data to get the final data result.
- Remaining bits, [1057105......80858372] are the original data to be sent (compressed or uncompressed data). Whether to compress should correspond to the 9th bit compression flag. The compression/decompression is done using the GZip toolkit.

8.4 Advantages

- Link channel security: This channel provides identity verification, realises encrypted authentication transmission and resists man-in-the-middle attacks.
- Protocol multiplexing: The underlying network service can accommodate multiple protocols running on the same connection. Peers use the names and versions they implement to communicate protocols, the underlying services identify compatible protocols and start peer connections on each matching protocol.

- Delivery guarantee: The protocol message has a delivery guarantee, that is, a delivery failure caused by a network problem will result in a direct error response. The order of message delivery in each protocol is guaranteed.
- Ideally, the underlying protocol provides priority: If the protocol is multiplexed on the same transmission channel, this probably means framing so that long messages will not block higher priority messages.
- Message serialisation: The protocol message structure supports arbitrary data structure serialisation conventions.

9. Storage Module

The storage module is responsible for storing events, transactions, ledger data and historical read-write set data on the blockchain. When an event is submitted, these data will be stored by the storage module.

9.1 Processing Flow of Ledger Storage

9.1.1.Event Submission and Storage Process

1. First, the serialised event data, read-write set list, and the latest event height are written to Eventbinarylog(wal) for recovery after abnormal interruption. At the same time, to improve performance, the cache layer is added, after the new Event submission request has updated the Eventbinarylog, the Event data (including Event, transaction, status data, read-write set) is written into the cache. After updating log and cache, the background thread will update EventDB, StateDB and HistoryDB asynchronously.
2. Record event information and transaction information in EventDB, where transaction information is stored with TxID as the key, event information is stored

with EventHeight as the key and only the transaction ID list is recorded in the event information. At the same time, the mapping relationship between EventHash and EventHeight is indexed, and the latest is recorded in EventDB. The event height 35 (LastEventHeight) is used as a checkpoint and submitted in a batch transaction to ensure the atomicity batch processing.

3. Record the state data modified by the transaction in StateDB, the key is the combination of the contract name and the primary key of the object: and the latest block height (LastEventHeight) is recorded as the checkpoint, submitted in a batch transaction to ensure the atomicity of batch processing.
4. Record the read-write set of the transaction in HistoryDB. The read-write set uses TxID as the key and records the latest event height (LastEventHeight) as the checkpoint, which is submitted in the form of batch transactions to ensure the atomicity of batch processing.

9.1.2. Ledger Recovery Process

If an exception occurs in the storage of a single database during the event submission process, the data between the databases will be inconsistent and the program will voluntarily exit after encountering this situation. The system will then enter the recovery process when restarting:

1. Get the latest event height from Eventbinarylog, EventDB, StateDB, HistoryDB and use the height in Eventbinarylog as the reference height to determine whether other DBs are behind the reference height.
2. If there are databases behind the reference height, get the missing blocks and read-write set from Eventbianrylog and submit them to these databases in turn.

3. After all databases are synchronized to the reference height, the storage module is started and the EventChain module continues to schedule other modules to complete the startup process.

9.1.3. Ledger Query Process

The query request first queries the kv data in the cache and return on hit, if the cache doesn't exist, then query from the database. For the delete operation, the cache provides a mark delete to indicate that the latest key has been deleted. For range queries, multiple pieces of data may exist in both cache and database, and data needs to be merged.

9.1.4. Ledger Database Types

Ledger database supports multiple different databases to match different business needs:

- LevelDB
- RocksDB
- MySQL/Distributed MySQL

10. Transaction Pool

The transaction pool module is used to store transactions received by nodes from the network. There are two ways to receive transactions from the network: 1) Clients (users/high-level Apps) add transactions to nodes through RPC; 2) Receive other nodes received and broadcast through P2P that they have received a transaction. When the transactions stored in the transaction pool reach the capacity limit, the core module will be notified to try to generate a new Event and DAG.

10.1 Transaction Types

The transactions stored in the transaction pool are divided into two types: configuration transactions and common transactions:

- **Configuration Transactions:** modify the chain configuration; if the event contains a chain configuration transaction, the event is limited to a total of only one transaction.
- **Common Transactions:** such as creating contracts, calling contracts and so on.

10.2 Feature Description

- **Start:** Start the transaction pool service.
- **Stop:** Close the transaction pool service.
- **AddTx:** Add transactions to the transaction pool. Source is the source of the transaction and there are three types: RPC, P2P, INTERNAL. Transactions from different sources correspond to different checks.
  - **RPC:** The transaction from RPC does not verify the basic transaction information (such as whether the transaction ID and timestamp meet the specifications) because the RPC module has done such verification. The transaction successfully added to the transaction pool will be broadcast to other connected nodes.
  - **P2P:** Perform all verification.
  - **INTERNAL:** If a node receives multiple valid verification events at the same height, when one of the events is on the chain, the transactions in the remaining events of the same height will be re-added to the transaction

pool to prevent these transactions from being discarded. At this time, the database will be used to check the existence of transactions added to the transaction pool and transactions that are not on the chain will be added to the transaction pool.

- **GetTxByTxId:** Query the transaction in the transaction pool based on the transaction ID. If the transaction exists, return transaction data and the height of the transaction recorded in the transaction pool. There are three types of transactions in the transaction pool:
    - Transactions that do not exist in the transaction pool, and the returned height is -1.
    - Transactions that exist in the ordinary queue of the transaction pool (ie: the queue of blocks to be packaged) and the returned height is 0.
    - Transactions that exist in the pending queue of the transaction pool (ie: packaged but not chained transaction) and the returned height is the event height of the exchange.
- **GetTxsByTxIds:** A batch interface for querying transactions in the transaction pool based on the transaction IDs;
    - Return value txsRet, store transaction content information, key is txId, value is transaction content; when the transaction does not exist, value is nil.
    - Return value txsHeightRet, stores the event height of the exchange, the key is txId, and the value is the height information
- **TxExists:** Check whether the transaction exists in the transaction pool, return true if it exists, and false otherwise
- **RetryAndRemove:** re-add the transaction of parameter 1 to the ordinary queue of the transaction pool, and if these transactions exist in the pending queue, they will be deleted from the pending queue, and the transaction of parameter 2 will be

deleted from the transaction pool. This interface is mainly called by the core module. When a node receives multiple different events at the same height, the transaction of the event to be chained will be deleted from the transaction pool; other transactions that are not connected to the chain event are re-added to the transaction pool.

  - Note: The internal implementation of this interface is: first add the transaction of parameter one and then delete the transaction of parameter two, so that even if the transaction of parameter one and parameter two partially overlap, they will be deleted from the transaction pool eventually.

- **FetchTxBatch:** Get a batch of transactions from the transaction pool. The maximum number of acquisitions is the number of transactions that can be accommodated in a single event; and the parameter is the height of the event to be packaged. It is called by the core module and uses the acquired transaction package to generate a new event.
  - When acquiring transactions, this batch of transactions will be moved from the normal queue of the transaction pool to the pending queue.
- **AddTxsToPendingCache:** Add the transaction to the pending queue of the transaction pool and if this batch of transactions exists in the ordinary queue of the transaction pool, they will be deleted from there; the second parameter is the event height of this batch of transactions.
  - When a node receives a new event, after the verification is passed, the transaction in the event is added to the pending queue of the transaction pool.

11. Component Description

The component description is divided into two parts: the components of the interactive module and the components of this module:

11.1 Components of the Interactive Module

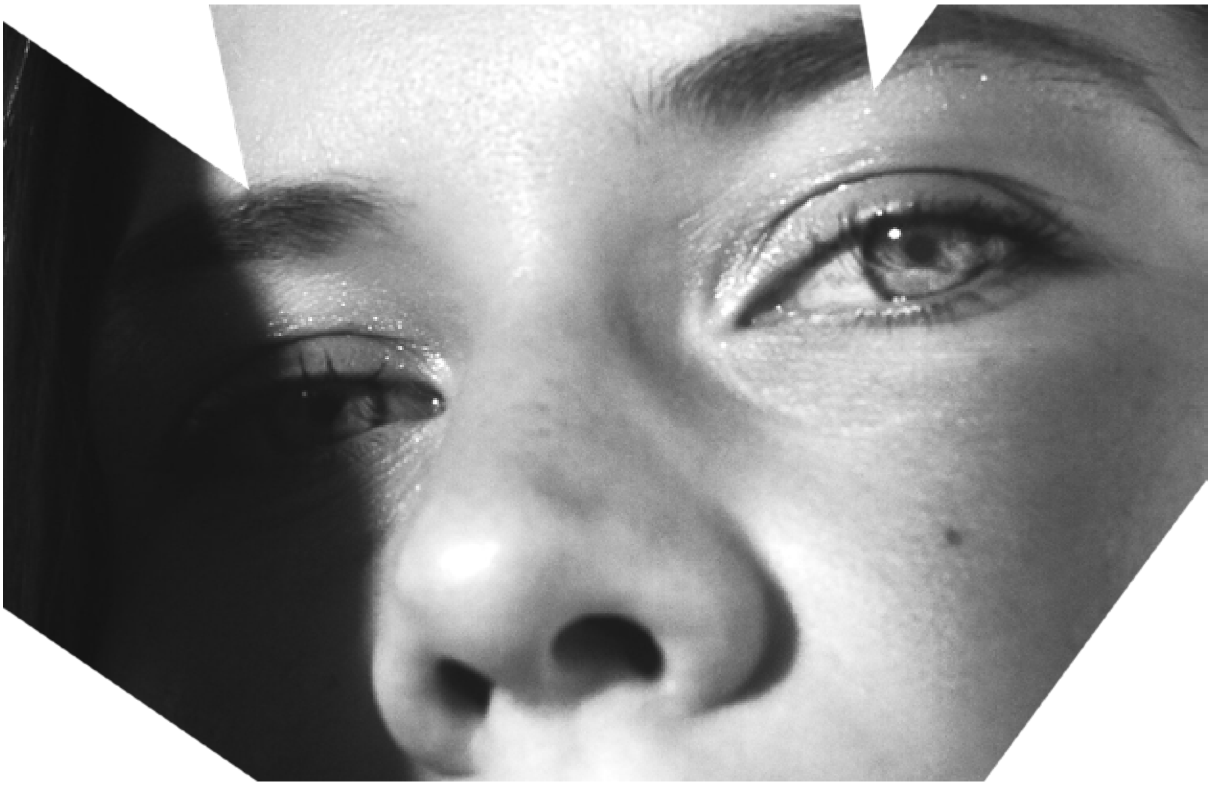
Uses the components of other modules to provide services such as network message communication, event verification and new events on the chain.

- **protocol.NetService:** sends or receives network requests and provides services for network information interaction with other nodes.
- **msgbus.MessageBus:** sends or receives messages to other internal modules and provides services for data interaction between the internal modules of the node.
- **protocol.BlockchainStore:** provides database query services to obtain information on the chain, such as obtaining event data at a specified height.
- **protocol.LedgerCache:** gets the latest on-chain status of the cache of the current node.
- **protocol.BlockVerifier:** provides verification services for the acquired events.
- **protocol.BlockCommitter:** adds verified events to the chain

11.2 Components of this Module

- **BlockSyncServer:** the overall structure of the sync module to provide external services, which relies on external module components and internal components.
- **Routine:** a tool which provides the hosting function of internal services, and uses a separate routine to run the registered service. It contains a priority task queue – the caller can add tasks to the queue and use the managed service to execute the priority queue in turn and return the execution result to the upper caller.

- **Scheduler:** an event request service. Internally maintains the status of all nodes linked to this node (the latest height of the peer node), the status of current known height events and the status of the requested event. When the upper caller receives the node status, it updates the internally maintained event status; at the same time, the upper caller periodically triggers the event request task. After the service receives the request, it selects an event to be synchronised and a request node according to the internal maintenance status, then sends the event request message to the selected request node. The received event information is then sent to the processor service.
- **Processor:** handles events services, internal maintenance received event information and the height of the next event to be chained. The upper caller periodically triggers the event processing task. The service processes the event to be connected to the chain according to its internal state. If the event does not exist, it skips the task processing until the event is received and returns the processing result of the event to the scheduler service.



Matt McGuire
CEO

Extensive experience across digital innovation sectors.
CTO of Heycar Group, Alphabet Collective & Critical Mass.

Alex Tung
CTO

Senior Tech Developer for over 20 years. Expertise in gaming, mining, Blockchain and consensus algorithims.

Justin Duffy
CFO

A finance director with big 4 audit training and over 15 years' experience as CFO/FD in international FinTech and the fast-moving consumer goods industry.

Tim Bullman
Advisor

Co-founded GlobalBlock in 2017 to provide agency broker solutions to the crypto industry. Guided the business to a successful TSXV listing in 2021.

Ander Tsui
Advisor

A Blockchain enthusiast and industry pioneer active in the space since its inception, as a creator, investor and advisor. Specialises in Strategic planning, tokenisation and financial modelling.

Bobby Chow
COO

Tech enthusiast with a Masters in finance and over 14 years' experience running international businesses. Specialises in designing business structure across different sectors, including Blockchain and FinTech.

Bertie Worsley
Advisor

Former senior partner at Digital RFQ, responsible for digital assets. Previously an investment analyst and consultant at FinTechs including HausBanc, Yokiki and Volopa Capital.

David Parrish
Advisor

An early advocate of Blockchain and crypto, advising banks and institutions on global strategy.
A senior VC partner and investment manager, specialising in early-stage DeFi, Blockchain and mining projects.

Jamie Khurshid
Advisor

Shortlisted by Financial News 2014 as one of the 'Top 40 under 40 in trading and technology' and 'Top 1000 most influential people in global financial markets' by Exchange Invest.

14. Conclusion

NaduceusProtocol, by adopting the MetaverseGraph(mGraph) consensus mechanism that uses concepts of "gossip," "gossip about gossip" and virtual voting, solves problems of standard consensus-building algorithms such as proof of work (PoW), by achieving greater speed and higher efficiency as it does not send any votes or details over the network, which often leads to congestion and delays.

With the provided set of development tools (SDK, Command Line tools and IDE), building scalable decentralised blockchain systems has never been easier, more convenient or more efficient.

15. Disclaimer

Issuers and Naduceusdo not intend to make representations, warranties or commitment s to any entity or individual. In general, without limiting the foregoing, the issuer and Naduceusteam do not guarantee the accessibility of tokens, quality, suitability, accuracy, adequacy or completeness, nor do they make any express or implied or other statements. Services related to the Naduceusplatform or Naduceustoken do not give any guarantee s

including non-infringement of third party rights, title, merchantability, satisfactory quality or fitness for a particular purpose of the guarantee.

15.1 Risk and Uncertainty

Potential buyers and holders of tokens should carefully consider and evaluate the issuer, Naduceusplatform, and the potential risks of token rights. Prior to purchasing or acquiring tokens, it is imperative to read this white paper and terms and conditions. If any of these risks and uncertainties develop into actual events, the distributor and/or Naduceus platform business, financial condition, results of operations and prospects may be materially adversely affected. In this case, you may lose all or part of the value of the token. Risks set forth in this paper are not an exhaustive list of the risks of issuers, Naduceusplatform and/or tokens face, or may develop in the future. There may be other risks not described here or currently unknown by the distributor, or other risks that the distributor currently considers unimportant and these risks may become important in the future. Other known or unknown risks might occur in the future that have significant adverse effects and damage the business operations of the Naduceusplatform and/or the Naduceustoken.

15.2 Regulatory Risk

In Singapore, the regulation of tokens is still in its infancy. There is a high degree of uncertainty regarding how to deal with digital tokens and token-related activities. The applicable legal and regulatory framework may change after the publication of this white paper. Such changes (whether expected or retroactive) may be very rapid or unpredictable and it is impossible to predict the nature of such legal or regulatory changes in any deterministic way. In view of this, issuers and Naduceusstate that laws or regulations of tokens will be affected by any legal or regulatory changes.

If regulatory actions or legal or regulatory changes result in illegal operations or commercially undesirable ramifications of obtaining the necessary regulatory approvals in the jurisdiction, the issuer (or its affiliates) or Naduceusmay stop operating in that jurisdiction.

In Singapore, MAS regulations generally do not extend to the security and reliability of cryptocurrencies, cryptocurrency intermediaries, or the proper processing of cryptocurrency transactions. However, if a cryptocurrency intermediary is found to have used cryptocurrency illegally, law enforcement agencies may shut down its operations. If any digital token exchange, issuer, or intermediary violates Singapore Securities Law, MAS will take firm action. The public should be aware that if they choose to trade on unregulated digital token exchanges or invest in digital tokens that are beyond the scope of the MAS regulations, there is no regulatory guarantee.

15.3 Tax Risk

The tax characteristics of tokens are not yet clear. Therefore, the tax treatment they will receive is uncertain. All people who wish to receive tokens should seek independent tax advice before deciding whether to accept any tokens.

15.4 Security Risk

The security, transferability, storage and accessibility of the token depend on factors beyond the issuer's control, such as (but not limited to) mining attacks, malware attacks,a nd spoofing. The issuer cannot guarantee that such external factors can prevent any direct or indirect adverse effects on any token. Those who intend to receive replacement tokens should note that adverse events caused by such external factors may result in the loss of some, or all, of the tokens. This loss may be irreversible. Neither the issuer, or Naduceusteam members, are responsible for taking steps to retrieve any tokens lost in this way.

15.5 Other Risks

The potential risks mentioned briefly above are not exhaustive, and there are other risks associated with the purchase, holding and use of tokens, including risks that the issuer cannot anticipate. Before purchasing or buying tokens, you should conduct a comprehensive due diligence on the issuer, its subsidiaries and Naduceusteam and understand the overall framework, mission and vision of the Naduceusplatform.

15.6 Other Considerations

No part of this white paper may be copied, reproduced, disseminated or distributed in any way without the issuer's prior written permission. The distribution or dissemination of this white paper or any part of it may be prohibited or restricted by the laws, regulations and rules of any jurisdiction. If there are any prohibitions or restrictions, you should inform them at your own expense and to comply with any applicable prohibition or restrictions you might.

This white paper or part thereof (as the case may be) does not assume any responsibility that relates to the issuer and / or Naduceus.