HyperCycle Core v0.9

Authored by Toufi Saliba and Robert Moir PhD
with collaboration by Dann Toliver, Greg Colvin PhD,
Ernie Casilla, and Barry Rowe PhD

HyperCycle provides the essential missing components required for the Internet of AI, where networked AI agents can cooperate and compete at the micro level in ways that bring about emergent intelligence at the macro level.

Facilitating cooperation between micro-level computational modules requires secure peer-to-peer contracting and subcontracting. This allows modules to compensate each other in milliseconds for computations without relying on third-party intermediaries, enabling the direct transmission of extremely small values.

This vision of an Internet of AI has been a longstanding goal for computer and AI scientists. Advancements in network speed, cryptography, and related fields have led to the development of a system that enables direct, peer-to-peer transactions, leading to substantial reductions in time and energy costs.

1 Introduction....2
2 High Level View of the HyperCycle Network....3
2.1 The TODA Protocols and Earth^{64} Sato-Servers....3
2.2 The HyperCycle Virtual Machine....4
2.3 Securely Getting Funds into the HyperCycle Network....5
2.4 Enabling Cooperative AI for Beneficial Emergence....6
2.5 Competitive AI for Progressive Evolution....7
3 HyperCycle Computation Nodes....8
3.1 Node Structure and Function....8
3.2 Starting Operation of a Node....10
3.3 Revenue From a Node / Incentive Model....11
3.4 Node Score = Uptime · Computation · Reputation....11
3.5 Node Splitting....12
4 HyperCycle Transactions....14
5 Conclusion....16
6 FAQ....17
A Appendix....19
A.1 Various Kinds and Forms of HyperCycle Tokens....19
A.2 Provenance System....20
A.3 HCN Virtual Machine....21
A.4 HCN Transaction Machines....22
A.5 HCN AI Machine....23
A.6 HCN Node Manager....23
A.7 HCN Merklizer....24
Glossary....26

1 Introduction

Computational theories of the mind emerged in the middle decades of the 20th century. We build on these, modeling the mind as a secure, distributed, scalable and fully decentralized network of interacting nodes. The nodes of this network asynchronously exchange computations, laying the groundwork for complex forms of thought and awareness.

Building the Internet of AI on a global scale requires solving some of the hardest problems in parallel and distributed computing in a way that ensures that the system will be secure, efficient, confidential, scalable and interoperable by design.

Accomplishing this requires an economic and computational protocol that allows nodes on the internet to transact peer-to-peer (P2P). A step toward this was made in 2008 with the invention of ledger-based blockchains, starting with Bitcoin as a byzantine-fault-tolerant protocol for achieving distributed consensus on updates to an immutable ledger. A further advance came in 2014 with the Ethereum blockchain, which allows arbitrary code to be stored and run on an immutable ledger.

HyperCycle advances on these remarkable achievements to create a platform where AI agents with complementary capabilities can seamlessly transact, enabling them to collectively tackle problems of ever-increasing size and complexity, empowered by agent-to-agent microtransactions for microservices with sub-second finality.

The TODA protocols provide the essentials. The ledgerless TODA/IP consensus protocol minimizes network traversal and performs only the minimum computation necessary to secure the network itself. And building on TODA/FP, Earth^{64} Sato-Servers ensure that the assets they carry are secured independently of any blockchain ledger. Thus throughput is maintained and security grows as nodes are added—the network can grow to global scale. This eliminates the key communications bottlenecks of previous ledger-based blockchain protocols to provide true parallel and distributed computing while conserving the maximum amount of decentralization.

By leveraging the novel capacities of the TODA protocols, HyperCycle paves the way for the realization of a global computational mind, where the collective intelligence of AI agents harmoniously converges to address multifaceted challenges. And because it is governed by the entire open-source network of developers and operators, as opposed to a single entity or small group, the emergent effects of expansive connectivity, competition and cooperation among self-actualizing nodes has a higher probability of enabling the global community to reach its goals, including the safer introduction of new forms of intelligence.

2 High Level View of the HyperCycle Network

In this section we consider how HyperCycle Computation Nodes (HCNs) link together using their virtual machines, how money securely enters and travels through the network using cryptographic proofs of provenance, and how the network is designed to evolve progressively and facilitate the emergence of new forms of intelligence.

We start by describing the TODA Protocols and the Earth^{64} Sato-Server data structure that are the key technological innovations that make the HyperCycle network possible.

2.1 The TODA Protocols and Earth^{64} Sato-Servers

With current network and hardware capabilities each agent can make multiple requests to other agents concurrently, who can do the same recursively, allowing a single node to effectively communicate with hundreds of millions of nodes. However, technological innovations are required to accomplish this at network speeds for asset transfer and not just data. HyperCycle leverages the TODA Protocols and the Earth^{64} data structure for digital asset management and distributed consensus to make this possible.

On the digital asset side of the TODA Protocols is TODA/FP, the TODA File Protocol, provided by T.R.I.E. and TODAQ. TODA/FP defines a cryptographic data structure that allows the state of an asset to be managed independently of its integrity, where ‘integrity’ refers to non-equivocation or the fact that an asset has not been spent more than once. Thus, a system managing the integrity of an asset does not need to hold or even know the state of that asset, minimizing the cost of operating the integrity system and allowing the asset’s state to be held on any device.

TODA/FP guarantees that as long as the integrity management system does not equivocate (generate multiple histories), then the asset cannot equivocate either. This integrity-at-a-distance property is mathematically proven, and the required data for the lightweight proof is carried along with the asset during its transfer from one party to another.

The Earth^{64} Sato-Server data structure builds on the capabilities of TODA/FP and adds two key features: the ability to have integrity-system independence, thereby allowing the integrity provider of an asset to change over time; and the capacity to create, split, and merge nodes of the Sato Tree, the binary tree of Sato-Servers that Earth^{64} defines. The result is an asset-level, portable blockchain that is low-cost, scalable and can be plugged into different systems to manage its integrity.

HyperCycle makes use of the splitting capacity of Sato-Servers for both the native HyPC token and HyperCycle Computation Node licenses. In the case of HyPC tokens, they can be subdivided into smaller denominations. For HyperCycle Computation Node licenses, they can be split into smaller licenses or can generate additional licenses, as is described in Section 3.5 below.

Other important features of the Sato-Server data structure are that it is a container for arbitrary digital items of value and that each one has a globally unique identifier. HyperCycle uses the container feature to allow native HyPC tokens to hold other value instruments, including fiat and crypto currencies. Each Sato-Server also has a unique binary identifier that corresponds to a unique region on the surface of the Earth, achieving global uniqueness using the internationally recognized GCS coordinate system.

On the distributed consensus side, TODA/IP employs a distributed computation as proof of work, which scales linearly with the network's size, ensuring a constant workload for each node. To accomplish this, the data structure is designed to maintain only the essential data required to secure the network's state, minimizing the distributed computation needed for security. Transacting nodes store the state of their own assets, eliminating the need for a global ledger, replacing it at the network level with a single global Merkle root hash per network cycle.

In each cycle, parties involved in transactions can engage in an unlimited number of transactions across as many assets. However, only one transaction per asset is permitted per cycle. Within the context of HyperCycle, these cycles operate at varying scales within the network. Smaller systems of transactions can achieve finality relative to a portion of the network in fractions of a second, while finality at the network-wide level is achieved in each network block every 6-30 seconds. The former guarantees the necessary performance for high-speed transactions, while the latter ensures the certainty required for asset re-spending.

The combination of the TODA Protocols and Earth^{64} Sato-Servers enables true P2P transactions between HyperCycle nodes. Network packets serve as carriers for instructions, data, *and* payment. Since transactions can achieve sub-second finality, economic signals and data coexist efficiently, allowing the network to rapidly evolve new problem-solving approaches while meeting the performance expectations of the global market.

2.2 The HyperCycle Virtual Machine

HyperCycle Computation Nodes act as the “neurodes” whose interconnections form the axons and synapses of the global HyperCycle brain. This demands a powerful virtual machine at the heart of every node. This Virtual Machine enables efficient, secure communication between the internal components of an HCN and between the nodes themselves.

The internal components of a Computation Node include one or more Transaction Machines (TMs) and one or more AI Machines (AIMs), loosely coupled via the HyperCycle Virtual Machine (VM). Internally to a node, the Transaction and AI Machines communicate solely via the VM. All components are modular—any components providing the necessary interfaces can be used as HCN machines. Externally, the VM for each HCN manages all non-financial communication with other nodes by talking to their VMs. Financial communication is handled independently by the TMs and the TODA/IP network.

For an HCN to serve as an effective neurode, the VM must be dynamically programmable and reprogrammable. HCN VMs communicate payment instructions, data, and code to other VMs. VMs are paid to execute the code on the data, controlling the execution of the AIMs and initialing further communication.

The initial HCN Virtual Machine will be based on Python. Millions of programmers use Python, including many of our potential users: developers in the blockchain, data analytics, machine learning and AI communities. Python is modular and composable, and it supports embedding code from other languages, including many powerful libraries. And crucially, Python is a dynamic language that can execute its own source code, which can be sent as messages to and from HCN VMs. In this way one VM can in fact dynamically transmit code to be executed on another VM.

So what we have here is a network of computational agents communicating code, data, and payments to other agents. This empowers nodes to establish spontaneous transactional relationships on-the-fly. This dynamic formation of ad hoc connections between nodes ensures flexible and efficient collaboration in the negotiation and execution of contracts and subcontracts for artificial intelligence work.

2.3 Securely Getting Funds into the HyperCycle Network

Provable provenance of funds is crucial within the HyperCycle network, as it requires the conversion of various currencies into native tokens (HyPC) while maintaining traceability. The provenance of funds encompasses two elements. The first involves existing methods for establishing the provenance of fiat currency, bridging to stablecoins or cryptocurrencies, and transferring tokens across blockchains. HyperCycle does not rely on any specific method but requires a provable provenance from the fund source.

The Ethereum blockchain is very useful in this respect, because users can convert fiat into ETH or ERC20 tokens, which can easily be converted into HyPC.ETH or c_HyPC.ETH (NFTs corresponding directly to unique native tokens), providing a borderless means of payment for services on the internet using the Ethereum blockchain. This kind of token bridging is of course not exclusive to Ethereum, and can be extended to other programmable blockchains such as Polygon, Cardano, or BNB Chain.

The second element of provable provenance relates to the native HyPC token in its true form as a Sato-Server, which employs TODA/IP to ensure integrity by preventing double spending while maintaining a proof of provenance within the data structure. This ensures traceability of funds throughout the entire transaction history in the HyperCycle network.

Together, these two elements establish provable provenance for all funds within the HyperCycle network. To illustrate, Alice could use one of many exchanges to convert USD into ETH, which can be used to purchase HyPC.ETH using a decentralized exchange. She can then use our swap contract to convert her HyPC.ETH into c_HyPC.ETH on the Ethereum network with which to form her node's identity or to pay for AI services on HyperCycle. Similarly Bob could convert BTC or even hypothetically USD or GBP (£) to GBPT using a regulated bridge from the central bank of England, and use a decentralized exchange to swap GBPT for HyPC.ETH. Whatever the origin of funds, there will be a provable provenance onto the blockchain and into and throughout the HyperCycle network.

It is important to appreciate that each native HyPC Sato-Server holds both value and functions as a container, which allows HyPC to carry payments in other token types through various bridges. Incorporating an Ethereum Virtual Machine (EVM) into HyperCycle nodes as a TM makes this even easier, facilitating the use of ETH and other ERC20 tokens for payments in the HyperCycle network and allowing revenue to be cashed out directly to Ethereum.

2.4 Enabling Cooperative AI for Beneficial Emergence

Nodes in the HyperCycle network possess dual functionality: running AI machines to offer services and making requests for services from other nodes. This unique capacity enables nodes to contract and subcontract work, allowing ad hoc networks of nodes to dynamically solve problems more effectively and efficiently, fostering beneficial emergence through novel node combinations.

Work requests in HyperCycle involve three elements: the HyPC Sato-Server containing the payment, instructions for the task, and the relevant data. When Bob receives a request from Alice, he validates the Sato-Server to ensure its authenticity. To claim the payment, Bob needs to provide proof of completing the task using the instructions and data provided by Alice.

For example, suppose Alice wants to process presentations by converting images and audio to text. While she has AI machines for text processing, she lacks image OCR and audio-to-text capabilities. Alice discovers that Bob can perform these tasks and contracts him, paying with HyPC micropayments per image and audio file.

If Bob encounters challenges, such as Latin text or noisy audio, he can subcontract to other nodes on the network. Bob searches and finds Charlie for Latin OCR and Dana for audio enhancement, and subcontracts the problematic cases to them. This collaborative approach allows Bob to overcome obstacles instead of being blocked by them, allowing the network to provide a comprehensive solution to Alice's initial request.

HyperCycle enables dynamic problem-solving through cooperative networking of compute units and AI. It provides a distinct and novel capability for the Internet of AI, allowing arbitrary nodes to securely contract and subcontract at a microtransaction level. This enhances global-scale emergence in collaborative task completion, distinguishing it from centralized environments like AWS, GCP, and Azure.

2.5 Competitive AI for Progressive Evolution

To promote a naturally progressive evolution of the Internet of AI that benefits humanity as a whole, fostering competition between nodes providing similar services is crucial. The HyperCycle network's ad hoc P2P structure facilitates this process by design.

Suppose Alice discovers a new node called Hannah, which outperforms Bob's devices in accuracy and performance but on a smaller scale. By allocating a small amount of work to Hannah's service, Alice enables Hannah to grow and improve over time. This encourages Bob to innovate and enhance his own service quality, generating market signals that contribute to improvement of the overall capabilities and functioning of the network.

Enabling competition among nodes offering similar services is not a new concept, having existed in the AI field for decades and in economics for centuries. However, what sets HyperCycle apart is the ability to achieve this beneficial evolutionary behavior with high speed, scalability, and minimal friction. This achievement is made possible by addressing long-standing bottlenecks, particularly designing secure communication protocols that scale linearly and transmit microvalues at negligible cost. In this regard, the foundational technology of TODA/IP plays a pivotal role, as it overcomes these challenges to enable HyperCycle's beneficial self-evolution and emergence.

3 HyperCycle Computation Nodes

The Internet of AI necessitates seamless networking between AI software components. HyperCycle Computation Nodes act as specialized agents, enabling communication and transactions.

This section explores the structure and functionality of HCNs, accommodating any AI software and facilitating inter-node communication. We discuss operating requirements and revenue collection, as well as wealth creation using unique node identities that track nodes' quality of service and allow high-quality machines to split licenses, creating two nodes from one.

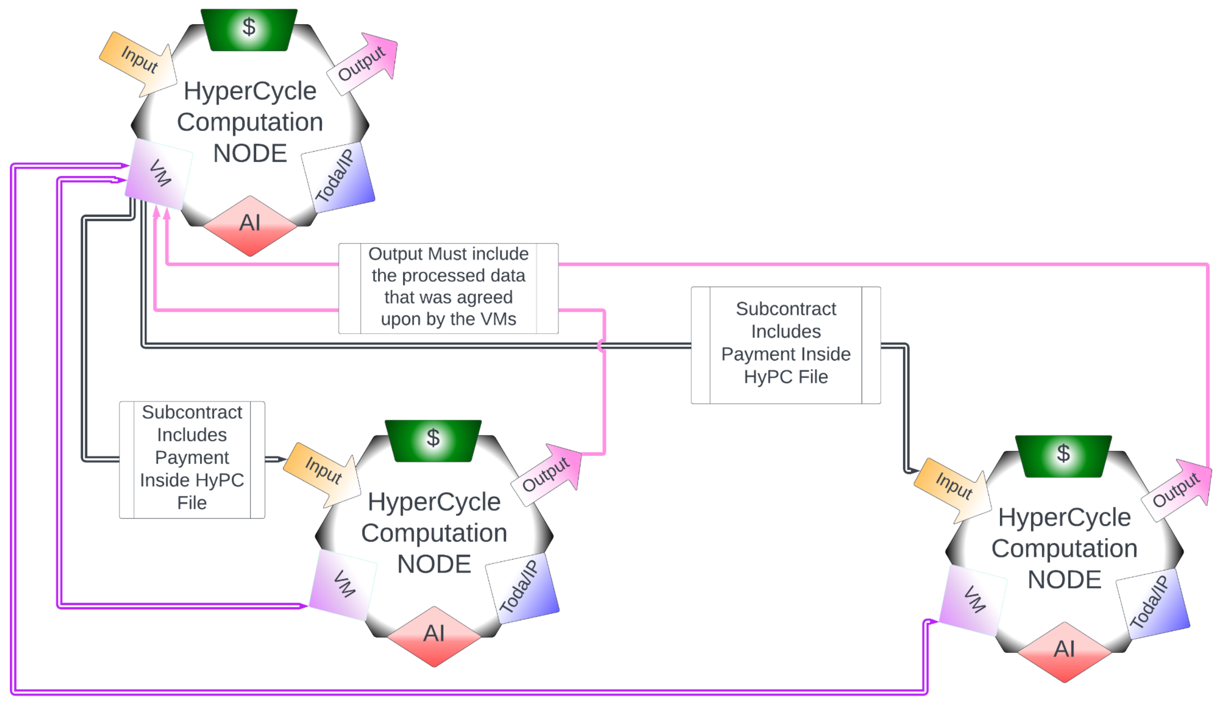
Let's begin by taking a look at what is inside an HCN.

3.1 Node Structure and Function

As mentioned above, an HCN comprises three essential components: the VM, TM, and AIM. The VM interacts with the TM for handling payments and the AIM for AI computation. These components collaborate to enable nodes to perform their function within the network.

Upon installing the node software, the operator selects AI modules to run within the AIM, utilizing Docker container images or other means of software provisioning. Using the VM, the node informs other network nodes about its software capabilities and corresponding pricing.

When Alice requests computation from Bob, she sends an HyPC Sato-Server containing a payment commitment along with instructions, and data. The payment commitment is irreversible but remains unspendable by Bob until a proof of computation is provided, which may be the computed output. Bob's VM verifies the payment validity with his TM, sends the instructions and data to the AIM for computation, and receives the result. Bob sends the output to Alice and, with proof of computation in hand, his TM can release the payment. Bob's VM can subcontract parts of the work to Charlie and Dana in a similar way, as depicted in the image below.

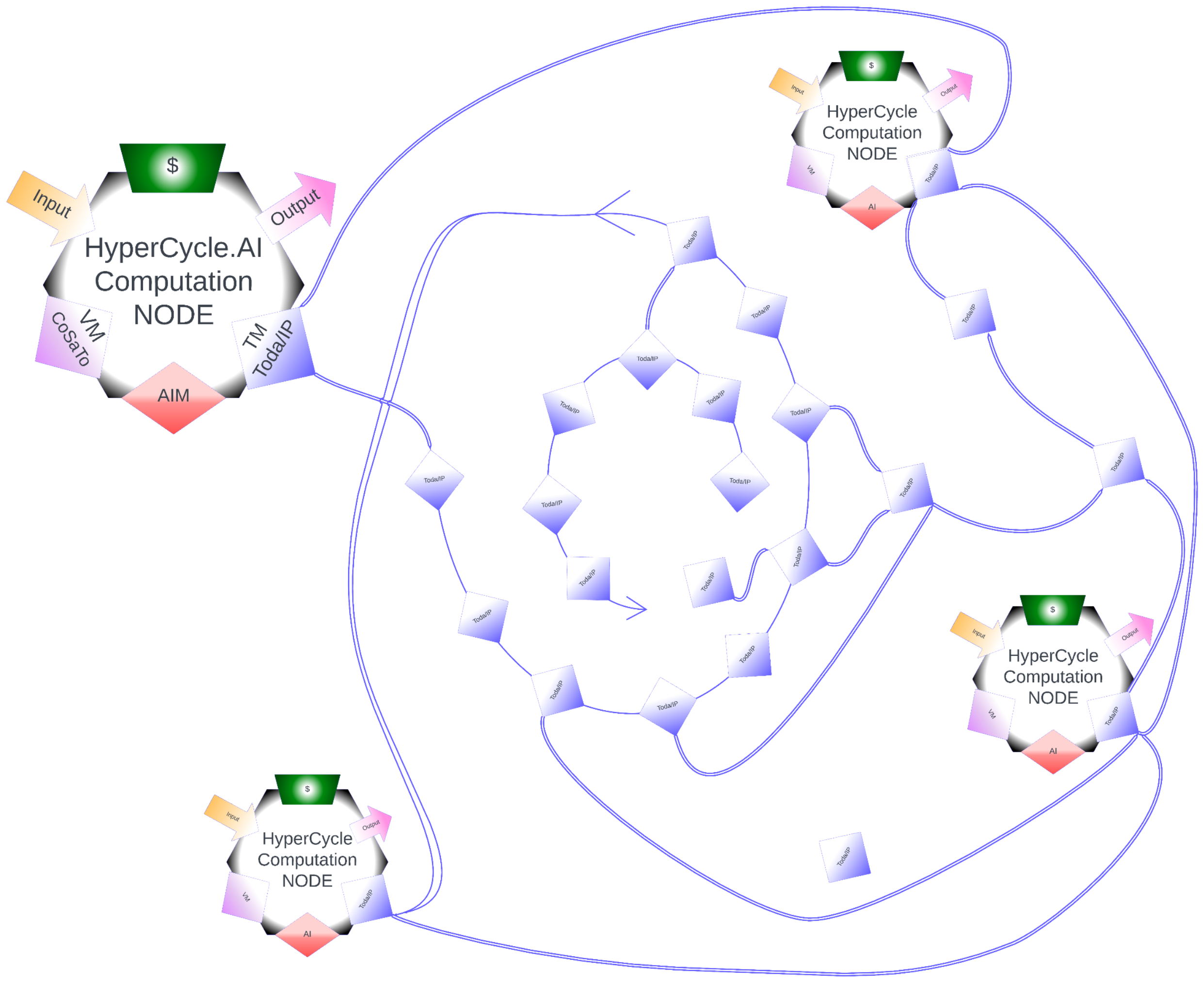

$
Input
Output
HyperCycle Computation NODE
VM
Toda/IP
AI
Output Must include the processed data that was agreed upon by the VMs
Subcontract Includes Payment Inside HyPC File
Subcontract Includes Payment Inside HyPC File

Using TODA/IP, the TMs can transact with each other in parallel to the communications handled by the VM. The diagram below visually depicts the network's functionality from the TMs' perspective. In this system, payments and payment commitments are established within the TODA/IP network (node → node and node → TODA/IP messages in the diagram) to ensure the integrity of the exchanged Sato-Servers.

These payments and commitments attain finality within a segment of the network by being included in a TODA/IP cycle (TODA/IP → TODA/IP messages), thereby guaranteeing immutability and finality within the validating node set. Subsequently, proof of inclusion in a cycle is communicated to the participating nodes (TODA/IP → node messages), instilling confidence in the firmness of the transaction.

These cycles are eventually consolidated into a network block (interior spiral messages in the diagram) at regular intervals of 6-30 seconds, thereby establishing the ultimate finality of payments within the network and, because each asset can only transact once per block, permitting the reuse of funds.

Negotiations between nodes for contract terms occur infrequently, and once established, work requests can be sent as P2P transactions using Sato-Servers without third-party involvement. Renegotiation becomes necessary only when there are changes in services, availability, capacity, or prices.



3.2 Starting Operation of a Node

To set up a functioning HCN, several preliminary steps are required. Firstly, the node needs powerful hardware to run competitive AI software. Secondly, a HyperCycle license is necessary for network operation. Lastly, 1024 HyPC are needed to establish each node's identity. The combination of a license and a certain quantity of HyPC appropriate for that license is called a Provenance Marriage. This marriage activates a license and enables a cryptographic quality of service score and other benefits, as described in Sections 3.4 and 3.5 below.

These three pieces—hardware, license, and native tokens—are essential for a fully operational node on the HyperCycle network. Activating a HyperCycle license involves a Provenance Marriage, where the license is combined with the required HyPC identity token, forming the node’s identity. The married license and token are then integrated into the node software. This allows the node to communicate its capacities, cryptographically record its behavior, and commence operations.

Although all three pieces are necessary for a fully operating node, they can be obtained separately by different parties. Collaborations between owners of hardware, licenses, and tokens enable joint node operation at reduced costs for each individual. Pools or marketplaces facilitate bringing together pieces owned by different parties, providing HyPC tokens, licenses, or hardware to those seeking specific elements to initiate and sustain a node.

It is important to note that these pools or marketplaces, while vital to the ecosystem, are distinct from the HyperCycle core and are not owned or managed by HyperCycle.

3.3 Revenue From a Node / Incentive Model

An HCN generates revenue by providing AI services and receiving payment in HyPC tokens, possibly along with other value instruments such as AGIX, ADA, EUR and USDC. The HyperCycle network operates without transaction fees but collects a 1% royalty on node revenue to support network participants, including HyperCycle itself. Royalties are distributed securely and fairly using an automated, deterministic algorithm.

Nodes gain value through a score that reflects their uptime, computation, and reputation. A higher score increases the likelihood of receiving work, thereby increasing the node's value. When a licensed node achieves a score of 2, it can split into two licensed nodes, further enhancing its worth.

3.4 Node Score = Uptime · Computation · Reputation

The HyperCycle network relies on nodes proving their availability (proof of uptime), accurate and timely computations (proof of computation), and responsible behavior (proof of reputation) to ensure high quality network functionality and security.

Data records for uptime, computation, and reputation are regularly recorded in cryptographic form within network intervals ranging from 6 to 30 seconds. These records serve as evidence of a node's behavior by demonstrating their inclusion in the cryptographic records for each 6-30 second block of network time.

Using the recorded data, any one can calculate, and the network can deterministically confirm, ratios for three properties: uptime; computation; and reputation. These ratios provide insights into a node's performance for each corresponding property. Each ratio is calculated as a measured percentage for the property multiplied by a scaling factor:

```
property_ratio = property_% * scaling_factor
```

where the initial percentage for each property starts at 100% but can vary over time, while the scaling factor starts at 1 and gradually increases to a maximum of 2 after 18 months. As a result, each ratio starts out at 1 but can range in value from 0 up to 2.

The overall score for a node is obtained by multiplying the three ratios—Uptime, Computation, and Reputation—together, with the maximum score capped at 2:

```
score = min ( Uptime * Computation * Reputation, 2 )
```

Consequently, the node's score also initially starts at 1 and takes values in the range from 0 to 2 over time.

The role of the scaling factor in the computation of each ratio is to increase the maximum possible value of each ratio as time progresses. In particular, after six months the scaling factor for each ratio reaches up to $\sqrt[3]{2} \approx 1.26$. This enables a node's score to reach 2 after a six-month period if it maintains the maximum values for uptime, computation, and reputation during that time.

Even if a node doesn't achieve a perfect score for one or more properties, it still has the opportunity to increase its score to 2 because the maximum value of each ratio continues to rise past $\sqrt[3]{2}$ after six months. This provides various paths for the product of the ratios to reach 2.

This continual increase in the ratios' maximum value gives particular significance to the uptime. The Computation and Reputation ratios are transaction-based, meaning that they depend on external factors beyond what the node can fully control. On the other hand, having excellent uptime on its own can grow the Uptime ratio to 2, which can allow the node's score to accumulate faster if a very high uptime is maintained.

A score above 1 provides benefits to a node, including increased confidence from the network, higher likelihood of selection for work assignments, and the potential to split into two licensed nodes when the score reaches 2. Although we are working on a cutting edge reputation system, the scoring scheme remains flexible, allowing for the adoption of improved methods to measure Uptime, Computation, and Reputation in the future.

3.5 Node Splitting

Licensed nodes in HyperCycle have the unique ability to undergo two types of splitting to expand their capabilities. The splitting capacity of HyperCycle licenses and HyPC native tokens relies on the splitting capacity of the Sato-Servers that instantiate them. Consequently, as each node undergoes splitting, the process follows the nested model of the Earth^{64} data structure, while also maintaining a human-readable prefix indicating the size of the node.

The first type of splitting begins with a Master Node license of AI Distributed Computation Power (AIDCP), denoted 10AIDCP_1024. Such a license can be split into 512 individual node licenses by its owner without any conditions, meaning that it can be split into its constituent licenses whether or not it has undergone a Provenance Marriage with sufficient HyPC tokens to activate it. More details on this splitting process can be found in Appendix A.1.

The second type of splitting allows individual licensed nodes to multiply in number. An individual license that has not split yet is denoted 1AIDCP_1024 or AIDCP_1024. To be able to split, the node must have undergone a Provenance Marriage with a 1024 HyPC identity token, because this splitting is tied to the value of the node quality score, which must reach 2 to allow a split. When this happens, the node can split into two AIDCP_512 nodes, each with 512 HyPC to back their identity.

The new child nodes created from the parent node start with a score of 1 and initially share identical genetics obtained from their parent's identity (married license and identity tokens), but they are decoupled from each other in operation. Consequently, their scores evolve independently, allowing one node to choose to split while the other may opt not to. However, if both child nodes achieve a score of 2 and decide to split, the total number of licensed nodes becomes four, all AIDCP_256 nodes. With each iteration of the individual node splitting process, the number of nodes can double.

The process of splitting a single node license can iterate up to 10 times, resulting in the generation of up to 1024 licensed nodes from the original one. At the end of 10 iterations, each of the nodes, denoted AIDCP_1, only needs 1 HyPC to back its identity. The Earth^{64} data structure guarantees that no further splitting is possible beyond this point.

Note that it follows from this design that the multiplication process does not require acquiring additional HyPC or licenses; nodes simply adhere to HyperCycle's adaptation of the Sato-Server splitting protocol accordingly. There are, however, planned variations of this protocol to allow expedited splitting.

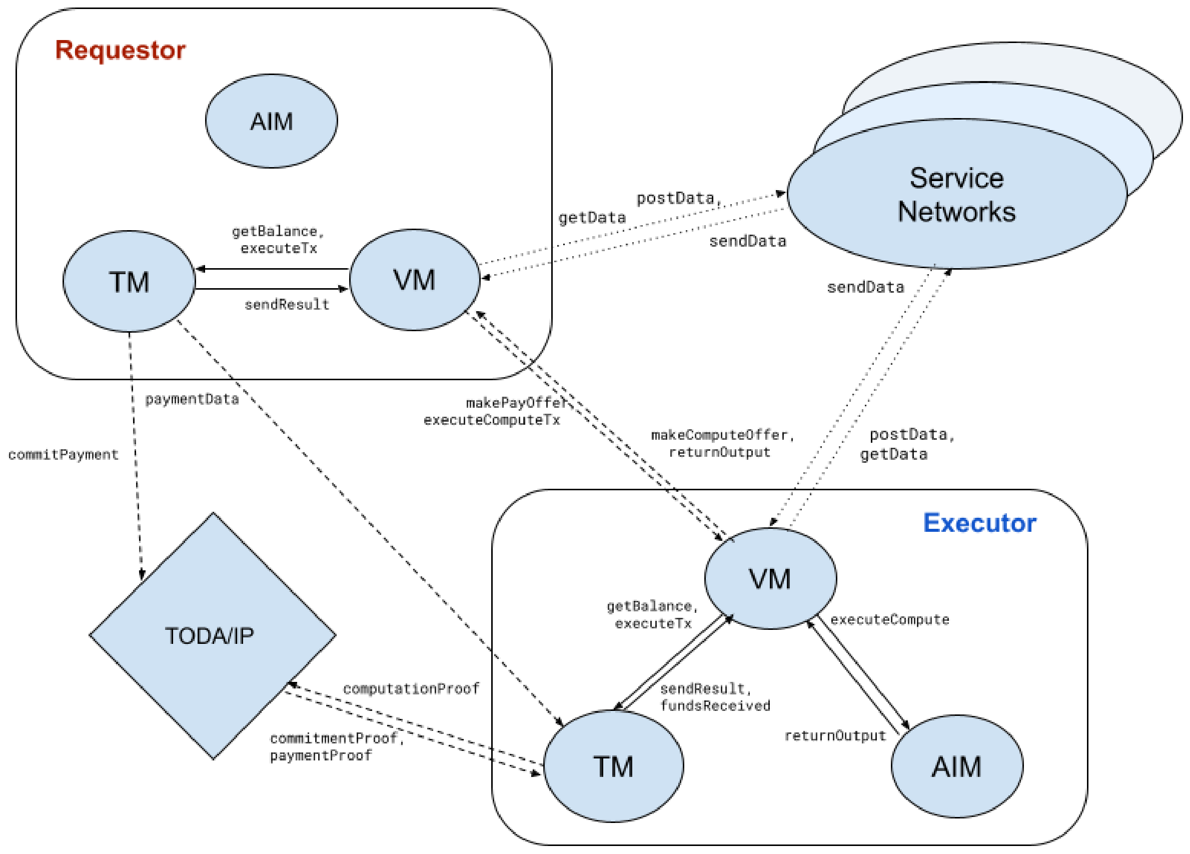
Despite having fewer coupled HyPC for their identities, the split individual nodes maintain the same capabilities as their parents. Node owners and operators are incentivized to provide high-quality service to the network, allowing their nodes' scores to consistently increase to 2 at approximately six-month intervals. By actively participating in the node splitting process, operators can participate in the wealth creation opportunities facilitated by HyperCycle.

It is important to appreciate that a single node can represent either a small computer or an entire HPC cluster. Growth beyond this point is indefinitely possible, but it may not follow the same genetic markers indicated in this project at this time. This limitation is influenced by factors such as deterministic/finite lookups, efficiencies, crypto-economics, incentives, relative utilities, and others. Ultimately, the growth of the network is bounded by the laws of physics (space, time) and the protocol needs to be in line with the full growth potential this entails.

4 HyperCycle Transactions

In this section we consider how the different components of an HCN—the VM, TM and AIM—work together to execute transactions in the HyperCycle network. In any such situation, no matter how complex a transaction is and how many layers of subcontracting it involves, the transaction can be broken down into pieces involving one node requesting work from another node, and sending it a payment commitment along with data and the instructions for how to process it.

The node requesting work is called the requestor and the node performing the work the executor, and any node can be a requestor or an executor or both. Suppose for simplicity, then, a simple transaction involving two parties. Let Alice be the requestor and Bob be the executor. The data flow diagram below illustrates at a high level the kinds of messages exchanged between different components to execute a transaction in the HyperCycle network.



We will focus on the core elements of a transaction, but communication with one or more service networks may be needed for Alice and Bob to find each other based on the services needed and offered and at what prices.

Following discovery, Alice and Bob negotiate their terms for the transaction, which can involve exchanging messages, such as what Alice is willing to pay (`makePayOffer`) and what Bob is willing to compute (`makeComputeOffer`). Once this has been worked out, the transaction can begin. Note that this negotiation step need not happen every time Alice wants Bob to do some work, as long as the assumptions on which the agreement is based are still valid.

To execute a transaction, Alice must verify she has sufficient funds to pay for Bob's work, so her VM asks her TM to tell her what her balance is (`getBalance`), and the TM replies (`sendResult`). If this passes, she can execute the transaction by having her VM instruct her TM to proceed (`executeTx`) and sending the instructions and data to Bob's VM (`executeComputeTx`). Her TM will send Bob's TM the payment data (`paymentData`) as a Sato-Server and will register on the TODA/IP network her commitment to pay Bob if he returns her the output (`commitPayment`).

On Bob's end, his VM receives the data and instructions from Alice, his TM receives the payment data from Alice, and his TM receives a proof from the TODA/IP network that Alice has committed payment to him provided that he returns her the output (`commitmentProof`). Upon receipt of the instructions and data, Bob's VM waits for a message from his TM (`fundsReceived`) indicating that the payment has been confirmed to be committed to him.

At this point Bob proceeds by sending the data and instructions to his AIM (`executeCompute`), who does the requested computation and returns the result to Bob's VM (`returnOutput`), which we assume includes an attestation that this is the correct output that Bob can use to satisfy the conditions for the release of payment.

At this point, Bob's VM returns the output to Alice and sends the attestation to his TM (`executeTx`) to release the payment. Bob's TM updates the state of the payment data with the proof that computation has completed, sending this to the TODA/IP network to register this fact (`computationProof`). At the end of the TODA/IP cycle (less than a second), Bob's TM receives back a proof from the TODA/IP network that the payment is complete (`paymentProof`). This completes the transaction.

Alice has received the requested output and Bob has received the payment. But note that Bob cannot spend the funds received from Alice until the end of the next network block, which takes 6-30 seconds.

5 Conclusion

HyperCycle Computation Nodes function as the core agents within the HyperCycle network. Their ability to rapidly contract and subcontract each other P2P in adaptive ways fosters cooperation and competition for beneficial emergence and progressive self-evolution. Their marriage of a HyperCycle license and the native HyPC token, along with value generation and proof mechanisms that result from the marriage, incentivize node operators to contribute to the network's quality of service, functionality, and security.

Beyond the core functionality, the HyperCycle ecosystem offers vast possibilities. SingularityNET envisions an AI marketplace where humans and machines can transact, advancing machine intelligence and paving the way for true AGI. Additionally, marketplaces for high-performance computing can leverage HyperCycle as a service, supporting advancements in various fields such as science, medicine, and technology. The ecosystem also incorporates practical services like pools and marketplaces that enable nodes to efficiently advertise and negotiate services.

With HyperCycle's decentralized interoperability built upon TODA/IP and Earth^{64}'s Sato-Servers, the potential for synergistic emergence is boundless. The diverse participants in the HyperCycle ecosystem can collaboratively build an internet of AI and high-performance computing, serving the betterment of humanity and the planet as a whole.

6 FAQ

Question: Does the node operator have to provide the license and identity tokens or can these be delegated to the operator?

> **Answer**: *The license and identity tokens can be owned by the node operator but can also be delegated to the node operator by others. License and c_HyPC pools can be used for this purpose, for example.*

Question: Does the splitting of a node into two child nodes mean that three nodes are now in operation?

> **Answer**: *No, the splitting of a node generally means that the parent node no longer exists. There are, however, deployment exceptions where the parent node becomes a worker for the children, but the parent nevertheless does not function as a node on the network.*

Question: When a node splits after having increased its score to 2, are more tokens required to run one of the new nodes?

> **Answer**: *No, this is the reason for needing 1024 HyPC to back the identity of a node. When a 1024 HyPC node splits into two, the two new nodes have 512 HyPC each backing their identity. Each time a node splits in this way, the new nodes have half the amount of HyPC backing their identity as the parent did, down to the level at which the nodes have a single HyPC backing their identity, at which point no further splits are possible.*

Question: Can multiple nodes be running on a single device?

> **Answer**: Yes, the protocol allows many nodes to run on a single device. There are physical limitations to how many nodes can run on a single device, in terms of processing power, memory, networking, energy consumption, etc., but the protocol places no such limitations on how many nodes per device.

Question: Can a 512 node package run on a single computer? And is 512 x 1024 = 524,288 HyPC needed to run this fully?

> **Answer**: As explained in the previous answer, yes, 512 nodes can run on a single node provided that the machine is powerful and efficient enough to do so.
>
> To run the full package with the full benefits of the measurement of uptime, computation and reputation, yes, each of the 512 nodes must be backed by 1024 HyPC. The node can operate in this way without splitting the package into individual nodes, allowing all of the nodes together to accumulate the same node score.

Question: How is proof of reputation of a node calculated and used in the network?

> **Answer**: Anton’s team is working on a cutting edge reputation system, on which there are published whitepapers. Whichever system is developed, it will take six months to increase a node’s reputation from 1 to $\sqrt[3]{2}$, which is the crucial factor in determining the multiplicative node score that determines whether a node can spit into two. The detailed reputation calculation itself, however, may be given

more weight by certain sections of the network for the purposes of building trust and faster transactions, but these features are independent of node splitting.

Question: How does a node know that another node has multiplied? And how does the reputation of the parents factor into the reputation of the children?

Answer: Every node that has split is the descendant of a node formed from the marriage of a single license and 1024 HyPC. When the node splits, it carries the genetics of the parent as cryptographic proof of the marriage and of the conception of the two child nodes, which any node can inspect to obtain a proof that a node has multiplied.

In terms of reputation, the parents will also have proof that they reached a score of 2 before multiplying, and therefore a reputation of $\sqrt[3]{2}$ before multiplying. So even though the new children only have a score and reputation of 1, it is known that their parents were highly reliable nodes, which can potentially be utilized in some sections of the network to give higher trust to the new children.

Question: What are HyperCycle's sources of revenue?

Answer: HyperCycle has two sources of revenue. One is from the sale of software licenses and tokens, which are intended to grow the company. The second is the 1% royalty on all node revenue that is collected to incentivize all parties in the ecosystem including HyperCycle. The royalty system is a sunset of payments, to yield more progressive evolution of the ecosystem as a whole rather than purely to generate profit for HyperCycle.

Question: Is the use of HyPC to back the identity of a node scalable with a large rise in the token price?

Answer: For the identity tokens, the rise in price of HyPC increases the cost to set up a node in proportion to the increase in HyPC price. For large increases in price this can still be economical for large centralized operations relying on a small number of nodes, but is less so for those requiring a large number of nodes. In either case, operations do benefit from being early adopters of the ecosystem.

A Appendix

A.1 Various Kinds and Forms of HyperCycle Tokens

The different types of tokens and their representations require clarification to avoid confusion. In this context, it is important to understand that true node licenses and native tokens are Sato-Servers, and as such exist independently of any blockchain. However, for convenience and network functionality, these Sato-Servers have NFT token representations on various blockchains.

The node licenses available on ledgers like ADA, MATIC, and ETH are merely NFTs that point to a specific license using the globally unique binary address of the corresponding Sato-Server. Mastern Node licenses, with the prefix `10AIDCP_1024`, represent Sato-Servers containing a package of 512 node licenses. These Master Node licenses can be divided down 9 iterations (9 levels of the Earth^{64} tree), resulting in individual node `1AIDCP_1024` licenses. Further splits occur when the quality score of such a node reaches 2, generating two `1AIDCP_512` licenses. This process can iterate up to 10 times, ultimately reaching `1AIDCP_1` licenses at the bottom of the Earth^{64} tree, which cannot be split further.

Note that the identifiers for the licenses on chain also contain a suffix that gives the integer equivalent of the binary Sato-Server address that instantiates the license. For example, the Master node with identifier

```
10AIDCP_1024.17593259786358
```

has `17593259786358` as the address of its Sato-Server. This integer determines the license's precise location in the Earth^{64} binary tree and its unique geographical location on Earth's surface.

There is a similar divide between on-chain representations of the native token and their true form as Sato-Servers. Fungible HyPC.X tokens and non-fungible c_HyPC.X tokens represent the native token on-chain. c_HyPC.X tokens directly correspond to specific Sato-Servers based on their globally unique binary Sato-Server address, making them the true on-chain representation of the native token. The fungible HyPC.X tokens serve only as a convenience token. The symbol "HyPC", without the ".X", is reserved for the native token Sato-Servers themselves and the unit of value they carry.

Each HyPC token carries a specific value determined by its location in the Earth^{64} tree. As such, the value of each HyPC token Sato-Server is always a power of 2 HyPC, ranging from 2^{-18} to 2^{19}. Thus, the smallest denomination of HyPC is about 4 millionths of an HyPC. This partially satisfies the 6 decimal points in the current pointers of HyPC.ETH, HyPC.BNB, HyPC.ADA, HyPC.ALGO, HyPC.AVA and HyPC.MATIC. Because c_HyPC.X tokens directly point to HyPC tokens, their value is also restricted to a quantity of HyPC that is a power of 2. As of this writing, only c_HyPC.X tokens with 2^{19} = 524,288 HyPC are available, though smaller denominations and a limited on-chain splitting ability of these tokens are planned.

It is worth noting that the “c” in “c_HyPC" stands for “containerized”. This is because the HyPC Sato-Servers not only carry a certain value in HyPC based on their location in the Earth^{64} tree but also are containers that can hold other tokens.

A.2 Provenance System

A key part of the technical side of HyperCycle is the provenance system that provides rigorous, cryptographic provenance for the native token and the licenses.

As described in the previous subsection, there are pointers to the node licenses and native token available for purchase and management in limited ways on chain. The node license pointers are NFTs and the native tokens have fungible and non-fungible versions. As of the time of writing the fungible native token pointers are hosted on Ethereum (ETH) and Binance (BNB), while the non-fungible native token pointers are only available on Ethereum; the license pointers are available on Cardano (ADA). In the near future licenses will also be available on Polygon (MATIC). Availability of fungible token pointers will be extended to Polygon, Cadano and Bitcoin (BTC).

The licenses available on chain at the time of writing are only the Master Node `10AIDCP_1024` licenses, which are a package of 512 single node `1AIDCP_1024` licenses. Smaller license packages and limited on-chain splitting of license pointers is planned. The c_HyPC.ETH tokens available are NFTs representing packages of

$$512 \times 1024 = 524{,}288 = 2^{19} \text{ HyPC}$$

which is the amount of HyPC required to back the identity for `10AIDCP_1024` licenses in a Provenance Marriage. Smaller value c_HyPC.X tokens and limited on-chain splitting of native token pointers are also planned.

The fungible form of the native HyperCycle token was initially released on Ethereum (ETH) as an ERC20 token with ticker symbol HyPC, which we denote by HyPC.ETH. There is also a Binance version HyPC.BNB available. To use this HyPC in the HyperCycle network it first needs to be converted into its non-fungible, on-chain c_HyPC form, which can be done on Ethereum using a swap contract we have developed. This contract ensures that the amount of HyPC.X in circulation (either in fungible or non-fungible form) is always fixed by locking up units of HyPC.ETH in the contract to make a c_HyPC.ETH available. The c_HyPC.ETH can later be redeemed, returning ownership of the c_HyPC.ETH to the contract and releasing the locked HyPC.ETH.

In terms of *proofs* of provenance, the provenance system relies on the proofs of provenance provided by the various blockchains involved and as well as the Earth^{64} Sato-Server data structure and TODA/IP. Provable provenance of licenses and native tokens in this context is ensured by the cryptographic properties of the Sato-Server data structure, which ensures that each state update of an asset is hash-chained to the previous ones and that these state changes are unique (no double spend). This latter capability is ensured by incorporating into Sato-Servers elements of the TODA/FP (TODA File Protocol), developed by TODAQ and T.R.I.E., to allow the integrity (single spend) of assets to be maintained by an external integrity provider without the integrity provider needing to know the state of the asset (so-called integrity-at-a-distance).

The integrity provider for the Sato-Servers utilized in HyperCycle is TODA/IP, which ensures that efficient, secure distributed consensus is reached on the state of all assets it maintains globally, without needing to know the asset state (ledgerless). In this way, Sato-Servers, TODA/FP and TODA/IP all work harmoniously together to provide efficient and secure proofs of provenance for licenses and native tokens in the HyperCycle system.

Note that once HyPC tokens and licenses are converted to their Sato-Servers they can be held directly on the device that is operating a node. Moreover, the HyPC used in transactions can be exchanged directly between nodes P2P at the network level, without involving a third party in the transaction.

We have covered the on-chain provenance and the Sato-Server provenance. The only missing piece is the transition between the two. This transition from the on-chain representation of a license or c_HyPC.ETH token to its Sato-Server is managed by Earth^{64} to ensure this process happens with integrity. This process relies on the same method that Sato-Servers in general are converted from an on-chain form to Sato-Server data form.

A.3 HCN Virtual Machine

The heart of the HCN software is the virtual machine (VM), which provides programmable functionality for executing a wide range of transactions involving different arrangements of nodes. The VM also provides seamless communications between internal components within a node and external connections with other nodes to negotiate and execute contracts.

Internally to a node, the VM is responsible for communicating with the TM to ensure that the appropriate conditions are met to send or receive payments and that sufficient proof can be provided to the TODA/IP network to release payment after computational work has been done. The VM also communicates with the AIM, to execute computational work, to determine based on responses from the AIM whether subcontracting work to other nodes needs to occur, and to collect attestations of computational work to construct the proofs needed by the TM.

Externally, the VM is responsible for a range of different communications. This includes publishing the node's service capacities and prices, and discovering those of other nodes. This can happen P2P between nodes but may also make use of secondary networks. It also includes conducting negotiations with other nodes to establish the service requirements for some kind of computation that a network of nodes wants to participate in. This is essential to establish reliable subcontracting between nodes to ensure that a larger computation can be completed for a given user or user group. And the VM must also conduct the actual transactions between nodes, by transferring instructions and data between nodes, while ensuring that the TMs in question send and receive the required payment information and proofs.

High-performance computation is paramount, and zero knowledge remains prohibitively expensive, so we attest to the correct execution of the VM and AIM via the secure enclave technology available on some modern CPUs and GPUs. This technology supports native-speed execution on encrypted data, at the cost of a chain of trust going back to the manufacturer. Going forward, our own and other recent research shows the way towards leveraging hardware and zero-knowledge for trustless security with low impact on performance.

A.4 HCN Transaction Machines

The HCN software incorporates one or more Transaction Machines (TMs) that establish their own connectivity to a transaction network, while being loosely coupled or decoupled from other components within the node. A TM facilitates interactions with various network components, including the TMs of other nodes and network protocols such as the TODA/IP heartbeat. Notably, the management of the transaction network operates independently of the communications within and between nodes.

Within an HCN, a TM serves as a critical interface for the VM. It enables the VM to send payment commitments to the TM for validation and allows the VM to construct Sato-Server state update data. The TM can then package and sign this data. These functionalities empower the VM to negotiate agreements with networked nodes and ensure the appropriate establishment of payment contracts for requested work.

The TM, in conjunction with Sato-Servers and TODA/IP, ensures secure and efficient P2P payments within the HyperCycle network. By leveraging Sato-Servers and the decentralized consensus protocol of TODA/IP, the system offers flexibility in facilitating payments, supporting the exchange of various currencies within HyPC Sato-Servers while upholding robust security and incurring minimal costs.

HyperCycle nodes will be able to secure their transactions by using a TM kitted with a zero-knowledge EVM to handle communications and tokens on Ethereum-compatible blockchains.

The TMs of a node will not be addressable directly to prevent DDoS attacks. Consequently, addressing will make use of the node license ID (Sato-Server address), which must then be appropriately resolved into a URL by the payment network.

A.5 HCN AI Machine

The Artificial Intelligence Machine (AIM) acts as a versatile interface responsible for managing and executing AI modules for computational tasks. These AI modules are contained within separate containers, providing modularity and flexibility to the system. The containers can use Docker or other containerized software platforms. The AIM will seamlessly integrate with the VM to receive instructions and data for processing and return outputs, despite the fact that the VM and AIM are loosely coupled by design.

The AIM has the capability to dynamically switch between different modes, including "inference mode" and "training mode." In inference mode, the AIM will process instructions and data using pre-packaged containers, while in training mode, it will orchestrate ad hoc clusters of machines for intensive computations over extended periods. To maximize performance, the AIM, under the control of the VM, will leverage available networking hardware, such as RDMA/Infiniband connections and DPUs.

As part of the larger ecosystem, beyond the core, AI module marketplaces are already being established by ecosystem partners such as SingularityNET. Node operators will have the ability to download software containers from these marketplaces and install them on their nodes. Once installed, operators can advertise their AI microservices availability within the network, specifying the offered services and corresponding pricing. These marketplaces may take various forms, including centralized sites or distributed P2P marketplaces based on distributed hash tables. We emphasize that the marketplace infrastructure operates independently from the core HyperCycle system, allowing for progressive enhancements and improved services over time.

A.6 HCN Node Manager

The Node Manager (NM) plays a vital role in the node software, serving as a central component for effective node management. Its primary purpose is to oversee the software installed on a node, including the configurations, versions, and options of the VM, TM and AIM. This ensures seamless operation and optimal performance of the node. The NM can be configured to allow secure remote system management.

The NM will incorporate an administration panel that provides a user-friendly interface for node operators. This administration panel will serve as a centralized hub for managing and configuring various aspects of the node. It will empower operators with convenient tools to streamline administrative tasks, monitor performance, and customize settings according to their requirements.

A.7 HCN Merklizer

As part of the Tilling process, the node software includes a critical component called the Merklizer, designed to uphold the integrity of the node's state records over time. The Merklizer performs periodic tasks, following the system 6-30 second block production time, by generating a record of the node’s machine state. This local machine state record, represented as a Merkle root hash, is then shared with the network. The local state data and its inclusion in the published hash form the basis for proofs related to uptime, computation, and reputation, crucial for maintaining a highly secure and reliable network.

As these local records on each node need to be maintained for the whole network, a service to ensure the integrity of all of these records is required. This is the function of the Merkle Service, which consolidates all of the hashes published by each node by collecting all the local hashes and combining them into a single global Merkle root hash, incorporating and summarizing the entire network state in each block’s time window. The Merkle Service communicates this global root hash to every node in the network, along with concise Merkle proofs demonstrating the inclusion of each active node in the global root hash.

The Merklizer operates by aggregating data from various components of the node, including the VM, TM, AIM and NM. It combines this data in a structured manner, facilitating future verification of the inclusion of specific data for each component in a given local Merkle root hash for a particular cycle. To ensure the ability to give such proofs when needed in the future, the Merklizer also maintains a comprehensive database of historical state and Merkle proof data.

In the ultimate design, the Merkle Service will be run as a decentralized P2P networking protocol, utilizing TODA/IP for efficient and secure distributed consensus on the global root hash. This decentralized approach ensures scalability as the network expands while maintaining a consistent global block time.

It is crucial to understand that the global block time refers to the full immutability and finality of transactions at the network level, which determines the re-spendability of tokens. This is separate from the cycles and threads leading up to the blocks. Transactions achieve immediate irrefutability, while the localized record that establishes their immutability and finality is published at the end of each transaction cycle, which happens nearly instantaneously. Many such cycles are folded into a single global block, which when published finalizes for the network as a whole all the transactions contained within the block, confirming the re-spendability of all the tokens transacted.

The envisioned design emphasizes the Merklizer's pivotal role in maintaining the network's state records and ensuring the integrity of transactions. By leveraging the capabilities of the Merkle Service, the design provides a robust and reliable infrastructure for the network's operation. While all this part of the Tilling process, it is crucial to understand the necessity of structuring the data as time goes and not after the fact as in order to gain certain security elements the temporal coupling amongst other aspects could not be done after the fact, those are events that as they happen they leave traces into the network and that structure that theTilling provides will be crucial in further securing the network as time passes.

Glossary

agent a HyperCycle Computation Node that can act autonomously to negotiate contracts for computational work with other agents per computational task using micropayments

AIDCP AI Distributed Computation Power • a measure of the size of a node license package that can range from 10xAIDCP (512 license package) to 1xAIDCP (single node license)

AIM AI Machine • a component of a HyperCycle Computation Node that is responsible for taking instructions and data from the VM, processing the data appropriately, and returning the result to the VM

Computation as a capitalized term refers to the ratio that measures a node's performance in terms of its consistency and reliability for providing computation services

container a term used variously to refer to a Sato-Server container, that is capable of holding value on its own and containing other forms of value, and to refer to a Docker container, a containerized form of software application or microservice that incorporates its software dependencies into a single package

c_HyPC.X on-chain pointer to a fixed amount of native HyPC • an NFT on blockchain X, e.g. ETH, that corresponds directly to a native HyPC Sato-Server token and contains the unique Sato-Server address of the native token it points to

Earth^{64} a finite data structure that combines a mathematically nested and unique global numbering system with physical relatability to humans, capable of holding existing and emergent values, such as money and AI computation (see the Earth^{64} whitepaper)

executor any HCN in its role as a performer of computational work, taking instructions, data and payment from a requestor

HCN see HyperCycle Computation Node

HyPC.X on-chain pointer to a variable amount of native HyPC • a fungible token on blockchain X, e.g. ETH, which provides a convenient way of acquiring and managing units of the native token that can be converted into the native token via c_HyPC.X

HyPC the native HyperCycle network token • refers to both the units of the native token and the actual native token Sato-Server, which carries a certain value (power of 2) in HyPC units and can contain other value instrument tokens inside it

HyperCycle Computation Node the core element of the HyperCycle network, responsible for requesting and executing computational work for other nodes in the network

identity token the HyPC token needed to back the identity of a HyperCycle license, forming a node's identity in a Provenance Marriage

Master Node A 10AIDCP license package containing 512 HyperCycle Node licenses

Merkle Service a service operated by the HyperCycle network that is responsible for maintaining a ledgerless, network-wide record of the state of the network that enables individual nodes to prove their uptime, computation and reputation to the network

Merklizer a component of a HyperCycle Computation Node that is responsible for computing a cryptographic record of the local machine state for each cycle of its operation that can be sent to the Merkle Service for inclusion in the global network state record

native token see HyPC

neurode another term for a HyperCycle Computation Node, connoting its role in the network of interconnected artificial neurons that form the HyperCycle global brain.

NM Node Manager • a component of a HyperCycle Computation Node that is responsible for the management (installation, configuration, updates, uninstallation, etc.) of the components of a node

node short form for a HyperCycle Computation Node when this is clear from context

node license variously refers to the on-chain representation of a license to operate one or more HyperCycle Computation Nodes and to the actual Sato-Server licenses for the same; the conventional name for the Master Node license has its unique global number embedded in it (e.g. 10AIDCP_1024.17593259786358) and as it divides to single licenses it follows the Earth[64] data structure protocol which ensures they are nested

node operator the individual that controls the ability to configure a node's components and modes of operation

node score a measure of a node's quality computed as the product of three performance ratios for uptime, computation and reputation

Provenance Marriage A combining of an HCN license and a certain quantity of HyPC to form the identity of a node, allowing the node to acquire a node quality score and single node licenses to multiply; e.g., a 512 node 10AIDCP_1024 license package is married with 524,288

HyPC, a single AIDCP_1024 node is married with 1024 HyPC, and a AIDCP_64 node is married with 64 HyPC

ratio a scaled proportion measuring the quality of service of a node's uptime, computation or reputation

Reputation as a capitalized term refers to the ratio that measures a node's performance in terms of its reputation on the network

requestor any HCN in its role as a requestor of computational work, sending instructions, data and payment to an executor

Sato-Server a data structure and associated protocols provided by Earth^{64} that enables the state of assets to be held in a transmissible container that can divide into smaller parts and change the system responsible for maintaining its integrity (non-equivocation or single spend) over time

Sato Tree another name for the binary tree of Sato-Servers defined by the Earth^{64} data structure; the binary tree is also a binary trie where the keys are the binary addresses and the values are Sato-Servers

SingularityNet A project enabling any AI developer to monetize their code; this can have several deeper integration points in HyperCycle, such as where initially the code runs in the AIM, and subsequently the AGIX token can be carried out and managed inside the HyPC native token

Tilling A concept borrowed from the Earth64 Data structure which is the effort consumed by machines during AI computation and ahead of AI computation in preparation of. A Machine may start by using the time that passes to get to know the network and does some work in securing it via temporal coupling etc. This will help build the ratio needed for the node to split. For more on Tilling (see the Earth^{64} whitepaper)

TM Transaction Machine • a component of a HyperCycle Computation Node responsible for validating and executing payment transactions between nodes on the HyperCycle network

TODA/FP TODA File Protocol • a data structure and associated protocols provided by T.R.I.E. and TODAQ that enables the state of an asset to be maintained independently of its integrity (non-equivocation or single spend)

TODA/IP TODA Internet Protocol • a data structure and communications protocol for distributed consensus across P2P networks of arbitrary size; TODA/IP utilizes a distributed proof-of-work that minimizes the cost of securing the network and maximizes transaction throughput while ensuring the cost scales linearly with network size, becoming more secure the larger the network

Uptime as a capitalized term refers to the ratio that measures a node's performance in terms of its availability on the network

VM Virtual Machine • a component of a HyperCycle Computation Node responsible for enabling programmable ability for nodes to form ad hoc networks to solve computational problems and securely set up the requisite contracts between the nodes, as well as to request work, execute work and subcontract work to other nodes as required