Hyperspace: A Peer-to-Peer Artificial Intelligence Network

HyperspaceAI (gutenberg@hyperspace.us)

<Nov 19th 2023>

Contents

Abstract

In an age of unprecedented technological growth, the US has recently introduced an executive order that seeks to regulate large language models and the wider realm of artificial intelligence (AI). While the intentions may be to maintain security and national interests, this policy document inadvertently poses significant constraints on innovation, with far-reaching implications that spill over to the entirety of the digital world. As the definition of AI gets stretched to its broadest form, virtually any online recommendation system – from Google's search to Yelp's restaurant suggestions – falls under the purview of these new regulations. The brunt of this is felt the hardest by open-source projects, which face severe constraints, particularly those with models exceeding 10B parameters. This restrictive atmosphere threatens to stifle the smaller players while benefiting those with the resources to navigate the intricate compliance landscape – a classic example of regulatory capture.

Moreover, the policy demands an unprecedented level of surveillance on the location and compute of even decentralized networks, heralding what might be termed as the dawn of the "AI-industrial-complex". The added bureaucratic layers not only increase costs for companies but also unjustly treat models requiring significant computational power as potential threats.

In response to these stringent policies, we introduce "HyperspaceAI", an open standard protocol designed for distributed model inference. As staunch advocates of decentralization, we are committed to promote genuine use-cases for trustless protocols. The current regulatory landscape presents a compelling case where blockchains and decentralized systems could potentially supersede "yet to be established" regulations. HyperspaceAI champions the cause of decentralization, ensuring that the power of AI remains widespread and doesn't consolidate among a select few. Through HyperspaceAI, we offer a glimpse of a future where AI is democratized, without the constraints imposed by misguided policy decisions.

1 BACKGROUND

Going through important properties of AI that plays central roles in the protocol.

1.1 Distributed Hash Tables

Distributed Hash Tables (DHTs) are foundational components in decentralized systems, enabling the storage and retrieval of data across a distributed set of nodes without relying on a central authority.

1.1.1 Kademlia DHT

Kademlia (Maymounkov, Petar and Mazières, David, 2002) is a distributed hash table (DHT) protocol that provides a decentralized way of storing and retrieving data across a network of peers.

- Distributed Hash Table (DHT): Enables decentralized storage and retrieval of data.
- XOR Metric: Used as a distance measure for efficient routing.
- Node Lookup: Nodes locate each other based on their IDs for low-latency data retrieval.
- Bucket Lists: Nodes maintain lists of peers in "buckets" based on distance.
- P2P Applications: Backbone for many peer-to-peer networks, including BitTorrent's DHT system.

1.1.2 S/Kademlia

S/Kademlia (Baumgart, Ingmar and Mies, Sebastian, 2007) is an extension of the Kademlia protocol, designed to offer enhanced security features.

- Enhanced Security: Specifically designed to improve upon Kademlia's resistance to Sybil and Eclipse attacks.
- Node ID Generation: Incorporates cryptographic puzzles to make ID generation computationally expensive.
- Public Key Infrastructure: Every node has a pair of public and private keys, making identity spoofing difficult.
- Node Lookup: Ensures nodes interact with trustworthy peers by using signed messages.
- Refresh Mechanism: Regularly changes node IDs to counter long-term Sybil attacks.
- Bucket Refreshment: Mandates regular bucket updates to guard against Eclipse attacks.

1.1.3 Suzaku and Kirin

In our study, we have been deeply impressed by the advancements in Distributed Doubly Linked Lists and the Suzaku Distributed Hash Table (DHT) concerning CHURN resiliency. Notably, the ability to achieve liveness without necessitating locking, all while preserving the integrity and consistency of the DHT, stands out as a significant achievement.

- Novel key-order preserving structured overlay network designed for efficient range queries.
- Unlike Chord# (Stoica, Ion and Morris, Robert and Liben-Nowell, David and Karger, David R. and Kaashoek, M. Frans and Dabek, Frank and Balakrishnan, Hari, 2003), Suzaku features a bi-directional finger table, achieving $\lceil \log_2 n \rceil - 1$ maximum lookup hops when converged.
- Suzaku's algorithms handle node insertion and deletion, maintaining strong lookup performance even amidst network changes (churn).
- Simulation evaluations showed Suzaku outperforming conventional networks such as Chord# and Skip Graph.

1.2 Equilibrium of Computation Exchange

We're looking into defining a protocol that relies on global computational collaboration, that means that we need to give valid incentive to all the participents of the network to share and exchange their computation power.

Going a BitTorrent-like route we'd like to quote (Cohen, Bram, 2003) **Incentive builds robustness**

In the realm of distributed inference, we are endeavoring to formulate a protocol contingent upon collaborative global computation. This necessitates the establishment of cogent incentives to galvanize all network participants to proactively allocate and exchange their computational resources.

Invoking a paradigm analogous to BitTorrent, we reference Cohen's assertion in (Cohen, Bram, 2003): "Incentive builds robustness."

There exists a potential avenue to explore a reciprocity-based equilibrium, reminiscent of BitTorrent's "tit-for-tat" strategy. Alternatively, we are contemplating a framework where computational resources are furnished at a legitimate cost, as dynamically determined by the network. This approach demands intricate modelization and predictive analysis, aligning with the computational characteristics of HyperspaceAI. More on that.

2 Hyperspace Protocol Design

The model presupposes an asynchronous distributed system. Within this system, nodes are interconnected through a nodes called Hyperspace Community Server (HCS). It's acknowledged that this network might exhibit certain inefficiencies: it might not deliver messages, could delay, duplicate, or even rearrange them.

2.1 Identity

Entities within the system, referred to hereafter as nodes, are uniquely recognized by an identifier termed the NodeAddress. This **NodeAddress** is not merely a manifestation of a node's public key; rather, it represents the cryptographic hash of such public-key. The rationale for adopting the cryptographic hash in lieu of the raw public-key stems from specific security concerns intrinsic to decentralized systems, notably the mitigation of both sybil and eclipse attacks, especially when operating in the absence of a centralized, trustworthy authority.

2.1.1 Address Assignment

The employment of a cryptographic puzzle, particularly the Proof of Work (PoW) mechanism, serves to fortify the network against the aforementioned attacks. However, the use of such crypto puzzles as identity confirmations is a contentious topic in the academic realm. For instance, Castro et al. in their seminal work in 2002 (Castro, Miguel and Druschel, Peter and Ganesh, Ayalvadi and Rowstron, Antony and Wallach, Dan S., 2002) contended against the utilization of crypto puzzles for identity verification. Their reservations emanated from two primary reasons: firstly, the inherent inability of crypto puzzles to entirely obviate the risk of an attack, and secondly, the computational overhead they introduce. In this context, "overhead" pertains to the computational cost associated with resolving a crypto puzzle. For optimal functionality, this cost needs to be manageable for the least performant, legitimate node. Concurrently, the puzzle's complexity should be robust enough to deter or considerably decelerate an adversary equipped with high computational resources.

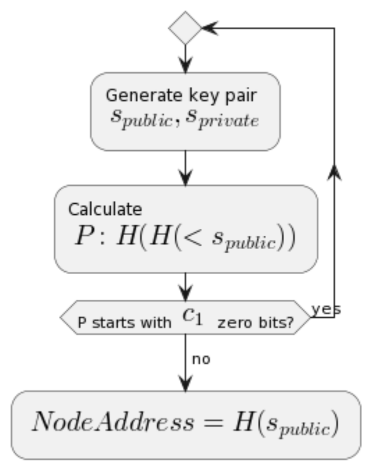

Figure 1: Static node adress generation crypto challenge

Notwithstanding the above concerns, our position aligns with the perspective that, in environments devoid of centralized trust entities, crypto puzzles emerge as the most pragmatic approach for the generation of distributed node IDs. This is due to their potential in amplifying the difficulty level for potential adversaries aiming to compromise the network. In essence, for networks operating in entirely decentralized milieus, leveraging cryptographic techniques to maximize attack resilience becomes not just preferable, but imperative.

2.1.2 Message Signing

In advocating for the utilization of a hash over a public key to generate the nodeId, we emphasize the ability to employ this public key for signing messages exchanged among nodes. Given computational constraints, we classify message signatures into two distinct types:

- Weak signature: The weak signature does not encapsulate the entirety of the message in its signature. Instead, it confines its scope to the IP address, port, and an associated timestamp, which delineates the signature's validity duration. This design counters replay attacks if dynamic IPs are used. To accommodate synchronization variances, timestamps might be designed with a broad granularity. The weak signature is used where complete message integrity is not deemed critical (PING messages for example).

- Strong Signature: Contrary to the weak signature, the strong variant signs the full message content, ensuring the message's integrity and bolstering defenses against potential Man-in-the-Middle attacks. To thwart replay attacks, nonces are incorporated within the RPC messages.

2.2 Model

We analyze a system with a predetermined set n nodes, denoted by $i \in [n]$ where $[n] = \{1, \ldots, n\}$. A subset $F \subseteq [n]$ contains up to $f = |F|$ nodes that exhibit Byzantine faults, while the others are considered correct. These Byzantine nodes are frequently alluded to as being under the control of an adversary, who is privy to all the internal states of these replicas, including their cryptographic keys and any prior data.

The network receives generative and prompt execution queries from the application and returns the execution result. Given an upper bound f of malicious nodes for a particular model, the network offers reliable execution in the following sense:

- If a nodes obtains the same result after k redundant inferences, that result is correct with probability $1 - f^k$

- If a nodes obtains different results, he can claim a portion of collaterals from one of the two nodes

2.2.1 Threshold Signature

All replicas possess a unified public key. Each of the n nodes retains a unique private key. The i -th node contributes a partial signature $\rho_i \leftarrow \mathrm{tsign}_i(m)$ on a given message m.

Partial signatures, represented as $\{\rho_i\}$ with $i \in I$ and $|I| = k$, where each $\rho_i \leftarrow \mathrm{tsign}_i(m)$, can be aggregated to yield a digital signature $\sigma \leftarrow \mathrm{tcombine}(m, \rho_i)$ on m. Any replica can authenticate this signature employing the public key through the function tverify. It is a necessary condition that if $\rho_i \leftarrow \mathrm{tsign}_i(m)$ for every $i \in I$ (with $|I| = k$), and $\sigma \leftarrow \mathrm{tcombine}(m, \rho_i)$, then $\mathrm{tverify}(m, \sigma)$ confirms as true.

2.2.2 Hash Functions

We further uphold the primitives of the hash function previously discussed, emphasizing its collision resistance.

2.2.3 Hyperspace community servers and Hyperspace inference nodes

In the ecosystem, Hyperspace Community Servers (HCS) play act as orchestrators, oracles, and sequencers within the protocol framework.

Hyperspace Inference Nodes (HIN), in this context, establish connections with the HCSs, choosing them based on the node operator choice. They communicate their computational capabilities and the range of models they can execute. The initiation process involves sending a *joinmessage*, which includes the sender's address. This message is weakly signed, a measure to prevent forgery and ensure that it cannot be falsely associated with a different node identity (see 2.1.2). During the signature verification process, the HCS examines the cryptographic puzzle associated with the inference node (HIN), confirming its validity. The HCS node then informs the HIN about the dynamic complexity c_2 of the cryptographic puzzle, as outlined in Section 2.1.1. This puzzle serves as the identity of the HIN under the HCS sub-network.

Subsequently, Hyperspace Inference nodes make a secondary communication, the *registrymessage*, to the HCS node. This message details the Inference node claimed specifications and the AI models it is prepared to support and run inference on.

Following this registration, the HCS node issues an *inferencefortification* challenge to the Inference Node (HIN). This challenge is in the form of a prompt puzzle, the nature of which is determined at the HCS node's discretion. The HIN is then required to complete the puzzle and deliver the inference results through a verify-inference call.

This verification process includes a network gas fee mechanism. If an HIN inference is successfully challenged by another node in the network, the fee initially submitted is forfeited and transferred to the challenger. This system forms part of a broader challenge fraud proof mechanism, ensuring the integrity and reliability of the decentralized AI system's operations.

2.2.4 Fraud Proof

If a client receives two different replies or a clearly suspicious one, he can commit a fraud claim to the blockchain to receive compensation. When a fraud proof is submitted, other nodes can compute the query and verify integrity of output. If output is corrupt, the node submits an on-chain challenge.

The challenge is a synchronous process that happens on-chain and monitored by an on-chain smart-contract only requiring the LLM hash. Every node has a time-window to submit his answer to the other node until the challenge is done.

2.2.5 Challenge Model

After a challenge is committed, the challenged node must submit intermediary state roots. The challenger replies by determining the first corrupt state root and challenges it. The challenged node then submits intermediary state roots between the challenged state root and the one before it. The process continues until execution is narrowed down to one transaction, which is settled on-chain. The total number of submitted messages during the challenge is logarithmic in the number of execution steps.

When the challenge ends, the challenged node gets slashed if the execution is correct, otherwise the challenger is slashed. The honest node, as well as the client who submitted the fraud claim, receive compensation from the slashed node's collateral. Note that all nodes must be synchronized with the blockchain at all times.

In the case of neural nets, some parallelism optimisations may make the cost of defining a deterministic state root sequences heavy on computation. As a proof of concept, we will explain in detail how the process works for a simple feed-forward net.

A feed-forward net execution has every neuron receive the sum of incoming weights times the values of the neurons before, applies the activation function and stores it as its value. The value of the neuron is generally always stored in the RAM, and hence can be part of the Merkle state.

Take a typical neural network with 10 million neurons, and suppose they have some order. The challenged node must submit the state root of the neurons 1 million by 1 million, resulting in 10 Merkle roots. The challenger then computes locally the state roots and submits the first that is incorrect as a challenge. There must be an incorrect one since the last one is incorrect.

The challenged node sends the state root of the 1 million neurons challenged by packets of 100 thousands. The challenger does the same procedure until it is left to one neuron. We will also have the correct state of the neurons that precede it in the root just before. The process will take 7 iterations.

The challenger and challenged nodes then bring up the list of incoming weights and parse it in 10. The challenged nodes serializes the parallel weighted sum into 10 packets of parallel weight sums and posts the state roots on-chain. Similarly, the challenger nodes challenges the first incorrect computation, until it boils down to one single link. Again, we will have a correct state root of the sum of all incoming links prior to the specific challenged link, and a contested final link. The on-chain proof will consist of a simple sum and activation function verification, authentified by the Merkle root of the link stored in the LLMs hash. This procedure is logarithmic in the number of outgoing links.

This is just to reap benefits of parallelism. Otherwise, we could just make the weights computed serially and challenged consecutively.

2.2.6 Assumptions

- The weight matrix is stored in a way that allows parsing (for optimizing the computation process)
- Execution is deterministic
- The activation functions can be understood by the execution environment

2.3 Economics

In the aforementioned mechanims, all constituent entities are driven to operate with integrity due to the intrinsic economic structure and the incentivization mechanism. Customarily, emerging blockchain ecosystems introduce distinct tokens to facilitate this cryptoeconomic protection. Nonetheless, these nascent tokens may initially struggle to amass the requisite volume and distribution, thereby impeding the secure foundation of the ecosystem. This conundrum was adeptly addressed by EigenLayer, which devised a framework to harness Ethereum's cryptoeconomic safeguards by involving Ethereum validators. The HyperspaceAI Protocol assimilates this framework and employs EigenLayer operators to amplify security within the HyperspaceAI Network.

3 Reference

Baumgart, Ingmar and Mies, Sebastian (2007). *S/kademlia: A practicable approach towards secure key-based routing*, IEEE.

Castro, Miguel and Druschel, Peter and Ganesh, Ayalvadi and Rowstron, Antony and Wallach, Dan S. (2002). *Secure routing for structured peer-to-peer overlay networks.*

Cohen, Bram (2003). *Incentives build robustness in BitTorrent*, Berkeley, CA, USA.

Maymounkov, Petar and Mazières, David (2002). *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*, Springer Berlin Heidelberg.

Stoica, Ion and Morris, Robert and Liben-Nowell, David and Karger, David R. and Kaashoek, M. Frans and Dabek, Frank and Balakrishnan, Hari (2003). *Chord: a scalable peer-to-peer lookup protocol for internet applications.*