The NEX Network: A new architecture for a fully AI integrated massively scalable and secure web4 DLT v0.0.1

Jonathan Catalano Macpherson-Gray
0n@thenex.world

Abstract

This paper proposes a new distributed ledger technology (DLT) architecture, inspired by the current theoretical definitions of web4, that aims to address many of the security issues facing DAG based network design as well as the scalability problems of PoH and transitive learning based networks by combining the two, and then additionally we would like to propose a series of AI models that could be integrated seamlessly, resulting in a robust and constantly self iterating network, far more scalable and secure than anything on the market today. Additionally the network can support ZK subnetworks to provide users custom privacy options. Ultimately creating a totally novel architecture that is far more efficient than the sum of it’s parts.

This network would be based upon Proof of Experience (PoE) - a proof for establishing a network recollection through deep learning and other AI models to improve on network security and some of the other issues facing leading DLT network designs. PoE leverages the encoding of trustless passage of time into a ledger & the transitive learning approach of PoH and combines it with DAG technology as well as a completely novel AI model supported network

architecture. Reducing messaging overhead in a Byzantine Fault Tolerant replicated state machine resulting in finality times of fractions of a second, increasing security, robustness, and scalability.

PoE like PoH can also encode trustless passage of time into a ledger but takes it a step further by establishing a ”network experience” through AI models that provides unique advantages both internally and externally for builders on top of the network. Reducing verification and validation node requirements in the network, resulting in a faster, highly scalable, and more secure network than it’s predecessors.

This paper proposes 3 more algorithms that leverage the experience tracking properties of the PoE ledger system - a Proof of Experienced Stake (PoES) algorithm that can further improve the recovery from partitions of any size, and an efficient real-time Network Intelligence Control System (NICS) system used in tandem with a Dynamic Sub-Sharding (DSS) partition system. The combination of NICS, DSS, and PoES provides a better defense against forgery of the ledger with respect to time (ordering) and storage.

The main advantages of this type of a network include;

1. **Fraud detection:** AI algorithms can be used to identify and prevent fraudulent transactions in real-time, ensuring the security and integrity of the network.
2. **Consensus algorithms:** AI can be used to optimize consensus algorithms in network systems, leading to faster and more efficient decision-making.
3. **Network optimization:** AI can be used to analyze networks and identify bottlenecks or inefficiencies, allowing for optimization and improved performance.
4. **Predictive maintenance:** AI can be used to predict potential issues in a network and prevent them from causing disruptions.
5. **Smart contract development:** AI can be used to automate the development and deployment of smart contracts, enabling faster and more efficient contract execution.
6. **Scalability:** the use of a DAG-based consensus mechanism would allow for high scale-ability and high transaction throughput, as transactions can be processed in parallel.
7. **Security:** the integration of PoH technology would provide a secure and tamper-proof timeline of events, making it difficult for malicious actors to manipulate the history of the system.
8. **Efficiency:** the combination of DAG and PoH technologies would provide a more efficient system compared to traditional blockchains, as the network can handle a high volume of transactions with low latency.

This novel network design also faces the following disadvantages or potential hurdles to overcome;

1. **Complexity:** The combination of these technologies may result in a more complex system compared to traditional blockchains, making it more difficult to develop, implement, and maintain.
2. **Maturity:** DAG and PoH technologies are relatively new and may still have some issues that need to be addressed, such as potential security vulnerabilities or scalability challenges
3. **Compatibility:** This architecture may not be compatible with existing systems, making it more difficult to integrate with existing infrastructure.

We will address all of this in the following report.

In the context of the network we have been designing, the analysis of the protocol was conducted on a z (1 gbps) network. The findings indicate that with today’s hardware, the protocol can achieve a throughput of up to x (710k) transactions per second.

1 Introduction

”Some people can read War and Peace and come away thinking it’s a simple adventure story.” [1] We are currently in the midst of a technological renaissance and tipping point that will bring distributed ledger technology and AI as a core and indispensable part of every human’s every day life. It is imperative that we continue to constantly learn and innovate on the technology so that the networks used by the builders of tomorrow are as fast, scalable, and secure as possible. Imagine if you can a world where the networks themselves could learn and become more robust alongside the developers who work on them.

1.1 NEX Goals and Principles

The NEX is a high-performance, scalable, customizable, and secure distributed ledger platform. It targets three broad use cases:

1. Creating arbitrarily complex digital assets with personalized regulations, agreements, and stipulations (smart assets)
2. Building application-specific distributed ledgers, encompassing both permissioned (private) and permissionless (public) implementations
3. Building and launching highly scalable and decentralized applications (Dapps)

The overarching aim of NEX is to provide a unifying platform for the creation, transfer, and trade of digital assets while working towards full AI integrated web4 technologies. Incorporating a self iterating AI model control system to monitor, optimize, and safeguard the web4 network

By construction, NEX possesses the following properties:

- **Scalable** The design of NEX is geared towards massive scalability, durability, and efficiency. The core consensus engine is able to support a global network of potentially billions of internet-connected, low and high powered devices that operate seamlessly, with low latencies and very high transactions per second.
- **Secure** NEX is engineered with the aim of being secure and resilient. Traditional consensus protocols have a limited ability to withstand a maximum of f attackers, and collapse completely when faced with an adversary of size f+1 or greater. On the other hand, the Nakamoto consensus is not secure when more than 51% of the miners behave in a malicious manner. In contrast, DAGs such as the Avalanche network provide a substantial guarantee of security when faced with attackers below a certain limit, which can be determined by the system designer. When faced with an attacker who surpasses this limit, the network experiences a gradual decline, while still maintaining safety guarantees, even when the attacker exceeds 51%. The NEX system is designed to be proactive in detecting and addressing potential security threats. By analyzing past attacks and conducting simulations, the network can continuously adapt and prepare for future challenges. With its dynamic PoE clusters, the network promises improved consensus, stronger security measures, increased scalability and more efficient verification process, making it the first open network to offer such robust and scalable security guarantees.
- **Decentralized** The NEX network is committed to unparalleled decentralization, with multiple client implementations and absence of centralized control. The ecosystem is structured to eliminate any differences between groups with varying interests, making it a level playing field for miners, developers, and users alike. We are committed to ensuring that all AI models created by us are as impartial as possible, with a focus on utilizing unbiased data during the training process.
- **Governable and Democratic** $XP is designed to be a highly accessible platform that allows anyone to join its network and actively participate in validation and governance. With $XP, every token holder has a say in crucial financial decisions and can help shape the future direction of the system.
- **Interoperable and Flexible** NEX aims to provide comprehensive and adaptable infrastructure for a variety of distributed ledgers and assets, with the base $XP serving as a means of security and a unit of exchange. The platform has been designed to accommodate a wide range of DLTs, allowing for easy migration of existing ones and support for multiple scripting languages and virtual machines. NEX intends to provide value-neutral support for multiple deployment scenarios, including private DLTs and transactions.

2 Outline

The remainder of this article is organized as follows. Overall system design is described in Section 3. In depth description of Proof of Experience is described in Section 4. In depth description of the proposed Network Intelligence Control System is described in Section 5. In depth description of the proposed Dynamic Sub Sharding algorithm is described in Section 6. Proof of Experienced Stake consensus is described in section 7. System Architecture and performance limitations are analyzed in Section 8.

3 Network Design

In an effort to combine the efficiency of PoE with Dynamic Sub-Sharding, a competitively scalable sharding approach proposed in this paper, an AI control system composed of a variety of ML models can be used to efficiently control the network. The network intelligence control system (NICS) will learn from previous consensus block states to inform future consensus as well as assist with management and control of the network's complex dynamic node sharding system.

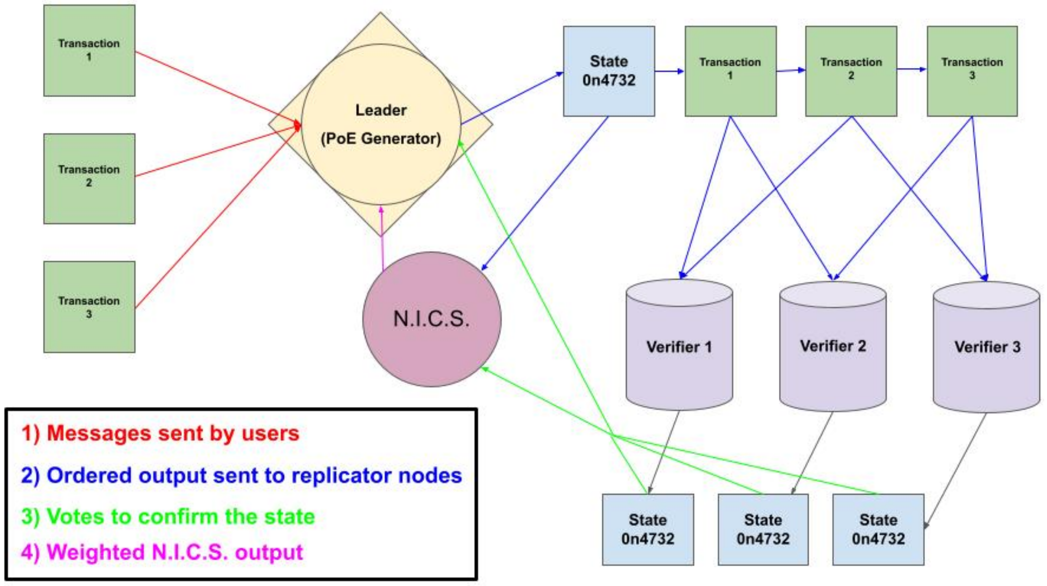


Figure 1: PoE transaction flow throughout the network

At any given point in time a system node is designated as Leader to generate a Proof of Experience sequence, which is a PoH sequence combined with output variables of AI models tracking the system in real-time, providing the network system global read consistency, a verifiable passage of time, and a recollection of all network states and previous metrics through the networks lifetime (experiences). 1 The Leader sequences user messages and orders them such that they can be efficiently processed by other nodes in the system, additionally the final validated hash chain sequence is then processed in parallel along side all other cluster leader nodes within the systems DAG network maximizing throughput & security. It executes the transactions on the current state that is stored in RAM and publishes the transactions and a signature of the final state to the replications nodes called Verifiers.

Verifiers execute the same transactions on their copies of the state, and publish their computed signatures of the state as confirmations. The published confirmations serve as votes for the consensus algorithm.

The NICS simultaneously takes the user messages data, as well as the order sequenced by the Leader and voted on by the Verifiers. This data is used by the NICS to inform future predictions alongside future state Verifier votes.

In a non-partitioned state, at any given time, in each node cluster, there is one Leader in the network. Each Verifier node has the same hardware capabilities as a Leader and can be elected as a Leader, this is done through PoES based elections. Elections for the proposed PoES algorithm are covered in depth in Section 7. In terms of CAP theorem, Consistency is almost always picked over Availability in an event of a Partition. In case of a large partition, this paper proposes a mechanism to recover control of the network from a partition of any size. [2]

3.1 Mechanism and Properties

The Experience* protocols function through repeated sampling of the network. Each node inquires a randomly selected group of neighbors, which is of a dynamic size, and changes its suggestion if the majority supports a different value. The process of sampling is repeated until the network reaches convergence, which typically occurs quickly in standard operations.

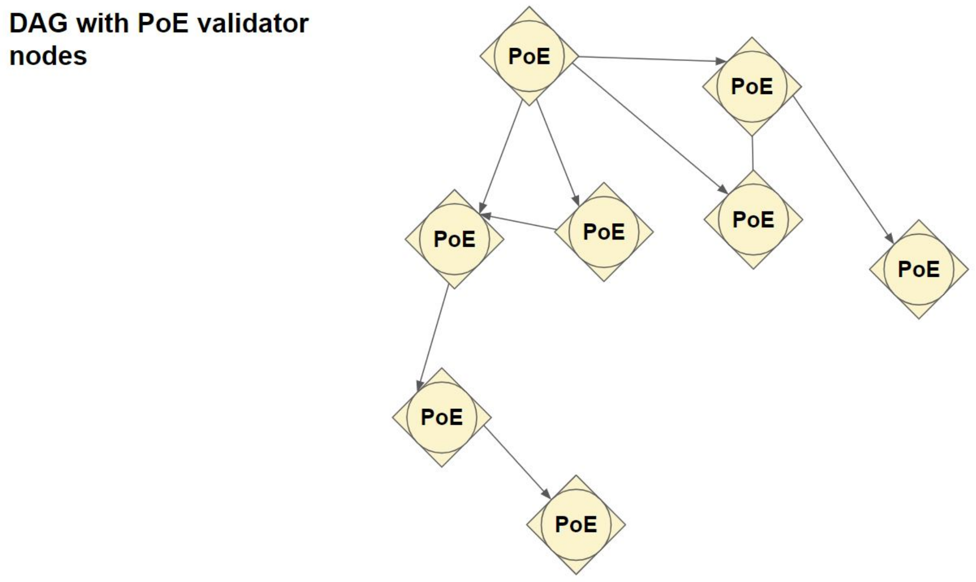


Figure 2: NEX network DAGxPoE visual representation

The process of reaching consensus starts with a user creating a transaction. The transaction is then sent to a validating node, which is part of the consensus process. The validating node spreads the transaction to other nodes in the network using a method known as gossiping. To determine the valid transaction among conflicting transactions and prevent doublespending, every node selects a random subset of other nodes and asks for their opinion on the conflicting transactions. If the majority of the nodes queried support a particular transaction, the inquiring node changes its own stance to support that transaction. This process is repeated by every node in the network until the entire network reaches agreement on a single transaction.

While the core mechanism of operation is quite simple, these protocols result in advantageous system dynamics that make them ideal for large-scale implementation.

- **Permissionless, Open to Churn, and Robust** Experience* protocols have the advantage of providing strong security even when there are discrepancies in the network views of different nodes. This makes them well-suited for public blockchains, as they can be validated without the need for full knowledge of the network's members, which can be difficult to obtain in open and decentralized systems. Unlike many traditional blockchain projects that use classical consensus protocols and require complete knowledge of all participants, Experience* protocols are robust and secure.
- **Scalable and Decentralized** The Experience protocol stands out for its ability to scale without sacrificing its core qualities. Unlike other blockchain projects that rely on classical consensus protocols and require a complete list of participants, Experience protocols offer robust security even when there are discrepancies between network views of nodes. The system allows for validation without needing continuous full membership knowledge, making it ideal for public blockchains. Unlike proof-of-stake protocols that delegate validation to subcommittees, Experience protocols enable every node to validate and participate in the system, providing the highest level of decentralization and security. Scaling strategies such as subcommittee election and state sharding, which parallelize transaction serialization to independent networks of validators, compromise the security of the system by only being as secure as the easiest corruptible independent component. Experience protocols, on the other hand, offer scalability without these trade-offs.
- **Adaptive** Unlike other voting-based systems, Experience* protocols achieve higher performance when the adversary is small, and yet highly resilient under large attacks.
- **Asynchronously Safe** Unlike traditional longest-chain protocols, Experience* protocols do not rely on synchronicity to prevent double-spends even when the network experiences partitions. For instance, in the case of Bitcoin, if the synchronicity assumption is broken, it can result in separate forks of the network operat-

ing independently for a significant amount of time, causing transactions to become invalid once the forks eventually merge.

- **Low Latency** Today, most blockchains are not capable of supporting practical business needs, such as trading or routine purchases, due to long wait times for transaction confirmation. Hence, the time to finalization is a critical aspect of consensus protocols that is often overlooked. Experience* protocols attain finality in a matter of seconds, which is faster than longest-chain protocols and sharded blockchains, which usually take several minutes. Furthermore, Experience* protocols boast high throughput and can handle thousands of transactions per second while maintaining full decentralization. Other blockchain solutions claiming high TPS often compromise decentralization and security for centralization and less secure consensus mechanisms. Some projects may report inflated performance numbers from controlled environments, but the performance figures for Experience* are taken from a real network running on 2000 nodes globally, using low-end machines. The performance can be improved with dedicated hardware and higher bandwidth allocation for each node. It's worth mentioning that these metrics are based on the base-layer and can be significantly increased through layer-2 scaling solutions.

4 Proof of Experience

4.1 Description

PoE, short for ”Proof of Experience,” is a unique consensus algorithm that utilizes historical data from the network and its participants to select bookkeeping nodes. It builds upon the concept of Proof of History (PoH), which uses cryptographic methods to verify the passage of time while also incorporating machine learning models and historical data to make informed decisions. This results in a more decentralized network, as it reduces the system requirements for nodes through the use of DSS. In addition, Proof of Experienced Stake (PoES) factors in not only a validator's current stake in the network, but also their historical staking metrics. Both PoES and DSS are discussed in more detail in sections 6 and 7, respectively.

By learning from past experiences, PoE seeks to improve the efficiency, scalability, and security of the network, reducing the risk of malicious activities. Essentially, PoE can be seen as an evolved version of PoH with added iterative learning capabilities, thus the name ”Proof of Experience.

4.2 PoE AI Models

In PoE each block's transaction's data is used to train a variety of unsupervised and supervised models. Some potentially beneficial AI models to be used in the consensus step are as follows;

4.2.1 Reinforcement Learning

This type of AI model could be used to learn from the decisions made by nodes in the network and improve the consensus process over time. Reinforcement learning models can be used to optimize the time for block creation in a PoE consensus mechanism. The agent could learn from past experiences and adjust the block creation time to balance the trade-off between the time taken for block creation and the amount of rewards received.

Suppose we have a network with a PoE consensus mechanism that takes into account both the stake of a validator in the network as well as the historical stake over time of that same validator. The network uses a reinforcement learning model to determine the weight of each validator in the consensus mechanism.

The reinforcement learning model works as follows:

The model receives as input the historical stake of each validator over a certain time period, the current stake of each validator, and the current state of the network.

Based on this input, the model outputs a weight for each validator, which determines its influence in the consensus mechanism.

The network uses these weights to determine the consensus on a block.

The model is then updated using reinforcement learning to improve its accuracy in predicting the correct weights for each validator. The model receives a reward if the consensus mechanism produces a correct and valid block, and a penalty if the consensus mechanism produces an invalid or incorrect block.

Over time, the model learns which validators are more likely to produce correct blocks and assigns higher weights to those validators, while assigning lower weights to validators that are less likely to produce correct blocks.

In this way, the reinforcement learning model improves the accuracy and efficiency of the PoE consensus mechanism by learning from past performance of validators and adjusting the weights of each validator accordingly.

4.2.2 K-Means Clustering

K-means clustering is a type of unsupervised machine learning algorithm that is used to group similar data points together. In the context of a proof of experience type consensus method, K-means clustering could be used to group validators with similar levels of experience or stake. Here's a hypothetical example:

Suppose there is a network that uses a proof of experience consensus method, and there are 100 validators participating in the network. Each validator has a certain level of experience, which is represented by a numerical value. Let's say that the experience levels range from 1 to 10, with 10 being the highest level of experience.

To use K-means clustering in this context, we could divide the validators into 5 clusters based on their experience levels. The algorithm would group validators with similar levels of experience together and assign them to a cluster. For example, validators with experience levels 1-2 might be grouped into one cluster, validators with experience levels 3-4 might be grouped into another cluster, and so on.

Once the validators have been grouped into clusters, the network could adjust the weight of each validator's vote based on their cluster membership. Validators in clusters with higher levels of experience or stake could have a larger weight, while validators in clusters with lower levels of experience or stake could have a smaller weight. This would give more weight to the votes of validators with more experience, which could help improve the overall security and reliability of the network.

4.2.3 Naive Bayes

Naive Bayes models are a family of probabilistic algorithms based on applying Bayes' theorem with a ”naive” assumption of independence between features. In the context of a proof of experience type consensus method, one potential use of Naive Bayes could be in identifying validators with similar behavior in terms of their decision-making and validation accuracy.

For example, suppose we have a set of validators in the network, each of whom makes decisions on whether to accept or reject proposed transactions. We can train a Naive Bayes model to learn patterns in how these validators make their decisions, using features such as the number of transactions accepted or rejected, the time taken to make decisions, and the type of transactions being processed.

Once the Naive Bayes model is trained, we can use it to cluster validators into groups with similar decision-making behavior. Validators with similar behavior may be more likely to reach consensus and therefore be grouped together in sub-shards. This could help to increase the speed and efficiency of the consensus process, while maintaining high security.

However, it's important to note that Naive Bayes is just one of many possible machine learning techniques that could be applied to a proof of experience consensus method. The choice of algorithm will depend on the specific features and characteristics of the network, and on the goals and objectives of the consensus method.

4.2.4 Time Series Models

Time series models, such as ARIMA (AutoRegressive Integrated Moving Average) or LSTM (Long Short-Term Memory) networks, can be used to analyze time-dependent data in PoE consensus.

Let's say validators are expected to process a certain number of transactions per second. We want to track the performance of each validator over time to ensure that they are meeting their obligations and are contributing to the network in a consistent and reliable way.

We can use time series models to analyze the historical transaction data of each validator and identify any patterns or trends in their transaction processing speed. For example, we could use a moving average or exponential smoothing model to track the average transaction processing speed of a validator over time.

We can then use these models to predict the expected transaction processing speed of a validator in the future, based on their historical performance. If a validator's predicted transaction processing speed falls below a certain threshold, it could be flagged as a potential issue and subjected to additional scrutiny or penalties.

By using time series models in this way, we can improve the overall reliability and consistency of the network by ensuring that validators are consistently meeting their obligations and contributing to the network in a stable and predictable manner.

4.3 Verification

4.3.1 Parallel processing

The network hash sequence verification can be completed faster by further leveraging the asynchronous capabilities of the sharded network. By using multiple processing units to process different parts of the network hash sequence verification simultaneously, the time taken to complete the verification can be reduced. **??**

Expanding on the idea of splitting hash sequence verification between cores on a single node, instead of redundantly verifying the same hashes on every verifying node in the network, hash sequence chunks can be further split up between distributed nodes and verified quicker asynchronously.

Given some number of verifiers in the network with some average number of cores, say 10 verification nodes with modern 4000 core GPUs, instead of redundantly verifying the historical hash sequence 10 times, the verifiers can distribute the workload amongst themselves, further split up their chunked sequences of hashes and their indexes into 4000 slices, and in parallel confirm each slice is correct from starting hash to the last hash in the last node.

If the expected time to produce the sequence is going to be:

$$\frac{Total\ number\ of\ hashes}{Hashes\ per\ second\ for\ 1\ core} \tag{1}$$

If we want to verify the sequence n times to maintain security standards, the expected time to verify that the sequence is correct is going to be:

$$\frac{\frac{Total\ number\ of\ hashes}{Hashes\ per\ second\ for\ 1\ core}}{(Number\ of\ cores\ available\ to\ verify)(Number\ of\ nodes\ in\ cluster\ delegated\ to\ verify)} \tag{2}$$

In the example in Figure X, each node is able to process it's chunk of the sequence in parallel, and each core is able to verify each slice of each chunk in parallel.

They can divide the hash sequence into 4000 slices and each node can verify a different slice. For example, Node 1 can verify slices 1 to 400, Node 2 can verify slices 401 to 800, and so on. This way, each node is responsible for verifying a different part of the hash sequence, making the process faster and more efficient.

Each node can verify the integrity of the hashes in its slice, starting from the first hash in its slice to the last hash in the last slice (Node 10). Once all nodes have confirmed that their slices are correct, they can collectively confirm that the entire hash sequence has been verified accurately.

4.3.2 Use of optimized algorithms

By using optimized algorithms for the hash function or the consensus method, the time taken to complete the verification can be reduced.

4.3.3 Caching

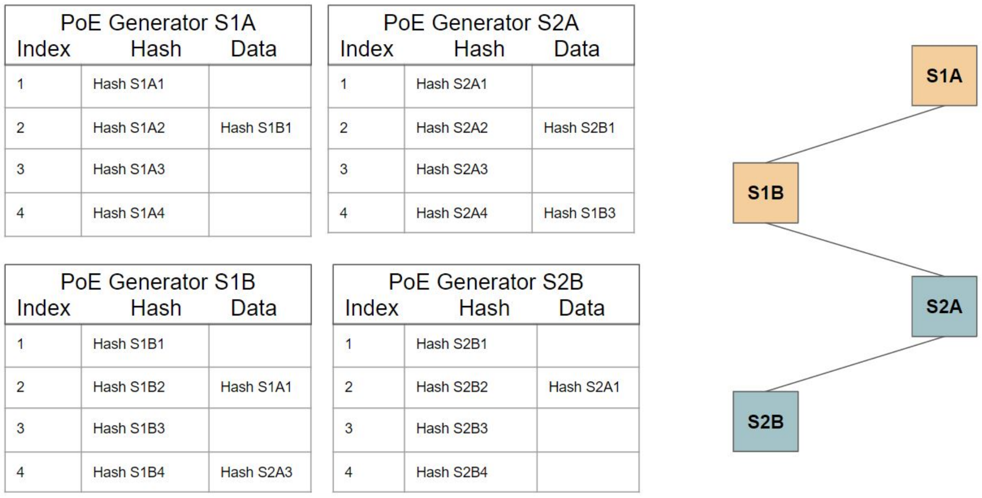
By caching the results of previous verifications, the time taken to complete the verification can be reduced by avoiding redundant computations.

4.3.4 Hardware acceleration

By using specialized hardware, such as graphics processing units (GPUs) or field-programmable gate arrays (FPGAs), to perform the hash function or consensus method, the time taken to complete the verification can be significantly reduced. To scale vertically, the network needs to support the upgrade of hardware components in a seamless and non-disruptive manner. This can be done through the use of virtualization technology, which allows multiple virtual nodes to run on a single physical node. This way, existing nodes can be replaced with higher performance hardware, or additional nodes can be added to the network as needed, all while maintaining compatibility with the existing infrastructure and minimizing downtime.

4.4 Horizontal Scaling

It is possible to synchronize multiple shards running multiple Proof of Experience generators by first mixing the sequence state from each generator to each other generator, and then mixing each final mixed sequence state together into 1 final network sequence, thus achieving much stronger horizontal scaling of the network. The output of all shards is necessary to reconstruct the full order of events in the system.



PoE Generator S1A

Index	Hash	Data
1	Hash S1A1	
2	Hash S1A2	Hash S1B1
3	Hash S1A3	
4	Hash S1A4	

PoE Generator S2A

Index	Hash	Data
1	Hash S2A1	
2	Hash S2A2	Hash S2B1
3	Hash S2A3	
4	Hash S2A4	Hash S1B3

PoE Generator S1B

Index	Hash	Data
1	Hash S1B1	
2	Hash S1B2	Hash S1A1
3	Hash S1B3	
4	Hash S1B4	Hash S2A3

PoE Generator S2B

Index	Hash	Data
1	Hash S2B1	
2	Hash S2B2	Hash S2A1
3	Hash S2B3	
4	Hash S2B4	

Figure 3: Transitive learning in a sharded network

Given shards S1 and S2 generators S1A & S1B on shard 1, and S2A & S2B on shard 2, S1A & S1B share data packets transitively shards share data transitively amongst other nodes in their shards, and then these shards are all consolidated into 1 final network sequence. The generators between different shards are able to also share their data transitively. Only a subset of the generators in a shard will share data amongst other shards, thus reducing the load on the average generator in each shard.

4.5 Attacks

4.5.1 Reversal

Generating a reverse order would require an attacker to start the malicious sequence after the second event. This delay should allow any non malicious peer to peer nodes to communicate about the original order.

4.5.2 Speed

Having multiple generators may make deployment more resistant to attacks. One generator could be high bandwidth, and receive many events to mix into its sequence, another generator could be high speed low bandwidth that periodically mixes with the high bandwidth generator. The high speed sequence would create a secondary sequence of data that an attacker would have to reverse.

4.5.3 Long Range Attacks

Long range attacks involve acquiring old discarded client Private Keys, and generating a falsified ledger [**?**]. Proof of History provides some protection against long range attacks. A malicious user that gains access to old private keys would have to recreate a historical record that takes as much time as the original one they are trying to forge. This would require access to a faster processor than the network is currently using, otherwise the attacker would never catch up in history length. Additionally, a single source of time allows for construction of a simpler Proof of Replication (more on that in Section 6). Since the network is designed so that all participants in the network will rely on a single historical record of events. PoRep and PoH together should provide a defense of both space and time against a forged ledger.

5 N.I.C.S.

5.1 Description

The network intelligence control system or NICS is a complex amalgamation of a number of AI models and machine learning supported systems.

5.2 Inputs

An AI model that aimed to track data from a network like this would typically take in the following inputs:

- **Network structure:** Understanding the structure of the network such as the number of nodes, the number of edges, and their relationships. This information helps the AI model to understand the network's topology and how data is flowing through the network.
- **Node characteristics:** Information about individual nodes such as their processing capacity, storage capacity, and communication capabilities. This information helps the AI model to understand the node's potential and limitations, and to optimize data flow through the network.
- **Node utilization:** Information about the current utilization of each node in terms of processing and storage capacity. This information helps the AI model to understand the current workload of each node and make decisions about how to allocate additional workload.
- **Network traffic:** Information about the data flow between nodes, including the volume and type of data. This information helps the AI model to understand the network's data patterns and optimize data flow.
- **Performance metrics:** Information about the performance of the network, such as latency, throughput, and error rate. This information helps the AI model to understand the network's performance and identify bottlenecks.
- **Historical data:** Historical data about the network, including previous network configurations, node utilization, and performance metrics. This information helps the AI model to make predictions and inform future decisions.

Each input parameter is crucial to understanding the network, optimizing data flow, and ensuring optimal performance.

5.3 NICS AI Models

There is a wide range of AI models that could be used to process information from a network, depending on the specific goals of the model. Some common models include:

5.3.1 Time-series forecasting models

These models could be used to track metrics such as network traffic or resource utilization over time, with the goal of predicting future trends. Examples of these models include ARIMA and LSTM.

For example, NICS could use a seasonal autoregressive integrated moving average (ARIMA) model to forecast the network's traffic volume and bandwidth usage over time.

The NICS could collect data on the network's historical traffic volume and bandwidth usage, as well as other metrics like latency and packet loss, and use this data to train the seasonal ARIMA model. The model could then be used to generate forecasts of these metrics for future time periods, allowing the NICS to anticipate and proactively address any potential issues before they occur.

For example, if the seasonal ARIMA model forecasts a spike in traffic volume in a particular geographic region, the NICS could dynamically allocate additional resources to that region to ensure that the network continues to operate smoothly. Similarly, if the model predicts an increase in latency or packet loss, the NICS could take steps to optimize the network's routing algorithms or adjust the sharding parameters to improve performance.

Overall, time series forecasting models can be a powerful tool for the NICS to help manage and optimize the network's performance and ensure its long-term scalability and reliability.

5.3.2 Anomaly detection models

These models could be used to identify unusual or suspicious activity within the network, such as unexpected spikes in traffic or sudden changes in resource utilization. Examples of these models include Isolation Forest and One-Class SVM.

For example, a simple approach to anomaly detection might be to model the traffic in the network as a time series, and use statistical techniques such as moving averages or exponential smoothing to identify trends and seasonality in the data. The model can then be used to predict the expected behavior of the network, and any deviations from this expected behavior can be flagged as anomalies.

More sophisticated anomaly detection models can be trained using machine learning techniques such as clustering or classification. For example, clustering techniques such as k-means clustering can be used to group similar network behavior into clusters, and any behavior that falls outside of these clusters can be flagged as anomalous. Similarly, classification techniques such as decision trees or random forests can be trained to identify patterns in the network traffic that are indicative of malicious behavior.

Once anomalies have been detected, the NICS can take action to mitigate the impact of the anomalous behavior. For example, it could flag the behavior for further investigation, or it could trigger automatic responses such as routing traffic through alternative paths or even blocking certain nodes from the network.

5.3.3 Graph-based models

These models could be used to analyze relationships between nodes in the network, with the goal of identifying key players or communities within the network. Examples of these models include PageRank and Community Detection.

For example, the NICS could use graph-based models to analyze the network's communication patterns and identify nodes that are communicating more frequently or less frequently than expected. This could help detect abnormal behavior that may indicate a security threat or a node failure.

To demonstrate this, let's say that the NICS is analyzing a section of the network and has collected data on the communication patterns between nodes. The data is represented as a graph, with nodes representing the network's components (e.g., nodes, servers, routers) and edges representing the connections between them.

The NICS could then use graph-based algorithms to identify patterns in the data. One approach is to use a clustering algorithm, such as the Louvain method, to group nodes based on their communication patterns. The Louvain method is a popular community detection algorithm that identifies groups of nodes that are more densely connected to each other than to the rest of the network.

Once the NICS has identified the node clusters, it can analyze the communication patterns within and between clusters to detect anomalies. For example, if a node within a cluster starts communicating more frequently than the other nodes in the same cluster, this could indicate abnormal behavior that requires investigation.

In summary, graph-based models are a powerful tool for detecting anomalies in complex networks like the one we've been discussing. By analyzing the network's topology and communication patterns, the NICS can identify patterns and anomalies that may indicate security threats or other issues.

5.3.4 Cluster analysis models

These models could be used to group similar nodes in the network based on shared characteristics, with the goal of understanding the underlying structure of the network. Examples of these models include K-Means and Hierarchical Clustering.

For example, if a cluster of nodes is consuming significantly more resources or generating a larger number of transactions than other clusters, this may be a sign of malicious activity or a vulnerability in the network.

To demonstrate this, let's say that the NICS is monitoring a DAG-based cryptocurrency network and using cluster analysis to identify potential threats. The NICS has collected a large amount of transaction data and has identified several clusters of nodes based on their transaction history.

Using cluster analysis, the NICS can identify anomalous behavior in a cluster of nodes that is different from the behavior of the other clusters. For example, if the nodes in one cluster are consistently generating a high number of transactions with unusual patterns or timings, this may be an indication of a coordinated attack.

The NICS can use this information to trigger an alert and take action to prevent further damage to the network, such as isolating the cluster of nodes and launching a deeper investigation into the potential threat.

Overall, cluster analysis models can be a valuable tool for the NICS to detect and respond to potential threats in the network, and can be integrated with other models to form a comprehensive defense strategy.

5.4 Outputs

Some possible outputs that could be tracked by these models include:

- **Predictive metrics** These metrics could include forecasts for network traffic, resource utilization, and other important metrics over time.
- **Anomaly scores** These scores could indicate the likelihood that a given node or network behavior is unusual, and help identify potential security risks.
- **Centrality measures** These measures could indicate the importance of a given node within the network, based on factors such as connectivity, in-degree, and out-degree.
- **Cluster labels** These labels could indicate which nodes belong to which groups within the network, helping to identify key communities or subgroups.

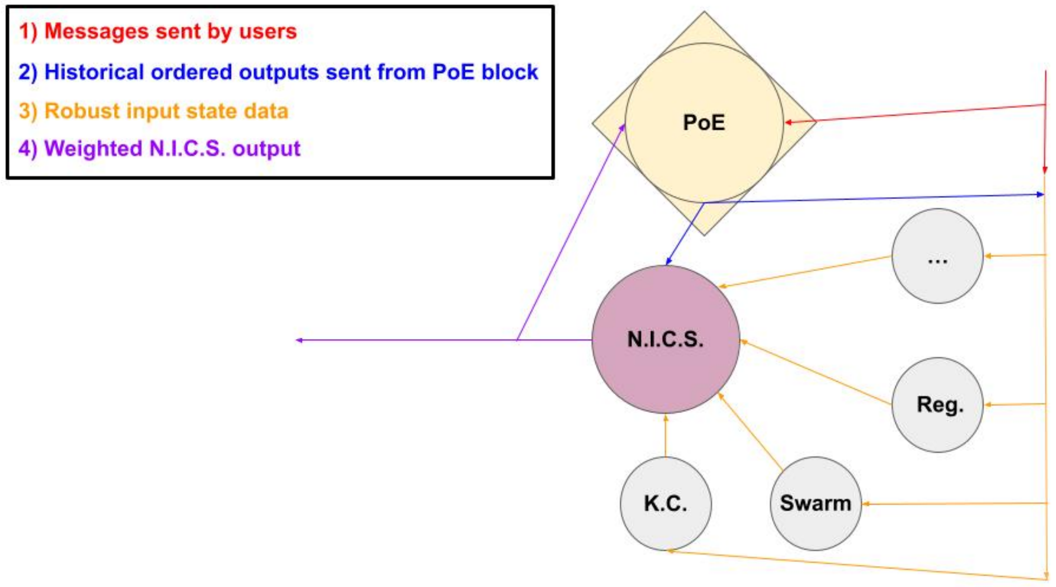

1) Messages sent by users
2) Historical ordered outputs sent from PoE block
3) Robust input state data
4) Weighted N.I.C.S. output
PoE
...
N.I.C.S.
Reg.
K.C.
Swarm

Figure 4: N.I.C.S. visual representation

5.5 Verification Models

AI models can improve the verification process in various ways, such as:

5.5.1 Fraud Detection

AI models can be used to detect anomalies and suspicious behavior within the network, which can be indicative of fraudulent activity. These models can be trained on large amounts of data to identify patterns in the data that may not be apparent to humans.

5.5.2 Optimization of Verification Processes

AI models can be used to optimize the verification process by finding the most efficient way to perform verifications. This can include identifying which nodes to target, which data to verify, and how often to perform verifications.

Here’s a demo example:

Let's say that the NICS receives a large number of new transactions, some of which may be invalid. The NICS could use a machine learning algorithm, such as a decision tree or a random forest, to analyze the features of each transaction (e.g. sender address, amount, transaction fee) and predict whether the transaction is likely to be valid or invalid.

Once the machine learning algorithm has made its predictions, the NICS could then prioritize the verification of transactions that are predicted to be valid, and deprioritize transactions that are predicted to be invalid. This could help to improve the efficiency of the verification process by reducing the amount of time and resources spent on verifying invalid transactions.

Of course, this is just one example of how AI models could be used by the NICS to optimize the verification process. There are many other possible approaches, depending on the specific needs of the network and the types of transactions being processed.

Another example of how AI models might be used by the NICS to optimize the verification process is by using natural language processing (NLP) to analyze and classify network traffic based on the content of the messages being transmitted.

For example, the NICS could use NLP models to analyze chat logs, emails, and other forms of communication exchanged between network participants. The models could identify patterns and anomalies in the communication, such as suspicious or fraudulent activity, and trigger alerts or initiate appropriate responses.

The NICS could also use NLP models to analyze documentation and other types of data related to network operations and maintenance, such as software patches, system logs, and user manuals. The models could help to identify potential vulnerabilities, provide insights into system performance and usage, and inform decisions around resource allocation and network optimization.

5.5.3 Automation

AI models can be used to automate the verification process, reducing the amount of manual effort required. This can increase the speed and accuracy of the verification process, as well as reduce the potential for human error.

Let's say that the NICS receives a large number of verification requests every day from different nodes on the network. Each request contains a set of data points, and the NICS needs to verify the authenticity and accuracy of each data point before it can be added to the ledger.

To automate this process, the NICS could use AI models such as supervised learning algorithms to train a model on a large dataset of verified data points. The model could be trained to identify patterns in the data that indicate whether a data point is likely to be genuine or not.

Once the model is trained, the NICS could use it to automatically verify new data points as they are received. The model would compare each new data point to the patterns it has learned from the training data, and assign a confidence score indicating the likelihood that the new data point is genuine.

If the confidence score is high enough, the NICS could automatically add the new data point to the ledger. If the confidence score is low, the NICS could flag the data point for manual review by a human analyst.

By using AI to automate the verification process in this way, the NICS could significantly increase the speed and efficiency of the verification process, while also reducing the risk of errors or fraud.

5.6 Horizontal Scaling

To scale horizontally, the system can be designed to be distributed across multiple nodes, with each node responsible for processing a portion of the data and running the AI models. The nodes can communicate with each other to share information and collaborate on the decision-making process.

One approach to scaling the system horizontally is to use containerization and container orchestration technologies, such as Docker and Kubernetes. This allows the system to be deployed and managed across multiple nodes in a scalable and efficient manner.

Another approach is to use cloud-based services, such as Amazon Web Services (AWS) or Microsoft Azure, to provide the necessary computing resources for the system. These services can be used to provision and manage large numbers of virtual machines, which can be used to run the system across multiple nodes.

5.7 Consistency

In a blockchain network, consistency refers to the agreement between nodes in the network on the state of the blockchain. AI models could potentially improve the consistency of the network by:

Detecting and preventing malicious behavior: AI models could be trained to identify and prevent malicious activity, such as double-spending, on the network.

Improving consensus algorithms: AI models could be used to optimize consensus algorithms, such as PoS or PoW, by identifying and correcting inefficiencies in the validation process.

Enhancing data validation: AI models could be utilized to validate data transactions, ensuring that the data being processed is accurate and trustworthy.

Monitoring network behavior: AI models could monitor network behavior and alert nodes if any irregular or suspicious behavior is detected. This could prevent potential attacks on the network and improve its overall stability.

Improving network efficiency: AI models could be used to identify and mitigate bottlenecks in the network, improving the overall speed and efficiency of the network.

5.8 Overhead

NICS can improve the overhead of a network by using machine learning algorithms to optimize various network processes such as resource allocation, load balancing, and data management.

For example, AI models could be used to predict network usage patterns and dynamically adjust resource allocation to improve performance and reduce waste. AI models could also be used to identify and prevent potential bottlenecks in the network, improve the accuracy of data validation, and increase the overall efficiency of network operations. By automating these processes, AI models could reduce the overhead and complexity of network management, freeing up resources for other important tasks.

5.9 Attacks

NICS can improve the defensive capabilities of a network by detecting and preventing malicious behavior in real-time.

For example, AI models can be trained on network data to identify patterns of behavior associated with attacks, such as Denial-of-Service (DoS) or Sybil attacks. This allows the network to take proactive measures to defend against these types of attacks.

Additionally, AI models can help identify network vulnerabilities and suggest changes to the network infrastructure that can improve its overall security. This can include improving encryption methods, implementing firewalls, and adding additional layers of authentication.

NICS can be used to monitor the network in real-time, detecting any anomalies in behavior that might indicate an attack is taking place. This can allow the network to respond quickly and effectively to any threats, minimizing the damage done and helping to keep the network secure.

5.9.1 Reversal

In theory, NICS can improve the ability of a network to deal with reversal attacks by providing enhanced security measures.

For example, AI models can be trained on historical network data to identify potential reversal attacks and to take preventative measures before they occur. By analyzing network traffic, AI models could detect abnormal behavior and respond with countermeasures such as rate limiting or temporary blocking of the attacker's IP address.

Additionally, AI models could be used to improve the accuracy of consensus algorithms, such as by reducing the time required for block confirmation or by enhancing the ability of nodes to reach consensus on the correct chain state. By improving these underlying mechanisms, the network as a whole would be better equipped to defend against reversal attacks.

5.9.2 Long Range Attacks

NICS can potentially improve the network's defense against long-range attacks by analyzing network activity and detecting anomalies that could indicate an attack.

For example, an AI model could monitor the frequency and volume of transactions in the network, compare them to historical data, and flag any significant deviations as potential signs of an attack. The model could also analyze the distribution of transactions and flag any abnormal patterns, such as a large number of transactions originating from a single node or a sudden increase in transactions from a particular geographic region. By monitoring and analyzing these patterns, AI models can help detect long-range attacks and provide additional data to the network's security systems, which can then respond more effectively.

6 Dynamic Subsharding

6.1 Description

Dynamic sub-sharding is a method of breaking down a large network into smaller, more manageable parts, or shards, that can be processed in parallel. In a blockchain network, dynamic sub-sharding can be used to improve the scalability, efficiency, and security of the network by distributing the workload, reducing the overhead, and reducing the risk of centralization.

In dynamic sub-sharding, the network is divided into multiple sub-networks or shards, each of which can have its own consensus mechanism, data structure, and security mechanism. The process of dividing the network into shards can be done dynamically, based on the current state of the network, the current workload, the number of nodes, and other factors. The sharding process is managed by the NICS mechanism, which determines the size and number of shards, assigns nodes to shards, and reassigns nodes as needed.

The main advantage of dynamic sub-sharding is improved scalability, as the network can handle a larger number of transactions and nodes without sacrificing performance. This is achieved by distributing the workload across multiple shards, reducing the overhead, and reducing the risk of centralization. In addition, dynamic sub-sharding can improve the security of the network by reducing the attack surface and making it harder for malicious actors to compromise the network.

Dynamic sub-sharding is a complex process that requires advanced AI models and algorithms to manage the sharding process, ensure the consistency of the network, and prevent attacks. These models must be able to detect anomalies, predict network behavior, and make real-time decisions based on the current state of the network.

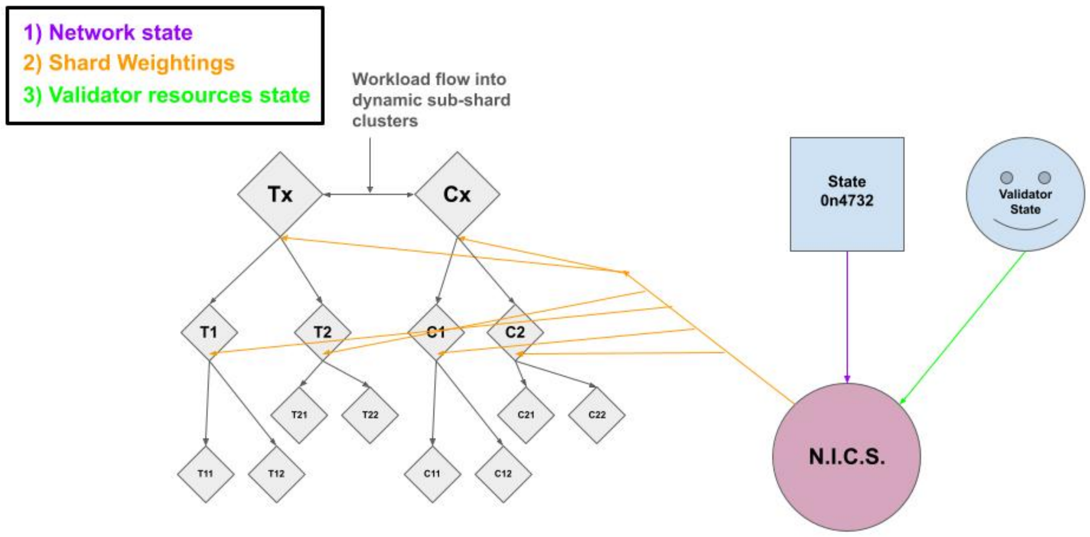


Figure 5: Dynamic Sub Sharding

6.2 DSS AI Models

6.2.1 Swarm AI

Swarm AI models could potentially be useful in controlling the dynamic sub-sharding system in our proposed network design. Swarm AI models are based on swarm intelligence, which is a decentralized approach to problem solving that relies on collaboration between simple agents. This approach aligns well with the principles of sub-sharding, which involves dividing the network into smaller, more manageable pieces.

In the context of dynamic sub-sharding, swarm AI models could be used to help manage and optimize the process of dividing the network, as well as to ensure that sub-shards are functioning optimally and making the best use of resources. For example, swarm AI algorithms could be used to determine which transactions should be assigned to which sub-shards, to monitor the performance of sub-shards in real time, and to make adjustments to the sub-sharding process as needed to ensure the network is operating efficiently.

Particle Swarm Optimization (PSO) - This model uses a swarm of particles to search for optimal solutions. It could be useful in finding the best partitioning of the network into sub-shards.

Ant Colony Optimization (ACO) - This model uses the behavior of ants to solve optimization problems. It could be useful in finding the best routes for data and transactions to traverse the network.

Artificial Bee Colony (ABC) - This model uses the behavior of bees to solve optimization problems. It could be useful in finding the most efficient way to distribute work across the network and ensure the network remains balanced.

Artificial Fish Swarm Algorithm (AFSA) - This model uses the behavior of fish to solve optimization problems. It could be useful in finding the most efficient way to organize the network and respond to changes in load and demand.

7 Proof of Experienced Stake Consensus

7.1 Description

Proof of Experienced Stake (PoES) combines the traditional Proof of Stake (PoS) mechanism, which takes into account the amount of stake held by a validator, with a historical stake component. By considering both the current stake and the historical stake, PoES aims to provide an additional layer of security and to prevent malicious actors from accumulating a large stake over time and then potentially disrupting the network.

7.2 Terminology

bonds: In Proof of Work, the cost of hardware and electricity is considered a capital expense. The miner invests in these resources and dedicates them to a specific branch within the blockchain. In contrast, a bond in a blockchain represents a validator's commitment to use their coins as collateral while validating transactions.

slashing: This solution aims to address the nothing-at-stake problem in Proof of Stake systems [3]. If a validator publishes proof of voting for a different branch, their bond can be destroyed, which serves as an economic disincentive to prevent validators from confirming multiple branches.

super majority: A "super majority" refers to a vote by two-thirds of the validators based on the weight of their bonds. This indicates that the network has achieved consensus, and at least one-third of the network would need to vote maliciously for the branch to be considered invalid. This implies that the cost of an attack would be one-third of the market cap of the coin.

7.3 Bonding

When a bonding transaction is made, a certain amount of coins is transferred to a bonding account that belongs to the user. These coins cannot be used or spent and must stay in the account until the user decides to remove them. The user can only remove coins that have timed out. A super majority vote by the current stakeholders must confirm the sequence before the bonds become valid.

7.4 Voting

The Proof of Experience generator is expected to publish a state signature at regular intervals. Bonded identities must confirm this signature by publishing their own signed version. A vote is cast with a simple "yes" and there is no option to vote ”no.” If a super majority of bonded identities have voted within a specified time frame, the branch will be deemed valid.

7.5 Unbonding

If a certain number of votes (N) are missing, the coins associated with those votes are considered stale and cannot be used for voting anymore. To remove them, the user can initiate an unbonding transaction. N is calculated dynamically based on the proportion of stale votes to active ones, which means that N increases as the number of stale votes grows. This feature helps the larger branch to recover more quickly than the smaller one in case of a significant network partition.

7.6 Elections

When a failure of the PoE generator is detected, a new election is held. The validator with the highest voting power or public key address (if there's a tie) becomes the new generator, with their historical success rate taken into account. Super majority of confirmations are required for the new sequence. If the new leader fails before there are enough confirmations, the next highest validator is selected, and a new set of confirmations is required. Vote switching requires a higher PoE sequence counter and the inclusion of the votes to switch to avoid being slashable. Once a PoE generator is established, a Secondary can be elected to take over transaction processing. The platform is designed so that the Secondary becomes Primary in case of exception or on a predefined schedule, with lower rank generators being promoted.

7.7 Election Triggers

7.7.1 Forked Proof of Experience Generators

PoE generators are created with an identity that signs the generated sequence, and a fork can only happen if the PoE generator's identity has been compromised. A fork is identified when two distinct historical records are published on the same PoE identity.

7.7.2 Runtime Exceptions

If a hardware failure, bug, or intentional error occurs in the PoE generator, it could generate an invalid state and publish a signature that doesn't match the validators' results. Validators will share the correct signature through gossip, triggering new elections. Validators who accept an invalid state will have their bonds slashed.

7.7.3 Network Timeouts

An election would is triggered in the event of a network timeout.

7.8 Slashing

If a validator votes for two separate sequences, it will result in slashing of the bonded coins and adding them to the mining pool as a proof of malicious vote. However, a vote that includes a previous vote on a contending sequence is not eligible as proof of malicious voting. In such cases, the currently cast vote on the contending sequence will be removed instead of slashing the bonds. Additionally, if a vote is cast for an invalid hash generated by the PoE generator, the validator's bonds will be slashed. It is expected that the generator may randomly generate an invalid state, triggering a fallback to Secondary.

7.9 Secondary Elections

Proposals to elect Secondary and lower ranked Proof of Experience generators can be made and approved. These proposals include a timeout, and if a super majority of the vote approves the motion before the timeout, the Secondary is elected and will take over duties as planned. The Primary can initiate a smooth handover to the Secondary by adding a message to the generated sequence signaling the handover, or by creating an invalid state that prompts the network to fallback to the Secondary.

In case the Primary fails, a Secondary that has been elected will be considered the first option during an election.

7.10 Availability

In our network design, we face a trade-off between Consistency and Availability in the event of partitions. We prioritize Availability but also have a measure of time to ensure Consistency with reasonable timeouts. Our Proof of Experienced Stake verifiers stake a certain amount of coins to vote for transactions and their stake is locked up as a transaction in the PoE stream. Their votes, also entered as transactions, allow us to determine how much time has passed between each vote and how long each verifier has been unavailable during a partition. To handle partitions, we propose dynamically unstaking unavailable verifiers within reasonable human timeframes. The speed of this process is dependent on the number of verifiers available, with faster unstaking for more available verifiers and slower unstaking for fewer available verifiers. In large partitions, where more than half of the verifiers are missing, the unstaking process is very slow, and full consensus may not be achieved until a significant number of hashes are generated and unavailable verifiers are unstaked. Our approach allows us to choose which partition to continue using within reasonable human timeframes.

7.11 Recovery

In the system we propose, the ledger can be fully recovered from any failure, which means that any person can create a valid fork by adding newly generated hashes and transactions to any random point in the ledger. If there are no verifiers available for this fork, achieving 2/3 super majority consensus will take a very long time, and a significant number of hashes must be added to the ledger for full recovery with no available validators. Only after unstaking all the unavailable validators will new bonds be able to validate the ledger.

NICS could potentially improve recovery by automating the process of identifying and recovering from failures. For example, AI models could be trained to detect patterns in the ledger data that indicate a failure has occurred and automatically initiate recovery procedures. Additionally, AI could be used to optimize the recovery process by identifying the most efficient way to add new hashes and transactions to the ledger to achieve consensus. However, it is important to note that AI is not a silver bullet and may not be able to solve all recovery issues on its own. Human oversight and intervention may still be necessary in some cases.

7.12 Finality

In our network design, the PoE system enables verifiers to observe past events with a reasonable degree of certainty. The PoE generator produces a stream of messages that requires all verifiers to submit their signatures of the state within a specified time limit, which is currently set at 500ms and can be adjusted based on network conditions. Each verification is entered into the stream, allowing everyone in the network to validate that all verifiers submitted their votes within the required timeout without directly observing the voting process.

7.13 Attacks

7.13.1 Tragedy of Commons

In our network design, the PoES verifiers confirm the state hash produced by the PoE generator. However, there is a risk that verifiers may not perform any work and simply approve every generated state hash. To prevent this, the PoE generator must randomly inject an invalid hash at intervals, and any voters for this hash should be penalized. Once the invalid hash is detected, the network should immediately promote the secondary elected PoE generator. Verifiers are required to respond within a short timeout, which should be set low enough to discourage malicious behavior and prevent an observer from getting their votes into the stream quickly.

7.13.2 Collusion with the PoE Generator

A verifier who colludes with the PoE generator would have advance knowledge of when the invalid hash will be generated and could choose not to vote for it. However, this situation is similar to the scenario where the PoE identity has a larger verifier stake. Regardless, the PoE generator still bears the responsibility of producing the state hash.

NICS could potentially be used to monitor the behavior of verifiers and detect any patterns or anomalies that may indicate collusion with the PoE generator. For example, machine learning algorithms could be trained on historical data to identify unusual voting behavior or sudden changes in voting patterns that could be indicative of collusion. Additionally, AI could be used to analyze network traffic and detect any suspicious activity that may indicate attempts to manipulate the system.

7.13.3 Censorship

In case one-third of bond holders refuse to validate new bond sequences, censorship or denial of service may occur. The protocol can prevent this attack by dynamically adjusting the rate at which bonds become stale. In the event of a denial of service, the larger partition will fork and censor the Byzantine bond holders, while the larger network recovers as Byzantine bonds become stale. The algorithm involves electing a new Leader who censors the Byzantine bond holders, generating a sequence with the Proof of Experience generator until enough Byzantine bonds become stale, and dynamically adjusting the rate at which bonds become stale based on the percentage of active bonds. The smaller Byzantine partition would have to wait much longer to recover a super majority, while the majority fork recovers faster. Once a super majority is established, slashing could permanently punish the Byzantine bond holders.

NICS could potentially be used in this context to monitor and analyze network activity to detect and prevent potential denial of service attacks or collusion among bond holders. AI algorithms could also be used to optimize the dynamic adjustment of the bond stale rate based on network conditions and activity. Additionally, AI could be used to identify and analyze patterns of behavior among bond holders that may indicate malicious activity or collusion.

7.13.4 Long Range Attacks

PoE provides a natural defense against long-range attacks by making it difficult for attackers to recover the ledger from any point in the past. They would need to overtake the valid ledger in time by outpacing the speed of the PoE generator. The consensus protocol serves as a second layer of defense by requiring any attack to take longer than the time it takes to unstake all the valid validators. This creates an availability gap in the ledger's history. When comparing two ledgers of the same height, the one with the smallest maximum partition can be considered valid based on objective criteria.

7.13.5 ASIC Attacks

There are two opportunities for ASIC attacks - during partition and cheating timeouts in Finality. The rate at which bonds are unstaked during partitions is non-linear, making ASIC attacks inefficient for networks with large partitions. During Finality, ASIC attacks can be carried out by Byzantine validators who collaborate with a PoE generator to inject their votes. The PoE generator can then use its faster ASIC to generate 500ms worth of hashes in less time, allowing for network communication between the generator and collaborating nodes. However, if the PoE generator is also Byzantine, it may communicate the exact counter when the failure is expected to be inserted. This scenario is similar to a PoE generator and all collaborators sharing the same identity and having a single combined stake, only using one set of hardware.

NICS could potentially be used to protect the network from ASIC attacks. One possible approach could be to use machine learning algorithms to detect patterns of behavior that are indicative of ASIC attacks. For example, AI could be used to identify patterns in the rate at which bonds are being unstaked during partitions, and to detect any abnormal behavior that might indicate an ASIC attack is taking place.

Another possible approach could be to use AI to monitor the behavior of Byzantine validators and PoE generators during Finality, looking for patterns that might indicate collusion between the two. AI could also be used to detect any abnormal patterns of communication between PoE generators and collaborating nodes, which could be an indicator of an ASIC attack.

Finally, AI could be used to develop new algorithms and protocols that are specifically designed to protect against ASIC attacks. By analyzing large amounts of data and simulating different attack scenarios, AI could help identify vulnerabilities in the network and develop new protocols that are more resistant to ASIC attacks.

7.13.6 DDOS Attacks

AI can be used to protect the network from DDoS attacks by implementing anomaly detection algorithms that can detect and respond to unusual network traffic patterns. These algorithms can be trained using techniques such as semi-supervised or self-supervised learning, even with limited labeled data. Additionally, model fairness indicators in TensorFlow can be utilized to evaluate the effectiveness of the anomaly detection models and ensure that they do not introduce bias or discrimination. By continuously monitoring and analyzing network traffic using these AI-powered tools, the network can proactively defend against DDoS attacks and ensure its availability to legitimate users. [4]

8 System Architecture - Dynamic PoE Cluster Shards

8.1 Components

8.1.1 Leader, Proof of Experience Generator

The Proof of Experience generator is elected as the designated Leader. Its role is to receive transactions from users and produce a unique global order for them through a Proof of Experience sequence. Upon completion of processing each batch of transactions, the Leader generates a signature that represents the final state of the network, which is then signed with the Leader's identity and propagated to the overarching DAG data structure.

8.1.2 State

The theoretical state might include various data structures to represent the current state of the network. For example, there might be a table or database indexed by users' addresses, which contains information such as their account balance and transaction history. Each cell in this table might contain the user's full address and the memory required for this computation, totaling 32 bytes per cell. Additionally, there might be a Proof of Experienced Stake bonds table, which tracks the bonds staked by validators in the network throughout their history, totaling 64 bytes. These data structures would be constantly updated as new transactions are added to the network and new validators join or leave the network.

8.1.3 Verifier, State Replication

To replicate the state of the network, the verifier nodes would need to synchronize with the other nodes on the network and download a copy of the distributed ledger (DL) or DAG data structure, which contains all the historical transaction data and state updates. Once the verifier nodes have a copy of the DL or DAG, they can verify the validity of transactions and blocks by executing the same consensus and validation algorithms as the other nodes on the network. This ensures that the verifier nodes have an accurate and up-to-date copy of the state of the network. Additionally, the verifier nodes may also periodically check the integrity of their copy of the DL or DAG by comparing it with other nodes on the network to detect any inconsistencies or anomalies.

8.1.4 Validators

These nodes are utilizing Verifiers' bandwidth. They are virtual nodes that can operate on the same machines as the Verifiers or the Leader, or on separate machines that are configured specifically for the consensus algorithm of this network.

8.2 Network Limits

The network is limited by the number of verifiers that can validate each shard, which is directly related to the computational power and memory available to the verifiers and the leader nodes. As the number of shards increases, the number of verifiers required to validate them also increases, potentially leading to a bottleneck in the validation process. The Leader in charge of generating the Proof of Experience sequence is expected to optimize the sequence's efficiency by ordering incoming user packets in the most efficient way possible, minimizing faults and maximizing prefetching.

However, this process also increases the computational and memory requirements, which can limit the scalability of the network. Finally, the network is also limited by the available bandwidth for communication between nodes, which can cause congestion and slower transaction times as the number of nodes and data being transmitted increases. On a 1gbps network connection, the maximum number of transactions possible is 1 gigabit per second divided by the packet size of 176 bytes, which allows for approximately 710k transactions per second, but some loss is expected due to Ethernet framing. To increase availability, Reed-Solomon codes can be used to code the output and stripe it to the available downstream Verifiers.

8.3 Computational Limits

The computational limits of our network design are determined by the amount of computational power and memory available to the verifiers and the leader nodes. As the size of the network and the number of transactions increase, the computational and memory requirements also increase, potentially limiting the scalability of the network. Additionally, each transaction requires a digest verification, which can be parallelized independently and may be limited by the number of cores available on the system. GPU-based ECDSA verification servers have had experimental results of 900k operations per second, indicating a potential performance ceiling. [2]

8.4 Memory Limits

In theory, a basic implementation of the state that uses a 50% full hashtable with 32-byte entries for each account could accommodate 10 billion accounts within 640GB. The system can handle 1.1 x 10^7 writes or reads per second during steady state random access. Based on 2 reads and 2 writes per transaction, memory throughput is capable of processing 2.75 million transactions per second. These measurements were obtained using an Amazon Web Services 1TB x 1.16 x large instance.

9 System Architecture - Overarching DAG structure

9.1 Integration with the PoE clusters

A network design combining PoE and DAG technology can use PoE as a method for securing and verifying the historical order of transactions in the network, and DAG for organizing and structuring these transactions into a directed graph data structure.

The PoE generator can be responsible for generating a hash chain of the transactions that are informed by the NICS, and inputted into the network. The hashes generated would be verified and anchored onto the DAG, allowing for a tamper-evident historical record and artificial system memory of all transactions. The directed graph structure of the DAG can be used to represent the relationships between transactions, providing a compact and efficient representation of the network's state.

By combining these two technologies the network could benefit from the security and verification guarantees of PoE and the scalability and efficiency of DAGs and the NICS. This would result in a network design that can handle a large number of transactions while maintaining a secure and verified record of their order and relationships.

9.2 Subnetworks

A subnetwork, or subnet, is a dynamic set of validators working together to achieve consensus on the state of a set of distributed ledgers. Each DL is validated by one subnet, and a subnet can validate arbitrarily many distributed ledgers. A validator may be a member of arbitrarily many subnets. A subnet decides who may enter it, and may require that its constituent validators have certain properties. The Experience platform supports the creation and operation of arbitrarily many subnets. In order to create a new subnet or to join a subnet, one must pay a fee denominated in $XP.

The subnet model offers a number of advantages:

– If a validator doesn't care about the distributed ledgers in a given subnet, it will simply not join that subnet.

This reduces network traffic, as well as the computational resources required of validators.

– Since subnets decide who may enter them, one can create private subnets. That is, each DL in the subnet is validated only by a set of trusted validators.

– One can create a subnet where each validator has certain properties. For example, one could create a subnet where each validator is located in a certain jurisdiction, or where each validator is bound by some real-world contract. This may be beneficial for compliance reasons.

There is one special subnet called the Default Subnet. It is validated by all validators. (That is, in order to validate any subnet, one must also validate the Default Subnet.) The Default Subnet validates a set of pre-defined distributed ledgers, including the distributed ledger where $XP lives and is traded.

9.3 Virtual Machines

A Virtual Machine (VM) serves as a blueprint for each distributed ledger instance. Just like a class is a blueprint for an object in an object-oriented programming language, a VM defines the interface, state, and behavior of a distributed ledger. Various properties of a distributed ledger, including those not explicitly mentioned, are determined by the VM it runs on:

– The contents of a block – The state transition that occurs when a block is accepted – The APIs exposed by the distrbuted ledger and their endpoints – The data that is persisted to disk

We say that a distributed ledger ”uses” or ”runs” a given VM. When creating a distributed ledger, one specifies the VM it runs, as well as the genesis state of the distributed ledger. A new distributed ledger can be created using a pre-existing VM, or a developer can code a new one. There can be any number of distributed ledgers that run the same VM. Each distributed ledger, even those running the same VM, is logically separate from others and maintains its own state.

9.4 Bootstrapping

The first step in participating in Experience is bootstrapping. The process occurs in three stages: connection to seed anchors, network and state discovery, and becoming a validator.

9.4.1 Seed Anchors

For any decentralized network without a predetermined set of identities, a peer discovery mechanism is necessary. Crypto networks typically employ DNS seed nodes, known as seed anchors, which provide a defined set of seed-IP addresses from which network members can be found. In Experience, the majority of seed anchors must be correct for the network to function properly, whereas Bitcoin Core only requires one correct DNS seed node. The set of bootstrap nodes can be provided by the user, but clients may also offer default settings that include reputable actors such as exchanges. Any node can become a seed anchor, and no set of seed anchors can control network entry since nodes can attach to any set of seed anchors to discover the latest network of peers.

9.4.2 Network and State Discovery

After connecting to the seed anchors, a node requests the latest set of state transitions, which we refer to as the accepted frontier. For a chain, the accepted frontier is the last accepted block, while for a DAG, it is the set of vertices that are accepted but have no accepted children. The state transitions accepted by a majority of the seed anchors are defined as accepted, and the correct state is extracted by synchronizing with the sampled nodes. This state discovery process is also used for network discovery. The network’s membership set is defined on the validator chain, so synchronizing with the validator chain allows the node to discover the current set of validators. [5]

10 System Architecture - Network Intelligence Control System

10.1 Components

The system architecture of a Network Intelligence Control System (NICS) can be broken down into the following high-level components:

10.1.1 Data Collection & Analysis

The responsibility of this component is to gather data from a range of sources, including network devices, servers, and applications, and then analyze it to obtain insights into network performance, traffic patterns, usage trends, security threats, and other pertinent metrics.

10.1.2 Network Monitoring

This module is accountable for continuously monitoring the network and triggering notifications when any network problems are identified. It may consist of tools like Network Performance Monitoring (NPM), Security Information and Event Management (SIEM), and User and Entity Behavior Analytics (UEBA).

10.1.3 Network Control

This component is responsible for implementing the policies and rules that govern the network, ensuring optimal network performance, and preventing security threats. It includes tools such as firewalls, intrusion detection/prevention systems, and application delivery controllers.

10.1.4 Network Orchestration

The responsibility of this component is responsible for automating network management tasks, such as configuration changes, policy updates, and provisioning/deprovisioning of network resources. It can include tools such as Software-

Defined Networking (SDN), Network Function Virtualization (NFV), and Network Automation and Orchestration (NAO) platforms.

10.1.5 User Interface

A user interface for network administrators to manage and control the network. It includes tools such as dashboards, reports, and analytics, which enable administrators to gain insights into network performance and make data-driven decisions.

10.1.6 Integration

This module enables integration with third-party tools and systems such as enterprise resource planning (ERP) systems, customer relationship management (CRM) systems, and security information and event management (SIEM) systems.

These components work together to provide network administrators with a comprehensive set of tools for managing and controlling the network.

11 Conclusion

In conclusion, the network we have designed using a combination of PoE, PoES, DSS, NICS, and other innovative protocols has several advantages over existing blockchain and distributed ledger technologies. Our network provides a more scalable, efficient, and secure platform for decentralized applications and services. Additionally, by introducing novel mechanisms for consensus and network discovery, we have improved the resilience and flexibility of our network. It is worth mentioning that this is the first web4 network, which further emphasizes its uniqueness and importance in the emerging decentralized ecosystem. We hope that our work will inspire further research and development in this field, and ultimately contribute to a more open, decentralized, and democratized web for everyone.

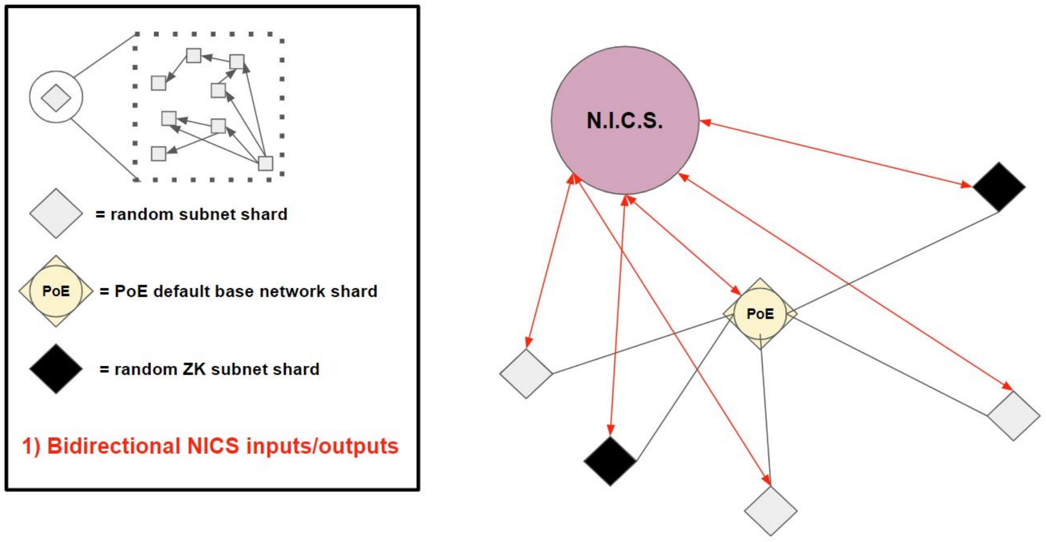

= random subnet shard
PoE
= PoE default base network shard
= random ZK subnet shard
1) Bidirectional NICS inputs/outputs
N.I.C.S.
PoE

Figure 6: Visual representation of a theoretical web4 network

Acknowledgements The author thanks Dr. Harry Gaebler for his help and mentorship throughout the writing and publishing process.

References

[1] E. Cline, *Ready Player One*. New York City: Crown Publishing Group, 2011.

[2] A. Yakovenko, 2017.

[3] V. Buterin, “Slasher: A punitive proof-of-stake algorithm,” Jan 2014.

[4] V. Basu, “An approach to detect ddos attack with a.i.,” Oct 2020.

[5] K. Sekniqi, D. Laine, S. Buttolph, and E. G¨un Sirer, “Avalanche platform,” Jun 2020.