# POCKET NETWORK

**(VERSION 0.3.0)**

Andrew Nguyen
andrew@pokt.network

Michael O'Rourke
michael@pokt.network

Luis C. de León
luis@pokt.network

## ABSTRACT

Reliable node infrastructure is fundamental to the success of any decentralized architecture. It is not practical for application developers to host their own full nodes to provide back-end support for their applications. Currently, the developer community is heavily reliant on solutions consisting of trusted central entities leading to centralization risk for protocols. In part this is due to lack of native relay node incentivization, but also a high level of complexity and inconvenience to the developer. To solve this problem, Pocket Network incentivizes individuals and companies globally to deploy nodes for any blockchain that has developer demand, by financially rewarding them for providing access points for decentralized applications.

This paper describes a decentralized relay network comprised of applications and nodes, who ensure the honesty and integrity of each other by using a combination of cryptoeconomics, cryptographic proofs, and pseudo-random selection algorithms. Pocket Network significantly reduces centralization risk, decentralized application development costs, and lowers the barrier for developers to create peer-to-peer applications for any blockchain.

## 1. INTRODUCTION

The world is headed towards a multi-blockchain future. Infrastructure is a vital part of what allows for the maturation of the blockchain application ecosystem in this vision of the future. In April 2017, Ethereum infrastructure platform, Infura, was handling 175 million API requests per day. Two years later, that number increased by more than 7,000% to over 12 billion requests per day. By lowering the barrier to entry for developers to build decentralized applications and providing easy to use, scalable infrastructure, Infura laid the groundwork for Ethereum's exponential growth in 2017 [1].

This growth however, came with a reliance on centralized blockchain infrastructure solutions creating central points of failure for decentralized applications. This is a significant existential risk to the Ethereum, Bitcoin and broader blockchain community [2], [3], [4]. This risk can largely be attributed to the node incentivization problem, where individuals are not incentivized to run full nodes for any of these blockchains. Bitcoin and Ethereum have infrastructure and developer advantages that other blockchains do not. They have well supported and documented infrastructure and APIs, allowing developers to build applications easily on their networks [5], [6]. New and future blockchains need similar support. Pocket's single interface can provide this much-needed infrastructure support for *any* blockchain.

This paper introduces Pocket Network, a decentralized system that provides an incentive for individuals and businesses to run full nodes. This, in turn, solves the node incentivization problem, relieving infrastructure centralization risks and providing a common interface for all blockchains.

## 2. NETWORK DESIGN

The Pocket Network is comprised of 3 components: Applications, Nodes and the Network Layer. An Application submits Relays, or API requests meant to be routed to any public database node. Nodes service these Relays, by submitting them to the public databases they are meant for, and sending the response (if any) back to the Application. The Network Layer is comprised of all the rules, protocols and finality storage that serve as the backbone of the interactions between Applications and Nodes, including (but not limited to), configuration, record tracking, governance and economic policy.

The mechanism the Network uses to regulate the interactions between Applications and Nodes are Sessions. Sessions are a data structure that are generated following the established Session Generation Algorithm, which uses data from the finality storage layer of the network to pseudo-randomly group one Application with a set of Nodes which will provide service to it for a limited timeframe.

### 2.1 Sessions

A Pocket Network Session is the relationship between an Application and the Nodes that service it during the Session duration within the Pocket Network. This data structure holds the Nodes responsible for the Application along with other Session specific information.

#### 2.1.1 Session Generation

Session Key

A session key is the hash used to calculate the session nodes. Node selection needs to be network-wide pseudo-random, deterministic, and unpredictable. By using the hash of the latest block as seed data for the Session Key generation, the Network can guarantee all of these requirements.

SessionKey Algorithm

```
Hash(
  Data from the latest block (block hash) +
  The Application's Public key +
  The hash of the encoding of the requested Non-Native Chain
  )
= Session Key
```

Session Node Selection

Every active Node registered for the specific Non-Native chain on the Pocket Network is XOR'ed against the Session Key to find the computation distance from the node to Session. The top five (5) nodes are chosen as the Service Nodes for the session.

Amount of Selected Nodes

Mathematically speaking we need an odd number of Nodes for majority consensus. Three (3) is the absolute minimum due to the number needed for a majority consensus. However, all you need is to control two (2) nodes out of all Nodes in the Session (ALLNODES) to win validation.

The probability due to randomized selection without replacement is:

$$P(A \cap B) = P(A)\,P(B|A)$$

Thus the probability in our network is:

$$1/(\text{ALLNODES} * (\text{ALLNODES}-1))$$

If we increase the number of Nodes per session to five (5), the probability becomes:

$$1/(\text{ALLNODES} * (\text{ALLNODES}-1) * (\text{ALLNODES}-2))$$

This means that 5 nodes increases our probability by a factor of ~ALLNODES, with only a linear increase of two (2) Nodes, meaning the performance trade-off is negligible compared to the increased security of the Session.

### 2.1.2 Session Customizations

When an Application registers itself within the Pocket Network it provides additional metadata options that can configure the Session accordingly. A Node will not receive payment for any service they provided while breaking Session rules. This is enforced by the Block Producers in the Finality Storage of the network by verifying all the work reported to the network by Nodes against the session rules. These configurations can be, but are not limited to:

- Required Non-Native Blockchains by the Application.
- Stake allocation per Non-Native Blockchain.
- Application instance service configuration (e.g. Amounts of Relays per session, service expiration).

### 2.1.3 Session Tumbling

Since Session Generation relies on the latest block data, it is relatively trivial to tumble sessions periodically based on how many blocks have passed since the last generation. This tumbling mechanism, allows for much greater Application security, as the same (5) nodes will only service the application for a certain amount of time.

## 2.2 Applications

An Application is any software application, registered on the Pocket Network as such, which submits Relays meant to be routed to any public database node. The rate at which these Relays can be sent through the network is determined by the Application allocated throughput, which is consistent Network-wide. Applications are responsible for the validation of their own data integrity, and can enforce security via cross-check verification of results from multiple Nodes in the network. This cross-check mechanism is called Client-Side Validation.

### 2.2.1 Allocated Throughput

The only way to obtain sanctioned throughput from Nodes within the Pocket Network is by staking the native cryptocurrency within the network. See **Section 3.2** for a detailed specifics on this mechanism.

### 2.2.2 Throughput Utilization and Service Patterns

Due to the architecture of the Session mechanism, Applications are provided five (5) individual Service Nodes per requested chain. The way the Applications allocate their throughput is up to their sole discretion. Though there are many configurations, there are two recommended and anticipated patterns the Applications should be aware of when designing their clients.

Optimization for speed

The first Application service pattern described in this document is an optimization for speed. Each Application designed under this pattern will send distinct Relay requests to each individual Node. In this pattern, there is no Client-Side Validation, but by parallelizing the requests, the Application is able to do far more requests than the security optimization pattern.

Optimization for security

The next Application service pattern described in this document is an optimization for security. In this model, each Application sends the same request to all (5) Nodes. Once the responses from each Node are collected, the Application can perform Client-Side Validation by selecting the answer with the highest majority. Though this is expected to be significantly slower than the speed optimization pattern, the data security guarantees are much higher.

### 2.2.3 Application Authorization Token

An Application signed token is needed by each client to authorize the use of the Application throughput. Similar to other existing token models, such as JSON Web Tokens (JWT) [7], the Application Auth Token (AAT) provides security guarantees around the usage of the service for the application. Similar to the Service Patterns, there are two basic design patterns for AAT distribution that are recommended and anticipated.

Optimize for safety

A simple backend server that distributes signed tokens using the Application's Pocket Blockchain Account. Though this pattern is of higher effort for the Application, it provides the highest security guarantee around their AAT.

Optimize for performance and UX

Embed a token production system within the Application code. This guarantees the smoothest UX for the clients and easiest setup for the Application. However, reverse engineering a source code level token generator can be trivial if obfuscation methods are not applied. The upside to this approach is not having the need to have an additional component in the Application that generates the AAT dynamically while keeping the Application private key secure.

### 2.2.4 Client-Side Validation

In order to ensure the truthfulness of the data received by nodes in the Pocket Network, Applications have the choice to use their allocated throughput to cross check the results and assume that the majority of nodes that return consistent results are being truthful while the minority is lying. This assumption is made thanks to the pseudo-random Session Generation described in **Section 2.1.1**. If they choose to, Applications can submit a "Challenge Transaction" to the Pocket Network using the results of the cross check as proof in order to burn the lying Nodes.

### 2.2.5 Application Optimization

Though each Application is able to customize the design of their Decentralized Application as they see fit, there are several recommended optimizations that should be taken into consideration.

Session Management

It is recommended to cache and store ongoing Sessions to minimize impact on Nodes providing Dispatch services and to improve UX for the application. Session generation can be a high computational task for the network, and too many requests can result in negative security consequences for the Application.

Dispatch Routing Table

It is recommended to build an extensive routing table of Nodes providing Dispatch services in order to increase decentralization of the Pocket Network and to improve security for the Application.

## 2.3 Nodes

Nodes are highly available infrastructure running on existing cloud providers or local data centers. At its core, a Pocket Network Node is an ABCI Application [8], which means it is a Tendermint Core [9] compatible state machine. At a high level, this state machine can be broken down into 3 core components:

1. *Dispatch:* Serves Session information to an Application, serving as the first point of contact between it and the Nodes that will service it.
2. *Service:* Services Relays submitted by Applications, if they belong to the corresponding Session.
3. *Finality Storage:* Stores the Pocket Network state, including account balances, Application and Node records, and contains the business logic to validate work reports submitted by Nodes.

### 2.3.1 Dispatch

Dispatch Nodes, more traditionally known as edge nodes in distributed systems, are Nodes on the network that provide the Dispatch Service to Applications. It is important to note that every active node registered for service (via staking the native cryptocurrency) on the network is a Dispatch Node. Dispatch Nodes serve two functions to Applications:

- *Session Request*
- *Peer Sharing*

#### 2.3.1.1 Session Request

The Dispatch Node runs the Session Generation Algorithm for the specific Application Client and returns the Session information to the client. It is important to note that every session is predetermined pseudorandomly by the seed data passed from the Application Client, meaning that every actor on the network can generate the same session if given the same seed data (using the Finality Storage as the consistency guarantee). This allows each client to know who their Service Nodes are by connecting with any active Dispatch Node on the network.

#### 2.3.1.2 Peer Sharing

In **Section 2.2.5**, the peer routing table is described. This routing table increases both network decentralization and application security, as they can go to multiple nodes to find out their session information.

### 2.3.2 Service

Nodes that service the Relays of the Application clients through the Pocket Network are known as Service Nodes. Service Nodes must be able to scale with the average throughput of the network, or be left behind. Service Nodes must run at least one full non-native blockchain node to provide up-to-date information to the Application Clients upon request.

#### 2.3.2.1 Relays

In the context of the Pocket Network, a relay is a Non-Native Blockchain API request and response transmitted through the Pocket Network.

#### 2.3.2.2 Relay Data Structure

```
Relay(
  Non-native Network Identifier +
  Payload +
  Application Auth Token +
  Increment Counter
  )
```

Non-native Network Identifier

Every non-native network available for service in the Pocket Network has to be properly identifiable via a unique, collision-free hash according to the following parameters, which must be agreed upon off-chain by Service Nodes and made available to the rest of the network:

```
Hash(
  Network Ticker / Short name +
  Network Identifier +
  Network Interface Version
  )
= Non-native Network Identifier
```

Payload

The Relay payload is an arbitrary data packet, formatted according to whatever specification the non-native network interface specification follows. It is not the responsibility of Pocket Network's Service Nodes to interpret or transform this payload in any shape or form. Any errors in formatting, encoding or parameters are the responsibility of the Application.

Application Auth Token

As described in **Section 2.2.3**, an Application Auth Token is the means of authentication of an Application to any Node. In the case of a Relay, the AAT serves as proof that the submitted Relay is serviceable. The AAT can optionally also include service configuration information, such as service limits for particular Application clients, or expiration dates of service provisioning.

Increment Counter

An Increment Counter is an arbitrary integer, representing the amount of Relays serviced during a Session by a Service Node, that is used as the security mechanism for Service Node payment. The signing of an IC can be thought of as an Application client's notarization of work completed. The Application client is incentivized to notarize IC's, because it will be denied service if it doesn't.

#### 2.3.2.3 Relay Lifecycle

1. Application sends a Relay request and (Signed) Increment Counter to the service node (if it's the first IC then it will be total=0)
2. Service Node runs signature verification on the Application Access Token to ensure that the client is authorized.

```
(If signature valid)
        Service node stores the signed IC for Proof of Relay completed
        Service node routes the relay request to the Non-Native chain and executes
        it (storing the response in memory)
(If signature invalid)
        Service node responds with invalid relay signature error code. (END)
```

3. The Service Node signs the response and sends it back to the client with an increment counter
4. Repeat 1-3

#### 2.3.2.4 Relay Batch

A Relay Batch is the product of a Session that is submitted to the Finality Storage. This data structure holds the number of Relays completed, the signed evidence of work completed (increment counter, see **Section 2.4.1.1**), the AAT (see **Section 2.2.3**) and the Session Key (see **Section 2.1.1**). The Relay Batch can be thought of as the Proof of Relay completed for the Service Nodes. The acceptance of a Relay Batch into the Finality Storage confirms the legitimacy of the work done, and automatically mints the native cryptocurrency to the sender of the Relay Batch, according to the applicable monetary policy.

#### 2.3.2.5 Relay Batch Data Structure

```
RelayBatch(
  Completed Relays Count +
  Evidence Increment Counter +
  Application Auth Token +
  Session Key
  )
```

Completed Relays Count

Number of Relays serviced during the Session.

Evidence Increment Counter

By using the Relay Batch Evidence Selection Algorithm, the Service Node produces a hash using the following parameters:

```
Hash(
  Session Key +
  Total # Relays +
   Node Public Key
  ) = Evidence Selection Hash
```

The resulting Evidence Selection Hash gets XOR'ed to find the closest index in range between 0 and the Completed Relays Count, which is used to select the corresponding Evidence Increment Counter.

Application Auth Token

The AAT (**Section 2.2.3**) corresponding to the selected Evidence Increment Counter.

Session Key

The Session Key (**Section 2.1.1**) for the serviced session.

#### 2.3.2.6 Relay Batch Lifecycle

1. Service Node computes the Relay Batch data structure.
2. Service Node sends the Relay Batch as a transaction to the Finality Storage layer.
3. Transaction gets picked up by Leader from the Finality Storage mempool, and gets validated according to the Relay Batch Evidence Selection Algorithm.
4. One of two outcomes can happen depending on the validation result:
    a. The transaction is valid: the protocol automatically mints new native cryptocurrency corresponding to the applicable monetary policy of the network.
    b. The transaction is invalid: If the transaction doesn't match the expected result of the Relay Batch Evidence Selection Algorithm, the responsible Service Node gets burned and/or punished according to the applicable monetary policy of the network and other network rules.

### 2.3.3 Finality Storage

In order to secure consensus amongst Applications and Nodes around infrastructure provisioning, a distributed, secure and immutable database publicly accessible to any participant of the network proves of utmost necessity. So in order to realize this component of the Pocket Network, the blockchain [10] database structure was selected, specifically the Tendermint Core implementation.

Tendermint provides Byzantine-fault tolerant state machine replication. The Generic Application Interface (ABCI) is what allows Pocket Network compatible software to interact with the Network as long as the protocol rules are followed.

#### 2.3.3.1 Security Model

The security model for the Finality Storage layer revolves around Proof of Stake, where every Node registered into the Network participates in a Validator Set, from which a weighted, pseudo-random, deterministic selection for the Block Producer role, which will allow them to produce the next block in the chain. Because every block gets validated by every other peer upon production, Block Producers have no incentive to produce invalid blocks, because if they attempt to do so, they will be punished according to applicable monetary policy and/or network protocol rules.

#### 2.3.3.2 Transactions

All of the fundamental interactions in the Pocket Network are based on using the Finality Storage as the single source of truth for all of the Network participants. In order to add to this state participants have to submit transactions to Nodes through a standardized versioned interface, from which Nodes will propagate them and they will get included in blocks by Block Producers.

Transaction Data Structure

```
Transaction(
  Recipient(Optional) +
  Signature +
  Transaction Type +
  Transaction Payload +
  Transaction Nonce
  )
```

- Recipient (Optional): The recipient of the Transaction. Depending on the Transaction Type, this field will be optional.
- Signature: The notarized signature from the sender of the transaction, produced using the agreed upon ECDSA [11] standards of the Network.

- Transaction Type: Because the Pocket Network is an application-specific network and not a generic computing platform, the types of transactions allowed will be finite and limited to the ones that are contained within this specification, or any others agreed upon by the Network.
- Transaction Payload: Arbitrary data field that will be encoded depending on the implementation of the accompanying Transaction Type. If the Transaction Type and Payload don't match, the Transaction will be considered invalid.
- Transaction Nonce: A unique auto-increment integer that identifies the order in which this transaction is sent from the sender. The Nonce sequence will be unique per account, and helps mitigate double spend and inconsistent state attacks that could possibly be inflicted on the Network state.

Transaction Types

The following list isn't considered final, but a baseline of the Transaction Types required to fulfill the specifications and behaviours described throughout this paper.

- *Cryptocurrency Transfer*: The mainstay transaction that allows an account to send the native cryptocurrency from its balance to another.
- *Application Stake*: The entry point to register as an Application actor in the Network. Locks tokens from the account balance into a bond within the Network to provide an allocation of throughput for Relay execution via the Network Nodes. The rules of the bond are determined by applicable monetary policy and/or protocol rules.
- *Node Stake*: The entry point to register as a Node actor in the Network. Locks tokens from the account balance into a bond within the Network to earn the right to participate in the Dispatch, Service and Block Production activities. The rules of the bond are determined by applicable monetary policy and/or protocol rules.
- *Unstake*: This transaction type allows any account to undo any bonds currently active in the Network. The rules and conditions for unstaking are determined by applicable monetary policy and/or protocol rules.
- *Relay Batch*: As described in **Section 2.3.2.4** of this paper, a Relay Batch is the means of reporting work done by Service Nodes. Relay Batch transactions are validated by Block Producers, by using the Relay Batch Evidence Selection Algorithm described in **Section 2.3.2.5**. The change of state in the network upon validation of this transaction can be either reward or punishment to the Transaction Sender, as determined by applicable monetary policy and/or protocol rules.

Transaction Cost

In order to avoid spam and oversaturation of the Finality Storage, to include a transaction in a block, the Transaction Sender has to pay a cost in the form of the native cryptocurrency coming from their balance. The destination of these tokens will be decided by applicable monetary policy and/or protocol rules (see **Section 3.4.2**).

# 3. ECONOMIC MODEL

Pocket's economic model involves a continuous, inflationary token issuance that utilizes staking and burning for long-term economic sustainability. The native cryptocurrency of Pocket is POKT. The core concepts in the economic model are:

1. Unit of Work
2. Staking
3. Inflation
4. Burning

These economic primitives are meant to enforce economic sustainability through fair value transfer among all participants.

## 3.1 Unit of Work

Each unit of work is a valid Relay agreed upon by Service Nodes and Client-Side validation. The protocol rewards the participants with POKT. It is important to note that unlike Proof of Work or Proof of Stake the unit of work in Pocket Network is separated from how the nodes reach a consensus on the blockchain. Validation of relays happens off-chain through the Service Nodes and the client, please refer to **Section 2.2.4** for more details.

## 3.2 Staking

Staking is the locking up of the POKT cryptocurrency within the protocol for a minimum time period. Pocket's economic model is designed to have as much POKT staked within the system as possible. There are two reasons a participant would stake:

1. To pay for throughput as an Application
2. To participate as a Service Node and Finality Storage Consensus

Applications pay in advance for relays to non-native blockchains by staking POKT. Paid through the opportunity cost of leveraging those funds to access Service Nodes within the network and through dilution each time they use the service. The amount of Relays allotted to an Application is directly

dependent on the number POKT staked. The protocol throttles the number of relays a developer can send to a node in a given time period based on how much POKT is staked. After an initial bonding period, the developer may exit the network by starting the unstaking process of their POKT.

The rate at which Applications must stake to access throughput will be automatically adjusted through the protocol's Participation rate. The protocol's Participation rate is defined as the percentage of staked POKT. Once an Application stakes POKT at a certain Participation rate, that rate will be locked indefinitely unless POKT is unstaked and restaked again.

Nodes stake POKT to ensure the security of the network and earn rewards by validating Relays and transactions. All nodes must have a minimum amount of POKT staked before participating, this stake is their incurred risk; their staked POKT will be burned if the protocol's rules are not followed.

### 3.3 Inflation Reward

The POKT cryptocurrency is permanently inflationary, with block rewards based off of a unit of completed work. Relay Batches are submitted and inflation reward is awarded to the corresponding parties involved. See **Section 2.3.2.4** for more information about Relay Batches.

The reward for each validated Relay is split as follows.

1. 89% to the Service Node who has completed a validated Relay
2. 10% to Pocket DAO
3. 1% to Service Nodes participating in Finality Storage Consensus

### 3.4 Burning

From a security standpoint, burning is the mechanism that economically disincentivizes bad acting. From an economic standpoint, burning is the mechanism that reduces the total supply of the POKT token. Burning is critical to avoid hyperinflation over the long run. At network maturity, the total supply will be slightly deflationary or as stable as possible through the following burning mechanisms.

#### 3.4.1 Service Nodes Misbehaving

If a Service Node miscasts a vote on a Relay, the protocol will burn the equivalent amount of POKT that would have been its inflationary reward from the work done. Should a Node continue miscasting votes, burning will continue, becoming more severe until the Node has run out of POKT.

##### 3.4.1.1 Incorrect Block Validation

There are two primary functions that lead to burning within the Tendermint consensus protocol - Evidence and Liveness [10]. Both are defined through the custom ABCI application as defined in **Section 2.3**.

### 3.4.2 Transaction Fees

Traditionally, the expectation is that at maturity, protocols will run on these transaction fees. Due to Pocket's nodes relying on Relays and inflation as their primary source of income, this is not needed as a long term goal for the protocol.

Every Transaction to the Pocket blockchain incurs a transaction fee. 99% of this fee will be burned, where 1% goes to the node including the transactions in the block. This 1% is to ensure leader elected nodes are economically incentivized to include transactions submitted to the network, into the latest block.

### 3.4.3 DAO Proposals

To avoid spam and sybil attacks on the on-chain DAO, there will be a minimum amount of POKT burnt to submit proposals.

### 3.4.4 Burning Application Stake

Burning bad actors, transaction fees and DAO proposals alone is not enough to balance out any inflation created by Applications using the protocol. Burning Application stake is what ensures Pocket's long term economic sustainability and replaces fees as a full circle value transfer mechanism. While Service Nodes do not depend on fees over the long run, Application stake being burned ensures that the total supply of POKT remains stable or deflationary in an equilibrium state. By using factors such as protocol-wide participation rates to trigger burning Application Stake, early Applications participating in Pocket will derive more value out of the protocol than late entrants.

## 4. GOVERNANCE

A mechanism for human governance is critical for creating a fault-tolerant system that ensures the longevity and scalability of the Pocket Network. This mechanism is Pocket's Decentralized Autonomous Organization (DAO). To uphold total fault tolerance, we are designing the Pocket DAO to minimize reliance on any other blockchain (by executing all DAO actions on the Pocket

blockchain) and to minimize reliance on any other tool (by communicating all DAO actions through an agnostic governance interface).

### 4.1 Interface

Tendermint's RPC module enables us to write data to the Pocket blockchain (by submitting transactions) and to read data from the Pocket blockchain (by submitting calls). These two directions of communication are the fundamental building blocks of our governance design, because if the DAO can read the blockchain, then it can make informed decisions, and if the DAO can write to the blockchain, then it can ensure that its decisions are final (included in the Finality Storage layer).

To facilitate this communication between the Pocket DAO and the Pocket blockchain, we must design a modular governance interface that can:

1. *Write:* Standardise governance outputs (i.e. any decision made using any organizational technology) and submit them as a transaction to the RPC module
2. *Read:* Submit relevant calls and relay the returned data to DAO members

### 4.2 Writing (Transactions)

This whitepaper defines a limited set of transactions which are necessary for the Pocket Network specification. These are likely to be actions that our governance interface facilitates for the DAO, for example, to execute a transfer of funds from the DAO's Pocket account. However, we may also need to introduce new transaction types specifically for governance. For example, an emergency fallback mechanism would include a simple transfer of funds from the DAO's compromised account, but may require more complex actions that are as yet undefined.

### 4.3 Reading (Calls)

The governance interface should be able to relay any data that the DAO needs to make informed decisions. These will fall under two main buckets:

1. *Network Data:* Data about network activity and economic metrics such as the stake rate, burn rate, and emission (inflation) rate.
2. *Governance Data:* Determined by the transactions we define for the DAO, e.g. historical data about all fund transfers initiated by the DAO. This could include metadata about these transactions, but we'll prioritise blockchain data since in theory, anyone should be able to compute the metadata off-chain.

### 4.4 Implementation

In designing this governance interface, our priority will be connecting the Pocket blockchain to a treasury management system, which will allocate 10% of all POKT inflation rewards. We will then turn our attention to facilitating DAO decisions about monetary policy and protocol upgrades.

## 5. FUTURE CONSIDERATIONS

While Pocket's design doesn't enable direct cross-chain communication between protocols through mechanisms like atomic swaps and Hashed TimeLock Contracts (HTLCs), Pocket is in a unique position to act as connective tissue for any open-source protocol. The expectation is that thousands of Pocket Nodes are running dozens or hundreds of full nodes for other blockchains. This enables novel solutions to common areas of research for the blockchain industry.

**Multi-chain Applications and Universal Wallets**

Through Pocket's integration with existing developer SDK's, interacting with many chains at once through the same web or mobile user experience is easily accomplished. The advent of blockchain agnostic Web 3 browsers can simply add in Pocket as a dependency and developers can quickly integrate features from decentralized applications on completely different blockchains.

Since Pocket Network consists of a worldwide network of nodes, pluggable to any chain, a universal wallet is now a feasible product. An entire crypto portfolio on a single wallet file is not far off.

**Oracle Networks**

Pocket plugins can easily be built to support any publicly verifiable database. Instead of Service Nodes accessing their local copy of a blockchain, they can access public APIs such as sports scores, weather, or even political events. Most attempts at agnostic Oracle protocols force application developers to conform to the specifications of their protocol. Pocket's plugin system acts as a dependency for existing applications, making a truly censorship resistant Oracle much simpler to implement.

**Decentralized Exchanges**

Millions of dollars of research have gone into atomic swaps and Hash Time-Locked Contracts to enable cross-chain communication and on-chain decentralized exchanges. Due to Pocket Validator Nodes having to stake POKT, they can act as a trustless conduit to provide liquidity across blockchains, without the technical difficulties of on-chain communication.

**Meta Reputation**

The challenge with blockchains is that they are siloed from each other, and have no way to communicate data across themselves. The record of a relationship between an Application User and the blockchain they are communicating with is effectively being recorded through the Pocket blockchain. If this relationship was backed through individual staking of POKT and able to be transferred across chains, meta reputation would unlock the potential for a number of promising research areas such as Universal Basic Income, identity, and decentralized financial applications.

## 6. ACKNOWLEDGMENTS

The Pocket Network derives from countless ideas and projects that existed far before its inception. There are many individuals who must be given considerable credit, thanks, and praise. Firstly, to the co-founders of Pocket Network Incorporated: Pabel Nuñez and Valeria Benitez Florez for their expertise in their respective fields and for envisioning a groundbreaking idea of this magnitude. To Tracie Myers for her protocol economic analysis. And a personal thanks to Shawn Regan for numerous discussions and contributions to the core protocol. Without the help of many brilliant people, the Pocket Network would not exist.

## 7. REFERENCES

[1] M. Wuehler, "Scalable Web3 Infrastructure With Infura", Rice University, 2017.
[2] J. Pitts, “ethresear.ch”. 26-Feb.-2018.
[3] J. Ray, “EIP: Reward for clients and full nodes validating transactions #908”. 28-Feb.-2018.u
[4] B. Bernstein, “Twitter”. 10-Jul.-2018.
[5] “Ethereum Javascript API”. 30-Jul.-2015.
[6] W. Bins, “Bitcoin Developer Documentation”. 2009.
[7] https://jwt.io/
[8] https://github.com/tendermint/tendermint/tree/master/abci
[9] https://tendermint.com/docs/introduction/what-is-tendermint.html
[10] https://cosmos.network/docs/spec/slashing/04_begin_block.html#evidence-handling
[11] http://www.cs.miami.edu/home/burt/learning/Csc609.142/ecdsa-cert.pdf

## 8. GLOSSARY

Application User

An individual or group using an app built by an application developer.

Application Developer

An individual or group who is building an application that utilizes a blockchain.

Blockchain Developer

An individual or group who is building a blockchain or decentralized digital database.

Burn

The deletion of the native cryptocurrency from the blockchain ledger.

DAO

A decentralized autonomous organization that is governed by the protocol's primary actors.

Dispatch Node

The first point of contact between the network and the Application User.

Service Node

A Node registered in the Network servicing Relays for Applications.

Blockchain Node

A Node registered in the Network to participate in the Finality Storage consensus by producing blocks.

Mint

Newly created POKT proportional to the number of relays completed.

Network Improvement Proposal

A formalized publication specifying Pocket propositions.

Pocket Developer Plugin

An extension of the Pocket Developer SDK that provides a single interface for developers to connect to blockchains.

Pocket Developer Toolkit

A single interface for emerging blockchains to be supported by the Pocket Network.

POKT

The native cryptocurrency needed to participate within the Pocket Network.

Session

An ongoing relationship between an application developer and the Pocket Network.

Stake

The process of putting POKT in escrow to pay for throughput as an Application or to participate as a Node in the Pocket Network.

Relay

A blockchain API request and response transmitted through the Pocket Network.

Relay Batch

The product of a session that is submitted to the blockchain.

Finality Storage

Refers to the public database replicated across all of the Pocket Network compatible Nodes.

API

Application Programming Interface, usually refers to a standardized interface of functions callable from other applications or software components.

XOR

Abbreviature for “Exclusive Or”, it’s an algorithm that allows the calculation of the distance between 2 hashes in a time-constant manner.