# POCKET NETWORK

**(VERSION 0.1.0)**

Andrew Nguyen
Pocket Inc.
andrew@pokt.network

Michael O'Rourke
Pocket Inc.
michael@pokt.network

Luis C. de Leon
Pocket Inc.
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 full nodes to provide back-end support to their application. Currently, the developer community is heavily reliant on solutions consisting of trusted central entities leading to centralization risk for protocols. This is due to not only a 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 a federation of nodes, ensuring the honesty and integrity of every read and write request performed within the network. 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 maturation of the blockchain application ecosystem and a multi-blockchain future. In April 2017, Ethereum infrastructure platform, Infura was handling 175 million API requests per day. Fourteen months later, that number increased by more than 3,400% to over 6 billion requests per day. By lowering the barrier of 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].

Centralized blockchain infrastructure solutions lead to central points of failure for decentralized applications. This is a significant worry to the Ethereum, Bitcoin and broader blockchain community [2], [3], [4]. Largely due to the node incentivization problem, where individuals are not incentivized to run full nodes for any blockchain.

Bitcoin and Ethereum have infrastructure and developer advantages that other blockchains do not. They have well supported and documented infrastructure and API's,

allowing developers to build applications easily on their networks [5], [6]. New and future blockchains need similar support. Pocket's single interface can provide infrastructure support for any blockchain.

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

## 2. BACKGROUND

Pocket takes ideas and concepts from various other blockchains and distributed system technologies and puts them together in a novel way:

### 2.1 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) [7] has served as an inspiration for many consensus algorithms and blockchains in production such as Hyperledger Fabric [8] and Tendermint [9].

*2.1.1 Hyperledger Fabric -*

Hyperledger Fabric is an enterprise blockchain that has several plugins for consensus, one of them being Simplified Byzantine Fault Tolerance (SBFT):

1. Can handle up to ⅓ Byzantine validators and replicas
2. Requires 3 phases:
    a. Endorsement phase driven by policy upon which participants endorse a transaction.
    b. Ordering phase accepts the endorsed transactions and agrees to the order.
    c. Validation phase takes a block of ordered transactions and validates the correctness of the results.

*2.1.2 Tendermint -*

Tendermint's consensus algorithm is similar to Hyperledger, but inspired by the DLS algorithm [10]. A key difference is the idea of leaders getting chosen in a round robin manner. Consensus is reached in three phases:

1. Blocks are proposed by the correct proposer and gossiped to the other proposers
2. A pre-vote that ensures enough validators have witnessed the proposed block
3. A pre-commit that when ⅔ of validators have signaled, the block is officially committed

Tendermint introduces a locking mechanism which determines how a validator may pre-vote on a block or pre-commit to a block based on previous pre-votes and pre-commitals.

### 2.2 Dash Masternode Governance

The most successful and well known masternode blockchain is Dash [11]. Using the classic Nakamoto Consensus algorithm [12], Dash implemented a layer above where node operators can stake 1,000 coins as collateral for 45% of the mining reward and additional governance capabilities within the protocol.

The Masternode layer has created two key innovations for a sustainable, decentralized cryptocurrency:

1. Fixes the node incentivization problem by giving participants a piece of the block reward for running full archival nodes.
2. Enables sustainable governance of the protocol via the DAO, whereby 10% of the mining reward gets allocated for proposals which are then voted on by masternodes.

### 2.3 Protocol Token Issuance

Protocols have three types of token issuance [13]:

1. Bitcoin and protocols like it release tokens gradually over time, yet there is a known fixed supply that will never exceed a certain total.
2. Protocols built as a set of smart contracts such as Augur [14] or 0xProject [15] issue a fixed supply at launch.
3. Protocols with continuous token models such as Ethereum [16], Livepeer [17], and Aragon [18], with a fixed but adjustable inflation rate and the total amount of tokens being uncertain.

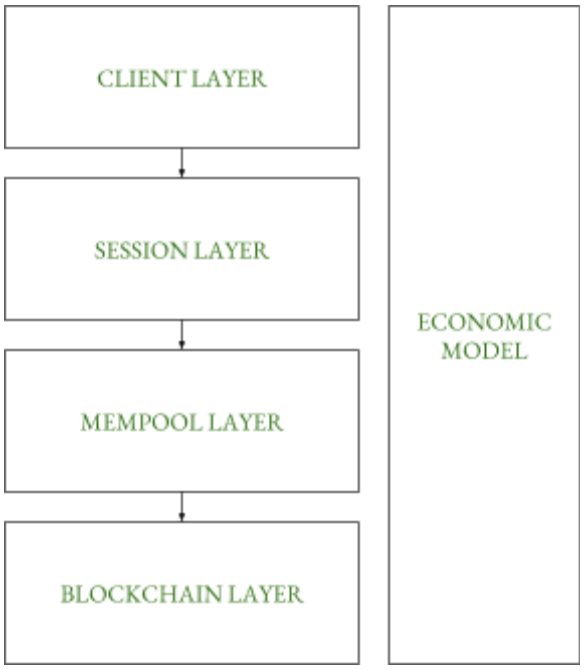
Each protocol is incentivizing different types of economic behavior based on their issuance model.

## 3. POCKET DESIGN

Within the context of this document, the Pocket Network architecture is represented through the characterization of four abstraction layers.

1. Client Layer
2. Session Layer
3. Mempool Layer
4. Blockchain Layer

Similar to that of other conceptual computing models, the four layers of the Pocket Network interact and communicate with adjacent layers [19].



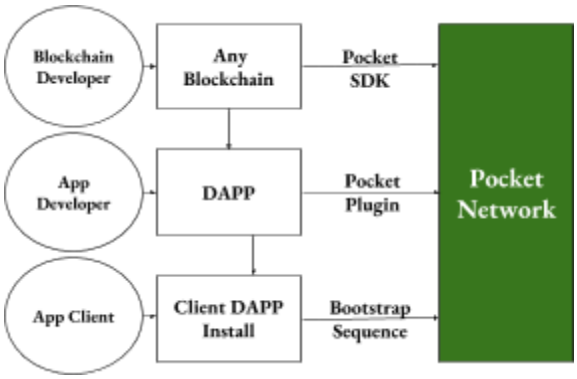

Though there are many mechanisms and complexities within the four layers, it is essential to start with the high-level structural design to understand the network as a whole. For the sake of clarity and breadth of intended

audience, this paper takes a top-down approach in explaining the abstraction layers.

### 3.1 Client Layer

In the Pocket ecosystem, there are three types of end users:

1. **Blockchain Developer**
2. **Application Developer**
3. **Application Client**




#### *3.1.1 Blockchain Developer*

Within the context of the Pocket ecosystem, a blockchain developer is an individual or group who is building a blockchain or decentralized digital database. Any blockchain developer can allow their chain to be accessed through the Pocket Network by following a two-step process:

1. Utilize the developer toolkit to build a Pocket Relay Node Plugin (allows the Pocket Network to relay transactions from the blockchain).
2. Utilize the developer toolkit to build a Pocket Developer Plugin (allows developers to have a single interface to develop DApps on the new network).

*The developer toolkit is a SDK built by Pocket Incorporated. The toolkit provides a single interface that allows emerging blockchains to be supported by the Pocket Network.*

Blockchain Developer Toolkit

Pocket is "Blockchain Agnostic," meaning Pocket protocol can provide infrastructure to any blockchain.

This plugin system introduces a single interface that allows any decentralized architecture to access the global Pocket network, removing the need for blockchain developers to maintain their own nodes.

#### *3.1.2 Application Developer*

Within the context of the Pocket ecosystem, an application developer is an individual or group who is building an application that utilizes a blockchain. For an application developer to access Pocket's global infrastructure of full nodes, two things must happen:

1. Use the Pocket Developer Plugin for whatever blockchain the DApp requires.
2. Stake X pocket for P*X utilization (P is a dynamically changing variable based on the market value and overall usage).

*A Pocket Developer Plugin is an extension of the developer SDK. The plugin system provides a single interface that allows developers to connect to whatever blockchain they want to use within their DAPP.*

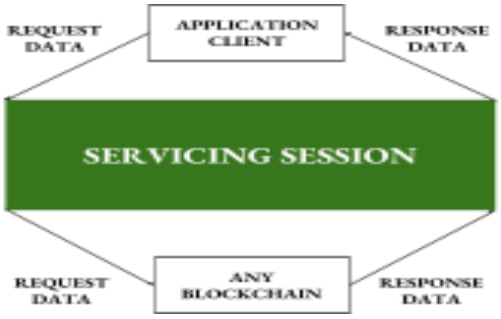
Pocket Developer Plugin

To provide complete support to developers who require less common or multiple blockchains, Pocket Inc. introduces a plugin system that supports any blockchain. From the developer's perspective, a few lines of familiar code is the difference between a transaction on Ethereum or a transaction on Bitcoin.

#### *3.1.3 Application Client*

Within the Pocket Ecosystem, a client is any person using an app built by an application

developer (defined in 3.1.2).
See the figure below for a high-level relationship between a client and the Pocket Network.




Application Client Bootstrapping Sequence

The Pocket bootstrapping sequence for an application client consists of a two-tiered approach.

1. The first tier of the Pocket bootstrapping sequence is centralized DNS seeds hosted by the network.
2. The second tier is a master list of the connected IP's hosted on the Pocket Incorporated block explorer.

For the sake of efficiency, a Pocket Edge Node is selected by geolocation proximity.

### *3.1.4 Client Layer Nodes*

*NOTE: It is important to note that the separation and differentiation of nodes only refer to the roles that nodes play within the abstraction layer and not a tangible divide. Thereby, for clarity, nodes are delineated based on the function that they perform within the specific abstraction layer.*

```
Node interface{
  Hash uniqueID;
  // identifier of the node

  Geolocation geo;
  // location of the node based on IP

  PeerList peerlist;
  // a list of other nodes

  SessionList sessions;
  // a list of ongoing Servicing
Sessions
}
```

The first point of contact between the application client and the Pocket Network is an Edge Node.

```
EdgeNode interface extends Node{
  boolean SessionExists(Hash devID);
  // Edge Node checks for ongoing
sessions for a developer

  void NewSession(Hash devID, Geo
geoLoc, List blockchains);
  // Edge Node creates a new session
for a developer

  void AddClient(Hash uniqueID, Hash
devID);
  // Edge Node adds a client to an
ongoing session
}
```

The Edge Nodes are responsible for connecting application clients with a Servicing Session. This process is known as the dispatching sequence.

1. The application client requests a connection through the client bootstrapping sequence and finds the nearest Edge Node.
2. The application client passes its geolocation information, its unique identifier, the developer id, and the public keys from any given wallet that

is writing transactions to a hosted blockchain.
3. The Edge Node connects the client to the given Servicing Session.

### *3.1.5 End User Functionality*

At its highest level, the client layer encompasses the functionality of the network from an end user standpoint.

As a blockchain developer, the Pocket Network enables anyone to read from and write to the newly developed blockchain without needing to understand the underlying complexities.

As a DApp developer, the Pocket Network establishes an entirely new realm of blockchain application possibilities by providing a decentralized and trustless relay network.

## 3.2 Session Layer

A Servicing Session is an ongoing relationship between an application developer and the Pocket Network. A Servicing Session is linked by a developer public address which application clients use along with their unique identifier during the dispatching sequence.

The figure below is the most primitive form of a Servicing Session.


### *3.2.1 Session Layer Nodes*

To understand the capacity and extent of node involvement within the session layer, nodes are abstracted into three distinct forms.

1. Edge Node
    a. Nodes that are responsible for creating Servicing Sessions by selecting the Service Nodes and pseudo-randomly generating the Validator Nodes are Edge Nodes.
2. Service Node
    a. Within the session layer, nodes that service the API requests of the application clients and relay the data through the Pocket Network are Service Nodes.
3. Validator Node
    a. Nodes that verify the requests from the application clients and the data from the Service Nodes are Validator Nodes.

<u>Edge Node</u>

Within the session abstraction layer, the role of a Pocket Edge Node inherently extends from the dispatching sequence as defined in the client layer.

As previously stated, the Edge Node is responsible for connecting application clients with Servicing Sessions. However, if a Servicing Session does not currently exist for a specific developer, the Edge Node is responsible for creating one. The Edge Node generates the validator set by a publicly verifiable pseudo-random algorithm described below.

```
Validator Set = Hash(N Block hash + DevID)
```

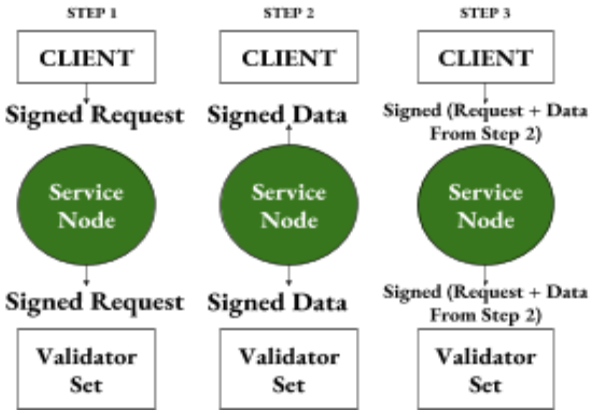
The selection of the Service Nodes is based on geolocation proximity and hosted blockchain specifications.

```
EdgeNode interface extends Node {
  function FindServiceNode(Hash devID);
  // finds service node for client based off location and blockchains needed

  function GenerateValidatorNodes(Hash devID);
  // responsible for generating random Validator Nodes for a new session
}
```

Service Node

The role of a Service Node within the session layer is further abstracted into five steps.

1. Receive a data relay request from the application client and confirm the digital signature.
2. Execute the relay request on the hosted blockchain.
3. Broadcast the data and request to both the application client and the Validator Nodes.
4. Repeat steps 1-3.



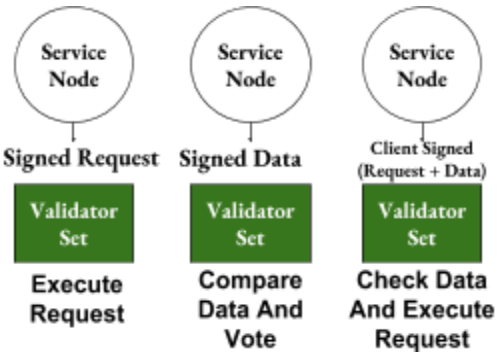

```
ServiceNode interface extends Node{
  function confirmSignature(Hash signature);
  // confirm signature from the application client

  function  digitalSignature(Data data);
  // signs the request receipt and response data

  function executeRequest(List blockchains);
  // service node executes request from client on hosted blockchains

  function relayRequest(List validators);
  // service nodes relay request from client to validators

  function relayData(Client cli, List validators);
  // service nodes relay data to client and validators

  function reimburse(Amount x);
```

```
  // service nodes can submit
proposals to DAO for compensation of
costs
}
```

Validator Nodes

The role of the Validator Nodes is abstracted into seven steps.

1. Receive a relay request from the intermediate Service Node and verify the client's digital signature.
2. Execute the relay request on the hosted blockchain.
    a. If it is a read request, synchronously execute the read on the hosted blockchain.
    b. If it is a write request, asynchronously wait for data confirmation
3. Receive signed relay receipt from the Service Node and confirm signature.
4. Compare Service Node data with step 2 data.
5. Vote on correctness and integrity.
6. The majority consensus receives positive karma (a scoring system described in section 3.3.1), and the minority consensus receives negative karma scoring.
7. Repeat steps 1 - 6.
    a. In step 1, if the request contains data received from the previous relay, compare data.
8. Submit relay batch when finished.




```
ValidatorNode interface
extends FederatedNode {
  function validateData();
  // Validator Nodes validate the
data received from the service nodes
by comparing it with their results

  function vote();
  // validators vote on integrity and
accuracy of data; this allows the
karma scoring of the session by the
protocol

  function signRelayBatch();
  // validators sign the batch of
relays from any given session

  function submitRelayBatch();
  // validators submit the signed
relay batch once consensus is reached
}
```

Relay Batch

The product of a session in the Pocket Network is known as a relay batch. The relay batch is a single transaction within the Pocket Network that summarizes the work and karma scoring of a session completed within the confines of a maximum amount of time or relays.

```
RelayBatch interface extends
```

```
Transaction{
  Hash developerID;
  // the developerID;

  BigInteger relaysCompleted;
  // total number of relays completed

  List serviceNodes;
  // the ids of the service nodes

  List validatorNode;
  // the ids of the Validator Nodes

  Hash validatorNodeSignatures;
  // signatures of the Validator
Nodes participating

  Score score;
  // structure holding the scores of
the session participants. This
structure also allocates mint
}
```

Developer Policies

Client access of a developers allotted throughput was a major security concern throughout the development of the core protocol. To maximize developer flexibility and control, Developer Policies allow for specific guidelines, rules, and regulations that will be enforced by the validators of the Servicing Session. To understand Developer Policies better, it is helpful to compare the concept to Amazon's AWS policies and permissions [20].

### 3.3 Mempool Layer

Between the session and the blockchain layer exists an abstraction layer known as the mempool. Node intercommunication, off-chain validation, liveness checks, and relay consensus are all features of the mempool layer.

The mempool is a P2P internal table structure that is held by all nodes. This structure holds important network information and potential blocks that could be submitted to the chain after a consensus among Federated Nodes upon validation.

Mempool contains

1. Peers
2. Open sessions
3. Active Validator Nodes
4. Active Service Nodes
5. Pending block
6. Designated minter

```
Mempool interface {
  List Peers;
  //  list of peers

  List OpenSessions;
  //  list of open sessions

  NodeList validators;
  //  list of active Validator Nodes

  NodeList service;
  //  list of active service nodes

  Block pending;
  //  pending block

  Node minter;
  //  designated minter

  function CheckTX();
  //  crosscheck transactions based
on rules and state conflicts

  function BroadcastChanges();
 // mempool synchronization must  be
achieved. Project any changes using
peer list
```

```
}
```

### *3.3.1 An Introduction To Federated Nodes*

There are three distinct characteristics of Federated Nodes within the Pocket Network.

1. Federated Nodes have a significant economic stake within the network.
2. Federated Nodes are the only block producers.
3. The network security relies on the integrity and solidarity of the Federated Nodes.

In a sense, Pocket is a PoS network with a governing Federation. However, Federated Nodes include significant additions to traditional PoS delegates for added security and specification.

```
FederatedNode interface extends Node{
  boolean receivesMint = true;
  // Federated Nodes receive mint for
signing blocks, voting on DAO
proposals, and minting blocks

  function signblock(Block
pendingBlock);
 // Federated Nodes must reach
consensus when validating a pending
block

  function exitFederation();
  // Federated Nodes can exit the
federation gracefully by simply
submitting a notification transaction
to the Pocket Network

  function rejoinFederation();
  // Federated Nodes can re-enter the
federation gracefully by simply
submitting a notification transaction
to the Pocket Network
}
```

There are three admission requirements to become a Federated Node:

1. High karma
2. High stake
3. Inveterated Participation (aged)

Karma

Up to this point, scoring, also known as karma, has been brushed over in the interest of partitioning concepts and expediting understanding. Within a session, Service and Validator Nodes are scored based on a majority consensus among participants. Scoring is cumulative and is tethered to the unique identifier of the node. Achieving a certain high karma threshold is the first requirement to become a Federated Node.

High Stake

Another admission requirement of becoming a Federated Node is a high requirement of stake. The purpose of the higher stake is to increase accountability and disincentivize bad acting.

Inveterated Participation

The last admission requirement of becoming a Federated Node is inveterated participation, informally known as node age. Similar to requirements of a Proof of Minimum Aged Stake Network [21], Federated Nodes must have an extensive and active history of involvement.

Federated Vs. Validator Nodes

By virtue of the robust nature of Pocket nodes,

the abstraction of Validator Nodes thus far has been independent of the explanation of Federated Nodes. However, because the security of the network relies on the regulation of relays, Validator Nodes are required to be Federated Nodes.

### *3.3.2 Session And Mempool Layer*

As defined earlier in the session layer, validators submit a relay batch to the mempool. More specifically, the Validator Nodes submit relay batches to Federated Nodes for validation. Following validation, the relay batch is added to the temporary block held within the mempool structure.

### *3.3.3 Transactions*

Though relay batches are the primary function of the network, they are not the only transaction type on the Pocket Network.

```
Transaction interface {
  type BatchRequest;
  // contains # of relays, mint
allocation, karma scoring,
dispatcher, and devID

  type Staking;
  // stake Pocket on behalf of a node

  type Unstaking;
  // unstake Pocket on behalf of a
node

  type FedJoin;
  // join the federation after
reaching a karma score threshold

  type FedExit;
  // exit the federation voluntarily
(no burn)

  type Blacklist;
  // node blacklisted (Network call
only)

  type Burn;
  // burn stake (Network call only)

  type Transfer;
  // transfer Pocket

  type Minting;
  // mint allocation

  type UpdateNodeData;
  // update node data (supported
blockchains, maturity, etc.)
}
```

Transactions are rejected or accepted based on a set of rules:

1. Formatting, similar to that of the Bitcoin Protocol Rules [22].
2. Checking any state conflicts against the blockchain.

## 3.4 Blockchain Layer

The purpose of the Pocket blockchain is to maintain a decentralized ledger of the state of consensus of all the transactions relayed by the Pocket nodes. Much like other traditional cryptocurrency blockchain applications, the Pocket blockchain can be thought of as the ultimate source of truth, meaning any transactions confirmed on the Pocket blockchain are provable and authentic.

Block Proposing

A pseudorandomly (see formula below) chosen Federated Node is selected as the miner. In the situation of a missed block, the difficulty algorithm adjusts, the node responsible is unfederated, and the chain stalls another X seconds. A UTC timestamp, a block header,

and the correct miner, are the required qualities of a valid block.

```
Pseudorandom Delegated Miner Formula =
Hash(Block N Header) + Hash (Block N-1)
Take Closest Fed Node ID
```

Block Time

Block time within the Pocket network is comparable to that of traditional cryptocurrency blockchain applications like Ethereum and Bitcoin. The Pocket network block time is X and Y seconds and is dynamically adjusted based off of a traditional difficulty algorithm. However, unlike traditional blockchain systems, the Pocket network block submission is formulated on the number of relays and not the number of transactions. Section 4.0 Economic Model describes the intent of this design.

```
Block interface {
  Hash blockHeader;
  // checked by Federated Nodes

  List transaction;
  // a list of the current
transactions executed during this
block

  function calculateDifficulty(List
prevBlocks);
  // the difficulty algorithm of
Pocket averages out the previous X
blocks to anticipate # of
transactions for a block time between
X and Y seconds
}
```

### *3.4.1 POKT*

Within the Pocket blockchain abstraction layer, there are five generalized currency-based transactions.

1. Staking
2. Unstaking
3. Burning
4. Minting
5. Transfering

Staking

POKT is the token needed to participate within the Pocket network. Any actor on the network that is participating in the relay mechanism of Pocket must stake a certain amount of POKT to maintain both data integrity and network security. This means the only unstaked nodes on the Pocket network are passive nodes maintaining the database only.

Though section 4.0 Economic Model describes the economic sustainability behind network staking, it is important within the context of this introduction to understand the viability of the concept. Staking serves two purposes, it permits advance “payment” for the developer API service and disincentivizes participants within the network from acting maliciously.

Burning

In general, any bad acting within the Pocket network will result in a burn of their stake. More specifically, the stake burn comes in many different degrees based on the misconduct committed. The karma scoring consensus among the participants is the quantification of bad acting. Any other bad acting on the network results in the radical’s POKT being burned by the network protocol.

### *3.4.2 End User Economics*

For the Pocket end user, the amount of POKT staked is directly proportional to the allotted infrastructure utilization.

For the sake of brevity and consistency, Pocket uses the following terms to describe units of time.

Units Of Time

- Epoch: A summation of blocks
- Era: A summation of Epochs
- Eon: A summation of Eras

To accommodate for the dynamic nature of applications and to ensure fairness to developers, this paper introduces two new economic concepts: rollover and overflow.

Rollover: a design concept that conserves unused developer infrastructure utilization by carrying over leftover network usage into the next epoch. To prevent an extensive buildup of rollover network utilization, rollover is reset at the expiration of every era.

*Bob stakes X POKT which allotts him Y network utilization per epoch. Bob only uses 50% * Y utilization an epoch. Thus the next epoch he is granted 150% utilization.*

Overflow: a design concept that averages the application utilization over an Era. Developers are bound to a maximum overflow per epoch to prevent uncontrollable throughput.

*Bob's DApp spikes in epoch 1 and results in 200% usage (100% overflow). However, Bob is not penalized for the overflow as long as the average utilization for his DApp is under 100% usage at the end of the Era.*

### *3.4.3 An Introduction To Mint Allocation*

Though mint allocation and Pocket economic specifics are described in more detail in section 4.0, it is essential to understand the economic incentive of the network and why the overall design is sensible.

- Validator Nodes within sessions receive the majority of mint allocation (per relay completed).
- Service Nodes do not receive any mint for servicing.
- Edge Nodes receive a small amount of mint for completing the dispatch sequence, proportional to the number of relays completed in the corresponding session.
- Federated Nodes receive mint for block proposing.

In essence, the only actors who receive mint within the Pocket Network are those who are Federated. The economic incentivization of the Unfederated Nodes within the network are to become Federated to receive mint. As detailed in section 3.3.1, the only way to become a Federated Node is to satisfy the threshold requirements, one of which is a karma threshold. The only way to boost karma is to act as a Service Node within a Servicing Session. Hence, economic incentivization is circular and the network continues to maintain order.

## 3.5 Known Attack Vectors

Spam-for-profit attack

Definition:
Attacker "self deals" POKT token by servicing themself.

Solution:
Removing Service Node POKT payments and introduce Karma concept.

One-kill-at-a-time attack:

Definition:
Corrupt group of Federated Nodes collude to falsely report a selected individual node as a bad actor anytime they are in a quorum with that victim but play honestly in all other interactions. Eventually, the targeted node gets burned and banned. The corrupt nodes then move on to the next victim.

Solution:
(Edge Case) No solution yet

DDOS to prevent new block:

Definition:
Attacker will know the governance order for the block minting process.

Solution:
Use a consensus algorithm that does a round robin leader selection with a timeout such as Tendermint

Bot Net App (Malicious Developer):

Definition:
App takes down the service nodes before they can report anything to the validators. In turn, they DDOS the network and their stake is never burned.

Solution:
Possible middleware security mechanism to prevent overflow.

Dispatcher DDOS:

Definition:
Dispatchers are taken down by spam requests (Depends on dispatcher implementation).

Stealing Developer Stake: Developer Policies and revocable private keys embedded in the application code.

## 4. ECONOMIC MODEL

Pocket's economic model involves a continuous, inflationary token issuance that reduces the rate of mint over time and utilizes staking and burning for long term economic sustainability. The core concepts in the economic model are:

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

### 4.1 Unit of Work

Each unit of work is a valid relay agreed upon by Validator Nodes. The protocol rewards the participants with POKT tokens or Karma.

It is important to note that unlike Proof of Work or Proof of Stake the unit of work in Pocket Network is fundamentally separated from how the nodes reach consensus on the

blockchain. Validator nodes reach consensus on each unit of work and then batch the total as one transaction into the ledger.

Blocks are based off how many relays are completed in a given amount of time as explained in section 3.4.

### 4.2 Minting

Like legacy blockchain systems, POKT token is minted based off of a unit of work. Within the Pocket Network, relay batches are submitted per Servicing Session and proportional mint is awarded to the corresponding parties involved.

For each relay completed, the total mint reward is split among the following participants.

1. Validator Nodes
2. DAO
3. Federated Node
4. Edge Node
5. Application Developers

The validator nodes will receive the vast majority of the allocation, a significant portion will go to the DAO for governance, less will go to the Federated Nodes for block signing, less will go toward the Edge Nodes for dispatching, and a nominal amount will be awarded to the developers who do not use their network throughput.

By having participants mint tokens instead receive direct fees the protocol provides a significant user experience improvement in how developers interact with the services in building peer-to-peer applications. No need for refilling a wallet like a traditional software as a service business. Should traction and API requests increase for an application, the developer just needs to stake more POKT.

### 4.3 Staking

Staking is both a security mechanism and an economic sustainability device. By enforcing network staking, all actors within the network are economically incentivized to act within the confines of the rules of the protocol.

Staking also limits the circulating supply of POKT tokens, an important economic mechanism for the sustainability of the protocol. Similar to Proof of Stake implementations, participants cannot stake tokens for work, and unstake immediately.

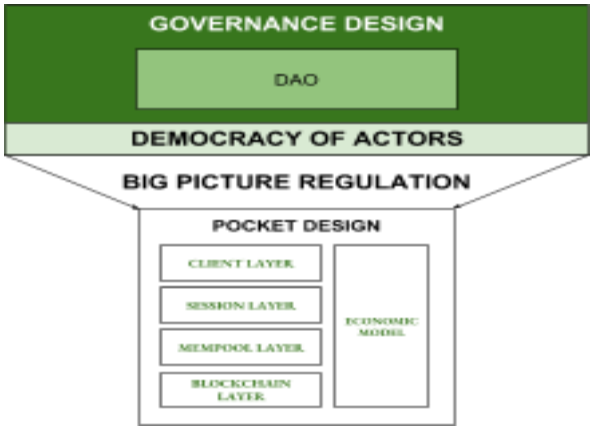
### 4.4 Burning

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. At network maturity, the goal is for the circulating supply will be deflationary or with as little deflation as possible through the following burning mechanisms:

1. POKT tokens being burned through DAO proposals
2. Developers overusing their stake and throttle
3. Nodes not serving honest requests

## 5. GOVERNANCE

When considering blockchain longevity and scalability of the Pocket Network, a mechanism for human governance is critical for creating a fault tolerant system. To better clarify the magnitude of governance, a representative diagram is shown below.




Pocket Improvement Proposals

In legacy blockchain technology, community driven renovation and enhancement ensures a maximal level of quality. An organized system of peer reviewed development, referred to as proposals, drives innovation and strengthens security [23].

### Off-Chain Stage

The Pocket Network implements a similar contribution system with Pocket Improvement Proposals. A centralized repository of formalized publications specifying Pocket propositions.

Types of PIPs

There are currently three different types of PIPs.

1. Protocol PIPs
2. Funding PIPs
3. Blacklist PIPs

Protocol PIP: can be thought of as a prototypical proposal. Any document introducing new standards to Pocket Network.

Funding PIP: is a formal document requesting funds from the Pocket DAO treasury.

Blacklist PIP: is a security mechanism allowing human governance over bad acting.

After completing the peer reviewing process, proposals will advance to the on-chain stage of vetting.

### On-Chain Stage

For Protocol and Funding PIPs, a democratic election is held semi-monthly. For a proposal to be approved, there are specific threshold requirements needed to be met.

| ELECTION PROCESS | FEDERATED NODES | UNFEDERATED NODES | APP DEVELOPERS |
|---|---|---|---|
| Estimated Population | 5% | 25% | 70% |
| Approval Percentage | 67% | 67% | 67% |
| Threshold Population | 10% | 5% | 5% |

Due to the expected frequency of the Blacklist PIP, a different process is followed. A small, pseudo-random sample of network actors are asked to vote on specific Blacklist PIPs bi-monthly. In addition, if the chosen actor

does not vote on the blacklist ruling, a nominal amount of their stake is burned.

DAO Fund Economics

Naturally, the reserve currency for the Pocket Network DAO Treasury is POKT, the cryptographic token of the network. For every POKT token minted, 10% of the value is transferred to the DAO Treasury.

## 6. THE FUTURE

Though use cases of the Pocket Network are limited only by the strength of imagination, only the near future concepts will be discussed for brevity.

Decentralized Exchange

With a newfound ability to create multi-blockchain applications with relative ease, a decentralized exchange would be a perfect example of the power of Pocket Network. With developers able to now easily access nodes from any chain needed, building this complex device becomes much more tangible.

Universal Wallet

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.

## 7. 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 Incorporated: Pabel Nunez 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 predictive big data advising and 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.

## 8. 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.

[4] B. Bernstein, “Twitter”. 10-Jul.-2018.

[5] “Ethereum Javascript API”. 30-Jul.-2015.

[6] W. Bins, “Bitcoin Developer Documentation”. 2009.

[7] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” Proceedings of the Third Symposium on Operating Systems Design and Implementation, 1999.

[8] “Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus,” Hyperledger Architecture, Volume 1, 2017.

[9] “Tendermint: Byzantine Fault Tolerance in the Age of Blockchains,” A Thesis Presented To The University of Guelph, 2016.

[10] C. Dwork, N. Lynch, and S. Larry, "Consensus in the Presence of Partial Synchrony ," Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing , 1984.

[11] E. Duffield and D. Diaz, "Dash: A Privacy-Centric Crypto-Currency," 2015.

[12] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Oct. 2008.

[13] D. Petkanics, "Inflation and Participation in Stake Based Token Protocols," Medium, Dec. 2017.

[14] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a Decentralized Oracle and Prediction Market Platform," Jul. 2018.

[15] W. Warren and A. Bandeali, "0X: An open protocol for decentralized exchange on the Ethereum blockchain," 0xproject.com, Feb. 2017.

[16] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," ethereum.org, Feb. 2013.

[17] D. Petkanics and E. Tang, "Livepeer: Protocol and Economic Incentives For a Decentralized Live Video Streaming Network," livepeer.org, 2016.

[18] L. Cuende and J. Izquierdo, "ARAGON NETWORK: A DECENTRALIZED INFRASTRUCTURE FOR VALUE EXCHANGE," aragon.one/network, Apr. 2017.

[19] R. Stewart and B. Gilbert, "CONCEPTUAL MODELING: DEFINITION, PURPOSE AND BENEFITS ". 2015.

[20] "Amazon AWS Docs: Access Policies".

[21] C. Lacina, "Coin Telegraph: The Inevitable Failure of Proof-of-Stake Blockchains and Why a New Algorithm is Needed (Op-Ed)". 24-May-2015.

[22] "Bitcoin Wiki: Protocol rules". 25-Aug.-2017.

[23] "Bitcoin Improvement Proposals," Bitcoin Wiki. 13-Feb.-2016

## 9. GLOSSARY

Application Client

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.

Bootstrapping Sequence

The initialization process to first connect to the Pocket Network.

DAO

A decentralized autonomous organization that is governed by the Federated Nodes.

Edge Node

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

Eon

A unit of time consisting of a summation of Eras.

Epoch

A unit of time consisting of a summation of blocks.

Era

A unit of time consisting of a summation of epochs.

Federated Node

A trusted node that maintains a series of threshold requirements.

Karma

An autonomous node scoring system based off of the consensus within a Servicing Session.

Mint

Newly created POKT proportional to the number of relays completed.

Overflow

Allowed developer throughput overages per epoch.

Pocket 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 cryptographic token needed to participate within the Pocket Network

Servicing Session

An ongoing relationship between an application developer and 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.

Rollover

Unused developer throughput added to the next epoch's allocation.

Servicing Node

The act of a node receiving API requests and responding with the data.

Validation Node

The act of a Federated Node verifying the integrity of the actors and data within a Servicing Session.