OT-RFC-14 DKG v6 TRAC Tokenomics

Authors: OriginTrail Core Developers

Version: 1
Date: 2022-10-14

From data to assets - gateway to the knowledge economy

The upcoming new version 6 of OriginTrail Decentralized Knowledge Graph (DKG) is introducing a paradigm shift to the OriginTrail ecosystem and marks the beginning of a new era. Until now, OriginTrail network delivered key enhancements to the world of decentralized data (such as integrity, interoperability and interconnectedness). With the new version, we are making the leap from exchanging data to creating assets from the data. This is possible with features of discoverability, verifiability and ownership of assets on the DKG representing anything in the physical and digital world.

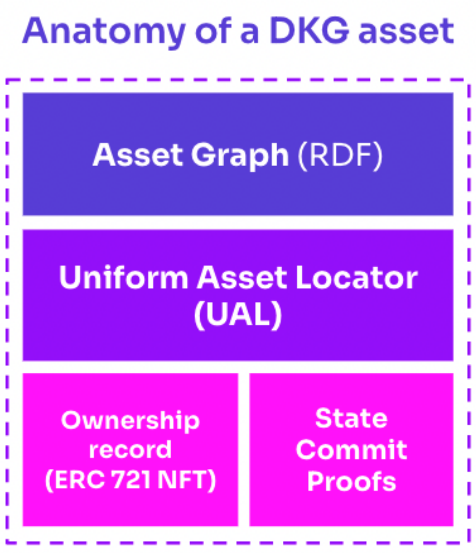
The new paradigm enables anyone to create DKG assets and own their connected "real estate" in the global Web3 graph, enabling a plethora of applications previously impossible due to lack of true data ownership primitives on the internet. The new functionality takes nothing away from the original value propositions but rather upgrades them with ownership, the OriginTrail technology and TRAC tokenomics need to be upgraded as well.

Before jumping into the tokenomics, let's take a closer look at what the core building blocks - DKG assets - are from the perspective of DKG and how they influence the TRAC tokenomics. Assets consist of 4 components implemented across two network layers:

- **The DKG layer** hosts the **asset graph state (1)** in semantic form (RDF), identified by a new Web3 primitive - the **Uniform Asset Locator (UAL) (2).** Similar to how Uniform Resource Locators (URLs) made it possible to reference an object (e.g. a website or document) in Web2, UALs make it possible to reference anything physical or digital, in Web3. UALs extend existing web URI standards, revolutionizing URLs as their counterparts in Web2. A UAL also inherits capabilities described in the W3C recommendation on Decentralized Identifiers (DIDs). The UAL format (to be formally specified in further documents) therefore is in the following standardized form:

  dkg: [// DID] / UAI ["?" query] ["#" fragment]

- **The Blockchain layer** is responsible for maintaining asset ownership record and asset graph state verifiability proofs. Information about ownership is kept in **the form of an ERC 721 compatible Non Fungible Token (3)**. This allows us to tie DKG and blockchain layers even more closely together and enable ownership to be transferable in the form of NFT token transfer. **State commit proofs (4)** are used to verify that the asset graphs DKG nodes persist are available, haven't been deleted or in any form tampered with, as well as enabling complex verifiable presentation implementations (according to W3C Verifiable Credentials)



Moving from data bundles to unique assets

The introduction of DKG assets is expected to induce an update of existing applications built on OriginTrail significantly, as previously bundled publishings (being just “data”) now get “unbundled” and packaged into unique assets, each published with their own asset graphs and UALs for referencing. In versions prior to v6, publishers would publish bundled “data dumps” in consolidated form actually containing information on sometimes hundreds or even thousands of actual physical objects - each of which in v6 is expected to become a uniquely published DKG asset with its own UAL as a verifiable, ownable identifier. Bundling can still be performed if needed by the asset publisher however - builders and developers now have more freedom to create different solutions on the DKG.

Assets can be continuously updated by their owners with new (immutable) assertions, allowing assets to increase their asset graph and their value. Assets also act as building blocks - they can be connected to other assets,be indexed in various ways (for search) and extended in many ways to improve the discoverability of the asset.

DKG as a Multi-chain Decentralized Knowledge Graph

OriginTrail technology is a system of 3 logical layers. The top layer is the application layer where web and mobile applications reside with their interfaces, backends and connectors to the DKG. Sitting below the applications we find OriginTrail Decentralized Knowledge Graph as layer 2. It consists of OriginTrail Decentralized Network (ODN), a permissionless network of nodes that are maintaining a distributed knowledge graph of assets. Finally the consensus layer (layer 1) is implemented via blockchain enabling trusted interactions needed for the OriginTrail protocol. Since inception, the DKG was designed to enable multiple blockchains to support a single DKG. This design was for the first time implemented in 2021 when the DKG became multichain and leveraged the many efficiency gains this unlocked which led to over 10-fold jump in its adoption. After the initial Ethereum deployment, we have seen both Gnosis and Polygon blockchains get integrated with the DKG.

After successfully obtaining the parachain slot on the Polkadot network earlier this year, OriginTrail will now also receive a tailored blockchain layer network in OriginTrail Parachain, allowing for yet another jump in its overall capabilities. This is particularly important for the OT-RFC-14 as the DKG v6 will have its inaugural launch on the OriginTrail Parachain. The OriginTrail Parachain and OriginTrail v6 DKG integration is effectively starting with the Teleporting process to bring TRAC tokens from Ethereum to OriginTrail Parachain (explained in OT-RFC-12 and OT-RFC-13). The TRAC holders participating in the Teleport will also be the first to engage with the OriginTrail v6 and its new tokenomics presented below.

Guidelines to TRAC tokenomics

The OT-RFC-14 document is serving as a foundational piece upon which the first implementation of TRAC tokenomics in the v6 of OriginTrail DKG will be done. As with all TRAC tokenomics so far, the latest version is equally molded to support the adoption of the OriginTrail DKG, however this is the first time it takes full advantage of the upcoming OriginTrail DKG v6. When assessing the various aspects of TRAC tokenomics it's important to keep a few high-level goals in mind, as they guide us to growing the ecosystem:

- **Ease of use -** asset publishers are the core users of the OriginTrail DKG and in order to allow them to engage with OriginTrail technology, it should be reasonably easy to use.
- **Cost efficiency -** allow the market to achieve attractive prices by not wasting resources.
- **DKG node rewards** - allow DKG nodes to get compensated for providing useful services to the network. By enabling an efficient market for the "supply side" of the DKG network, infrastructure runners can get fairly compensated.
- **Quality of service -** ensure that applications querying the DKG (asset consumers) benefit from a high quality service.
- **Scalability of network activity** - implement a system that has capacity to support the rising levels of adoption.
- **Decentralization -** as a core consideration in any implementation and tokenomics decision, avoiding system control by any single party

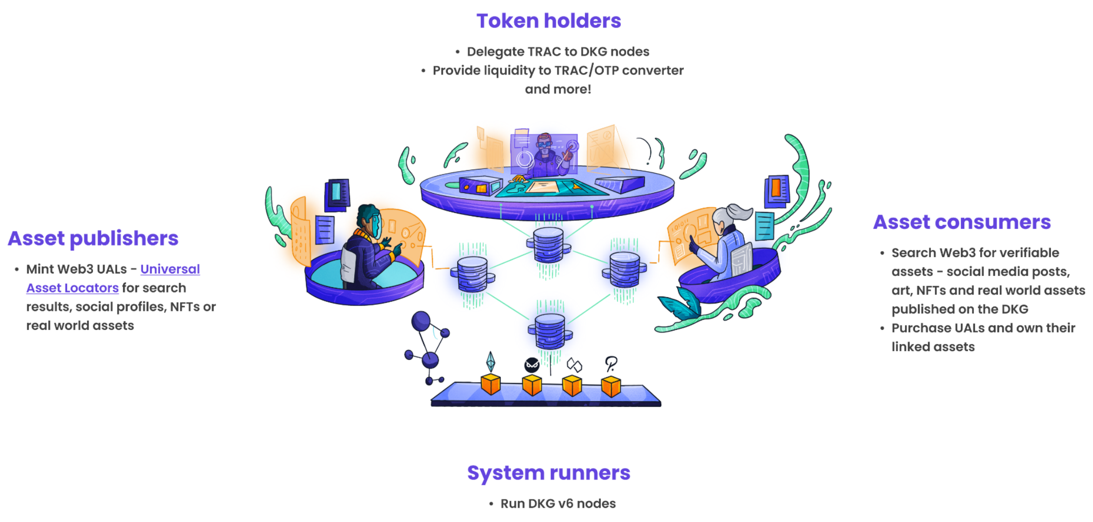
TRAC tokenomics stakeholders

The TRAC tokenomics stakeholders can be described through 4 system roles:

1. **DKG asset publishers** are agents which create and manage knowledge graph entities (assets) by publishing asset graphs to the DKG according to the OriginTrail Protocol
2. **DKG asset consumers** are agents who discover, query and consume the DKG asset graphs created and owned by DKG asset publishers

3. **DKG node runners,** who run and maintain the DKG infrastructure nodes, providing the DKG services
4. **TRAC Token holders**, who participate in the ecosystem by providing economic support for system activities

The roles indicated above however do not need to be distinct - one ecosystem actor can hold different roles at the same time. For example, DKG node runners are by definition also TRAC token holders as TRAC tokens are necessary for running DKG nodes.


Token holders
• Delegate TRAC to DKG nodes
• Provide liquidity to TRAC/OTP converter
and more!
Asset publishers
• Mint Web3 UALs - Universal Asset Locators for search results, social profiles, NFTs or real world assets
Asset consumers
• Search Web3 for verifiable assets - social media posts, art, NFTs and real world assets published on the DKG
• Purchase UALs and own their linked assets
System runners
• Run DKG v6 nodes

DKG node services

The OriginTrail Decentralized Knowledge Graph (DKG) natively operates on the OriginTrail Decentralized Network (ODN) which represents a permissionless decentralized system of nodes communicating according to OriginTrail protocol. Anyone with sufficient technical knowledge is able to run an OriginTrail DKG node and contribute to the system by doing so. By design, the DKG nodes receive compensation for the network services provided in the form of TRAC tokens.

The DKG nodes are able to perform various different activities, specified by OriginTrail core protocol operations or through extensions. On the network level we can categorize these activities into two groups:

- **Publishing protocols**, which introduce additional asset graphs into the DKG (equivalent to semantic web publishing, or a database insert operations)
- **Querying protocols**, which enable asset discovery through network queries (equivalent to database query operations)

In OriginTrail v6 we distinguish two generic types of DKG network nodes:

- ***full* or *service* nodes** which provide network services implementing full DKG protocol choreographies. Full nodes persist the public DKG state, make it available and execute semantic queries, in return for getting compensated in TRAC tokens.
- ***light* nodes** which do not provide network services, rather act as network gateways. They are not responsible for persisting the public DKG state as they do not have such capability (don't respond to publish requests, therefore don't qualify for rewards). Light nodes are analogous to the notion of light clients in blockchain networks, eliminating the need for gateways such as RPC services with having direct access to the DKG .

Full DKG nodes implement services to facilitate the publishing and querying choreographies. Nodes can generally implement a variety of such protocols, building on the core components: **assertions, shards** and **service agreements.**

Assertions present graph data entries into the system. They constitute **asset (data) graphs** coupled with corresponding on-chain state commit proofs**.** In a general sense, assertions are "data building blocks" which represent asset graph state, search index entries or any such data structure implemented on the OriginTrail DKG. In OriginTrail v6 graph assertions are natively modeled in RDF (Resource Description Framework), therefore all tooling of the Semantic Web stack is interoperable with OriginTrail on the core level. Assertions can be versioned, and are expected to be further aligned with standards and recommendations, such as the latest initiative led by Phil Archer of GS1.

Assertion shards or **replicas** represent DKG entries positioned in the *DKG index space (the index space is formed by all full nodes in the network)*, centered around a *replica key*, hosted by a number of DKG nodes. Specific groups of nodes in a replica are subsets of the DKG index space node "neighborhoods" (similar to distributed database shard concept), formed according to **service agreements** facilitated by the protocol. At any moment, one full node hosts a number of logical shards, and due to node id difference (different positions in the index space), different nodes are intended to host different pieces of the DKG. This mechanism achieves a randomized, yet highly organized, distribution of shards and data over the decentralized network.

Service agreements are defined on assertion level and incentivise availability for a particular set of assertion shards over a period of time. When service agreements no longer require assertion replicas or expire, the assertion replicas availability is no longer incentivized. For ease of implementation, service agreements are defined in terms of space quants, represented in bytes, and time quants called *epochs*.

Full DKG nodes receive assertions from asset publishers, form shards in qualifying neighborhoods hosting assertion replicas, which make assertion graphs available for querying during the service period. For these services, (full) nodes receive compensation in TRAC tokens. "Light" nodes enable interacting with the DKG (publishing, querying) without the capability of providing above mentioned services.

Publishing protocol choreography

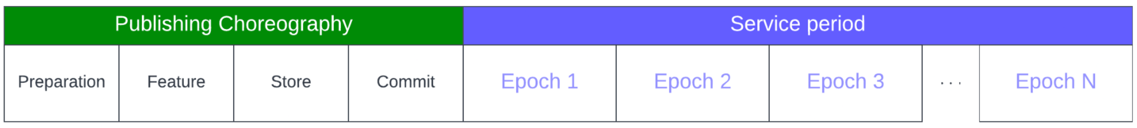
The publishing protocol choreography is a coordinated activity between the publishers and DKG nodes whose goal is to form service agreements and establish assertion replicas on the DKG. Due to the decentralized nature of the system, the DKG nodes receive compensation for their services according to their defined **service asks,** expressed in TRAC tokens required for kb-epoch units. Using the publishing choreography of the DKG one can publish various different data structures on the DKG (assets, index entries for keyword search, vector search in the future etc).

A publishing choreography initiates a service agreement and is successful if a service bid provided by the publisher satisfies the required service asks posted by DKG nodes for the assertion replication in the given neighborhood. Service asks are posted by DKG nodes on-chain and updated on a regular basis, forming the service price.

An asset publishing choreography consists of 4 phases:

1. Preparation phase: the asset publisher prepares an assertion in the format expected by the protocol
2. Feature phase: initiates the appropriate functionality of the desired feature (e.g. updating an asset state) on the blockchain
3. Store phase: invokes the store protocol to create an assertion replica on the DKG
4. Commit phase: used to finalize the choreography and initiate the service period

DKG Asset lifecycle (in OriginTrail v6)



Service market mechanics

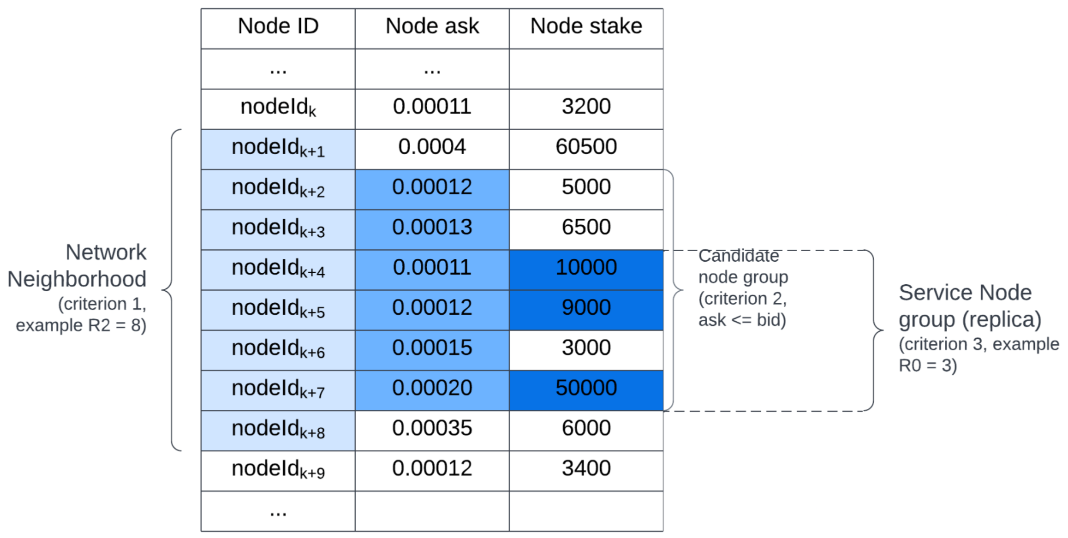
The DKG service market is formed between asset publishers & consumers on the demand side, and DKG node runners on the supply side. As the DKG is a decentralized environment no entity controls the market and each participant in the DKG gets to act based on their economic preferences.

For ease of use in OriginTrail v6, the asset publisher client derives a suggested bid for a service based on the price (according to on-chain asks provided by DKG nodes). Once a publisher creates a bid for their asset on chain (feature phase) and successfully creates required assertion replicas on the DKG (store phase), the nodes from the assertion replica submit commit proofs to the blockchain (commit phase). Once the commit phase is completed, the publisher tokens are locked in the system (in the bid amount, to be collected by DKG nodes), the publishing choreography is completed and the service period starts.

Assertion replicas are hosted by groups of DKG nodes forming **shards**, based on the following 3 criteria in order of importance:

1. **Key-space proximity criteria**: assertions need to be replicated to nodes and maintained in a specific “neighborhood” of the DKG index space, specifically around the key where an assertion is to be kept. Nodes form neighborhoods based on the XOR distance of the assertion key to the node IDs, similar to how distance is determined in Kademlia DHT. This criteria needs to satisfy efficient data discovery in the decentralized network environment by deterministically locating assertion replicas in querying choreographies. Additionally, due to assertion keys being consistently hashed, assertion replicas are evenly and randomly distributed across the network key-space. A neighborhood is formed by the R2 number of nodes (neighborhood size) which is a protocol parameter.
2. **Node compensation asks:** once the appropriate neighborhood is determined (satisfying the first criterion), the second requirement is to ensure that there are enough nodes willing to provide the service in that neighborhood, indicated by their compensation ask. A bid satisfying a minimum number of R1 nodes (R1 being the minimum set of “willing” nodes within an R2 neighborhood, R1 <= R2) is considered a valid bid in terms of the second criterion, where R1 is a protocol parameter
3. **Security signal criterion:** after both previous criteria have been satisfied, finally R0 nodes get chosen (R0 being the minimum replication factor) out of the R1 set, **based on the amount of provided TRAC token stake.** A higher stake of TRAC tokens in the system indicates higher quality of service guarantees by nodes, as the stake represents a collateral guarantee for nodes providing expected services and acting according to service agreements.

An illustration of the publishing choreography criteria with regards to R0, R1 and R2 is provided below as an example for R2 = 8 and R0 = 3.


Network Neighborhood
(criterion 1, example R2 = 8)
Candidate node group
(criterion 2, ask <= bid)
Service Node group (replica)
(criterion 3, example R0 = 3)

Node ID	Node ask	Node stake
...	...	
nodeId$_{k}$	0.00011	3200
nodeId$_{k+1}$	0.0004	60500
nodeId$_{k+2}$	0.00012	5000
nodeId$_{k+3}$	0.00013	6500
nodeId$_{k+4}$	0.00011	10000
nodeId$_{k+5}$	0.00012	9000
nodeId$_{k+6}$	0.00015	3000
nodeId$_{k+7}$	0.00020	50000
nodeId$_{k+8}$	0.00035	6000
nodeId$_{k+9}$	0.00012	3400
...		

Service period

During the service period, nodes involved in the service agreement need to provide service proofs during each epoch of the service to the blockchain. Providing service proofs consists of answering randomly generated cryptographic challenges for hosted assertions and submitting appropriate answers to the blockchain for verification. Successful proof generation and submission to DKG smart contracts in the designated period of time guarantees that a node indeed hosts the assertions required by the service agreement. Once a successful proof submission is performed on a blockchain integrated with the DKG, a portion of the locked tokens (during the publishing choreography) is released and designated for the hosting node.

In case a service node fails to provide a proof in an epoch, replacement nodes from the candidate node group can replace it in the service group by submitting service proofs (in case they host the required assertions) during the replacement period. The same criteria (as indicated above) apply for the replacement process (self-healing). Therefore by design, the entire network neighborhood is incentivised to host assertions opportunistically in order to be able to perform “self-healing” of replicas.

In the case of a node failing to provide services (and submit proofs), it is subject to **stake slashing**. Slashing is no way intended to punish nodes for reasonable latency in submitting proofs (such as short downtimes for updates, or while under load) but rather acts as a deterrent to longer service outages. Slashing is implemented as a lockup system, where a designated percentage S (S being a protocol parameter, suggested value being 5%) is being locked from the available amount of stake for a period T (T also as protocol parameter, suggested value being 2 years). In this way the node’s stake isn’t lost or redestributed among other stakeholders (which could drive malicious targeting of nodes), but rather incapacitated to be utilised in network activities for a significant time period.

Additionally, node replacements are possible in case when new (full) nodes join the DKG network. Since new joining (full) nodes will inevitably increase the coverage of the DKG index key space and become part of neighborhoods adjacent to their Node IDs, in order to provide high quality discoverability services (via the querying choreography) new nodes are incentivised to provision assertion replicas expected in those neighborhoods. Therefore all DKG v6 nodes joining the network for the first time will perform an opportunistic synchronization sequence loading the assertions in their immediate neighborhoods, enlarging replicas and making assertions available, with the aim to compete for the available compensation via replacement. The DKG tokenomics are designed therefore to **incentivise the maintaining of healthy assertion replicas** even with network growth as neighborhoods are changing. Replacements due to better positioning in the DKG index key space (and not due to lack of proof submission) do not initiate stake slashing.

Querying protocol choreography

We consider querying protocols as any protocols that enable discovery and information retrieval from the DKG. They present coordinated activities between asset consumers and the DKG node runners, aimed at finding and querying assets on the DKG, by leveraging the core protocol querying choreography and the DKG index.

The querying choreography on the DKG directs a generic query (such as a request for an asset by its identifier - the UAL, a SPARQL query or similar) to one or more assertion shards, effectively querying the appropriate “neighborhood” as designated by the publishing choreography. The query routing is equivalent to the Kademlia protocol among full nodes of the DKG. Light clients are not part of the routing protocol, however can invoke queries and publish assets through the network. (It is the responsibility of clients to implement publishing and querying specifics on the application layer to utilize the DKG sharded nature when designing information retrieval systems).

An asset querying choreography consists of 4 phases:

1. Preparation phase: the asset consumer client prepares a query in some way (e.g. taking input from a search box)
2. Feature phase: initiates the query prepared in the previous phase on the DKG, on the appropriate DKG shard, and performs necessary URI resolution if needed
3. Verification phase: verifies the obtained results from the feature phase using on-chain verifiability proofs (verifying immutability, state consistency, issuer or other)
4. Presentation phase: presenting the query result to the client in requested form

Note that **no global SPARQL endpoint in the system exists**, due to its decentralized nature. Each of the nodes can however freely decide to build and expose their own SPARQL endpoints, having all implications of such a decision in mind (e.g. availability). The DKG enables queries on its shards via asset graph traversal facilitated by UALs. DKG nodes also implement local graphs (for storing private asset assertions, in their local triple stores) for which they can perform local SPARQL queries of any sort, on both the local asset graph and the loaded assets from the public DKG. More information on querying capabilities will be provided in the official documentation.

DKG Node service market

Nodes providing services to the DKG network are subject to the market forces (supply and demand), driven by the choreographies explained above. With the updated implementation in version 6, it is expected that DKG node runners obtain an easier and clearer way of determining the optimal market value of compensation for their services, enabling them to achieve profitability easier. Nodes are expected to maintain their asks on-chain, enabling the asset publisher applications to obtain the current market price for DKG services.

In unison, the market becomes more efficient due to an improved ability of DKG node runners to differentiate their services based on service quality (through availability, throughput, capacity) and security (proxied through the amount of TRAC collateral posted).

We envision a set of tools to enable easy management of service parameters for DKG node runners to achieve the best possible market return. Such tools would indicate the fluctuations in supply and demand, for example indicating network load increase (due to higher demand) which would have an impact on the service price in the positive direction, until the service supply is increased (via additional nodes joining the network).

Tokenomics role in system security

In order to reach the necessary system security level the OriginTrail network requires all full nodes to post an amount of TRAC tokens as security deposit in the system. With the new implementation of tokenomics as proposed above, the stake provided by nodes has a function also in the publishing choreography, as described by the third criterion. As nodes compete in the market based on prices and deployed stake in the network, it is expected from nodes to attract a higher amount of stake (either by directly depositing more TRAC by the node runner, or accepting delegated TRAC from the community) to position themselves better in the market. Delegated node staking becomes one of the additional features in OriginTrail v6 and will allow TRAC token holders to participate even further in the TRAC economy by delegating their tokens to nodes they pick, being eligible for sharing the compensation for services by those nodes. This new component of the tokenomics not only introduces new ways for ecosystem stakeholders to participate, but also motivates them to act as “quality signal providers”, as token holders need to carefully assess who to delegate their TRAC to to minimize the probability of slashing. DKG node runners are expected to become further specialized in capabilities needed to provide a high quality service to the network.

As stake slashing needs to be a meaningful deterrent to potential service outages, given the proposed slashing implementation above, the minimum required amount of posted stake for a full node will be increased to 50 000 TRAC in order to be eligible for operation. Nodes can achieve this minimum both through direct staking or by collecting delegated TRAC tokens from the community. (Note: delegated TRAC tokens cannot be controlled by node runners, only by delegators). Being a stake delegator is therefore similar to the notion of nominators in the Polkadot network.

Detailed delegation and staking mechanics are to be presented in the coming RFCs.

OriginTrail v6 scalability considerations

The implementation of OriginTrail v6 introduces, as already mentioned, the notion of full and light nodes. It also clearly separates the functionalities to be implemented by clients and network nodes (which were previously one monolithic structure).

The responsibility of the dkg-client is therefore to implement the preparation and feature phases of both publishing and querying choreographies. DKG nodes implement all the network communication related phases and incentivisation functionalities, however are "relieved" of e.g. the preparation phase complexities and load, which are moved towards the network "edge" (clients). With the separation of concerns between the clients and nodes, as well as the new implementations choreographies illustrated above the DKG implementation becomes far more scalable.

Furthermore, with the new indexing logic and deterministic shard resolution, the nodes need to communicate significantly less, making the load of publishing even less strenuous on nodes, making them able to run more operations at the same time than previously. In this way the ODN becomes "less chatty" and achieves an order of magnitude scalability improvement in the DKG layer 2.

Another consideration in scalability is the OriginTrail Layer 1 (blockchain), which in the system illustrated above is expected to handle a number of transactions per specific epoch. The limit of scalability in terms of blockchain can be calculated by the following equation:

*Number of assets secured on DKG in one epoch = Epoch_length * block_size / number of proof transactions in an epoch.*

The *epoch_length* is a system parameter that, based on the equation above, essentially determines the number of assets that can be supported by one blockchain in OriginTrail layer 1. We conclude that architecturally the blockchain layer doesn't present an immediate bottleneck for the growth of the DKG. Iterative deployments and optimizations on the blockchain level, such as through more efficient smart contract code, or direct implementation of components in Substrate, are also expected over the course of OriginTrail evolution.

Conclusion

This RFC presents the updates to the TRAC tokenomics in the sixth version of OriginTrail protocol with details of the proposed implementation and activities the system needs to incentivise, grouped around publishing and querying choreographies. It introduces all the system stakeholders, their roles and activities the network needs to support each of the functionalities required from a decentralized knowledge graph. We welcome the feedback and comments from the OriginTrail community as always. Trace on!

References

1. OriginTrail Whitepaper
2. Semantic Web (Wikipedia)
3. W3C Resource Description Framework
4. RDF Dataset Canonicalization and Hash Working Group Charter
5. Kademlia: A Peer-to-peer Information System Based on the XOR Metric
6. Polkadot Wiki
7. W3C Decentralized Identifiers
8. W3C Verifiable Credentials Data Model
9. OT-RFC-13 TRAC Token Deployment on Polkadot
10. OT-RFC-12 OriginTrail Parachain TRAC bridges (v2)