Yggdrasil Ecosystems, a decentralized application network

Abstract

YGGDRASIL ECOSYSTEMS -DAN is a cross-blockchain layer-2 network specifically focused on decentralized applications and their related infrastructure (storage, computing servers, security). Our aim is to decentralize and revolutionize the web and the cloud as we know it. Current decentralized applications are generally unreliable and slow or tied to single blockchain architecture. Decentralized applications need to not only overcome these issues, but they also need to be able to communicate with other projects. The large majority of current blockchain-related technologies cannot scale to the levels needed for large applications (social networks, web apps, IoT providers, etc) we use on a daily basis. YGGDRASIL ECOSYSTEMS -DAN intends to provide a solution to these issues by offering fast single cross-technologies and cross-chain solution on a decentralized and reliable ecosystem.

Contents

1. Imtroductiom

The Internet was originally built to connect computers together. It has been used as a marvelous communication and data storage model. Protocols originally designed for decentralization (TCP, HTTP, SMTP… and even blockchains) are already showing their limitations.

In a purely economically driven world; simplicity, centralization, and non-standard data models tend to win out.

What is our internet today? Silos loosely connected together, users having data in each of those silos (facebook, Google, and many others), with no simple way to port it over. (see Drake 2015)

There are a lot of attempts at breaking those silos, from connection from silo to silo (iftt, rgpd for data request, opendata programs…), to decentralized networks of silos (mastodon). We are living in a world of data silos.

In itself, that isn't bad if those silos were easily accessible and decentralized. Users don't know it, but they need to reclaim their data.
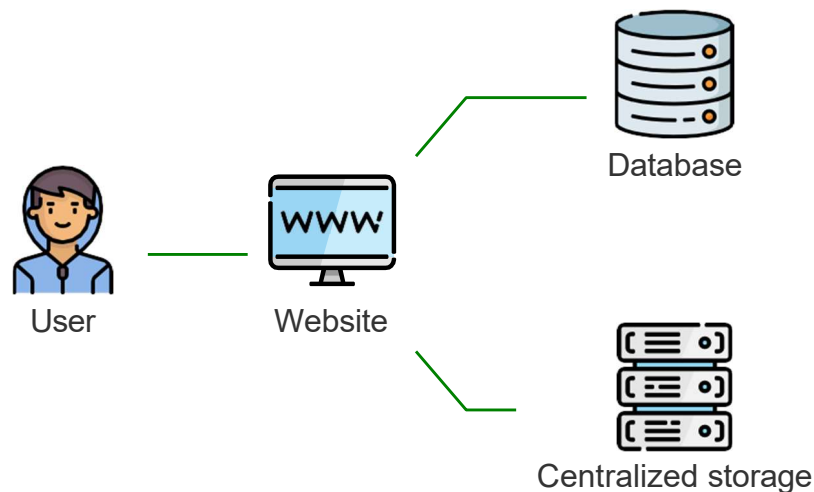
figure 1.1: Typical centralized application

2. Existimg solutioms

2.1 Blockchaim

DLT[1] allows us to have a single data silo (albeit specialized) that is spread across all users of the network. Smart-Contracts make use of that ability to allow decentralized computing (among other benefits).

Common naming of dApps[2] is using smart contracts on blockchain platforms like Ethereum. A few issues arise from its model:

- Paid transactions by the user (meaning a dApp user should buy some underlying network asset to use the Dapp)
- Reliance on inclusion in a block (dependant on the block time of the relevant blockchain, making dApps slow)
- On-chain storage is very expensive, making it unsuitable for media content or large documents.

There are a lot of existing blockchain platforms currently active (see "Coinmarketcap" n.d.).

---

[1]Distributed Ledger Technology, mostly BlockChain

[2]Decentralized applications

2.2 DHT storages like IFFS

Storage networks like IPfS or Swarm are great to find a resource based on its hash in the network. Some things are currently needed or not easily useable for our dApps:

- Incentivization of storage (to be implemented in their fileCoin for IPfS, implemented on Ethereum for Swarm)
- Unique resource naming system (provided in a slow way currently by IPNS, fix being worked on, or through Ethereum smart contracts in Swarm)
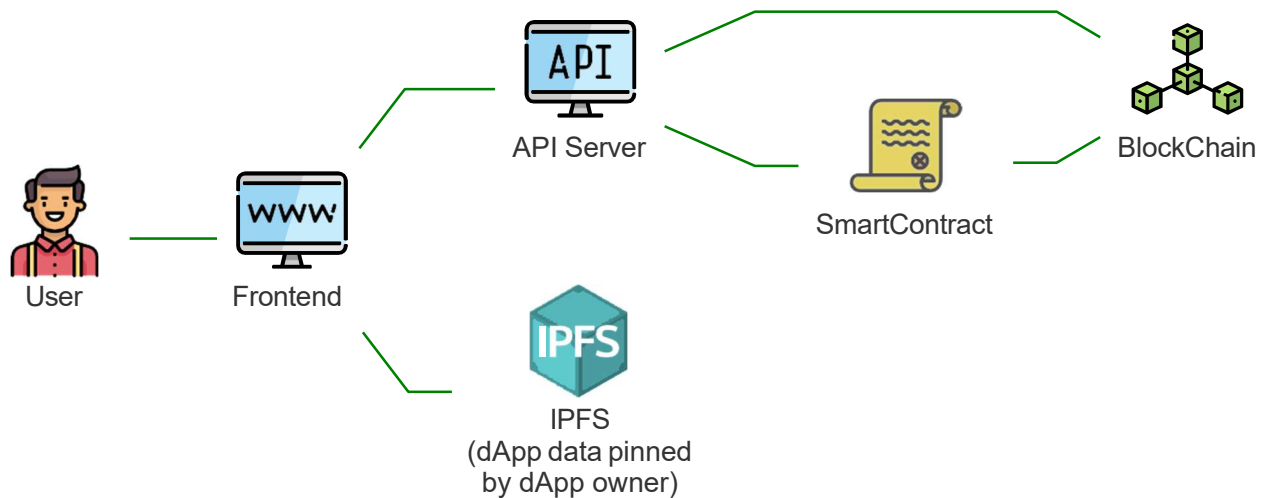

figure 2.1: Typical smart-contract based dApp using IPfS

3. YGGDRASIL ECOSYSTEMS -DANArchitecture

In current solutions involving smart contracts, the current state of applications is written (or computed from) on a blockchain, this is immutability. But what if you want instantaneous actions, batching user requests for non critical items?

You'll need a second layer of truth, a distributed state that you can influence by signing messages.

Agents will then commit the messages on the blockchain for the users, incentivized by the network token. They will spend the native chain asset and will receive the token in exchange for their service, provided by the next authorized agent. Then that agents work will be verified by the previous agent, changing it's credit score on the fly.

This allows for free and instant interaction with dapps for users, with efficient batched inserts on the blockchain for immutability.

Moreover, this more easily facilitates agents running API servers that can be used by the dapps as entry points to the network.

Below are a few focal actions of the YGGDRASIL ECOSYSTEMS -DANnetwork:

- Instant (low-latency) global status update
- Bulk content write on blockchain to allow fee-less commits
- Pinning IPfS data (data replication)
- API server decentralization
- Virtual machines code execution and verification
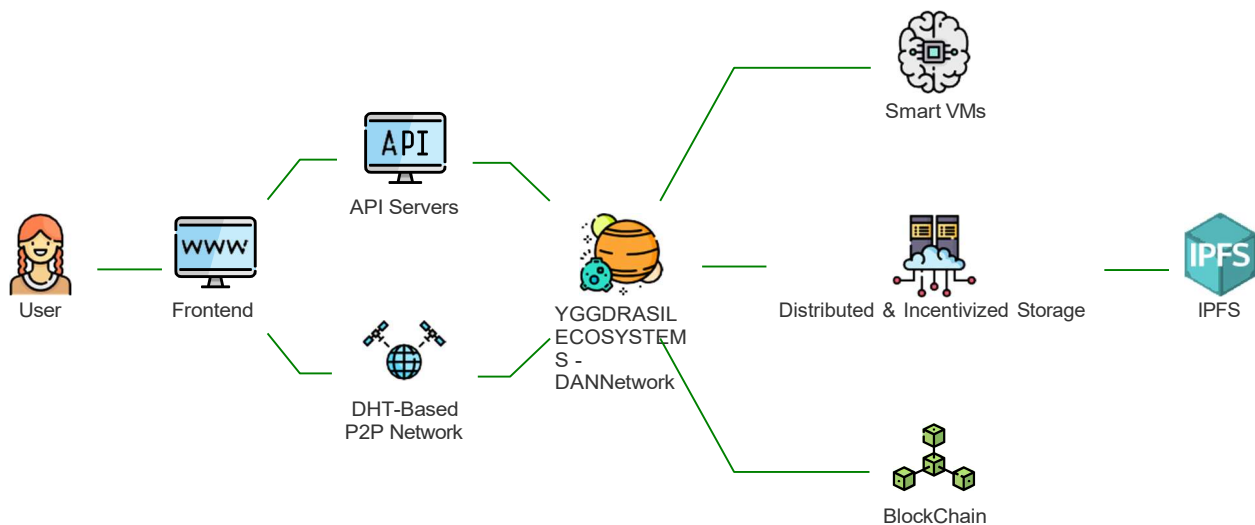
3.1 Blockchaims

The first underlying blockchain supported by YGGDRASIL ECOSYSTEMS -DAN To allow easier port of existing dapps and ecosystems, later in the developement process, other blockchains can be supported such as Ethereum, NEO and even the Bitcoin blockchain.

The address in the YGGDRASIL ECOSYSTEMS -DAN network is the underlying network of choice (, Ethereum, NEO…) address used to sign messages. A user can post in his personal aggregate (key-value hash table linked to his primary address) his addresses on other networks. That way he can redeem or receive tokens from other blockchains.

If a user has a account and an Ethereum account both linked together, a request for his aggregates (profile or settings for example), posts (images, blog posts…), or any other linked content will return the same information from any of the two addresses.

3.2 Data storage

The hashes of the data are stored on chain. The data itself is stored encrypted or not on IPfS. Data is pinned by participating nodes.

Posting is done either via API by the dapps to an API node, or by IPfS directly if available on the browser.

The current state is stored on both blockchain (nodes are made of a blockchain explorer) and current received data that is not posted yet (queue, mempool).

Once the data signature is verified, and it is broadcasted to at least 2 nodes, it is considered validated and included in the coming blocks.
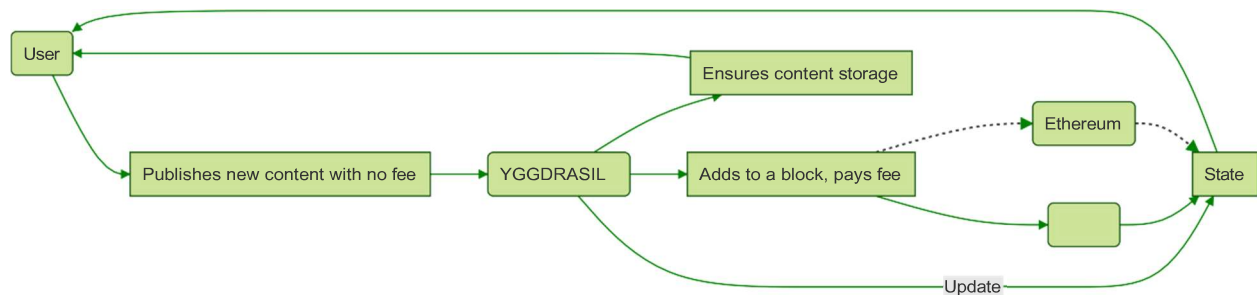
figure 3.1: Data posting procedure

3.3 Data type

Table 3.1: Data types

Data Type	Description	Details
Aggregates	Key-value store linked to an account[1]	Each key can be updated separately and its content is merged with previous. *Example*: user profile

---

[1]Underlying blockchain of choice address and linked public key.

Data Type	Description	Details
Fosts	Single data entry	Has a type field, an optional ref (reference) field that references another post or application-specific string. Can be amended with new posts with type "amend" and ref to the previous post. Data inside post content can be optionally encrypted with either owner key or a recipient key. *Example:* blog post, comment, picture gallery, video entry, new data point, new event…
VM State	State of a virtual machine	fields depending on underlying engine (dockerized language-specific contracts or WASM for example)

3.4   Data exchange

The exchange of data between the dapps and the nodes is done either by pubsub (if available) or API posting. The node will do the pubsub action on behalf of the user.

Using pubsub, all nodes will get the user posts and actions.

Pubsub is using dht and ensuring all subscribed nodes will receive all users posts to get the current state.
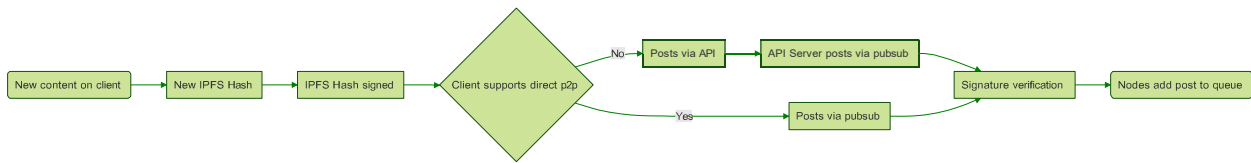
figure 3.2: Data posting procedure

ç.5   State

State encompasses both onchain committed data and uncommitted data received by pubsub. State can be recomputed by getting all TX from a user or containing messages signed by him, and adding the uncommitted messages.

- for smart contracts/VMs, it should be recomputed from last committed, signed and non revoked (no litigation) onchain state.
- for aggregates (user hash tables) from the last onchain commit for each key.
- for posts, from the original post plus all the amends. If last amend contains all the fields, original plus last amend is acceptable.

figure 3.3: State composition

3.6 Virtual machimes

Sometimes called smart contracts, the virtual machines have a state that is commited as an object in the chain. All nodes don't need to keep all the VM's states, only the last one if no litigation occurred.

3.6.1 Hash limks

Each mutation to a VM state contains a relation to a previous state (previous hash). If two data blocks relate to the same hash, the first to be commited on-chain will prevail (in case of data sync), and the first to be received (in case of off-chain processing before block inclusion).

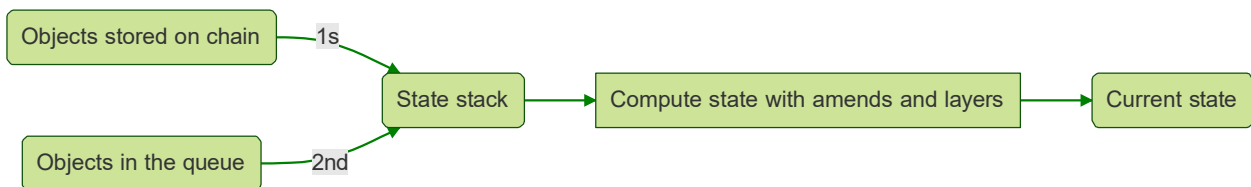In the event a new block is posted with a different history than what is computed off-chain, two possibilities occurr:

- On-chain content is deemed bad (according to specific rules) and discarded in a new commited transaction, off chain content taking its place
- Off-chain content is discarded and clients are notified

3.6.2 VM State content

The VM state object is specific to the engine used (WASM, docker-like language specific VM, JVM…), but typically contains:

- Relation to the previous state (object chain using hashes)
- function called
- Arguments
- function result
- New serialized state (can be a diff only for big states, in special cases)

3.6.3 Concurrency

While view-only functions that don't modify the state can be done concurrently, the write functions can't happen concurrently (or will end up in a fork). A node can send a pubsub message to broadcast it is working on/executing a "write" function on a specific VM. Once that has been completed, it will broadcast the new state to the other nodes in the same pubsub channel, thus enabling them to execute a write function from this point in history.

3.7 Cross-application data exchange amd data ownership

All applications in the YGGDRASIL ECOSYSTEMS -DAN ecosystem are talking to the same data storage entity. Conventions exist and have to be documented on data structures to be used by applications.

This enables applications to use the same user profiles (example .space and .world/social using the same profiles, and .world/apps/vote using the profile and same post format than the formers for its content) and content sources if needed. A new developer can come and make a new frontend to all the existing data posted by users. Users own their data.

3.8 Moderation and requests for removal

3.8.1 Network and data moderation

Each application can have its own storage of black-listed content and addresses to handle its own moderation features (the application will then omit certain content or content creators from the user interface).

for illegal content, the network will also have its own storage that will be synchronised between nodes of blacklisted content and addresses. Linked content hashes will be automatically unpinned from all nodes, leading to its destruction from the YGGDRASIL ECOSYSTEMS -DAN network.

3.8.2 Removal request

To comply with regulations (GPDR in particular), in a similar fashion to the illegal content data unpinning, a user can request certain hashes he posted to be removed, or even his full address data to be put into blacklist by signing a message using his public key.

3.9   Nodes

3.9.1   Clients

Simple clients to the network are either using the API to API node or p2p pubsub connection to the network. If they are using the p2p system then they can act like a full node and validate messages themselves (and run VM themselves if it's available for their platform, WASM working in web browsers).

They should be careful to verify the signatures of the received messages and states if they don't trust the API server or if they are connected directly to the network.

3.9.2   Packing Nodes

The Packing nodes are those which validate the messages and submit their hashes to the underlying blockchains. They get rewarded for their action (token reward) or punished if they don't do their duty correctly.

If a packing node is found committing nefarious acts by the consensus, its address is blacklisted and his rewards (and/or deposit, this is to be defined, see Roadmap chapter) frozen.

Those nodes must be both API nodes and storage nodes too.

3.9.3   Storage Nodes

The storage nodes pin incoming IPfS hashes to ensure distributed (geographically and across multiple hosts to ensure the data won't go away) storage of the content.

Those nodes must be API nodes and may optionally be packing nodes too.

They are rewarded in tokens for their duty.

3.9.4   Full Node

Packing and storage nodes are using this codebase, activating some modules and configuration to do their work.

The APIs are a module that can be activated (mandatory one for incentivized nodes).

features of an YGGDRASIL ECOSYSTEMS -DAN Node:

- Indexes the underlying blockchain it is configured for (can be more than 1).
- Has an IPfS client built-in
- Keeps track of modifications to the chain (forks, history rewrite…)
- Connected to pubsub channels to update its off chain state

- Retrieves IPfS content (from storage nodes or sender) commited onchain or recieved in pubsub to review and index content (and pins it if needed)
- Executes VMs code (if configured for it, and only relevant ones)

3.9.4.1 Indexing

Depending on the underlying chain volume, a full index will be done, or only an index based on specific events (triggered by a smart contract for example -on Ethereum-, a tx type -business data on - or op_return & simi- lar).

for maximum security, the incentivized YGGDRASIL ECOSYSTEMS -DANnodes in charge of sensitive parts of the network will connect to a local node of this blockchain (to detect forks for example), simpler nodes can connect to explorers or public api servers of those chains to act as light wallets. The blocks informations will also be   published   on   IPfS   for   faster   sync   of   nodes

4. YGGDRASIL ECOSYSTEMS -DAN Tokem

4.1 Rewards

Three main sources of reward and token creation are:

- Reward each signed message written to the underlying blockchain
- Reward storage of application data (pin items) and availability of API, both are mandatory for nodes
- POCM token locking in underlying chain (see token distribution section)

Of those rewards, a part will come from monetary creation, and a part from dApp owner incentive to prioritize their apps and their user fees.

If a dApp owner wants to get the benefit of the network without having to rely on third parties only, he can run a node of the network where he prioritizes his dApp data (but he can't completely ignore others, having a backup for his users on other nodes, while he gives the same allowance to other dApps).

4.2 Token use

Most node owners will accept free storage of dApp data, especially at start of the network. Once the data grows bigger, and it begins to become impratical for hosts to have all the data, dApp owners can pay for storage and writes to the blockchain of his dApp data.

Alternatively, application users can pay themselves for the storage of their data (example, big data volume like picture storage applications).

4. 3 Token details

Initially, the first issuance of the YGGDRASIL ECOSYSTEMS -DAN utility token will be made as an YRC-20 token. This will be able to be swapped for the native asset at a later date.

If technically possible (part under research):

- the switch between YRC-20 and native YGGDRASIL ECOSYSTEMS -DAN token will be available in both directions.
- the token will also be made available as an ERC-20 token

4.4   Initial token distribution

The exact details of the initial token distribution plan will be announced at a later date. Below is what is currently planned for the token distribution (can be subject to changes).

To reward the community and ecosystem, a large percentage of the supply will be distributed to various token holders through the POCM system.

1,000,000,000 (1 billion) tokens will be issued initially.

Of those :

- 150M will be devoted to airdrop, marketing and bounties (details of airdrop part will be announced through official channels)
- 150M will be reserved for the community/foundation (100M locked for the POCM mining program and 50M for other incentives)
- 350M for private / institutionnal investors / OTC sales
- 350M to the YGGDRASIL ECOSYSTEMS -DAN team (who will use this for bootstrap period rewards, might be able to sell some for development funding, allocate a part for a community or foundation fund or any other use it might deem necessary)
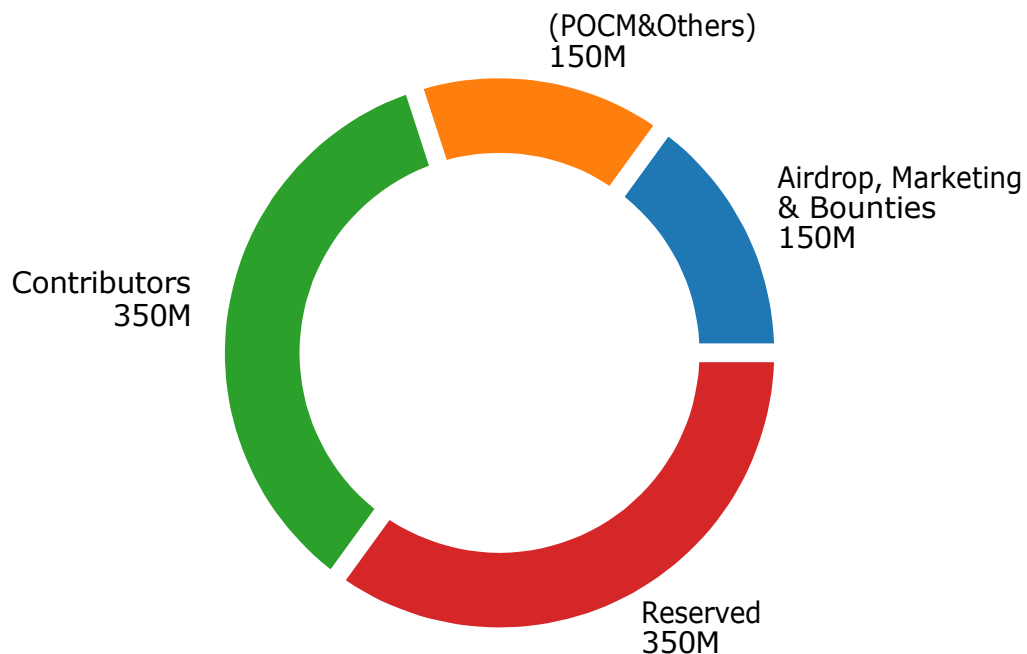
figure 4.1: Token Distribution

4.4.1 DAN staking program (POCM)

In addition to a regular airdrop, holders can lock a part of their tokens to get YGGDRASIL ECOSYSTEMS - DANtokens.

While their tokens are locked, they will receive YGGDRASIL ECOSYSTEMS -DAN tokens based on the amount they have "staked"/"locked". The reward will be higher at the start of the POCM period (approximately 10 years) and lower at the end.

Those tokens will come from an initial pool of 200M tokens, locked for that specific purpose. Once that pool dries up at the end of the period, the reward will be lowered and the tokens minted using this program will be taken from the Supply Evolution.

4.5 Supply evolution

Once the full network is ready (see roadmap chapter) and through the described rewards, the token supply will grow by approximately 10% the first year. Then the inflation rate will decrease by 1% each year until it reaches 1% per year and stay at that value.

Table 4.1: Supply per year

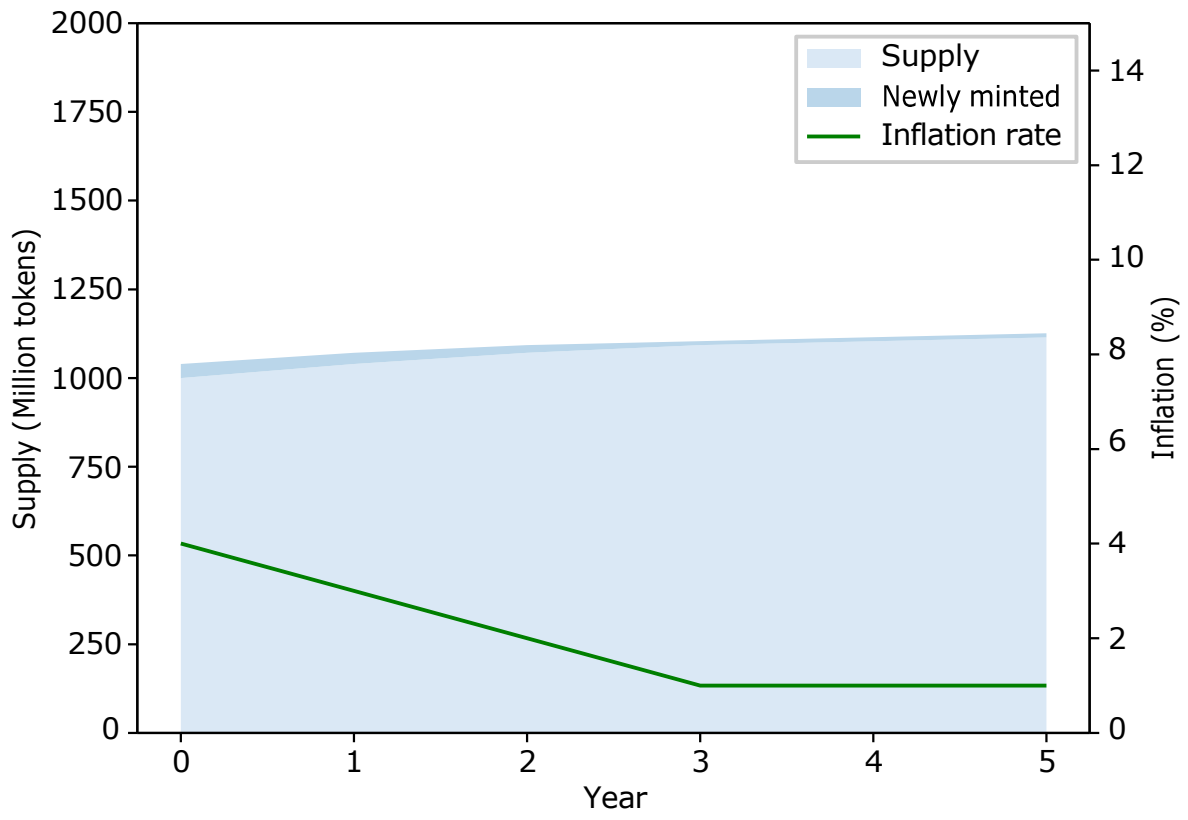|	1st year	2nd year	3rd year	4th year	5th year	5th year
inflation	4%	3%	2%	1%	1%	1%
tokens minted	40M	31.2M	21.42M	10.y2M	11.03M	11.14M
total supply eoy	1040M	1071M	10y2M	1103M	1114M	1125M

figure 4.2: Supply evolution

5. Roadmap

5.1 Genesis ,

The only address format and public keys supported on the network will be addresses.

At the start of this period, a proof of concept will be ready and applications can be developed on Aleph, supporting all but cross-chain and virtual machines.

5.2 Cross-chain,

The cross-chain infrastructure will be developed at this milestone, allowing other types of addresses to be used on the YGGDRASIL ECOSYSTEMS -DAN network (Ethereum first).

5.3 Placeholder token issuance,

A placeholder token is issued as an yrc-20 token (and possibly erc-20), the POCM system starts.

In the first step of the project, the token will live on the blockchain as an NRC-20 token. Rewards to the nodes of the network will be given from the YGGDRASIL ECOSYSTEMS -DAN reserve funds.

5.4 YGGDRASIL ECOSYSTEMS -DAN storage dApps SDK

YGGDRASIL ECOSYSTEMS -DAN dApps based on YGGDRASIL ECOSYSTEMS -DAN storage (posts and aggregates) will have an SDK available for JavaScript client-side code. With login using underlying chain addresses (web3, ledger wallet…).

5.5 IoT SDK

A SDK will be made available for IoT embedded devices, using MicroPython and/or Arduino-compatible code.

5.6 Virtual machines and Mainnet

The "Smart Apps" virtual machine will be developed for this milestone. The 1st version of DAN main net will be started.

5.7 YGGDRASIL ECOSYSTEMS -DAN VM dApps SDK,

Full YGGDRASIL ECOSYSTEMS -DAN dApps based on YGGDRASIL ECOSYSTEMS -DAN virtual machines will have an SDK available, both for the virtual machine code, and JavaScript client-side code. With login using underlying chain addresses (web3, ledger wallet…).

5.8 Network Nodes and Economy,

The Yggdrasil-DAN native token will be minted. A token swap will take place from the NRC-20 (and ERC-20 if issued) token to the native token.

If technically possible, other chains tokens (YGGDRASIL ECOSYSTEMS -DAN token as Yrc-20, erc-20 or nep-5 for example) will be able to be swapped from and to the YGGDRASIL ECOSYSTEMS -DAN native token.

The network nodes will now receive their token rewards corresponding to the plan described in the Token chapter.

5.9 Specialized/Hybrid Nodes,

The priorization feature will now be developed and all nodes won't store all the data or run all the VMs. Some nodes can focus on a few dApps for their owners while still being able to help the whole network (and their dApp content can still be accessible to others).

5.10 Underlying chains acceleration and SDK,

A SDK will be made available at this milestone to easily migrate underlying chain dApps and accelerate them (Ethereum first).

6. Notes and appendices

6.1   Hardware wallet and 2 factor support

Since the signing is done using the underlying chains mechanisms, their support for hardware wallets is used. Wherever possible, those supports will be used.

Examples:

- hardware wallet support done on ledger, we use the built-in signing mechanism to sign the messages.
- Ethereum supports Ledger and Trezor, we use their support and signing mechanism.

Another way to go is using the web authn standard, that will use whatever hardware support is available (built-in crypto chip in mobile device as 2 factor auth, usb u2f key), specific to used website. Wherever possible, webauthn support will be added (generating addresses using secp 255k1 support). An exception to the underlying protocol signing mechanism can be done to support json object signing specific to webauthn.

References

Drake, Kyle. 2015. "HTTP Is Obsolete. It's Time for the Distributed, Permanent Web." September 8, 2015. https://ipfs.io/ipfs/QmNhfJjGcMPqpuYfxL52VVBy528NXqDNMfXiqN5bgfYiZ1/its-time-for-the-permanent-web.html.