cyberd: Computing the knowledge from web3

Notes on `cyber` release of `cyber://` protocol reference implementation using Go.

cyber•Congress: @xhipster, @litvintech, @hleb-albau, @arturalbov

Abstract

A consensus computer allows computing of provably relevant answers without opinionated blackbox intermediaries such as Google, Youtube, Amazon or Facebook. Stateless content-addressable peer-to-peer communication networks such as IPFS and stateful consensus computers such as Ethereum provide part of the solution, but there are at least three problems associated with implementation. Of course, the first problem is the subjective nature of relevance. The second problem is that it is hard to scale consensus computer for a huge knowledge graph. The third problem is that the quality of such a knowledge graph will suffer from different attack surfaces such as sybil, selfish behaviour of interacting agents. In this paper, we (1) define a protocol for provable consensus computing of relevance between IPFS objects based on Tendermint consensus of cyber•rank computed on GPU, (2) discuss implementation details and (3) design distribution and incentive scheme based on our experience. We believe the minimalistic architecture of the protocol is critical for the formation of a network of domain-specific knowledge consensus computers. As a result of our work some applications never existed before emerge. We expand the work with our "after Genesis vision" on features and apps.

Introduction to web3

Original protocols of the Internet such as TCP/IP, DNS, URL, and HTTPS brought a web into the point where it is now. Along with all the benefits they have created they brought more problem to the table. Globality being a vital property of the web since inception is under real threat. The speed of connections degrades with network grow and from ubiquitous government interventions into privacy and security of web users. One property, not evident in the beginning, become important with everyday usage of the Internet: it's ability to exchange permanent hyperlinks thus they would not break after time has passed. Reliance on "one at a time ISP" architecture allows governments effectively censor packets. It is the last straw in a conventional web stack for every engineer who is concerned about the future of our children.

Other properties while being not so critical are very desirable: offline and real-time. Average

internet user being offline must have the ability to work with the state it has and after acquiring connection being able to sync with global state and continue to verify state's validity in realtime while having a connection. Now, these properties offered on the app level while such properties must be integrated into lower level protocols.

The emergence of a web3 stack creates an opportunity for a new kind of Internet. Community call it web3. We call it "The Great Web" as it is expected that some low level conventions must become immutable and not being changed for decades. e.g. immutable content links. It has a promise to remove problems of a conventional protocol stack and add to the web better speed and more accessible connection. However, as usual in a story with a new stack, new problems emerge. One of such problem is general-purpose search. Existing general-purpose search engines are restrictive centralized databases everybody forced to trust. These search engines were designed primarily for client-server architecture based on TCP/IP, DNS, URL and HTTPS protocols. Web3 creates a challenge and opportunity for a search engine based on developing technologies and specifically designed for them. Surprisingly the permission-less blockchain architecture itself allows organizing general purpose search engine in a way inaccessible for previous architectures.

On adversarial examples problem

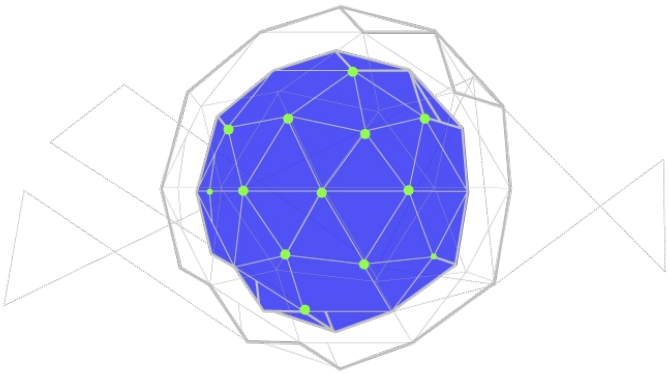
Conventional architecture of search engines there one entity process and rank all the shit suffers from one hard but the particular problem that still has not been solved even by brilliant Google scientists: adversarial examples problem. The problem Google acknowledge is that it is rather hard to algorithmically reason either this particular sample is adversarial or not independently on how cool the learning technology is. Obviously, a cryptoeconomic approach can change beneficiaries in this game effectively removing possible sybil attack vectors and removing the necessity to make a decision on example crawling and meaning extraction from one entity to the whole world. Learning sybil-resistant model will probably lead to orders of magnitude more predictive results.

Cyber protocol at `cyber`

- compute `cyber` inception of cyber protocol based on the Genesis distribution rules
- def knowledge graph state
- take cyberlinks
- check the validity of signatures
- check bandwidth limit
- check the validity of CIDv0
- if signatures, bandwidth limit, and CIDv0 are valid than cyberlink is valid
- every round calculate cyber•rank deltas for the knowledge graph

Knowledge graph

We represent a knowledge graph as a weighted graph of directed links between content addresses, or content identifications, or CIDs, or simply ipfs hashes. In this paper, we will use them as synonyms.



Content addresses are essentially a web3 links. Instead of using non-obvious and mutable thing:

```
https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md
```

we can use pretty much exact thing:

```
Qme4z71Zea9xaXScUi6pbsuTKCCNFp5TAv8W5tjdfH7yuH
```

Using content addresses for building a knowledge graph we get so much needed superpowers of ipfs-like p2p protocols for a search engine:

- mesh-network future proof
- interplanetary
- tolerant
- accessible
- technology agnostic

Web3 agents generate our knowledge graph. Web3 agents include itself to the knowledge graph by transacting only once. Thereby they prove the existence of private keys for content addresses of revealed public keys. Using this basic proof mechanics consensus computer could have provable differentiation between subjects and objects in a knowledge grapth.

Our `cyber` implementation is based on `cosmos-sdk` identities and `cidv0` content addresses.

Web 3 agents generate knowledge graph by applying `cyberlinks`.

Cyberlinks

To understand cyberlinks, we need to understand the difference between `URL link` aka hyperlink and `IPFS link`. `URL link` points to the location of content, but `IPFS link` point to the content itself. The difference in web architecture based on location links and content links is drastical, hence require new approaches.

`Cyberlink` is an approach to link two content addresses or `IPFS links` semantically:

```
QmdvsvrVqdkzx8HnowpXGLi88tXZDsoNrGhGvPvHBQB6sH.Qme4z71Zea9xaXScUi6pbsuTKCCNFp5TAv8W
```

This `cyberlink` means that cyberd presentation on cyberc0n is referencing Cosmos whitepaper. A concept of `cyberlink` is a convention around simple semantics of communication format in any peer to peer network:

`<content-address x>.<content-address z>`

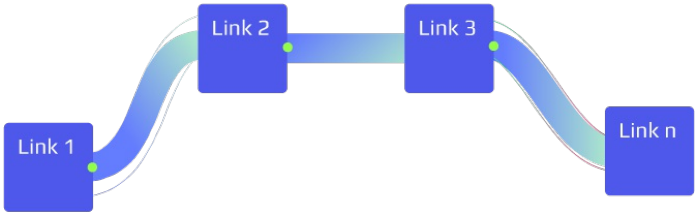
You can see that `cyberlink` represents a link between two links. Easy peasy!

Cyberlink is a simple yet powerful semantic construction for building a predictive model of the universe. That is, using `cyberlinks` instead of `hyperlinks` gives us superpowers inaccessible for previous architectures of general purpose search engines.

Cyberlinks can be extended, e.g. can form link chains if exist a series of two cyberlinks from one agent in which the second link in the first cyberlink is equal to the first link in the second cyberlink:

```
<content-address x>.<content-address z>
<content-address z>.<content-address z>
```

Using this simple principle, all interacting agents can reach consensus around interpreting clauses. So link chains are helpful for interpreting rich communications around relevance.



Also using the following link: `QmNedUe2wktW65xXxWqcR8EWWssHVMXm3Ly4GKiRRSEBkn` the one can signal the start and stop of execution in the knowledge graph. A lot of cool stuff can be done using cyberlinks.

If web3 agents expand native `IPFS links` with something semantically richer as `DURA` links than web3 agents can easier reach consensus on the rules for program execution. Indeed, `DURA` protocol is a proper implementation of a cyberlink concept.

`cyber` implementation of `cyberlinks` based on `DURA` specification is available in `.cyber` app of browser `cyb`.

Based on `cyberlinks` we can compute the relevance of subjects and objects in a knowledge graph. That is why we need a consensus computer.

Notion of consensus computer

Consensus computer is an abstract computing machine that emerges from agents interactions.

A consensus computer has a capacity in terms of fundamental computing resources such as memory and computing. To interact with agents, a computer needs a bandwidth.

Ideal consensus computer is a computer in which:

```
the sum of all computations and memory available for individuals
is equal to
the sum of all verified computations and memory of a *consensus computer*
```

We know that:

```
verifications of computations < computations + verifications of computations
```

Hence we will not be able to achieve an ideal consensus computer ever. CAP theorem and scalability trilemma also prove this statement.

However, this theory can work as a performance indicator of a consensus computer.

After 6 years of investments into consensus computers we find out that Tendermint consensus has a good balance between coolness for our task and readiness for production. So we decide to implement cyber protocol using Tendermint which is very close to Cosmos Hub setting.

The `cyber` implementation is a 64-bit tendermint consensus computer of the relevance for 64-byte string space that is as far from ideal at least as 1/146, because we have 146 validators who verify the same computation using the knowledge graph of the same size.

We must bind computational, storage and bandwidth supply of consensus computer with maximized demand on queries. Computation and storage in case of basic relevance machine can be easily predicted based on bandwidth, but bandwidth requires a limiting mechanism.

Bandwidth

`Bonded stake` - stake, that deducted from your acc coins and put as deposit to take part in consensus. Due to passive inflation model and slashing, deposit not match 1-to-1 to final reward. So, for example, stakeholders may wish to set up a script, that will periodically withdraw and rebound rewards to increase their bonded stake.

`Active stake` - currently available for direct transfer, not-bonded stake.

`Bandwidth stake` = `active stake + bonded stake`.

Cyberd use a very simple bandwidth model. Main goal of that model is to reduce daily network growth to given constant, say 3gb per day.

Thus, here we introduce `resource credits`, or RS. Each message type have assigned RS cost. There is constant `DesirableNetworkBandwidthForRecoveryPeriod` determining desirable for `RecoveryPeriod` spent RS value. `RecoveryPeriod` is defining how fast user can recover their bandwidth from 0 to user max bandwidth. User has maximum RS proportional to his stake by formula:

`user_max_rc = bandwidth_stake * DesirableNetworkBandwidthForRecoveryPeriod`

There is period `AdjustPricePeriod` summing how much RS was spent for that period `AdjustPricePeriodTotalSpent`. Also, there is constant `AdjustPricePeriodDesiredSpent`, used to calculate network loading.

`AdjustPricePeriodTotalSpent / AdjustPricePeriodDesiredSpent` ratio is called fractional reserve ratio. If network usage is low, fractional reserve ratio adjust message cost (by simple multiplication) to allow user with lower stake to do more transactions.
If resource demand increase, fractional reserve ratio goes `>1` thus increase messages cost and limiting final tx count for some long-term period (RC recovery will be `<` then RC spending).

There are only two ways to change acc bandwidth stake:

1. Direct coins transfer.
2. When distribution payouts occurs. For example, when validator change his commission rates, all delegations will be automatically unbounded. Another example, delegator itself unbond some part or full share.

So agents must have CYB tokens in accordance to their will of learning the knowledge graph. However, proposed mechanics of CYB tokens work not only as spam protection but as the economic regulation mechanism to align the ability of validators to process knowledge graph and market demand for processing.

Relevance machine

We define relevance machine as a machine that transition knowledge graph state based on the will of agents to learn the knowledge graph. The more agents will learn the knowledge graph the more valuable the graph become. This machine enables simple construction for search question querying and answers delivering.

The will is projected on every agent's cyberlink. A simple rule prevents abuse by agents: one content address can be voted by a coin only once. So it does not matter for ranking from how much accounts you voted. The only sum of their balances matters.

A useful property of a relevance machine is that it must have inductive reasoning property or follows the blackbox principle.

```
She must be able to interfere predictions
without any knowledge about objects
except who cyberlinked, when cyberlinked and what was cyberlinked.
```

If we assume that a consensus computer must have some information about linked objects the complexity of such model growth unpredictably, hence a requirement for a computer for memory and computations. That is, deduction of meaning inside consensus computer is expensive thus our design depends on the blindness assumption. Instead of deducting a meaning inside consensus computer we design a system in which meaning extraction is incentivized because agents need CYB to compute relevance.

Also, thanks to content addressing the relevance machine following the blackbox principle do not need to store the data but can effectively operate on it.

Human intelligence organized in a way to prune none-relevant and none-important memories with time has passed. The same way can do relevance machine. Also, one useful property of relevance machine is that it needs to store neither past state, nor full current state to remain useful, or more precisely: *relevant*. So relevance machine can implement aggressive pruning strategies such as pruning all history of knowledge graph formation or forgetting links that become non-relevant.

`cyber` implementation of relevance machine is based on the most straightforward mechanism which is called cyber•Rank.

cyber•Rank

Ranking using consensus computer is hard because consensus computers bring serious resource bounds. e.g. Nebulas still fail to deliver something useful on-chain. First, we must ask ourselves why do we need to compute and store the rank on-chain, and not go Colony or Truebit way?

If rank computed inside consensus computer, you have an easy content distribution of the rank as well as an easy way to build provable applications on top of the rank. Hence we decided to follow more cosmic architecture. In the next section, we describe the proof of relevance mechanism which allows the network to scale with the help of domain-specific relevance machines that works in parallel thanks to IBC protocol.

Eventually, relevance machine needs to find (1) deterministic algorithm that allows computing a rank for a continuously appended network to scale the consensus computer to orders of magnitude that of Google. Perfect algorithm (2) must have linear memory and computation complexity. The most importantly it must have (3) highest provable prediction capabilities for the existence of relevant cyberlinks.

After some research, we found that we can not find silver bullet here. So we decided to find

some more basic bulletproof way to bootstrap the network: the rank from which Larry and Sergey have bootstrapped a previous network. The key problem with original PageRank is that it is not resistant to sybil attacks.

Token weighted PageRank limited by token-weighted bandwidth do not have inherent problems of naive PageRank and is resistant to sybil attacks. For the time being, we will call it cyber•Rank until something better emerge.

In the centre of spam protection system is an assumption that write operations can be executed only by those who have a vested interest in the evolutionary success of a relevance machine. Every 1% of stake in consensus computer gives the ability to use 1% of possible network bandwidth and computing capabilities.

As nobody uses all possessed bandwidth, we can safely use up to 100x fractional reserves with 2-minute recalculation target. This mechanics offers a discount for cyberlinking thus effectively maximazing demand for it.

We would love to discuss the problem of vote buying mainly. Vote buying by itself is not such bad. The problem with vote buying appears in the systems where voting affects the allocation of inflation in the system like Steem or any state-based system. So vote buying can become easily profitable for adversary employing a zero-sum game without a necessity to add value. Our original idea of a decentralized search was based on this approach, but we reject this idea removing incentive on consensus level for knowledge graph formation completely. In our setting in which every participant must bring some value to the system to affect predictive model vote buying become NP-hard problem hence is useful for the system.

We understand that the ranking mechanism will always remain red herring. That is why we expect to rely on on-chain governance mechanism to define the wining one. We would love to switch from one algorithm to another, based on economic a/b testing through hard spoons of domain specific relevance machines though.

Current implementation of consensus computer based on relevance machine for cyber•Rank can answer and deliver relevant results for any given search request in the 64 byte CID space. However, to build a network of domain-specific relevance machines, it is not enough. Consensus computers must have the ability to prove relevance for each other.

Proof of relevance

We design a system under the assumption that regarding search such thing as bad behaviour does not exist as anything bad can be in the intention of finding answers. Also, this approach significantly reduces attack surfaces.

> Ranks are computed on the only fact that something has been searched, thus linked and as a result, affected the predictive model.

A good analogy is observing in quantum mechanics. That is why we do not need such things as negative voting. Doing this we remove subjectivity out of the protocol and can define proof of relevance.

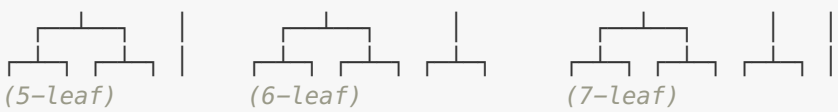
```
Rank state = rank values stored in a one-dimensional array and merkle tree of those
```

Each new CID gets a unique number. The number starts from zero and incrementing by one for each new CID. So that we can store rank in a one-dimensional array where indices are CID numbers.

Merkle Tree calculated based on RFC-6962 standard. Since rank stored in a one-dimensional array where indices are CID numbers (we could say that it ordered by CID numbers) leaves in merkle tree from left to right are `SHA-256` hashes of rank value. Index of the leaf is CID number. It helps to easily find proofs for specified CID (`log n` iterations where `n` is a number of leaves).

To store merkle tree is necessary to split the tree into subtrees with a number of leaves multiply of the power of 2. The smallest one is obviously subtree with only one leaf (and therefore `height == 0`). Leaf addition looks as follows. Each new leaf is added as subtree with `height == 0`.
Then sequentially merge subtrees with the same `height` from right to left.

Example:



To get merkle root hash - join subtree roots from right to left.

Rank merkle tree can be stored differently:

Full tree - all subtrees with all leaves and intermediary nodes
Short tree - contains only subtrees roots

The trick is that *full tree* is only necessary for providing merkle proofs. For consensus

purposes and updating tree, it's enough to have a *short tree*. To store merkle tree in database use only a *short tree*. Marshaling of a short tree with `n` subtrees (each subtree takes 40 bytes):

```
<subtree_1_root_hash_bytes><subtree_1_height_bytes>
....
<subtree_n_root_hash_bytes><subtree_n_height_bytes>
```

For `1,099,511,627,775` leaves *short tree* would contain only 40 subtrees roots and take only 1600 bytes.

Let us denote rank state calculation:

`p` - rank calculation period
`lbn` - last confirmed block number
`cbn` - current block number
`lr` - length of rank values array

For rank storing and calculation we have two separate in-memory contexts:

1. Current rank context. It includes the last calculated rank state (values and merkle tree) plus
   all links and user stakes submitted to the moment of this rank submission.
2. New rank context. It's currently calculating (or already calculated and waiting for submission) rank state. Consists of new calculated rank state (values and merkle tree) plus new incoming links and updated user stakes.

Calculation of new rank state happens once per `p` blocks and going in parallel.

The iteration starts from block number that `≡ 0 (mod p)` and goes till next block number that `≡ 0 (mod p)`.

For block number `cbn ≡ 0 (mod p)` (including block number 1 cause in cosmos blocks starts from 1):

1. Check if the rank calculation is finished. If yes then go to (2.) if not - wait till calculation finished
   (actually this situation should not happen because it means that rank calculation period is too short).
2. Submit rank, links and user stakes from new rank context to current rank context.
3. Store last calculated rank merkle tree root hash.
4. Start new rank calculation in parallel (on links and stakes from current rank context).

For each block:

1. All links go to a new rank context.
2. New coming CIDs gets rank equals to zero. We could do it by checking last CIDs number and `lr` (it equals the number of CIDs that already have rank). Then add CIDs with number `>lr` to the end of this array with the value equal to zero.
3. Update current context merkle tree with CIDs from the previous step
4. Store latest merkle tree from current context (let us call it last block merkle tree).
5. Check if new rank calculation finished. If yes go to (4.) if not go to next block.
6. Push calculated rank state to new rank context. Store merkle tree of newly calculated rank.

To sum up. In *current rank context*, we have rank state from last calculated iteration (plus, every block, it updates with new CIDs). Moreover, we have links and user stakes that are participating in current rank calculation iteration (whether it finished or not). The *new rank context* contains links and stakes that will go to next rank calculation and newly calculated rank state (if a calculation is finished) that waiting for submitting.

If we need to restart node firstly, we need to restore both contexts (current and new).
Load links and user stakes from a database using different versions:

1. Links and stakes from last calculated rank version `v = lbn - (lbn mod n)` go to current rank context.
2. Links and stakes between versions `v` and `lbn` go to new rank context.

Also to restart node correctly, we have to store following entities in database:

1. Last calculated rank hash (merkle tree root)
2. A newly calculated rank short merkle tree
3. Last block short merkle tree

With *last calculated rank hash* and *newly calculated rank merkle tree* we could check if the rank calculation was finished before node restart. If they are equal, then rank wasn't calculated, and we should run the rank calculation. If not we could skip rank calculation and use *newly calculated rank merkle tree* to participate in consensus when it comes to block number `cbn ≡ 0 (mod p)` (rank values will not be available until rank calculation happens in next iteration. Still validator can participate in consensus so nothing bad).

Last block merkle tree necessary to participate in consensus till the start of next rank calculation iteration. So, after the restart we could end up with two states:

1. Restored current rank context and new rank context without rank values (links, user

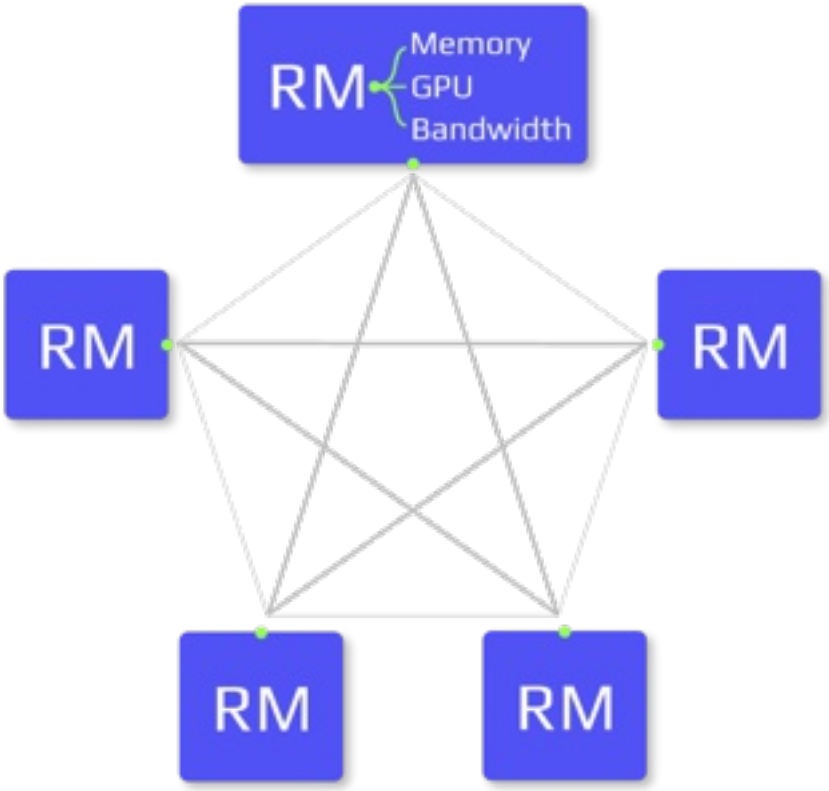
stakes, and merkle tree).
2. Restored current rank context without rank values. Restored new rank context only with links and user stakes.

A node can participate in consensus but cannot provide rank values (and merkle proofs) till two rank calculation iterations finished (current and next). Search index should be run in parallel and do not influence the work of the consensus machine. The validator should be able to turn off index support.

Now we have proof of rank of any given content address. While the relevance is still subjective by nature, we have a collective proof that something was relevant for some community at some point in time.

```
For any given CID it is possible to prove the relevance
```

Using this type of proof any two IBC compatible consensus computers can proof the relevance to each other so that domain-specific relevance machines can flourish. Thanks to inter-blockchain communication protocol you basically can either launch your own domain-specific search engine by forking cyberd which is focused on the *common public knowledge*, or plug cyberd as module in existing chain, e.g. Cosmos Hub. So in our search architecture, domain-specific relevance machine can learn from common knowledge.



In our relevance for commons `cyber` implementation proof of relevance root hash is computed on Cuda GPUs every round.

Speed and scalability

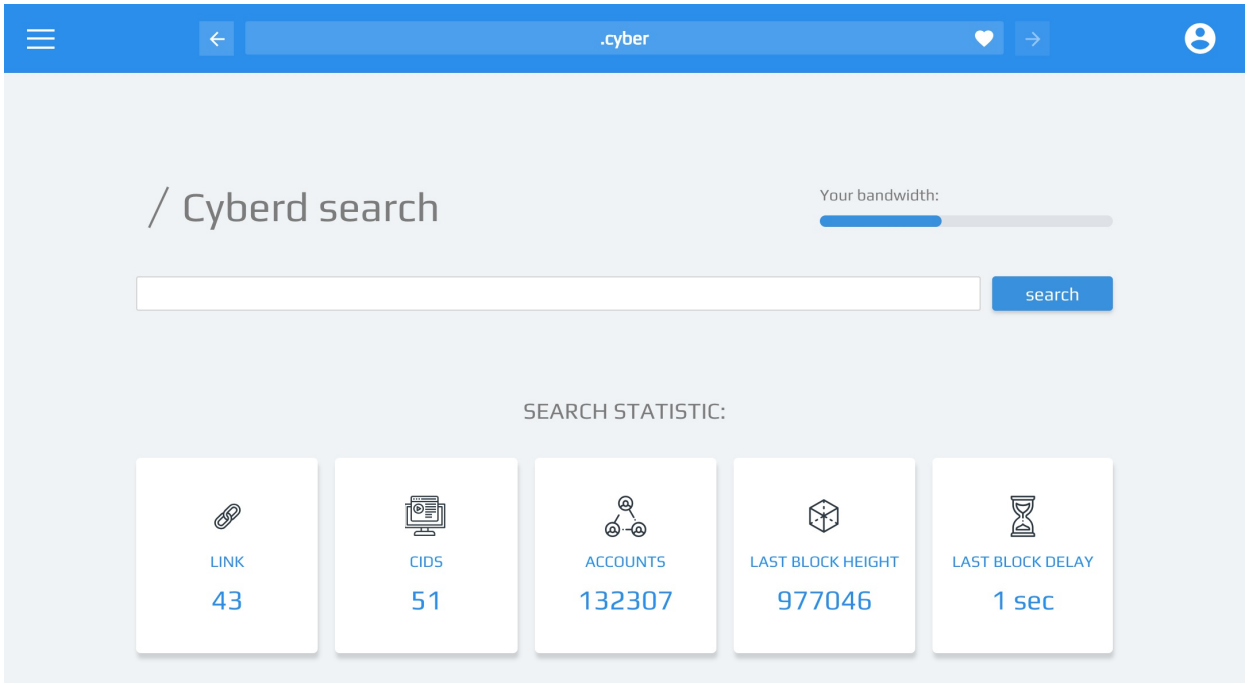
We need speedy confirmation times to feels like the usual web app. It is a strong architecture requirement that shape an economic topology and scalability of the cyber protocol.

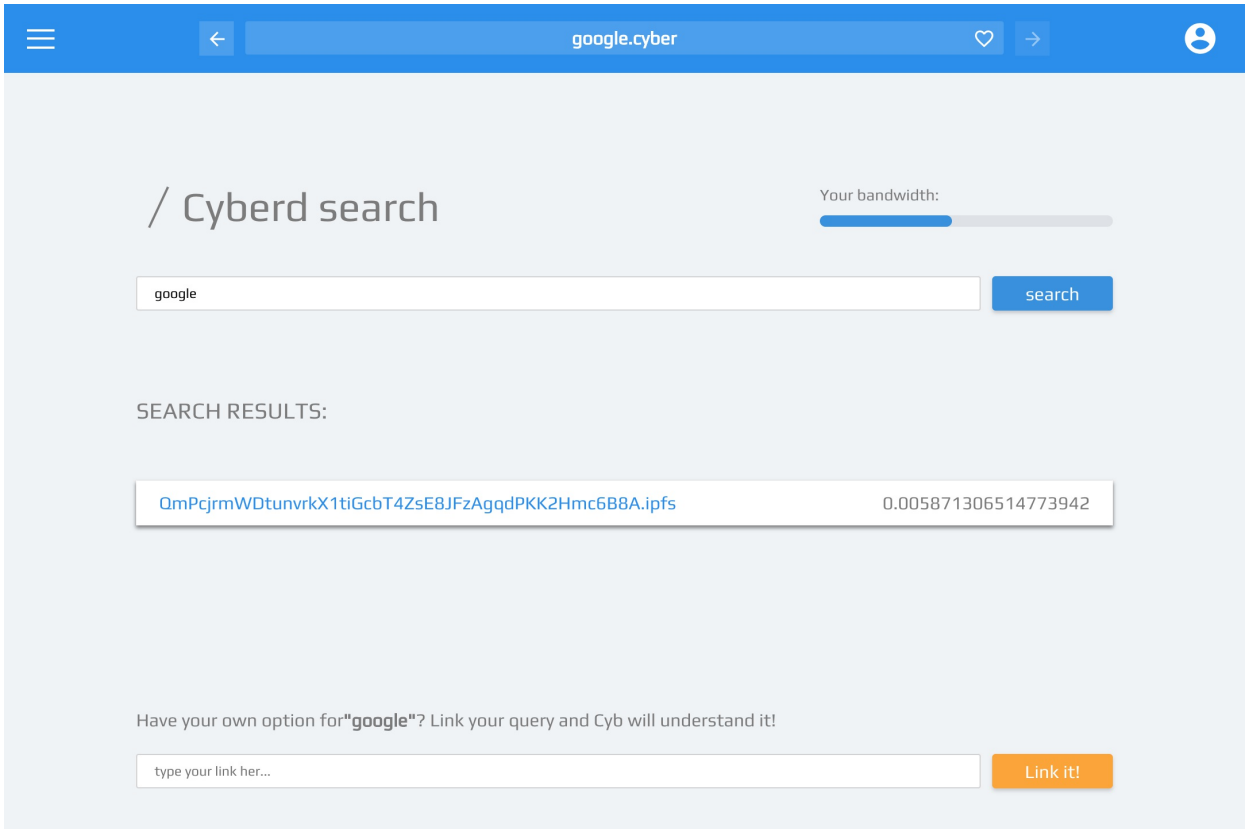
Proposed blockchain design is based on Tendermint consensus algorithm with 146 validators and has very fast 2 second finality time. Average confirmation timeframe at half the second with asynchronous interaction make complex blockchain search almost invisible for agents.

Let us say that our node implementation based on `cosmos-sdk` can process 10k transactions per second. Thus every day at least 8.64 million agents can submit 100 cyberlinks each and impact results simultaneously. That is enough to verify all assumptions in the wild. As blockchain technology evolves we want to check that every hypothesis work before scale it further. Moreover, proposed design needs demand for full bandwidth in order the relevance become valuable. That is why we strongly focus on accessible, but provable distribution from inception.

In-browser implementation

We wanted to imagine how that could work in a web3 browser. To our disappointment we was not able to find the web3 browser that can showcase the coolness of the proposed approach in action. That is why we decide to develop the web3 browser cyb that has sample application .cyber for interacting with `cyber://` protocol.





As another good example, we created Chrome extension that allow anybody to pin any web page to ipfs and index it by the any keywords, thus make it searchable.

Current search snippets are ugly, but we expect they can be easily extended using IPLD for different types of content so they can be even more beautiful than that of Google.

During implementation of proposed architecture we realize at least 3 key benefits a Google probab;y would not be able to deliver with conventiaon approach:

- search result can be easily delivered from p2p network right into search results: eg. .cyber can play video.
- payment buttons can be embedded right into search snippets, so web3 agent can interact with search results, eg. agents can buy an items right in `.cyber`. Also so e-commerce can flourish because of transparent conversion attribution.
- search snippets must not be static but can be interactive, eg. `.cyber` eventually can answer location based answers.

Philosophical approach toward distribution

While designing the initial distribution structure for Cyber protocol we aimed to achieve the following goals:

- Develop provable and transparent distribution in accordance with best industry practices
- Allow equal participation irrespectively of political, regulatory or any other restrictions which may be imposed by outside agents
- Prevent attacks on privacy such as installment of KYC requirements
- Spread distribution in time to grant equal access to all agents to initial distribution without any limitations such as hard caps or any other restrictions
- Honour genesis Cosmos investors for development of technology which made possible simplified development of Cyber protocol
- Attract the most professional validators from Cosmos ecosystem for bootstrapping the network
- Allow easy early access for active users of Ethereum ecosystem in order to accelerate growth of the Knowledge graph and solve chicken and egg problem
- Decentralize management of auction donations starting from day 0
- Honour 30 months of cyber•Congress R&D and community behind it

The goal of creating alternative to a Google-like structure requires extraordinary effort of different groups. So we decide to setup cyber•Foundation as a fund managed via decentralized engine such as Aragon DAO filled with ETH and managed by agents who participated in initial distribution. This approach will allows to safeguard from excessive market dumping of native platform CYB tokens in the first years of work, whereby ensuring stable

development. Additionally this allows to diversify underlying platform and extend the protocol to other consensus computing architecture should the need arise.

While choosing token for donations we followed three main criteria: the token must be (1) one of the most liquid, (2) the most promising, so a community can secure a solid investment bag to be competitive even comparing to giants like Google and (3) have technical ability to execute auction and resulting organization without relying on any third party. So the only system matches this criteria is Ethereum, hence the primary token of donations will be ETH. That is why we decide to create 2 tokens: THC and CYB:

- THC is a creative cyber proto substance. THC being an Ethereum ERC-20 compatible token have utility value in form of control cyber•Foundation (Aragon DAO) ETH from auction proceeds. THC was emitted during the creation of cyber•Foundation as Aragon organization. Creative power of THC came from ability to receive 1 CYB per each 1 THC for locking it during Game of Thrones and cyber•Auction.
- CYB is native token of sovereign Cyber protocol under Tendermint consensus algorithm. It's also has 2 primary uses: (1) is a staking for consensus and (2) is bandwidth limiting for submitting links and computing the rank.

Both tokens remains functional and will track value independently due to very different utility nature.

Initial distribution happens in a 3 different by nature and goals epochs and is spread in time for almost 2 years:

1. Pre-genesis: 21 days. Is needed to launch cyber protocol in a decentralized fashion with independent genesis validators.

2. Genesis: 21 days. Launch of main net and Game of Thrones: needed to involve the most active Ethereum and Cosmos crypto players into an engaging game with ability to fully understand and test a software using real network and incentives. Game of Thrones is a kind of distribution game with some discount for attraction of necessary critical mass in order to make possible of learning the early knowledge graph by the most intelligent community.

3. Post Genesis: 600 days. Continuous distribution of CYB based on cyber•Auction proceeds: needed in order to involve into initial distribution existing crypto community and beyond.

Pre-genesis

Only 2 distribution events happens happens prior to Genesis:

1. 700 000 000 000 000 THC tokens are minted by cyber•Foundation. Allocations of THC tokens is the following:

- 100 000 000 000 000 THC is allocated to cyber•Congress
- 600 000 000 000 000 THC is allocated to cyber•Auction contract

1. 4 July 2019 donation round in ATOMs will be started for validators who want participate in the Genesis of the network. The round will be limited with either 21 days or 100000 ATOMs. All excess ATOMs will be counted as Game of Thrones contribution. This round is necessary because the network must be started by independent validators. 5% of CYB will be allocated to participants of this donation round in Genesis.

Genesis and Game of Thrones

Genesis of the network and contracts for Game of Thrones will be launched at 04 September 2019

Genesis of `cyber` protocol will contains 1 000 000 000 000 000 CYB (One Quadrillion CYB) broken down as follows:

- 600 000 000 000 000 CYB under multisig managed by cyberCongress for manual distributions during cyber•Auction for those who stake THC until the end of cyber•Auction
- 200 000 000 000 000 CYB under multisig managed by cyberCongress: the Game of Thrones for ATOM and ETH holders, 100 TCYB for each.
- 100 000 000 000 000 CYB for top 80% ETH holders by stake excluding contracts
- 50 000 000 000 000 CYB as drop for all ATOM stakeholders
- 50 000 000 000 000 CYB for pre-genesis contributors in ATOM

Game of Thrones - is a game between ATOM and ETH holders for being the greatest. As a result of 21 day auction after Genesis every community earn 10% of CYB. In order to make the game run smoothly we concisely adding arbitrage opportunity in the form of significant discount to ATOM holders, because the system needs provably professional validators and delegators at the beginning and basically for free.

We can describe the discount in the following terms: Currently buying power of all ATOMs against all ETHs based on current caps is about 1/24. Given that 10% of CYB will be distributed based on donation in ATOMs and 10% of CYB will be distributed based on donations in ETHs the discount for every ATOM donation during Game of Thrones is about 24x which is significant enough to encourage participation based on arbitrage opportunity during the first 21 days of Genesis auction and stimulate the price of ATOMs as appreciation for all cosmic community.

Post-genesis and cyber•Auction

Post Genesis stage called cyber•Auction starts after the end of the Game of Thrones and lasts 600 rounds 23 hours each. During this phase CYBs are continuously distributed based on locked THC bough in continuous auction.

The role of cyber•Auction is twofold:

- It creates non-exlusive long lasting and provable game of initial distribution without necessity to spend energy on proof of work. It is crucial that early knowledge graph were created in some sense fairly by engaged community which was formed during a non-exclusive game.
- As a result of auction community will has access to all raised resources under Aragon organisation. We believe in a true decentralized nature of the thing we created so we do not want to grab all the money from the funding as we already funded the creation of the system ourselves and we kindly ask fair 10% CYB cut for pre-genesis investors, founders and developers. Competing with Google is challenging and will be more viable if community will sits on the bag of ever-growing ETH. Given current growth rate of ETH this bag can be very compelling in some years after launch. Also this bag can be the source of alternative implementation of the protocol in the cased the community will want to diversify technology involved, e.g. ETH2, Polkadot or whatever.

After genesis CYB tokens can be created only by validators based on staking and slashing parameters. The basic consensus is that newly created CYB tokens are distributed to validators as they do the essential work to make relevance machine run both regarding energy consumed for computation and cost for storage capacity. So stakeholders decide where the tokens can flow further.

After Genesis inflation adjusted using `TokensPerBlock` parameter. Given that the network has 1 second target block and ~7% target inflation the starting parameter will be 50 MCYB.

There is no currently such thing as maximum amount of CYB due to continuous inflation paid to validators. Currently CYB is implemented using 64int so creation of more CYB make significantly more expensive compute state changes and rank. We expect that lifetime monetary strategy must be established by governance system after complete initial distribution of CYB and activation of smart contract functionality.

The following rules apply for CYBs under cyber•Auction multisig:

- will not delegate its stake and as result will remain as passive stake until become distributed
- after the end of cyber•Auction all remaining CYBs will be provably burned

Role of ATOMs

Overall 15% of CYB will be distributed based on donations in ATOMs during 2 rounds:

- 50 000 000 000 000 CYB for genesis ATOM contributors
- 100 000 000 000 000 CYB for ATOM contributors at the start of Smith Epoch

All ATOM donations goes to cyber•Congress multisig. The role of ATOM donations is the following: thanks to ATOM we want to secure lifetime commitment of cyber•Congress in the development of Cosmos and Cyber ecosystems. ATOM donations to cyber•Congress will allow us to use staking rewards for continuous funding of the Cyber protocol without necessity to dump CYBs

Roadmap

We foresee the demand for the following features community could work on after launch:

- Parametrization
- KV
- IBC
- WASM VM for gas
- Onchain upgrades
- CUDA VM for gas
- Privacy by default

Applications of knowledge graph

A lot of cool applications can be built on top of proposed architecture:

Web3 browsers. It easy to imagine the emergence of a full-blown blockchain browser. Currently, there are several efforts for developing browsers around blockchains and distributed tech. Among them are Beaker, ~~Mist~~, Brave, and Metamask. All of them suffer from trying to embed web2 in web3. Our approach is a bit different. We consider web2 as the unsafe subset of web3. That is why we decide to develop a web3 browser that can showcase the cyber approach to answer questions better.

Programmable semantic cores. Currently, the most popular keywords in a gigantic semantic core of Google are keywords of apps such as youtube, facebook, github, etc. However, developers have very limited possibility to explain Google how to better structure results. The cyber approach brings this power back to developers. On any given user input string in any

application relevant answer can be computed either globally, in the context of an app, a user, a geo or in all of them combined.

Search actions. Proposed design enable native support for blockchain asset related activity. It is trivial to design applications which are (1) owned by creators, (2) appear right in search results and (3) allow a transact-able call to actions with (4) provable attribution of a conversion to search query. e-Commerce has never been so easy for everybody.

Offline search. IPFS make possible easy retrieval of documents from surroundings without a global internet connection. cyberd can itself can be distributed using IPFS. That creates a possibility for ubiquitous offline search.

Command tools. Command line tools can rely on relevant and structured answers from a search engine. That practically means that the following CLI tool is possible to implement

```
>  cyberd earn using 100 gb hdd

Enjoy the following predictions:
- apt install go-filecoin:     0.001   BTC per month per GB
- apt install siad:            0.0001  BTC per month per GB
- apt install storjd:          0.00008 BTC per month per GB

According to the best prediction, I made a decision try `mine go-filecoin`

Git clone ...
Building go-filecoin
Starting go-filecoin
Creating a wallet using @xhipster seed
You address is ....
Placing bids ...
Waiting for incoming storage requests ...
```

Search from CLI tools will inevitably create a highly competitive market of a dedicated semantic core for bots.

Autonomous robots. Blockchain technology enables the creation of devices which can earn, store, spend and invest digital assets by themselves.

> If a robot can earn, store, spend and invest she can do everything you can do

What is needed is a simple yet powerful state reality tool with the ability to find particular things. cyberd offers minimalistic but continuously self-improving data source that provides necessary tools for programming economically rational robots. According to top-10000 english words the most popular word in English is defined article `the` that means a pointer to

a particular thing. That fact can be explained as the following: particular things are the most important for us. So the nature of our current semantic computing is to find unique things. Hence the understanding of unique things become essential for robots too.

Language convergence. A programmer should not care about what language do the user use. We don't need to know about what language user is searching in. Entire UTF-8 spectrum is at work. A semantic core is open so competition for answering can become distributed across different domain-specific areas, including semantic cores of different languages. The unified approach creates an opportunity for cyber•Bahasa. Since the Internet, we observe a process of rapid language convergence. We use more truly global words across the entire planet independently of our nationality, language and race, Name the Internet. The dream of truly global language is hard to deploy because it is hard to agree on what means what. However, we have the tools to make that dream come true. It is not hard to predict that the shorter a word, the more its cyber•rank will be. Global publicly available list of symbols, words, and phrases sorted by cyber•rank with corresponding links provided by cyberd can be the foundation for the emergence of genuinely global language everybody can accept. Recent scientific advances in machine translation are breathtaking but meaningless for those who wish to apply them without Google scale trained model. Proposed cyber•rank offers precisely this.

This is sure not the exhaustive list of possible applications but very exciting, though.

Apps on top of knowledge graph

Our approach to the economics of consensus computer is that users buy an amount of RAM, CPU, and GPU as they want to execute programs. OpenCypher or GraphQL like language can be provided to explore semantics of the knowledge graph. The following list is simple programs we can envision that can be built on top of simple relevance machine with support of onchain WASM-like VM.

Self prediction. A consensus computer can continuously build a knowledge graph by itself predicting the existence of cyberlinks and applying these predictions to a state of itself. Hence a consensus computer can participate in the economic consensus of the cyber protocol.

Universal oracle. A consensus computer can store the most relevant data in the key-value store, where the key is cid and value is bytes of actual content. She is doing it by making a decision every round about which cid value she want to prune and which she wants to apply based on the utility measure of content addresses in the knowledge graph. To compute utility measure validators check availability and size of content for the top-ranked content address in the knowledge graph, then weight on the size of cids and its ranks. The emergent key-value store will be available to write for consensus computer only and not agents, but values can be used in programs.

Proof of location. It is possible to construct cyberlinks with proof-of-location based on some existing protocol such as Foam. So location-based search also can become provable if web3 agents will mine triangulations and attaching proof of location for every link chain.

Proof of web3 agent. Agents are a subset of content addresses with one fundamental property: consensus computer can prove the existence of private keys for content addresses for the subset of knowledge graph even if those addresses has never transacted in its own chain. Hence it is possible to compute much provable stuff on top of that knowledge. E.g., some inflation can be distributed to addresses that have never transacted in the cyber network but have the provable link.

Motivation for read requests. It would be great to create cybernomics not only for write requests to consensus computer but from read requests also. So read requests can be two order of magnitude cheaper, but guaranteed. Read requests to a search engine can be provided by the second tier of nodes which earn CYB tokens in state channels. We consider implementing state channels based on HTLC and proof verification which unlocks amount earned for already served requests.

Prediction markets on link relevance. We can move the idea further by the ranking of knowledge graph based on prediction market on links relevance. An app that allow betting on link relevance can become a unique source of truth for the direction of terms as well as motivate agents to submit more links.

Private cyberlinks. Privacy is foundational. While we are committed to privacy achieving implementation of private cyberlinks is unfeasible for our team up to Genesis. Hence it is up to the community to work on wasm programs that can be executed on top of the protocol. The problem is to compute cyberRank based on cyberlink submitted by a web3 agent without revealing neither previous request nor public keys of a web3 agent. Zero-knowledge proofs, in general, are very expensive. We believe that privacy of search should be must by design, but not sure that we know how to implement it. [Coda]() like recursive snarks and [mimblewimble]() constructions, in theory, can solve part of the privacy issue, but they are new, untested and anyway will be more expensive regarding computations than a transparent alternative.

Conclusion

We define and implement a protocol for provable communications of consensus computers on relevance. The protocol is based on a simple idea of content defined knowledge graphs which are generated by web3 agents using cyberlinks. Cyberlinks are processed by a consensus computer using a concept we call relevance machine. `cyber` consensus computer is based on `CIDv0` and uses `go-ipfs` and `cosmos-sdk` as a foundation. IPFS provide significant benefits regarding resources consumption. CIDv0 as primary objects are robust in its

simplicity. For every CIDv0 cyber•rank is computed by a consensus computer with no single point of failure. Cyber•rank is CYB weighted PageRank with economic protection from sybil attacks and selfish voting. Every round merkle root of the rank tree is published so every computer can prove to any computer a relevance value for a given CID. Sybil resistance is based on bandwidth limiting. Embedded ability to execute programs offer inspiring apps. Starting primary goal is indexing of peer-to-peer systems with self-authenticated data either stateless, such as IPFS, Swarm, DAT, Git, BitTorrent, or stateful such as Bitcoin, Ethereum and other blockchains and tangles. Proposed semantics of linking offers a robust mechanism for predicting meaningful relations between objects by a consensus computer itself. The source code of a relevance machine is open source. Every bit of data accumulated by a consensus computer is available for everybody if the one has resources to process it. The performance of proposed software implementation is sufficient for seamless user interactions. Scalability of proposed implementation is enough to index all self-authenticated data that exist today and serve it to millions of web3 agents. The blockchain is managed by a decentralized autonomous organization which functions under Tendermint consensus algorithm with standard governance module. Thought a system provide necessary utility to offer an alternative for conventional search engines it is not limited to this use case either. The system is extendable for numerous applications and, e.g. makes it possible to design economically rational self-owned robots that can autonomously understand objects around them.

References

- cyberd
- Scholarly context adrift
- Web3 stack
- Search engines information retrieval in practice
- Motivating game for adversarial example research
- An idea of decentralized search
- IPFS
- DAT
- cosmos-sdk
- CIDv0
- Thermodynamics of predictions
- DURA
- Nebulas
- Colony
- Truebit
- SpringRank
- PageRank
- RFC-6962
- IBC protocol

- Tendermint
- Comparison of web3 browsers
- Cyb
- SpringRank
- How to become validator in cyber protocol
- Top 10000 english words
- Multilingual neural machine translation
- Foam
- Coda
- Mimblewimble
- Tezos
- Software 2.0