Rings Network - A Distributed Network Based on Chord DHT P2P Lookup Protocol

0xEA[1], 0xCA[2]
[1]*CTO of Rings Network, 0xea@ringsnetwork.io*
[2]*Architect of Rings Network, 0xca@ringsnetwork.io*

ABSTRACT

Most of the current web applications are based on DNS(domain name system) to establish network communications for the modern Internet. However, hosting DNS and providing queries requires experienced engineers to maintain the service. Still, lots of them face DDoS and malicious poisoning problems. DNSSEC helps a bit by verifying. Nevertheless, end-users still have risks to connect and trust, especially when ISP's DNS is down or polluted. SSL(secure sockets layer) is one of the most used secure standards for Internet connections. But SSL or other encrypting method is not enabled by default. Low-level data transactions have different kinds of secure protocols such as TLS and these protocols are not widely used by all software.

Nor are data centers and server providers trusted all the time. They are subject to the jurisdiction of the government where the server is located. In the event that Tornado Cash was banned by the US government, server providers and CDN providers all took immediate action to block users from accessing their front-end pages. Even other decentralized protocols decided to ban wallet addresses had interacted with Tornado Cash. And infura RPC node service did the same bad thing. Only the smart contract on Ethereum remains for it's decentrailzation feature. Tornado Cash is not the only and special one. Any website or protocol may face this situation in the near future. Making everying part of services decentralized, getting rid of the dependence on the data center is the way to save.

This paper presents a new secure network built with dDNS(distributed DNS) and a new data transfer infrastructure based on crypto algorithms and Chord DHT technology. *Rings Network* aims to build a standalone p2p network to provide basic DNS service and secure network traffic for traditional web2 and web3 applications on blockchains. The nodes of Rings Network can run on servers and in browsers(as WebAssembly) and nodes in browsers and servers are functionally equivalent.

I. INTRODUCTION

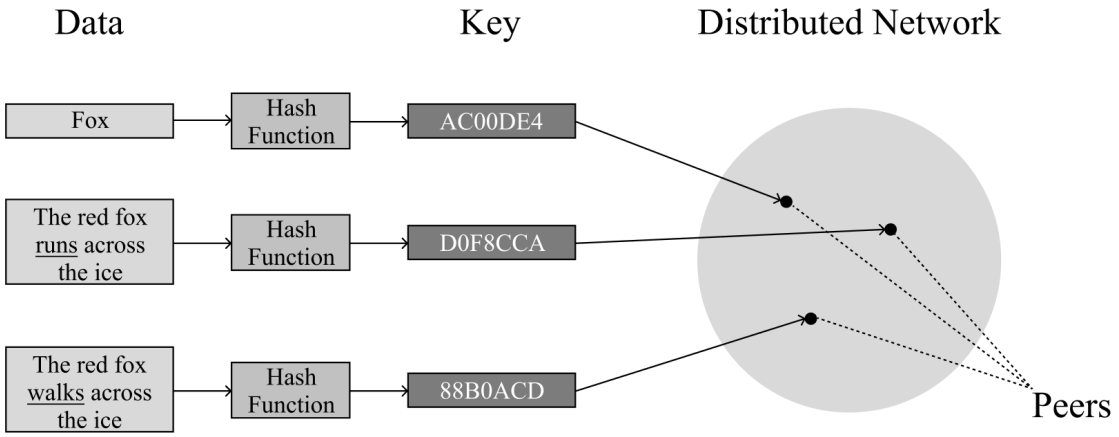
The Stanford Research Institute (now SRI International) maintained a HOSTS.TXT text file starting from around 1970 to store the mapping records of domains and IPs for connecting remote computers for ARPANET. Ten years later this text file became quite larger. Modifying requests is a quite large number. Providing this service on a single node made it slow. In 1983, RFC 882 and RFC 883 were created based on the first version of the Domain Name System standard. Later with RFC 1034 and RFC 1035, DNS specifications were extensible with additional protocols. DNSSEC was created in the 1990s. It provides a strong mechanism for clients to verify the records in DNS are authentic. DNSSEC prohibits forged IPs and similar attacks. Researchers began to focus on performance and maintenance problems. Name servers are difficult to host for administrators; ordinary users rely on their ISPs to server domain name data. ISPs face huge challenges in maintaining DNS services.

DNS hijacking(poisoning) is widely happening all over the world. Hackers modify DNS records for malicious purposes such as phishing, for self-serving purposes by ISPs such as serving advertisements, and for political or legal purposes by governments or stakeholders. Banks(such as Brazilian Banks) and brokers' DNS records are the most frequently hijacked ones. ISPs such as AT&T, Verizon, and Sprint used DNS hijacking to display ads or collect statistics. Telstra(an ISP of Australia) has used DNS poisoning technology to block The Pirate Bay.

According to the degree of centralization, P2P networks can be divided into three main categories: Unstructured networks, structured networks, hybrid models network. The hybrid network is a combination of the unstructured and the structured network. We'll not talk too much abount the byrid model here.

Nodes in unstructured network act as both client and server. There is no central server or central router in an unstructured network. BitTorrent, eDonkey and Bitcoin are all unstructured networks.

In structured networks the overlay is organized into a specific topology, and the protocol ensures that any node can efficiently search the network for a file/resource, even if the resource is extremely rare. The most common type of structured P2P networks implement a distributed hash table (DHT), in which a variant of consistent hashing is used to assign ownership of each file to a particular peer. This enables peers to search for resources on the network using a hash table: that is, (key, value) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Not all structured network has a central server or router. Rings Network is a structured network based on Chord DHT, but without any centralized controller.



DHT Prototype

This paper contributes the basic algorithm to proof of the correctness of domain name records, and a mechanism for running a structured peer-to-peer network to store name data as well as to provide queries and modification interfaces for users. Benefits from the distributed infrastructure, DDoS attacks, and other centralized disadvantages would be solved to a certain extent in most cases. A full copy of DHT data will be stored, retrieved, and editable on multiple mainstream blockchains like Ethereum, Arbitrum, etc. to make sure anyone can build or run a full-data node of *Rings Network* from downtime or for scaling.

II. Related Work

Chord can provide the same service by converting each hostname to a key. Chord-based DNS does not require special servers, while normal DNS relies on a special set of root servers. DNS requires manual management of routing information (NS records) that allow clients to navigate a hierarchy of nameservers; Chord automatically maintains the accuracy of analog routing information. DNS works correctly only if the hostname structure reflects administrative boundaries; Chord does not enforce a naming structure. While DNS is specialized for the task of finding named hosts or services, Chord can also be used to find data objects that are not tied to a specific machine.

Pastry is an overlay network and routing network for the implementation of a distributed hash table (DHT) similar to Chord. The key-value pairs are stored in a redundant peer-to-peer network of connected Internet hosts. The protocol is bootstrapped by supplying it with the IP address of a peer already in the network and from then on via the routing table which is dynamically built and repaired. It is claimed that because of its redundant and decentralized nature there is no single point of failure and any single node can leave the network at any time without warning and with little or no chance of

data loss. The protocol is also capable of using a routing metric supplied by an outside program, such as ping or traceroute, to determine the best routes to store in its routing table.

Freenet is a peer-to-peer platform for censorship-resistant, anonymous communication. It uses a decentralized distributed data store to keep and deliver information and has a suite of free software for publishing and communicating on the Web without fear of censorship. Both Freenet and some of its associated tools were originally designed by Ian Clarke, who defined Freenet's goal as providing freedom of speech on the Internet with strong anonymity protection.

The Globe system has a wide-area location service to map object identifiers to the locations of moving objects. Globe arranges the Internet as a hierarchy of geographical, topological, or administrative domains, effectively constructing a static worldwide search tree, much like DNS. Information about an object is stored in a particular leaf domain, and pointer caches provide search shortcuts. The Globe system handles a high load on the logical root by partitioning objects among multiple physical root servers using hash-like techniques. Chord performs this hash function well enough that it can achieve scalability without also involving any hierarchy, though Chord does not exploit network locality as well as Globe.

PNRP(Peer Name Resolution Protocol) is a peer-to-peer protocol designed by Microsoft. PNRP enables dynamic name publication and resolution and requires IPv6. PNRP was first mentioned during a presentation at a P2P conference in November 2001. Other hosts can then resolve the peer name, retrieve the corresponding addresses and other information, and establish peer-to-peer connections. Internally, PNRP uses an architecture similar to distributed hash table systems such as Chord or Pastry. The cache maintenance algorithm ensures that each node maintains adequate knowledge of the "cloud". It is designed to ensure that the time to resolve a request varies as the logarithm of the size of the cloud. PNRP now works on the released Windows systems but no evidence shows it could be used on other platforms.

DoX is a peer-to-peer DNS networking method for detecting and correcting inaccurate DNS records caused by cache poisoning attacks. DoX uses Chord to store and synchronize hostnames and keys. Only detecting or correcting is the solution after the problem occurs. We believe that using crypto algorithms to verify and transfer data among nodes is better and safer. Two-way hostname data synchronization between DNS network and blockchain datastores is necessary for long-term data storing and full decentralized implementation.

Onion Routing(the core principle of Tor) is implemented by encryption in the application layer of the communication protocol stack. As Tor cannot encrypt the traffic between an exit node and the target server, any exit node is in a position to capture traffic passing through it that does not use end-to-end encryption such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Shreds of evidence show that hackers can exploit usernames and passwords for email accounts by operating and monitoring Tor exit nodes.

III. System Design

I. Chord Lookup Protocol

The Chord protocol specifies how to find the locations of keys, how new nodes join the system, and how to recover from the failure (or planned departure) of existing nodes.

Nodes and keys are assigned an m-bit identifier using consistent hashing. The SHA-1 algorithm is the base hashing function for consistent hashing. Consistent hashing is integral to the robustness and performance of Chord because both keys and nodes (in fact, their IP addresses) are uniformly distributed in the same identifier space with a negligible possibility of collision. Thus, it also allows nodes to join and leave the network without disruption. In the protocol, the term node is used to refer to both a node itself and its identifier (ID) without ambiguity. So is the term key.

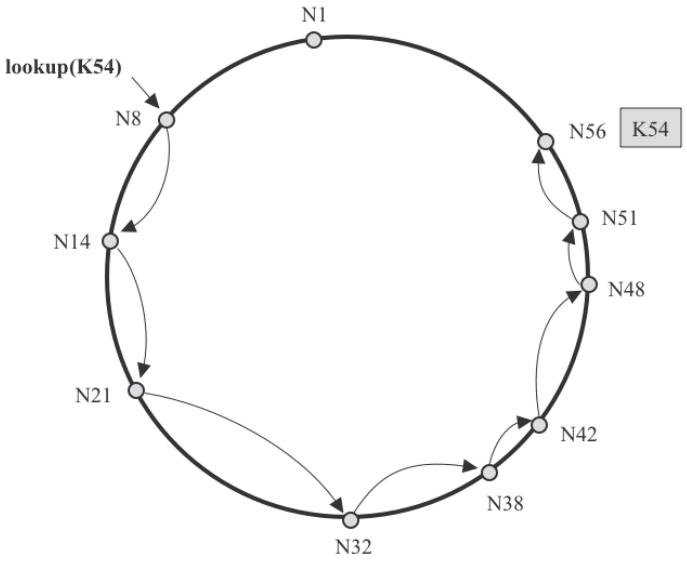
Using the Chord lookup protocol, nodes and keys are arranged in an identifier circle that has at most 2^m nodes, ranging from 0 to $2^m - 1$. (m should be large enough to avoid the collision.) Some of these nodes will map to machines or keys while others (most) will be empty.

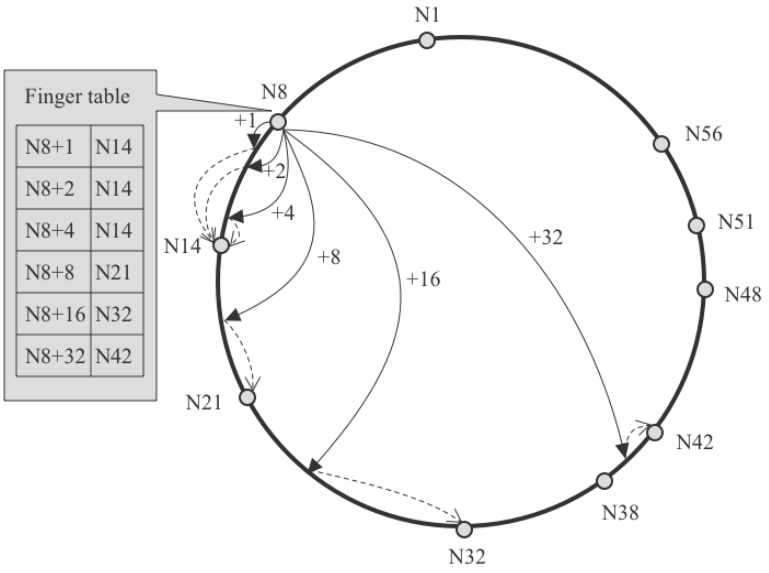
Each node has a successor and a predecessor. The successor to a node is the next node in the identifier circle in a clockwise direction. The predecessor is counter-clockwise. If there is a node for each possible ID, the successor of node 0 is node 1, and the predecessor of node 0 is node $2^m - 1$; however, normally there are "holes" in the sequence. For example, the successor of node 153 may be node 167 (and nodes from 154 to 166 do not exist); in this case, the predecessor of node 167 will be node 153.

The concept of successor can be used for keys as well. The successor node of a key k is the first node whose ID equals to k or follows k in the identifier circle, denoted by $successor(k)$. Every key is assigned to (stored at) its successor node, so looking up a key k is to query $successor(k)$.

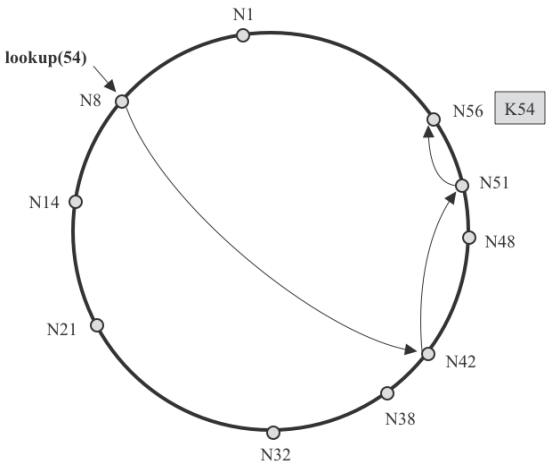
Since the successor (or predecessor) of a node may disappear from the network (because of failure or departure), each node records a whole segment of the circle adjacent to it, i.e., the r nodes preceding it and the r nodes following it. This list results in a high probability that a node is able to correctly locate its successor or predecessor, even if the network in question suffers from a high failure rate.

The core usage of the Chord protocol is to query a key from a client (generally a node as well), i.e. to find $successor(k)$. The basic approach is to pass the query to a node's successor if it cannot find the key locally. This will lead to an $O(N)$ query time where N is the number of machines in the ring.





To avoid the linear search above, Chord implements a faster search method by requiring each node to keep a finger table containing up to m entries, recalling that m is the number of bits in the hash key. The i^{th} entry of node n will contain $successor(n + 2^{i-1} \bmod 2^m)$. The first entry of the finger table is actually the node's immediate successor (and therefore an extra successor field is not needed). Every time a node wants to look up a key k, it will pass the query to the closest successor or predecessor (depending on the finger table) of k in its finger table (the "largest" one on the circle whose ID is smaller than k), until a node finds out the key is stored in its immediate successor.



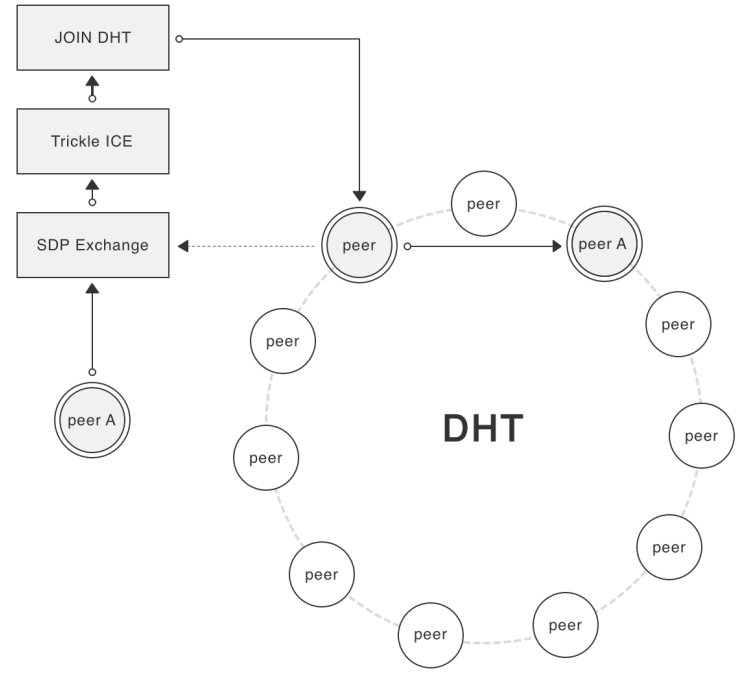
The pseudocode of the find successor operation, extended to use finger tables:

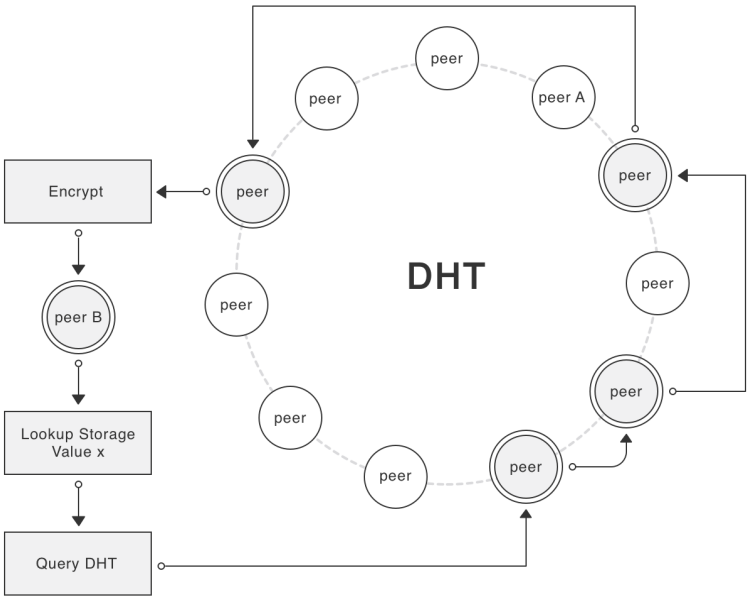
```
// ask node n to find the successor of id
n.find_successor(id)
  if (id ∈ (n, successor])
    return successor;
  else
    n' = closest preceding node(id);
    return n'.find_successor(id);

// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
  for i = m downto 1
    if (finger[i] ∈ (n, id))
      return finger[i];
  return n;
```

If id falls between n and its successor, find_successor is finished and node n returns its successor. Otherwise, n searches its finger table for the node n' whose ID most immediately precedes id, and then invokes find successor at n'. The reason behind this choice of n' is that the closer n' is to id, the more it will know about the identifier circle in the region of id.

The concise schematic diagram of node-joining and keys lookup:





II. Sub-Rings Protocol

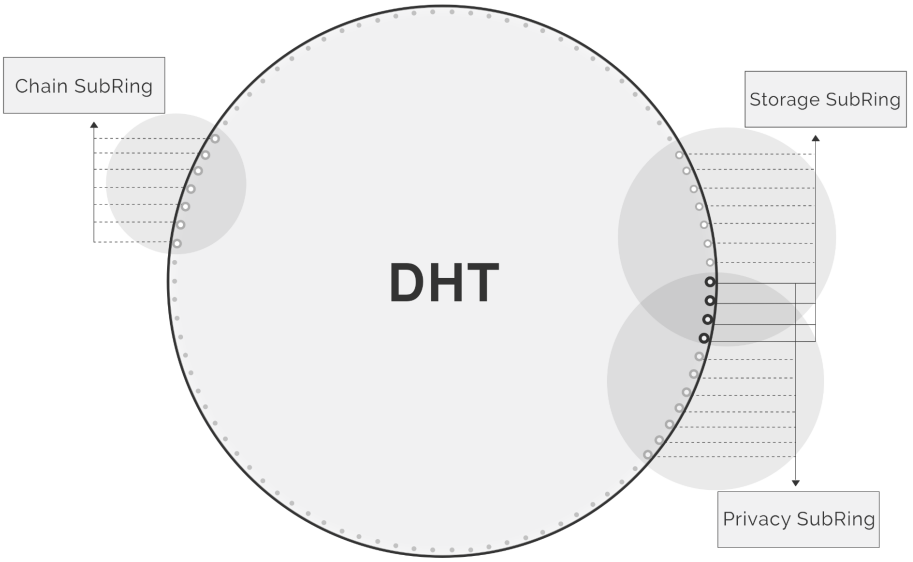
The global DHT in *Rings Network* stores system shared data. Meanwhile, many applications need to meet customization needs, such as storing files, encrypted privacy storage, private data, and specially structured data. These data are not suitable to be stored in all nodes, which can cause node data

redundancy, reduce synchronization speed and read speed, and increase node stress. For this reason, we designed the subring system.

Upper layer developers can design their own subring and DHT schemes, and user nodes can add one or more special-purpose subrings to meet their needs based on joining the global ring. This subring mechanism is dynamically scalable and can be withdrawn at any time. With subring, when you develop applications on rings network, you can do low-coupling R&D design, and you don't need to introduce multiple distributed systems to meet the requirements.

Officially provided subrings:

1. Storage subring for storing static data or files
2. Consensus subring
3. Ledger subring for blockchains
4. Privacy subring for hiding identities



For example, if a developer wants to develop a decentralized video chat system, they can first join the Rings Network to link the network. Users use the blockchain public key as identity information, and developers can access a subring designed for file storage to store user avatars and other files, a database subring to store chat logs, a face recognition and video processing subring to add filters, and a push stream processing subring for peer-to-peer video data transfer. In this example, the developer incorporates different blockchain and web 2.0 technologies and integrates storage systems for different data types, but without introducing additional technical solutions. Traditional blockchain applications may have to implement storage through file storage systems such as IPFS, and rely on centralized servers for streaming data and video processing, etc. A system coupled with too many different solutions is extremely weak, and if one of them fails, the whole service will be unavailable. subring design solves this problem perfectly.

III. Relay Protocol

Relay mode is also called proxy mode. When node A send messages to node B, it couldn't or designed not to connect node B directly, messages will be sent via detour A-C-B or more complex path A-C-D-E-B. Relayed path may not be a single route, all the middle nodes may try to find a possible reach to the final one by limited broadcasting.

When nested NAT of the beginning or destinating node limit direct connections, relay protocol ensure messages would be dispatched to end users if one way is possible.

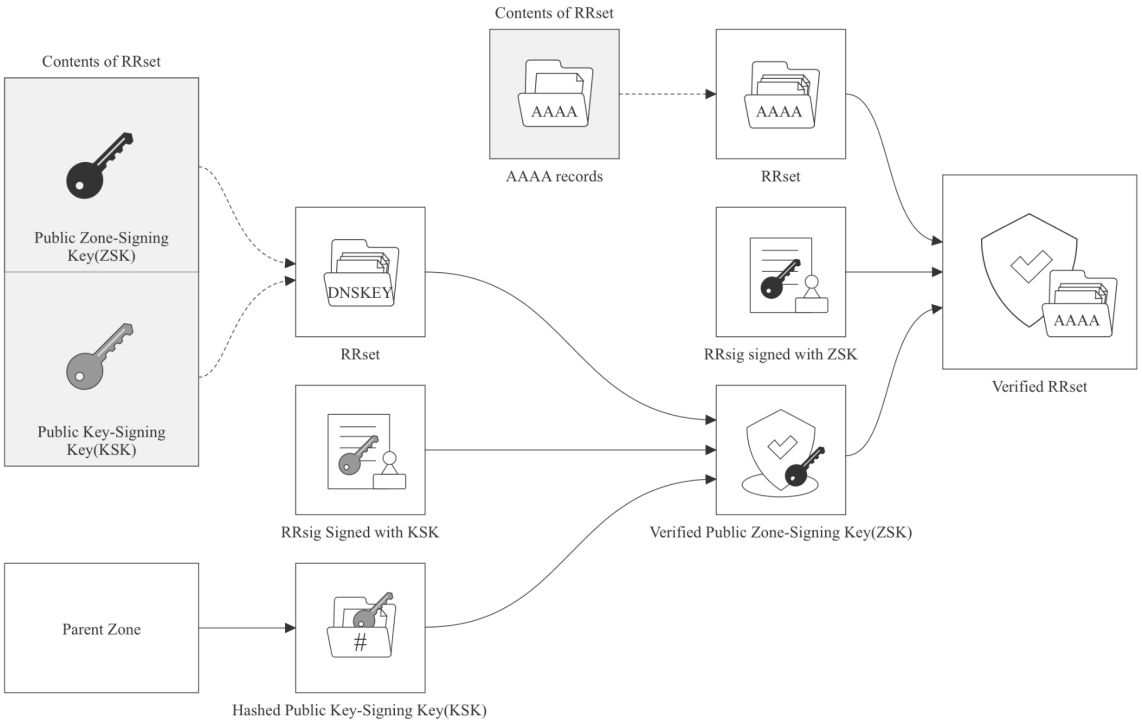
Connecting via P2P without VPN may cause the leak of ethernet IP address as well as LAN IP address or LAN mDNS hostnames. In this case relay protocol can provide more than connecting each others. Privacy sensitive decentralized apps need to protect user's IP and hostname from physical locating by enabling relay mode.

Rings Network can also let user use different private keys for the network layer and the product layer. The pair of keys would be associated together and recognized with zero-knowledge proof. The network layer isolation could help to protect user's main private key. With relay protocol and zero-knowledge-proof based mutable double keys, users's IP and public key address could not be easily binded together even there're bad guys hosting multiple malicious nodes, because you'll never know if the next hop is the final one unless you're the receiver.

IV. dDNS Protocol

Our system handles lookups at the granularity of resource record sets (RRSets), as in conventional DNS. An RRSet is a list of all the records matching a given domain name and resource type.

DNSSEC uses public-key cryptography to sign re- source record sets. When we retrieve an RRSet from an arbitrary server, we need to verify the signature (included as a signature (SIG) record). To find the public key that should have signed the RRSet, we need to execute another DNS lookup, this time for a public key (KEY) RRSet. This RRSet is in turn signed with the public key for the enclosing domain.



dDNS stores and retrieves resource record sets using DHash, a Chord-based distributed hash table. DHash has two properties useful for this discussion: load balance and robustness.

DHash uses consistent hashing to allocate keys to nodes evenly. Further, as each block is retrieved, it is cached along the lookup path. If a particular record is looked up n times in succession starting at random locations in a Chord ring of m nodes, then with high probability each server transfers a given record only log m times total before every server has the record cached.

DHash is also robust: as servers come and go, DHash automatically moves data so that it is always stored on a fixed number of replicas (typically six). Because the replicas that store a block are chosen in a pseudo-random fashion, a very large number of servers must fail simultaneously before data loss occurs.

To create or update a DDNS RRSet, the owner prepares the RRSet, signs it, and inserts it into DHash. The key for the RRSet is the SHA1 hash of the domain name and the RRSet query type.

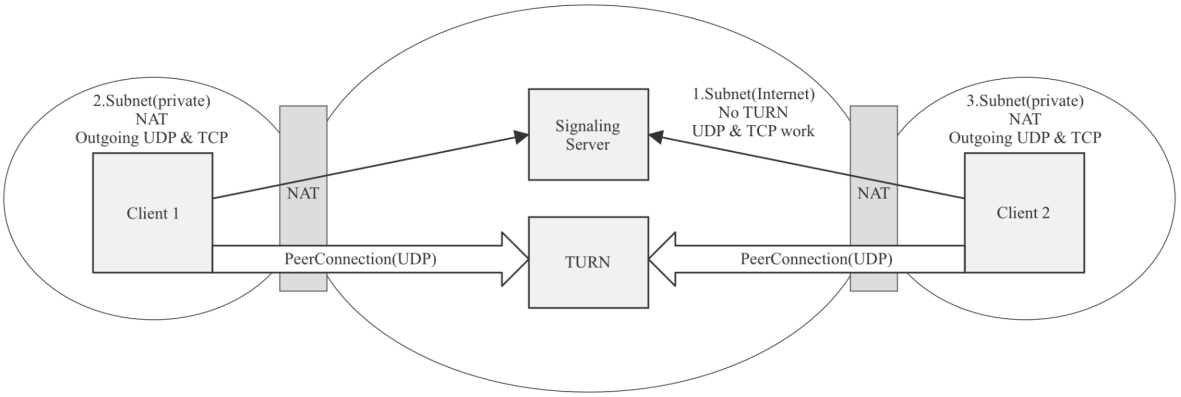
Naively verifying a DNS RRSet for a name with n path elements requires n KEY lookups. We address this problem by allowing the owner to present additional relevant KEYs in the RRSet. To avoid inflating the responses, we can omit KEY RRSets for popular names.

To ease the transition from conventional DNS to our system, a simple loopback server listening on 127.1 could accept conventional DNS queries, perform the appropriate Chord lookup, and then send a conventional response. Then systems could simply be configured to point at 127.1 as their name server.

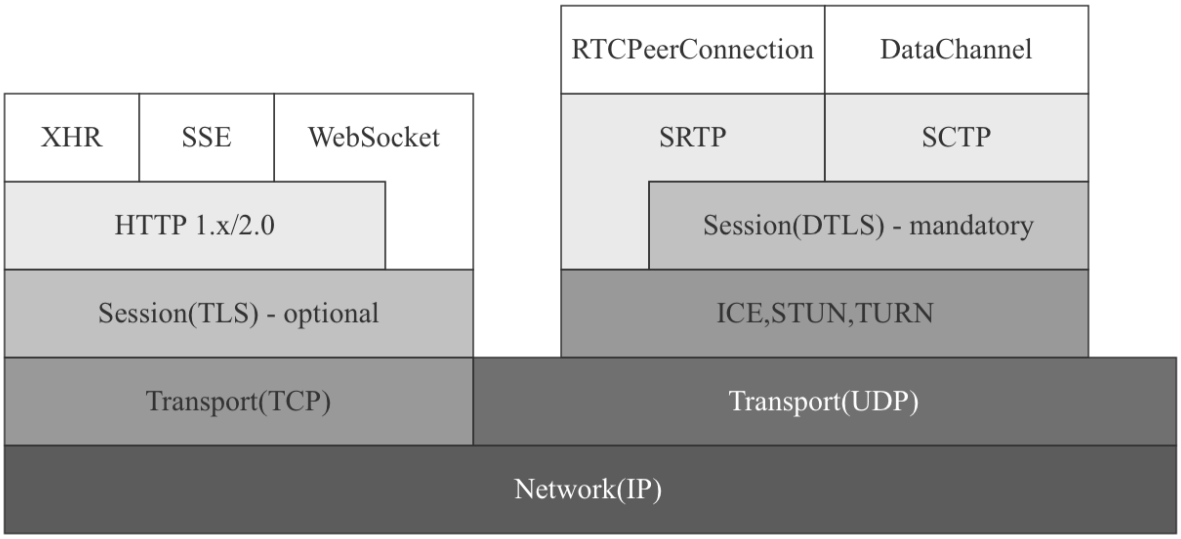
V. Peer-to-peer Communication Protocol

Rings Network builds communication between nodes based on WebRTC(Web Real-Time Communication) protocol. WebRTC is a free and open-source project providing web browsers and mobile applications with real-time communication (RTC) via application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.

The advantage of WebRTC compared to the existing IETF STUN (RFC 5389), TURN (RFC 5766) and ICRings NetworkE (RFC 5245) protocols are that it does not require a public host and port mapping for each participating localhost, and it works with more restrictive firewall policies. WebRTC implements ICE which combines STUN and TURN probes to automatically find the best connection between two peers who want to communicate. In corporate networks, simple hole punching, and NAT traversal techniques typically do not work, e.g. because of symmetric NATs. Dynamic allocation of ports on an external 3rd party relay service is also typically blocked on restricted hosts.

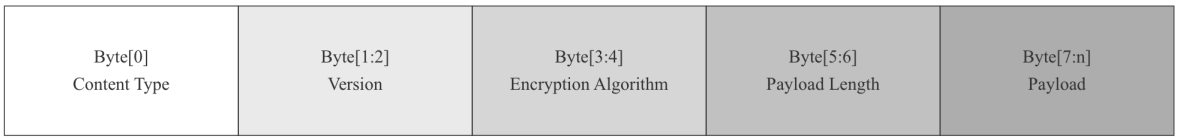


An *RTCPeerConnection*instance allows an application to establish peer-to-peer communications with another *RTCPeerConnection* instance in another browser, or to another endpoint implementing the required protocols. Communications are coordinated by the exchange of control messages (called a signaling protocol) over a signaling channel which is provided by unspecified means.



Traffic between nodes is protected by SDES and DTLS-SRTP to avoid MiTM attacks. Data will be packed with a timestamp and nonce to avoid replay attacks. TCP flow or UDP message is transmitted between the nodes. Before the node sends the data to another node, the user access address will be added in front of the user data, and the address is also encrypted transmission. The raw message will be encrypted by AES-128-CTR and other common-trusted crypto algorithms.

A possible packet data abstract format:

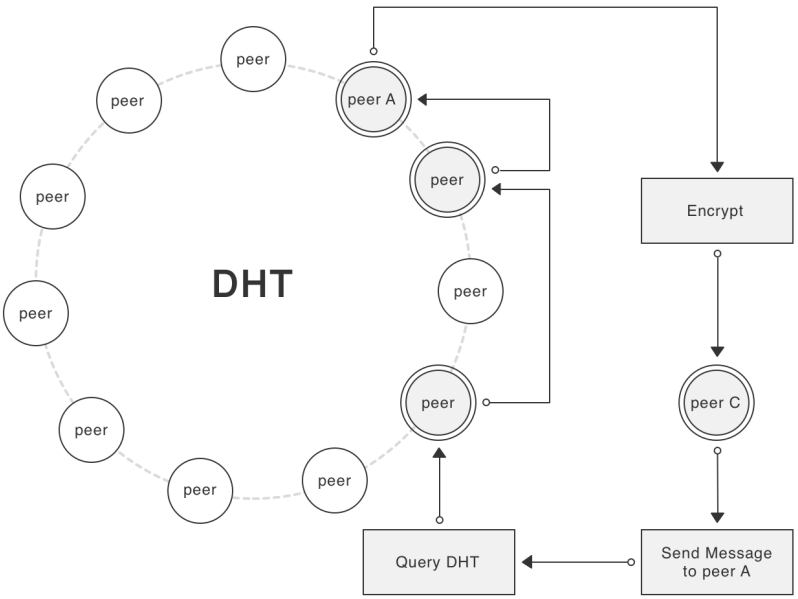


When a new node successfully has a WebRTC connection to an existing *Rings Network* node, it calls a $join(known\_node)$ function to clarify itself joined. Every node can run stabilizing function to learn about the newly joined nodes.

To ensure correct lookups and the network's robustness, the stabilization protocol is running in the background to update successor pointers and the hash table every 30s. With stabilization enhanced, each node will have 160 concurrent peer connections at the same time, which connects the nodes with a hyper inner net. Sudden disconnecting between a few nodes will not affect the connectivity because other connections still work. In our tests, even if 50% of connections halt unexpectedly, the network stability and data consistency remain around 95%.

For example, suppose node n joins the system, and its ID lies between nodes n_p and n_s. In its call to $join()$, n acquires ns as its successor. Node n_s, when notified by n, acquires n as its predecessor. When n_p next runs $stabilize()$, it asks n_s for its predecessor (which is now n); n_p then acquires n as its successor. Finally, n_p notifies n, and n acquires n_p as its predecessor. At this point, all predecessor and successor pointers are correct. At each step in the process, n_s is reachable from n_p using successor pointers; this means that lookups concurrent with the join are not disrupted.

Node reaching and communication logic:



VI. Blockchain Integration

Rings Network chain part will be deployed on the Ethereum main network, test networks, Layer 2 networks like Arbitrum, Avalanche, and other active blockchains such as BSC, Polygon, Aptos, Solana, Fantom, etc. Users on these blockchains can retrieve or modify name data. A blockchain datastore is a permanent place to store name hashes and values to initialize or restart the whole peer-to-peer network. The network's data will remain and go back online even in the worst case: the entire *Rings Network* is down and all data destroyed. We still have a copy of the data on blockchains.

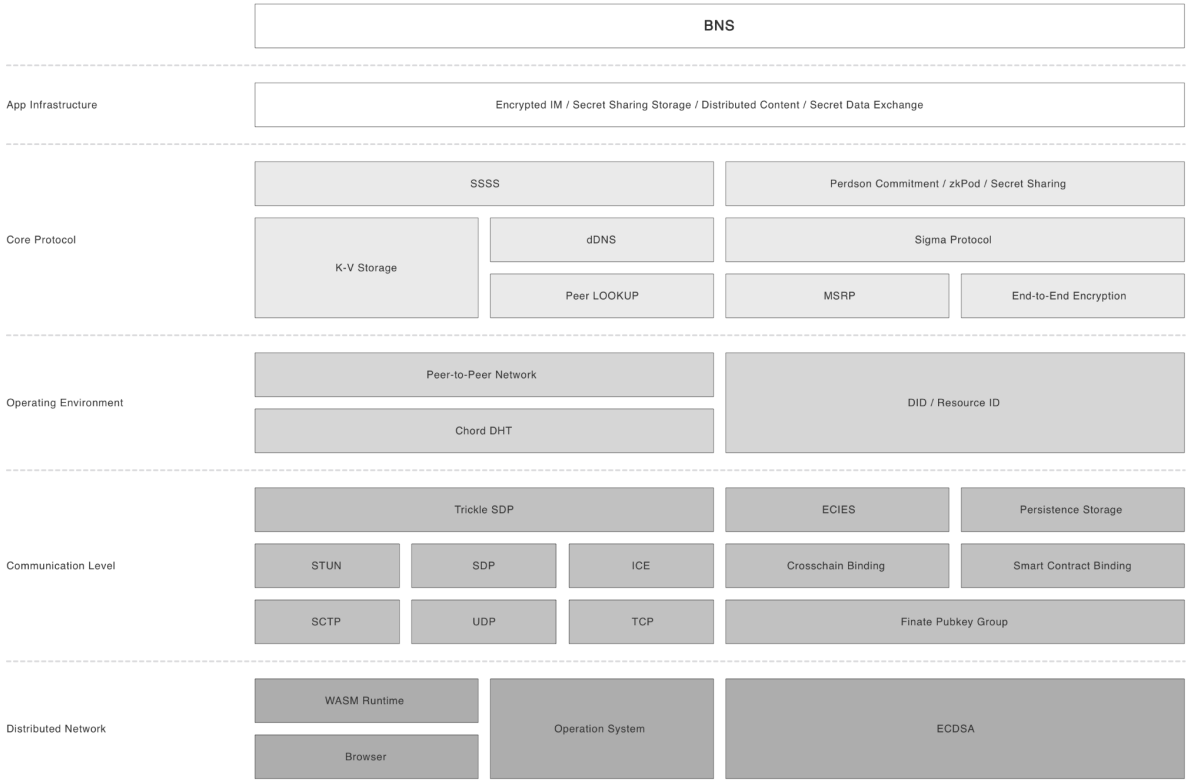
More than data backups, contract on-chain can also interact with *Rings Network* contracts to support their web3 applications, or build a bridge from web2 to web3, and per se.

Rings Network on the blockchain is similar to ENS(Ethereum Name Service) at name service storing and querying parts. We could do more than just name-address mappings. Users who own a private key will take the whole censorship of their domain and data. You could store addresses, hashes, web2 sub-domains, IP addresses, or anything you want with the *Rings Network* contract. Transferring tokens or coins is just a basic use of *Rings Network*, users could easily interact with web3 applications if a domain or sub-domain stands for a contract address, at meantime other domains stand for regular web2 IP addresses or centralized service endpoint.

Suppose we are running a game based on *Rings Network*, web2 low-latency controls, web2 team audio, web3 character buying, and web3 token gaining happen together in one network without any

painful on-chain to off-chain switching behaviors. A user's public key is his DID(decentralized identifier) to everything he/she gets involved in.

VII. System Infrastructure



IV. Conclusion

Rings network aims to build a new generation of Internet infrastructure, integrating web2.0 and web3.0 (especially blockchain technology) to provide a new generation of decentralized Internet solutions. Humanity created the first generation of the Internet out of the need for easy information sharing, and the broader need gave rise to infrastructures such as large server rooms, submarine fiber optic cables, and satellite communications. The human nature of creation and communication allowed web 2.0 to develop, but the basic functionality no longer met the demand. People need new financial systems, privacy systems, we desire independence, privacy, ownership, we want knowledge to be preserved and passed on, and blockchain technology gives us hope.

Web3.0 shows us a new chapter of the Internet, where people expect a fairer, more democratic, and freer Internet in the face of political, geopolitical, nationalistic, and economic sanctions that block communication. rings network hopes to build a new generation of decentralized peer-to-peer networks based on cryptographic systems to make all this possible. We are just moving a small step forward on the shoulders of giants.

We do not oppose web2, nor blindly oppose centralization, nor do we oppose government regulation to protect the interests of the people. We respect all anarchists and those who oppose anarchism. However, we insist that technology is innocent. Those who create advanced technology are innocent. Only those who abuse technology to infringe the rights and interests of others are guilty. Technology itself should not bear any crime.

The original intentions of the blockchain and the new generation of decentralized networks starting from Bitcoin are all good intentions, but we firmly believe that the principle of "Don't trust, verify it." can give us maximum security. We can’t trust anyone (including regulators, of course) to be kind, and to do the most optimistic things with the most pessimistic and malicious speculations will give us a brighter future.

The technology that can protect the privacy of the bad guys can protect the privacy of the good guys. Technology that can easily invade the privacy of bad guys can easily invade anyone's privacy. Cyber privacy is already a basic human right now, and we agree to bring the bad guys to justice, but not at the expense of everyone's rights.

We hope to provide web3 users with a sense of cybersecurity.

