Distributed Decentralized Domain Name Service

Brendan Benshoof Andrew Rosen
Department of Computer Science, Georgia State University
34 Peachtree St NW
Atlanta, Georgia 30303
bbenshoof@cs.gsu.edu rosen@cs.gsu.edu

***Abstract*—We present D^{3}NS, a system to replace the current top level DNS system and certificate authorities which offers increased scalability, security and robustness. D^{3}NS is based on a distributed hash table, utilizes a domain name ownership system based on bitcoin and addresses previous criticism that a DHT would not suffice as a DNS replacement. D^{3}NS provides solutions to current DNS vulnerabilities such as DDOS attacks, DNS spoofing and Censorship. D^{3}NS eliminates the need for certificate authorities by providing a decentralized authenticated record of domain name ownership. Unlike previous DNS replacement proposals, D^{3}NS is reverse compatible with DNS and allows for incremental implementation within the current system.**

I. Introduction

The Domain Name System, commonly referred to as DNS [10] [11], is a fundamental component of the Internet. DNS maps memorable names to the numerical IP addresses used by computers to communicate over IP.

Two recent events in the United states have brought DNS to the forefront of networking and security research. First is recent legislation proposed in the US House of Representatives and US Senate. The Stop Online Piracy Act (SOPA) [17] and PROTECT IP Act (PIPA) [14] were both introduced in 2011. There were numerous aspects to both bills, but essential to both was that DNS (non-authoritative?) servers located in the US would be required filter DNS records on demand, essentially fracturing the DNS system. There would be no guarantee that DNS could serve the same information to two different users.

More recent are the leaks of classified information elucidating the extent of the NSA's spying capabilities. These leaks have raised questions about the security of SSL and TLS, as well as the level of trust that users place in certificate authorities.

There have been many explorations and attempts[6][13][15] to propose a DNS system based on a distributed hash table[18]. We extend on those papers by implementing a DHT which minimizes latency distance rather than hop distance and implementing a shared record of ownership based on recent developments in cryptocurrency.[12][1]

These types of threats to DNS, along with security concerns, were not considered when designing the protocol, but DNS is too widely used and too integrated with the Internet as a whole to be replaced. Extensions such as DNSSEC [5] add authentication and data integrity, but do not alter the fundamental architecture of the DNS network.

This paper proposes the Distributed Decentralized Domain Name Service, or D^{3}NS. D^{3}NS is a completely decentralized Domain Name Service operating over a Distributed Hash Table (DHT). D^{3}NS does not replace the DNS protocol, but rather adds robustness to the architecture as a whole. Internally, D^{3}NS signs all DNS records using public/private keys.

We show that D^{3}NS satisfies the objections to a decentralized DNS system posed by Cox et al [6], specifically: dramatically reducing latency compared to other DHT systems, retaining the extensibility of the original DNS system, and changing the intended scope of use to address incentive issues. We show D^{3}NS allows for new authentication methods and a means of decentralized proof of ownership beyond that of Cox et al's work.

The rest of the paper is consists of the following sections. Section gives an overview of DNS and identifies prior research in the area of distributed DNS. Section covers the modified blockchain used for record authentication in D^{3}NS. Section presents VHash, a DHT that we designed for D^{3}NS. Section defines the various components of D^{3}NS, while Section details our implementation of D^{3}NS. We discuss our conclusions and future work in Section.

II. Background

This paper is intended to address concerns raised by Cox et al[6] and propose a viable decentralized DNS replacement based on a DHT. Our proposed improvements on the DNS alternative presented by Cox et al are full reverse compatibility with the current hierarchical

DNS system and a shared authenticated public record that allows for DNSSEC style authentication.

A. DNS Overview

DNS queries proceed recursively the DNS hierarchy, beginning with a query to a root server, which then yields a record for a server for the requested top level domain. This server then directs the request to another DNS server responsible for the domain under that, which yields an answer or another DNS server, which is queried in the same manner.

One of the key concepts of the DNS architecture is that no matter which servers end up being queried, a user can expect to receive a record consistent with what the rest of the DNS system will serve for that particular request.

Multiple recent legislative motions reflect DNS's weakness to be influenced by local government intervention [17] [9] [7].

These bills would have required that servers maintained in the US filter specified domain names, preventing users from obtaining the correct IP address for the domain name in question. Multiple governments have been noted to preform systematic attacks on DNS queries [19]. This filtering is incomparable with the DNS Security Extensions (DNSSEC) [7] and DNS's intended usage.

The mandated dns filtering would drive users to unregulated DNS servers, which would create more attack vectors where users could have their traffic redirected to a malicious website by attackers or the governments they are attempting to circumvent.

B. Related Work

Cox et al devised a distributed DNS using CHORD[18] as the storage medium for DNS records. They examined the possibility of extending DNSSEC with new options of storing keys in the DHT. They encountered the then unsolved problem of proof of ownership for domain names and found no means of enhancing security. They noted that the overlay topology of a DHT like CHORD did not take into account latency optimization and thus DNS was not a viable application of a DHT due to significantly greater latency. They noted several possible improvements of using a DHT, namely an increased robustness, auto-balancing structure, resistance to DDOS attacks and packet injection based DNS spoofing.

They considered the optimization and security problemssolvable but they considers two issues to render the system unviable. They intended for this system to replace all DNS servers and traffic, which removed extensibility and customizability of the original DNS protocol. Because they considered replacing the entire DNS system rather than a meaningful subset the presented a question incentive. What incentivized companies to support the system where their servers had to share the load of other companies traffic?

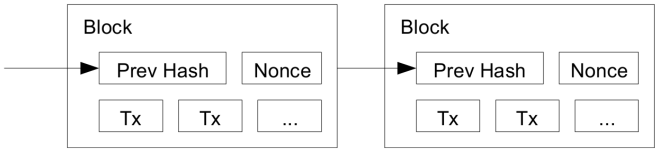

Fig. 1: A section of the blockchain as defined by Bitcoin [12].

We address the each issues they raise. We utilize a side channel method of confirming domain named ownership via Blockchain to enable DNSSEC style security at all layers of the network. We pose a DHT structure wich allows for minimum latency optimization. We pose only to replace authoritative Top Level Domain servers currently managed by registrars, where most records are simple a forward to an authoritative DNS server managed by the domain owner rather than replacing all levels of DNS. This limiting of scope allows us to continue to take advantage of DNS extensions as well as places responsibility of managing the network with those who have incentive for its continued function.

III. Blockchain

We use a tool called a *blockchain* for maintaining and authenticating our DNS records. Blockchains have their roots in the cryptocurrency Bitcoin [12], where it is used to authenticate financial transactions and verify account balances.

A. Blockchains in Bitcoin

Bitcoin is a decentralized electronic currency. Here we are particularly concerned with Bitcoin's blockchain. Bitcoin's blockchain consists of a shared authenticatable transaction record [1] [12] .

Bitcoin's blockchain is essentially a shared ledger. The blockchain (Figure 1 is a record of every single transaction made using the Bitcoin. Each transaction refers to previous transactions to indicate the funds handled by a given transaction are in fact own by the user initiating the transactions. The record is validated by traversing the tree of transactions and marking referenced transactions as used. A valid blockchain has all non-leaf node transaction marked as used only once.

Transactions are grouped together and verified in a *block*. Each block in the chain is a series of transactions published during the time it takes to generate that block. The process of authenticating these transactions and generating a new block is called mining (Algorithm 1, analogous to the concept of gold mining, as the incentive for successfully mining a block a meaningful sum of new bitcoins.

A block is mined by generating a nonce field on the block chain such that the hash of the entire block is less than a global difficulty value. This difficulty sets the rate at which new blocks are mined and it adjusted in reference to the number of miners[1]. When a block is mined, it is transmitted to the network and each transaction in it is validated by each peer. The network will then work on mining the next block.

Algorithm 1 Blockchain mining

Given previous Block B_{-1}
Given New Transaction Set T
Given Difficulty D
Given Reward destination R
New Block B_0 = $HASH(B_{-1})|T|R|Timestamp$
Block Attempt b = $B_0|Nonce$
while $HASH(b) > D$ **do**
    Block Attempt b = $B_0|Nonce$
end while
Propogate b as next block

B. Using the Blockchain to validate DNS records

We utilize the transaction record of bitcoin to record ownership of domain names. Rather than rewarding miners with currency, the reward and incentive for mining a new block is a record that allots the miner the right to claim a domain name. Algorithm 2 shows the process for validating transactions on using the blockchain.

The transactions in each block indicate miners claiming a new domain or the transfer of domain ownership. Claims of new domains are validated by a reference to an unclaimed mining reward owned by the claiming user. Transfers are validated by a pointer to an unused previous transfer record or claim record indicating ownership by the transferring party. This way every domain name in the system can be associated with an owner's private key. New domains can be claimed and old domains can be transfered between owners.

[1]Bitcoin adjusts the difficulty rate every 2016 blocks such that the network will then mine a block every ten minutes on average [4]

Algorithm 2 Blockchain Transaction Validation

A new transaction t consists of: Award domain D to user U with proof reference P and signature S
The transaction set T is the set of all transactions considered valid
if P is not marked used **then**
    if owner indicated in P matches signature S **then**
        if D matches domain referenced in P **then**
            Mark P as used
            Consider t valid
        else
            if P is a mining reward **then**
                P is an unclaimed mining reward
                if D is not yet claimed **then**
                    Mark P as used
                    Mark D as claimed
                    Consider t valid
                end if
            end if
        end if
    end if
end if

C. Using a Blockchain to Replace Certificate Authority

The shared record of the blockchain allows any participant in the mining network to act as a trusted third party to clients. This way trust is not centralized in single points of failure. Internally members of the DHT are also members of the blockchain network (as it is convenient to use the DHT overlay as the Blockchain network overlay network) and thus all pushed records to the DHT and retrieved records can be confirmed as legitimate before transmission to the end user. This limits the viability of replay or injection based attacks.

D. Unaddressed Security Issues

The Blockchain does not solve all security issues relevant to DNS authentication and security. Exit nodes could lie or have packets inject to clients until the protocol from DNS server to client is improved. The DHT structure opens up unexplored disruption attacks on the overlay topology.

IV. VHash

VHash was created to allow for spacial representations to be mapped to hash locations, a feature lacking in many current distributed hash tables. In particular, we aimed to construct a mechanism for creating a more efficient global scale DHT built on a minimal latency overlay. Rather than focus on minimizing the amount

of hops required to travel from point to point we wish to minimize the time required for a message to reach its recipient. VHash actually has a worse worst case hop distance ($O(\sqrt[d]{n})$) then other comparable Distributed Hash Tables ($O(lg(n))$) however VHash can route messages as quickly as possible rather than a grand tour an overlay network may describe in the real world.

The naive method of doing so is to assign coordinates to servers based on embedding location of nodes. More complex approaches would approximate a minimum latency space based on inter-node latency. VHash can be considered a generalized extension of VoroNet [3]. Algorithm 3 describes the process for performing a minimum latency embedding using VHash.

Algorithm 3 Vhash Minimum Latency Embedding

1: d is the dimensions of the hash space
2: seed the space with $d+1$ nodes at random locations
3: A node n wishes to join the network
4: n pings a random subset of peers to find latencies L
5: Normalize L onto (0.0,1.0) to yield L_N
6: Choose position p such that

$$\sum_{i \in peers} (L_N[i] - dist(p,i))^2$$

is minimized
7: Re-evaluate location periodically

A. Toroidal Distance Equation

Given two vector locations $\vec{a}$ and $\vec{b}$ on a d dimensional unit cube

$$distance = \sqrt[|d|]{\sum_{i \in d} (\min(|\vec{a}_i - \vec{b}_i|, 1.0 - |\vec{a}_i - \vec{b}_i|))^2}$$

B. Mechanism

VHash maps nodes to an overlay arbitrary d dimension toroidal unit space. This is essentially a hypercube with wrapping edges. The toroidal property makes visualization but allows for a space without a sparse edge as all nodes can translate the space such as to consider themselves the center.

VHash nodes are responsible for the address space defined by their Voronoi region. This region is defined by a list of peer nodes maintained by the node. A minimum list of peers is maintained such that the node's Voronoi region is well defined. The links connecting the node to its peers correspond to the links of a Delaunay Triangulation.

C. Relation to Voronoi Diagrams and Delaunay Triangulation

VHash does not strictly solve Voronoi diagrams [2], as the toroidal nature of the space preclude the traditional means of solving Voronoi regions. However, VHash's peer management approximates a graph with similar properties. An online algorithm maintains the set of peers defining the node's voronoi region. The set of peers required to define a node's voronoi region describe a solution to the dual Delaunay Triangulation.

D. Messages

Maintenance and joining are handled by a simple periodic mechanism. A notification message consisting of a node's information and active peers is the only maintenance message. All messages have a destination hash location which is used to route them to the proper server. This destination can be the hash location of a particular node or the location of a desired record or service. The message is recived by the node responsible for the location. Services running on the DHT define their own message contents.

E. Message Routing

Messages are routed over the overlay network using a simple algorithm. A node maintains a minimal list of peers to define it's own voronoi region. From the perspective of the node, the entire voronoi region is defined only by its peers.

When routing a message to an arbitrary location, a node calculates who's voronoi region the message's destination is in amongst the node and its peers. If the destination falls within its own region then it is responsible and handles the message accordingly. The node otherwise forwards the message to the closest peer to the destination location. This process describes a precomputed and cached A* [8] routing algorithm and is shown in Algorithm 4.

F. Joining and Maintenance

Joining the network is a straightforward process. A new node first learns the location of at least one member of the network to join. The joining node then choses a location in the hash space either at random or based on a problem formulation (for example, based on geographic location or latency information).

After choosing a location, the joining node sends a "join" message to its own location via the known node. The message is forwarded to the current owner of that location who can be considered a "parent" node. The

Algorithm 4 Vhash Routing

P_0 is this node's set of peers
N is this node
m is a message addressed for L
$Forwards$ is the set $P_0 \cup N$
find C: member of $Forwards$ which has the shortest distance to L
if C is N **then**
    N is the responsible party.
    Handle m
else
    Forward m to C for handling or further routing
end if

parent node immediately replies with a maintenance message containing its full peer list. This message makes its way to the joining node, who then uses this to begin defining the space he is responsible for.

The joining nodes final peers are a subset of the parent and the parent's peers. The parent adds the new node to it's peer list and removes all his peers occluded by the new node. Then regular maintenance propagates the new node's information and repairs the overlay topology. This process is given by Algorithm 5.

Algorithm 5 Vhash Join

new node N wishes to join and has location L
N knows node x to be a member of the network
N sends a request to join, addressed to L via x
node $Parent$ is responsible for location L and receives the join message
$Parent$ sends to N its own location and list of peers
$Parent$ integrates N into its peer set
N builds its peer list from N and its peers
regular maintenance updates other peers

Each node in the network performs maintenance periodically by sending nodes a maintenance message. The maintenance message consists of the node's information and the information on that node's peer list. When a maintenance message is received, the receiving node considers the listed nodes as candidates for its own peer list and removes any occluded nodes from their own peer list (Algorithm 6).

When messages sent to a peer fail, it is assumed the peer has left the network. The leaving peer is removed from the peer list and candidates from the set of 2-hop peers provided by other peers move in to replace it. Maintenance is described by Algorithms 7 and 8. Figures 2, 3, 4, and 5.

Algorithm 6 VHash Greedy Peer Selection

$Candiates$ is the set of candidate peers
$Peers$ is the set of this node's peers
$Canidates$ is sorted by each node's closeness to this node
The closest member of $Canidates$ is popped and added to $Peers$
for all n in $Canidates$ **do**
    c is the midpoint between this node and n
    if Any node in $Peers$ is closer to c than this node **then**
        reject n as a peer
    else
        Add n to $Peers$
    end if
end for

Algorithm 7 VHash Maintenance Cycle

P_0 is this node's set of peers
T is the maintenance period
while Node is running **do**
    for all node n in P_0 **do**
        Send a Maintenance Message containing P_0 to n
    end for
    Wait T seconds
end while

Algorithm 8 VHash Handle Maintenance Message

P_0 is this node's set of peers
Receive a Maintenance Message from peer n containing its set of peers:P_n
for all Peers p in P_n **do**
    Consider p as a member of P_0
    if p should join P_0 **then**
        Add p to P_0
        for all Other peers i in p **do**
            if i is occluded by p **then**
                remove i from P_0
            end if
        end for
    end if
end for

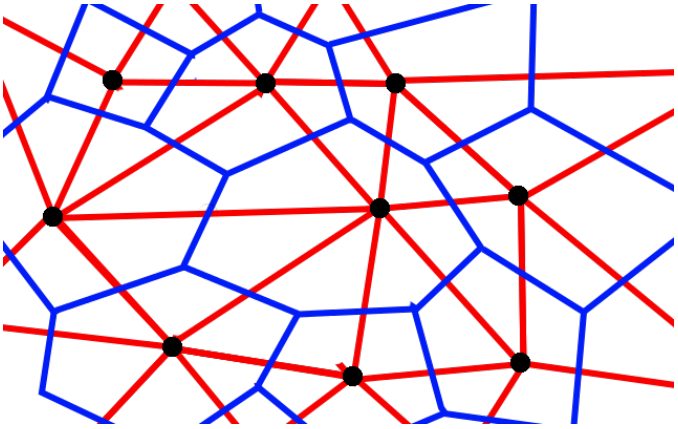

Fig. 2: The starting network topology. The red edges connecting the nodes correspond to the Delaunay Triangulation edges.

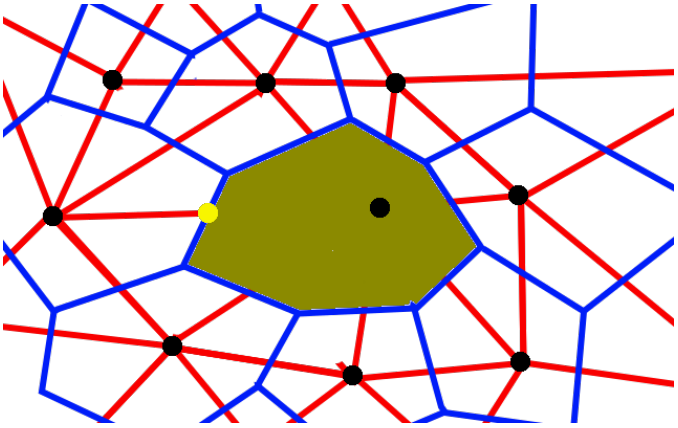

Fig. 3: Here, a new node is joining the networks and has established his position falls in the the yellow shaded voronoi region.

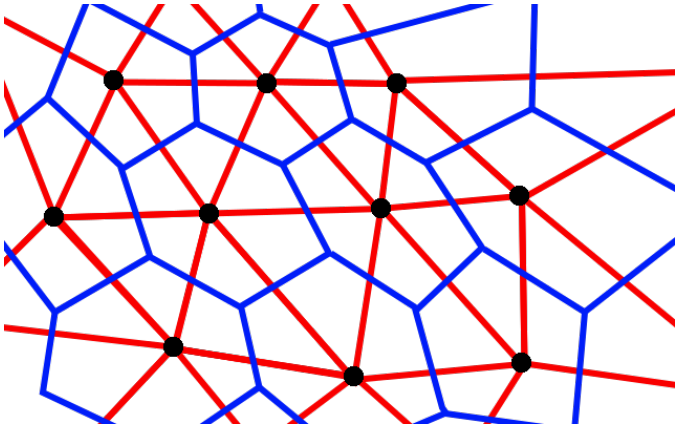

Fig. 4: The network topology after the new node has finished joining.

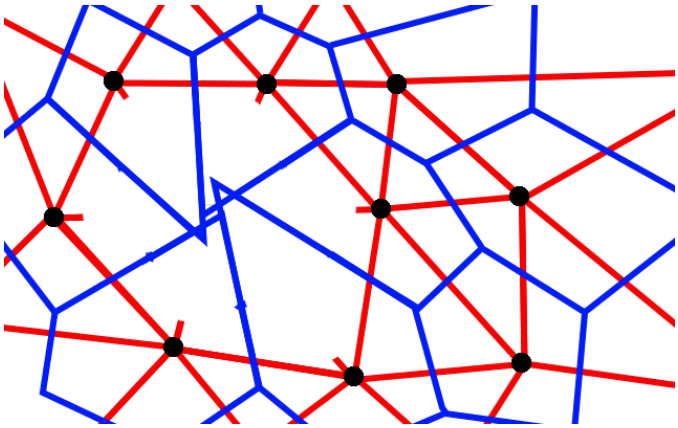

Fig. 5: The topology immediately after the new node leaves the network. After maintenance takes place, the topology repairs itself back to the configuration shown in Figure 2

There is no protocol for a ”polite” exit from the network. The DHT protocol assumes nodes will fail and the difference between an intended failure and unintended failure is unnecessary. The only issue this causes is that software should be designed to fail totally when issues arise rather then attempt to fulfill only part of the protocol.

G. Data Storage and Backups

The primary goal of a DHT is to provide a distributed storage medium. We extend this idea to distribute work and information among nodes using the same paradigm. Resources in the network, be it raw data or assigned tasks, are assigned hash locations. The node responsible for a given hash location is responsible for themaintenance of that resource. When a node fails, its peers take responsibility of its space. Thus it is important to provide peers with frequent backups of a node’s assigned resources. That way, when a node fails, its peers can immediately assume its responsibilities.

When a resource is to be stored on the network, it is assigned a hash location. The hash locations assigned could be random, a hash of an identifier, or have specific meaning for an embedding problem. The node responsible for that resource’s hash location stores the resource.

A resource is accessed by contacting the node responsible for the resource. However, the requester generally has no idea which node is responsible for any particular resource. The data request messsage is addressed to the location corresponding to the resource, rather than the node responsible for that location. The message is forwarded over the overlay network, each hop bringing the node closer until it reaches the responsible node, who sends the resource or an error if the resource does not exist.

Some options are immediately apparent for dealing

with wasted storage space. A system that is primarily read driven can record the time of the last read or a frequency of reads such that resources that are not read often enough after a certain period of time are deleted. If a system is write driven, allow the resource to be assigned a time to live, which can be updated as needed.

H. Backups

At a frequency set by the node manager or at a frequency set by any adaptive process to optimize backup frequency, a node send a message containing backups of the resources for which it is newly responsible to each of its peers. To minimize bandwidth and time wasted by backups, the node should only send the records changed since last backup

I. Unaddressed Issues

A series of issues relevant to VHash are not addressed in this paper due to limiting of scope, space, time, and a lack of actual solutions to the problems: The exact method of caching to optimize lookup time under real world usage. The overlay network being mapped onto latency results in nodes whose failure due to natural disaster to be comorbid with it's neighbors resulting to data loss. Comorbidity could be counteracted by more complex backup schemes.

V. D^{3}NS

D^{3}NS has logically discrete components which provide DNS efficient record storage, Domain name ownership management and verification, DNS backwards compatitibility, all of which may be logically replaced or have individual optimizations. D^{3}NS uses a DHT to store DNS records in a distributed fashion, A blockchain and Namecoin[1] analog to manage domain name ownership.

D^{3}NS utilizes public and private key encryption for signing and verifying records.

A. Distributed Hash Table

Our implementation is not specific to any particular Distributed Hash Table. We examined using Chord [18] with DNS, similar to DDNS [6]. However, Chord's unidirectional ring overlay topology does not take actual network topology into account and using it for a global scale system is not viable because messages will be routed very inefficiently. A DHT which allows the routing overlay to be optimized to the network topology and conditions in real time is required.

As a result we chose to develop a prototype DHT to meet this requirement to act as backend to our DNS system called Vhash. Vhash is built on the idea of an "overlay space" which is a k-dimensional unit cube which wraps around the edges in a toroidal fashion. Each record in the DHT is assigned a location in that space. Each node is assigned a location and is responsible for records to which it is the closest node. The variable dimensionality is allowed so that problems can be embedded into the space with relative ease and records can be assigned locations which have meaning concerning the problem in which they are a part. This way, records that are close to each other in the problem formulation are close to each other in the DHT and are likely hosted on the same node. This offers speedup for many distributed algorithms which require traversal of data.

B. DNS frontend

Because this system is intended to be reverse compatible with the existing DNS protocol, we serve the data provided by the DHT after it has been authenticated by the block chain to other DNS servers or clients. DNS nodes incorporated into the D^{3}NS system will not request data from other DNS servers and will only exchange data via the DHT

VI. Implementation

Im not sure what to put here, this indicates it needs to be re-organized

A. Establishment of a New domain

Under the current DNS system, a new domain name is purchased from a company registered with the Internet Corporation for Assigned Names and Numbers (ICANN). That company adds the domain name and a record provided by the owner to the TLD servers. The owner or management company then maintains a name server to answer DNS requests for the purchased domain. In D^{3}NS new domain names are instead awarded as part of the blockchain mining process or purchased from a previous owner, then transferred to the new owner. These assignments and transfers are both recorded in the blockchain.

A prospective domain owner can create their own mining software and mine a domain name or purchase a domain name voucher from a miner and exchange it for a domain name. In the blockchain, domain name owners are referred to by their public key which is used to authenticate records and transfer domains. Loss of the private key of an account will result in the loss of ability to update DNS records and ability to transfer the domain.

B. Updating records for a domain

A domain name record in the current DNS system is used to configure a record on your own Name Server or to configure the record held by the TLD server to contains an address record. Using P2PDDNS all records must be signed using their owner's private key.

A properly configured P2PDNS server should not accept a DNS records which has not been signed by its blockchain confirmed owner or accept a record with an older version number. To push a new DNS record for a domain the owner must create the record set for the domain and then sign and submit it to a node on the DHT. The DHT will forward the record to the responsible party and store it after confirming its validation. The new record will begin to be broadcast to clients after old records begin to expire and caches seek new records.

C. Looking up a DNS record

In the current DNS a record is looked up using a UDP system that queries a tree of requests. clients send queries to a local DNS server which acts as a resolver and cache. If a portion of the domain name is unknown the resolver sends a request to the responsible server and looks up recursively from there.

This system is largely unchanged under D^{3}NS from the point of view of the resolver and client. Ideally the resolver or client has chosen a nearby member of the D^{3}NS network as its root domain server (it can also maintain a large list of backup servers should its current one fail). The resolver requests a domain's record from the D^{3}NS node. The node then forwards the request to the responsible party. If any node along the route has a valid cache of the required record, then that server responds and routes the message back to the entry node.All nodes along the route cache the response to aid future queries.

D. Caching

DNS needs to aggressively cache lookups. Previous investigations into optimizing caching on a DHT saw good results [16], however with the specific application of DNS in mind, the time-to-live field on DNS records defer the caching optimization problem onto users who we may or may not trust.

Integrated File Replication and Consistency Maintenance (IRM) [16] views the process of caching and keeping the cache up to date as components of larger problem. Nodes in the DHT keep track of how often records are requested and cache those records once a defined rate is passed. Nodes then request an update for the cache based on how often the record is requested and how often that request is expected to be changed by the owner.

In the current implementation of DNS, (who does it) caching involves a time to live field defined by the domain owner. This means that there's not really any sort of cache optimization done by the server; rather the server that thew records have a sensible time-to-live value. IRM [16] can be used to approximate the correct time-to-live value.

While it may be tempting to utilize the built in time to live field for caching on the DHT, the possible cyclical nature of cache passing may result in difficulty in propagating new records.

VII. Practical Demonstration

D^{3}NS was successfully implemented to act as a DNS server. A user would query the a computer we set up as a DNS gateway which was a member of the DHT and mining network. If the queried domain had a record stored in the DHT and an owner established in the blockchain the server would reply with the stored DNS records. Otherwise the server would reply with a DNS failure. We ran the DHT and mining network on a cluster of GSU computers and artificially low difficulty so that a live demo of mining would be viable.

VIII. Conclusion and Future Work

Using all of these components together allows us to create a system with the following features:

- Robustness - The DHT and Blockchain are both robust to failures and attacks
- Extensibility - The DNS reverse compatibility allows any DNS extension to be utilized, if dynamic resolution is required a name server record can be stored in the DHT to point to a user's specialized DNS servers.
- Decentralization - Both the DHT and Blockchain can operate without the support of any controlling organization, this offers security against corruption and abuse. This paper leaves open a series of relevant problems: Integration with DNSSEC, optimization of Caching and optimization of latency embedding. The technique of allying a DHT value store for real-time data and a Blockchain for ownership verification may prove a viable technique for decentralizing over web services and enabling further shared storage and computation mediums.

References

[1] Namecoin. https://github.com/vinced/namecoin. Accessed: 11/29/2013.

[2] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[3] Olivier Beaumont, A-M Kermarrec, Loris Marchal, and Etienne Rivière. Voronet: A scalable object network based on voronoi tessellations. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[4] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on*, pages 1–10, 2013.

[5] David Blacka and Samuel Weiler. Clarifications and implementation notes for dns security (dnssec). 2013.

[6] Russ Cox, Athicha Muthitacharoen, and Robert T Morris. Serving dns using a peer-to-peer lookup service. In *Peer-to-Peer Systems*, pages 155–165. Springer, 2002.

[7] Steve Crocker, David Dagon, Dan Kaminsky, DKH Danny McPherson, and Paul Vixie. Security and other technical concerns raised by the dns filtering requirements in the protect ip bill. *White Paper*, 2011.

[8] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[9] Mark Lemley, David Levine, and David Post. Don't break the internet. *Stanford Law Review Online*, 64:34, 2011.

[10] Paul Mockapetris. Rfc 1034: Domain names: concepts and facilities (november 1987). *Status: Standard*, 2003.

[11] Paul Mockapetris. Rfc 1035: Domain names: implementation and specification (november 1987). *http://www.ietf.org/rfc/rfc1035.txt*, 2004.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1:2012, 2008.

[13] Vasileios Pappas, Daniel Massey, Andreas Terzis, and Lixia Zhang. A comparative study of the dns design with dht-based alternatives. In *INFOCOM*, volume 6, pages 1–13, 2006.

[14] IP PROTECT. Preventing real online threats to economic creativity and theft of intellectual property act of 2011. s. 968. In *112th Congress, 1st Session, May*, volume 26, 2011.

[15] Venugopalan Ramasubramanian and Emin Gün Sirer. The design and implementation of a next generation name service for the internet. *ACM SIGCOMM Computer Communication Review*, 34(4):331–342, 2004.

[16] Haiying Shen. Irm: integrated file replication and consistency maintenance in p2p systems. *Parallel and Distributed Systems, IEEE Transactions on*, 21(1):100–113, 2010.

[17] Lamar Smith. Stop online piracy act. *US Government.[Links]*, 2011.

[18] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 149–160. ACM, 2001.

[19] Matthäus Wander, Christopher Boelmann, Lorenz Schwittmann, and Torben Weis. Measurement of globally visible dns injection.