P^{3}DNS

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

***Abstract*—We created a software package to replace the top level of the Domain Name system. It is reverse compatible with traditional DNS and requires no modifications on the end of the user**

I. Introduction

The Domain Name System, commonly referred to as DNS [3] [4], is a fundamental component of the Internet. DNS is really cool and important. Without it, we'd be force to punch in numbers instead of names to find a website.

There are some issues with DNS, which is pretty much the same now as it was when it was proposed. Now we've inherited those issues

The most pressing issues are: An ill advised law that doesn't understand how the internet works. The big trusted certificate issuers are native to the US and don't want get on the bad side of DHS/NSA/FBI/ETC

We have a system that can eliminate or at least lessen the severity of either issue.

¡intro fluff crap¿ This paper is intended to address concerns raised by Cox et al[2] 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.

II. Background or somesuch

A. DNS Overview

B. Related Work

What were the concerns of the cox paper [2]? Higher latency because chord (plus none of the chord advantages). Wasn't extensible. Served static records. Really nothing more than a distributed text file Not incentivised. No reason not be a defector in the system.

Advantages: DDNS is more resistant to DDOS attacks. Load balancing

III. Components

Our solution required many components.

A. Distributed Hash Table

Our implementation is not specific to any particular Distributed Hash Table. We examined using Chord [7] with DNS, similar to[2]. 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. Blockchain

Our DNS records are kept via a block chain

What is a blockchain (in general I mean) [5] [1]. Bitcoin's blockchain is essentially a shared ledger. The blockchain is a record of every single transaction made using the bitcoin system. This can get quite large.

How does our blockchain differ? Why do we a different blockchain. Ours is smaller because we don't want to keep all those records. That's expensive!

We still mine for blocks. Difficulty still to be determined.

C. 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 DDDNS system will not request data from other DNS servers and will only exchange data via the DHT

IV. Usage

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

A. Establishment of a New domain

With the current DNS system a new domain name is purchased from a company registered with ICANN and that company adds the domain name to the TLD servers with a record provided by the owner. The owner or management company then maintains a name server to provide responses to DNS requests for the purchased domain. In P2PDDNS new domain names must be awarded as part of the blockchain mining process or purchased from a previous owner and transferred in the blockchain. A prospective domain owner can establish 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 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. In P2PDNS this system is largely unchanged from the resolver and clients point of view. Ideally the resolver or client has chosen a nearby member of the P2PDNS 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 P2PDNS 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 caching the response

D. Caching

DHT needs to aggressively cache lookups. [6]

E. The Big Picture

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 users' 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.

References

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

[2] 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.

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

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

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

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

[7] 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.