An Efficient Cloud Based Approach for Decentralized DNS System

Sindhoor Tilak 1PE13CS148
Shashish Jha 1PE13CS139
Arisha Siddiqui 1PE13CS032
Venugopal 1PE14CS431

PESIT-Bangalore South Campus Guided by Dr. Annapurna D
Batch 35

Problem Statement

The existing legacy DNS (Domain Name Service) systems on the internet, have issues ranging from slowness, does not support updates, vulnerable to DoS (Denial of Service).

- Knocking down the DNS can take down the whole Internet.
- Recently, massive DDoS attack on Dyn, which hosts DNS servers took down the Internet for entire East Coast of US.

Problem Statement

We propose a solution which is cloud based and decentralized. This advantages of cloud based approach with services like(e.g AWS) include

- **Resource Pooling and Elasticity**
- **On-Demand & Self Services**
- **QoS (Quality of Service)**

The Cloud Service could be private entities or large TLD (Top Level Domain) companies.

Literature Survey

- **Francesca Musiani**,A Decentralized Domain Name System? User-Controlled Infrastructure as Alternative Internet Governance,MA:The MIT Press.
- **Harry Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, Arvind Narayanan**,An empirical study of Namecoin and lessons for decentralized namespace design, Princeton University.
- **Aaron Wright,Primavera De Filippi**,Decentralized blockchain technology and the rise of lex cryptographia, intGovt Forum.

Literature Survey

- **Christopher Allen, Arthur Brock, Vitalik Buterin**,Decentralized Public Key Infrastructure,Web Of Trust.
- **Venugopalan Ramasubramanian,Emin Gün Sirer**,The Design and Implementation of a Next Generation Name Service for the Internet,Cornell University.

Proposed Approach

Since we want a system which can coexist with the existing infrastructure, this system provides the following operations:

- **Registrations**
- **Key updates** (of the public key associated with the domain)
- **Revocations**

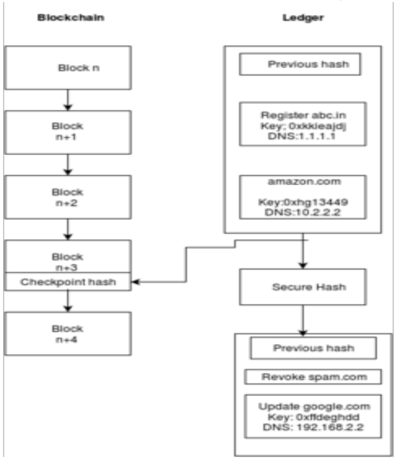
All these must be provided/controlled by the existing organization responsible for any given TLD .It places no artificial restrictions and the list above can easily be extended.

A Public Ledger

In this system, every TLD has a public ledger, maintained by TLD. With every update (any of the above actions) an entry is appended to this ledger. The ledger is broken up into blocks. A block contains all updates in the last 30 minutes.

Proposed Approach (cont'd.)

Proposed DNS Blockchain with Ledger.



Blockchain

- Rather than stuffing these into a blockchain directly, the latest additions to the ledger are hashed, together with the hash of the last update block. Thus creating a chain of update blocks.
- This hash is added to a public blockchain (e.g the Bitcoin blockchain). Information for locating the hash is served by TLD together with the ledger (this information could also be included in the following block).
- TLD must insert a new checkpoint in the blockchain every interval of time and there is one ledger block for every checkpoint.

Authentication

- Public keys are added to the DNS records stored in the public ledger. These keys are not added to the legacy DNS entries, but may also be validated against the existing PKI(Public Key Infrastructure).
- When Server Admin generates a new key for alice.com, she requests that TLD updates the ledger to reflect this. In addition she contacts a CA (Certificate Authority) to have the key signed and thus maintains backwards compatibility.
- The user keeps a copy of the blockchain and the public ledger (downloaded from TLD). When visiting a site user queries the local copy of the public ledger (verified against the blockchain) and finds the corresponding public key.

Trusted Entities

The Trusted Entities play a major role in this approach.

- The user needs to follow the blockchain. If he goes offline, he needs to download the entire blockchain and TLD ledger before visiting any site.
- The user needs to store a large amount of data. The ledger is potentially large and he needs to store one for every TLD he wishes to use (potentially hundreds).
- This can be solved by offloading the work to a trusted entity (Trusted Entity).The user may have multiple trusted entities. He may switch them at any time (or operate his own) and they are not globally trusted by all users.

Trusted Entities(cont'd.)

He may choose to:

- Cross check these against each other (since they should all return the same key)
- Have one for each TLD (an entity may only follow a subset of TLDs)
- Rotate these for privacy reasons.

All the Trusted Entities are cloud services owned by Private or Public organizations.

Alpha-Beta Pruning

The list of Trusted Entities are listed in a tree structure. We employ the **Alpha-Beta Pruning** which is adversarial graph search algorithm to identify the nearest and best trusted entity for the user.

Need for Alpha-Beta Pruning

- Faster record fetching for the local DNS servers.
- Minimizes the latency and erases the need of finding a best match trusted server.

The algorithm takes into account the following:

- Geographic Location
- Latency

The pseudo-code for the above Algorithm is shown below:

```
function Max-Value(state, game, œ, ß) returns the mimimax value of
state inputs:
state, current state in the game
game, game description
œ, the best score for MAX along the path to state
ß, the best score for MIN along the path to state
if CUTOFF-TEST(state) then return EVAL(state)
for each s in SUCCESSORS(state) do
œ <-- MAX(œ,MIN-VALUE(s,game,œ,ß)) if œ >= ß then return ß
end
return œ
```

Efficiency of Alpha-Beta Pruning

With Alpha-Beta Pruning the number of nodes on average that need to be examined is O(b d/2) as opposed to the Minimax algorithm which must examine 0(b d) nodes to find the best move. In the worst case Alpha-Beta will have to examine all nodes just as the original Minimax algorithm does. But assuming a best case result this means that the effective branching factor is b (1/2) instead of b.

Implementation

We plan on using the following tools to implement the project.

- Python, for the implementation of Blockchain.
- CloudSim / CloudSched, Java based open-source cloud simulators.

The end product would be GUI based simulation, demonstrating the resolving of DNS Queries as well as Cloud features such as Resource Pooling etc..

Targetted Results

The Cloud based approach for Trusted Entities utilizes the Alpha-beta algorithm with enough efficiency can give great optimization and advantages in the following:

- Faster access time.
- Easy migration from one server to another.

Targetted Results

The following project intends to deliver on the following:

- **Ease of migration:** The solution coexists with the existing DNS and PKI (Public Key Infrastructure). Adopting this should not break backwards compatibility.
- **Authentication of domains:** Users can verify that they are talking to the legitimate service, without a globally trusted central authorit(y|ies). An attempt at impersonation should become publicly known.
- **Performance:** Minimal overhead compared with the existing systems.

Conclusion

This can run parallel to existing DNS and does not require dedicating TLDs to the system.

- TLD may implement this system if he pleases (or opt-out)
- TLD does not need to contact ICANN (Internet Corporation for Assigned Names and Numbers) to implement this.
- TLD still retains full control of the TLD (including revocations)

Future Scope

- Implementation of secure TLDs.
- A scalable architecture for managing the Trusted Entities.