Internet Naming: Current Systems and Future Directions

Tom Callahan
Case Western Reserve University

December 3, 2012

Introduction

- Mapping human-usable and meaningful names to objects in computer systems is crucial to usability
- Name to object mapping systems also allow for late binding
- The DNS provides this usability and agility with respect to Internet addresses, and is a crucial component of today's Internet
- Many actors influence the mappings provided by the DNS, with many different versions and design objectives

Introduction

- Mapping human-usable and meaningful names to objects in computer systems is crucial to usability
- Name to object mapping systems also allow for late binding
- The DNS provides this usability and agility with respect to Internet addresses, and is a crucial component of today's Internet
- Many actors influence the mappings provided by the DNS, with many different versions and design objectives
- **We must analyze the DNS using both active and passive measurement techniques to examine its behavior and build reliable systems**

Introduction (cont'd)

- The simplicity of the DNS protocol and its unique place in the workflow of Internet usage has encouraged complex implementations
- This simplicity has also enabled other applications to be built wholly on top of the DNS

Introduction (cont'd)

- The simplicity of the DNS protocol and its unique place in the workflow of Internet usage has encouraged complex implementations
- This simplicity has also enabled other applications to be built wholly on top of the DNS
- The DNS is only sufficient for some types of name $\Rightarrow$ object mappings, and the Internet is ripe for new, user-centric naming systems

Areas of Work

- Active Measurement of DNS resolvers on the Internet

Areas of Work

- Active Measurement of DNS resolvers on the Internet
- Analysis of Passive DNS measurements for two user populations

Areas of Work

- Active Measurement of DNS resolvers on the Internet
- Analysis of Passive DNS measurements for two user populations
- A unique, globally distributed key-value store implemented on top of the DNS

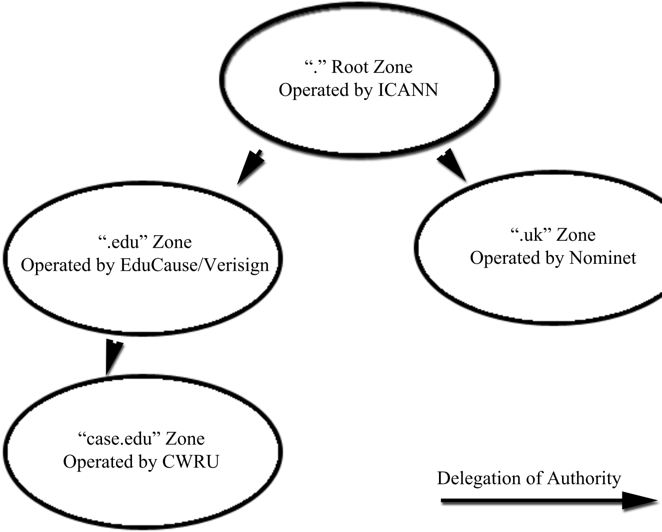
Areas of Work

- Active Measurement of DNS resolvers on the Internet
- Analysis of Passive DNS measurements for two user populations
- A unique, globally distributed key-value store implemented on top of the DNS
- A new foundational system for storing and sharing user-specific meta-information

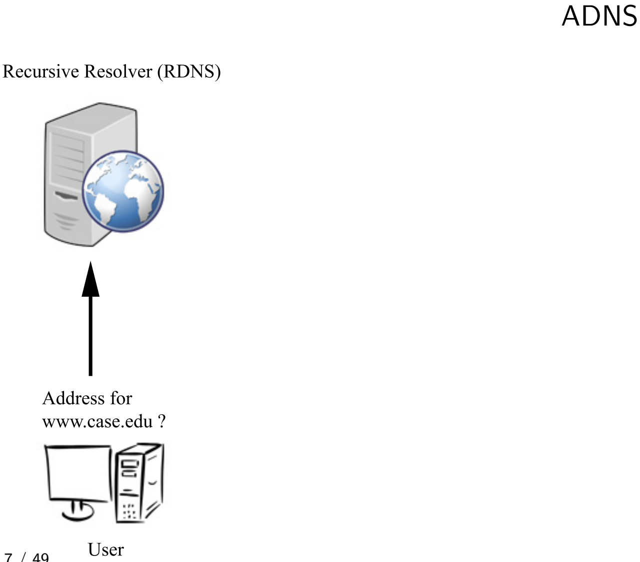
DNS Introduction

- DNS is responsible for converting names to IP addresses
  - www.case.edu ⇒ 129.22.104.136
- Responsible for identifying well-known services
  - case.edu mail exchange (MX) ⇒ smtp.case.edu
- UDP-based protocol with two major actors
  - Recursive DNS Resolvers (RDNS)
    - Do the work of looking up names
  - Authoritative DNS Servers (ADNS)
    - Responsible for handing out answers
    - "Own" a portion of the namespace

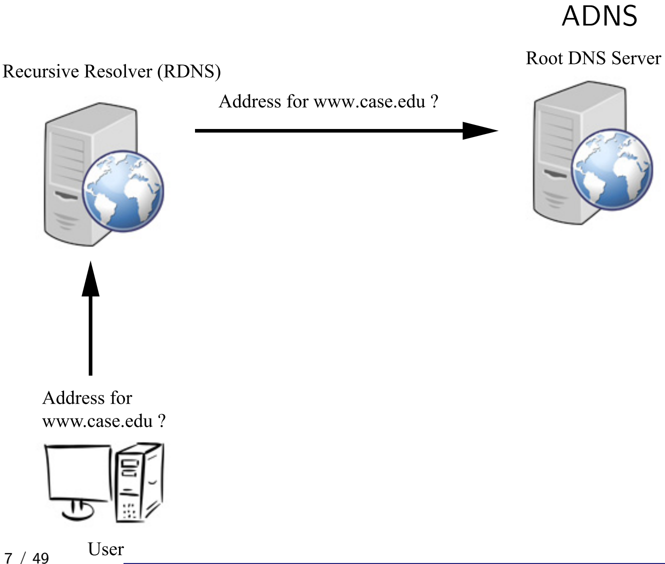
DNS Namespace



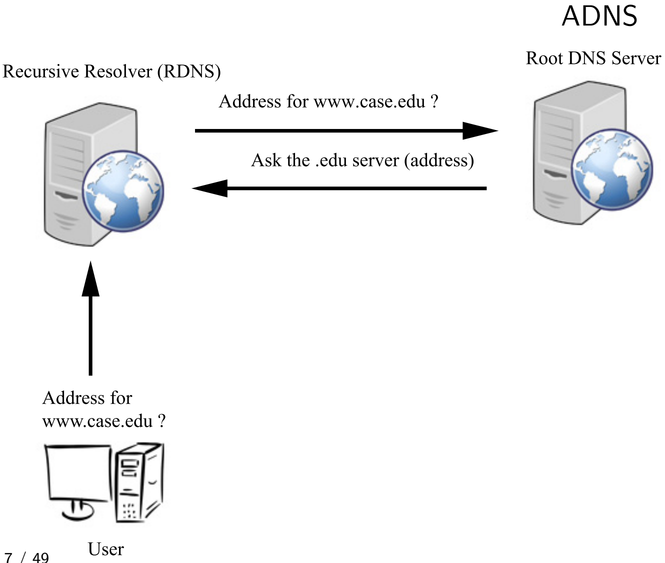
DNS Resolution Process



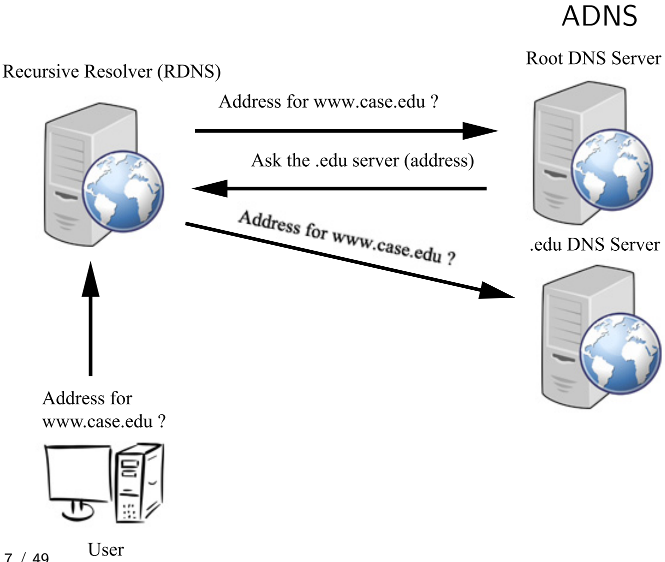
DNS Resolution Process



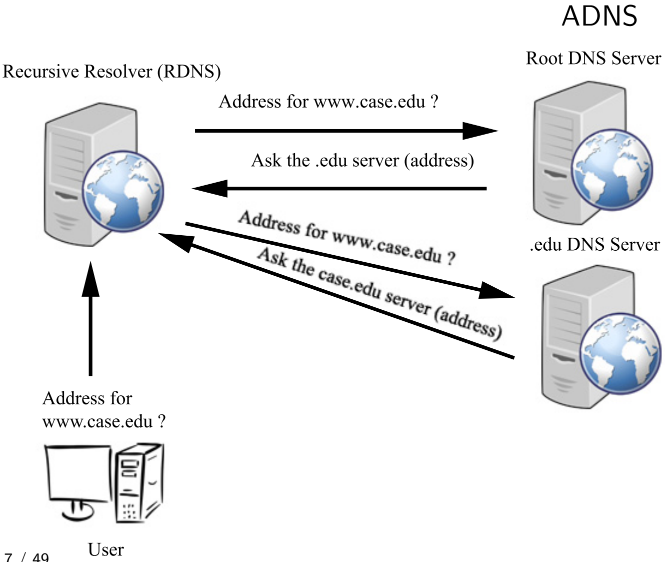
DNS Resolution Process



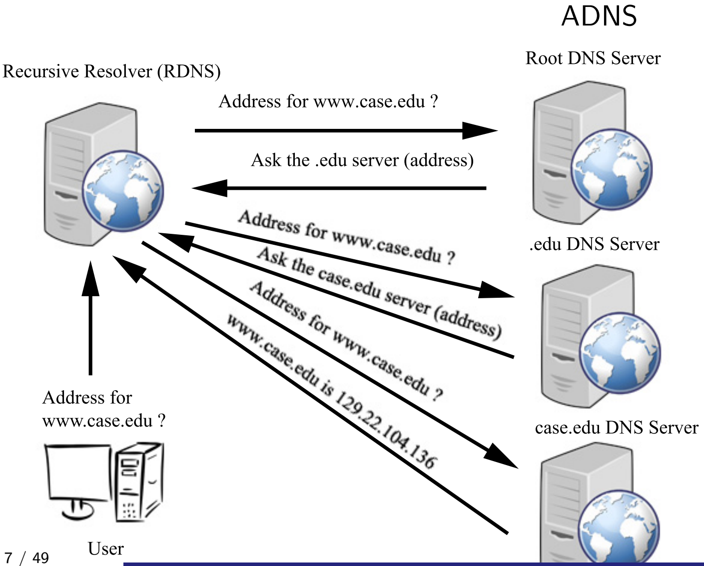
DNS Resolution Process



DNS Resolution Process



DNS Resolution Process



Active DNS Measurement
Joint work with Kyle Schomp

Active Measurement - Problem & Aims

- The 15M open resolvers on the Internet have often been enumerated and sometimes used for measurements, but are not well understood

Active Measurement - Problem & Aims

- The 15M open resolvers on the Internet have often been enumerated and sometimes used for measurements, but are not well understood
- Probe a portion of the millions of systems providing open recursive DNS service
- Characterize the use and misuse of the DNS protocol
- Evaluate the security and topology of DNS resolution paths

Methodology

- Use PlanetLab to scan IPV4 for open resolvers by sending a query falling under a domain we control
- When a resolver is found, send a variety of queries to evaluate aspects of resolver behavior
- By controlling both the initial query and the authoritative response, we get a more complete view of behavior than studies only examining a single aspect

Resolver Structure

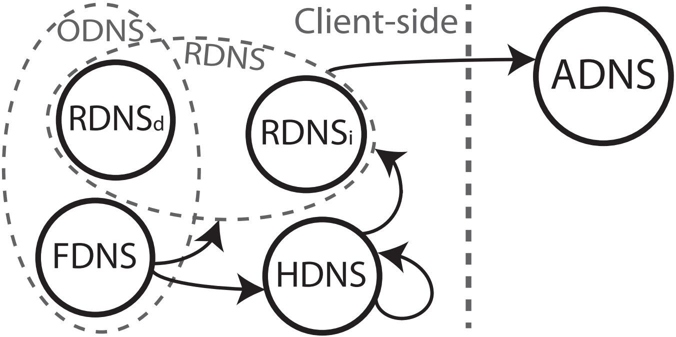


Figure : General structure of the client-side DNS infrastruture[1]

[1] This figure shamelessly stolen from Kyle Schomp

High-level Findings

- Measured nearly 1.1M IP addresses providing open recursive DNS service (ODNS)
- Observed 69K IP addresses visiting our Authoritative DNS (ADNS) server on behalf of these ODNS
- 1.37% (about 16K) of ODNS actually visited our ADNS directly (we define these as $RDNS_d$)
- Of the $RDNS_i$ ($\approx$44K), only 38% would successfully resolve a query sent to it directly

High-level Findings

- Measured nearly 1.1M IP addresses providing open recursive DNS service (ODNS)
- Observed 69K IP addresses visiting our Authoritative DNS (ADNS) server on behalf of these ODNS
- 1.37% (about 16K) of ODNS actually visited our ADNS directly (we define these as $RDNS_d$)
- Of the $RDNS_i$ ($\approx$44K), only 38% would successfully resolve a query sent to it directly
- **Measuring RDNS through their ODNS allows evaluation of firewalled/otherwise prohibited resolvers**
- Full details will appear in thesis

Topology

- Most ODNS access the DNS through a pool of RDNS
- Many ODNS are close to their RDNS – 50% of all ODNS:RDNS pairs have a GeoIP distance of < 100 miles
- Some ODNS are quite far from their RDNS – 10% of pairs have a distance of > 6000 miles (subject to GeoIP accuracy)

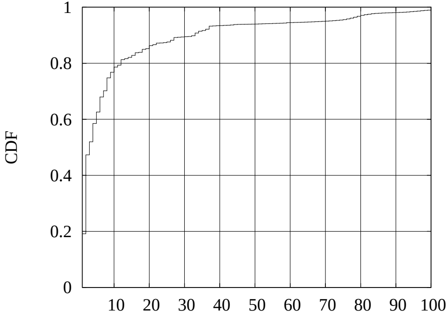


Figure : # RDNS seen on behalf of each ODNS

ODNS Properties

- Previous work [2] has found that ≈2/3 of ODNS are transient on the order of weeks
- We find 41% of ODNS are transient on the order of days
- We often find little competition for cache space – the median duration a record stayed in an ODNS cache is 4.5 hours.

% of Servers Measured	Time Observed Alive
0.6%	$<=$ 10 min
2.2%	(10min, 60min]
11.1%	(60min, 9hr]
15%	(9hr, 1day]
12.1%	(1day, 3day]
58.1%	Alive throughout study

Table : Time Spent Alive

RDNS Properties

- We find that 12.9% of RDNS and 8.3% of $RDNS_i$ remain vulnerable to the Kaminsky attack
- Only 0.3% of RDNS encountered use 0x20 encoding to incorporate additional entropy
  - This may be an underestimate, as some RDNS providers (Google) are known to use 0x20 with only whitelisted ADNS
- NXDOMAIN rewriting is widespread – 25% of ODNS experience this

TTL Modification

Expected (sec)	% Liars	Most Common Lie	% of Liars
0	11.43%	10,000	27.19%
10	11.1%	10,000	28.7%
100	2.96%	300	26.85%
1Ks	1.76%	80	30.07%
10K	2.85%	3,600	26.14%
100K	21.82%	86,400	52.6%
1M	89.35%	604,800	74.43%
10M	89.57%	604,800	74.16%
100M	89.58%	604,800	74.11%
1B	89.57%	604,800	74.12%

Table : Summary of TTL Deviations

Passive DNS Observations

Passive Measurements - Aims

- DNS traffic is often a prelude to inter-host communication
- DNS is increasingly used not simply for lookup, but for traffic engineering (replica selection)
- We must re-appraise the state of DNS traffic on the Internet in order to understand how it is changing

Methods and Data

- We examine DNS traffic logs from the border routers of two edge networks
  - Case Connection Zone in Cleveland, OH
    - Fourteen months of daily logs with visibility into Client⇒RDNS traffic
    - 200M DNS queries of which 162M returned an IPV4 answer
  - International Computer Science Institute in Berkeley, CA
    - Over 6 years of logs (one week a month) with visibility into RDNS⇒ADNS traffic
    - 526M DNS queries of which 139M returned an IPV4 answer

TTL Treatment

- We find a year-by-year downward shift in administrator-assigned TTL values

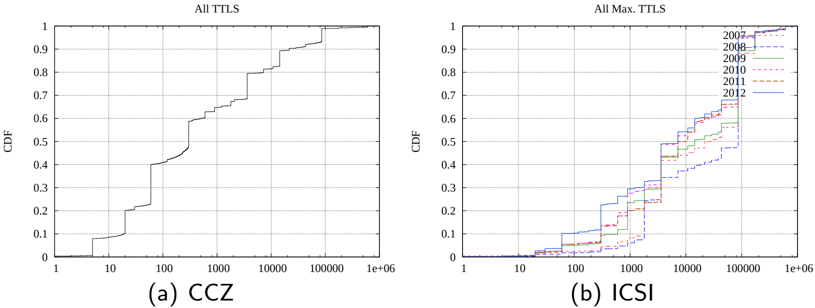
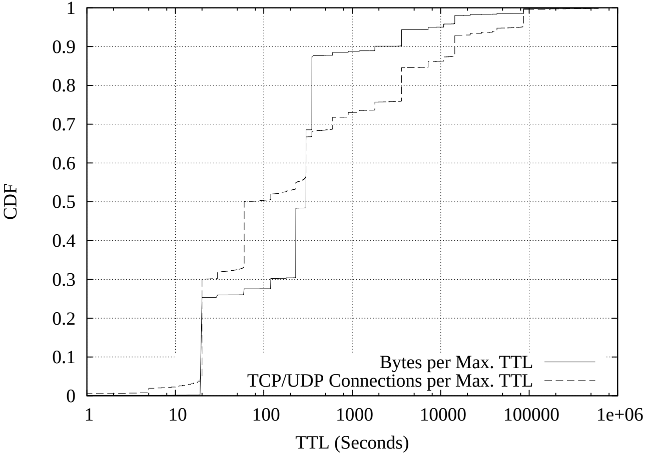


Figure : Max. Observed TTL for each answer record

TTL Treatment (cont'd)

- TTLs of commonly requested DNS records and DNS records corresponding to large data transfers are lower than average



Record Usage

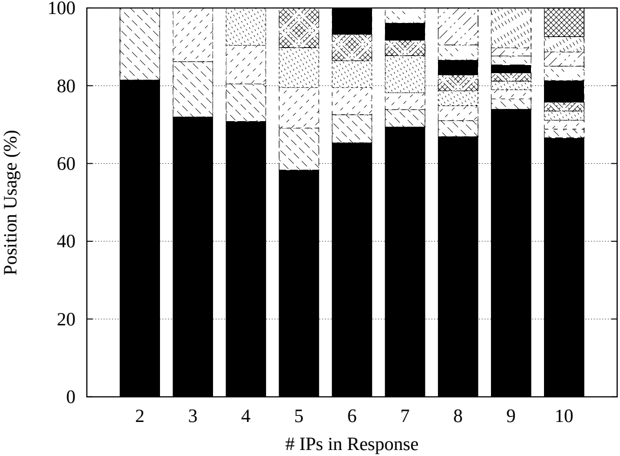
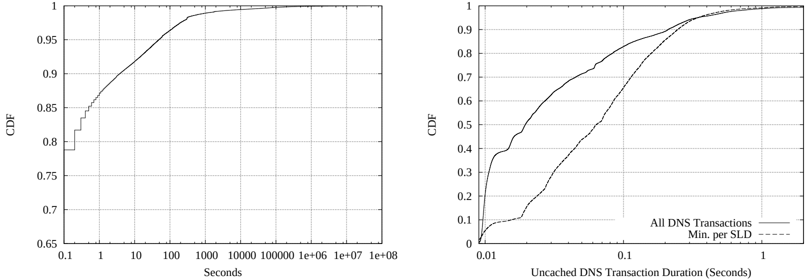


Figure : Position of DNS answer that is used

Performance



(a) Time from DNS response to first connection

(b) Duration of uncached transactions

Figure : Performance

Other observations

- Akamai and Google dominate in the set of DNS answers. 23.5% of successful DNS responses include a mapping to an Akamai server and 13.4% of responses include a mapping to a Google server.
- We generally find a lower cache hit rate than previous work [1]. While others have observed a 90% cache hit ratio, CCZ users fulfull 2/3 of requests from the cache.
- Our performance observations indicate generally faster DNS performance for CCZ users than in the literature. However, when we examine response time on a per-SLD basis, we find behavior much closer to the literature.

DNS Bootstrapping

Bootstrapping Problem

- Peer-to-peer technology has eliminated the need for centralized infrastructure for many applications
  - Notable exception: finding an initial set of peers (bootstrapping)
- Many times policy-based blocking of P2P services is based upon blocking these “rendezvous servers”

Bootstrapping Problem

- Peer-to-peer technology has eliminated the need for centralized infrastructure for many applications
  - Notable exception: finding an initial set of peers (bootstrapping)
- Many times policy-based blocking of P2P services is based upon blocking these "rendezvous servers"
- We aim to design a distributed infrastructure for peer bootstrapping without relying on any fixed infrastructure

Components

- Utilize the 15M [2] ODNS on the Internet as rendezvous points for P2P applications
  - One out of every 300 IP addresses is suitable
- Leverage the caching and aging properties of DNS records to encode arbitrary information in FDNS/RDNS caches
  - Without using a domain we control

Finding the same server

- Assume both clients share some secret "secret"
- Both clients do the following:
  - First IP to scan: sha1("secret"+"IPNumber1")[Last4Bytes]

Finding the same server

- Assume both clients share some secret "secret"
- Both clients do the following:
  - First IP to scan: sha1("secret"+"IPNumber1")[Last4Bytes]
    - "secret" and "IPNumberX" are only strings
  - Second IP to scan: sha1("secret"+"IPNumber2")[Last4Bytes]
  - Scan until X DNS servers found
- This discovery process is independent of the IPs of the clients.

Scanning

- At full speed, hundreds or thousands of packets can be sent per second on a home Internet connection
- Median # of probes sent between detected recursive DNS server IPs is 194, mean 281.
- 99th percentile is 1,284 probes
- Even at slow scanning rates, this is tractable

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $ dig eecs.case.edu
eecs.case.edu.        86400        IN        A        129.22.104.78
```

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $ dig eecs.case.edu
eecs.case.edu.    86400    IN    A    129.22.104.78
eecs.case.edu.    86397    IN    A    129.22.104.78
```

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $ dig eecs.case.edu
eecs.case.edu.    86400    IN    A    129.22.104.78
eecs.case.edu.    86397    IN    A    129.22.104.78
```

We just stored a piece of data in our RDNS Server!

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $ dig eecs.case.edu
eecs.case.edu.    86400    IN    A    129.22.104.78
eecs.case.edu.    86397    IN    A    129.22.104.78
```

We just stored a piece of data in our RDNS Server!

```
eecs.case.edu.    86392    IN    A    129.22.104.78
```

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $ dig eecs.case.edu
eecs.case.edu.    86400   IN   A   129.22.104.78
eecs.case.edu.    86397   IN   A   129.22.104.78
```

We just stored a piece of data in our RDNS Server!

```
eecs.case.edu.    86392   IN   A   129.22.104.78
eecs.case.edu.    86388   IN   A   129.22.104.78
```

Storing Data

An RDNS Server certainly won't accept arbitrary data, but we can insert nearly any valid record into the cache.

```
anomaly@paragon ~ $  dig eecs.case.edu
eecs.case.edu.     86400     IN     A     129.22.104.78
eecs.case.edu.     86397     IN     A     129.22.104.78
```

We just stored a piece of data in our RDNS Server!

```
eecs.case.edu.     86392     IN     A     129.22.104.78
eecs.case.edu.     86388     IN     A     129.22.104.78
```

From the TTL we can determine how long a record has been in the cache

Storing Data (cont'd)

- Method One: test for a record's presence in the cache
  - We may make a request to the DNS server asking it NOT to perform a recursive lookup ("Recursion Desired"=0)
  - If the record is in the cache, it will be returned. Otherwise, it will not
- Method Two: compare the TTLs of multiple records
  - Publisher may request eecs.case.edu and art.case.edu in any order
  - If the received TTL for eecs.case.edu $<$ TTL for art.case.edu, call this a "1" bit
  - Else, consider this a "0" bit

Obtaining DNS Names

- We leverage DNS wildcarding
  - Many domains constructed such that *.domain.com $\Rightarrow$ 1.2.3.4
  - We can therefore leverage the cache hits of bit1.domain.com, bit2.domain.com, etc

Obtaining DNS Names

- We leverage DNS wildcarding
  - Many domains constructed such that *.domain.com $\Rightarrow$ 1.2.3.4
  - We can therefore leverage the cache hits of bit1.domain.com, bit2.domain.com, etc
- Several TLDs are themselves wildcarded

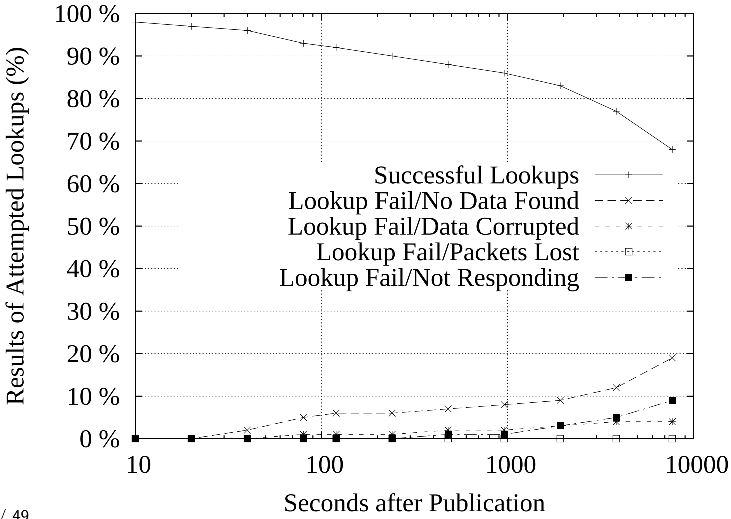
Obtaining DNS Names

- We leverage DNS wildcarding
  - Many domains constructed such that *.domain.com $\Rightarrow$ 1.2.3.4
  - We can therefore leverage the cache hits of bit1.domain.com, bit2.domain.com, etc
- Several TLDs are themselves wildcarded
  - including .ws and .tk

Recursion Desired Success Rate (Publication)

Attempted Publications	72400	100 %
Success	58808	81 %
No Data Found	3356	5 %
Corrupt data	5446	8 %
Packet loss	4790	7 %

Recursion Desired Success Rate (Lookup)



Extending

- Generic bit-pipe, so we can implement:
  - Forward Error Correction
  - CRC Checking
  - Encryption

Metadata Information Storage System

Metadata Problem

- Inter-application sharing of data is ad-hoc at best and nonexistent at worst
  - Facebook can use contacts to populate friends list, but the reverse direction doesn't work
- Users' social graphs are poorly utilized in desktop applications
  - My email client already knows who Mark is, why doesn't my IM app?
- Users now create much of the content on the Internet, but sharing that content often requires an arbitrary third party service
  - Furthermore, these third-party services end up dictating the *name* of the content

Proposed System: MISS

- MISS - Metadata Information Storage System
- Provide a user-controlled naming layer tasked with storing and serving meta-information
- Make meta-information available across hosts and applications in a secure manner
- Allow users to define a name for pieces of content untangled from specific providers or protocols
- Enable new functionality based on wide-spread access to meta-information

Requirements

- Extensibility: MISS must be agnostic to the to the types of data stored and able to handle future applications
- Accessibility: MISS must allow users to expose records at their discretion and on a per record-basis to user-defined groups
- Integrity: Records must be modifiable only by their owner and verifiable by others
- Portability: Users' MISS collections must not be permanently entagled with a particular service provider
- Usability: The compexity of MISS must be abstracted away by applications so that general users find it usable

Collection

- A container for all of a user's meta-information records
- Represented by the fingerprint of a user's public key
- Naming collections by keys ensures that collections may be generated by users without any external help or control
- MISS itself maps these collection identifier's to human-readable, context-sensitive names

Record

- Each record is identified by the collection it is in as well as a name and type (arbitrary strings)
- Names may be provided by users or by applications, types will usually be application-based
- Much like transport port numbers, MISS types and names may be well-known or ad-hoc
- Each MISS record is encoded in XML, and MISS is agnostic to the content of the data portion of the record

```
<miss_record>
  <name>foo</name>
  <type>frob</type>
  <expires>1278597127</expires>
  <signature> [...] </signature>
  <frob>
    <ex1>foo.example.com</ex1>
    <ex2>userA</ex2>
  </frob>
</miss_record>
```

Figure : Example MISS record.

Local Interface - Missd

- Runs on the same device as applications
- Provides a general interface into the global database without application-specific configuration
  - Insofar as its lookup capabilities, this is similar to a DNS resolver
- Provides applications with get() and put() primitives for accessing data repository
- Constructs records using application data, user's encryption keys and privacy settings, and uploads
  - Keeps items in the global repository up-to-date w.r.t. TTL
- Performs lookups on other collections and verifies data received

Global Access - MISS Server/DHT

- Hold and provide access to collections on behalf of users
- Participate in the MISS DHT, a global DHT holding only MISS master records
  - MISS master records identify the MISS server responsible for hosting a given collection ID
  - MISS master records are self-certifying, as they will be self-signed

MISS System Overview

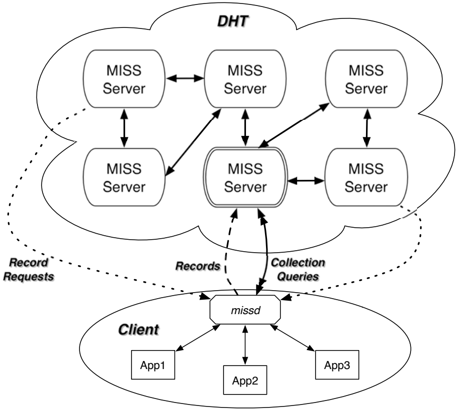


Figure : Conceptual diagram of MISS system.

Bootstrapping

- In order to associate a collection ID with a human-readable name, collection ID's could be shared:
  - Via NFC using smartphones
  - Using X- headers in emails
  - By embedding meta tags in HTML pages
  - Using vCards
  - Via standard directory services (e.g. LDAP, Active Directory)
  - etc...

Use Cases

- Email Clients - “mark:email” or “mark” in lieu of mallman@icir.org
  - Furthermore, email could be automatically encrypted in this case
- Web Bookmarks - “misha:webpage” or “misha” in lieu “of http://engr.case.edu/rabinovich_michael/”
- Application State - Keep tabs open cross-device and cross-browser
- Composable Services - publish desired spam settings to be implemented by all of a user’s email servers

Experiments

- Built a prototype MISS system
- MISS Server (Apache) could sustain up to 27K requests/second
- MISSD imposed parse/validation overhead of 26ms in the 95th percentile
- Built MISS DHT on 100 Planetlab nodes
  - Median record fetch time of 500ms
  - Likely a high overestimate due to lack of locality in PL experiment
  - Fetches mitigated by caching and prefetching
- Undergraduate students were able to build user-facing apps on top of this structure

That's all, folks!

Questions?

Bibliography

J. Jung, E. Sit, H. Balakrishnan, and R. Morris.
DNS Performance and the Effectiveness of Caching.
Networking, IEEE/ACM Transactions on, 10(5):589–603, 2002.

D. Leonard and D. Loguinov.
Demystifying service discovery: Implementing an internet-wide scanner.
In *Proceedings of the 10th annual conference on Internet measurement*, pages 109–122. ACM, 2010.