Understanding Internet Naming: From the Modern DNS Ecosystem to New Directions in Naming

Tom Callahan
Case Western Reserve University

March 25, 2013

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

Topics of Study

- Understanding the Modern DNS ecosystem

Topics of Study

- Understanding the Modern DNS ecosystem
- Communicating without Fixed Infrastructure

Topics of Study

- Understanding the Modern DNS ecosystem
- Communicating without Fixed Infrastructure
- New Directions in Naming

Introduction - Understanding the Modern DNS Ecosystem

- While the original purpose of DNS was to provide hostname lookups, its role has evolved over time
  - Load balancing, geographically-sensitive traffic distribution, blacklists
- DNS behavior varies based upon ISP resolvers and client devices
  - What devices are involved in the DNS resolution process? How do these devices color that process?
- DNS behavior is also driven by users and the hostnames embedded in content by providers

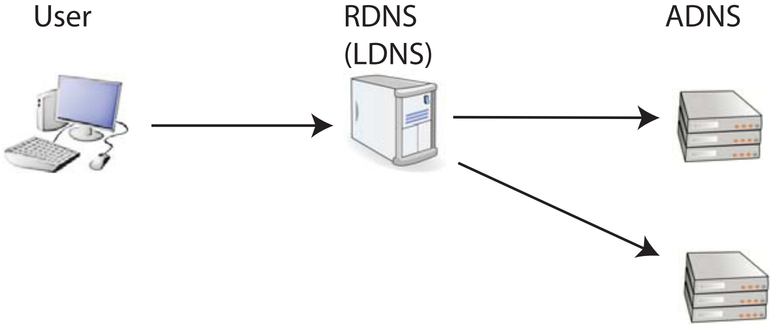


Figure: Simple Resolver Topology

Introduction - Understanding the Modern DNS Ecosystem

- While the original purpose of DNS was to provide hostname lookups, its role has evolved over time
  - Load balancing, geographically-sensitive traffic distribution, blacklists
- DNS behavior varies based upon ISP resolvers and client devices
  - What devices are involved in the DNS resolution process? How do these devices color that process?
- DNS behavior is also driven by users and content providers
- Modern DNS behavior informs design decisions in both current applications and future naming systems
- **We must keep an up-to-date understanding of modern DNS operation through empirical study of both system components and operational DNS traffic**

Introduction - Communicating without Fixed Infrastructure

- Internet transactions need a well-known rendezvous point to establish communication
  - Often a DNS name
- Well-known rendezvous points are inherently brittle
  - To adversaries: censors often block IPs or hostnames used for peer-to-peer traffic
  - To other failures: network problems, power failures, lapses in domain registration for DNS

Introduction - Communicating without Fixed Infrastructure

- Internet transactions need a well-known rendezvous point to establish communication
  - Often a DNS name
- Well-known rendezvous points are inherently brittle
  - To adversaries: censors often block IPs or hostnames used for peer-to-peer traffic
  - To other failures: network problems, power failures, lapses in domain registration for DNS
- **We introduce a mechanism that allows users to communicate without any centralized hub, using a secret name never manifested in the network**

Introduction - New Directions in Naming

- DNS does not encourage user-to-user information sharing
  - Publishing DNS records is often a manual process
  - DNS typically stores mappings to hosts, while users are interested in content and other users
  - DNS has no types suitable storing content URLs or instant-messaging screen names
- Modern names are typically controlled by service providers, rather than users (e.g., "trc36@case.edu")
  - This creates lock-in

Introduction - New Directions in Naming

- DNS does not encourage user-to-user information sharing
  - Publishing DNS records is often a manual process
  - DNS typically stores mappings to hosts, while users are interested in content and other users
  - DNS has no types suitable storing content URLs or instant-messaging screen names
- Modern names are typically controlled by service providers, rather than users (e.g., "trc36@case.edu")
  - This creates lock-in
- **We propose a new naming system centered around users, allowing for secure publication and consumption of records by users and their applications**

Understanding the Modern DNS Ecosystem
Part of this work joint with Kyle Schomp

Goals

- Evaluating DNS system components
  - How does client-side DNS resolution work? What devices are involved? How do they behave?
  - We probe over 1M open resolvers on the Internet to measure topology, security, and protocol compliance

Goals

- Evaluating DNS system components
  - How does client-side DNS resolution work? What devices are involved? How do they behave?
  - We probe over 1M open resolvers on the Internet to measure topology, security, and protocol compliance
- Understanding real DNS traffic
  - What is the nature of DNS traffic on the Internet? How to clients use DNS responses?
  - We examine traffic generated by users of the "Case Connection Zone" to study client requests, server responses, and response usage

Evaluating System Components - 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

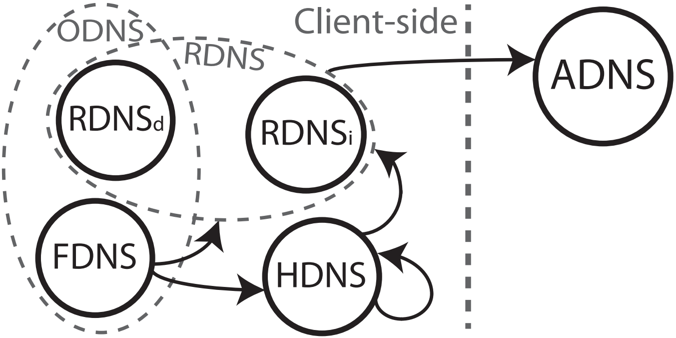


Figure: General structure of the client-side DNS infrastructure[1]

[1] This figure courtesy of Kyle Schomp

High-level Findings

- Measured nearly 1.1M IP addresses providing open recursive DNS service (ODNS)
- Observed 55K 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 *approx* 44K $RDNS_i$ tested for reachability, only 38% would successfully resolve direct query

High-level Findings

- Measured nearly 1.1M IP addresses providing open recursive DNS service (ODNS)
- Observed 55K 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 *approx* 44K $RDNS_i$ tested for reachability, only 38% would successfully resolve direct query
- **Measuring RDNS through their ODNS allows evaluation of firewalled/otherwise prohibited resolvers**
- Full details in dissertation

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 – 7% of pairs have a distance of > 6000 miles (subject to GeoIP accuracy)

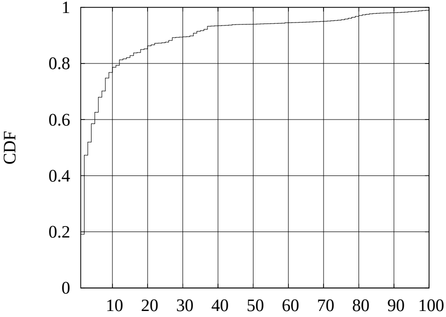


Figure: # RDNS seen on behalf of each ODNS

Security

- 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

Caching

- We find 41% of ODNS disappear before the end of third day
- Little competition for cache space – the median duration a record stayed in an ODNS cache is 4.5 hours.

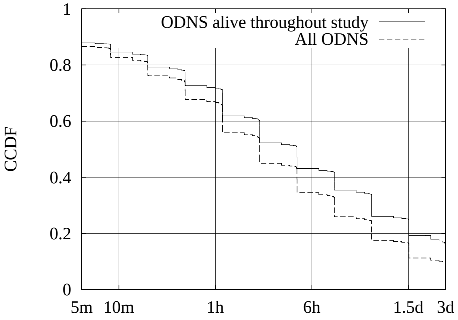


Figure: Cache Evictions over Time

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

Methods - Understanding Real DNS Traffic

- We examine DNS traffic logs from the Case Connection Zone (CCZ) in Cleveland, OH
  - Fourteen months of daily logs with visibility into Client$\Rightarrow$RDNS traffic
  - 200M DNS queries of which 162M returned an IPV4 answer

TTL Treatment

- Per-hostname, there is a variety of TTL modes from a few seconds to a day

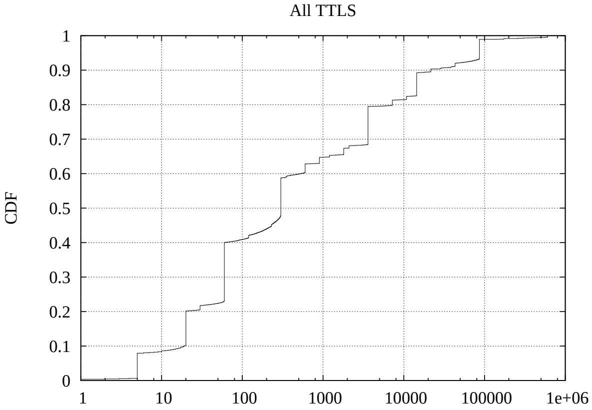


Figure: Max. Observed TTL for each answer record

TTL Treatment (cont'd)

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

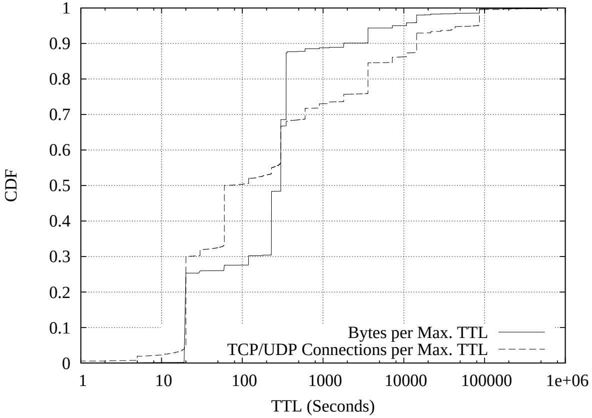


Figure: Weighted Record TTLs

Record Usage

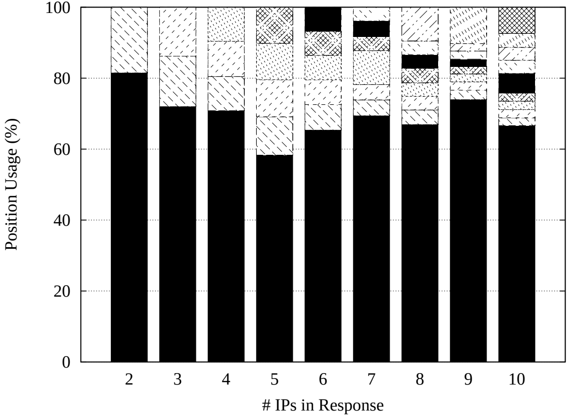
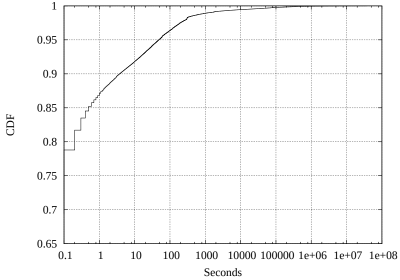
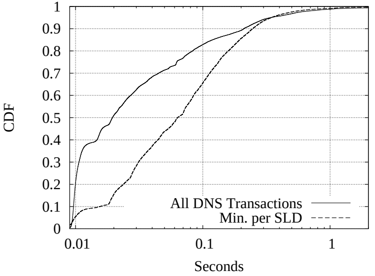


Figure: Position of DNS answer that is used

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 fulfill 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.

Enabling Decentralized Communication

Goal

- Enable users and applications to communicate free of tethers to fixed infrastructure
- Some applications are already free of fixed infrastructure (e.g., peer-to-peer networks)
  - Notable exception: finding an initial set of peers (bootstrapping)

Goal

- Enable users and applications to communicate free of tethers to fixed infrastructure
- Some applications are already free of fixed infrastructure (e.g., peer-to-peer networks)
  - Notable exception: finding an initial set of peers (bootstrapping)
- We design a decentralized mechanism for users sharing some secret (string) to communicate

Components

- Utilize the 15M [2] to 30M [Kyle Schomp] ODNS on the Internet as rendezvous points
  - One out of every 300 IP addresses is suitable
- We utilize these ODNS as independent storage devices
- Leverage the caching and aging properties of DNS records to encode arbitrary information in FDNS/RDNS caches
  - Without using a domain we control

High-level Method

- Publisher uses the secret to generate a list of IP addresses to scan for DNS service
  - Collect a set of suitable IP addresses
- Publisher uses the secret to generate a list of DNS names that will correspond to message bits
  - Store message on each IP address in set

High-level Method

- Publisher uses the secret to generate a list of IP addresses to scan for DNS service
  - Collect a set of suitable IP addresses
- Publisher uses the secret to generate a list of DNS names that will correspond to message bits
  - Store message on each IP address in set
- Using the same secret, the recipient discovers the same set of IP addresses and queries for the same domain names
  - Decodes the message

Finding the same servers

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

Finding the same servers

- 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 open 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 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 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 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 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 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 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 - TTL Method

- Compare the TTLs of multiple records
- Publisher requests messagebit1.tk before or after requesting belowmeare1.tk, based upon bit to transmit
- The recipient requests both records.
  - If the received TTL for messagebit1.tk $<$ TTL for belowmeare1.tk, 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

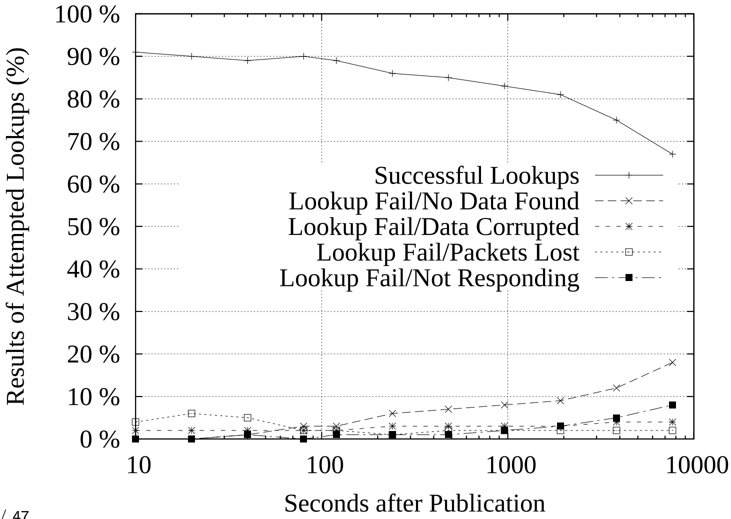
- We leverage DNS wildcarding
  - Many domains constructed such that *.domain.com $\Rightarrow$ 1.2.3.4
  - We can therefore leverage the cache hits of bit1.domain.com, bit2.domain.com, etc
- Several TLDs are themselves wildcarded
  - including .ws and .tk

Success Rate (Publication)

Given a usable server:

Attempted Publications	104K	100 %
Success	92K	88 %
No Data Found	3.6K	3.4 %
Corrupt data	5.0K	4.8 %
Packet loss	3.6K	3.4 %

Success Rate (Lookup)



Extending

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

Enhancements

- Successfully widened the channel by using the value of the difference between TTLs instead of binary comparison
  - We were able to publish and retrieve 140 character tweets
- Eliminated the reliance on wildcard domains
  - When a domain does not exist, an SOA record is returned with the negative response
  - This SOA record has a TTL that counts down
- Enabled communication using a different method relying on cache presence and not TTL

New Directions in Naming

Goals and Use Cases

- Simplify user-to-user information sharing by enabling ordinary users to publish name $\Rightarrow$ object mappings
- Move beyond the host-centric naming scheme of DNS to enable users to name arbitrary meta-information
  - Web Bookmarks - “misha:webpage” or “misha” in lieu “of http://engr.case.edu/rabinovich_michael/”
  - Service-specific identifiers - “misha:skype”
- Combat service-provider lock-in by giving users control over names untangled from specific providers or protocols
  - “mark:email” can be repointed to a new email provider at will

Goals and Use Cases (cont'd)

,

- Enable device mobility by allowing applications to publish configuration meta-information
  - An email account configured on one device could be available on all of a user's devices
  - Browser tabs on one device can be opened on another device in a different browser
- Composable Services - publish desired spam settings to be implemented by all of a user's email servers

Goals and Use Cases (cont'd)

'

- Enable device mobility by allowing applications to publish configuration meta-information
  - An email account configured on one device could be available on all of a user's devices
  - Browser tabs on one device can be opened on another device in a different browser
- Composable Services - publish desired spam settings to be implemented by all of a user's email servers
- Enable new functionality based on widespread access to meta-information
- We propose *MISS*, a new naming system centered around users, allowing for secure publication and consumption of names by users and their applications

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 entangled with a particular service provider
- Usability: The complexity 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

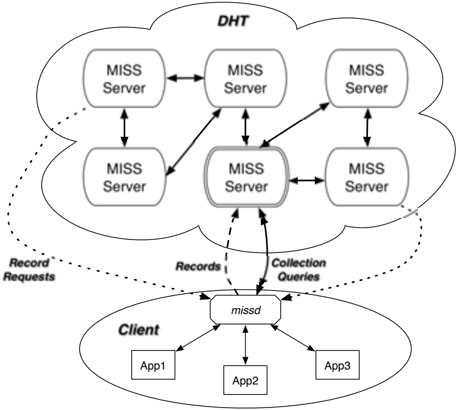


Figure: Conceptual diagram of MISS system.

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...

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 an overestimate due PL performance
  - 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

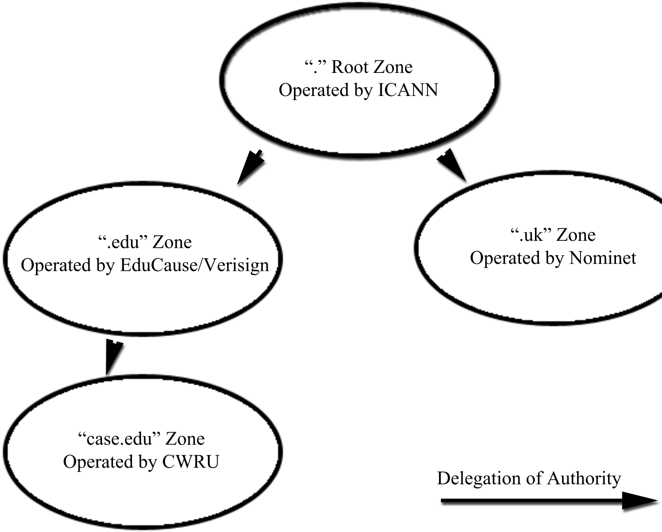
J. Jung, E. Sit, H. Balakrishnan, and R. Morris.
DNS Performance and the Effectiveness of Caching.
Networking, IEEE/ACM Transactions on, 10(5):589–603, 2002.

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.

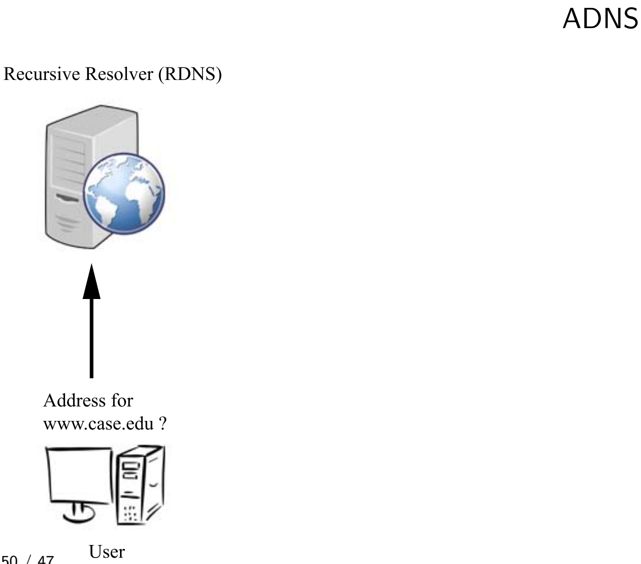
DNS Introduction

- 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

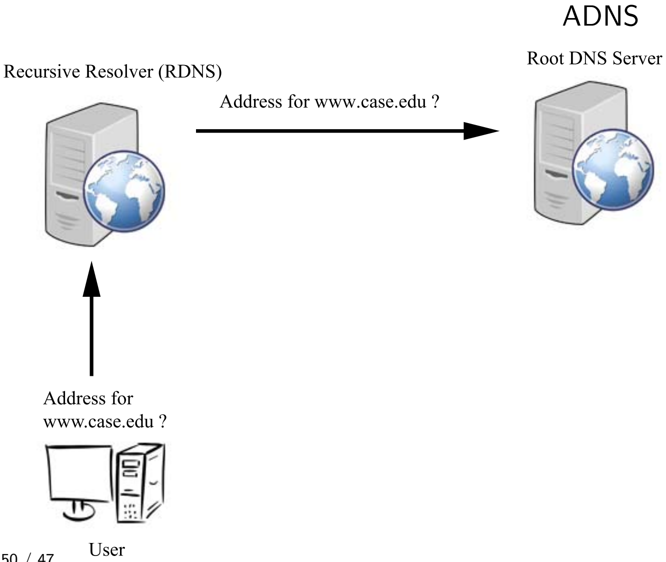
DNS Namespace



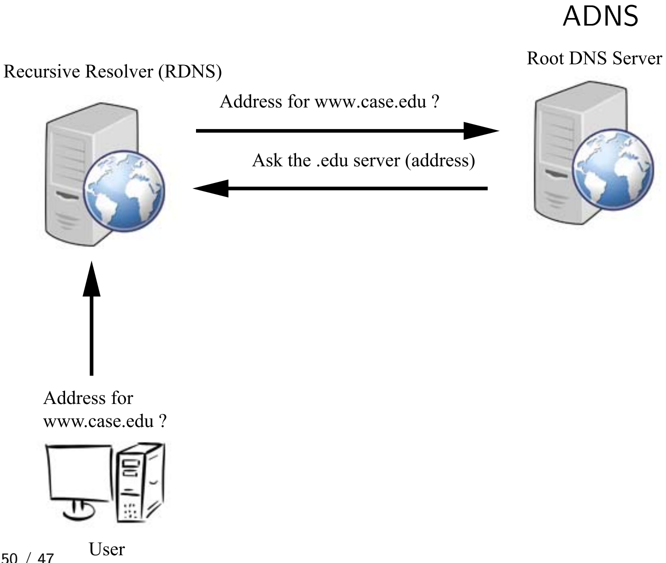
DNS Resolution Process



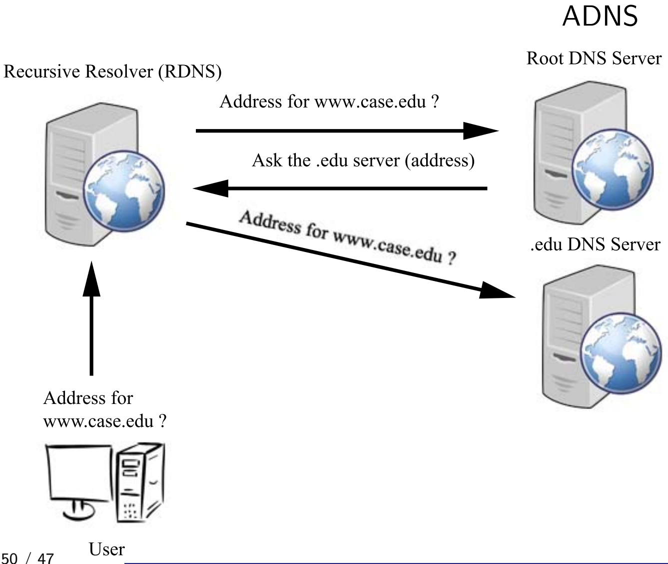
DNS Resolution Process



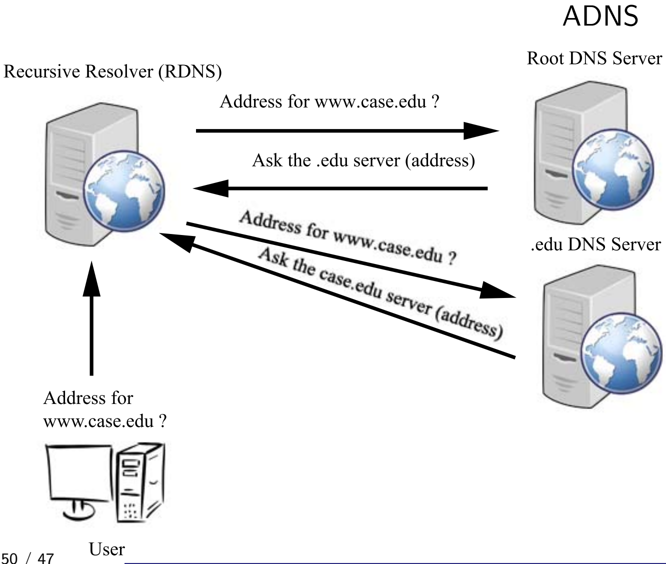
DNS Resolution Process



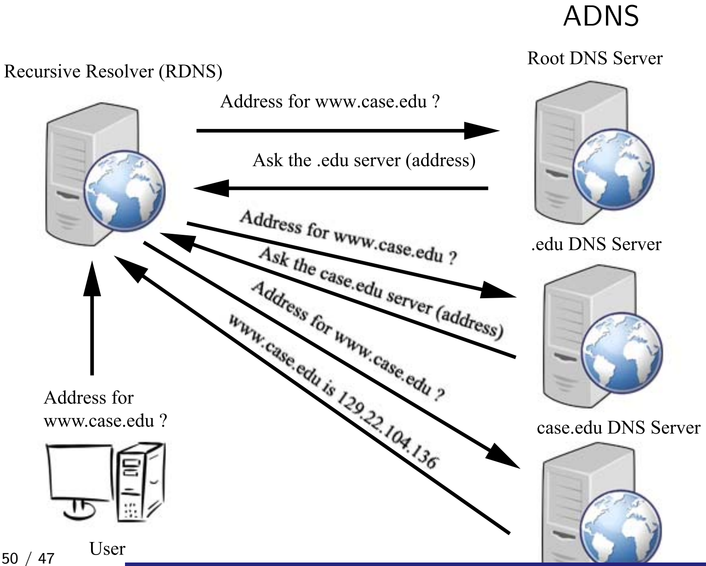
DNS Resolution Process



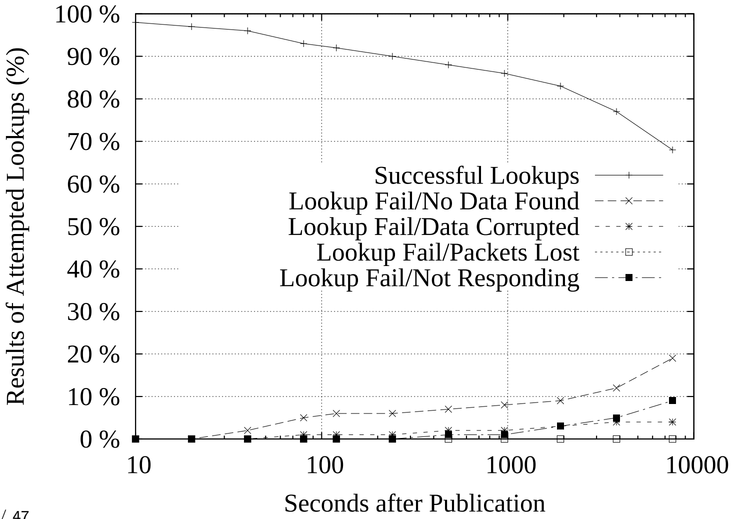
DNS Resolution Process



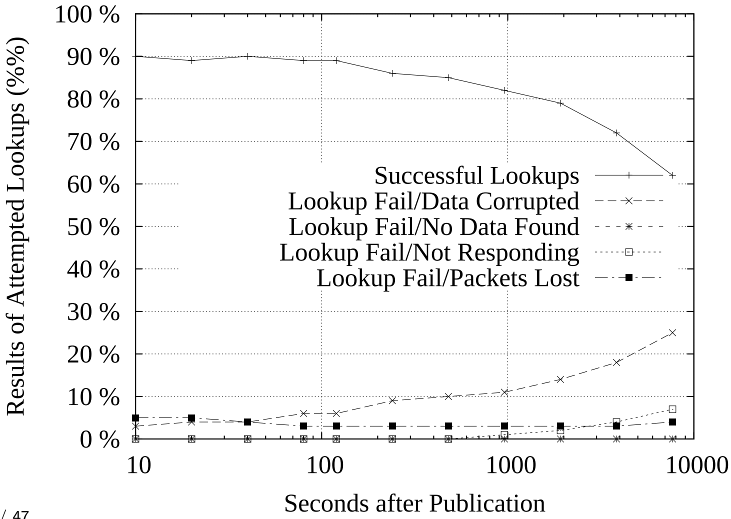
DNS Resolution Process



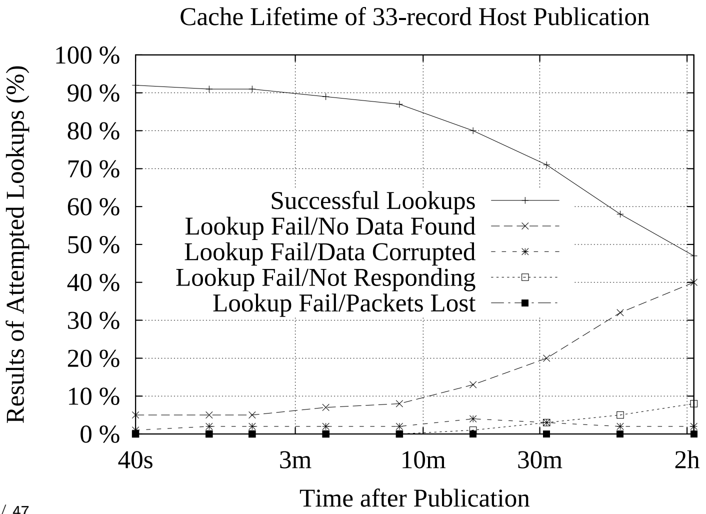
RD Success Rate (Lookup)



Twitter Success Rate (Lookup)



SOA Success Rate (Lookup)



Publications

- PhD papers:
  - Kyle Schomp, Tom Callahan, Michael Rabinovich, Mark Allman. Assessing the Security of Client-Side DNS Infrastructure, European Symposium on Research in Computer Security (ESORICS), March 2013. In preparation.
  - Tom Callahan, Mark Allman, Michael Rabinovich. On Modern DNS Behavior and Properties, ACM SIGCOMM Computer Communication Review, February 2013. Under submission.
  - Tom Callahan, Mark Allman, Michael Rabinovich. Pssst, Over Here: Communicating Without Fixed Infrastructure, IEEE InfoCom Mini-Conference, March 2012.

Publications (cont'd)

- PhD papers:
  - Tom Callahan, Mark Allman, Michael Rabinovich. Pssst, Over Here: Communicating Without Fixed Infrastructure. Technical Report 12-002, International Computer Science Institute, January 2012.
  - Tom Callahan, Mark Allman, Michael Rabinovich, Owen Bell. On Grappling with Meta-Information in the Internet. ACM SIGCOMM Computer Communication Review, 41(5), October 2011.
- MS paper:
  - Tom Callahan, Mark Allman, Vern Paxson. A Longitudinal View of HTTP Traffic. Passive and Active Measurement Conference, April 2010.