DNS Deep Dive

Wes Hardaker  <hardaker@isi.edu>
Geoff Houston  <gih@apnic.net>
João Damas  <joao@apnic.net>
Note Well

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam ([https://www.ietf.org/contact/ombudsteam/](https://www.ietf.org/contact/ombudsteam/)) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- [BCP 9](https://www.ietf.org/rfc/bcp0009.txt) (Internet Standards Process)
- [BCP 25](https://www.ietf.org/rfc/bcp0025.txt) (Working Group processes)
- [BCP 25](https://www.ietf.org/rfc/bcp0025.txt) (Anti-Harassment Procedures)
- [BCP 54](https://www.ietf.org/rfc/bcp0054.txt) (Code of Conduct)
- [BCP 78](https://www.ietf.org/rfc/bcp0078.txt) (Copyright)
- [BCP 79](https://www.ietf.org/rfc/bcp0079.txt) (Patents, Participation)
- [https://www.ietf.org/privacy-policy/](https://www.ietf.org/privacy-policy/) (Privacy Policy)
Overview

- Beyond the DNS basics
  - The underlying DNS distributed database model
  - DNS tree navigation basics
  - DNS Packet Evolution -- Some of the sharp / unusual edges of the protocol
  - Resource Record Types

- Resilience of the system

- DNS Software and APIs

- To be continued at IETF109?
DNS as the novice Internet user sees it

website

www.example.com
DNS as the Techy Internet user sees it

www.example.com?
hostname

HTTP to 93.184.216.34

93.184.216.34!

DNS Server

HTTP Server
DNS is Much Much More Complex

HTTP to 93.184.216.34

93.184.216.34!

www.example.com?

Home Router

ISP DNS1

ISP DNS2

CLOUD DNS

www.example.com?

example.com (x2 v4, 2 v6)

www.example.com?

www.example.com?

www.example.com?

www.example.com?

www.example.com?

DNS Root (x13 v4, 13 v6)

com (x13 v4, 13 v6)
The example.com web page

- Each line is a DNS request
- The center node is an ISP resolver

Query
Authoritative/DNSSEC
Truncated
Response
webmd.com (without caching)

TL;DR: Web pages generate many DNS requests
Lots of requests from you to your ISP

- The resolver remembers some answers
- But must resolve others
The Underlying Distributed Model of the DNS
DNS was created as a replacement for /etc/hosts

Distributed system to replace static information

Back in my day:

127.0.0.1 localhost localhost.localdomain
::1 localhost localhost.localdomain
93.184.216.34 www.example.com

is all we needed.
The DNS ‘tree’

The Root (aka “.”)

Top Level Domains (TLDs)

Second Level Domains (SLDs)

IMPORTANT: name server records in .net (13), .com (13), and .org (6) are not shown in these slides
Resolvers

Resolvers query the tree to find your answer

www.example.com?

DNS resolver types:
- Stub
- Recursive
- Forwarders
- Validating
- Pay Wall
(to be described later)
When resolvers start:

1. They have minimal information about the DNS tree: just the root server IP addresses.
2. The first thing they do is query them to ensure their hard-coded list is still correct

This is called a “priming query”
The DNS is a **distributed protocol via delegations**.

The .net zone delegates everything in .iana-servers.net and below to .iana-servers.net using nameserver (NS) records that point to the authoritative servers for that portion of the DNS tree.
Some DNS Terminology

- **Root**: The top level of the DNS hierarchy.
- **Net**: A zone containing the IP addresses for various organizations.
- **Iana-servers**: A zone containing IP address information.
- **Example.com**: A zone containing information about example.com.
- **Empty Zone**: A zone with no content, often used as a placeholder.
- **Zone Apex**: The top-level domain of a zone.
- **Non-Terminal**: A domain name that is not the terminal or leaf of a zone.
- **Terminal**: A domain name that is the terminal or leaf of a zone.

Diagram:
- The root node is at the top, with the net and example.com zones branching out.
- The iana-servers zone is connected to the net zone.
- The example.com zone contains the terminal domain `www.example.com` and the non-terminal domain `b.a.example.com`.
- The .iana-servers.net zone apex is connected to the iana-servers zone.

- Other domain names in the zone, such as `www.example.com` and `b.a.example.com`, are terminal or leaves.
Duplicate records needed in parent/child zones

The child is the authoritative source!

Should be in both zones

iana-servers.net. NS a.iana-servers.net.
iana-servers.net. NS b.iana-servers.net.
iana-servers.net. NS c.iana-servers.net.
Does this work? -- Yes but actually not well

Unfortunately many zones exist with exactly this problem

The result is timeouts and delays for clients

If a.iana-servers.net can’t answer, this is a “lame delegation” (it’s not authoritative but .net thinks it is)
Trees that refer to the Forest

- Let’s query .com’s servers about example.com:

```
  # dig @a.gtld-servers.net. www.example.com A
  ;; AUTHORITY SECTION:
example.com. 172800 IN NS a.iana-servers.net.
example.com. 172800 IN NS b.iana-servers.net.
```

- The answer: .com doesn’t know where www.example.com is
- But it does know where to send you next: to IANA-SERVERS.NET
- But where is IANA-SERVERS.NET???
  - (here we go again)
Finding Authoritative Servers -- Pictorially

If you ask .com where www.example.com is, they tell you to go ask a completely different part of the tree.
Tricky Tree Grafting -- AKA, what is glue?

# dig @c.gtld-servers.net. iana-servers.net ns (asking .net)

;; ANSWER SECTION:
iana-servers.net. 956 IN NS a.iana-servers.net.
iana-servers.net. 956 IN NS ns.icann.org.
iana-servers.net. 956 IN NS c.iana-servers.net.
iana-servers.net. 956 IN NS b.iana-servers.net.

How do I talk to a.iana-servers.net if it’s inside iana-servers.net itself??

;; ADDITIONAL SECTION:
a.iana-servers.net. 956 IN AAAA 2001:500:8f::53
b.iana-servers.net. 956 IN AAAA 2001:500:8d::53

... (note the random ordering of the answer section)
Including Glue

- `.net’s nameservers` knows where the authoritative source for `iana-servers.net` is
- “In-balliwick” name servers are within the zone itself
  - But `{a,b,c}.iana-servers.net` Must have glue records!
- “Out-of-balliwick” servers are external
  - `ns.icann.org` is out-of-balliwick for `iana-servers.net`
DNS Packet Evolution

```
<table>
<thead>
<tr>
<th>QR</th>
<th>Opcode</th>
<th>AA</th>
<th>TC</th>
<th>RD</th>
<th>RA</th>
<th>Z</th>
<th>RCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QDCOUNT</th>
<th>ANCOUNT</th>
<th>NSCOUNT</th>
<th>ARCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLASS</th>
<th>TTL</th>
<th>RDLENGTH</th>
<th>RDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

24
DNS - A very very simple protocol

- DNS packets ship resource records around
- All **Resource Records** are composed of a triplet
  - A Query Name: “www.example.com” (aka a “domain name”)
  - A Query Type: AAAA = IPv6 address
  - A Query Class: IN = Internet (aka, almost the only value used)
- **Resource Record Sets**
  - ALL matching combinations are an atomic unit
  - You can’t ask for “just 2”
  - They are **not ordered**
- **Response Records** also contain
  - A “**Time To Live**”
  - Response Data
DNS Packet Components

- Header
  - Transaction ID
  - Flags
  - Number of records in each section

- DNS Resource Record Sections
  - Question
  - Answer
  - Authoritative
  - Additional

RFC1035: [This] section contains QDCOUNT (usually 1) entries

Why are multiple questions a problem?
- Do you wait for all authoritative answers?
- What if one authoritative answer has an error and another doesn’t?
- What if there are two different errors?
- …
DNS Packet Sections

- **Question**
  - Where the (single) question goes
  - Repeated in a response

- **Answer**
  - The answer to the question

- **Authoritative**
  - What DNS server is the “true” source for the answers

- **Additional**
  - Anything else you might want to know
    - But shouldn’t trust!
  - E.G., Glue
What happens when DNS things go wrong?

The DNS packet headers contain an “response code” (RCODE) field, yay!

Drat, it’s only 4 bits... There are way more than 16 problems
Let’s get creative about the RCODE problem

What if…. 

   Now bear with me….  
   What if…. 
   We stuck the extra bits somewhere else?

And thus, the “OPT” (pseudo-) resource record was created
EDNS0’s “OPT” record -- more bits! RFC2671

- An “extend” pseudo resource record to add to the additional section
- DNS servers only respond with one if the client indicates support
- Required to support some protocol modifications (e.g. DNSSEC)
- Reuses the Resource Record byte format, but changes many fields

**Features:**
- Total RCODE size becomes $4 + 8 = 12$ bits
- Supports additional protocol flags
- Adds application level max message size / PMTU type discovery
- Adds support for additional DNS extensions

**Used for other extensions:**
- Client Subnet in DNS Queries (RFC7871)
- Extended errors (RFC-TBD)
- ...

## OPT Resource Record Field Reusage

<table>
<thead>
<tr>
<th>RR Field</th>
<th>New Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Must be empty</td>
</tr>
<tr>
<td>TYPE</td>
<td>OPT(41) (16 bits)</td>
</tr>
<tr>
<td>CLASS</td>
<td>UDP Payload Size (16 bits) -- max response accepted</td>
</tr>
<tr>
<td>TTL (32 bits)</td>
<td>Extended RCODE version Flags (8 bits), (8 bits = 0) and (16 bits)</td>
</tr>
<tr>
<td>RDLEN</td>
<td>Data length (same)</td>
</tr>
<tr>
<td>RDATA</td>
<td>Attribute (16-bit)/value (variable length) pairs</td>
</tr>
</tbody>
</table>
Truncation

What happens when a response is too big?

● Greater than the client said it could handle in the OPT/UDP Payload Size

A few things:

● The Truncation bit (TC) is set
● Resource records are removed from the response to make it fit. Maybe.
  ○ Some try to remove unimportant items (the additional section goes first)
  ○ Some servers drop everything and just expect clients to use TCP
  ○ Response Rate Limiting (RRL) -- a DDoS defense -- triggers the TC bit due to query frequency
● Clients need to come back over TCP to get the full answer
  ○ Sometimes clients come back and sometimes they don't if they got the answer they wanted
Ok, but what if you need MOAR errors, text, etc...

What if....

Now bear with me....

What if....

We stuck the extra bits somewhere else?

A soon to be RFC: extended errors! Another OPT

(it's errors all the way down)
DNS Resource Record Types
## Resource Record Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>IPv4 Address</td>
</tr>
<tr>
<td>AAAA</td>
<td>IPv6 Address</td>
</tr>
<tr>
<td>SOA</td>
<td>Zone information at the APEX</td>
</tr>
<tr>
<td>TXT</td>
<td>Free-form text blob</td>
</tr>
</tbody>
</table>
IPv4/IPv6 Deployment: Happy Eyeballs (RFC8305)

Step 1: Send a **AAAA** (IPv6) query
Step 2: Immediately send an **A** (IPv4) query
Step 3: **Wait** for answers from either query
Step 4: If first response is AAAA, open connection. If first response is A, wait a bit (50ms) for a AAAA and then give up and open an IPv4 connection with sadness.
Step 5: **Profit** from your dual-stack deployment!
CNAMEs and DNAMEs

js.example.org. 3600 IN CNAME javas.example.com.
CNAMEs cannot occur at the apex

example.org. 3600 IN DNAME example.com.

CNAMEs are aliases for other tree elements (can be in the same zone or in another)

DNAMEs are aliases for zones themselves

**IMPORTANT:** CNAMEs MUST exist alone at a name (minus DNSSEC entries)
**IMPORTANT:** CNAMEs point to ALL records at the other name (A, AAAA, NS, MX, etc)
MX Records

Mail Exchange (MX) records
- Where should e-mail for a domain-name be sent?
- Prioritized contact list

```
www.example.org. 3600 IN MX 5 smtp.example.org.
example.org. 3600 IN AAAA 93.184.216.34
example.org. 3600 IN MX 10 mail1.example.com.
example.org. 3600 IN MX 20 mail2.example.com.
```

Outsourcing mail service is very common
Wildcards (RFC4592)

- Generating responses for missing data
  - Left most label must be a “*” (and only a “*”)
  - Matches any label that doesn’t already exist
    - Including sub-labels under it
  - Causes a nameserver to **synthesize and answer**
  - **Please read RFC4592!** Good examples therein.

- Example records:
  
  ```
  * .example.com. 3600 IN MX 10 mail.example.com
  host1.example.com. 3600 IN A 192.0.2.1
  ```

- Responses:
  
  ```
  host1.example.com/MX MATCHES
  host2.example.com/MX MATCHES
  host1.example.com/A  DOESN’T MATCH (returns 192.0.2.1)
  host2.example.com/A  DOESN’T MATCH (returns NXDOMAIN)
  ```
For a long time people kept putting TXT records at the APEX
- SPF
- DKIM
- DOMAINKEY
- DNS ownership verification (google, facebook, docusign, …)
- …

The “right” solution was to use a new RRTYPE rather than TXT
- But this was slower to deploy

The new solution: use TXT and RRTYPE records at “_” prefixes
- _spf.example.com. IN TXT - The right “new” for SPF
- _domainkey.example.com. IN TXT - DKIM key publishing
- _25._tcp.mail.example.com. IN TLSA - DANE for secured SMTP (RFC7672)
- _imaps._tcp.example.com. IN SRV - Service host discovery
Summary: DNS is a global distributed identifier DB

Yes, but how does this all scale so well?

I have no idea

Let’s ask Geoff
Extended Errors RFC -- in the RFC editor’s queue

- **SERVFAIL** error is the standard “I couldn’t” response
  - Operators are clueless as to why
  - e.g. most types of DNSSEC validation failures triggers this
- Extended error **adds context** for SERVFAIL (and others)
- With **optional text** providing greater debugging detail