DNS (Domain Name System) Tutorial @ IETF-80

The Domain Name System as a building block in IETF protocol design.

Ólafur Guðmundsson

Shinkuro, Inc.

Peter Koch

DENIC eG

Tutorial Overview

- Goal:
  - Give the audience basic understanding of DNS to be able to facilitate new uses of DNS and take advantage of DNSSEC in the protocols they specify in the IETF.
- Tutorial Focus: Big picture
  - Not software help
    - DNS != BIND
  - No gory protocol details

  - Location of slides:
    - http://

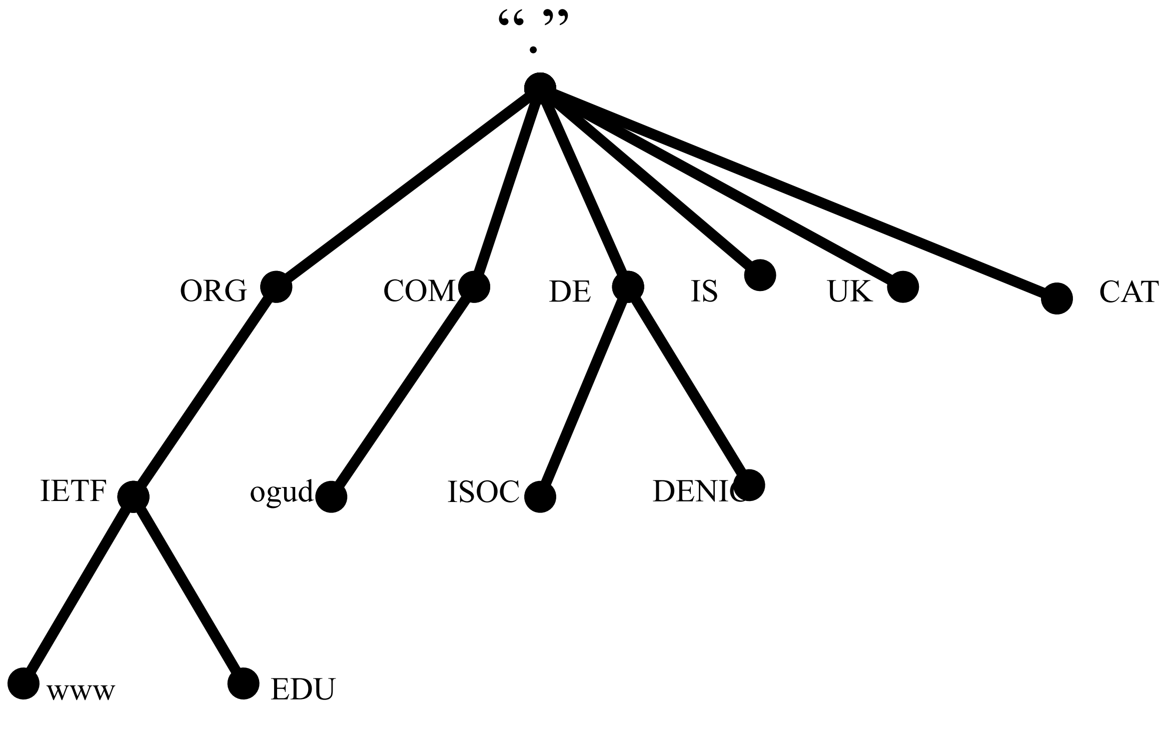
DNS protocol background

DNS Data Model

DNS is global "loosely consistent" delegated database

- delegated -> contents are under local control
- loosely consistent -> shared information (within constraints)
  - does not need to match or be up-to date.
  - operation is global with owners of "names" responsible for serving up their own data.
- Data on wire is binary
- Domain names are case insensitive for [A-Z][a-z],
  - case sensitive for others (`exämple.com` != `exÄmple.com`)
- Hostname [A..Z0..9-] RFC952
  - Restricts names that can be used
  - IDN provides standard encoding for names in non-US_ASCII

DNS tree



DNS Terms

- Domain name: any name represented in the DNS format
  - `foo.bar.example.`
  - `\0231br.example.`
- DNS label:
  - each string between two "." (unless the dot is prefixed by "\")
  - i.e. `foo.bar` is 2 labels `foo\.bar` is 1 label
- DNS zone:
  - a set of names that are under the same authority
  - `example.com` and `ftp.example.com`, `www.example.com`
  - Zone can be deeper than one label, example `.us`, ENUM
- Delegation:
  - Transfer of authority for/to a sub-domain
    - `example.org` is a delegation from `org`
    - the terms parent and child will be used.

DNS functional Elements

- Resolver
  - *stub*: simple, only asks questions
  - *recursive*: takes simple query and makes all necessary steps to assemble the full answer,
  - *caching*: A recursive resolver that stores prior results and reuses them
- Server
  - *authoritative*: the servers that contain the zone file for a zone, one Primary, one or more Secondaries,
- Some implementations perform resolver and server roles.

DNS retrieval mode

- DNS is a "lookup service"
  - Simple queries --> simple answers
    - No search
    - no 'best fit' answers
  - Limited data expansion capability
- DNS reasons for success
  - Simple
    - "holy" Q-trinity: QNAME, QCLASS, QTYPE
  - Clean
    - Explicit transfer of authority
      - Parent is authoritative for existence of delegation,
      - Child is authoritative for contents.

More DNS terms

- RR: a single Resource Record
- RRSet: all RRs of same type at a name
  - Minimum transmission unit
- Example:

```
- <name>     <TTL>  <Class> <RRtype>   <data>
◦ ogud.com.   13630   IN       MX      10 mail.ogud.com.
◦ ogud.com.   13630   IN       MX      90 smtp.elistx.com.
```

- TTL (Time To Live):
  - The maximum time an RRSet can be cached/ reused by a non- authoritative server

DNS Protocol on the wire

- Transport:
  - UDP 512 bytes Payload, with TCP fallback
    - RFC3226 increases to 1220 bytes
  - EDNS0 (OPT RR) (RFC2671) expands UDP payload size by mutual agreement.
  - TSIG (RFC2845) hop by hop authentication and integrity
- Retransmission: built in
  - Resends timed-out-query
    - Possibly to a different server.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA| Z|AD|CD|   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                QDCOUNT  == 1                  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                ANCOUNT                        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                NSCOUNT                        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                ARCOUNT                        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
Query section contains:
  QNAME: <name in domain name format, variable length>
  QCLASS: 2 bytes
  QTYPE:  2 bytes.

Set by query
Set by responder
Unused
```

DNS RR wire format

```
+------------------+-----+------+--------+----+-----------+
+ Domain name      |type | class| TTL    | RL |  RDATA    |
+------------------+-----+------+--------+----+-----------+
    <variable>        2      2      4       2   <variable>
```

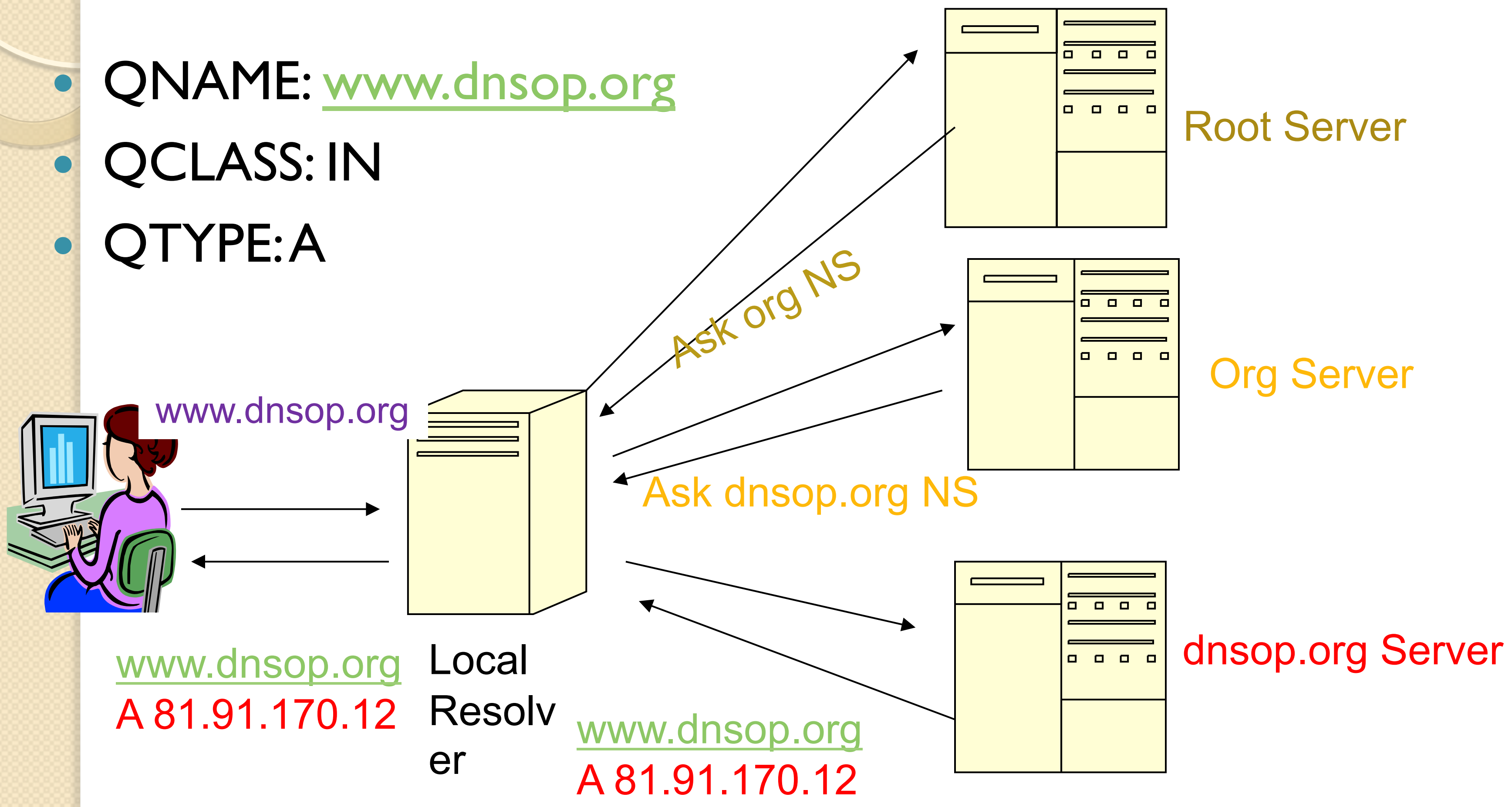
- Owner name (domain name)
  - Encoded as sequence of labels
    - Each label contains
      - Length (1 byte)
      - Name (n bytes [1..63])
      - example.com ➔ 07example03com00
- Type : MX, A, AAAA, NS …
- CLASS: IN (other classes exist, but none global)
- TTL: Time To Live in a cache
- RL: RD LENGTH: size of RDATA
- RDATA: The contents of the RR
  - Binary blob, no TLV (XXX Type Length Value).

DNS data operation

- DNS zone is loaded on authoritative servers,
  - servers keep in sync using information in SOA RR via AXFR, IXFR or other means.
- DNS caches only store data for a "short" time
  - defined by TTL on RRSet.
- DNS Resolvers start at longest match on query name they have in cache when looking for data, and follow delegations until an answer or negative answer is received.
  - Longest match := if resolver has some of the right hand side delegations it will use them rather than start all queries at the root servers.
  - DNS transactions are fast if servers are reachable.

DNS query

- QNAME: www.dnsop.org
- QCLASS: IN
- QTYPE: A



DNS Data property

- Whole or none of RRSet will arrive,
  - in non determined/random order.
- DNS Resolver API may apply RR type specific rules to the order the RR’s are returned.
- DNS data should reside in one place and one place only
  - at name, or at <prefix>.name
  - zone wide defaults do not exist
    - the "zone" is an artificial boundary for management purpose

Existing DNS Record Types:

- DNS Internal types
  - NS, SOA, DS, DNSKEY, RRSIG, NSEC, NSEC3
    - Only used by DNS for its operation
- Indirect RR:
  - CNAME, DNAME
    - Indirect DNS RR cause Resolver to change direction of search
      - Server must have special processing code
- Terminal RR:
  - Address records
    - A, AAAA,
  - Informational/Data
    - TXT, HINFO, KEY, SSHFP
      - carry information to applications
- Non Terminal RR:
  - MX, SRV, PTR, KX, A6, NAPTR, AFSDB
    - contain domain names that may lead to further queries.
- META:
  - OPT, TSIG, TKEY, SIG(0)
    - Not stored in DNS zones, only appear on wire

DNS: New (Unknown) RR types

- Some early DNS implementation **hard** coded RR types.
  - Unknown RR were/are dropped by some resolvers/API's
  - Unknown RR were not served by authoritative servers
    - Implication: introduction of **new** RR types took long time
- Solution:
  - RFC3597 defines that all DNS servers and resolvers MUST
    - support unknown RR types and rules for defining them.
    - suggests a common encoding in presentation format for them.
  - Deployment: (partial list)
    - BIND-9, BIND-8.2.2, ANS, CNS, MS DNS-2003, DNSCache, NSD, PowerDNS, Net::DNS, DNSJava, DNSpython, etc.
  - Issue: Not all DNS APIs are aware of unknown RR types

DNS Wildcards: The area of most confusion: FACTS

- Is not a default but a provisioning aid
- match **ONLY** non existing names
- expansion is terminated by existing names

  →do not expand past zone boundaries

DNS wildcards:

MYTHBUSTERS

- Record:

  `*.example MX 10 mail.example`

  - matches <span>any</span> name, below the name `example` !!
  - supplies RR type to names present, that are missing MX RRs.
    - Is added to the MX RRSet at a name
  - expands only one level
- `www.*.example` will expand

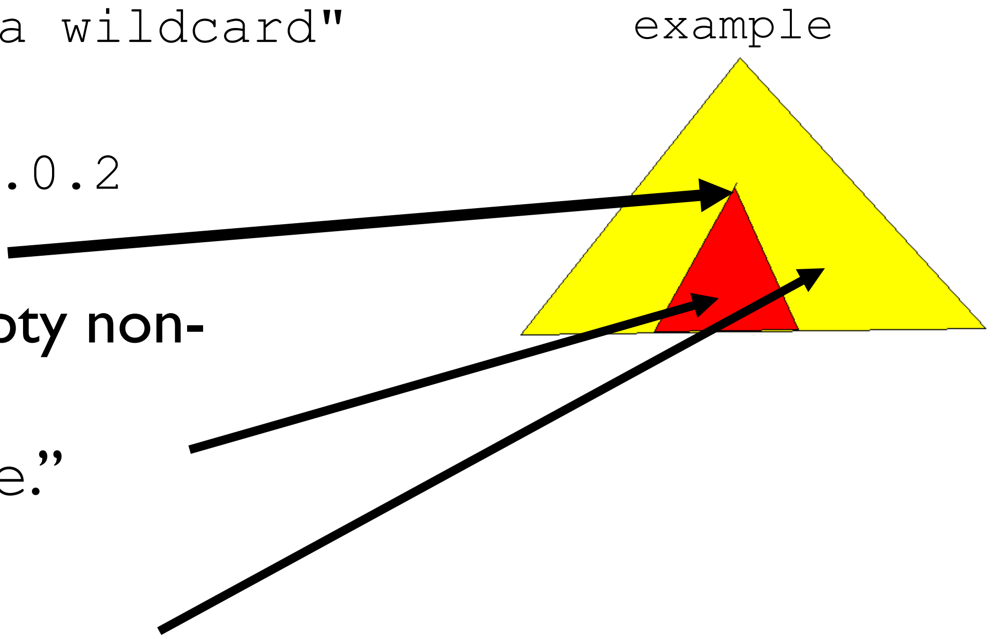
BUSTED

Wildcard Match

- Contents of a zone:

```
*.example. TXT "this is a wildcard"
www.example. A 127.0.0.1
jon.doe.example. A 127.0.0.2
```

- Name "`doe.example`" exists w/o any RRtypes ➔empty non-terminal
- Name "`tina.doe.example.`" will not be expanded from wildcard
- Name:"`tina.eod.example.`" Matched.



DNS rough corners

- Packet size:
  - 512 for standard DNS, 4K+ for EDNS0
    - Some middle boxes restrict UDP fragments → effective <1500 size restriction.
  - Keeping RRSets small is good practice.
- DNS API: not really good by default
  - Restricted to "known" types
  - One query at a time
  - No indication of credibility/security status
- Data integrity: Cache Poisoning
  - DNS answer can be forged, in particular if query stream is visible
  - use protected channel to recursive resolvers.
  - Kaminski attack → Resolvers got much harder to poison, RFC5452
- Broken DNS Software:
  - Middle boxes (firewalls, home routers, load balancers)
  - Not modern DNS resolvers, servers
- DNS name tricks
  - Not a DNS protocol issue but user interface or spoofing

DNS data can change ☺

- DNS Update (RFC2136):
  - adds the ability to change DNS contents of the fly ➔ used a lot.
    - SHOULD only be used for "leaf" data
- Difficult to add/modify data due to operator
  - DNS Secure Update (RFC3007) specifies how to securely delegate capability to update DNS names or name/type(s)
- One RR changes whole zone is sent to secondaries
  - Incremental Zone transfer (IXFR) (RFC1995) enables transfers of only the recently changed data
    - DNS any cast clouds with over 100's of servers use this to maintain large zones that are updated frequently
      - think seconds between updates
  - Notify (RFC1996) informs secondaries that update is available.

DNSSEC

DNS and security

- RFC 3833 covers the threats to DNS transport and resolution.
  - DNS provisioning threats uncovered.
    - Garbage In Signed Garbage Out (GISGO)
- *DNSSEC* is the solution in protocol space
- DNSSEC is gaining traction
  - Root is signed,
  - 69 TLDs signed, 64 have DS in root
    - .org, .net, .info, .cz, .us, .nl, .se, .jp, …
    - .com will add DS this week.
  - Many more soon

DNSSEC: Data integrity and authentication for DNS

- Role: Protect DNS
  - How done: view from 10 km.
    - A DNS RRSet is signed by the zone it belongs to.
    - DS RRSet is vouched for by parent zone.
      - Chain of trust DS→ DNSKEY → DS →DNSKEY
- What DNSSEC does not do:
  - Make data in DNS any more correct

DNSSEC: More details

- Data integrity protection
  - Each DNS RRSet is signed by a digital signature
    - RRSIG containing a signature by the zone private key, for a certain time period
- Existence proof:
  - Chain of NSEC or NSEC3 records lists all names in a zone and their RR types. (authentic proof/denial of existence)
- Parent signs a fingerprint of child's Key Signing DNSKEY (DS RR)
  - allows transition from a secure parent zone to a secure child zone.

DNSSEC and enterprises

- Vendor support is getting better.
- Modern tools allow take care of all the hard work.
- Zone maintenance processes need to change due
  - signing step for new data,
  - Periodic resigning.
- Cost benefit: signing zones may allow reduction in Certificate costs.

DNSSEC verification

- Just do it, almost nothing breaks, cost is small
  - Just make sure you are running RECENT software. (i.e. no extended support versions)
- Only configure the root key and turn on key maintenance.

What does DNSSEC provide to applications?

1. DNS answer with verifiably signed RRSet is known to be identical to what zone maintainer initially entered
2. Widely deployed DNSSEC allows applications to retrieve important data from DNS
   - unsigned keying info
     - IPSECKEY, SSHFP, ***DANE***
   - spoof proof service location
   - Remote Site "authorization"
     - Jabber.ogud.com CNAME jabber.outsourcer.example
   - No need for protocol specific keying infrastructure
   - other...

Design Considerations

To do DNS or not to do DNS

- If your data is small (<2K)
- If the naming of the application objects map into DNS names easily.
- If the providers of the information are close to the names
- If you need “global” access
- If the information is PUBLIC

To do DNS or not to do DNS

- Private/confidential data
- Access control needed
- Large data
- Unstructured
- Naming is difficult
- You need search or match capability
-

Other choices than DNS

- DHCP: if data is consumed locally
    - much better choice
- Service location (see above) and also depends on if accessed via local resource or more "global" one.
    - Enterprise vs site location
    - No search
- Distributed databases

Design Choices for placing new information in DNS.

- New class
    - You need to supply the root servers for it ☺
- New Suffix (TLD)
    - Talk to ICANN
- Use framework like SRV or NAPTR
- Reuse TXT (or some other type)
- <prefix>.name
- New RR Type
- Read RFC5507

Locating a new service: Name or port?

- Email uses port 25, ssh uses port 22, web uses port 80 …..
- What if you want to answer for many "names" with different contexts ?
- Service names free you from what port is used
  - same service can be provided on many ports on same address but in different contexts

SRV Record

- Extensively used in MS Active Directory and OS-X applications
  - Also used by Jabber, sip and other appliations
- SRV format[RFC2782]:
  - Priority, weight, port, host
  - _xmpp-client._tcp.jabber.org.
    - SRV   30 30 5222 hermes.jabber.org.
  - Priority + weight provides capability for simple load balancing.
- SRV works best if you have a TCP or UDP service and want to be able to delegate and distribute

NAPTR

- Role: map name to set of services represented by URI
- SRV doesn‘t help
  - No local part
  - No variable scheme
- Naming Authority Pointer: NAPTR

order	16 bit value
preference	16 bit value
flags	character-string
service	character-string
regexp	character-string
replacement	domain-name

NAPTR frameworks

- NAPTR record does not stand on its own
  - DDDS == Dynamic Delegation Discovery System
    - Used in ENUM and ONS (the RFID name space)
      - These create their own name spaces
      - RFC 3401-3405
  - S-NAPTR == SRV and NAPTR combined
    - Avoids application specific DDDS overhead
      - RFC 3958
  - U-NAPTR == NAPTR maps to single URI
    - Avoids the rewrites
      - RFC 4848
- QNAME for [SU]-NAPTR not easily determined.

Placing New information in DNS: Reuse existing Type

- Needs careful consideration if type is used by core protocols
  - Record type does not stand on its own, needs resolution context before it is useful
  - RBL use A for policy information
    - **BUT** only in non routable address space (127/8)
- TXT may appear as the obvious choice
  - No semantics
  - RFC 1464 sub-typing
  - prefixing could help, but has its own problems
  - TXT verbose for binary information,
  - If new RRSet is large you want EDNS0 support
    - Modern software does this and unknown types as well!!!!
      - MORAL: Fight for local upgrades, do not force the whole Internet to work around your local issues.

Placing New information in DNS: Name prefix, magic name

- Selector put in front of (underneath) domain name:
  - `_axfr.example.org APL 1:127.0.0.1`
  - May interfere with zone maintainer’s naming policy
  - Prefix may end up in a different zone
  - Wildcards will not work like expected, i.e. `_prefix.*.example.org` does not expand
  - No registry for prefixes
- Magic name, e.g. `www`
  - Overloading of multiple names in single application server
  - Again may conflict with naming policy

New RR Type Benefits

- Full control over contents
- Application centered semantics
- Simpler for applications to parse
  - If your specification is simple: KISS
- No collisions, smaller
- Resolution context provided

How to get a new DNS RR type

- Fill out template from RFC 6195 and send to
  - dns-rrtype-applications@ietf.org
- IANA will forward template to an expert and conduct a public review
- DNS expert will render decision based on guidance in RFC 6195.

New Type design guidance

- Tailor it to your needs,
  - Be specific
  - Restrict flexibility (avoid being overly generic)
- Be compact, binary fields are fine
- Ask the experts for help early
  - DNSEXT and DNSOP chairs will help

How to enable the use of new type?

- Provide tools to
  - convert new RR type from/to textual format to RFC3597 portable format for zone inclusion,
  - Provide dynamic update tool of new types.
    - Good tool kits: NET::DNS, DNSJava, DNSpython
- Assume software is modern !!
  - Modern Servers: (partial list)
    - BIND-9, MS DNSServer2003, NSD, PowerDNS, ANS, CNS

New type: TLSA (proposed)

- Goal: allow keying information for a TLS service to be distributed by DNS.
- Requirement:
  - TLS requires a CERT to create connection.
  - DNSSEC validation
- Approach:
  - Full cert or hash of the cert
  - _<port>._tcp.<domain>
  - New RR format, reuse by others expected

New Type: CDS (proposed)

- Goal: Allow child to signal to parent what to place in DS set
- Requirement: Parent has DS for child
- Approach: Reuse DS format.

Pointers to more information

- IETF working groups
  - DNS EXTensions:
    - http://tools.ietf.org/wg/dnsext
  - DNS Operations:
    - http://tools.ietf.org/wg/dnsop
- Individual sites
  - http://en.wikipedia.org/wiki/Domain_Name_System
- DNS book list
  - http://www.networkingbooks.org/dns

RFC starting reading list

- DNS related RFC 100+
  - Many obsolete
- Important ones
  - 1034, 1035 Original specification
  - 4033, 4034, 4035, 5155 DNSSEC
  - 1123, 2181 Clarifications
  - 3597, 2136, 1996, 1995, 3007 Major protocol enhancements
  - 3833 Threat Analysis for DNS
  - 5507 DNS design choices

End of talk

- Extra information provided in background slides
- Questions & Comments

Optimization considered evil

- Problem:
  - Frequently Non-terminal records proposed demand that, terminal records be returned in answer ==> Additional section processing
- Facts:
  1. Additional section processing is done in servers
  2. Before updated servers are deployed RRtype aware resolvers need to do all work.
  3. Not all authoritative servers may have the necessary glue
  4. Glue may not fit
  5. Recursive resolver may have data already
  6. Roundtrips are cheap, parallel is good
  7. Lacy resolver writer will ASSUME additional section processing is done
- Result:
  - Recursive Resolver has to be able to do work forever,
- Moral: Do not attempt to optimize DNS, it causes more problems than you can imagine.

DNSSEC: impacts

- Zones
  - become larger
  - need periodic maintenance
  - have to deal with key management
- Resolvers need to know Secure Entry Points to signed sub trees.
  - Changes over time, needs updating.
- implementations supporting DNSSEC:
  - NDS, BIND-9, DNSJava, DNSpython, Net:DNS, NDS, ANS, CNS, Microsoft Server 2008/Windows 7