DNS Tutorial @ IETF-63

Ólafur Gudmundsson

OGUD consulting

Peter Koch

DENIC

Tutorial Overview

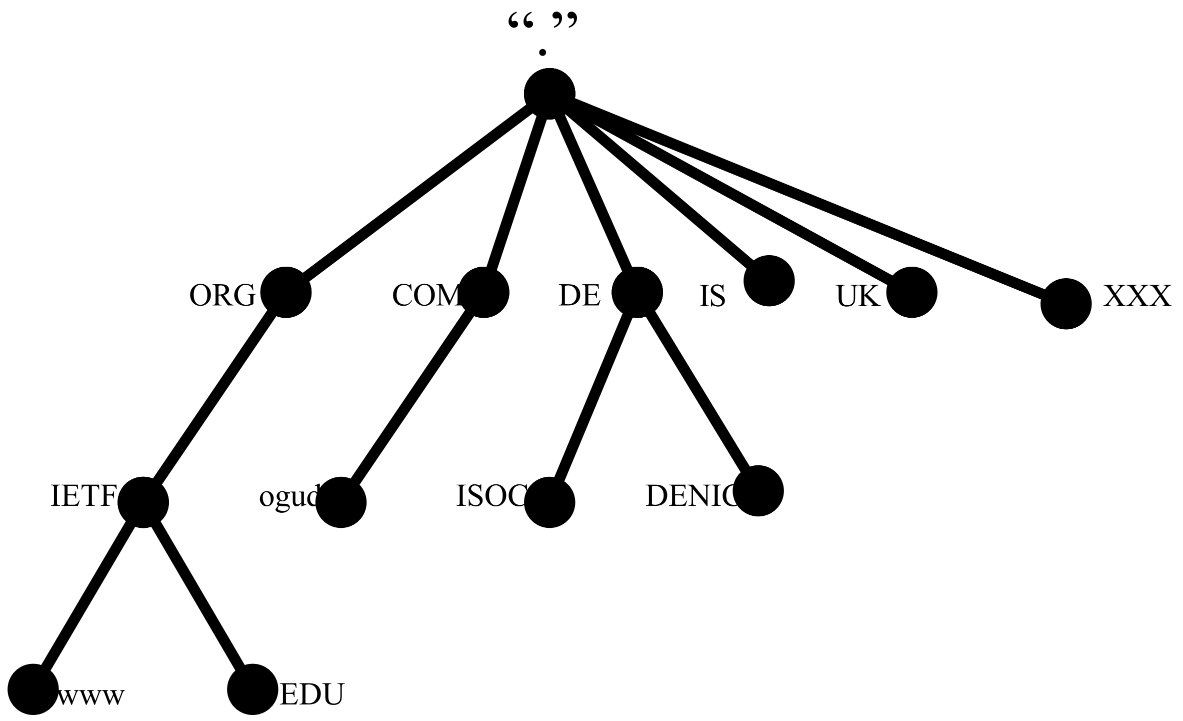
- Goal:
  - Give the audience basic understanding of DNS to be able to facilitate new uses of DNS
- Tutorial Focus: Big picture
  - Not software help
    - DNS != BIND
  - No gory protocol details
- Location of slides: http://www.techfak.net/DNStut.ppt

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
- 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 \
  - ie `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.net`
  - Zone can be deeper than one label, example .us, ENUM
- Delegation:
  - Transfer of authority for a domain
    - `example.org` is a delegation from org
    - the terms parent and child will be used.

More DNS terms

- RR: a single Resource Record
- RRset: all RRs of same type at a name
  - Minimum transmission unit
- TTL: The time a RRset can be cached/reused by non authoritative server

DNS Elements

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

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.

DNS Protocol on the wire

- Transport:
  - UDP 512 bytes Payload, with TCP fallback
  - EDNS0 (OPT RR) expands UDP payload size by mutual agreement.
  - TSIG hop by hop authentication and integrity
- Retransmission: built in
  - Resends timed out query 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.
```

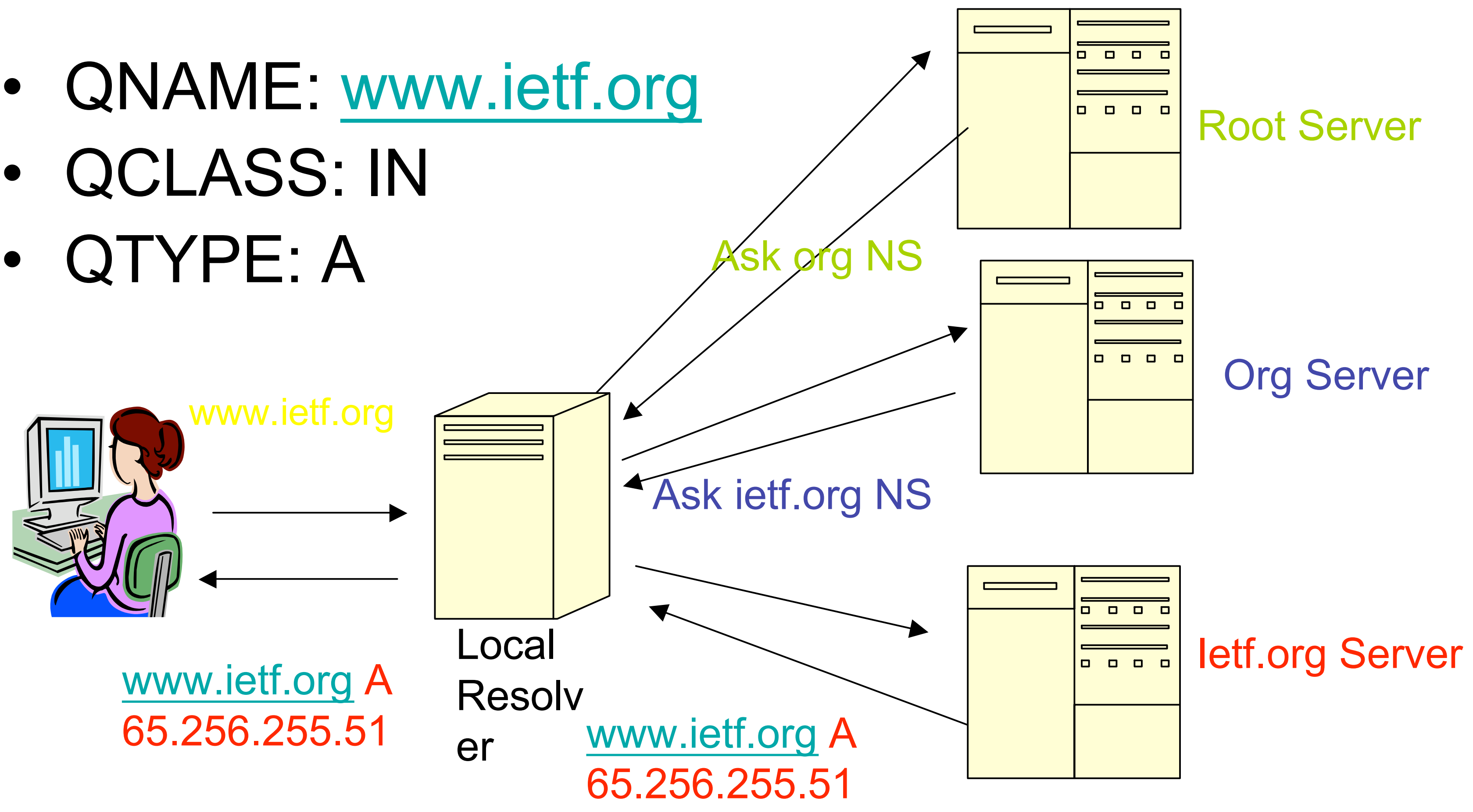
DNS RR wire format

```
+-------------------+-----+------+---------+----+-----------+
+ 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])
      - ogud.com ➔ 04ogud03com00
- Type : MX, A, AAAA, NS …
- CLASS: IN (other classes exist but not global)
- TTL: Time To Live in a cache
- RL: RD LENGTH: size of RDATA
- RDATA: The contents of the RR
  - Binary blob, no TLV.

DNS query

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



DNS Query Model: Question → Answer

```
 Stub_resolver -> Recursive_Resolver    → Auth Server[1]
                                        ←
…….
                 Recurisive_Resolver    → Auth Server[n]
                                        ←
              ←  Recursive_Resolver
```

Stub_resolver has an answer and returns that to the application.

DNS Record Types:

- DNS Internal types
  - NS, SOA, DS, DNSKEY, RRSIG, NSEC
    - Only used by DNS for its operation
- Indirect RR:
  - CNAME, DNAME
    - Internal DNS RR cause Resolver to change direction of search
      - Server must have special processing code
- Terminal RR:
  - Address records
    - A, AAAA,
  - Informational
    - 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 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 a answer or negative answer is received.
  - DNS packets are small
  - DNS transactions are fast if servers are reachable.
  - Tree climbing == BAD
  - Few applications have said that if RR does not exist at name then look for zone default at apex,
    - Zone cut is hard to find by stub resolvers,
    - hierarchy in naming does not necessarily imply hierarchy in network administration.
    - Although DNS name space is hierarchic, there's no inheritance zone wide defaults are also bad due to "apex overload"

DNS rough corners

- Packet size:
  - 512 for standard DNS, 4K+ for EDNS0
  - Keeping RRsets small is good practice.
- Lame delegations:
  - Parent and children must stay in sync about name servers.
  - Secondary servers must keep up-to date with Primary.
    - problems areas: permissions, transfer protocol not getting through, clock synchronization, old/renumbered primary/secondary.
- Data integrity: Cache Poisoning
  - DNS answer can be forged, in particular if query stream is visible
  - use protected channel to recursive resolvers.
- Broken/old DNS Software:
  - Small percentage, but persistent base

DNS API issues

- Whole or none of RRset will arrive,
  – in non determined order.
- DNS Resolver API should
  – Return known weighed DNS RRset in weighed order
  – other RRsets in in random order.
- DNS data should reside in one place and one place only
  – at name, or at <prefix>.name
  – zone wide defaults
    - there are no zone wide defaults, since the "zone" is an artificial boundary for management purpose

DNS Wildcards: The area of most confusion: FACTS

- match ONLY non existing names
- expansion is terminated by existing names
  - do not expand past zone boundaries

DNS wildcards: The area of most confusion: MYTHS

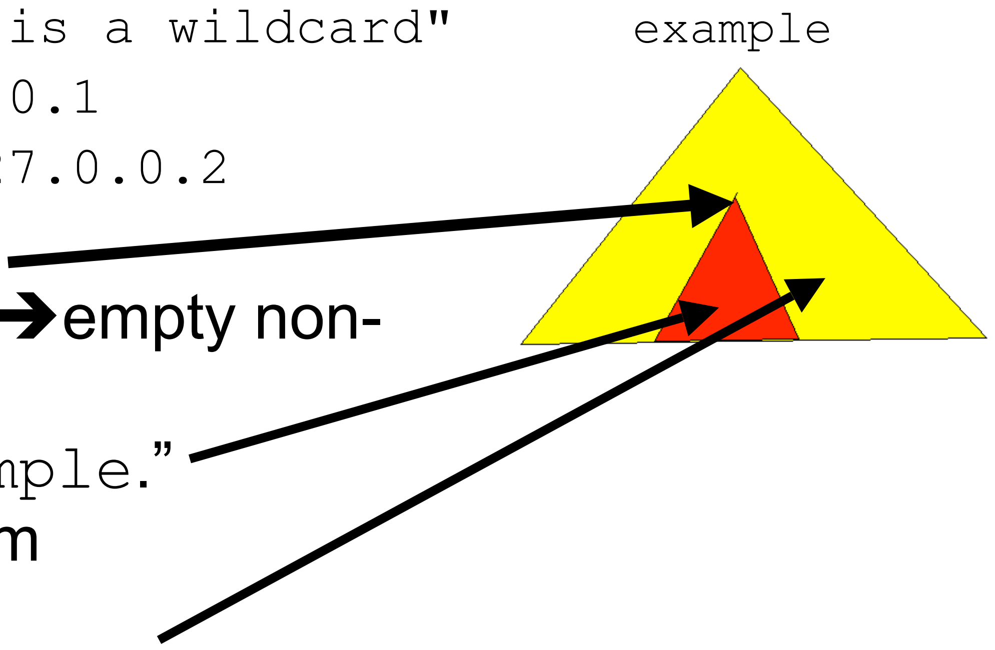
- Record:
  `*.example MX 10 mail.example`
  - matches any name in example !!
  - supplies RR type to names present, missing MX RRs.
    - Is added to MX RRset at a name
  - expands only one level
- `www.*.example` will expand

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.



SRV Record

- mostly used in MS Active Directory
  - Also used by some IM application like Jabber.

- recurring task: given (new) service named COOL, need to offer it
  - old solution: aliases "ftp", "www", ...
  - problem: needs well known port, no exceptions;
    - single target (server) or approximately evenly distributed across multiple addresses

Generalize MX: that COOL SRV

- COOL service in example.org

```
_cool._tcp.example.org  SRV  0  0 5133 srv55.mega.example
_cool._tcp.example.org  SRV 10 20 9876 srv33.mega.example.
_cool._tcp.example.org  SRV 10 20 3456 srv44.mega.example.
_cool._tcp.example.org  SRV 10 40 6738 srv66.mega.example.
```

  "_" avoids conflicts with hostnames
- Services need to be registered
  - currently under discussion: separate registry
  - this is not too good for local service location (-> tree climbing)

When to use SRV

- SRV works best if you have a TCP or UDP service and want to be able to delegate and distribute

- SRV is widely deployed and supported

NAPTR

- Other common task: map name to URL
- 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 and beyond: DDDS

- NAPTR is embedded in a complete framework
- DDDS == Dynamic Delegation Discovery System
- Used in ENUM and ONS (the RFID name space)
  - These create their own name spaces

S-NAPTR

- SRV and NAPTR combined
- Avoids application specific DDDS overhead
- NAPTR leads to more NAPTR or SRV
- SRVs lead to A (or AAAA)

DNS Historic problems and solutions: Static data

- DNS Update (RFC2136):
  - adds the ability to change DNS contents of the fly used a lot.
- 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)

DNS Historic problems and solutions: Unknown RR types

- Early DNS implementation **hard** coded RR types.
  - Unknown RR were/are dropped by some resolvers
  - Unknown RR were not served by authoritative servers
    - Implication: takes long time to introduce New RR types.
- 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:
  - BIND-9, BIND-8.2.2, ANS, CNS, MS DNS-2003, DNSCache, NSD, PowerDNS, Net:DNS, DNSJava, etc.

Current DNS Infrastructure problems

- Old implementations still around as authoritative/caching servers
- Middle boxes have old DNS software or their own that is broken.
  - Some Load balancers do stupid things,
  - Applications interfaces refuse to ask for unknown types

- Majority of the infrastructure
  - is RFC3597 enabled.
  - has EDNS0 support
    - TCP DNS query are frequently blocked.
- "Are you affected" Simple test:
  - http://stora.ogud.com/DNSSEC/unknown/index.html
    - Right now shell scripts, soon a java applet.

DNS Operational problems

- Low TTL: if TTL is low RRset is cached for short time and frequent lookups are required:
  - negative effects: DoS on self and infrastructure, slower lookup,
  - positive effects: Highly dynamic and allows primitive load balancing
- Bad delegations:
  - NS out date in parent
  - NS contains random data to overcome registry requirements
- Old Software still in use after vendor recommends retirement

DNS Operation

- Adding data to reverse tree is problematic
- Adding RR types is not accepted due to people saying
  - Not supported by our software
    - Provisioning, Authorative servers, resolvers, firewalls, middleboxes,
      - take your pick.
  - Do not feel like it
    - Turf war, politics ….
    - …..

DNSSEC: Data integrity and authentication for DNS

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

DNSSEC: More details

- data protections
  - Each DNS RRset has a special RRSIG containing a signature by the zone private key, for a certain time period
- existence proof:
  - Chain of NSEC 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: 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.
- Only few implementations support.
  - BIND-9, DNSJava, Net:DNS, NDS, ANS, CNS

What does DNSSEC provide to applications?

1. DNS answer with verifiably signed RR set(s) is known to be identical to what zone intended.
2. Widely deployed DNSSEC allows application to place more important data in DNS
    - unsigned keying info
        - IPSECKEY, SSHFP
    - spoof proof service location
    - other...

DNS Sub Typing ISSUE

- DNS responses MUST consist of complete RRSets
  - You cannot query for a subset of the RRSet
  - ... nor for partial matches (only QNAME, QTYPE, QCLASS)
- I.e. you cannot ask for, say, at most eight address records (A RRs) for a given name or for only those MX RRs with priority 10 or all TXT RRs containing "money".
- Some RR types are "containers", e.g.
  - KEY (the original)
  - NAPTR
  - TXT (with the RFC1464 convention)
- Subtyping means that the application will have to select their RRs from the response, potentially dumping larger parts of the RRSet, depending on one or more secondary qualifiers buried within RDATA
- ENUM NAPTR overload

SubTyping side effects

- Subtyping results in larger responses
  - (wasted bandwidth) [well, large RRsets are always a DDoS vector]
  - danger of truncation
  - TCP based re-queries
- Subtyping should be avoided when designing new types
- Subtyping can be avoided by
  - dedicated types instead of type/subtype
  - selector prefixes (cf SRV)
- Method of choice depends on number and nature of subtypes expected and the necessity to deal with wildcards

Design Choices for placing new information in DNS.

- New class
  - You need to supply the root servers for it ☺
- New Suffix
  - Talk to ICANN
- Reuse TXT (or some other type)
- <prefix>.name
- New Type

Placing New information in DNS: Reuse existing Type

- TXT may appear as the obvious choice
  - No semantics
  - RFC 1464 subtyping
  - prefixing could help, but has its own problems
  - TXT wastes space, this is still important
  - 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 Type Benefits

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

New type Phase-in:

- consider the "phase in"
  - Do not overdo the problem
  - When you design the new RR type to be used with the existing namespace
    - What does the absence of the type at a name mean? Specify:
      - Feature not available
      - Feature not supported
      - Use application default

How to get a new DNS type

- Rules (see RFC3597)
  1. No additional section processing
  2. No name compression of embedded domain names
  3. Clean definition, no overly complicated structure

- Process:
  1. Write an ID, get review by people that understand your protocol, update draft.
  2. Ask DNS experts (WG chairs) for quick review, update ID
  3. Ask WG(s) for review
  4. Submit to IESG, you get type code from IANA after IESG processes
  5. Advertise new type code.

How to enable the use of new type?

- Make sure your
  - software is "Modern"
  - middle-ware boxes do not get in the way, and/or are updated.
- Provide tools to
  - convert new RR type from textual format to RFC3597 portable format for zone inclusion,
  - perform dynamic update of new types.
    - Good tools: Perl NET:DNS, DNSJava,
- Modern Servers:
  - Bind-9, MS DNSServer2003, NSD, PowerDNS, ANS, CNS

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,
  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 problems.

Pointers to more information

- IETF working groups
  - DNS EXTensions: www.dnsext.org
  - DNS Operations: www.dnsop.org
- Individual sites
  - www.dns.net/dnsrd
  - www.dnssec.net

DNS More resources

- DNS book list
  - http://www.networkingbooks.org/dns

RFC starting reading list

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