DNS at 30

Old and new in a core Internet technology

Jim Hague

jim.hague@acm.org
@banburybill
Sinodun Internet Technologies

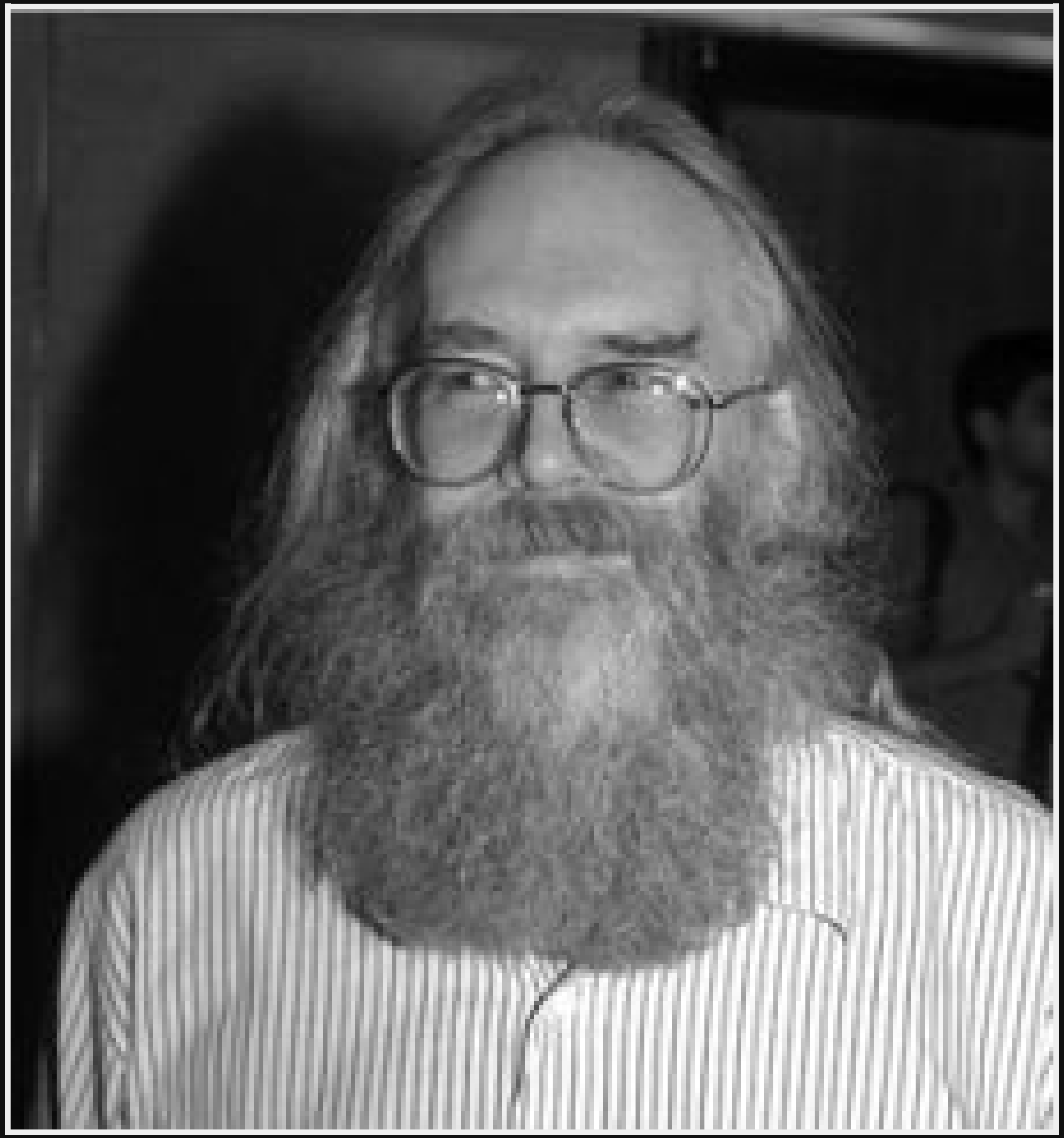
ACCU Conference 2017

Network Working Group   P. Mockapetris
Request for Comments: 1034   ISI
Obsoletes: RFCs 882, 883, 973   November 1987

DOMAIN NAMES - CONCEPTS AND FACILITIES

1. STATUS OF THIS MEMO

This RFC is an introduction to the Domain Name System (DNS), and omits many details which can be found in a companion RFC, "Domain Names - Implementation and Specification" [RFC-1035]. That RFC assumes that the reader is familiar with the concepts discussed in this memo.
A subset of DNS functions and data types constitute an official protocol. The official protocol includes standard queries and their responses and most of the Internet class data formats (e.g., host addresses).



```
...!mcvax!ukc!jmh
```

```
 cbosgd-----------------------------------clyde---------watmath--------utzoo
   |                                        |                            |
 ihnp4-------alberta------ubc-vision       burl                        linus
   | \                        |             |                            |
   | mgnetp                uw-beaver      ulysses--------------bellcore  |
   |                          |                                        \ |
   |                      tektronix-----------------------------------decvax
   |                          |                                        / |
 qantel---------------------\ | /---------oliveb-----glacier-----decwrl  |
   |                         \|/                                         |
   |  kddlab----------------hplabs-------sdcrdcf                         |
   |    |                     |             |                            |
   |    |                     |          sdcsvax                         |
   |    |  (see below)       hao            |                            |
   |    | /                   |          drillsys                        |
   |    |/                    |             |                            |
   |  mcvax-----------------seismo--------gatech----------akgua---------mcnc
   |                         /|\                                         |
lll-lcc-----lll-crg---------/ | \---------cmcl2-----------------------philabs
                                |
                             munnari
```

```
       penet  enea   diku   unido
         |      \     |      /
         |       \    |     /
         |        \   |    /
         |         \  |   /
vmucnam--+----------mcvax------cernvax
         |         /  |  \
         |        /   |   \
         |       /    |    \
         |      /     |     \
       vmars  ukc    prlb2   i2unix
```

```
penet -> Finland
enea -> Sweden
diku -> Denmark
unido -> W. Germany
vmucnam -> France
mcvax -> Netherlands
cernvax -> Switzerland
ukc -> Great Britain
prlb2 -> Belgium
i2unix -> Italy
vmars -> Austria
```

jmh@ukc

```
jmh@ukc.uucp
```

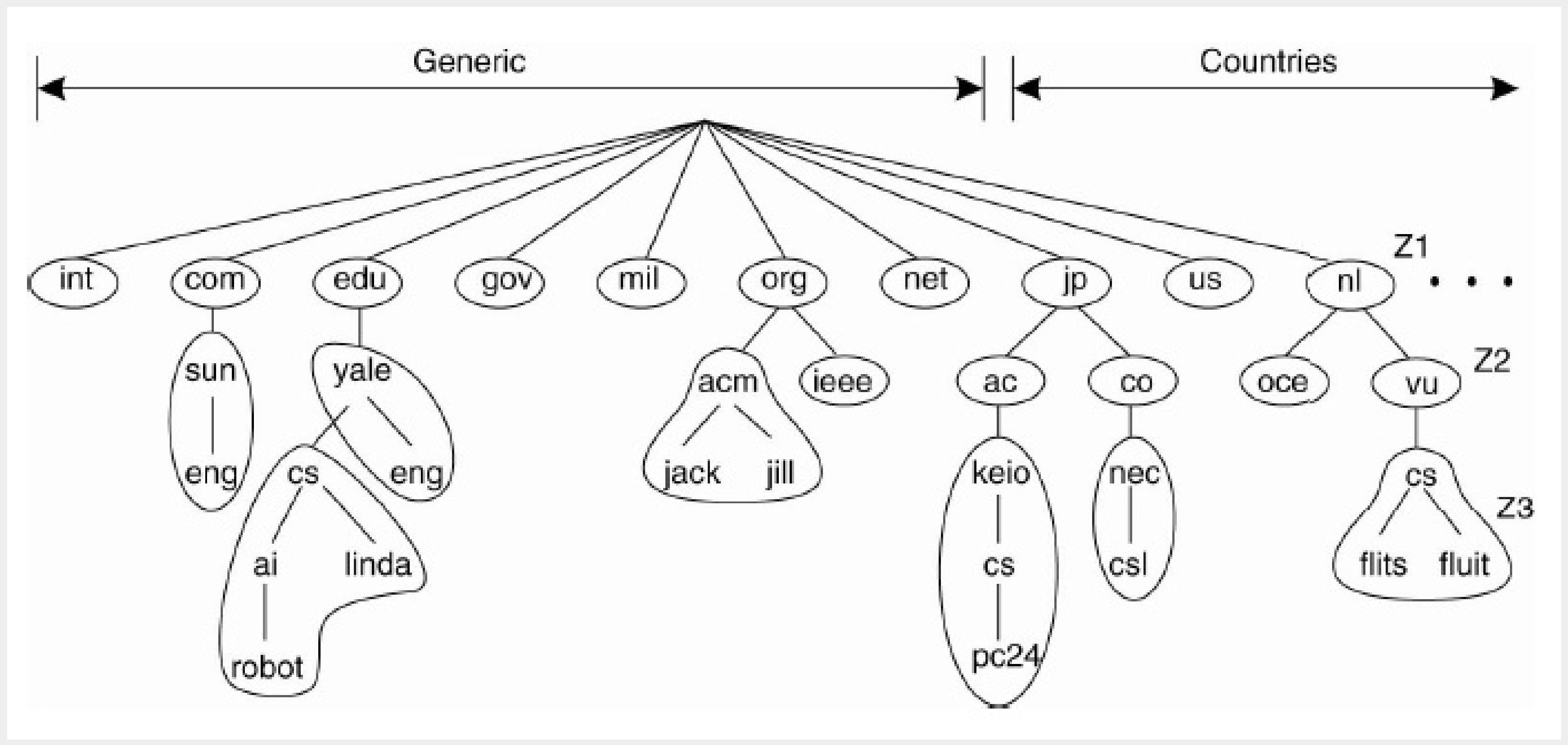
```
Network Working Group                                          J. Postel
Request for Comments:  805                                           ISI
                                                         8 February 1982



                       Computer Mail Meeting Notes

...
Name Domains

   One of the interesting ideas that emerged from this discussion was
   that the "user@host" model of a mailbox identifier should, in
   principle, be replaced by a "unique-id@location-id" model, where the
   unique-id would be a globally unique id for this mailbox (independent
   of location) and the location-id would be advice about where to find
   the mailbox.  However, it was recognized that the "user@host" model
   was well established and that so many different elaborations of the
   "user" field were already in use that there was no point in persuing
   this "unique-id" idea at this time.
```

```
Network Working Group                                     P. Mockapetris
Request for Comments:  882                                           ISI
                                                           November 1983

                 DOMAIN NAMES - CONCEPTS and FACILITIES

        +-----------------------------------------------------+
        |                                                     |
        | This RFC introduces domain style names, their use   |
        | for ARPA Internet mail and host address support,    |
        | and the protocols and servers used to implement     |
        | domain name facilities.                             |
        |                                                     |
        | This memo describes the conceptual framework of the |
        | domain system and some uses, but it omits many      |
        | uses, fields, and implementation details.  A        |
        | complete specification of formats, timeouts, etc.   |
        | is presented in RFC 883, "Domain Names -            |
        | Implementation and Specification".  That RFC        |
        | assumes that the reader is familiar with the        |
        | concepts discussed in this memo.                    |
        |                                                     |
        +-----------------------------------------------------+
```

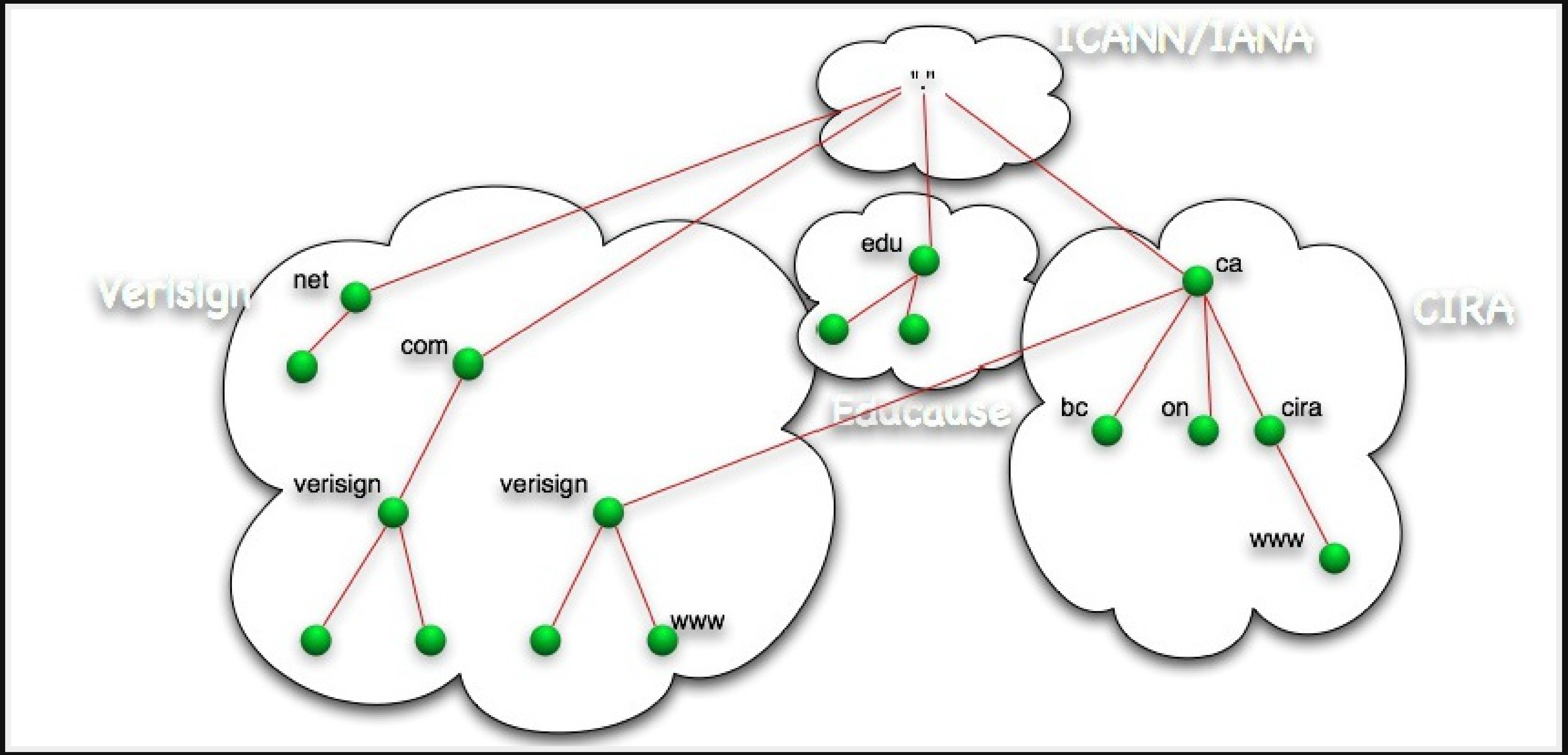
```
Network Working Group                                K. Harrenstien (SRI)
Request for Comments: 952                                M. Stahl (SRI)
                                                         E. Feinler (SRI)
Obsoletes:  RFC 810, 608                                     October 1985

                 DOD INTERNET HOST TABLE SPECIFICATION

STATUS OF THIS MEMO

   This RFC is the official specification of the format of the Internet
   Host Table.

...

LOCATION OF THE STANDARD DOD ONLINE HOST TABLE

   A machine-translatable ASCII text version of the DoD Host Table is
   online in the file NETINFO:HOSTS.TXT on the SRI-NIC host.  It can be
   obtained via FTP from your local host by connecting to host
   SRI-NIC.ARPA (26.0.0.73 or 10.0.0.51), logging in as user =
   ANONYMOUS, password = GUEST, and retrieving the file
   "NETINFO:HOSTS.TXT".
```

DNS

- A consistent namespace used for referring to resources.
- Maintained in a distributed manner.
- Local caching to improve performance.



Domain namespace



Delegation of authority

Authoritative servers

- Contain the data for a zone
- Run by the zone owner

Root servers

- A fixed list of IPv4 and IPv4 addresses for 13 servers
  - `a.root-servers.net` … `m.root-servers.net`
- Operated by various different organisations:
  - VeriSign, Inc.
  - University of Southern California
  - Cogent Communications
  - University of Maryland
  - NASA Ames Research Centre
  - Internet Systems Consortium, Inc.
  - US Department of Defence
  - US Army Research Lab
  - Netnod (Sweden)
  - RIPE
  - ICANN
  - WIDE Project (Japan)

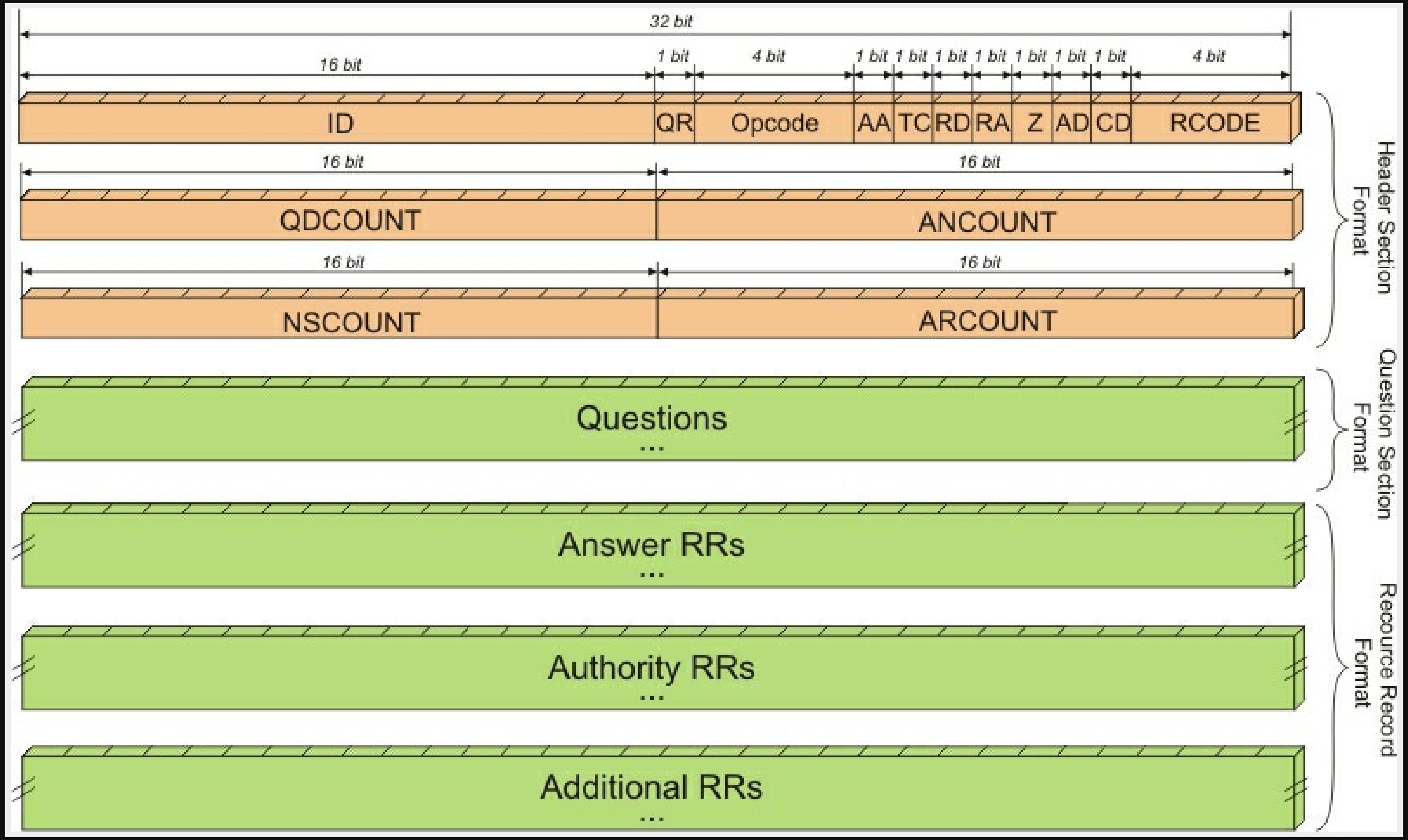
Recursive servers

- Search the hierarchy to resolve queries
- Cache results and reuse them in future queries
- Typically run by ISP or 3rd party, e.g. Google, OpenDNS

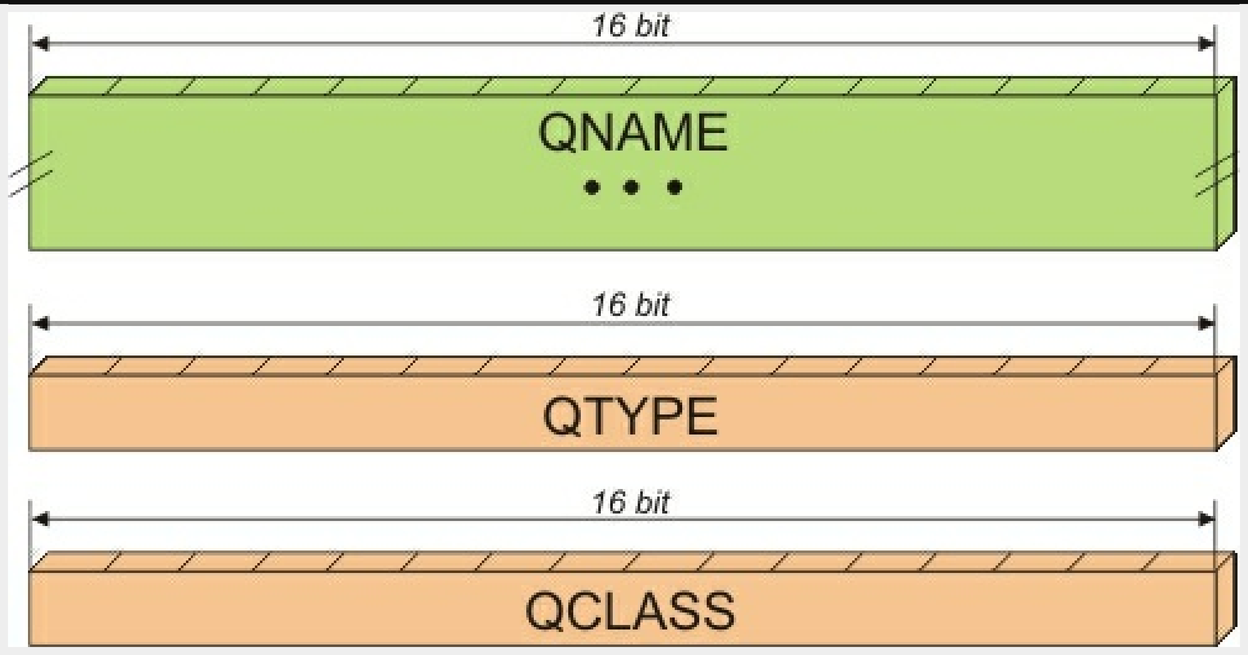
Stub resolver

- Your local name resolution
- Typically using recursive server supplied via DHCP

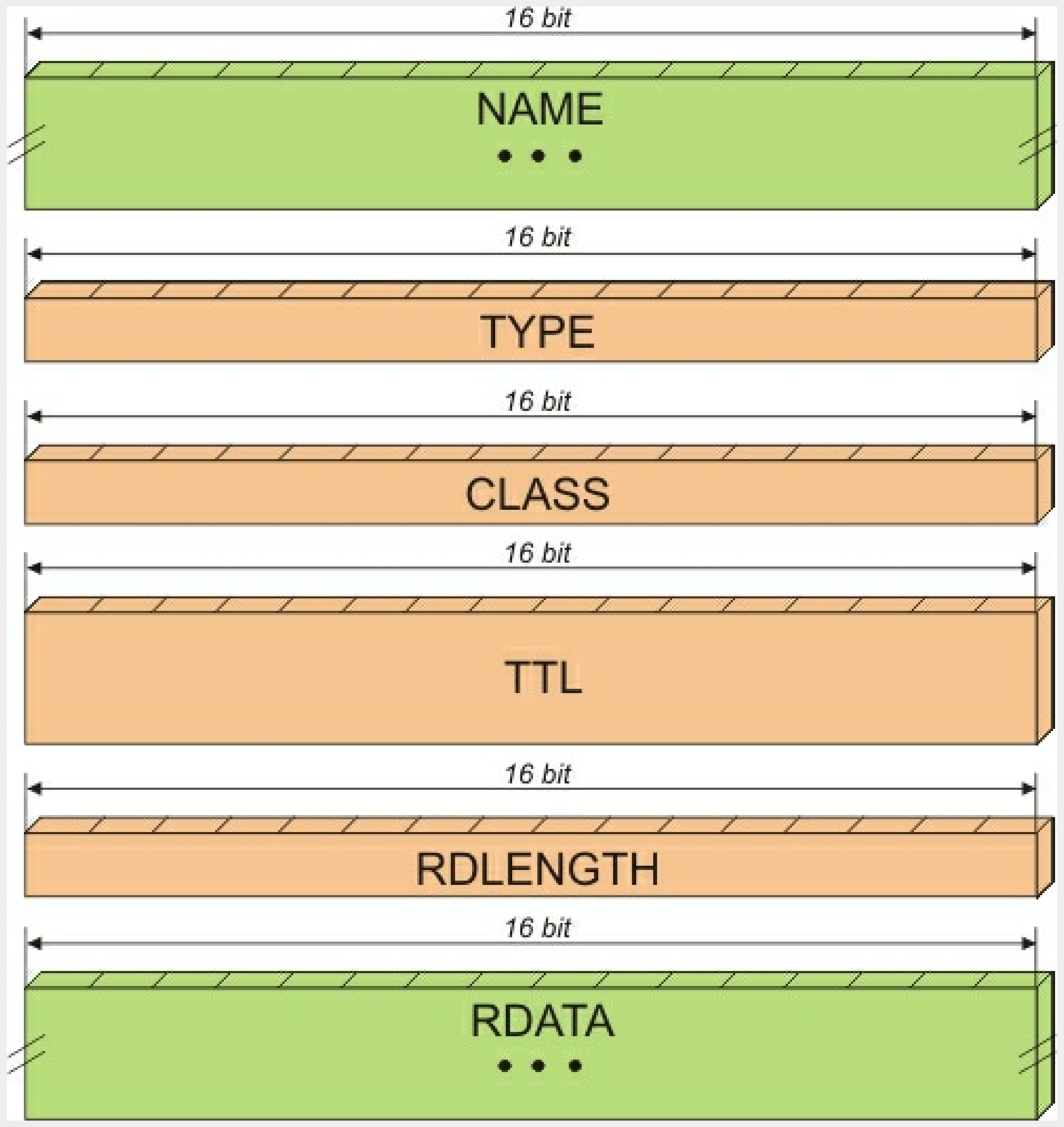
A look at the wire



Format of a DNS message



Format of a Question section



Format of a RR section

Common RR types

A	IPv4 address
AAAA	IPv6 address
MX	SMTP servers for domain
NS	Name servers for domain
PTR	Pointer to canonical name (for IP)
SRV	Specify location of servers for services
TXT	General textual information
SOA	Start of Authority record for zone

Transmission

DNS uses UDP

Transmission

DNS uses UDP

Except when it uses TCP

Terminal window time!

Lookups in C

```c
struct hostent *he;
struct in_addr **addr_list;

if ((he = gethostbyname("accuconference.org.uk")) == NULL) {
    herror("gethostbyname");
    return 2;
}

printf("Official name is: %s\n", he->h_name);
printf("    IP addresses: ");
addr_list = (struct in_addr **)he->h_addr_list;
for(int i = 0; addr_list[i] != NULL; i++) {
    printf("%s ", inet_ntoa(*addr_list[i]));
}
printf("\n");

struct in_addr ipv4addr;
struct in6_addr ipv6addr;

inet_pton(AF_INET, "46.43.8.89", &ipv4addr);
he = gethostbyaddr(&ipv4addr, sizeof ipv4addr, AF_INET);
printf("Host name: %s\n", he->h_name);
```

From Beej's Guide to Network Programming.

```c
struct addrinfo hints, *servinfo, *p;
int rv;
memset(&hints, 0, sizeof hints);
hints.ai_family = AF_UNSPEC; hints.ai_socktype = SOCK_STREAM;
rv = getaddrinfo("accuconference.org.uk", "http", &hints, &servinfo);
if ( rv != 0 ) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
    exit(1);
}
char str[INET6_ADDRSTRLEN];
for(p = servinfo; p != NULL; p = p->ai_next) {
    printf("Canonical name is: %s\n", p->ai_canonname);
    if ( p->ai_addr->sa_family == AF_INET ) {
        struct sockaddr_in *sin = (struct sockaddr_in*) p->ai_addr;
        inet_ntop(AF_INET, &(sin->sin_addr), str, sizeof(str));
        printf("IPv4 address: %s\n", str);
    }
    else if ( p->ai_addr->sa_family == AF_INET6 ) {
        struct sockaddr_in6 *sin = (struct sockaddr_in6*) p->ai_addr;
        inet_ntop(AF_INET6, &(sin->sin6_addr), str, sizeof(str));
        printf("IPv6 address: %s\n", str);
    }
}
freeaddrinfo(servinfo); // all done with this structure
```

```c
struct sockaddr_in6 sa; // could be IPv4 if you want
char host[1024];
char service[20];

// pretend sa is full of good information about the host and port...

getnameinfo(&sa, sizeof sa, host, sizeof host, service, sizeof service, 0);

printf("   host: %s\n", host);    // e.g. "www.example.com"
printf("service: %s\n", service); // e.g. "http"
```

From Beej's Guide to Network Programming.

EDNS0

```
Network Working Group                                          P. Vixie
Request for Comments: 2671                                          ISC
Category: Standards Track                                   August 1999


                 Extension Mechanisms for DNS (EDNS0)

...

Abstract

   The Domain Name System's wire protocol includes a number of fixed
   fields whose range has been or soon will be exhausted and does not
   allow clients to advertise their capabilities to servers.  This
   document describes backward compatible mechanisms for allowing the
   protocol to grow.
```

Obsoleted by RFC6891 (2013).

- Extends RCODE range and number of flags.
- Mechanism to allow larger UDP messages. This is necessary because of a increase in DNS RR sizes:
  - AAAA records
  - Large TXT records
  - DNSSEC

A pseudo-RR generated on the fly.

NAME		Always 00 (root)
TYPE	16 bits	OPT (41)
CLASS	16 bits	Sender UDP payload size
TTL	32 bits	uint8 extended RCODE uint8 version (0) uint16 flags
RDLEN	16 bits	Length of RDATA
RDATA		Options. Any number of: uint16 Option Code uint16 Option length Option data

Attack!

Recursive server cache poisoning

How does a client match a response to a query it sent?

Recursive server cache poisoning

How does a client match a response to a query it sent?

- Response arrives on the same UDP port

Recursive server cache poisoning

How does a client match a response to a query it sent?

- Response arrives on the same UDP port
- Question section in response matches Question in pending query

Recursive server cache poisoning

How does a client match a response to a query it sent?

- Response arrives on the same UDP port
- Question section in response matches Question in pending query
- Query ID matches the pending query

Recursive server cache poisoning

How does a client match a response to a query it sent?

- Response arrives on the same UDP port
- Question section in response matches Question in pending query
- Query ID matches the pending query
- Authority and Additional sections contain names in same domain as question ("bailiwick checking")

So, how hard is it to fake a response?

So, how hard is it to fake a response?

First good answer wins!

1. Send query for my.bank.com to victim nameserver

1. Send query for my.bank.com to victim nameserver
2. Victim nameserver will be asking ns.bank.com for address

1. Send query for my.bank.com to victim nameserver
2. Victim nameserver will be asking ns.bank.com for address
3. Flood victim with forged replies giving answer as your server

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

1. Request random name in target domain, e.g. abfgshgxy.bank.com

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

1. Request random name in target domain, e.g. abfgshgxy.bank.com
2. Send stream of forged answers to victim, but use NS records in Authority section to delegate to ANOTHER name server

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

1. Request random name in target domain, e.g. abfgshgxy.bank.com
2. Send stream of forged answers to victim, but use NS records in Authority section to delegate to ANOTHER name server
3. Can use real nameserver names but use glue records to fake address of server

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

1. Request random name in target domain, e.g. abfgshgxy.bank.com
2. Send stream of forged answers to victim, but use NS records in Authority section to delegate to ANOTHER name server
3. Can use real nameserver names but use glue records to fake address of server
4. If you match the Query ID, you now own the ZONE

Kaminsky attack (2008)

Don't poison the A record, poison the authority instead

(Thanks to UnixWiz.net)

1. Request random name in target domain, e.g. abfgshgxy.bank.com
2. Send stream of forged answers to victim, but use NS records in Authority section to delegate to ANOTHER name server
3. Can use real nameserver names but use glue records to fake address of server
4. If you match the Query ID, you now own the ZONE
5. Do this for a LOT of different hostnames

DNSSEC to the rescue?

Network Working Group
Request for Comments: 4033
Obsoletes: 2535, 3008, 3090, 3445, 3655, 3658, 3755, 3757, 3845
Updates: 1034, 1035, 2136, 2181, 2308, 3225, 3007, 3597, 3226
Category: Standards Track

R. Arends
Telematica Instituut
R. Austein
ISC
M. Larson
VeriSign
D. Massey
Colorado State University
S. Rose
NIST

March 2005

DNS Security Introduction and Requirements

....

The Domain Name System Security Extensions (DNSSEC) add data origin authentication and data integrity to the Domain Name System. This document introduces these extensions and describes their capabilities and limitations. This document also discusses the services that the DNS security extensions do and do not provide. Last, this document describes the interrelationships between the documents that collectively describe DNSSEC.

DNSSEC

DNSSEC

- Assures Authenticity of DNS data

DNSSEC

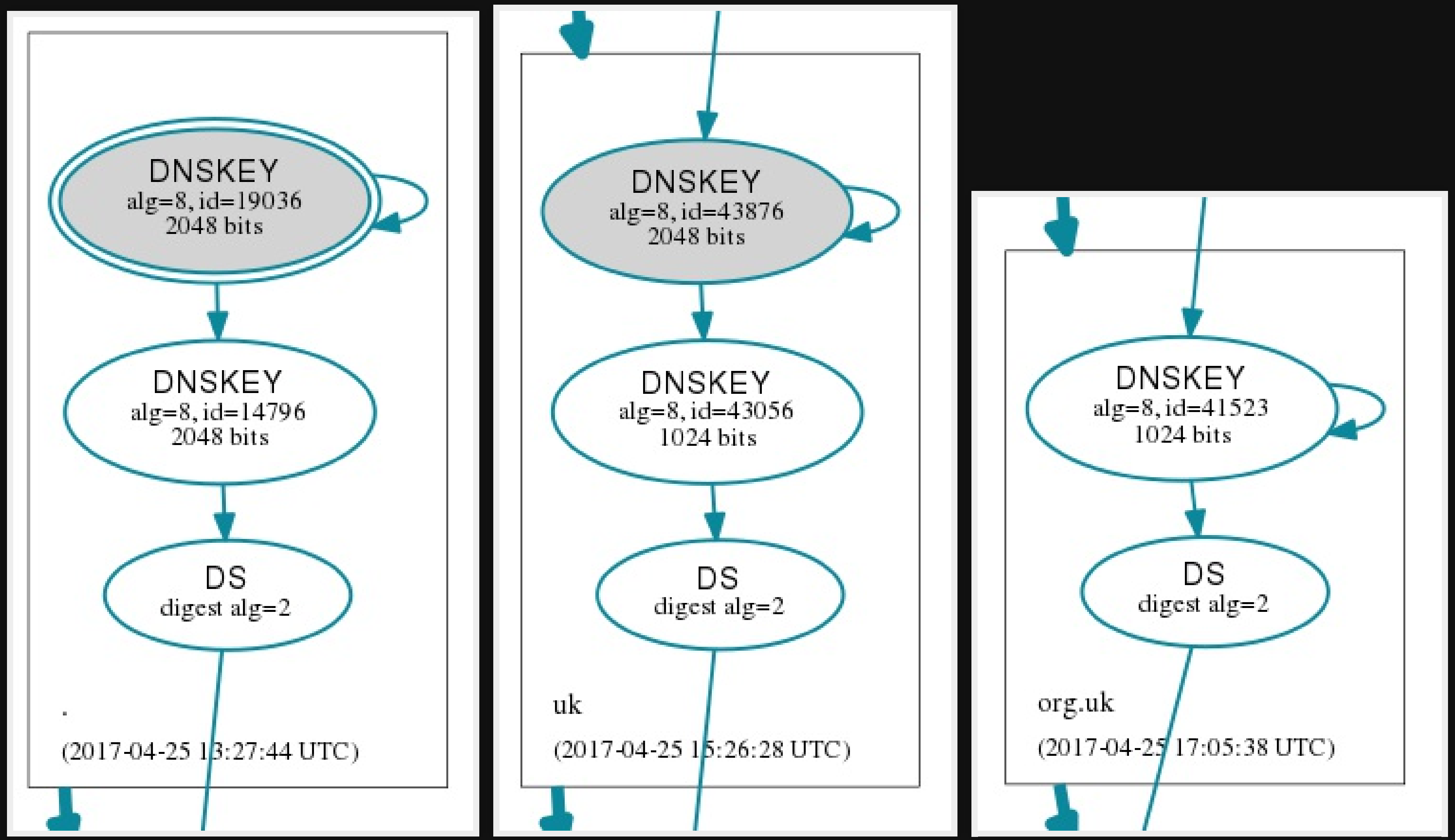
- Assures Authenticity of DNS data
- Assures Integrity of DNS data

DNSSEC

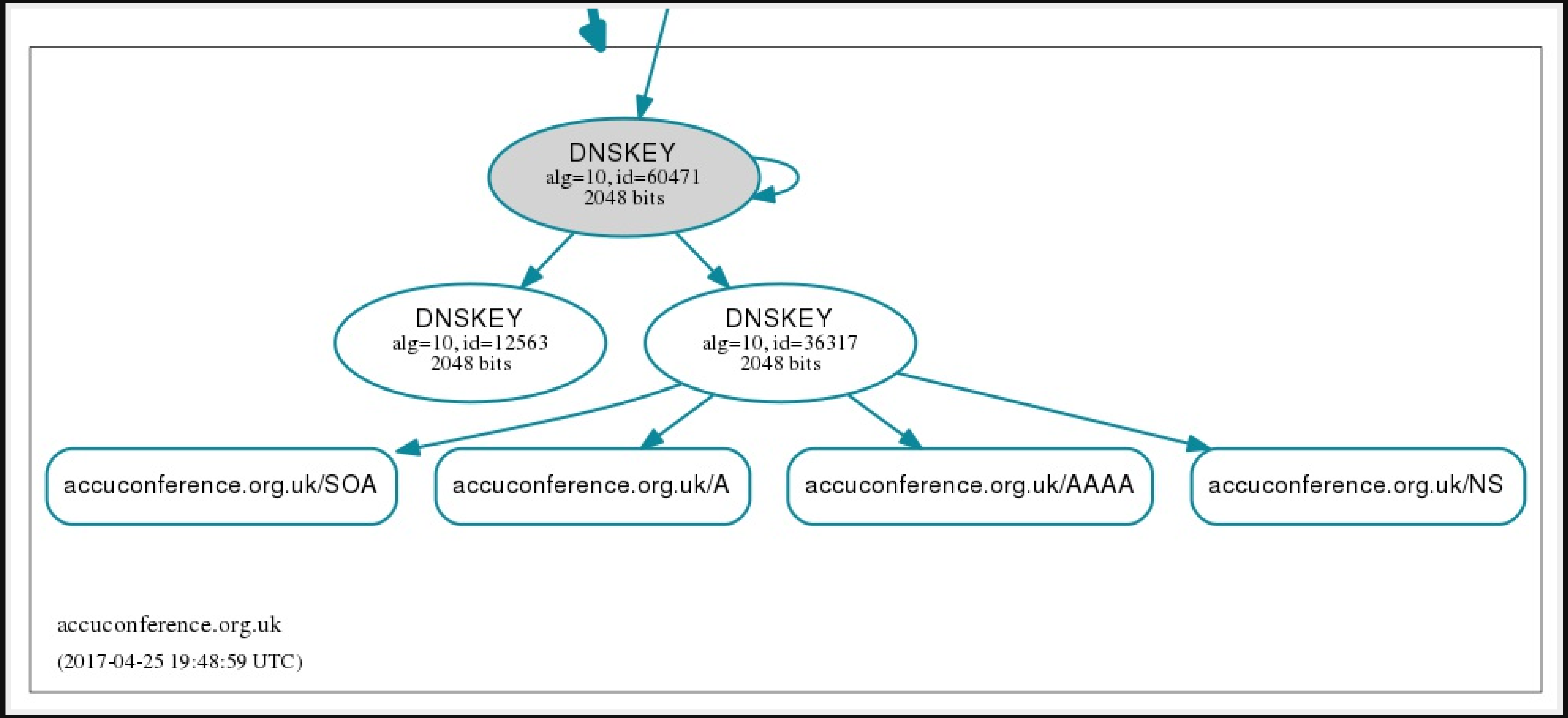
- Assures Authenticity of DNS data
- Assures Integrity of DNS data
  - Note it authenticates DNS data, not DNS servers

DNSSEC

- Assures Authenticity of DNS data
- Assures Integrity of DNS data
  - Note it authenticates DNS data, not DNS servers
- Does NOT ensure confidentiality


DNSKEY
alg=8, id=19036
2048 bits
DNSKEY
alg=8, id=14796
2048 bits
DS
digest alg=2
.
(2017-04-25 13:27:44 UTC)
DNSKEY
alg=8, id=43876
2048 bits
DNSKEY
alg=8, id=43056
1024 bits
DS
digest alg=2
uk
(2017-04-25 15:26:28 UTC)
DNSKEY
alg=8, id=41523
1024 bits
DS
digest alg=2
org.uk
(2017-04-25 17:05:38 UTC)

dnsviz.net


DNSKEY
alg=10, id=60471
2048 bits
DNSKEY
alg=10, id=12563
2048 bits
DNSKEY
alg=10, id=36317
2048 bits
accuconference.org.uk/SOA
accuconference.org.uk/A
accuconference.org.uk/AAAA
accuconference.org.uk/NS
accuconference.org.uk
(2017-04-25 19:48:59 UTC)

dnsviz.net

New DNSSEC RRs

- DNSKEY. A public key.
- RRSIG. Signature of RR sets.
- NSEC/NSEC3. Name existance.
- DS. Digest of DNSKEY record on parental side of delegation.

Back to the wire

- EDNS0 flag D0. Client groks DNSSEC.
- New main flags:
  - Authenticated Data (AD). Data is authenticated.
  - Checking Disabled (CD). Client is OK to receive non-authenticated data.

Using DNSSEC

- If your resolver does DNSSEC:
  - AD indicates data is authenticated
  - SERVFAIL if authentication fails

Last mile problem

Last mile problem

- Can your stub resolver validate?

Last mile problem

- Can your stub resolver validate?
- Can your resolving server validate?

Last mile problem

- Can your stub resolver validate?
- Can your resolving server validate?
- … and even if it can, can you trust the link between you and the resolving server?

Local validation

- DNSSEC-trigger
  https://www.nlnetlabs.nl/projects/dnssec-trigger/
- GetDNS Stubby
  https://getdnsapi.net/blog/dns-privacy-daemon-stubby/

Terminal window time again!

DNSSEC as Public Key Infrastructure

- IPSec keys (RFC4025)
- SSH host keys (RFC4255)
- Storing Certificates, CERT RR (RFC4398)
- DKIM keys (RFC4871)
- CA Authorisation (RFC6844)
- DNS Authentication of Named Entities (DANE), X.509 for TLS (RFC6698,7671)
- OpenPGP key (RFC7929)

Terminal window time again!

Using from your application

Using from your application

- DON'T write your own resolver

Using from your application

- DON'T write your own resolver
- May never go into standard C libraries

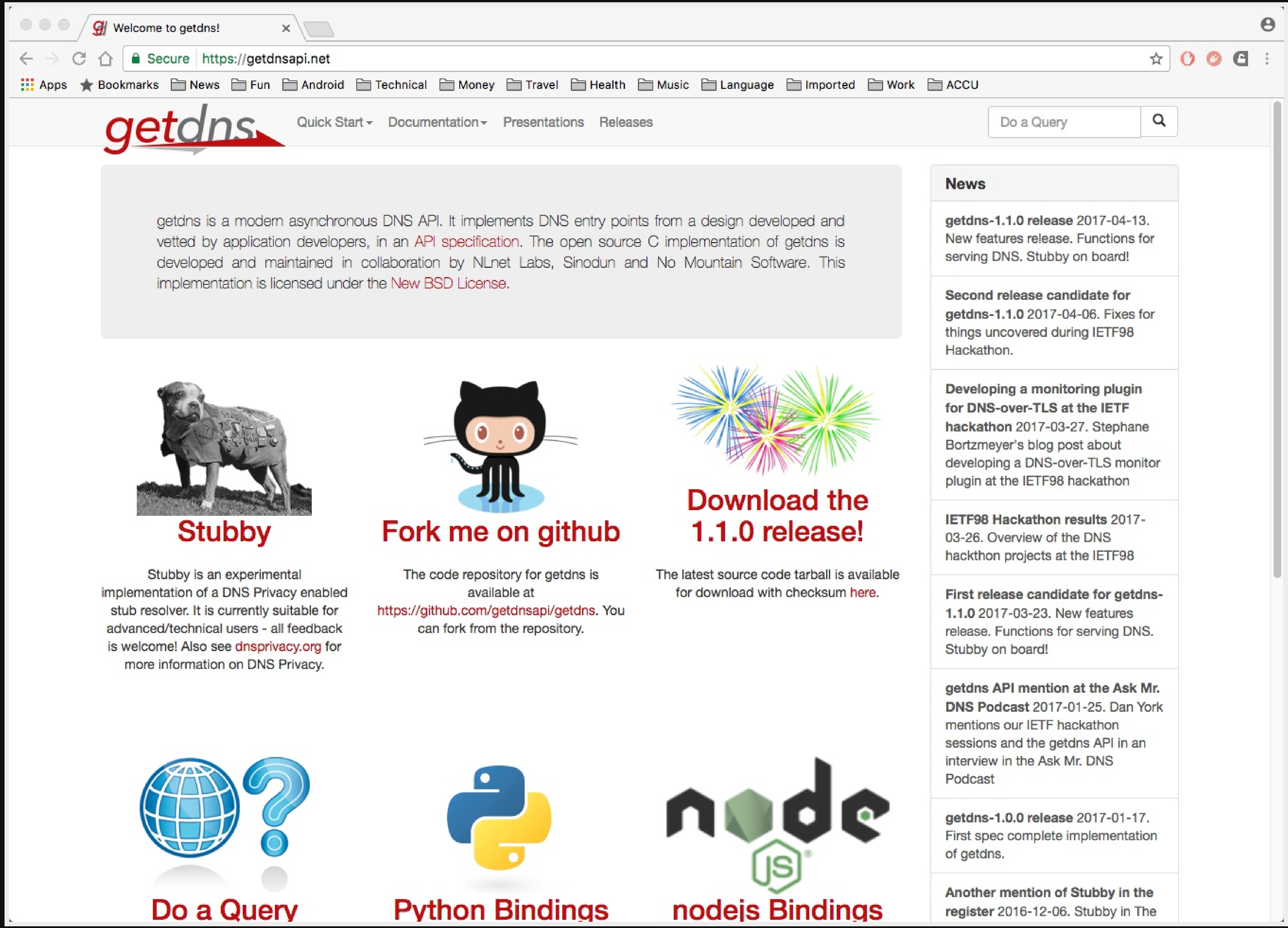
Using from your application

- DON'T write your own resolver
- May never go into standard C libraries
- Time for a library...

GetDNS

https://getdnsapi.net/

getdns


Welcome to getdns!
Secure https://getdnsapi.net
Apps Bookmarks News Fun Android Technical Money Travel Health Music Language Imported Work ACCU
getdns
Quick Start Documentation Presentations Releases
Do a Query
getdns is a modern asynchronous DNS API. It implements DNS entry points from a design developed and vetted by application developers, in an API specification. The open source C implementation of getdns is developed and maintained in collaboration by NLnet Labs, Sinodun and No Mountain Software. This implementation is licensed under the New BSD License.
Stubby
Stubby is an experimental implementation of a DNS Privacy enabled stub resolver. It is currently suitable for advanced/technical users - all feedback is welcome! Also see dnsprivacy.org for more information on DNS Privacy.
Fork me on github
The code repository for getdns is available at https://github.com/getdnsapi/getdns. You can fork from the repository.
Download the 1.1.0 release!
The latest source code tarball is available for download with checksum here.
Do a Query
Python Bindings
nodejs Bindings
News
getdns-1.1.0 release 2017-04-13. New features release. Functions for serving DNS. Stubby on board!
Second release candidate for getdns-1.1.0 2017-04-06. Fixes for things uncovered during IETF98 Hackathon.
Developing a monitoring plugin for DNS-over-TLS at the IETF hackathon 2017-03-27. Stephane Bortzmeyer's blog post about developing a DNS-over-TLS monitor plugin at the IETF98 hackathon
IETF98 Hackathon results 2017-03-26. Overview of the DNS hackthon projects at the IETF98
First release candidate for getdns-1.1.0 2017-03-23. New features release. Functions for serving DNS. Stubby on board!
getdns API mention at the Ask Mr. DNS Podcast 2017-01-25. Dan York mentions our IETF hackathon sessions and the getdns API in an interview in the Ask Mr. DNS Podcast
getdns-1.0.0 release 2017-01-17. First spec complete implementation of getdns.
Another mention of Stubby in the register 2016-12-06. Stubby in The

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

- Stub and full recursive operation

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

- Stub and full recursive operation
- Supports all RR types

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

- Stub and full recursive operation
- Supports all RR types
- Fine-grained access to response

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

- Stub and full recursive operation
- Supports all RR types
- Fine-grained access to response
- DNSSEC validation, even in stub mode

GetDNS overview

Collaboration between VeriSign Labs, NLNetlabs, No Mountain, Sinodun.

- Stub and full recursive operation
- Supports all RR types
- Fine-grained access to response
- DNSSEC validation, even in stub mode
- … and more

GetDNS overview (cont.)

GetDNS overview (cont.)

- C with bindings for Python, NodeJS and others

GetDNS overview (cont.)

- C with bindings for Python, NodeJS and others
- Only dependency is OpenSSL

GetDNS overview (cont.)

- C with bindings for Python, NodeJS and others
- Only dependency is OpenSSL
- Asynchronous by default

GetDNS overview (cont.)

- C with bindings for Python, NodeJS and others
- Only dependency is OpenSSL
- Asynchronous by default
- … with support for libevent, libev, libuv and custom event loops

GetDNS overview (cont.)

- C with bindings for Python, NodeJS and others
- Only dependency is OpenSSL
- Asynchronous by default
- … with support for libevent, libev, libuv and custom event loops
- JSON dict-like output with C support functions

Core API

`getdns_address`	`getaddrinfo()`-like address lookups
`getdns_hostname`	`getnameinfo()`-like name lookups
`getdns_service`	results from SRV lookups
`getdns_general`	looking up any type of DNS record

```
{     # This is the response object
  "replies_full": [ (bindata of the first response),
                    (bindata of the second response) ],
  "just_address_answers":
  [
    {
      "address_type": (bindata of "IPv4"),
      "address_data": (bindata of 0x0a0b0c01),
    },
    {
      "address_type": (bindata of "IPv6"),
      "address_data": (bindata of 0x33445566334455663344556633445566)
    }
  ],
  "canonical_name": (bindata of "www.example.com"),
  "answer_type": GETDNS_NAMETYPE_DNS,
  "intermediate_aliases": [],
  "replies_tree":
  [
```

```
{     # This is the first reply
  "header": { "id": 23456, "qr": 1, "opcode": 0, ... },
  "question": { "qname": (bindata of "www.example.com"),
                "qtype": 1, "qclass": 1 },
  "answer":
  [
    {
      "name": (bindata of "www.example.com"),
      "type": 1,
      "class": 1,
      "ttl": 33000,
      "rdata":
      {
        "ipv4_address": (bindata of 0x0a0b0c01)
        "rdata_raw": (bindata of 0x0a0b0c01)
      }
    }
  ],
  "authority":
  [
    {
      "name": (bindata of "ns1.example.com"),
      "type": 1,
      "class": 1,
      "ttl": 600,
      "rdata":
      {
```

```
    ]
    "additional": [],
    "canonical_name": (bindata of "www.example.com"),
    "answer_type": GETDNS_NAMETYPE_DNS
  },
  {     # This is the second reply
    "header": { "id": 47809, "qr": 1, "opcode": 0, ... },
    "question": { "qname": (bindata of "www.example.com"),
                  "qtype": 28, "qclass": 1 },
    "answer":
    [
      {
        "name": (bindata of "www.example.com"),
        "type": 28,
        "class": 1,
        "ttl": 1000,
        "rdata":
        {
          "ipv6_address": (bindata of 0x33445566334455663344556633445566)
          "rdata_raw": (bindata of 0x33445566334455663344556633445566)
        }
     }
    ],
    "authority": [  # Same as for other record... ]
    "additional": [],
  },
]
```

```c
getdns_return_t  r; /* Holder for all function returns */
getdns_context  *context     = NULL;
getdns_dict     *response    = NULL;
getdns_dict     *extensions = NULL;
getdns_bindata  *address_data;
char             *first = NULL, *second = NULL;

/* Create the DNS context for this call */
if ((r = getdns_context_create(&context, 1)))
    fprintf(stderr, "Trying to create the context failed");

else if (!(extensions = getdns_dict_create()))
    fprintf(stderr, "Could not create extensions dict.\n");

else if ((r = getdns_dict_set_int(extensions, "return_both_v4_and_v6",
                                  GETDNS_EXTENSION_TRUE)))
    fprintf(stderr, "Setting an extension do both IPv4 and IPv6 failed");

else if ((r = getdns_general_sync(context, "example.com",
                                  GETDNS_RRTYPE_A, extensions, &response)))
    fprintf(stderr, "Error scheduling synchronous request");
```

```c
else if ((r = getdns_dict_get_bindata(response,
                                        "/just_address_answers/0/address_data",
                                        &address_data)))
    fprintf(stderr, "Could not get first address");

else if (!(first = getdns_display_ip_address(address_data)))
    fprintf(stderr, "Could not convert first address to string\n");

else if ((r = getdns_dict_get_bindata(response,
                                        "/just_address_answers/1/address_data",
                                        &address_data)))
    fprintf(stderr, "Could not get second address");

else if (!(second = getdns_display_ip_address(address_data)))
    fprintf(stderr, "Could not convert second address to string\n");

if (first) {
    printf("The address is %s\n", first);
    free(first);
}
if (second) {
    printf("The address is %s\n", second);
    free(second);
}
```

```c
/* Clean up */
if (response)
    getdns_dict_destroy(response);

if (extensions)
    getdns_dict_destroy(extensions);

if (context)
    getdns_context_destroy(context);

if (r) {
    assert( r != GETDNS_RETURN_GOOD );
    fprintf(stderr, ": %d\n", r);
    exit(EXIT_FAILURE);
}
```

DNS Privacy

Edward Snowden

Internet Architecture Board (IAB)
Request for Comments: 6973
Category: Informational
ISSN: 2070-1721

A. Cooper
CDT
H. Tschofenig
Nokia Siemens Networks
B. Aboba
Skype
J. Peterson
NeuStar, Inc.
J. Morris

M. Hansen
ULD
R. Smith
Janet
July 2013

Privacy Considerations for Internet Protocols

Abstract

This document offers guidance for developing privacy considerations for inclusion in protocol specifications. It aims to make designers, implementers, and users of Internet protocols aware of privacy-related design choices. It suggests that whether any individual RFC warrants a specific privacy considerations section will depend on the document's content.

Internet Engineering Task Force (IETF)
Request for Comments: 7258
BCP: 188
Category: Best Current Practice
ISSN: 2070-1721
S. Farrell
Trinity College Dublin
H. Tschofenig
ARM Ltd.
May 2014

Pervasive Monitoring Is an Attack

Abstract

Pervasive monitoring is a technical attack that should be mitigated in the design of IETF protocols, where possible.

....

The IETF community's technical assessment is that PM is an attack on the privacy of Internet users and organisations. The IETF community has expressed strong agreement that PM is an attack that needs to be mitigated where possible, via the design of protocols that make PM significantly more expensive or infeasible.

Internet Architecture Board (IAB)
Request for Comments: 7624
Category: Informational
ISSN: 2070-1721

R. Barnes
B. Schneier
C. Jennings
T. Hardie
B. Trammell
C. Huitema
D. Borkmann

August 2015

Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement

Abstract

Since the initial revelations of pervasive surveillance in 2013, several classes of attacks on Internet communications have been discovered. In this document, we develop a threat model that describes these attacks on Internet confidentiality. We assume an attacker that is interested in undetected, indiscriminate eavesdropping. The threat model is based on published, verified attacks.

- Full domain name potentially sent to all authoritative servers from root down

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?
  - User certificates in DNSSEC

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?
  - User certificates in DNSSEC
  - Service discovery SRV records often contain a hostname e.g. Alice's Mobile

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?
  - User certificates in DNSSEC
  - Service discovery SRV records often contain a hostname e.g. Alice's Mobile
  - EDNS0 options

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?
  - User certificates in DNSSEC
  - Service discovery SRV records often contain a hostname e.g. Alice's Mobile
  - EDNS0 options
    - Parental filters and user information
    - CDRs and geographical info
    - Client equipment can add MAC and originating subnet

- Full domain name potentially sent to all authoritative servers from root down
- But it's not sensitive, is it?
  - User certificates in DNSSEC
  - Service discovery SRV records often contain a hostname e.g. Alice's Mobile
  - EDNS0 options
    - Parental filters and user information
    - CDRs and geographical info
    - Client equipment can add MAC and originating subnet
- Meta-data leaking allows re-identification of individuals

Internet Engineering Task Force (IETF)
Request for Comments: 7626
Category: Informational
ISSN: 2070-1721
S. Bortzmeyer
AFNIC
August 2015

DNS Privacy Considerations

Abstract

This document describes the privacy issues associated with the use of the DNS by Internet users. It is intended to be an analysis of the present situation and does not prescribe solutions.

Current standards activity

Various solutions proposed.

- DNS with STARTTLS
- DNS over DTLS to new port
- DNS over TLS to new port (853)

Current standards activity

Various solutions proposed.

- DNS with STARTTLS
- DNS over DTLS to new port
- DNS over TLS to new port (853)
  - Supported in GetDNS

Summary

Summary

- DNSSEC is arriving. Slowly.

Summary

- DNSSEC is arriving. Slowly.
- Issues remain with key length and IoT

Summary

- DNSSEC is arriving. Slowly.
- Issues remain with key length and IoT
- Lots of activity in DNS Privacy space

Summary

- DNSSEC is arriving. Slowly.
- Issues remain with key length and IoT
- Lots of activity in DNS Privacy space
- DNS-over-TLS can be used today if you need it

Sucesss is stumbling from failure to failure with no loss of enthusiasm
-- Winston Churchill