

CS 5523 Lecture 27: DNS Implementation

- Questions on Laboratory 4
- Practical issues in name resolution
- Name resolution from the viewpoint of the user
- Representation of naming information in DNS
- Resolver library and low-level DNS queries

Practical issues in name resolution:

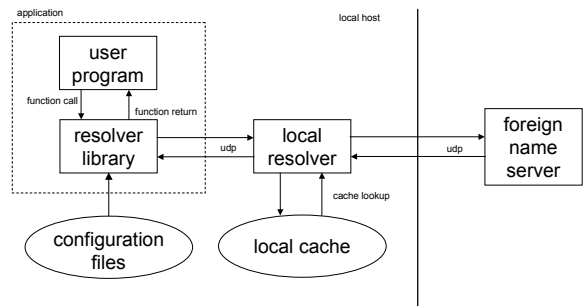
- Name resolution is needed in most communication because users usually specify hosts by name rather than IP address.
- `/etc/resolv.conf` specifies IP addresses of the name servers on most Unix systems.
- A single name may require many inquiries to be resolved.
- Local shared caching can considerably reduce network traffic.
- The name servers must be relatively reliable, or the network won't work.

DNS queries:

- *host resolution* - resolves host names into IP addresses
- *mail host location* - resolves domain names into IP addresses of mail hosts
- *reverse resolution* - returns a host name given an IP address
- *host information* - given a host name, return information about the host
- *well-known services* - return a list of services given a hostname

Give an example of a use of each of these types of queries.

Typical name resolution in a user program:



The `/etc/resolv.conf` file for medusa

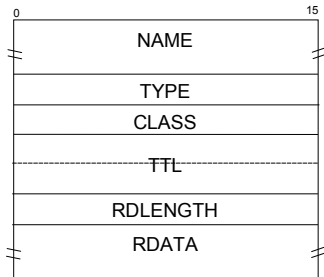
```
domain cs.utsa.edu
search cs.utsa.edu atm.utsa.edu utsa.edu
nameserver 129.115.11.19
nameserver 129.115.102.150
nameserver 128.83.139.9
```

- *domain* – indicates the local default domain name for resolving short names
- *search* – list to search for hostname lookup
- *nameserver* – foreign name servers to be queried in listed order

DNS databases:

- A domain name identifies a node.
- Each information at each node consists of resource records.
- Each resource record (RR) has:
 - owner (domain name)
 - type (A, CNAME, HINFO, MX, NS, PTR, SOA)
 - class (IN = internet)
 - TTL
 - RDATA (describes the information above)

DNS resource records (RR):



DNS name format:

- A domain name is a sequence of labels.
- A label is 1 byte length followed by that number of bytes.
- A domain name ends with a null byte (a zero-length label).
- The labels are printed separated by dots.
- Most DNS comparisons are case insensitive

Example of a DNS name:

2 c s 4 u t s a 3 e d u 0

Sizes defined in `arpa/nameser.h`:

```
/*
 * Define constants based on RFC 883, RFC 1034, RFC 1035
 */
#define NS_PACKETSZ 512 /* maximum packet size */
#define NS_MAXCDNAME 1025 /* maximum domain name */
#define NS_MAXCNAME 255 /* maximum compressed domain name */
#define NS_MAXLABEL 63 /* maximum length of domain label */
#define NS_FIXEDDNAME 12 /* #bytes of fixed data in header */
#define NS_QFIXEDSZ 4 /* #bytes of fixed data in query */
#define NS_RFIXEDSZ 10 /* #bytes of fixed data in r record */
#define NS_INFIXEDSZ 4 /* #bytes of data in a u_int32_t */
#define NS_INTFIXEDSZ 2 /* #bytes of data in a u_int16_t */
#define NS_INPSSZ 1 /* #bytes of data in a u_int8_t */
#define NS_INADDRSZ 4 /* IPv4 T_A */
#define NS_IN6ADDRSZ 16 /* IPv6 T_AAAA */
#define NS_CMPRSFLGS 0x00 /* Flag bits indicating name compression. */
#define NS_DEFAULTPORT 53 /* For both TCP and UDP. */
```

Where would you look for the file `arpa/nameser.h`?

How would you tell the compiler to look somewhere else?

Types defined in `arpa/nameser.h`:

```
typedef enum __ns_type {
    ns_t_a = 1, /* Host address */
    ns_t_ns = 2, /* Authoritative server */
    ns_t_md = 3, /* Mail destination */
    ns_t_mf = 4, /* Mail forwarded */
    ns_t_cname = 5, /* Canonical name */
    ns_t_soa = 6, /* Start of authority zone */
    ns_t_mb = 7, /* Mailbox domain name */
    ns_t_mg = 8, /* Mail group member */
    ns_t_mr = 9, /* Mail rename name */
    ns_t_null = 10, /* Null resource record */
    ns_t_wks = 11, /* Well known service */
    ns_t_ptr = 12, /* Domain name pointer */
    ns_t_hinfo = 13, /* Host information */
    ns_t_minfo = 14, /* Mailbox information */
    ns_t_mx = 15, /* Mail routing information */
    ns_t_txt = 16, /* Text strings */
    ns_t_rp = 17, /* Responsible person */
    ns_t_afdb = 18, /* AFS cell database */
    ns_t_nis = 19, /* X.25 calling address */
    ns_t_isdn = 20, /* ISDN calling address */
    ns_t_rt = 21, /* Router */
    ns_t_nsap = 22, /* NSAP address */
    ns_t_rt = 21, /* Router */
    ns_t_nsap = 22, /* NSAP address */
    ns_t_nsap_ptr = 23, /* Reverse NSAP lookup
    /* (deprecated) */
    ns_t_sig = 24, /* Security signature */
    ns_t_key = 25, /* Security key */
    ns_t_px = 26, /* X.400 mail mapping */
    ns_t_gpos = 27, /* Geographical position */
    /* (withdrawn) */
    ns_t_aaaa = 28, /* IPv6 Address */
    ns_t_loc = 29, /* Location Information */
    ns_t_next = 30, /* Next domain (security) */
    ns_t_eid = 31, /* Endpoint identifier */
    ns_t_nimloc = 32, /* Nimrod Locator */
    ns_t_srv = 33, /* Server Selection */
    ns_t_atma = 34, /* ATM Address */
    ns_t_naptr = 35, /* Naming Authority Pointer
    /* (deprecated) */
    /* Query type values not in resource records */
    ns_t_ixfr = 251, /* Incremental zone transfer */
    ns_t_axfr = 252, /* Transfer zone of authority */
    ns_t_mailb = 253, /* Transfer mailbox records */
    ns_t_maila = 254, /* Transfer mail agent records
    /* (deprecated) */
    ns_t_any = 255, /* Wildcard match */
    ns_t_max = 65536
} ns_type;
```

Class field definitions in `arpa/nameser.h`:

```
/*
 * Values for class field
 */
typedef enum __ns_class {
    ns_c_in = 1, /* Internet. */
    ns_c_chaos = 3, /* MIT Chaos-net. */
    ns_c_hs = 4, /* MIT Hesiod. */
    /* Query class values which do not appear in resource records */
    ns_c_name = 254, /* for prereq. sections in update requests */
    ns_c_any = 255, /* Wildcard match. */
    ns_c_max = 65536
} ns_class;
```

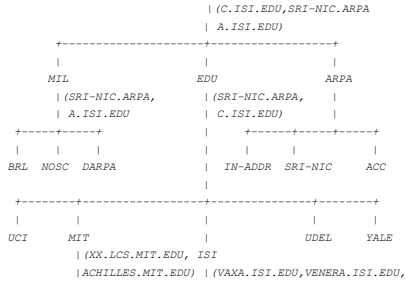
Zone partitioning of the DNS name space:

- zone - contains attribute data for names in domain minus the sub-domains administrated by lower-level authorities:

Example: UTSA has a name server for `utsa.edu`, but `cs.utsa.edu` names are resolved by the CS Division server

- at least two name servers that provide authoritative data for the zone
- names of the servers for the sub-domains
- zone management parameters

Zone partitioning of the DNS name space:



from RCF 1034: Domain Concepts and Facilities by Mockpetris

Authoritative name servers:

- a server may be an authoritative source for zero or more zones
- data for a zone is entered into a local master file
- the master (primary) server reads the zone data directly from the master file
- secondary authoritative servers download zone data from primary server
- secondary servers periodically check their version number against the master server

Sample query of the root server :

```

.      IN      SOA      SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
      870611      ;serial
      1800      ;refresh every 30 min
      300      ;retry every 5 min
      604800      ;expire after a week
      86400      ;minimum of a day
      NS      A.ISI.EDU
      NS      C.ISI.EDU
      NS      SRI-NIC.ARPA.
MIL.  86400  NS      SRI-NIC.ARPA.
      86400  NS      A.ISI.EDU.
EDU.  86400  NS      SRI-NIC.ARPA.
      86400  NS      C.ISI.EDU.
      ...Continued from previous column
      USC-ISIC.ARPA. CNAME C.ISI.EDU.
SRI-NIC.ARPA. A      26.0.0.73      73.0.0.26.IN-ADDR.ARPA. PTR      SRI-NIC.ARPA.
      A      10.0.0.51      65.0.6.26.IN-ADDR.ARPA. PTR      ACC.ARPA.
      MX      0 SRI-NIC.ARPA. 51.0.0.10.IN-ADDR.ARPA. PTR      SRI-NIC.ARPA.
      HINFO   DEC-2060 TOPS20 52.0.0.10.IN-ADDR.ARPA. PTR      C.ISI.EDU.
ACC.ARPA.  A      26.6.0.65
      HINFO   POP-11/70 UNIX 103.0.3.26.IN-ADDR.ARPA. PTR      A.ISI.EDU.
      MX      10 ACC.ARPA.  A.ISI.EDU. 86400 A      26.3.0.103
      C.ISI.EDU. 86400 A      10.0.0.52
... Continued in next column
  
```

Sample query of the root server :

```

Header | OPCODE=QUERY
-----+-----
Question | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
-----+-----
Answer | <empty>
-----+-----
Authority | <empty>
-----+-----
Additional | <empty>
-----+-----

response
Header | OPCODE=QUERY, RESPONSE, AA
-----+-----
Question | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
-----+-----
Answer | SRI-NIC.ARPA. 86400 IN A 26.0.0.73
      86400 IN A 10.0.0.51
-----+-----
Authority | <empty>
-----+-----
Additional | <empty>
-----+-----
  
```

query to c.isi.edu

Sample query of the root server :

query to c.isi.edu: QNAME=SRI-NIC.ARPA, QTYPE=*

response:

```

Header | OPCODE=QUERY, RESPONSE, AA
-----+-----
Question | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*
-----+-----
Answer | SRI-NIC.ARPA. 86400 IN A      26.0.0.73
      A      10.0.0.51
      MX      0 SRI-NIC.ARPA.
      HINFO   DEC-2060 TOPS20
-----+-----
Authority | <empty>
-----+-----
Additional | <empty>
-----+-----
  
```

The * wildcard for the type indicates all records.

Sample query of the root server :

query to c.isi.edu: QNAME=BRL.MIL, QTYPE=A

response:

```

Header | OPCODE=QUERY, RESPONSE
-----+-----
Question | QNAME=BRL.MIL, QCLASS=IN, QTYPE=A
-----+-----
Answer | <empty>
-----+-----
Authority | MIL.      86400 IN NS      SRI-NIC.ARPA.
      86400 NS      A.ISI.EDU.
-----+-----
Additional | A.ISI.EDU.      A      26.3.0.103
      SRI-NIC.ARPA.      A      26.0.0.73
      A      10.0.0.51
-----+-----
  
```

response contains the address RRs because cs.isu.edu isn't authoritative for mil and it didn't have BRL.MIL.

Using nstest to query name servers (example 1):

```
visual8% nstest 129.115.102.150
amedusa.cs.utsa.edu
;; res_mkquery(0, medusa.cs.utsa.edu, 1, 1)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46777
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;;      medusa.cs.utsa.edu, type = A, class = IN
;; Querying server (# 1) address = 129.115.102.150
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46777
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0;; medusa.cs.utsa.edu, type = A, class = IN
medusa.cs.utsa.edu.      1D IN A      129.115.11.62
```

Using nstest to query name servers (example 2):

```
nmedusa.cs.utsa.edu
;; res_mkquery(0, medusa.cs.utsa.edu, 1, 2)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46778
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;;      medusa.cs.utsa.edu, type = NS, class = IN
;; Querying server (# 1) address = 129.115.102.150
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46778
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;;      medusa.cs.utsa.edu, type = NS, class = IN
cs.utsa.edu.      1D IN SOA      jazz.cs.utsa.edu. root.jazz.cs.utsa.edu. (
                                424      ; serial
                                3H      ; refresh
                                1H      ; retry
                                1W      ; expiry
                                1D )    ; minimum
```

The resolver library:

```
cc [ flag ... ] file ... -lresolv -lsocket -lnsl [ library ... ]

#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>

int res_mkquery(int op, /* QUERY */
               const char *dname, /* domain */
               int class, /* IN */
               int type, /* A, NS, etc. */
               const char *data, /* RR */
               int datalen, /* length of RR */
               struct rrec *newrr, /* use null */
               uchar_t *buf, /* answer */
               int buflen /* answer length */);
```

The resolver library:

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>

int res_send(uchar_t *msg, /* preformatted msg */
            int msglen,
            uchar_t *answer,
            int anslen);
```

The resolver library:

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>

int res_query(const char *dname,
             int class,
             int type,
             uchar_t *answer,
             int anslen);
```

For next time:

- Read CDK Chapter 9.3-9.5
- Try some examples of nstest before the next class