System Administration

DNS
Topics

1. DNS Tutorial *
2. Configuring DNS in BIND
3. Live queries using ‘dig’

* - based on slides from conference
  - [DNS and DNSSEC, LOPSA PICC 12]
DNS

- Domain Name System
  - Distributed global database
  - Indexed by “domain names” (together with a type and class)
  - A domain name is a sequence of labels, eg.
    - sa.cs.ut.ee
  - Domain Names are case insensitive, but case preserving
  - Transport protocol: UDP and TCP port 53
Zone google.com

Zone ut.ee

Zone ee

Estonian Country zone

Root Zone

com
org
net
arpa

google

ietf

in-addr

ip6

128

130

91

ww
imap

www

Zone cs.ut.ee

courses

sa

ut
DNS

- DNS can be represented as a tree of labels
- Sibling nodes must have unique labels
- Domain name at a particular label can be formed by the sequence of labels traversed by walking up the tree from that label to the root
- Zone - autonomously managed sub-tree
- Delegations: boundaries between zones
Root and TLDs

- Root of the DNS ("empty label")
- Next level of names are called Top Level Domains (TLDs)
- Until recently 3 primary classes of TLDs
  - GTLD: Generic Top Level Domains (.com, .net, .edu, .org etc)
  - CCTLD: Country Code TLD (2 letter codes for each country, eg. .us, .fr, .jp, .de, ...)
  - Infrastructure: eg. .arpa etc (uses: reverse DNS e164, etc)
DNS main components

• Server Side:
  – Authoritative Servers
  – Resolvers (Recursive Resolvers)

• Client Side:
  – Stub resolvers (usually on DNS client machines)
Authoritative Server

- A server that directly serves data for a particular zone
- Said to be “authoritative” for that zone
- These servers are the ones specified in NS records
Resolver

• Aka “Recursive Resolver”, “Cache” etc
  – Used by endsystems (stub resolvers) to query (“resolve”) arbitrary domain names
  – Receives “recursive” queries from these endsystems
  – Resolvers query authoritative servers, following DNS delegations until they obtain the answer they need (this process is called “iterative” resolution)
  – Resolvers “cache” (remember) query results for the specified “TTL” (also some negative results are cached)
Stub Resolver

- The DNS client software component that resides on most endsystems
- Commonly implemented by the Operating System as a set of library routines
- Has a configured set of addresses of the Recursive Resolvers that should be used to lookup ("resolve") domain names
  - usually by manual configuration, or dynamically learned via DHCP
Stub resolver configuration

- `devel@T72:~$ cat /etc/resolv.conf`

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by
resolvconf(8)
#    DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.10.244
nameserver 193.40.5.39
nameserver 193.40.5.76
search mt.ut.ee at.mt.ut.ee
```
Endstation (using STUB resolver aka. /etc/resolv.conf)
Parts of a DNS query

• Each DNS query needs a query name, type, and class
  – qname: a domain name, eg. www.ut.ee
  – qtype: A, AAAA, MX, CNAME, PTR, SRV, TXT, NS, SOA, …
  – qclass: IN, CH, HS (only “IN” is commonly used)
  – Various flags: QR, RD, EDNS Opt, DO etc
Life of a typical DNS query

- Type “www.ut.ee” into browser
  - Browser calls a name lookup function (eg. getaddrinfo())
  - DNS may not be the only name lookup service in use. The lookup function might consult a nameservice switch table to figure out what order of services to consult (eg. /etc/nsswitch.conf -- flat file, LDAP, NIS, DNS etc)
  - If/when DNS is used, then call DNS specific calls in stub resolver
Life of a typical DNS query

- Stub resolver formulates and makes DNS query:
  - qname www.ut.ee, qtype=A, qclass=IN
    - Note: IPv6 enabled resolvers might try AAAA, then A
  - Sends query to DNS servers (resolvers) specified in stub resolver configuration (eg. /etc/resolv.conf) in the order specified until it gets a successful response, failure, or times out
  - If a “search” domain list is configured, on lookup failure, the stub retries queries with domain suffixes from this list appended to the original quer
Life of a typical DNS query

- DNS resolvers will get the answer:
  - from their authoritative zones if they have any relevant ones
  - from their cache if the answer is already there
  - by iterative queries of the DNS tree, as necessary, eg.
  - root servers, amazon.com servers, ...
Resource Records (RR)

• The fundamental unit of data in the DNS database
  – A grouping of a \{domain name, type, class\}, a TTL (time-to-live), and the associated “resource data”
  – Has a defined text “presentation format”


  name, or

  owner name

  ttl  class  type  rdata
Resource Record Sets

- A set of RRs with the same name, class, and type
- The rdata (resource data) associated with each RR in the set must be distinct
- The TTL of all RRs in the set also must match
- RR sets are treated atomically when returning responses

www.ucla.edu. 300 IN A 169.232.33.224
www.ucla.edu. 300 IN A 169.232.55.224
www.ucla.edu. 300 IN A 169.232.56.224
# Resource Record types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>marks <strong>Start Of a zone of Authority</strong></td>
</tr>
<tr>
<td>NS</td>
<td>NameServer record</td>
</tr>
<tr>
<td>A</td>
<td>IPv4 Address record</td>
</tr>
<tr>
<td>AAAA</td>
<td>IPv6 Address record</td>
</tr>
<tr>
<td>CNAME</td>
<td>Canonical name (ie. an alias)</td>
</tr>
<tr>
<td>MX</td>
<td>Mail Exchanger record</td>
</tr>
<tr>
<td>SRV</td>
<td>Service Location record</td>
</tr>
<tr>
<td>PTR</td>
<td>Pointer (most commonly for reverse DNS)</td>
</tr>
<tr>
<td>TXT</td>
<td>Text record (free form text with no semantics)</td>
</tr>
<tr>
<td>NAPTR</td>
<td>Naming Authority Pointer Record</td>
</tr>
</tbody>
</table>
Other special RRtypes

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSIG</td>
<td>Transaction Signature (RFC 2845)</td>
</tr>
<tr>
<td>TKEY</td>
<td>Transaction Key (RFC 2930) - estab secret keys</td>
</tr>
<tr>
<td>AXFR</td>
<td>Zone Transfer</td>
</tr>
<tr>
<td>IXFR</td>
<td>Incremental Zone Transfer (RFC 1995)</td>
</tr>
<tr>
<td>OPT</td>
<td>Opt pseudo RR (RFC 2671 - EDNS0)</td>
</tr>
</tbody>
</table>
SOA record

- Defines the start of a new zone; and important parameters for the zone
  - Always appears at the apex of the zone
  - Serial number should be incremented on zone content updates

```
  google.com. 86400 IN SOA ns1.google.com. (还不知道)
dns-admin.google.com.
2012042000 ; serial number
7200 ; refresh (2 hours)
1800 ; retry (30 minutes)
1209600 ; expire (2 weeks)
300 ; minimum (5 minutes)
)```

Artjom Lind (artjom.lind@ut.ee)
14.03.2013
https://courses.cs.ut.ee/2013/syshald
**NS record**

- Name Server record: owner is the zone name
- Delegates a DNS subtree from parent (ie. create new zone)
- Lists the authoritative servers for the zone
- Appears in both parent and child zones
- rdata contains hostname of the DNS server
  - `ut.ee. 86400 IN NS ns.ut.ee.`
  - `ut.ee. 86400 IN NS ns2.EENet.ee.`
  - `ut.ee. 86400 IN NS ns2.ut.ee.`
A record

- IPv4 Address Record
- rdata contains an IPv4 address
  - www.example.com. IN A 192.0.43.10
AAAA record

- IPv6 Address Record
- rdata contains an IPv6 address
- Note: there was another record called A6, which didn’t catch on, and which has now been declared historic (RFC 6563)

CNAME record

• An “alias”, ie. maps one name to another (regardless of type)
• Put another way, “this is another name for this name”
• rdata contains the mapped domain name (“canonical name”)

PTR record

- Pointer record
- The most common use is to map IP addresses back to domain names (reverse DNS mappings)
- IPv4 uses in-addr.arpa, and IPv6 uses ip6.arpa subtrees
IPv4 PTR records

• Uses “in-addr.arpa” subtree
• The LHS of the PTR record (“owner name”) is constructed by the following method:
  – Reverse all octets in the IPv4 address
  – Make each octet a DNS label
  – Append “in-addr.arpa.” to the domain name
IPv4 PTR example

host1.example.com. IN A 192.0.2.17

192.0.2.17 (orig IPv4 address)
17.2.0.192 (reverse octets)
17.2.0.192.in-addr.arpa. (append in-addr.arpa.)

Resulting PTR record:

17.2.0.192.in-addr.arpa. IN PTR host1.example.com.
IPv6 addresses

• 128-bits (four times as large)
  – 8 fields of 16 bits each (4 hex digits) separated by colons (:
  – [Hex digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f]
  – $2^{128}$ possible addresses (an incomprehensibly large number)

• 2001:0db8:3902:00c2:0000:0000:0000:fe04

• $(2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456)$
IPv6 addresses

- Zero suppression & compression for more compact format
  - Suppress (omit) leading zeros in each field
  - Replace consecutive fields of all zeros with a double colon (::) - only one sequence of zero fields can be compressed this way

2001:0db8:3902:00c2:0000:0000:0000:fe04

2001:db8:3902:c2::fe04
IPv6 PTR records

- Uses “ip6.arpa” subtree
  - The LHS of the PTR record (“owner name”) is constructed by the following method:
    - Expand all the zeros in the IPv6 address
    - Reverse all the hex digits
    - Make each hex digit a DNS label
    - Append “ip6.arpa.” to the domain name
IPv6 PTR example

host1.example.com. IN AAAA 2001:db8:3902:7b2::fe04

2001:db8:3902:7b2::fe04 (orig IPv6 address)

2001:0db8:3902:07b2:0000:0000:0000:fe04 (expand zeros)

2001:0db8:3902:07b2:0000:0000:0000:fe04 (delete colons)

40ef000000000002b7020938bd01002 (reverse digits)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.o.2.0.9.3.8.b.d.
0.1.0.0.2 (make DNS labels)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.o.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa. (append ip6.arpa.)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.o.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa. IN PTR host1.example.com.
MX record

• Mail Exchanger: defines the host receiving mail
• rdata consists of a preference field and the hostname of the mail receiver
• Lower preference = higher priority

example.com. 86400 IN MX 10 mail1.example.com.
example.com. 86400 IN MX 20 mail2.example.com.
SRV record

- Service Location record (RFC 2782)
- Allows designation of server(s) providing service for a particular application and transport at a domain name
- Owner name has special form: _service._transport.<domain>
- rdata contains priority, weight, port and server hostname
- Some applications using SRV records include: LDAP, Kerberos, XMPP, SIP, Windows AD, ...
SRV record

- Priority defines the order in which to query servers (lower number = higher priority)
- Weight defines the proportion in which to send queries to servers at the same priority level (load distribution)
TXT record

• free form descriptive text strings, with no defined semantics
• Although some applications have defined standardized meanings (eg. DKIM)
• rdata: one or more character strings

blah.example.com. 300 IN TXT “Hello World” “Goodbye”
Wildcards

- RRs with owner names starting with the label "*" (asterisk)
- When the wildcard is matched, the DNS server returns a response with:
  - query name returned as owner name
  - rest of RR content taken from the wildcard record

```
mail.example.com.  300 IN A  10.1.1.1
www.example.com.  300 IN A  10.1.1.2
*.example.com.   300 IN A  10.1.1.7
```

Here, query for blah.example.com returns:
```
blah.example.com.  300 IN A  10.1.1.7
```
Zone file example

Zone: example.com

```plaintext
@ 3600 IN SOA master.example.com. hostmaster.example.com. ( 1001514808 ; serial 10800 ; refresh (3 hours) 3600 ; retry (1 hour) 604800 ; expire (1 week) 3600 ; minimum (1 hour) )

86400 IN NS ns1.example.com.
86400 IN NS ns2.example.com.
86400 IN MX 10 mail1.example.com.
86400 IN MX 20 mail2.example.com.

ns1 86400 IN A 10.1.1.1
ns2 86400 IN A 10.1.1.2
www 900 IN A 10.1.2.2
mail1 3600 IN A 10.3.3.3
mail2 3600 IN A 10.3.3.4
```
Master Zone file format

- @ Denotes current origin; defaulting to zone name. Appended to any domain name not ending in a period.
- () Parents used to group data that crosses a line boundary
- ; Starts a comment
- $ORIGIN Resets the origin for subsequent relative names
- RRs beginning with whitespace implicitly inherit last owner name.
- TTL and Class fields are optional (default to last explicitly stated)
- Extensions usable in BIND master files:
  - $TTL Define TTL parameter for subsequent records
  - $GENERATE Programmatically generate records, eg.
    - $GENERATE 10-90 client-$ A 10.4.4.$
    - $GENERATE 0-62 blah-${0,3,x} A 192.168.154.${+64,0,d}
Size restrictions

- Label: 63 octets max
  - Domain Name: 255 octets max
  - TTL: positive signed 32-bit integer
  - Entire DNS message: 512 bytes (UDP) - plain DNS
  - Messages larger than 512 bytes requires:
    - Use of TCP (often truncated UDP response followed by TCP retry)
    - EDNS0 - a DNS extension mechanism allowing negotiation of larger UDP message buffers
DNS Packet Format

- DNS Header (12 bytes)
  - Question Section
  - Answer Section
  - Authority Section
  - Additional Section
DNS Header

0 08 15

<table>
<thead>
<tr>
<th>QR</th>
<th>OpCode</th>
<th>AA</th>
<th>TC</th>
<th>RD</th>
<th>RA</th>
<th>R</th>
<th>AD</th>
<th>CD</th>
<th>RCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit Query ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QDCOUNT (#records in query)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANCOUNT (#records in answer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSCOUNT (#records in authority)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARCOUNT (#records in additional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DNS Header

- QR: set to 1 in DNS response messages
- OpCode:
  - 0  Standard Query
  - 1  Inverse Query (deprecated)
  - 2  Status request (undefined and unused?)
  - 4  Notify
  - 5  Update
  - 3,6-15 Undefined
DNS Header

- **AA** Authoritative answer (ie. not from cache)
- **TC** message was truncated (exceeded 512 byte UDP limit)
- **RD** Recursion desired
- **RA** Recursion available
- **R** Reserved/Unused
- **AD** Authenticated Data (DNSSEC)
- **CD** Checking Disabled (DNSSEC)
DNS Response Codes

• Common Response codes:
  - 0  NOERROR  No Error
  - 1  FORMERR  Format Error
  - 2  SERVFAIL  Server Failure
  - 3  NXDOMAIN  Not existent domain name
  - 4  NOTIMPL  Function not implemented
  - 5  REFUSED  Query Refused, usually by policy

• Used by DNS Dynamic Update (RFC 2136):
  - 6  YXDomain  Name Exists when it should not
  - 7  YXRRSet  RR Set Exists when it should not
  - 8  NXRRSet  RR Set that should exist does not
  - 9  NotAuth  Server not authoritative for zone
  - 10  NotZone  Name not contained in zone
  - 11-15 Unassigned
```bash
devel@T72:~$ dig @193.40.5.39 www.ut.ee
```

```
; <<>> DiG 9.8.4-rcpz2+rl005.12-P1 <<>> @193.40.5.39 www.ut.ee
; (1 server found)
; global options: +cmd
; Got answer:
; ->>>HEADER<<- opcode: QUERY, status: NOERROR, id: 2812
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 7

;; QUESTION SECTION:
;www.ut.ee. IN A

;; ANSWER SECTION:
www.ut.ee. 86400 IN A 193.40.5.73

;; AUTHORITY SECTION:
ut.ee. 86400 IN NS Sneezy.physic.ut.ee.
ut.ee. 86400 IN NS ns.ut.ee.
ut.ee. 86400 IN NS ns2.ut.ee.
ut.ee. 86400 IN NS ns2.EENet.ee.

;; ADDITIONAL SECTION:
ns.ut.ee. 86400 IN A 193.40.5.99
ns2.ut.ee. 86400 IN A 193.40.5.76
ns2.EENet.ee. 3369 IN A 193.40.0.12
ns2.EENet.ee. 6847 IN AAAA 2001:bb8:2001::12
Sneezy.physic.ut.ee. 86400 IN A 193.40.11.12
Sneezy.physic.ut.ee. 86400 IN AAAA 2001:bb8:2002:b00:207:e9ff:fe04:d222

;; Query time: 2 msec
;; SERVER: 193.40.5.39#53(193.40.5.39)
;; WHEN: Mon Mar 18 14:12:29 2013
;; MSG SIZE rcvd: 278
Non existent domain answer

devel@T72:~$ dig @193.40.5.39 zzz.ut.ee

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> @193.40.5.39 zzz.ut.ee
; (1 server found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 13416
; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
; zzz.ut.ee. IN A

;; AUTHORITY SECTION:
; ut.ee. 86400 IN SOA ns.ut.ee. hostmaster.ns.ut.ee.

Query time: 3 msec
SERVER: 193.40.5.39#53(193.40.5.39)
WHEN: Mon Mar 18 14:17:29 2013
MSG SIZE rcvd: 77