DNS: Domain Name System

Domain Name System:

- Map between symbolic domain name and IP address
- *Distributed database:* implemented in hierarchy of many *name servers*
- *Application-layer protocol:* host, routers, name servers communicate to *resolve* names (address/name translation)
  - Core Internet function implemented as application-layer protocol

DNS name servers

No server has all name-to-IP address mappings

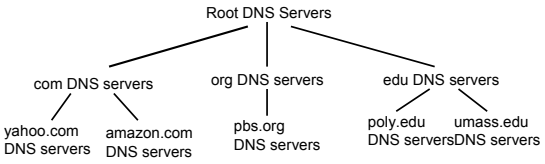
Local name servers (Resolvers):

- Each ISP, company has *local (default) name server*
- Query first goes to local name server

Authoritative name server:

- Authority for a *zone (= domain)*
- Can, e.g., perform name/address translation for a host's name

Distributed, hierarchical database



Example: Client wants IP address for www.ietf.org

- Client queries a root server, response contains .org DNS servers
- Client queries .org DNS server, response contains ietf.org DNS servers
- Client queries ietf.org DNS server, response contains IP address for www.ietf.org

Server types, zones and domains

- Authoritative DNS servers:
  - Responsible for a zone
  - Provide authoritative answers, e.g.,
    - Root servers: On top of DNS hierarchy. Know which servers are responsible for a particular top-level domain
    - Top-level domain (TLD) servers: responsible for com, org, net, edu, …, and all country code top-level domains (ccTLD) de, uk, fr, ca, jp, …
    - Organizations' name servers
- Resolving DNS servers (aka Resolver, "cache")
  - Perform domain name resolution on behalf of a client's stub resolver
  - Very often cache answers

Recursive queries

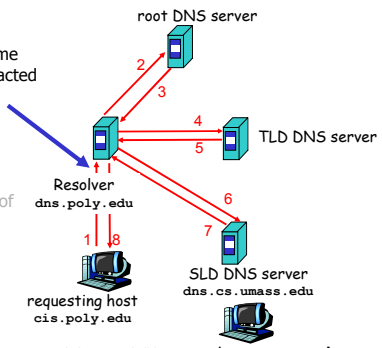
Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server
- To local resolver

Iterative query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Iterative queries

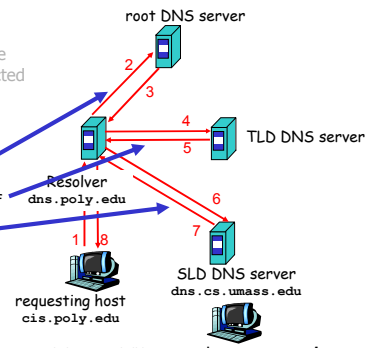
Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- Puts burden of name resolution on contacted name server
- To local resolver

Iterative query:

- Contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



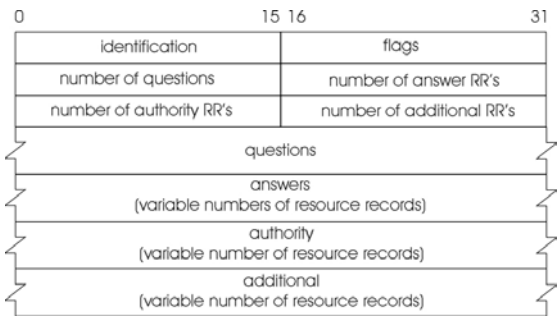
Replication

- Common case: several authoritative servers per zone
  - One primary/master server
  - Many secondaries/backups/slaves/…
  - Slave servers synchronize with master after timeout and notifies
- Caching on resolvers: once a resolving name server learns a mapping, it *caches* mapping
  - Cache entries timeout (disappear) after some time

Inside the DNS Protocol

- Uses UDP Port 53
  (TCP: only for server-to-server traffic or large volumes, and as a fallback)
- Limited Packet Size (about 500 Bytes, can become larger through extensions)
- Same packet/message format for both queries and responses
- Association of queries with responses by identification field: „query id"

Inside the DNS protocol: DNS packet



Inside the DNS protocol: DNS records

distributed db storing resource records (RR)

RR format: `(name, type, class, ttl, length, data)`

- For all practical purposes: Class=IN (Internet)
- Type=A
  - **`name`** is hostname
  - **`data`** is IP address
- Type=CNAME
  - for alias
- Type=MX
  - for mail
- Type=NS
  - **`name`** is domain (e.g., foo.com)
  - **`data`** is name of authoritative DNS server for this domain

Perl Continued

- pack()/unpack()
- UDP Socket Programming()

pack()

- $data = pack($template, @list)
- pack() takes a list of scalars (@list) and packs them into a binary structure (e.g., a bitfield) according to template.
- template specifies how wide the elements of the bitfields are, and how to interpret the results
- unpack() is the reverse operation

pack() -- Examples

```
$out = pack "cccc ", 65, 66, 67, 68;   # $out eq "ABCD"
$out = pack "c4", 65, 66, 67, 68;      # same thing
$out = pack ("B8ccc", "01000001", 66, 67,68);  # same thing
                                       (01000001_2 == 65_10)

# a 8-bit field with flags, followed by a 16 bit length field in
# network byte order
$flags="10011001";  # a string
$len = 25;   # an integer, not a string
$out = pack("B8n", $flags, $len);
$out .= pack .....   # add some other stuff
```


pack()

Some frequently used template characters:

b	Bit string, ascending bit order inside each byte
B	Bit string, ascending bit order inside each byte
C	Unsigned character / 8 bit
n	Short (16 bit) in network byte order
N	Long (32 bit) in network byte order
S	Unsigned short in host byte order
I	Unsigned integer in host byte order

Complete list: perldoc –f pack or try perldoc perlpacktut

IO::Socket::INET – UDP Client

```
use IO::Socket::INET;

$client = IO::Socket::INET->new(PeerAddr => "dns.hier.de",
                                PeerPort => 53,
                                Type => SOCK_DGRAM,
                                Proto => "udp");

$client->send($dnspacket);
$answer_packet = $client->recv();
$client->close();
```


Further Reading

- DNS:
  - Kurose & Ross: Computer Networking, 4th ed. (preliminary version of 1st ed online at: http://www.net.t-labs.tu-berlin.de/teaching/computer_networking/)
  - RFC 1034 and RFC 1035
- Perl / pack / unpack / socket programming

```
perldoc IO::Handle
perldoc IO::Socket
perldoc IO::Socket::Inet
perldoc -f pack
perldoc -f unpack
perldoc perlpacktut
```