Lecture4- DNS API and Programming

NET 445 – Internet Programming

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS: services, structure

DNS services

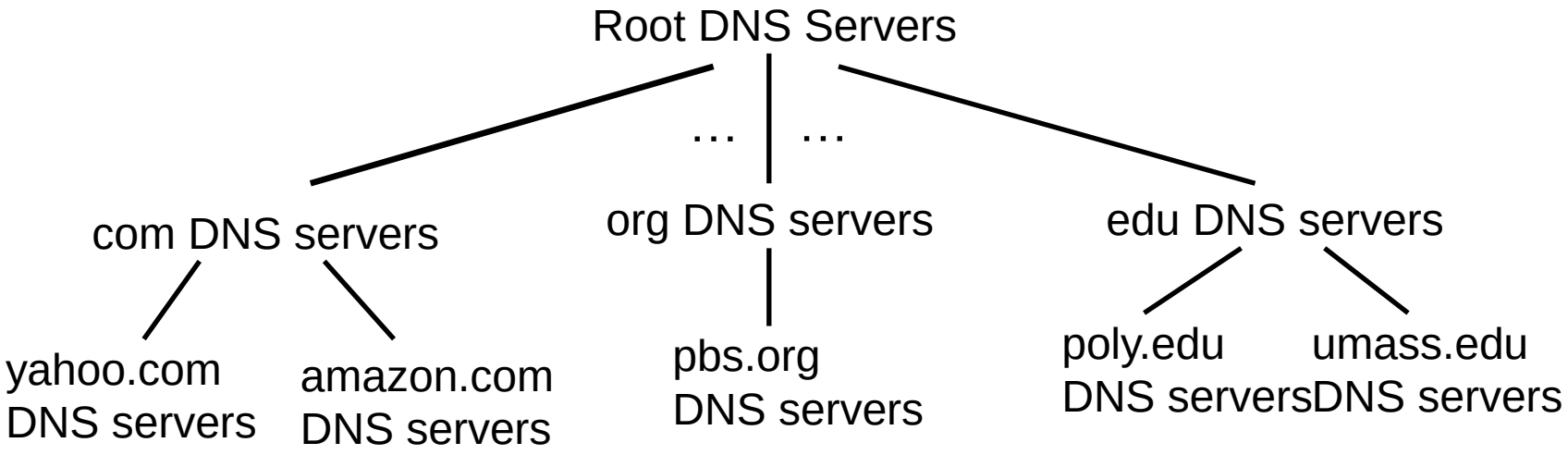
- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn't scale!

DNS: a distributed, hierarchical database

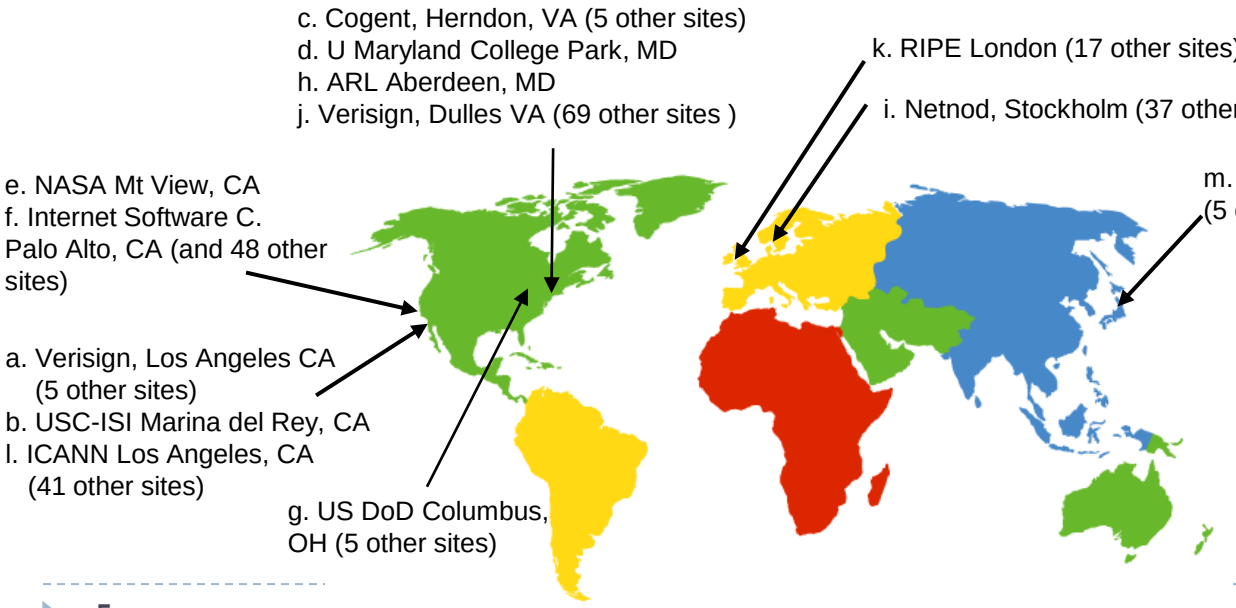


client wants IP for www.amazon.com; 1st approximation:

- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



13 logical root name "servers" worldwide

- *each "server" replicated many times*

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution

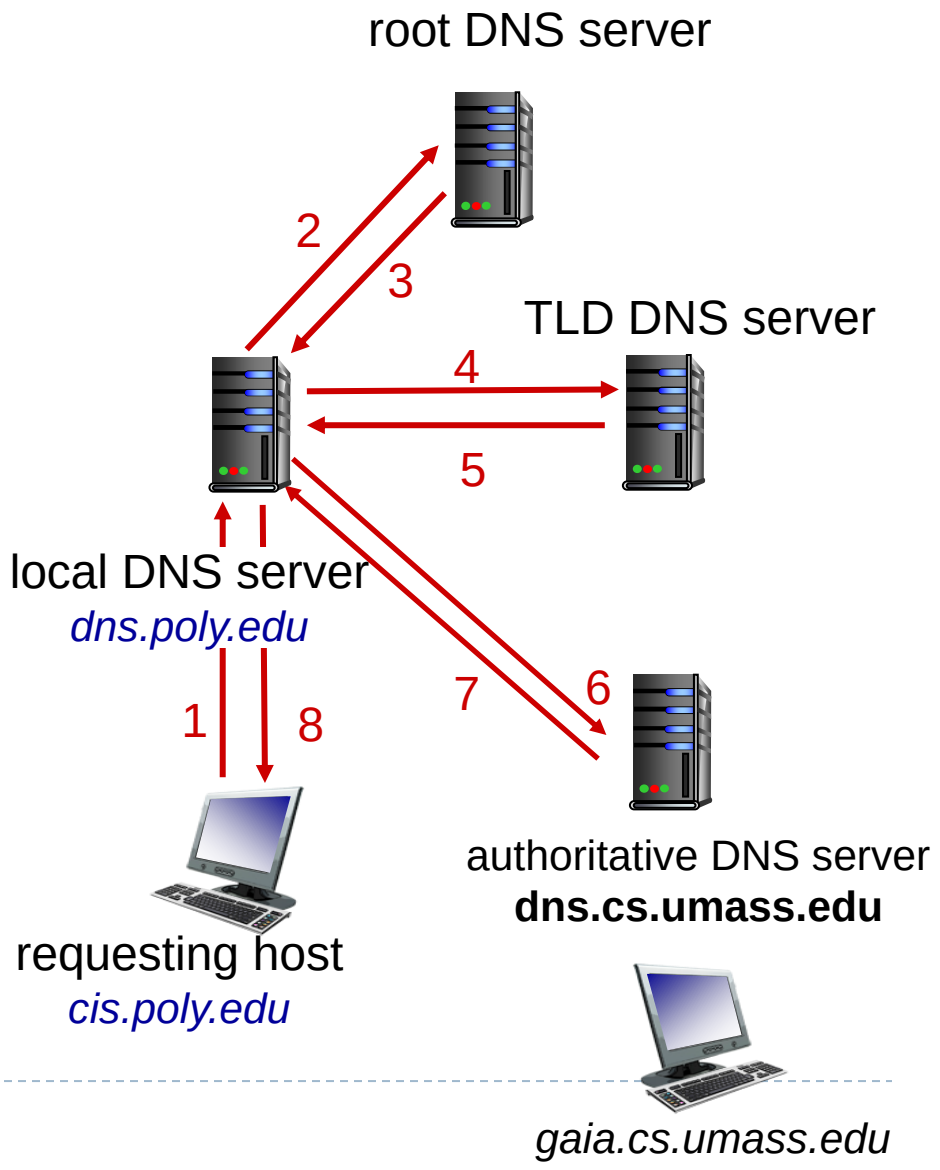
- Finding the IP address for a given hostname is called resolution and is done with the DNS protocol.
- Resolution:
  - Computer requests local name server to resolve
  - Local name server asks the root name server
  - Root returns the name server for a lower zone
  - Continue down zones until name server can answer
- DNS protocol:
  - Runs on UDP port 53, retransmits lost messages
  - Caches name server answers for better performance

DNS name : resolution example

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

iterated query:

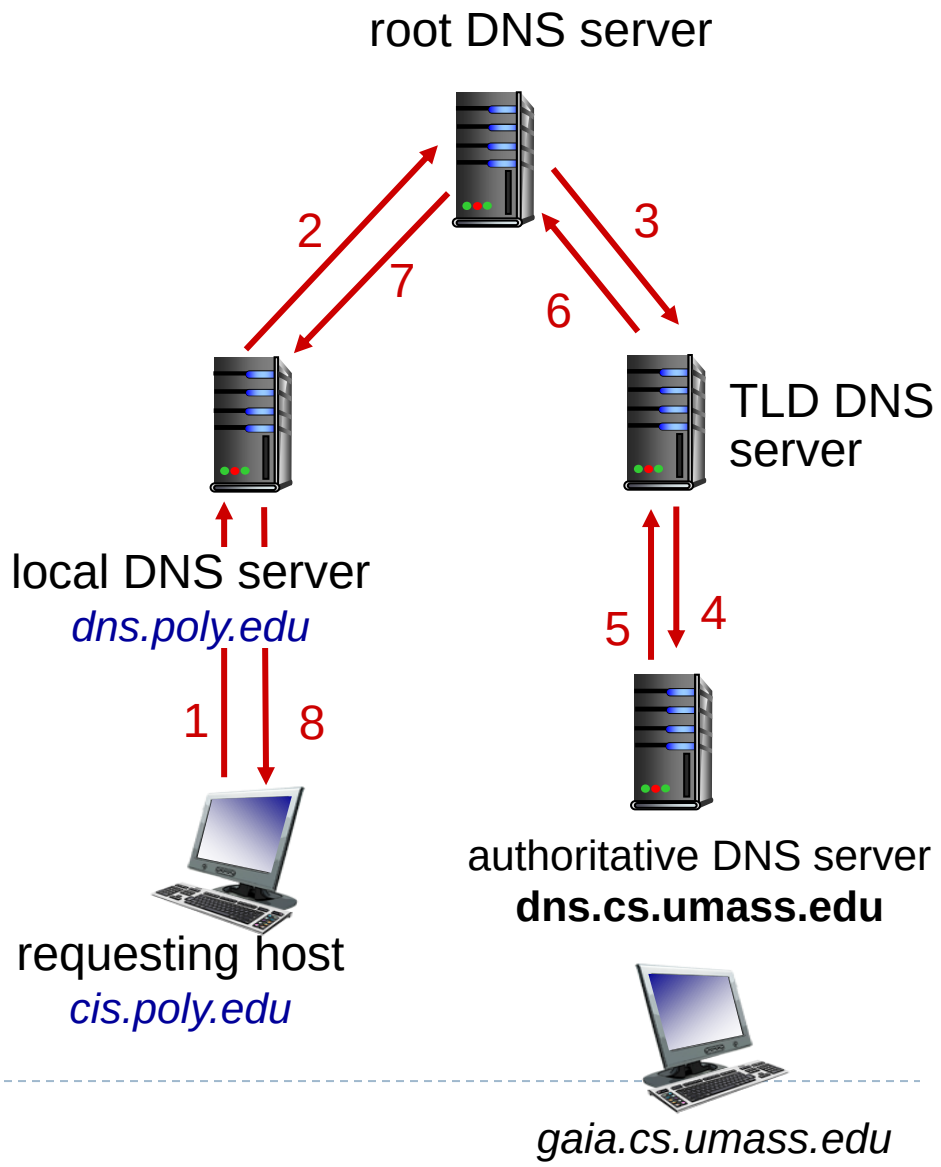
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name : resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
  - RFC 2136

DNS records

DNS: distributed database storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A

- **`name`** is hostname
- **`value`** is IP address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

- **`value`** is name of mailserver associated with **`name`**

DNS records

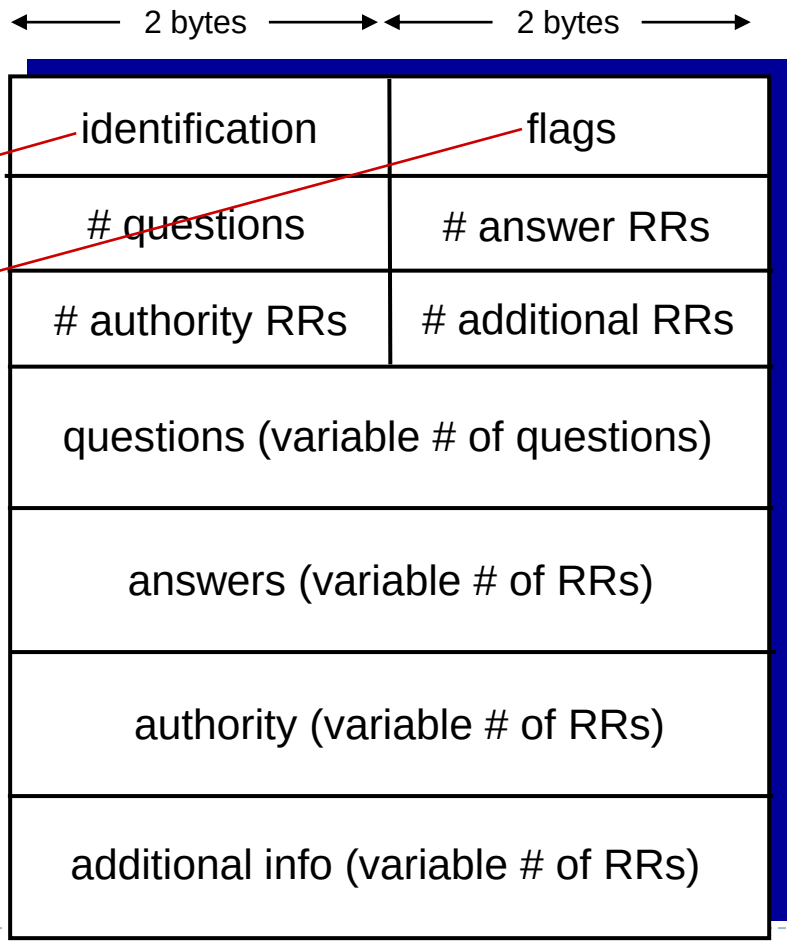
Record type	Purpose
A	IP Address record. Using a hostname to get an IPv4 adress.
AAAA	IP Address record. Using a hostname to get an IPv6 adress.
PTR	reverse DNS lookup. Using IP address to get hostname.
NS	Nameserver record responsible for the domain asked about.
MX	Mail Exchanger record. server responsible for handling email for the given domain.
SOA	Start of Authorities record describes some key data about the zone as defined by the zone administrator.
CNAME	Canonical Name or Alias, this allows providing an alternate name for a resource.
TXT	A generic Text record that provides descriptive data about domain.

DNS protocol, messages

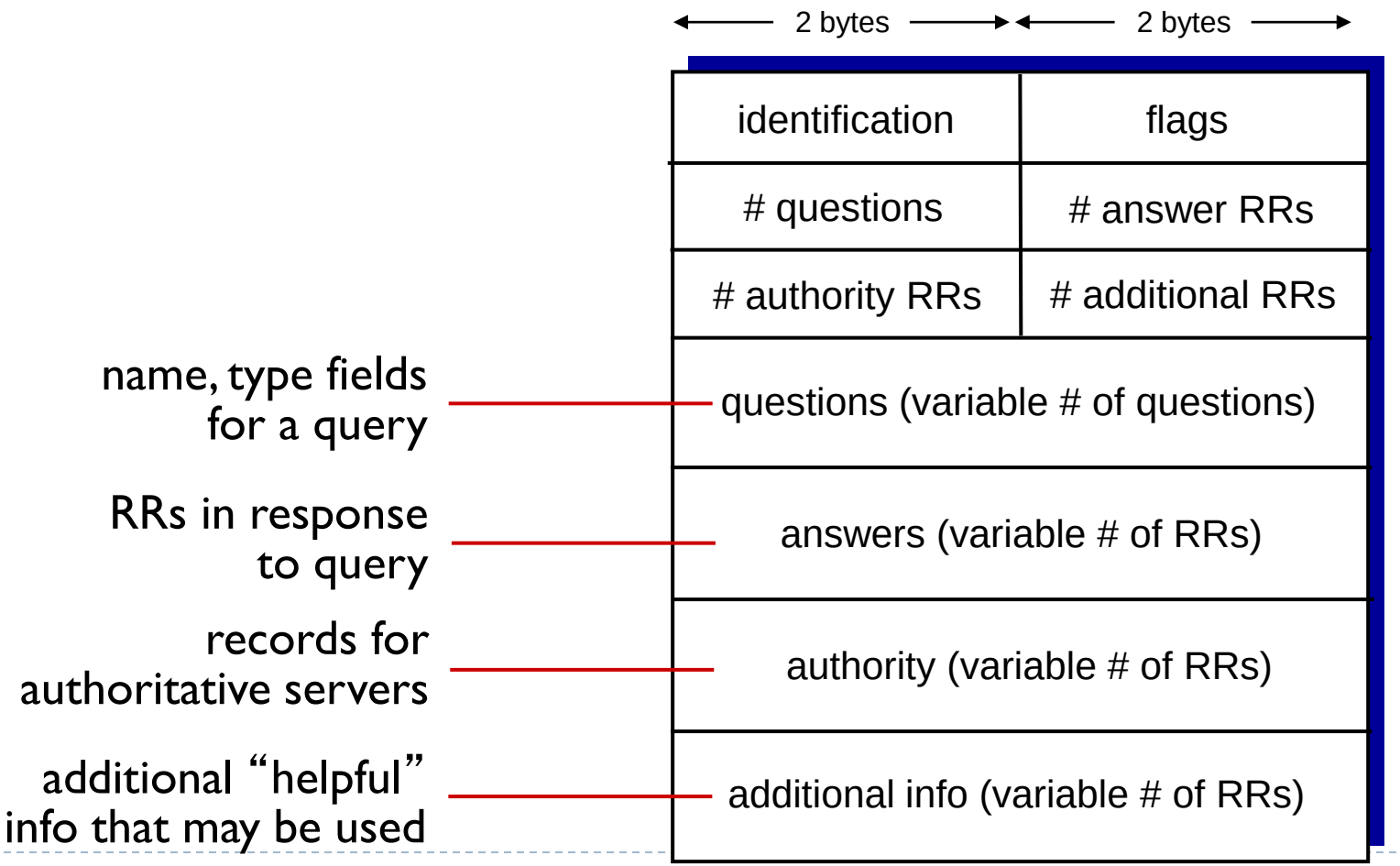
- *query* and *reply* messages, both with same *message format*

message header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



DNS protocol, messages



Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

Install DNS libraires

- Install pip to install python libraires

```
sudo apt install python3-pip
```

- Install dnspython

```
pip3 install dnspython
```

Simple DNS query via Python

- The dnspython module provides dns.resolver() helps to find out various records of a domain name.
  - The function takes two important parameters, the domain name, and the record type. Some of the record types with examples are listed below :
    - **A Record:** It is fundamental type of DNS record, here A stands for address. It shows the IP address of the domain.

```
# Import libraries
import dns.resolver
# Finding A record
result = dns.resolver.query('ksu.edu.sa', 'A')
# Printing record
for val in result:
    print('A Record : ', val.to_text())
```

Simple DNS query via Python

- The dnspython module provides dns.resolver() helps to find out various records of a domain name.
  - The function takes two important parameters, the domain name, and the record type. Some of the record types with examples are listed below :
    - **A Record:** It is fundamental type of DNS record, here A stands for address. It shows the IP address of the domain.

```
# Import libraries
import dns.resolver
# Finding A record
result = dns.resolver.query('ksu.edu.sa', 'A')
# Printing record
for val in result:
    print('A Record : ', val.to_text())
```

Simple DNS query via Python

- **AAAA Record:** This is an IP address record, used to find the IP of the computer connected to the domain. It is conceptually similar to A record but specifies only the IPv6 address of the server rather than IPv4.

```
# Import libraries
import dns.resolver
# Finding A record
result = dns.resolver.query('ksu.edu.sa', 'AAAA')
# Printing record
for val in result:
    print('A Record : ', val.to_text())
```

Simple DNS query via Python

- **PTR Record:** PTR stands for pointer record, used to translate IP addresses to the domain name or hostname. It is used to reverse the DNS lookup.

```
# Import libraries
import dns.resolver

# Finding PTR record
result = dns.resolver.query('116.62.218.34.in-addr.arpa', 'PTR')

# Printing record
for val in result:
    print('PTR Record : ', val.to_text())
```

Simple DNS query via Python

- A CNAME record also known as Canonical Name Record
- a type of record in the Domain Name System (DNS) used to map a domain name as an alias for another domain.
- CNAME records always point to another domain name and never directly to an IP address. In the query method below we specify the CNAME parameter to get the CNAME value.

```
import  dns
import dns.resolver
result = dns.resolver.query("mail.google.com","CNAME")
for cnameval in result:
    print (' cname target address:', cnameval.target)
```

References:

- Foundations of Python Network Programming Third Edition by Brandon Rhodes (2014)
- James F. Kurose, and Keith W Ross, Computer Networking: A Top-Down Approach, 6th Edition
- Python 3 documentation
- https://wiki.python.org/moin/UdpCommunication
- https://www.w3schools.com/python/
- https://www.tutorialspoint.com/python/