Lecture Notes for Lecture 14 of CS 5600 (Computer Systems) for the Fall 2019 session at the Northeastern University Silicon Valley Campus.

Domain Names and DNS

Philip Gust,
Clinical Instructor
Department of Computer Science

These slide highlight content from suggested readings.

Review of Lecture 13

- In this lecture we continued discussing how to use sockets to communicate between processes on different computers connected by a network,
- We also briefly discussed the use of *remote procedure calls* (*RPCs*) and the facilities available to implement them in different programming languages.
- Next we looked at the idea of services and how they are implemented using sockets and standardized ports.
- Finally, we reviewed the basics of the HTTP service, and how requests and responses are used to invoke the five basic HTTP methods: GET, HEAD, PUT, POST, and DELETE.

Domain Names and DNS

- So far we have used IP addresses to create sockets and access remote computers. In particular we used IPv4 style IP addresses
- In this lecture we will look at domain names, how they are constructed, and how domain names are resolved to IP addresses
- In particular, we will look at the Domain Name System (DNS) and its distributed network of DNS servers that resolve names on the internet.

Domain Names and DNS

Introduction

- The *Domain Name System* (*DNS*) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet.
- The DNS associates various information with domain names assigned to each of the participating entities.
- Most prominently, it translates more readily memorized domain names to the numerical IP addresses for locating computer services with underlying network protocols.
- The DNS distributed directory service has been an essential component of the functionality of the Internet since 1985.

Domain Names and DNS

Introduction

- The DNS delegates the responsibility of assigning domain names and mapping those names to Internet resources by designating authoritative name servers for each domain.
- Network administrators may delegate authority over sub-domains of their allocated name space to other name servers.
- This mechanism provides distributed and fault-tolerant service and was designed to avoid a single large central database.

Domain Names and DNS

Domain Names

- A domain name consists of one or more parts, technically called *labels*, that are concatenated, and delimited by dots, such as *example.com*.
- The right-most label conveys the top-level domain (TLD). For example, the domain name *www.example.com* belongs to the top-level domain *com*.

Domain Names and DNS

Domain Names

- The hierarchy of domains descends from right to left. Each label to the left specifies a subdivision, or subdomain of the domain to the right.
- For example, the label *www.example.com* specifies a subdomain of the *com* domain, and *www* is a subdomain of *example.com*.
- The tree of subdivisions may have up to 127 levels.

Domain Names and DNS

Domain Names

- A label may contain 0 to 63 characters. A domain name may not exceed the length of 253 characters in its textual representation.
- Although no technical limitation exists to use any character in domain name labels which are representable by an octet, hostnames use a preferred format and character set.
- The characters allowed in labels are a subset of the ASCII character set, consisting of characters ‘a’ through ‘z’, ‘A’ through ‘Z’, digits ‘0’ through ‘9’, and hyphen (‘-’).
- Domain names are interpreted in case-independent manner. Labels may not start or end with a hyphen.

Domain Names and DNS

Domain Names

- The limited set of ASCII characters permitted in the DNS prevented the representation of names and words of many languages in their native alphabets or scripts.
- To make this possible, ICANN approved the Internationalizing Domain Names in Applications (IDNA) system.
- User applications, such as web browsers, map Unicode strings into the valid DNS character set using *Punycode*, a representation with the limited ASCII character subset.

Domain Names and DNS

Domain Names

- Online converters from Punycode to Uncode are available, for example: https://www.name.com/punycode-converter
  - München (German name for the city of Munich) is encoded as: Mnchen-3ya.
  - Москва (Russian name for the city of Moscow) is encoded as: xn--o0a4bf5ats
  - 新加坡 (Chinese name for the city of Singapore) is encoded as: xn--eqrw53b41bnww7jd.
  - 沖縄県 (Japanese name for the city of Okinawa) is encoded as: xn--uuw26t5rf

Domain Names and DNS

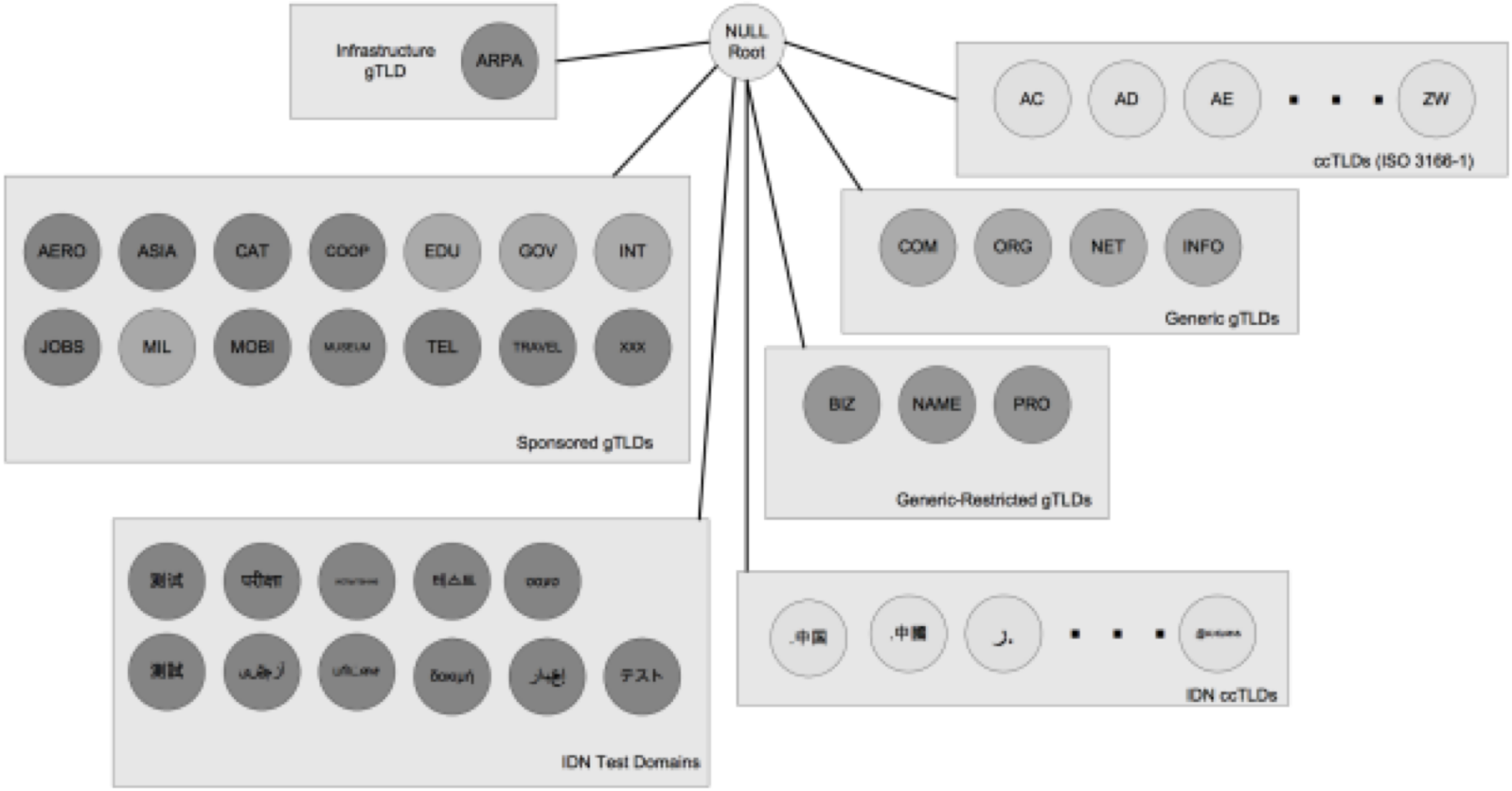
The DNS Name Space

- The DNS name space is the set of all names used with DNS. This space is partitioned hierarchically and is case insensitive.
- The current DNS name space is a tree of domains with an unnamed root at the top. The top echelons of the tree are the so-called *top-level domains* (*TLD*s), which include:
  - Generic TLDs (gTLDs)
  - Country-code TLDs (ccTLDs)
  - Internationalized country-code TLDs (IDN ccTLDs)
  - A special infrastructure TLD called, for historical reasons, ARPA [RFC3172].

Domain Names and DNS

The DNS Name Space

- The DNS name space forms the top levels of a naming tree.



Domain Names and DNS

The DNS Name Space

- The names below a TLD in the DNS name tree are further partitioned into subdomains, which is very common practice, especially for the ccTLDs.
  - Fully qualified domain name (FQDN)
  - Unqualified domain name

Domain Names and DNS

The DNS Name Space

- A **fully qualified domain name (FQDN)** represent the names seen so far. They are sometimes written more formally with a trailing period (e.g., *northeastern.edu.*).
- This trailing period indicates that the name is complete; no additional information should be added to the name when performing a name resolution.

Domain Names and DNS

The DNS Name Space

- An **unqualified domain name**, which is used in combination with a default domain or domain search list set during system configuration, has one or more strings appended to the end.
- During configuration, system is typically assigned a default domain extension and search list using DHCP.
- For example, the default domain *ccs.northeastern.edu* might be configured in systems at the computer science department at Northeastern University.
- If a user on one of these machines types in the name github, the local resolver software converts it to the FQDN *github.ccs.northeastern.edu*. before invoking a resolver to determine *github*'s IP address.

Domain Names and DNS

The DNS Name Space

- The hierarchical structure of the DNS name space allows different administrative authorities to manage different parts of the name space.
- For example, creating a new DNS name *elevator.css.northeastern.edu* would require dealing with the owner of the *ccs.northeastern.edu* subdomain only.
- The *northeastern.edu* and *edu* portions of the name space would not require alteration, so the owners of those would not need to be bothered.
- This feature of DNS is one key aspect of its scalability. No single entity is required to administer all the changes for the entire DNS name space

Domain Names and DNS

Name Servers and Zones

- The manager of an active DNS name space is supposed to arrange for at least two name servers or DNS servers to hold information about the name space so that Internet users can perform queries on the names even if one goes down.
- The DNS (formed by servers) is a distributed system whose primary job is to provide name-to-address mappings; however, it can also provide a wide array of additional information.

Domain Names and DNS

Name Servers and Zones

- Examples:
  - At a small campus, one person could do this each time a new server is added to the network;
  - In a large enterprise the responsibility would have to be delegated (probably by departments or other organizational units), as one person likely could not keep up with the work.

Domain Names and DNS

Name Servers and Zones

- A DNS server can contain information for more than one zone.
- At any hierarchical change point in a domain name, a different zone and containing server may be accessed to provide information for the name. This is called a *delegation*.

Domain Names and DNS

Name Servers and Zones

- A common delegation approach uses a zone for implementing a second-level domain name, such as northeastern.edu.
- In this domain, there may be individual hosts (e.g., *www.northeastern.edu*) or other domains (e.g., *cis.northeastern.edu*).
- Each zone has a designated owner or responsible party who is given authority to manage the names, addresses, and subordinate zones within the zone.
- Often this person manages not only the contents of the zone but also the name servers that contain the zone's database(s).

Domain Names and DNS

Name Servers and Zones

- For redundancy, zone information is supposed to exist in at least two places: there should be at least two servers containing information for each zone.
- All of these servers contain identical information about a zone.
- Among the servers, a *primary server* contains the zone database in a disk file, and one or more *secondary servers* obtain copies from the primary using a process called a *zone transfer*.
- DNS has a special protocol for performing zone transfers, but copies of a zone's contents can also be obtained using other means (e.g., the **rsync** utility).

Domain Names and DNS

Caching

- Name servers contain information (e.g. name-to-IP-address mappings) that may be obtained from three sources:
  - Directly from the zone database,
  - As the result of a zone transfer (e.g., for a slave server),
  - From another server in the course of processing a resolution.
- In the first case, the server is said to contain *authoritative information* about the zone and may be called an *authoritative server* for the zone.
- Authoritative servers are identified by name within the zone information.

Domain Names and DNS

Caching

- Most name servers (except some root and TLD servers) also cache zone information they learn, up to a time limit called the time to live (TTL).
- Using this cached information to answer queries greatly decreases the amount of DNS message traffic on the Internet.
- When answering a query, a server indicates whether the information it is returning has been derived from its cache or from its authoritative copy of the zone.
- When cached information is returned, it is common for a server to also include the domain names of the name servers that can be contacted to retrieve authoritative information about the corresponding zone.

Domain Names and DNS

Caching

- In Linux , the *Name Service Caching Daemon* (*NSCD*) provides a client-side caching capability.
- NSCD is controlled by the /etc/nscd.conf file that indicates which types of resolutions (for DNS and some other services) are cached, along with cache parameters such as TTL settings.
- The file /etc/nsswitch.conf controls how name resolution for applications takes place. It also controls whether local files, the DNS protocol, and/or NSCD is employed for mappings

Domain Names and DNS

DNS Name Resolution

- The DNS protocol consists of two main parts:
- Query/response protocol used for performing queries against the DNS for particular names
- Protocol for name servers to exchange database records (zone transfers)
- Other functionality includes:
  - Notifying secondary servers that the zone database has evolved and a zone transfer is necessary (DNS Notify)
  - Dynamically updating the zone (dynamic updates).

Domain Names and DNS

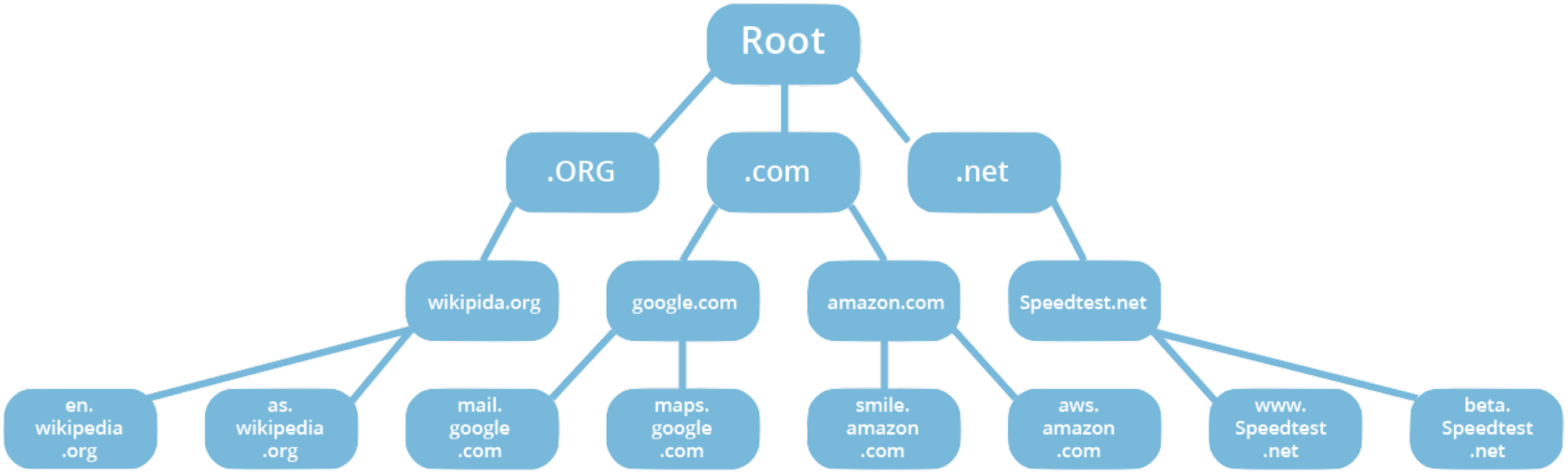
DNS Name Resolution

- DNS name resolution is the process of mapping a domain name to an IPv4 address, although IPv6 addresses mappings work in essentially the same way.
- DNS query/response operations are supported over the distributed DNS infrastructure consisting of servers deployed locally at each site or ISP, and a special set of root servers.
- There is also a special generic top-level domain servers used for scaling some of the larger gTLDs, including COM and NET.

Domain Names and DNS

DNS Name Resolution

- Example of hierarchical name servers



Domain Names and DNS

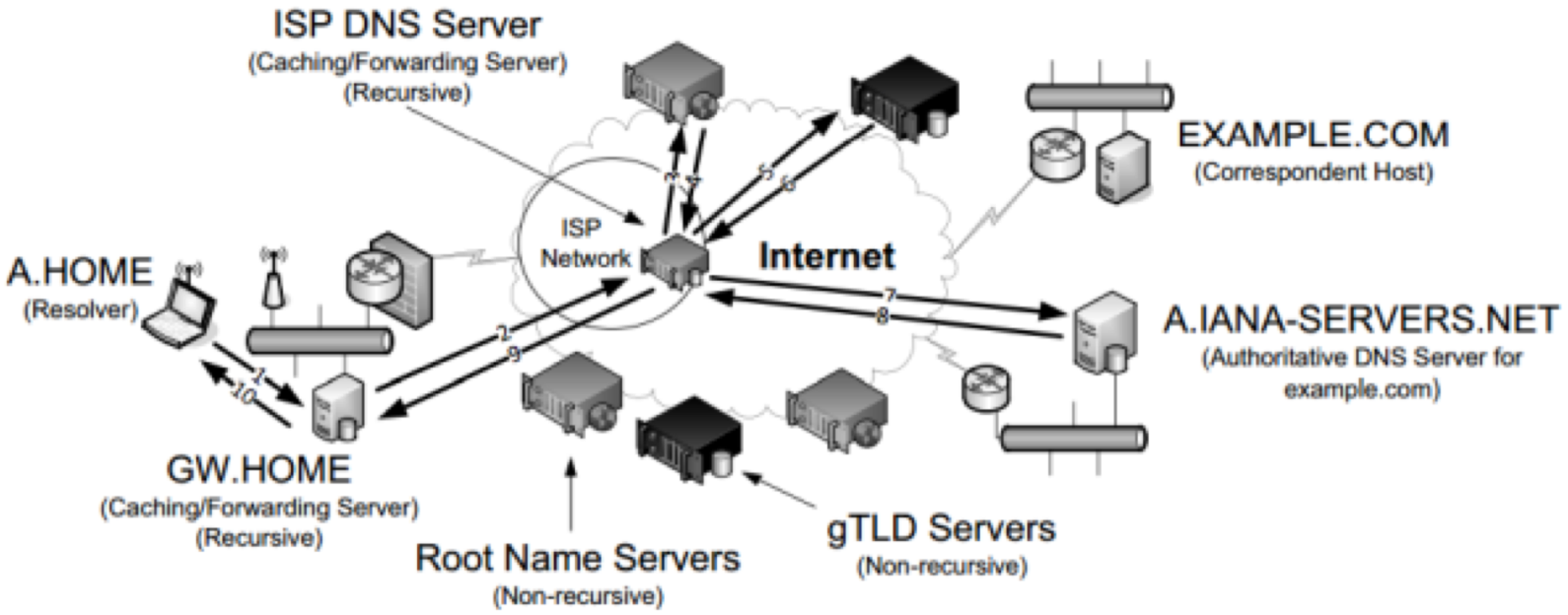
DNS Name Resolution

- As of mid-2011, there are:
  - 13 root servers named by the letters A through M; 9 of them have IPv6 addresses.
  - 13 gTLD servers named by A through M; 2 of them have IPv6 addresses.
- Some of them are not a single physical server but a group of servers (over 50 for the J root server) that use the same IP address (i.e., using *IP anycast addressing*)

Domain Names and DNS

DNS Name Resolution

- A full resolution that is unable to benefit from preexisting cached entries takes place among several entities.

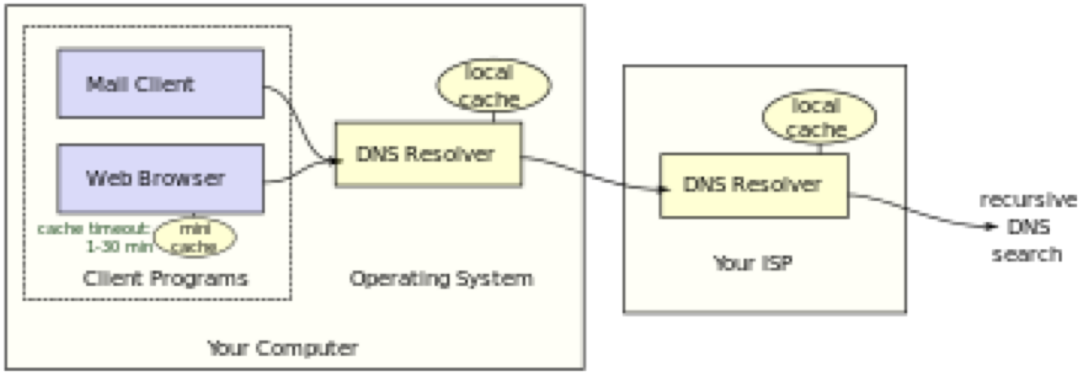


Domain Names and DNS

DNS Name Resolution

- Here is an outline of the name resolution process:

  Message 1: The resolver software (assuming it does not know the IP address for the server EXAMPLE.COM) on A.HOME first makes a request to its local name server, GW.HOME.



Domain Names and DNS

DNS Name Resolution

- Here is an outline of the name resolution process:

  Messages 2-6:

  - GW.HOME forwards the request to another DNS server (called recursion), in this case, an ISP DNS server.
  - The ISP DNS server contacts a root name servers (message 3).
  - The root servers are not recursive, so they do not process the request further but instead return the information required to contact a name server for the COM TLD.
  - The ISP-provided server contacts the gTLD server (message 5) and discovers the name and IP addresses of the name servers for the domain EXAMPLE.COM (message 6).

Domain Names and DNS

DNS Name Resolution

- Here is an outline of the name resolution process:

  Messages 7-10:

  - Given the correct server for the domain, the ISP-provided server contacts the appropriate server (message 7), which responds with the requested IP address (message 8). At this point, the ISP-provided server can respond to GW.HOME with the required information (message 9).
  - GW.HOME is now able to complete the initial query and responds to the client with the desired IPv4 and/or IPv6 address(es) (message 10)

Domain Names and DNS

DNS Name Resolution

- From A.HOME's side it seems the local name server was able to perform the request.
- What really happened is a recursive query, where the GW.HOME and ISP-provided servers in turn made additional DNS requests to satisfy A.HOME’s query.
- In general, most name servers perform recursive queries such as this. The notable exceptions are the root servers and other TLD servers that do not perform recursive queries.
- These servers are a relatively precious resource, so encumbering them with recursive queries for every DNS query would lead to poor global Internet performance.

Domain Names and DNS

DNS Name Resolution

- Here is a program that performs name resolution and return the IPv4 and IPv6 addresses of the specified domain name.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    if (argc != 2) {
        fprintf(stderr, "Usage: %s hostname\n", argv[0]);
        return 1;
    }
    char *hostName = argv[1];
```

Domain Names and DNS

DNS Name Resolution

```
struct addrinfo hints, *res;
memset (&hints, 0, sizeof (hints));
hints.ai_family = PF_UNSPEC; // any network family
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags |= AI_CANONNAME;  // include cannonical name

if (getaddrinfo (hostName, NULL, &hints, &res) != 0) {
        perror ("getaddrinfo");
        return -1;
}
```

Domain Names and DNS

DNS Name Resolution

```
        printf ("Host: %s\n", hostName);
        for (void *ptr ; res != NULL; res = res->ai_next) {
                switch (res->ai_family) {
                case AF_INET:
                        ptr = &((struct sockaddr_in *) res->ai_addr)->sin_addr;
                        break;
                case AF_INET6:
                        ptr = &((struct sockaddr_in6 *) res->ai_addr)->sin6_addr;
                        break;
                }

                // convert address to printable form
                char addrstr[100];
                inet_ntop (res->ai_family, ptr, addrstr, 100);
                printf ("IPv%d address: %s (%s)\n", res->ai_family == PF_INET6 ? 6 : 4,
                        addrstr, res->ai_canonname);
         }
       freeaddrinfo(res); // free the linked list
       return 0;
    }
```

Domain Names and DNS

Reverse DNS Lookup

- A reverse DNS lookup is a query of the DNS for domain names when the IP address is known. Multiple domain names may be associated with an IP address.
- The DNS stores IP addresses in the form of domain names as specially formatted names in pointer (PTR) records within the infrastructure top-level domain arpa.
- For IPv4, the domain is in-addr.arpa. For IPv6, the reverse lookup domain is ip6.arpa.
- The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.

Domain Names and DNS

Reverse DNS Lookup

- To perform reverse lookup, the DNS client converts the address before querying the name for a PTR record following the delegation chain as for any DNS query.
- For example, assuming the IPv4 address *129.10.116.5*. is assigned to NEU CCIS, it is represented as a DNS name in reverse order: *5.16.10.129.in-addr.arpa*.

Domain Names and DNS

Reverse DNS Lookup

- When the DNS resolver gets a pointer (PTR) request, it begins by querying the root servers, which point to the servers of American Registry for Internet Numbers (ARIN) for the 129.in-addr.arpa zone.
- ARIN's servers delegate *5.16.10.129.in-addr.arpa* to NEU CCIS to which the resolver sends another query for *5.16.10.129.in-addr.arpa*, which results in an authoritative response.

Domain Names and DNS

DNS Name Resolution

- Here is a program that performs reverse DNS lookup on a given IP address.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
   if (argc != 2) {
      fprintf(stderr, "Usage: %s ip-address\n", argv[0]);
      return 1;
   }
```

Domain Names and DNS

DNS Name Resolution

```
// get information about IP address
struct addrinfo hints, *res = NULL;
memset(&hints, '\0', sizeof hints);
hints.ai_family = PF_UNSPEC;
hints.ai_flags = AI_NUMERICHOST;  // no name resolution
if (getaddrinfo(argv[1], NULL, &hints, &res)) {
  printf("Invalid address: %s\n", argv[1]);
  return 1;
}
```

Domain Names and DNS

DNS Name Resolution

```
char buf[NI_MAXHOST];

if (res->ai_family == AF_INET6) {  // lookup IPv6 address
  struct sockaddr_in6 sa;
  sa.sin6_family = AF_INET6;
  inet_pton(AF_INET6, argv[1], &sa.sin6_addr);
  if (getnameinfo((struct sockaddr *) &sa, sizeof(sa), buf,
                              sizeof(buf), NULL, 0, NI_NAMEREQD) != 0) {
        printf("Could not resolve reverse lookup of IPv6 hostname for %s\n", argv[1]);
  } else {
        printf("Resolved reverse lookup for IPv6 address %s to %s\n", argv[1], buf);
  }
}
```

Domain Names and DNS

DNS Name Resolution

```
    else if (res->ai_family == AF_INET) {  // lookup IPv4 address
      struct sockaddr_in sa;
      sa.sin_family = AF_INET;  // address is IPv4
      inet_pton(AF_INET, argv[1], &sa.sin_addr);

      if (getnameinfo((struct sockaddr *) &sa, sizeof(sa), buf,
                                  sizeof(buf), NULL, 0, NI_NAMEREQD) != 0) {
            printf("Could not resolve reverse lookup of IPv4 hostname for %s\n", argv[1]);
      } else {
            printf("Resolved reverse lookup for IPv4 address %s to %s\n", argv[1], buf);
      }
    } else {  // unknown address format
      printf("Unknown address format: %s\n", argv[0]);
    }
    freeaddrinfo(res); // free the linked list
}
```