Networking Basics 1: TCP/IP Introduction

1. **TCP/IP Introduction**
   - **TCP/IP Protocol Suite**
   - **IPv4 vs. IPv6**
   - **IPv4 addresses**
   - **IPv6 addresses**
2. Routing and Ports
3. TCP/IP Protocols
4. Ethernet
5. DNS

The TCP/IP Protocol Suite

The **network protocols** that are used with the Internet are commonly known as **TCP/IP**.

TCP/IP is named for its most important parts:

- the Transmission Control Protocol (TCP)
- the Internet Protocol (IP)

Alternative, formal name: **Internet Protocol Suite**.

"Internet" here is not *the Internet*, but rather short for "interconnected network" or internetwork (since predates the (global) Internet).

Goal: to be able to connect networks that were potentially using different physical technologies.

Layered Protocol

The functionality that makes up networking protocols is typically conceptualized in terms of being partitioned into *abstraction layers*.

TCP/IP's **network model** has *4 layers*:

- **application** – user-level applications (e.g., FTP, SSH)
- **transport** – end-to-end data transfer protocols (e.g., TCP and UDP)
- **network** – abstract (non-physical) network protocol (e.g., IP and ARP)
- **link/network interface** – physical network protocol (e.g., Ethernet, Token-ring, ATM)

Layered Protocol (contd.)

The more common **OSI network model** uses 7 layers:

- **application** – user-level applications
- **presentation** – data mapping (format, encryption)
- **session** – inter-host connection control
- **transport** – data transfer protocols for applications
- **network** – inter-host data transfer and addressing
- **data** –link physical data transfer and addressing
- **physical** – media and transmission

IPv4 vs. IPv6

There are two main IP standards:

- **IPv4** (IP version 4)
- **IPv6** (IP version 6)

IPv4 is currently the most widely used standard, so we will concentrate on it.

IPv6 is the upcoming/eventual standard, but its use on the Internet remains limited.

IPv6 has many advantages, including:

- much larger addresses
- support for larger packets
- improved security and **multicasting**

IP Addresses

IP addresses are one of the most visible and important elements of TCP/IP.

An **IP address** is intended to *uniquely identify* a **host** on the Internet.

IPv4 addresses are 32-bit numbers.

This means there are only 2^{32} or less than 4.3×10^9 addresses.

The available IPv4 addresses have basically all been allocated as of 2014!

IPv6 addresses use 128 bits, so there are 2^{128} or 3.4×10^{38} possible addresses!!

IPv4 Addresses

IPv4 addresses are often written as the values in *decimal* of each of the *four bytes*, separated by **dots**:
e.g., 192.168.0.101 or 131.230.133.154

This format is referred to as "**dotted decimal**" or "**dotted quad**."

Each byte/quad/octet can range from 0 to 255.

An IPv4 address consists of two conceptual parts:

- **network number** – assigned to the network owner by an Internet authority (or ISP)
- **host number** – assigned to a machine by the network owner

IPv4 Addresses (contd.)

How many of the bits in the address are used for the network number determines how many hosts can be supported within the network.

Originally, the **address class** determined this:

- **class A** – first bit 0, network number is 7 bits, so 16,777,214 hosts (last 3 quads)
- **class B** – first bits 10, network number is 14 bits, so 65,534 hosts (last 2 quads)
- **class C** – first bits 110, network number is 21 bits, so 254 hosts (last quad)
- **class D** – first bits 1110, multicast
- **class E** – first bits 1111, reserved

IPv4 Addresses (contd.)

The class method of determining network vs. host number proved to be too inflexible.

Initially the notion of a **subnet** was introduced, by dividing the host number into two parts.

So an IP address consisted of three parts:
`<network number><subnet number><host number>`

In 1993, classes were abandoned completely, replaced by **classless inter-domain routing** (CIDR).

With CIDR the network-host division need not be on quad/byte boundaries.

IPv4 Addresses (contd.)

A CIDR block of addresses is specified using:

1. an IP address, and
2. the network prefix length (in bits)

Thus the equivalent of a class B address block would be specified like: 130.201.0.0/16

An address block where the network-host division is not on a quad/byte boundaries would be something like: 192.168.1.128/26

The **Internet Assigned Numbers Authority** (IANA) gives out IP ranges using CIDR specs.

IPv4 Addresses (contd.)

Consider the address block 192.168.1.128/26:

- 26 bits means the high order 3 quads/bytes plus the *high order 2 bits* in the last/lowest order quad/byte
- $128_{10} = 10000000_2$, so the two network bits are 10
- so all addresses in this block must have their high order 3 quads/bytes be 192.168.1 and then have the high order 2 bits in the last quad be 10: 192.168.1.(10xxxxxx)
- thus the address block range of IP addresses is:
  - 192.168.1.128 [which is 192.168.1.(10000000)] through
  - 192.168.1.191 [which is 192.168.1.(10111111)]

IPv4 Addresses (contd.)

The network vs. host parts of IP addresses are also often specified via a **subnet mask**:

- 32 bit number like an IP address, written similarly
- 1 bit means network part
- 0 bit means host part
- e.g., 255.255.255.192:
- $255_{10} = 11111111_2$ and $192_{10} = 11000000_2$
- so high order 26 bits are network part
- low order 6 bits are the host part
- thus subnet mask 255.255.255.192 is equivalent to a CIDR prefix of 26

IPv4 Addresses (contd.)

IP addresses are one of:

- **unicast** – intended for a single machine
- **broadcast** – intended for all (local) machines
- **multicast** – intended for multiple machines

Broadcast addresses have *all 1's in host portion*.

E.g., for 192.168.1.120/26 the local network broadcast address is 192.168.1.191 [192.168.1.(10111111)]

The *special broadcast address* 255.255.255.255 is interpreted as broadcast for the *local network*.

IPv4 Addresses (contd.)

An IP address with *all 0's in the host part* means *this machine*.

E.g., if machine is 192.168.1.129/26 then 192.168.1.128 also means this machine since $129 = 10000001_2$ and $128 = 10000000_2$.

Having *all 0's in the network part* means *this/local* network.

E.g., 0.0.0.1 is equivalent to 192.168.1.129 given the network defined above.

Special address 0.0.0.0 means this machine on this/local/zero network.

Multicast addresses start with 1110:
224.0.0.0 to 239.255.255.255

IPv4 Addresses (contd.)

Several ranges of IPv4 addresses are "special:"

- **loopback** – routed directly to the local machine
  - 127.0.0.0 to 127.255.255.255
  - 127.0.0.1 is the typical loopback/lo address
- **private** – not routable on Internet
  - 10.0.0.0 to 10.255.255.255
  - 172.16.0.0 to 172.31.255.255
  - 192.168.0.0 to 192.168.255.255
- **link-local** – not routable, automatically configured
  - 169.254.1.0 to 169.254.254.255
- **reserved** – currently unused
  - 240.0.0.0 to 255.255.255.254

IPv6 Addresses

As noted, IPv6 addresses use 128 bits, so the set of possible IPv6 addresses is vast.

Advantages over IPv4:

- will avoid **address exhaustion** forever
- large enough to allow every device to have an IP address ("**Internet of Things**")
- will allow pure hierarchical allocation of addresses, simplifying routing and changing ISPs
- **stateless address autoconfiguration** (SLAAC) possible (automatic address configuration for hosts)
- provides hosts both unchanging link-local address and global (routable) address at the same time

IPv6 Addresses (contd.)

128 bit address is written as 8 groups of 4 hex digits separated by colons:
e.g., 2001:0db8:0000:0000:5e72:287a:ccd4:00d1

Many groups/digits will be 0's, so a sequence of 0000 groups can be replaced by double colons (::) but only one :: per address.

Leading 0's in groups can be dropped:
e.g., 2001:0db8:0000:0000:5e72:287a:ccd4:00d1
or 2001:db8:0:0:5e72:287a:ccd4:d1
or 2001:db8::5e72:287a:ccd4:d1

IPv6 Addresses (contd.)

Typically first 64 bits used for network part and last 64 bits for host part.

Final 4 bytes can also be written in dotted quad:
e.g., xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:83e6:8552
or xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:131.230.133.82

Loopback address (localhost):
0000:0000:0000:0000:0000:0000:0000:0001
or 0:0:0:0:0:0:0:0:1
or just ::1

IPv4 Mapped Addresses (transition address):
::ffff:0:0/96 (96-bit prefix on IPv4 32-bit address)

IPv6 Addresses (contd.)

Unique local addresses (ULA's), a type of private (not globally routable) address: fc00::/7.

Multicast (set of hosts) addresses: ff00::/8

Multicast to all local hosts (i.e., broadcast): ff02::1

Can automatically generate 64-bit host part of address from a 48-bit NIC interface address (e.g., Ethernet MAC address):

- insert fffe between 3rd and 4th bytes
- complement 2nd lowest order bit of highest order byte
- e.g., 00:60:a8:52:f9:d8 becomes 0260:a8ff:fe52:f9d8

Networking Basics 2: Routing & Ports

1. TCP/IP Introduction
2. **Routing and Ports**
   - **routing**
   - **ports**
   - **NAT**
3. TCP/IP Protocols
4. Ethernet
5. DNS

Routing

Routing refers to the process of communicating/forwarding a packet.

A machine must know two pieces of information to route packets:

1. its own IP address
2. the subnet mask for its local (physical) network

Each outgoing packet address is then classified as either:

- *remote address* – destination IP address differs from machine's IP address in the *network part*
- *local address* – destination IP address differs from machine's IP address *only in the host part*

How a packet is sent depends on the packet's address type.

Routing (contd.)

Local addresses:

- packet is sent directly to the (local) destination machine
- get MAC address of machine corresponding to destination IP address using ARP
- send packet via Ethernet (encapsulated in Ethernet frame)

Remote addresses:

- packet is sent to the **gateway** (router) machine
- get MAC address of gateway device via ARP
- send packet via Ethernet to that device
- leave original destination IP address in IP packet header
- gateway machine forwards based on desination IP address

Routing (contd.)

Routing example:

- machine IP: 192.168.1.147
- netmask: 255.255.255.192
- 147_{10} is 10010011_2 so network 2 bits are 10
- thus *local addresses* have form: 192.168.1.(10xxxxxx)
- so range is 192.168.1.128 to 192.168.1.191
- however two of these addresses cannot be used for hosts due to special meanings:
  - 192.168.1.128 means "this machine"
  - 192.168.1.191 means broadcast
- all other addresses are *remote addresses*

Ports

While an IP address identifies a machine/host, additional info will be required to identify a particular *process* (e.g., server or client) that a remote machine wants to communicate with.

This is done by associating a *16 bit integer* called a **port** with the process.

16 bits means ports range from 0 through 65535.

An IP address and port number should together uniquely identify a process to communicate with.

A process associates itself with a port by creating a **socket** and giving it an *address*.

Sockets are the *communication endpoints* for network IPC.

Ports (contd.)

There are 3 classes of ports:

- **well-known**
  - 0 to 1023
  - assigned by Internet authority to particular type of server
  - can be opened only by root on UNIX/Linux systems
- **registered**
  - 1024 to 49151
  - more server ports, no root restrictions
- **dynamic/private**
  - 49152 to 65535
  - used by clients when open socket/connection

(**Ephemeral**: older term for ports 1024+.)

Ports (contd.)

See `/etc/services` for list of well-known ports.

This is the file that is used to map from numeric ports to service names in Linux.

Note that each transport protocol has its own *"port space,"* so TCP port 53 is different from UDP port 53.

Most servers use only TCP or UDP, but a few are capable of using both (e.g., DNS).

Most servers/services use only a single port, but some use multiple (e.g., FTP, Bittorrent, etc.).

Ports (contd.)

Key well-known service ports:

- FTP: 20 & 21
- SSH: 22
- SMTP (email): 25
- DNS: 53
- HTTP (web): 80 and HTTPS (secure web): 443
- POP (email): 109, 110
- NETBIOS/SMB (MS file sharing+): 137, 138, 139
- IPP (printing): 631
- X11: 6000 (and up)

Network Address Translation (NAT)

Network Address Translation (**NAT**) is a technique for using a *single routable IP address* and router/gateway to handle multiple machines:

- LAN machines use non-routable (private) IP addresses
- NAT router has a single (routable) IP address
- outgoing packets:
  - router rewrites packet's local (non-routable) IP address with its own routable IP address
  - router chooses new port and rewrites IP packet's source port to it
  - router stores new port to local IP address-port mapping

continues...

Network Address Translation (contd.)

...continued

- incoming packets:
  - router uses packet's destination port to lookup local machine IP address and true destination port in its mappings
  - router rewrites packet's destination IP address and port based on mapping
  - router fowards packet to correct local machine based on mapping

Networking Basics 3: TCP/IP Protocols

1. TCP/IP Introduction
2. Routing and Ports
3. **TCP/IP Protocols**
   - **IP**
   - **UDP**
   - **TCP**
   - **SCTP**
   - **ICMP**
4. Ethernet
5. DNS

The Internet Protocol (IP)

IP creates a *virtual network* that hides the characteristics of the underlying physical network(s).

It is an **unreliable, connectionless packet delivery protocol**:

- information is sent in **packets** called **datagrams**
- does not guarantee connection to target machine
- packets may not arrive or may be duplicated
- packets may arrive out of order
- no **flow control** (e.g., if network congestion)
- transport-level protocol must handle these issues

IP (contd.)

Each IPv4 packet consists of two basic parts:

- **header**
  - 20 bytes long
  - source and destination IP addresses
  - protocol number/type: e.g., TCP, UDP, ICMP
  - length, checksum, **time-to-live** (TTL)
  - **fragmentation** flags and fragment offset
- **data section/field (body)**
  - *encapsulates* (contains) a packet for a transport-level protocol like TCP
  - total packet size limited to 65k bytes

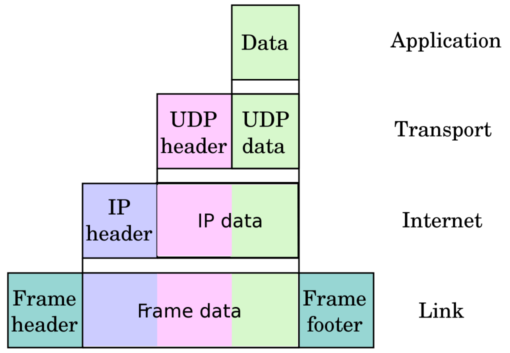
IP (contd.)

Each IPv6 packet consists of two basic parts:

- **header**
  - 40 bytes long
  - source and destination IP addresses
  - protocol number/type: e.g., TCP, UDP, ICMP
  - payload length, checksum, **hop limit**
  - **traffic class** (priority number)
- **payload**
  - *encapsulates* (contains) a packet for a transport-level protocol like TCP
  - total packet size up to 4GB (“**jumbograms**”)

Encapsulation

Higher-layer packets become the data/payload component of lower-layer packets: (figure from Wikipedia)



Transport-Level Protocols

The two most popular transport-level protocols that are layered on top of IP are:

- **Transmission Control Protocol** (**TCP**)
- **User Datagram Protocol** (**UDP**)

Other transport-level protocols:

- **Stream Control Transmission Protocol** (**SCTP**)
- **Datagram Congestion Control Protocol** (**DCCP**)

(SCTP and DCCP are supported in Linux, but not widely used.)

UDP: User Datagram Protocol

UDP is an **unreliable, connectionless, message-based** protocol:

- basically an application interface to IP
- adds concept of ports (processes) to IP datagrams
- can be viewed as a a **multiplexer/demultiplexer** for IP datagrams/packets
- i.e., receives/sends packets from/to the appropriate ports
- issues of dropped, duplicated, out of order packets must be handled by the application if important
- **message-based** (each `sendto` results in a separate datagram and each `recv` reads one datagram's data)

UDP (contd.)

UDP **datagram** ("packet") format is simple:

- source and destination ports
- length (of datagram)
- checksum
- data

UDP less costly, useful for small messages.

Standard servers that use UDP:
DNS, RPC, SNMP, NTP, etc.

Lost messages typically "handled" simply in that requests will be repeated if no response is received.

TCP: Transmission Control Protocol

TCP is a **reliable, connection-oriented, stream-based** protocol:

- acts as if there is a *dedicated two-way connection* between the two hosts
- verifies machines connected/reachable
- automatically takes care of dropped, duplicated, and out of order packets (so applications need not handle)
- supports flow control and congestion control
- **stream-based** (creates apparent byte stream, so appears to be like reading/writing from/to a file)

TCP (contd.)

TCP **segment** ("packet") format more complex:

- source and destination ports
- **sequence number**
- **acknowledgment number**
- **flags**: SYN, ACK, FIN, RST, URG, PSH
- checksum
- **window size** (number of bytes sender can receive)
- options: MSS, window size scaling, SACK, etc.
- data/payload

TCP (contd.)

The sequence and acknowledgement numbers allow:

- duplicated packets to be detected/ignored by receiver
- dropped packets to be detected and retransmitted
- out-of-order packets to be buffered and reordered

TCP is a more costly protocol than UDP, in order to guarantee connection, data reception, order, etc.

TCP is the transport protocol most widely used in servers: FTP, SSH, HTTP, POP, IMAP, SMTP, etc.

TCP (contd.)

Flags denote purpose of packet:

- **SYN** – synchronize sequence numbers (start connection)
- **ACK** – acknowledgement (part of ongoing connection)
- **FIN** – finish/close
- **RST** – reset connection (i.e., error)
- **URG** – urgent pointer is valid
- **PSH** – push (unsent) data to application

Only certain flag combinations are valid, and illegal combinations are often used to probe machines for servers and OS version.

TCP (contd.)

Three-way handshake establishes TCP connection:

1. Process 1 sends: SYN w/seq=n
2. Process 2 sends: SYN+ACK w/seq=m, ack=n+1
3. Process 1 sends: ACK w/ack=m+1

Four-way handshake terminates connection:

1. Process 1 sends: FIN
2. Process 2 sends: ACK
3. Process 2 sends: FIN
4. Process 2 sends: ACK

TCP (contd.)

TCP ports can be in many **states**:

- **LISTEN** – server end only, waiting for connection
- **ESTABLISHED** – after connection established
- **CLOSE_WAIT** – received FIN, sent ACK, waiting for local process to terminate to send FIN
- **TIME_WAIT** – sent final closing ACK, but waiting to be certain no duplicates will be received, by default wait 2MSL (twice the **Maximum Segment Lifetime**—i.e., the maximum roundtrip transmission time)
- **CLOSED**

SCTP

SCTP (**Stream Control Transmission Protocol**) is a relatively new IP transport protocol (2000).

It combines features of TCP and UDP, but also has several unique features:

- **message-based** like UDP
- **reliable** like TCP
- supports either **connection-oriented** or **connectionless** mode
- able to read *out of order data* (avoids blocking issues)
- **multi-homing**: multiple IP addresses for endpoints
- **multi-streaming**: multiple streams within connection
- four-way init handshake with cookies prevents attacks

Internet Control Message Prot. (ICMP)

ICMP packets are used to report routing errors and send basic routing information.

ICMP acts like a transport-level protocol (layered on IP) but is an integral part of IP.

Key ICMP message types:

- **Echo Request** – `ping` request
- **Echo** – `ping` response
- **Desination Unreachable** – router cannot route packet
- **Time Exceeded** – TTL expired, used by `traceroute`
- **Redirect** – change routing (very dangerous since can be abused, so routers often set to ignore)

Networking Basics 4: Ethernet

1. TCP/IP Introduction
2. Routing and Ports
3. TCP/IP Protocols
4. **Ethernet**
   - **ethernet**
   - **ARP**
5. DNS

Ethernet

Ethernet is the most commonly used physical networking technology for LANs.

Ethernet operates at the data link and physical network layers.

Each Ethernet device is identified by a **MAC** (**Media Access Control**) address:

- 48 bit number
- written in hex: AA:BB:CC:DD:EE:FF
- permanently burned into each Ethernet device (NIC)

Ethernet (contd.)

Ethernet sends data in **frames**:
- data/payload (e.g., IP datagram)
- destination and source MAC addresses
- error-detecting info (CRC)

Payload limits:
- standard is just 1500 bytes.
- IP datagrams larger than this must be **fragmented**
- **GigE** devices may support **jumbo frames** (up to 9000 bytes)

Maximum Transmission Unit (MTU):
the size in bytes of the largest data packets that can be transmitted over a network connection.

Ethernet (contd.)

Ethernet is actually a *family* of different wiring and signaling variants.

Twisted pair copper wire can handle four speeds:

- 10 Mbits/s (original 1980 spec speed)
- 100 Mbits/s, known as **Fast Ethernet**
- 1000 Mbits/s or 1 Gbits/s, **gigabit Ethernet** (GigE or GbE)
- 10000 Mbits/s or 10 Gbits/s, **10 gigabit Ethernet** (10 GigE or 10GbE or 10GE)

Ethernet (contd.)

The Ethernet connection between two devices can be **half duplex** or **full duplex**:

- **full duplex** means data can be simultaneously transmitted in both directions between two devices
- **half duplex** means data can be transmitted in only one direction at a time between two devices.
- most modern Fast Ethernet devices/cables support both
- GigE and up is full duplex only

Modern Ethernet devices typically support **autonegotiation**: allows connected devices to arrive at compatible settings for speed, duplex, and flow control.

Address Resolution Protocol (ARP)

ARP is the protocol that is used to convert IPv4 addresses to physical network addresses.

On the local physical network, hosts are known and addressed by a physical address (e.g, MAC address for Ethernet) rather than their IP address.

When we wish to send a packet to host x.y.z.w that is on the local network, the OS networking software must determine the physical address.

An OS maintains an IP to physical address mapping table, called the **ARP cache**.

ARP (contd.)

If the mapping for the desired IP address is not in the ARP cache, it must be determined:

1. host needing mapping sends *broadcast ARP Request frame* on the local network
2. machine that recognizes its IP address in the request sends back an ARP Reply containing its physical hardware address (e.g., MAC address)
3. mapping info is put into the ARP cache on requesting machine
4. info in the ARP cache is expired periodically to deal with network configuration changes

Networking Basics 5: DNS

1. TCP/IP Introduction
2. Routing and Ports
3. TCP/IP Protocols
4. Ethernet
5. **DNS**
   - **hostnames**
   - **the Domain Name System**
   - **DNS lookups**
   - **DNS records**
   - **Linux networking tools**

Hostnames and DNS

While TCP/IP identifies hosts using numeric IP addresses, humans do better with names.

The **Domain Name System** (**DNS**) was created to map between alphanumeric hostnames and numeric IP addresses.

Hostname: a designator for a particular host (machine) on the network (format depends on the naming system used, such as DNS, SMB, etc.).

Domain name: DNS designation for a realm of authority (e.g., siu.edu), but note that in DNS a hostname is also considered to be domain name.

Hostnames and DNS (contd.)

Fully qualified domain name (FQDN): domain name that uniquely identifies domain within DNS tree (so starts from top level).

A FQDN identifies a host/domain on the Internet, without need for any context: e.g., `www.cs.siu.edu.`
(trailing dot technically required for FQDN).

Domain names are written starting with highest level domain at end (on right) and lowest level at start (on left).

FQDN has a toplevel domain at far right: `.` (dot).

Hostnames and DNS (contd.)

Host/domain names may be *incomplete*, relying on domain context to uniquely identify a host: e.g., `www`

Target host will depend on domain context, e.g., `www.cs.siu.edu` or `www.siu.edu`, etc.

FQDNs are limited to 255 bytes total and 63 bytes for each (dot separated) **label**.

Labels contain only (Latin) alphabet characters (a-z), digits (0-9), plus the dash/hyphen character (-), and are *not case sensitive*.

ICAAN has approved an *internationalized character* set called **IDNA** (using **Punycode**).

DNS

DNS is a **hierarchical, distributed database**:

- tree structured (hierarchical), **root name servers** at top
- mappings distributed across servers, no single server has all mappings

There is to be at least one **name server** that contains the hostname to IP address mapping for each host on Internet (this is the **authoritative name server** for the hostname).

Each host must also have one or more **local name servers** that it can contact to resolve hostnames.

The so-called **resolver** code is a set of routines that handle requests to map a hostname to an IP address, using local resources (`/etc/hosts`) or initiating DNS lookups to the local name servers.

DNS Lookups

The DNS lookup procedure can be fairly complex.

We will consider an example of trying to connect to `www.cs.siu.edu` from a computer outside of the SIU network:

- the web browser code makes a "**resolver**" call to get the IP address for the hostname `www.cs.siu.edu`
- the resolver code (part of the OS networking stack) sends a DNS query to (first listed) **local name server**
- if local name server knows the IP address (because it is authoritative for the hostname or because it has cached it from a previous lookup), it simply returns the IP address
- if local name server does *not* know the address, it (the name server) automatically makes a series of queries until it can return the IP address...

DNS Lookups (contd.)

...continued...

- the local name server first queries a **root name server** to get a name server for the **toplevel domain** (TLD), `.edu`
- once the TLD server is known, a query is sent to it
- the TLD name server responds with the name server for the next lower domain level, `.siu.edu`
- the local name server then queries that nameserver, which may respond with the IP address if it is authoritative for the host or if it has the address cached, otherwise it will respond with another, lower level name server, `cs.siu.edu`
- the process continues until the local name server reaches an authoritative name server and gets the IP address (or finds that there is no mapping, i.e., the hostname is invalid)

DNS Records

Name servers store information as records, of various **types**:

- **A** – hostname to IPv4 mapping
- **AAAA** – hostname to IPv6 ("quad A")
- **CNAME** – map aliases to canonical name
- **MX** – mail servers for the domain
- **NS** – delegate to subdomain name servers
- **PTR** – IPv4 address to hostname mapping (reverse lookup)

Key organizations/sites:
icann.org, iana.org, root-servers.org

Key Linux Networking Tools

- `netstat` – show network connections
- `ifconfig` – show/configure network interfaces
- `route` – show/configure the IP routing table
- `ip` – show/manipulate network devices, routing, tunnels
- `ping/ping6` – send ICMP echo requests to hosts
- `traceroute` – trace packet route to host
- `arp` – show/configure ARP cache
- `iptables` – show/configure Linux firewall
- `wireshark` – network packet sniffer
- `nmap` – network port scanner