

Lecture 13:

Naming

CSE 123: Computer Networks
Aaron Schulman





Lecture 13 Overview

- Finish IPv6
- Discovering addresses (DHCP)
- User-friendly names (DNS)



IPv6 vs IPv4 header

Ver.	Traffic Class	Flow Label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

Ver.	Hdr Len	Type of Service	Total Length	
Identification			Flg	Fragment Offset
Time to Live	Protocol		Header Checksum	
Source Address				
Destination Address				
Options...				

- Gray bits are unique to each header
- Changes
 - Eliminate fragmentation-related fields
 - Eliminate header checksum
 - Added flow label
 - Quadruple size of addresses
 - IPv6 header (40 bytes) vs IPv4 (20 bytes)



Extension Headers

- Effectively a linked list of headers
 - ◆ The “next header” field is the pointer
- Two different types
 - ◆ [Destination](#), intended for the IP end point. E.g.,
 - » 44: Fragmentation Header (it’s baaack!)
 - » 43: Routing header (dictates how to route the packet)
 - ◆ [Hop-by-hop](#), processed by each node on the path



IPv6 Transition is slow

- Need to support both protocols at the same time
 - ◆ Complicated... if a destination has both a IPv4 and IPv6 address which to use?
- Less need in developed world -> slower adoption
- That said
 - ◆ All major operating systems now support IPv6
 - ◆ All major router vendors
 - ◆ US Mobile carriers (e.g., Tmobile, Verizon, etc)
 - ◆ Offered as option by many US ISPs
- In your lifetime it is likely that IPv6-based addressing will start to dominate

Addressing Summary



- You can't route efficiently on flat address spaces
 - ◆ You'd need a table the size of all hosts on the Internet
 - ◆ You'd need to send updates about that table to everyone
- Network-layer addressing is done hierarchically
 - ◆ Routing prefix + host suffix
 - ◆ Originally, this split was done statically (class-based addressing)
 - ◆ Now it is done dynamically (CIDR)
 - ◆ Requires more complex forwarding table lookup
 - ◆ Allows contiguous chunks of address space to be aggregated (for the purposes of routing) into fewer prefixes



Whence come IP Addresses?

- You already have a bunch from the days when you called Jon Postel and asked for them (e.g. BBN)
- You get them from another provider
 - ◆ E.g. buy service from Sprint and get a /24 from one of their address blocks
- You get one directly from a routing registry
 - ◆ ARIN: North America, APNIC (Asia Pacific), RIPE (Europe), LACNIC (Latin America), AFRINIC (Africa)
 - ◆ Registries get address from IANA (Internet Assigned Numbers Authority)



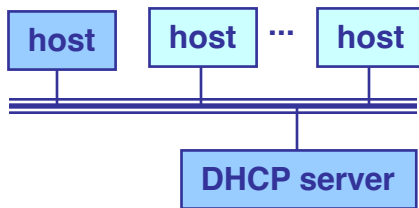
How Do You And I Get One?

- Well from your provider! Or you can buy them from ARIN
 - ◆ https://www.arin.net/resources/guide/ipv6/first_request/
- But how do you know what your IP is when you get it from a provider?
- Manual configuration
 - ◆ They tell you and you type that number into your computer (along with the default gateway, DNS server, etc.)
- Automated configuration
 - ◆ Dynamic Host Configuration Protocol (DHCP)



Bootstrapping Problem

- Host doesn't have an IP address yet
 - ◆ So, host doesn't know what source address to use
- Host doesn't know who to ask for an IP address
 - ◆ So, host doesn't know what destination address to use
- Solution: shout to discover a server who can help
 - ◆ Install a special server on the LAN to answer distress calls



DHCP



- Broadcast-based LAN protocol
 - ◆ Host broadcasts “DHCP discover” on LAN (e.g. Ethernet broadcast)
 - ◆ DHCP server responds with “DHCP offer” message w/IP address
 - ◆ Host requests IP address: “DHCP request” message
 - ◆ DHCP server sends address: “DHCP ack” message w/IP address
- Easy to have fewer addresses than hosts (e.g. UCSD wireless) and to *renumber* network (use new addresses)
- What if host goes away (how to get address back?)
 - ◆ Address is a “lease” not a “grant”, *has a timeout*
 - ◆ Host may have different IP addresses at different times?



Layers of Identifiers

- **Link-layer (MAC) address** (e.g., 58:B0:35:F2:3C:D9)
 - ◆ Used by *network interfaces* to identify interesting frames
 - ◆ Unique, hard-coded identifier “burned into” network interface chipset
 - ◆ Flat name space (of 48 bits in Ethernet)
- **IP address** (e.g., 128.54.70.238)
 - ◆ Used by *routers* to forward packets
 - ◆ Unique, topologically meaningful locator
 - ◆ Hierarchical namespace of 32 bits
- **Host/domain name** (e.g., www.ucsd.edu)
 - ◆ Used by *humans* to specify host of interest
 - ◆ Unique, selected by host administrator
 - ◆ Hierarchical, variable-length string of alphanumeric characters



Naming Hierarchy for Scale

- MAC addresses: **58:B0:35:F2:3C:D9**
 - ◆ **OUIs**: assigned to vendors by the IEEE
 - ◆ **Adapters**: assigned by the vendor from its block
- IP addresses: **128.54.70.238**
 - ◆ **Prefixes**: ICANN, regional Internet registries, and ISPs
 - ◆ **Hosts**: static configuration, or dynamic using DHCP
- Host name: **www.ucsd.edu**
 - ◆ **Domain**: registrar for each top-level domain (e.g., .edu)
 - ◆ **Host name**: local administrator assigns to each host



Mapping Between Identifiers

- Dynamic Host Configuration Protocol (**DHCP**)
 - ◆ Automates host boot-up process
 - ◆ Given a MAC address, assign a unique IP address
 - ◆ ... and tell host other stuff about the Local Area Network
- Address Resolution Protocol (**ARP**)
 - ◆ Given an IP address, provide the MAC address
 - ◆ To enable communication within the Local Area Network
- Domain Name System (**DNS**)
 - ◆ Given a host name, provide the IP address
 - ◆ Given an IP address, provide the host name

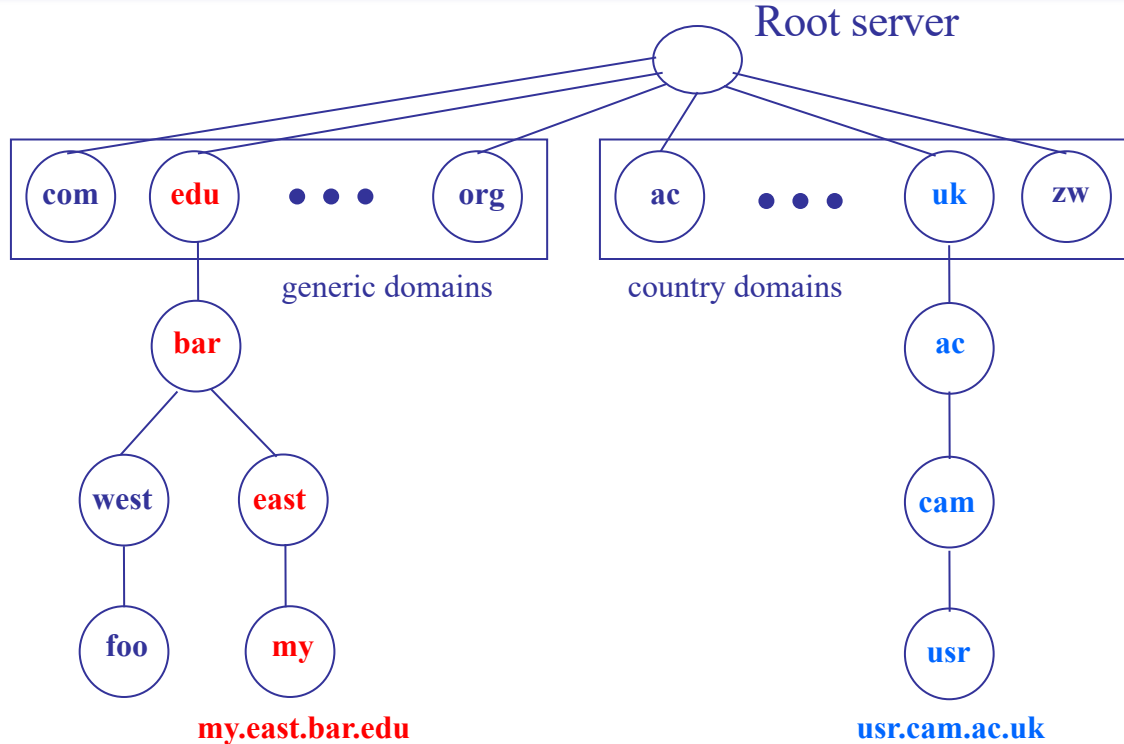


Domain Name System (DNS)

- Distributed administrative control
 - ◆ Hierarchical name space divided into zones
 - ◆ Distributed over a collection of DNS servers
- Hierarchy of DNS servers
 - ◆ Root servers
 - ◆ Top-level domain (TLD) servers
 - ◆ Authoritative DNS servers
- Performing the translations
 - ◆ Local DNS servers
 - ◆ Resolver software



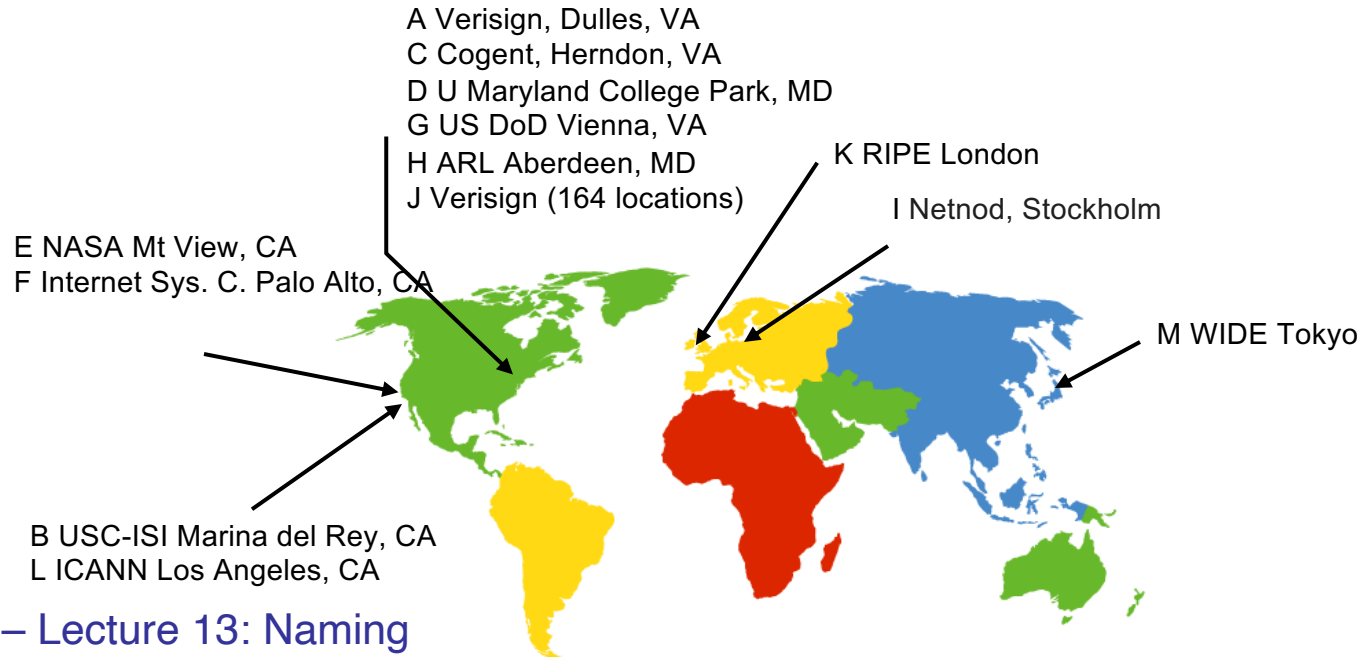
DNS: Distributed Database



DNS Root Servers



- 13 root servers (see <http://www.root-servers.org/>)
 - ♦ Labeled A through M



Using DNS

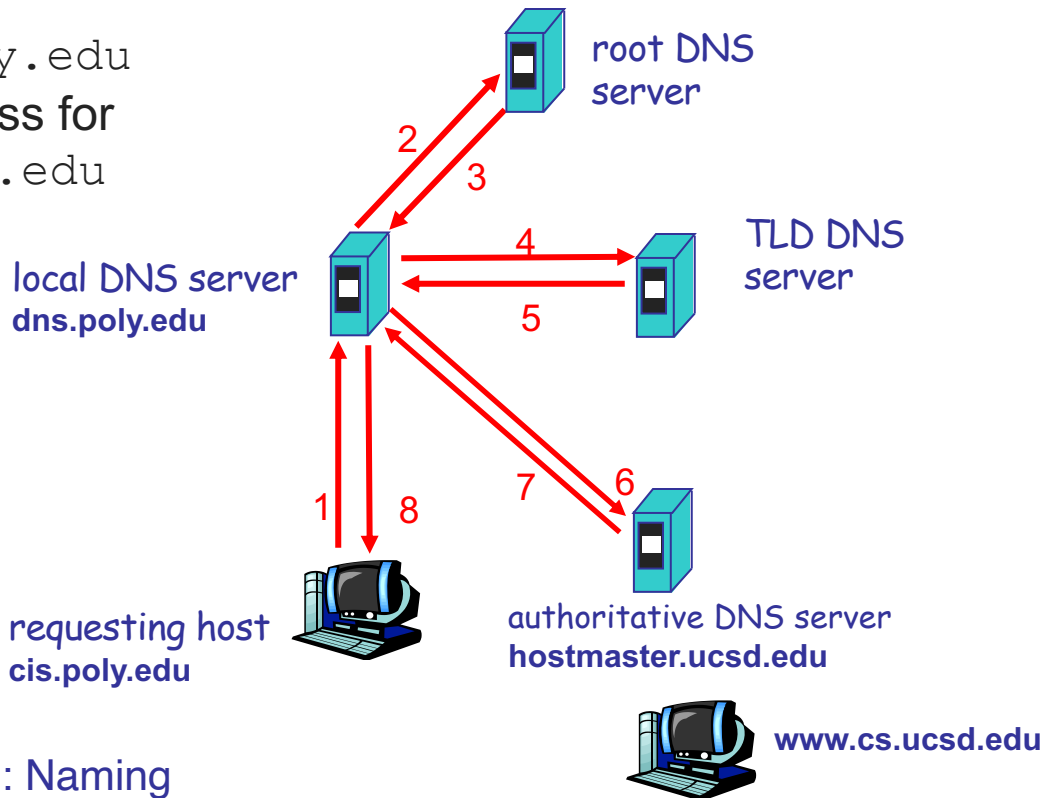


- Local DNS server (“default name server”)
 - ◆ Usually near the end hosts who use it
 - ◆ Local hosts configured with local server (e.g., `/etc/resolv.conf`) or learn the server via DHCP
- Client application
 - ◆ Extract server name (e.g., from the URL)
 - ◆ Do `gethostbyname()` to trigger resolver code
- Server application
 - ◆ Extract client IP address from socket
 - ◆ Optional `gethostbyaddr()` to translate into name

Example



Host at `cis.poly.edu`
wants IP address for
`www.cs.ucsd.edu`





A note on DNS reliability

- DNS servers are replicated
 - ◆ Name service available if at least one replica is up
 - ◆ Queries can be load balanced between replicas
- UDP used for queries
 - ◆ Need reliability: must implement this on top of UDP
 - ◆ Try alternate servers on timeout
 - ◆ Exponential backoff when retrying same server
- Cache responses to decrease load
 - ◆ Both at end hosts and local servers
- DNS used to be cleartext, now can be encrypted/validated

For Next Time



- Discussion Today – Discuss Homework 2
- Keep reading section 3.3.6 (ARP)