TCP/IP Networks

CSC 348·648

WAKE FOREST
UNIVERSITY
Department of Computer Science

Spring 2013

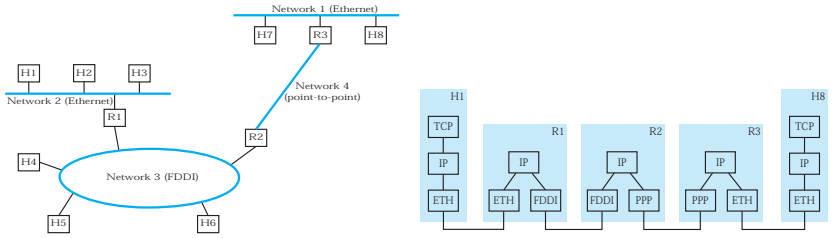
Network Review

- Physical layer
  - Concerns the physical transmission bits
  - Example digital encoding is +5 volts for a 1 and -5 volts for a 0
- Data link layer
  - Frames data and controls line access
  - Assumes all computers are *local* (**no** forwarding)
  - Example data link protocol is CSMA/CD
- Network layer
  - Allows datagrams (packets) to be forwarded to other networks
  - A router receives packets on one line and transmits on another
  - Example network layer is Internet Protocol (IP)

IP Operation

Assume a user has data to send over the network

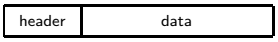
- IP receives data from upper layer
- Creates **packets** containing data
- Sends packets to *next-hop* (forward to next machine)
- Process repeats until the destination is reached



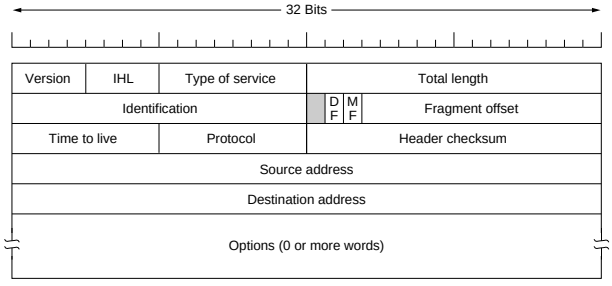
How does the next-hop (router) know who the packet belongs to and who is the next hop?

IP Protocol Datagrams

- Datagram (packet) consists of a header and data



- Header consists of: 20 byte *fixed part* and an *optional part*



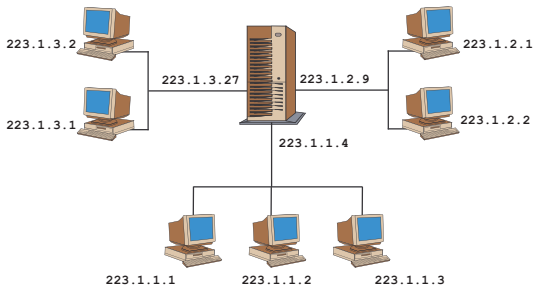
  - Header contains different *control fields*
  - We will focus on the address

IPv4 Addresses

- Every host or router (actually interface) has a unique IP address
  - IP addresses are 32 bits long (IP version 4)
- *Dotted-decimal notation* is used to represent an address
  - Each byte is represented via a decimal number
  - `193.32.216.9` $\Rightarrow$ [`11000001 00100000 11011000 00001001`]
- Addresses are hierarchical and encode two numbers
  - **Network** - identifies the network
  - **Host** - identifies the machine in the network

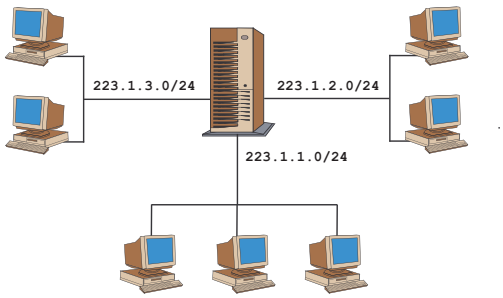
IP Network Example

Consider one router and seven hosts (*one address per interface*)



- Three hosts at bottom have similar addresses, `223.1.1.`x
  - The leftmost 24 bits they share is the **network** portion
  - Remaining 8 bits is the **host** portion
- Hosts of `223.1.1.`x form a network, interconnected via a LAN
  - The network address is `223.1.1.0/24`

  - The /24 is also called the **network mask** or **network prefix** (indicates the leftmost 24 bits are the network address)
  - Any additional host attached to this network must have a unique address of the form $223.1.1.x$
- Other networks have a similar structure

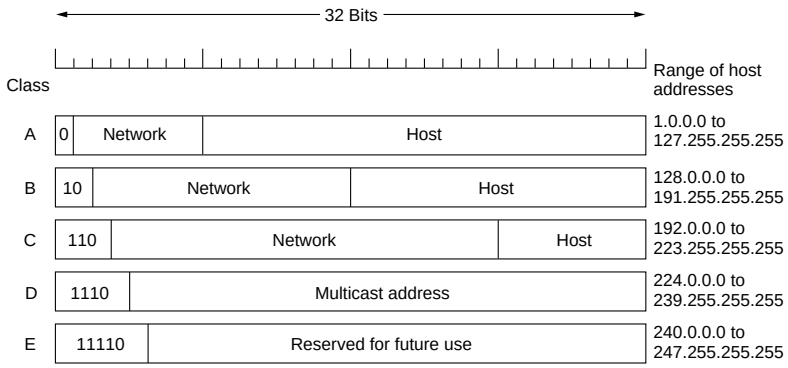


  - N.B. the router interconnects different networks

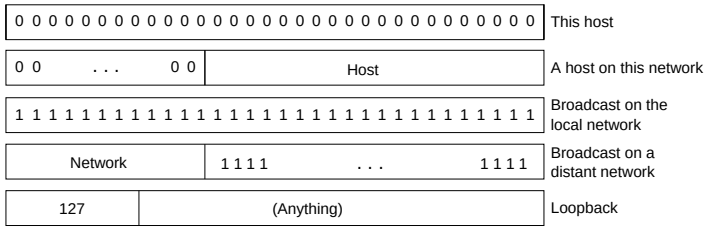
IPv4 Address Classes

The original Internet architecture defined 5 different IP address classes

- This is also know as **classful addressing**
- Given an IP address you can determine the network mask



IP Addresses with Special Meanings



- `0.0.0.0` only used by a host when booting
- All zeroes for the network number, refers to the local network
  - If `223.1.1.0/24` is the network and I am `223.1.1.52`, locally I can be reached using `0.0.0.52`
- Address of all ones is the broadcast address for the local network

  What is the dotted-decimal address?

- Address with the proper network number, and all ones for the host number allows host to broadcast to a different network
  - If `223.1.1.0/24` is a distant network, then `223.1.1.255` broadcasts to all hosts at the network
  - This will be used to create a simple Denial of Service (DoS) attack
- 127.x.y.z is reserved for loop-back testing
  - Packet is never placed on the network, processed locally

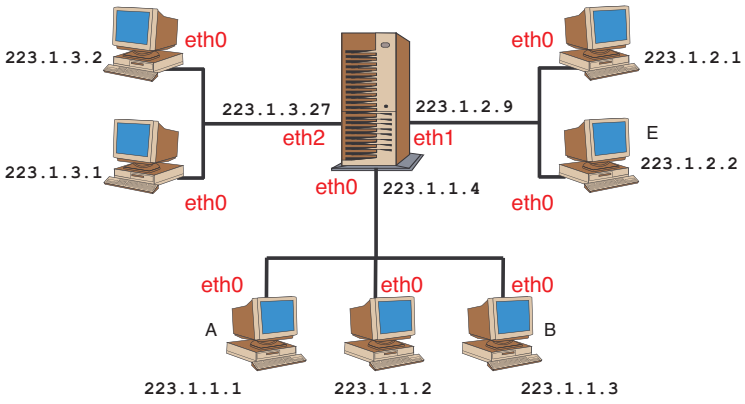
Routing Tables

How does a source host send a datagram to a destination host?

- The IP layer maintains a **routing table** in memory
  - Remember, routing tables are *next hop* oriented
  - Multiple hop paths are not recorded
- Each entry in the routing table has the following information[a]
  1. Destination address, either *host* or *network* address
  2. IP address of the *next-hop router*
  3. Flags specifying if next hop is host or network
  4. Identification of the interface the datagram should be passed to (e.g. multiple Ethernet cards attached)

[a] Abbreviated list of items, more later.

Example Routing Tables



- In the diagram, each interface (Ethernet card) is labeled (in red)
- For example, the router has 3 interfaces (eth0, eth1, and eth2)
  - Each interface must be uniquely identified, since it attaches a unique network

- An abbreviated routing table for host A would be

Routing Table for A		
Destination	**Next Hop**	**Interface**
223.1.1.0/24		eth0
223.1.2.0/24	223.1.1.4	eth0
223.1.3.0/24	223.1.1.4	eth0

  - First entry indicates 223.1.1.0/24 is the local network
  - The second and third entries indicate datagrams for destinations on network 223.1.2.0/24 or 223.1.3.0/24 must be sent to 223.1.1.4
  - eth0 is the Ethernet interface (only one card on A)

Each network is represented with one entry, how many would be required if each host had a separate entry?

- An abbreviated routing table for the router would be

Routing Table for Router		
Destination	**Next Hop**	**Interface**
223.1.1.0/24		eth0
223.1.2.0/24		eth1
223.1.3.0/24		eth2

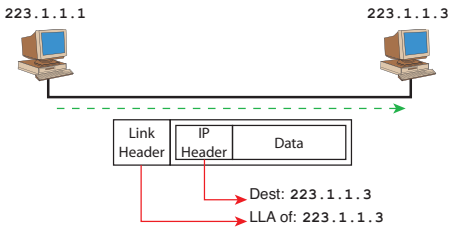
  - First entry indicates 223.1.1.0/24 is local on eth0
  - Second entry indicates 223.1.2.0/24 is local on eth1
  - Third entry indicates 223.1.3.0/24 is local on eth2

IP Routing Steps

- IP routing performs the following actions
  1. Search routing table for complete destination address, if found send packet to the next-hop entry
  2. Search routing table for an entry that matches the destination network number, if found send packet to the next-hop entry
     - Must take into account possible subnet mask
  3. Search for *default* entry, if found send to next-hop router
- IP search order is, host address → host network → default
- If all the steps fail, then the datagram is not deliverable

Routing Example: A → B

Assume A (223.1.1.1) sends datagram to B (223.1.1.3)



- There is no host entry for 223.1.1.3
- There is a network entry for 223.1.1.0/24
- A link layer frame (containing the datagram) is created and addressed to the link layer address of 223.1.1.3

  We are at layer 3, how do we get a layer 2 address?

- Ethernet frame is sent and received by host B

Routing Example: A → E

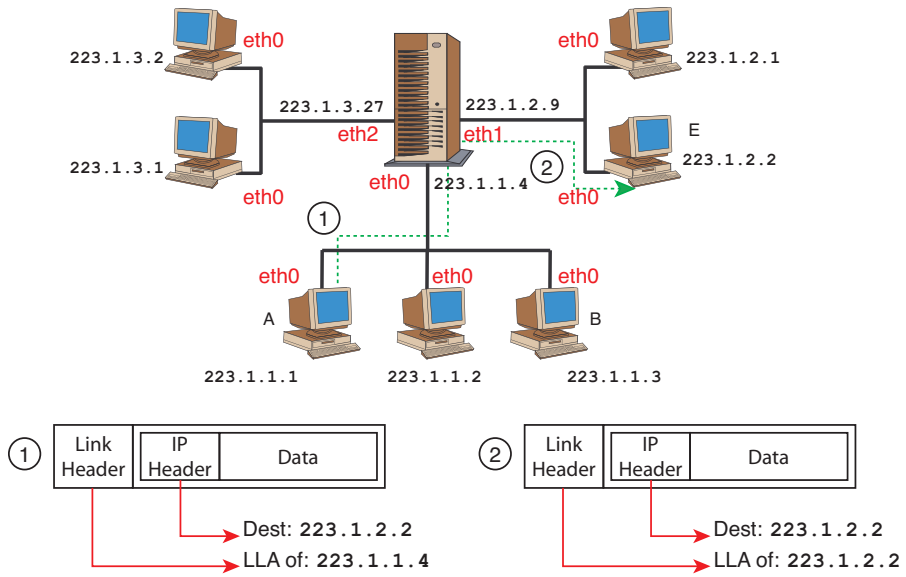
Assume A (223.1.1.1) sends datagram to E (223.1.2.2)

Routing Table for A		
Destination	**Next Hop**	**Interface**
223.1.1.0/24		eth0
223.1.2.0/24	223.1.1.4	eth0
223.1.3.0/24	223.1.1.4	eth0

Routing Table for Router		
Destination	**Next Hop**	**Interface**
223.1.1.0/24		eth0
223.1.2.0/24		eth1
223.1.3.0/24		eth2

- Host A finds entry for 223.1.2.0/24 network
  - Requires sending packet to 223.1.1.4
- Host A creates and sends link-layer frame (containing datagram) addressed to the link-layer address of 223.1.1.4
  - Therefore, the next-hop entry is used for the link-layer address
  - IP destination address remains unchanged

- Router 223.1.1.4 receives frame and removes datagram
  - Destination address is 223.1.2.2
  - Router is allowed to forward datagrams
- Router finds entry for 223.1.2.0/24 network
  - This is directly connected via eth1
  - Datagram will be forwarded
- Router creates and sends link-layer frame (containing datagram) addressed to the link-layer address of 223.1.2.2 on eth1
- Frame received by host E, datagram removed and processed
- N.B. operation of host and router are equivalent, except routers are allowed to forward datagrams

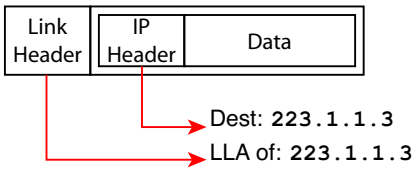


Static versus Dynamic IP Routing

- Already know how IP packets are routed using routing tables
  - *How were the entries generated... statically or dynamically?*
- In static routing entries are manually adjusted
  - Acceptable for small networks
- For larger networks, dynamically change table entries
  - Routes should change based on network conditions
  - Allow routers to pass route information to one another
  - Use variations of Bellman-Ford and Dijkstra's
- N.B. This will **not** change the way IP datagrams are routed, just how/when the routing table contents change

Layer 3 Addresses versus Layer 2 Addresses

- Sending IP datagrams requires knowledge of the link-layer address
  - The IP datagram is placed inside a link-layer frame then sent



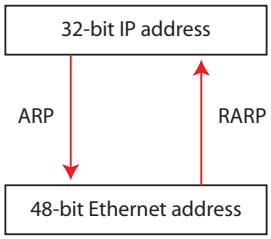
- Hosts are attached to the network via an interface card
  - The interface card is a layer 1 and 2 device
  - Can only understand **LAN addresses** (also called **hardware address** or **MAC address**)

- For example, every Ethernet board has a 48-bit Ethernet address
  - In DOS run `winipcfg` to determine the Ethernet address
  - Card manufacturers given a *block* of Ethernet addresses
  - Assignments given in RFC 1700 or a more up-to-date at `http://standards.ieee.org/develop/regauth/oui/public.html`
- Similar to IP, hardware address of all 1's is for broadcast

How do IP addresses get mapped to data-link addresses?

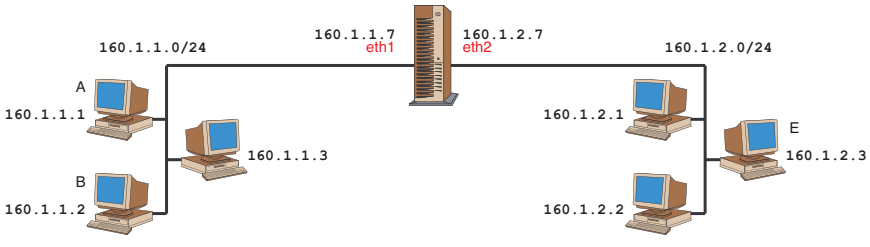
Address Resolution

- Address resolution - Mapping between two different address forms
  - IP addresses and data-link addresses
- Two different protocols in IP
  - Address Resolution Protocol (ARP)
  - Reverse Address Resolution Protocol (RARP)



ARP

- Dynamic mapping of IP address → hardware address [RFC 826]
- Assume A needs the hardware address of host B



- Host A broadcasts an ARP request
  - Ethernet address of all 1's is the broadcast address
  - Host asks *"Who owns IP address 160.1.1.2?"*
- Host B would reply with its hardware address

- ARP is intended for broadcast (shared medium) networks
  - Not needed for point-to-point links
- Broadcast at boot-time
  - When a machine connects, broadcast its mapping (ARP reply)
  - All other hosts on network (or subnet) will know the IP and hardware address of the new machine

ARP Optimization

There are various ways to improve the efficiency of ARP

- ARP Tables
  - Keep an ARP table for most recently used IP addresses
  - Entries can time-out
- Hey, this is lots of fun...
  - When your laptop is first connected to the network, enter the DOS command, `arp -a`
  - Ping a machine on our network, `ping www.wfu.edu`
  - Issue the `arp` command again, should see hardware (physical) address for the machine

    This may be a proxy ARP, how can you tell?

  - Wait a few minutes and issue the `arp` command again

ARP Security

- ARP introduces a security problem

  How could this protocol be used to obtain important information from users?

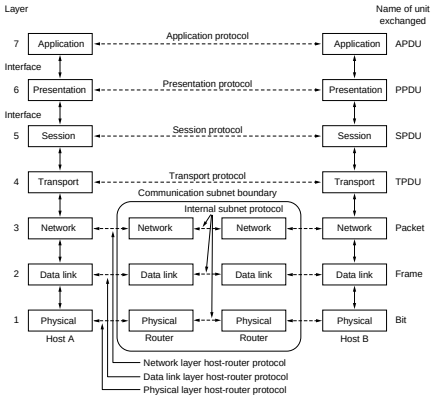
- A common way around this security issue is to use **static ARP**
  - Permanent ARP table created
  - *So what is ARPish about this solution...*

Transport Layer

- Reliable, cost-effective data transport from source to destination
  - End-to-end protocol, only implemented at hosts
- *Transport entity* provides services to upper layers
  - Type of service, QoS, data transfer, connection management, and flow control
- Two types of service available
  - Connection-oriented - Establishment, maintenance, and termination of connection
  - Connectionless - Unreliable service (delivery not guaranteed), reduces the overhead associated with transport layer

  Network layer also has connection-oriented and connectionless, why do we specify it here?

Network and Transport Layers



- The network layer is part of the communication subnet
  - What if the network-layer is connection-oriented but unreliable?
  - What happens if a router crashes?
  - *Provide a reliable service over an unreliable network*

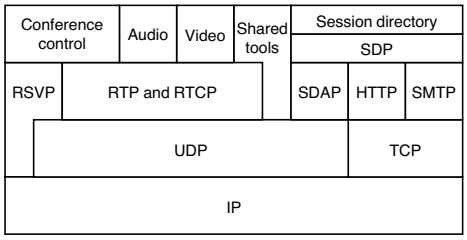
Transport Addressing

- When an application process wishes to set-up a connection to a remote application process, it must specify which one
  - `152.17.140.92` provides `http` and `ssh` services
  - Therefore network layer address is **not** sufficient
- In the Internet, transport services identified using **port numbers**
  - Some port numbers are *well-known*, 80 is for `http`
  - DOS command `netstat -a` shows ports in use

    So what is a port scan?

Internet Transport Layer

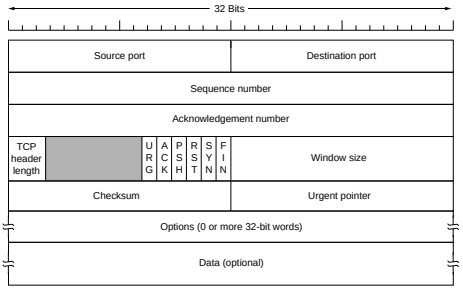
- There are two distinct Internet transport-layer protocols
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)



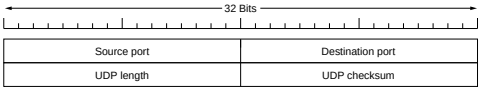
- UDP provides unreliable connectionless service
- TCP provides reliable connection-oriented service

Transport Layer Packets

- Packets have a header and data
  - Headers for TCP and UDP are different

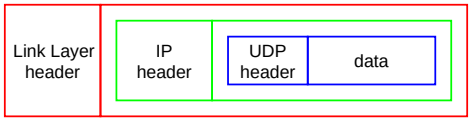


TCP header



UDP header

- At the physical layer the data looks like



When is UDP Preferred

UDP has the following benefits, making it better for some applications

- No connection establishment
  - TCP requires a three-way handshake before sending data
  - UDP sends data immediately, no initial connection delay
- No connection state
  - TCP requires state information about send/receive buffers, congestion-control, sequence numbers... *per connection*
  - UDP does not require any of this state information
- Small packet header overhead
  - UDP only adds 8 bytes of header information
- Unregulated send rate
  - Send as quickly as desired
  - Has introduced the idea of *TCP friendly* applications

When is TCP Preferred

- Transport Control Protocol (TCP) provides connection-oriented, reliable service [RFC 793, 1122, 1323, 2018, 2581]
- All connections TCP connections are full-duplex, point-to-point
  - Requires three-way handshake to establish connection
- Reliable transport service
  - Deliver all data without error and proper order
- Congestion control mechanism
  - Attempts to limit each connection to *fair share* of bandwidth (*Crouse says "très bon, l'equitabilité est tout"*)
  - **No** minimum bandwidth guaranteed
- TCP connection is a byte stream, not a message stream
  - Message boundaries are not preserved
  - Data is buffered before sent (additional delay)

Port Numbers

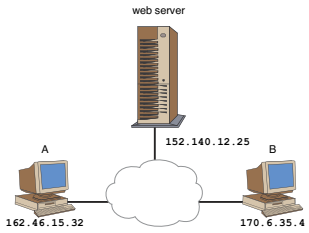
- Transport addresses are used to identify processes
  - For Internet transport layers, this called a **port number**
  - **Both** source and destination processes have a port number
- Source and destination ports *together* uniquely identify a process

  Do we really need source and destination port numbers to identify a process (single process at one end)?

- Port numbers are 16 bits, ranging from 0 to 65535
- Numbers ranging from 0 to 1023 are **well-known port numbers**
  - Reserved for use by well-known protocols
  - `http` is 80, `ftp` is 21, complete list at RFC 1700

Web Server Example

- Consider a web server and two stations connected via the Internet



- Web server runs `http` over port 80
  - Any station wishing to connect will use the IP address `152.140.12.25` and port 80
- Let station A connect to the web server
  - A free port is used for source (> 1023), for example 1123
  - Destination port is 80

- Web server creates a new process for each request (Unix `fork`)
  - Allows server to connect to multiple users simultaneously

What happens if station B connects to the web server? Station B will use the same destination port 80, how does the web server differentiate between station A and station B requests?

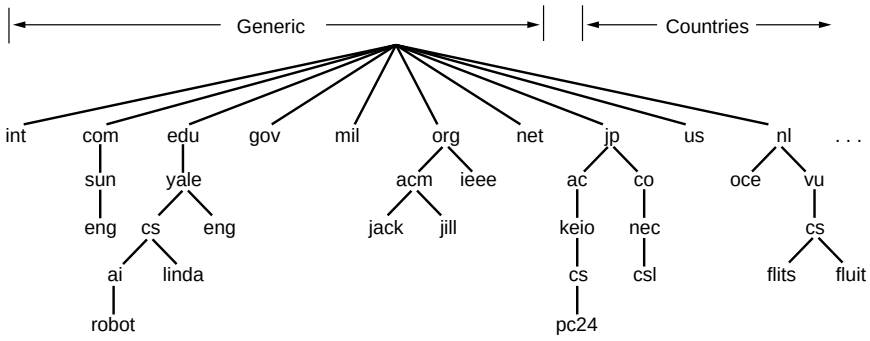
What happens if station B connects to the web server and happens to select the same source port number as A? How does the web server differentiate between station A and station B requests?

Domain Name System

- We can always use IP (network) addresses to refer to hosts
  - IP addresses are difficult to remember, `pluf@152.17.48.111`
    What happens if the machine address changes?
  - However, the network only understands IP addresses
  - Need a method to map *names* to IP addresses
- Domain Name System (DNS) provides this mapping service
  - Hierarchical, domain naming scheme
  - Given a text (ASCII) name provides the correct IP address
  - DNS protocol uses UDP for small queries and TCP for queries where the answer is over 512 bytes (*any security issues?*)

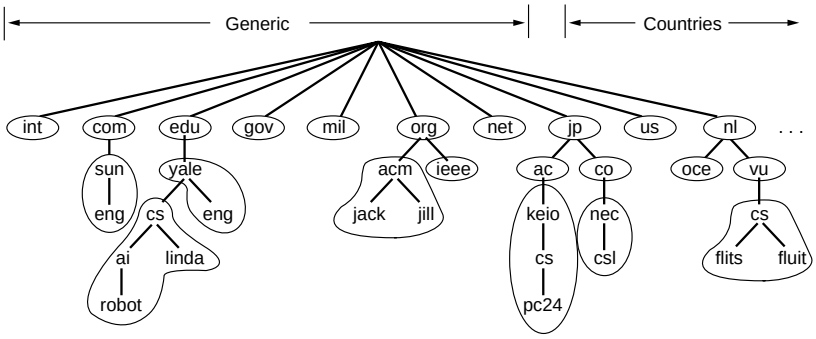
IP Address	Machine Name
152.17.48.111	mail.wfu.edu
152.17.48.77	www.wfu.edu

DNS Name Space



- Hierarchical scheme of names, can be depicted as a tree
  - 200 top-level domains, each domain covers many hosts
  - Two categories, generic and countries
- The domain manager registers new names, for example `.wfu.edu`

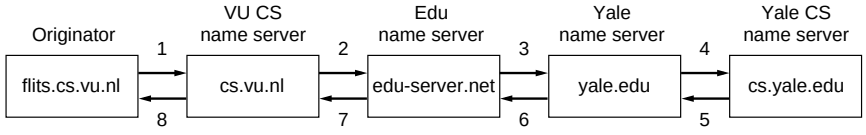
Name Servers



- DNS *name server* provides the name and IP mapping
  - A server could store the entire database, but it is not scalable
  - Will use the name hierarchy to provide scalability
- DNS is divided into non-overlapping zones
  - Zone is part of the overall tree and contains name servers
  - Name servers know about names in their zones

Name Resolution

- Assume `flits.cs.vu.nl` needs to resolve `linda.cs.yale.edu`
  1. `flits.cs.vu.nl` first searches local file/cache first
  2. If not found, asks its local name server `cs.vu.nl`
  3. If `cs.vu.nl` does not know, it asks `edu-server.net`
  4. If `edu-server.net` does not know, it asks `cs.yale.edu`
  5. `edu-server.net` should know and sends the IP



- Once `flits.cs.vu.nl` gets the IP it stores it in cache
  - This is an example of a **recursive query**, where searhes are performed on behalf of the original requester
  - Alternatively, IP of the next server to ask could have been sent

DNS Caching

- DNS requests are cached
  - Quick response for repeated translations
  - Other queries may reuse parts of lookup records for domains
- DNS negative queries are cached
  - Do not have to repeat past mistakes (misspellings)
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record