Translating Addresses

Reading: Section 4.1 and 9.1

COS 461: Computer Networks

Spring 2009 (MW 1:30-2:50 in COS 105)

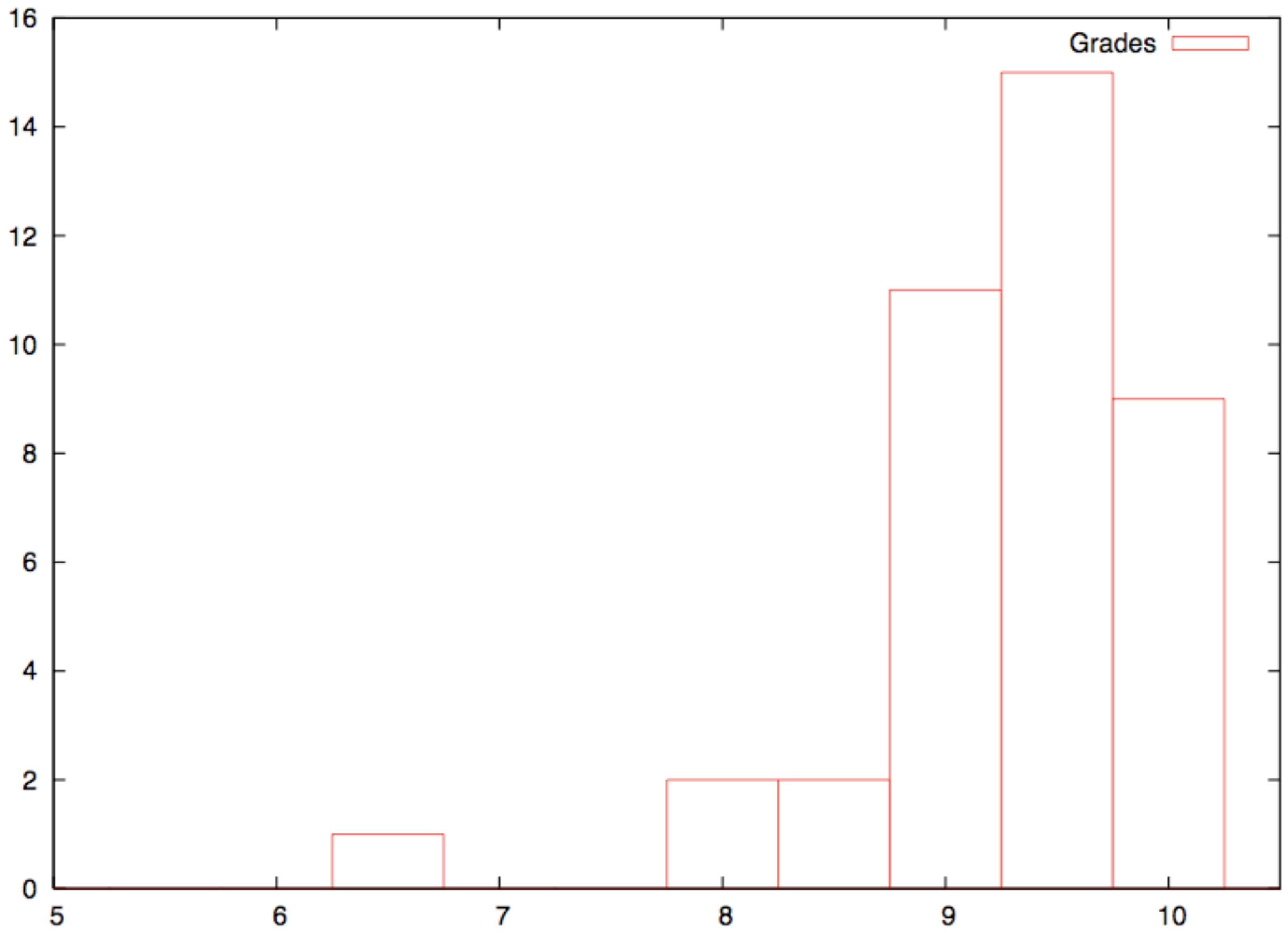
Michael Freedman

http://www.cs.princeton.edu/courses/archive/spring08/cos461/

Goals of Today's Lecture

- Three different kinds of addresses
  - Host names (e.g., www.cnn.com)
  - IP addresses (e.g., 64.236.16.20)
  - MAC addresses (e.g., 00-15-C5-49-04-A9)
- Protocols for translating between addresses
  - Domain Name System (DNS)
  - Dynamic Host Configuration Protocol (DHCP)
  - Address Resolution Protocol (ARP)
- Two main topics
  - Decentralized management of the name space
  - Boot-strapping an end host that attaches to the 'net

Grades for assignment #1



Separating Names and IP Addresses

- Names are easier (for us!) to remember
  - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

Separating IP and MAC Addresses

- LANs are designed for arbitrary network protocols
  - Not just for IP (e.g., IPX, Appletalk, X.25, …)
    - Though now IP is the main game in town
  - Different LANs may have different addressing schemes
    - Though now Ethernet address is the main game in town
- A host may move to a new location
  - So, cannot simply assign a static IP address
    - Since IP addresses depend on host's position in topology
  - Instead, must reconfigure the adapter
    - To assign it an IP address based on its current location
- Must identify the adapter during bootstrap process
  - Need to talk to the adapter to assign it an IP address

Three Kinds of Identifiers

- **Host name** (e.g., www.cnn.com)
  - Mnemonic name appreciated *by humans*
  - Provides little (if any) information about location
  - Hierarchical, variable # of alpha-numeric characters
- **IP address** (e.g., 64.236.16.20)
  - Numerical address appreciated *by routers*
  - Related to host's current location in the topology
  - Hierarchical name space of 32 bits
- **MAC address** (e.g., 00-15-C5-49-04-A9)
  - Numerical address appreciated *within local area network*
  - Unique, hard-coded in the adapter when it is built
  - Flat name space of 48 bits

Three Hierarchical Assignment Processes

- **Host name:** www.cs.princeton.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- **IP addresses:** 128.112.7.156
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP
- **MAC addresses:** 00-15-C5-49-04-A9
  - Blocks: assigned to vendors by the IEEE
  - Adapters: assigned by the vendor from its block

Mapping Between Identifiers

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

Domain Name System (DNS)

Proposed in 1983 by Paul Mockapetris

Outline: Domain Name System

- Computer science concepts underlying DNS
  - Indirection: names in place of addresses
  - Hierarchy: in names, addresses, and servers
  - Caching: of mappings from names to/from addresses
- DNS software components
  - DNS resolvers
  - DNS servers
- DNS queries
  - Iterative queries
  - Recursive queries
- DNS caching based on time-to-live (TTL)

Strawman Solution #1: Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: moving from a machine per domain to machine per user
  - Many more downloads
  - Many more updates

Strawman Solution #2: Central Server

- Central server
  - One place where all mappings are stored
  - All queries go to the central server
- Many practical problems
  - Single point of failure
  - High traffic volume
  - Distant centralized database
  - Single point of update
  - Does not scale

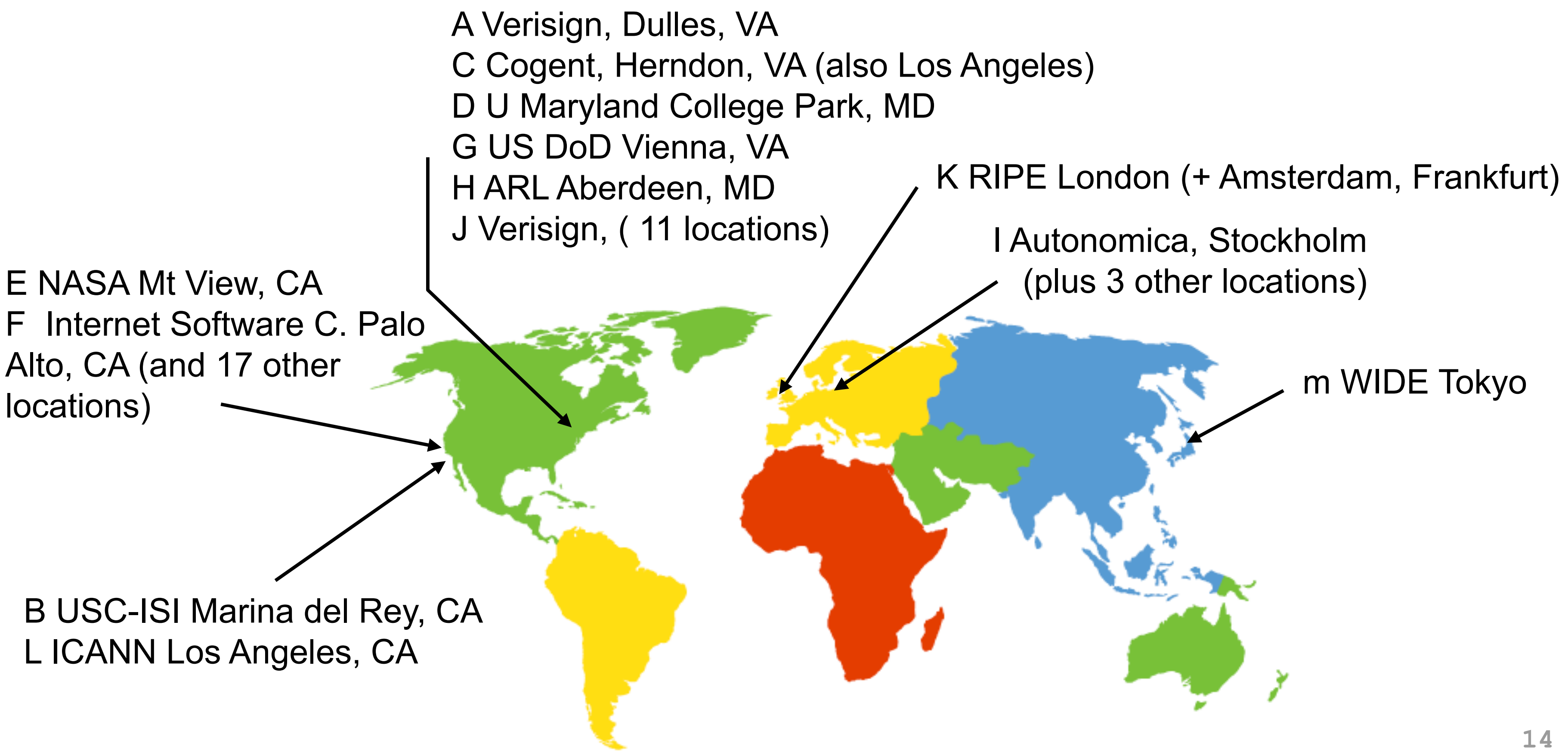
Need a distributed, hierarchical collection of servers

Domain Name System (DNS)

- Properties of DNS
  - 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 Root Servers

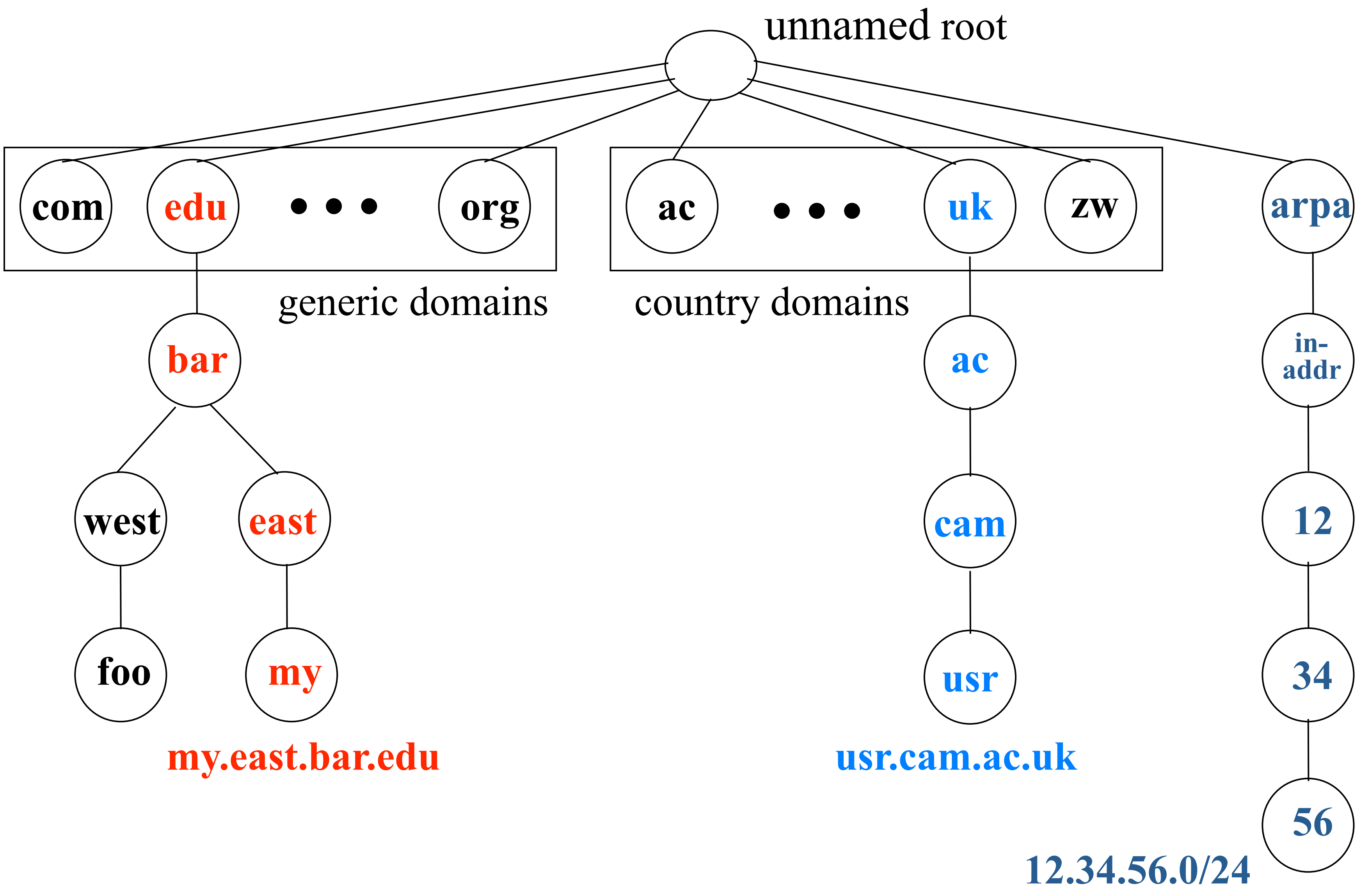
- 13 root servers (see http://www.root-servers.org/)
- Labeled A through M


A Verisign, Dulles, VA
C Cogent, Herndon, VA (also Los Angeles)
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign, (11 locations)
K RIPE London (+ Amsterdam, Frankfurt)
I Autonomica, Stockholm (plus 3 other locations)
E NASA Mt View, CA
F Internet Software C. Palo Alto, CA (and 17 other locations)
m WIDE Tokyo
B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA

TLD and Authoritative DNS Servers

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, ca, jp)
  - Typically managed professionally
    - Network Solutions maintains servers for “com”
    - Educause maintains servers for “edu”
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization’s servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

Distributed Hierarchical Database

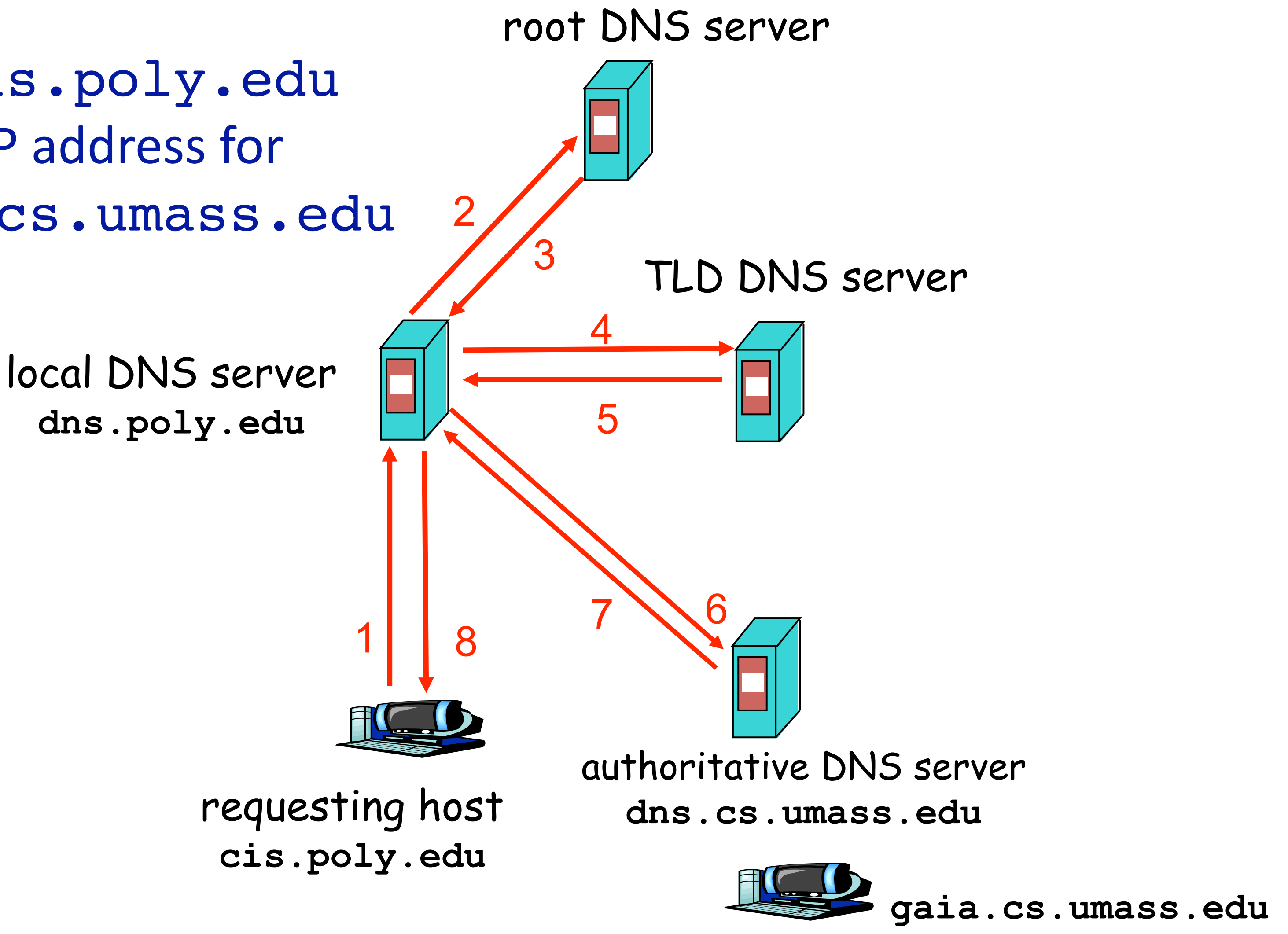


Using DNS

- 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

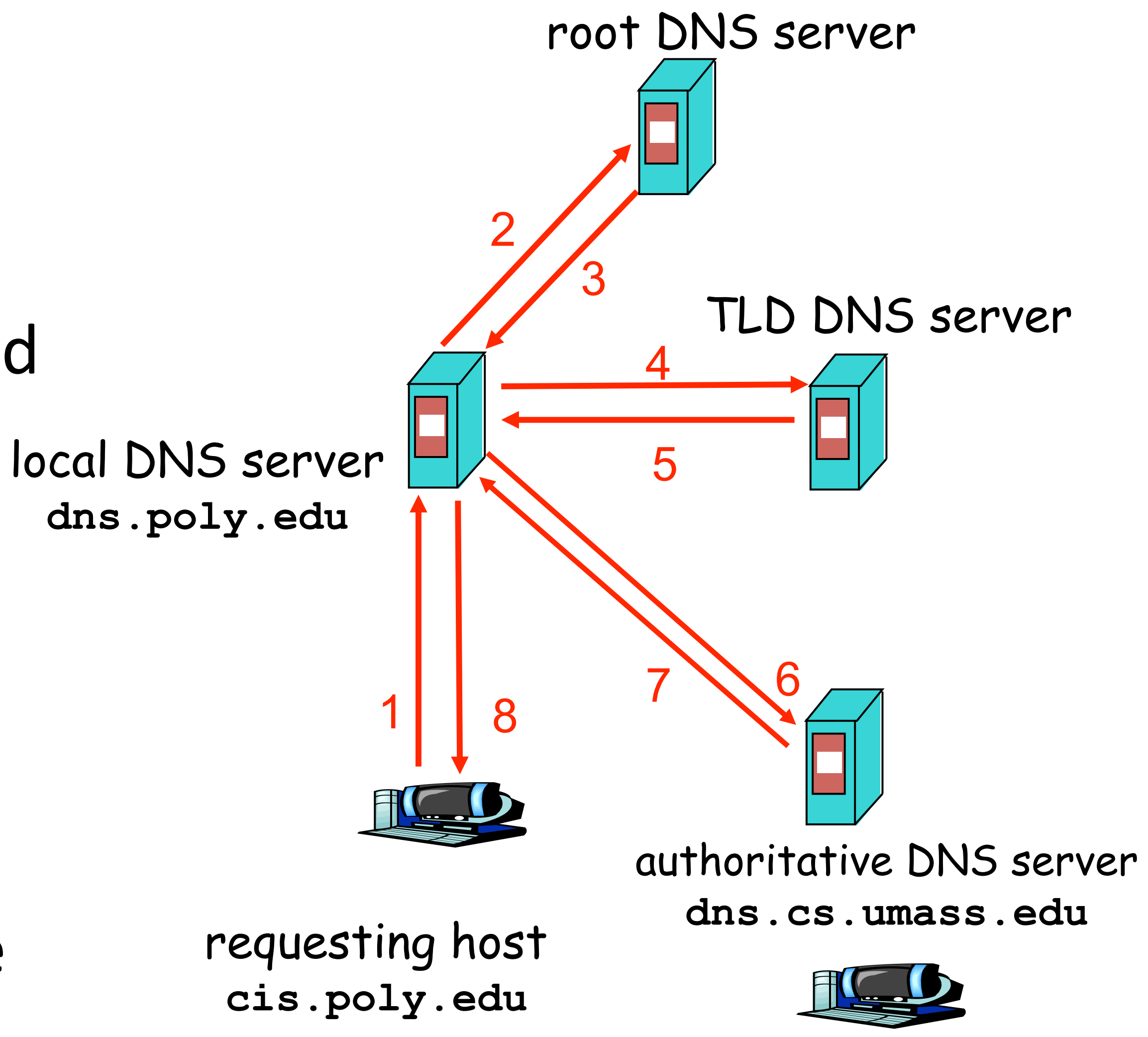
Example

Host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`



Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs



DNS Caching

- Performing all these queries take time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes the cached entry after TTL expires

Negative Caching

- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around

DNS Resource Records

DNS: distributed db storing resource records (RR)

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

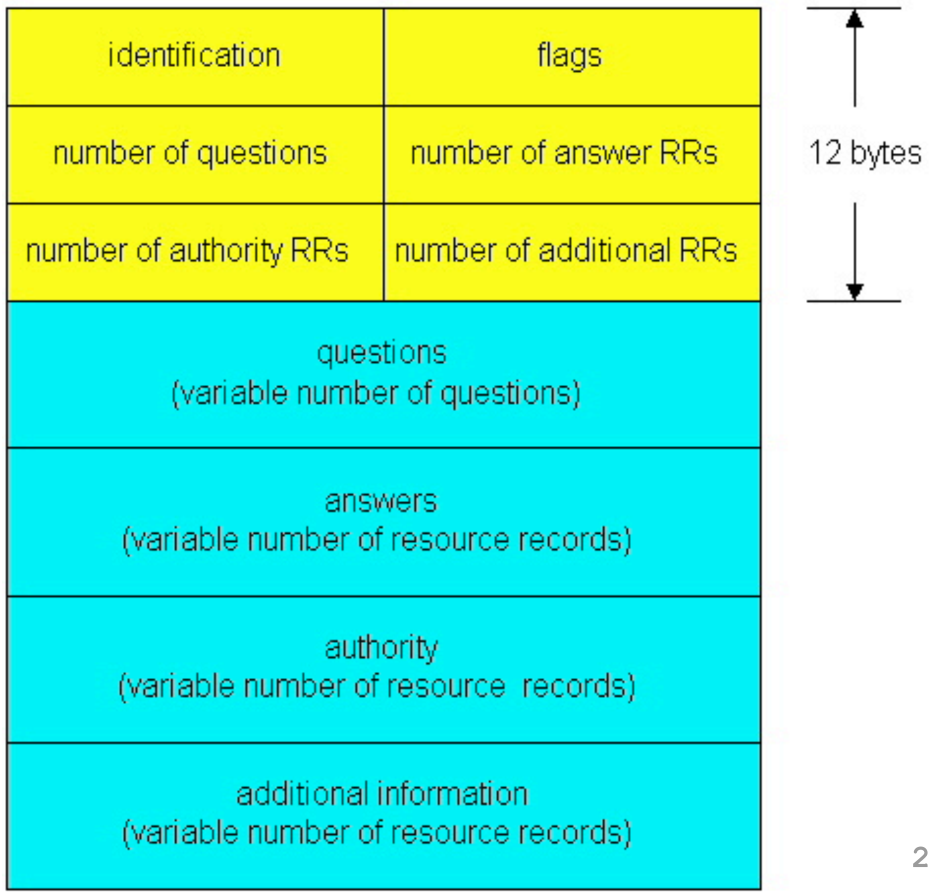
- Type=A
  - **`name`** is hostname
  - **`value`** is IP address
- Type=NS
  - **`name`** is domain (e.g. foo.com)
  - **`value`** is hostname of authoritative name server for this domain
- Type=CNAME
  - **`name`** is alias for some "canonical" (the real) name: `www.ibm.com` is really `srveast.backup2.ibm.com`
  - **`value`** is canonical name
- Type=MX
  - **`value`** is name of mailserver associated with **`name`**

DNS Protocol

DNS protocol : *query* and *reply* msg, both with same *msg format*

Message header

- Identification: 16 bit # for query, reply to query uses same #
- Flags:
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative

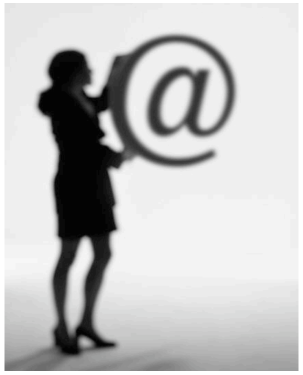


Reliability

- 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
- Same identifier for all queries
  - Don’t care which server responds

Inserting Resource Records into DNS

- Example: just created startup “FooBar”
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foobar.com
  - Type MX record for foobar.com
- Play with “dig” on UNIX



```
$ dig nytimes.com ANY
; QUESTION SECTION:
;nytimes.com.           IN   ANY

;; ANSWER SECTION:
nytimes.com.        267     IN   MX   100 NYTIMES.COM.S7A1.PSMTP.com.
nytimes.com.        267     IN   MX   200 NYTIMES.COM.S7A2.PSMTP.com.
nytimes.com.        267     IN   A    199.239.137.200
nytimes.com.        267     IN   A    199.239.136.200
nytimes.com.        267     IN   TXT "v=spf1 mx ptr ip4:199.239.138.0/24
   include:alerts.wallst.com include:authsmtp.com ~all"
nytimes.com.        267     IN   SOA      ns1t.nytimes.com.
   root.ns1t.nytimes.com. 2009070102 1800 3600 604800 3600
nytimes.com.        267     IN   NS   nydns2.about.com.
nytimes.com.        267     IN   NS   ns1t.nytimes.com.
nytimes.com.        267     IN   NS   nydns1.about.com.

;; AUTHORITY SECTION:
nytimes.com.        267     IN   NS   nydns1.about.com.
nytimes.com.        267     IN   NS   ns1t.nytimes.com.
nytimes.com.        267     IN   NS   nydns2.about.com.

;; ADDITIONAL SECTION:
nydns1.about.com.   86207   IN   A    207.241.145.24
nydns2.about.com.   86207   IN   A    207.241.145.25
```

```
$ dig nytimes.com +norec @a.root-servers.net

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53675
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14

;; QUESTION SECTION:
;nytimes.com.                   IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      K.GTLD-SERVERS.NET.
com.                    172800  IN      NS      E.GTLD-SERVERS.NET.
com.                    172800  IN      NS      D.GTLD-SERVERS.NET.
com.                    172800  IN      NS      I.GTLD-SERVERS.NET.
com.                    172800  IN      NS      C.GTLD-SERVERS.NET.

;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET.     172800  IN      A       192.5.6.30
A.GTLD-SERVERS.NET.     172800  IN      AAAA    2001:503:a83e::2:30
B.GTLD-SERVERS.NET.     172800  IN      A       192.33.14.30
B.GTLD-SERVERS.NET.     172800  IN      AAAA    2001:503:231d::2:30
C.GTLD-SERVERS.NET.     172800  IN      A       192.26.92.30
D.GTLD-SERVERS.NET.     172800  IN      A       192.31.80.30
E.GTLD-SERVERS.NET.     172800  IN      A       192.12.94.30

;; Query time: 76 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Feb 23 11:24:06 2009
;; MSG SIZE  rcvd: 501
```

```
$ dig nytimes.com +norec @k.gtld-servers.net

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38385
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;nytimes.com.                   IN      A

;; AUTHORITY SECTION:
nytimes.com.            172800  IN      NS      ns1t.nytimes.com.
nytimes.com.            172800  IN      NS      nydns1.about.com.
nytimes.com.            172800  IN      NS      nydns2.about.com.

;; ADDITIONAL SECTION:
ns1t.nytimes.com.       172800  IN      A       199.239.137.15
nydns1.about.com.       172800  IN      A       207.241.145.24
nydns2.about.com.       172800  IN      A       207.241.145.25

;; Query time: 103 msec
;; SERVER: 192.52.178.30#53(192.52.178.30)
;; WHEN: Mon Feb 23 11:24:59 2009
;; MSG SIZE  rcvd: 144
```

```
$ dig nytimes.com ANY +norec @ns1t.nytimes.com

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39107
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;nytimes.com.                   IN      ANY

;; ANSWER SECTION:
nytimes.com.      300   IN      SOA     ns1t.nytimes.com.
         root.ns1t.nytimes.com. 2009070102 1800 3600 604800 3600
nytimes.com.      300   IN      MX      200 NYTIMES.COM.S7A2.PSMTP.com.
nytimes.com.      300   IN      MX      100 NYTIMES.COM.S7A1.PSMTP.com.
nytimes.com.      300   IN      NS      ns1t.nytimes.com.
nytimes.com.      300   IN      NS      nydns1.about.com.
nytimes.com.      300   IN      NS      nydns2.about.com.
nytimes.com.      300   IN      A       199.239.137.245
nytimes.com.      300   IN      A       199.239.136.200
nytimes.com.      300   IN      A       199.239.136.245
nytimes.com.      300   IN      TXT     "v=spf1 mx ptr ip4:199.239.138.0/24
        include:alerts.wallst.com include:authsmtp.com ~all"

;; ADDITIONAL SECTION:
ns1t.nytimes.com.       300     IN      A       199.239.137.15

;; Query time: 10 msec
;; SERVER: 199.239.137.15#53(199.239.137.15)
;; WHEN: Mon Feb 23 11:25:20 2009
;; MSG SIZE  rcvd: 454
```

DNS security

- DNS cache poisoning
  - Ask for www.evil.com
  - Additional section for (www.cnn.com, 1.2.3.4, A)
  - Thanks! I won't bother check what I asked for

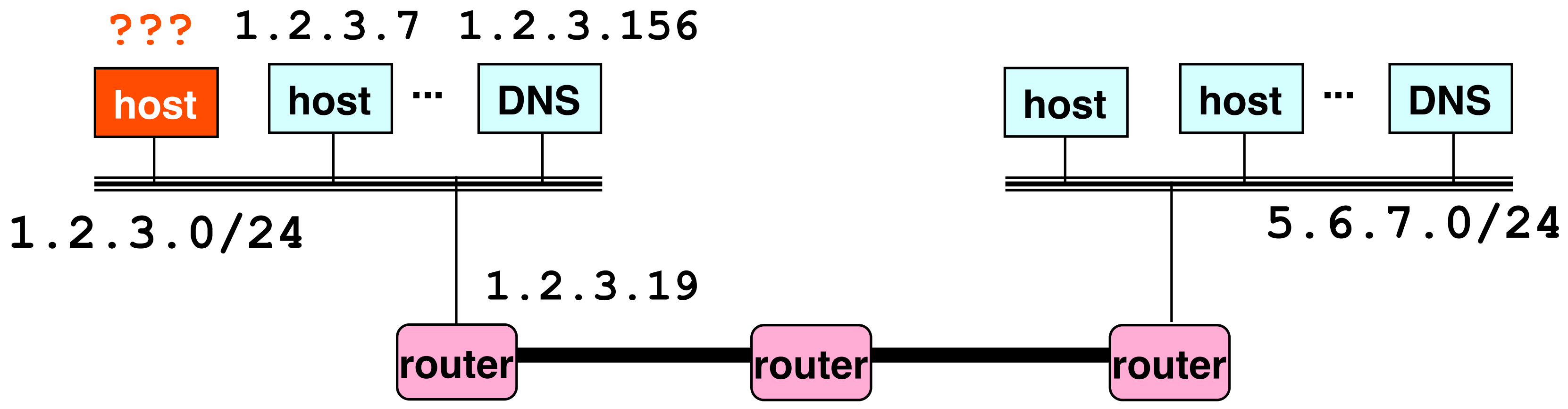
- DNS hijacking
  - Let's remember the domain. And the UDP ID.
  - 16 bits: 65K possible IDs
    - What rate to enumerate all in 1 sec? 64B/packet
    - 64*65536*8 / 1024 / 1024 = 32 Mbps
  - Prevention: Also randomize the DNS source port
    - E.g., Windows DNS alloc's 2500 DNS ports: ~164M possible IDs
    - Would require 80 Gbps
    - Kaminsky attack: this source port…wasn't random after all

Boot-Strapping an End Host

DHCP and ARP

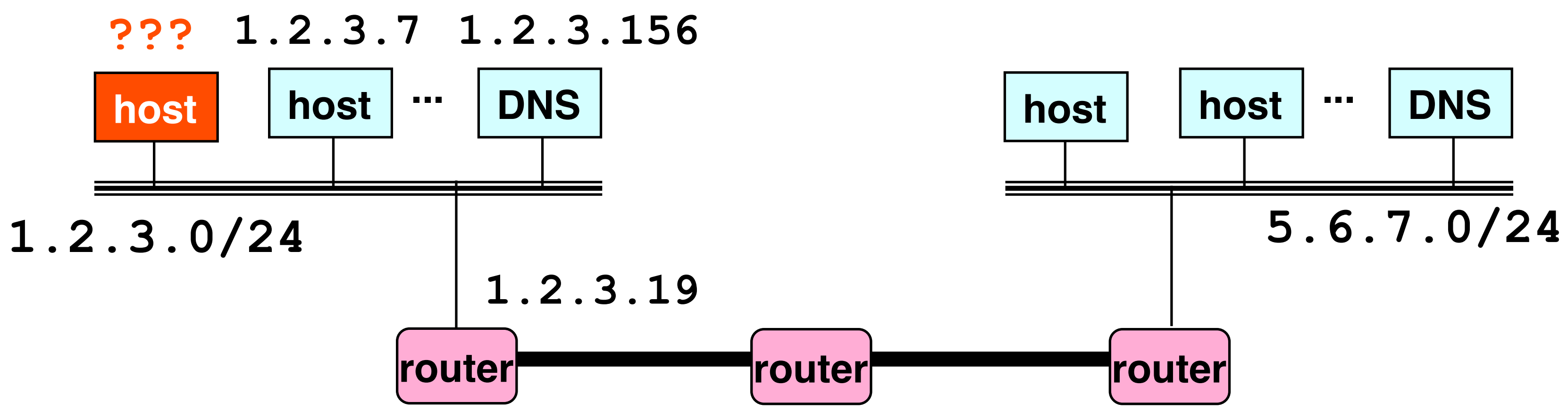
How To Bootstrap an End Host?

- What local Domain Name System server to use?
- What IP address the host should use?
- How to send packets to remote destinations?
- How to ensure incoming packets arrive?



Avoiding Manual Configuration

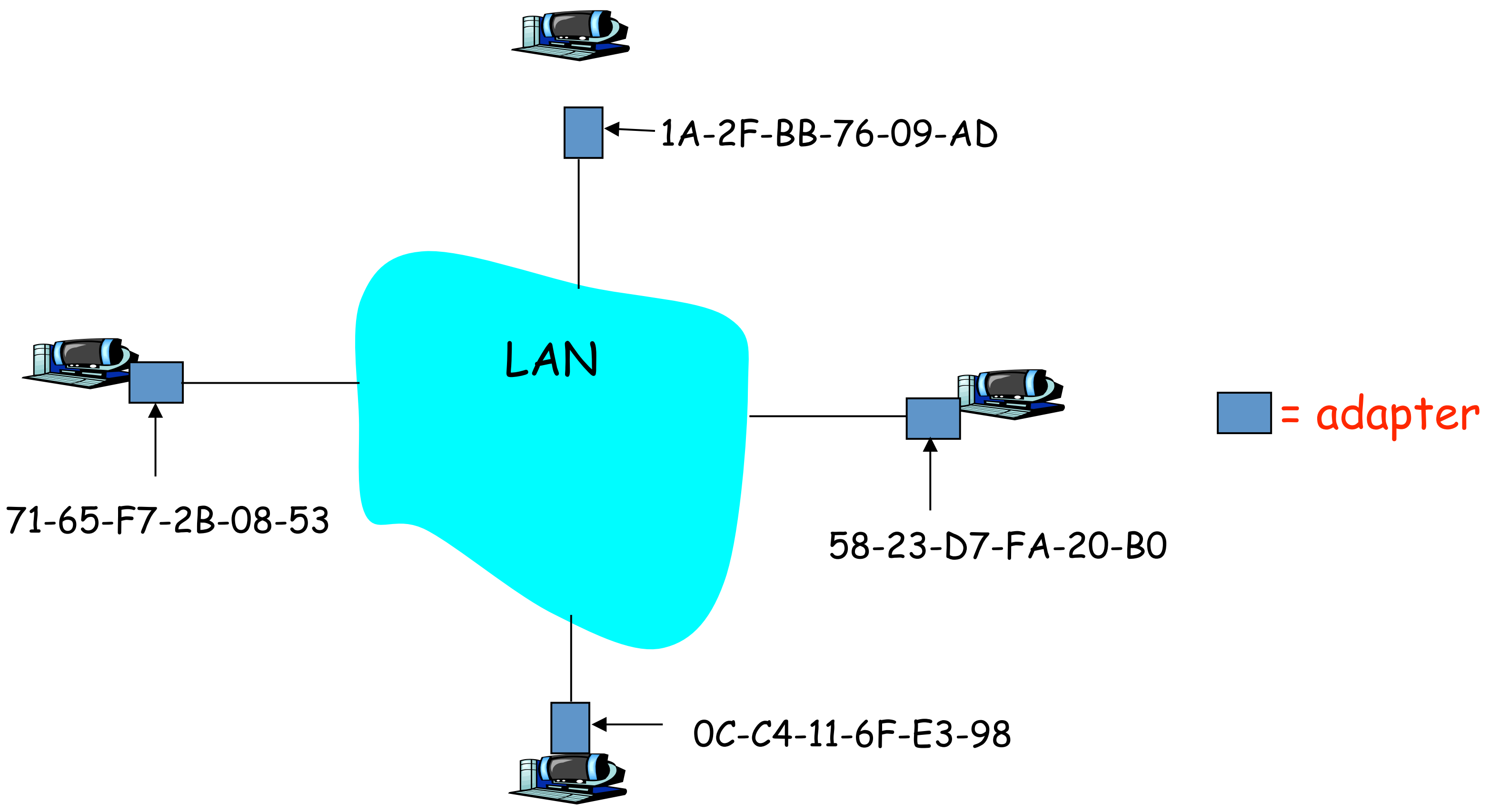
- Dynamic Host Configuration Protocol (DHCP)
  – End host learns how to send packets
  – Learn IP address, DNS servers, and gateway
- Address Resolution Protocol (ARP)
  – Others learn how to send packets to the end host
  – Learn mapping between IP address & interface address



Key Ideas in Both Protocols

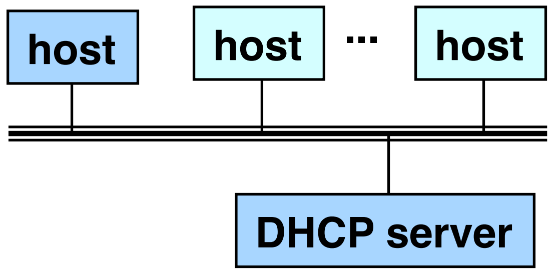
- **Broadcasting:** when in doubt, shout!
  - Broadcast query to all hosts in the local-area-network
  - … when you don't know how to identify the right one
- **Caching:** remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your own address & other host's addresses
- **Soft state:** … but eventually forget the past
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information
  - Key for robustness in the face of unpredictable change

Media Access Control (MAC) Addresses



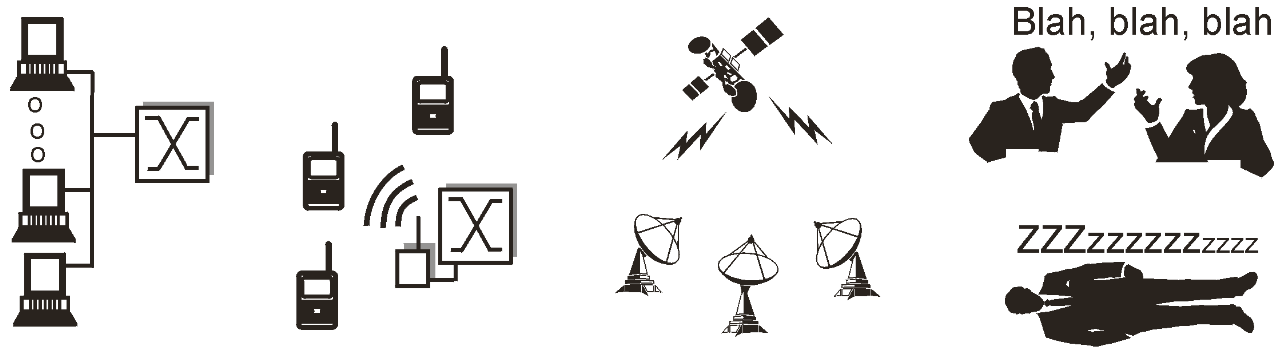
Bootstrapping Problem

- 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 addr to use
- Solution: shout to discover a server who can help
  - Broadcast a DHCP server-discovery message
  - Server sends a DHCP “offer” offering an address



Broadcasting

- Broadcasting: sending to everyone
  - Special destination address: FF-FF-FF-FF-FF-FF
  - All adapters on the LAN receive the packet
- Delivering a broadcast packet
  - Easy on a "shared media"
  - Like shouting in a room – everyone can hear you

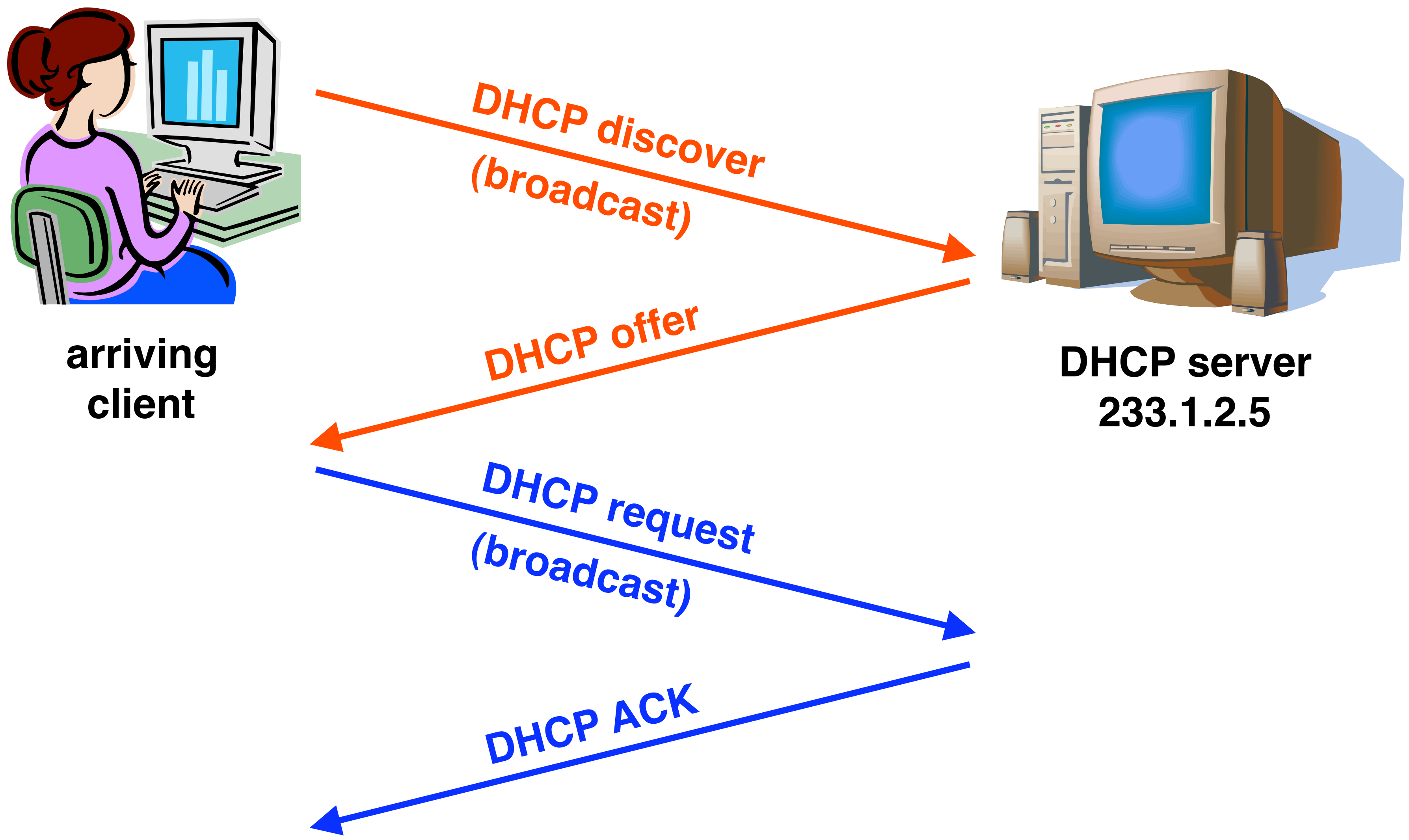


shared wire (e.g. Ethernet) | shared wireless (e.g. Wavelan) | satellite | cocktail party

Response from the DHCP Server

- DHCP “offer message” from the server
  - Configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (the time the information remains valid)
- Multiple servers may respond
  - Multiple servers on the same broadcast media
  - Each may respond with an offer
  - The client can decide which offer to accept
- Accepting one of the offers
  - Client sends a DHCP request echoing the parameters
  - The DHCP server responds with an ACK to confirm
  - … and the other servers see they were not chosen

Dynamic Host Configuration Protocol



Deciding What IP Address to Offer

- Server as centralized configuration database
  - All parameters are statically configured in the server
  - E.g., a dedicated IP address for each MAC address
  - Avoids complexity of configuring hosts directly
  - … while still having a permanent IP address per host
- Or, dynamic assignment of IP addresses
  - Server maintains a pool of available addresses
  - … and assigns them to hosts on demand
  - Leads to less configuration complexity
  - … and more efficient use of the pool of addresses
  - Though, it is harder to track the same host over time

Soft State: Refresh or Forget

- Why is a lease time necessary?
  - Client can release the IP address (DHCP RELEASE)
    - E.g., "ipconfig /release" at the DOS prompt
    - E.g., clean shutdown of the computer
  - But, the host might not release the address
    - E.g., the host crashes (blue screen of death!)
    - E.g., buggy client software
  - And you don't want the address to be allocated forever

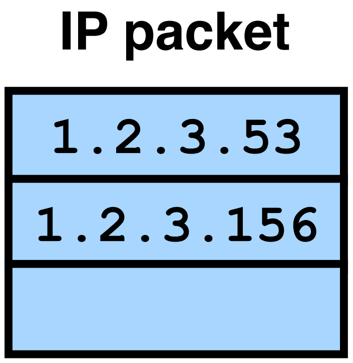
- Performance trade-offs
  - Short lease time: returns inactive addresses quickly
  - Long lease time: avoids overhead of frequent renewals

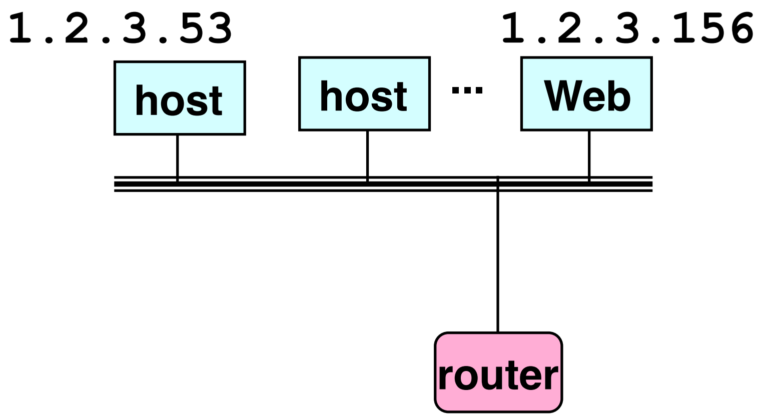
So, Now the Host Knows Things

- IP address
- Mask
- Gateway router
- DNS server
- …

- And can send packets to other IP addresses
  - But, how to learn MAC address of the destination?

Sending Packets Over a Link





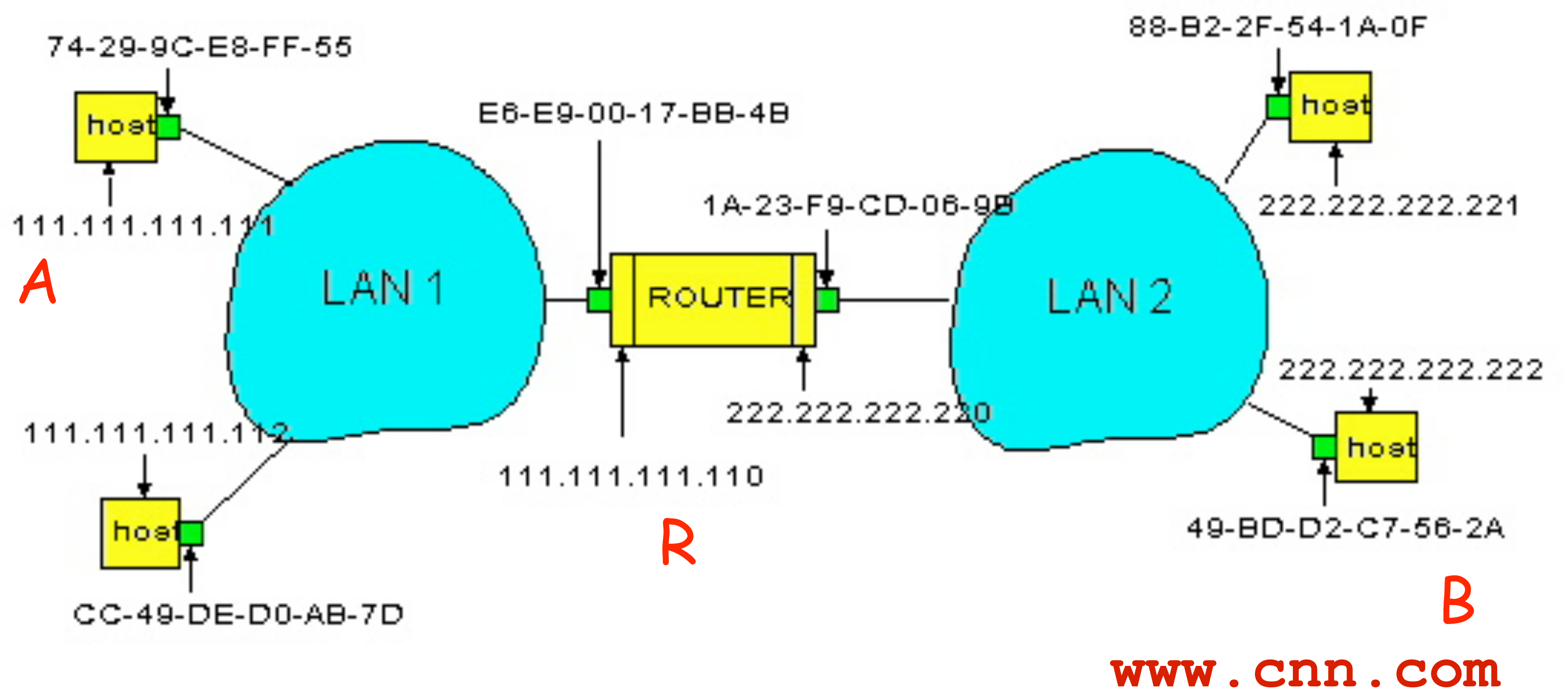
- Adapters only understand MAC addresses
  - Translate the destination IP address to MAC address
  - Encapsulate the IP packet inside a link-level frame

Address Resolution Protocol Table

- Every node maintains an ARP table
  - (IP address, MAC address) pair
- Consult the table when sending a packet
  - Map destination IP address to destination MAC address
  - Encapsulate and transmit the data packet
- But, what if the IP address is not in the table?
  - Sender broadcasts: “Who has IP address 1.2.3.156?”
  - Receiver responds: “MAC address 58-23-D7-FA-20-B0”
  - Sender caches the result in its ARP table
- No need for network administrator to get involved

Example: A Sending a Packet to B

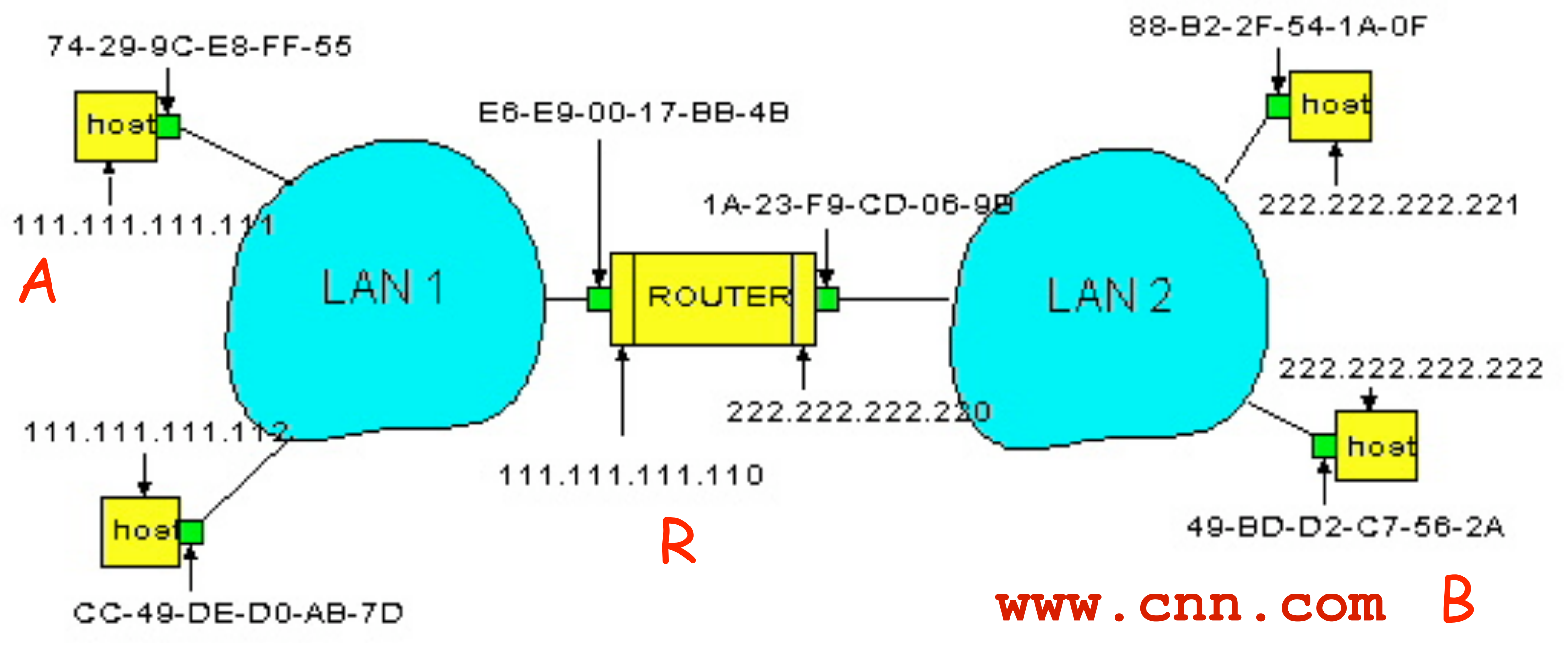
How does host A send an IP packet to B (www.cnn.com)?



A sends packet to R, and R sends packet to B

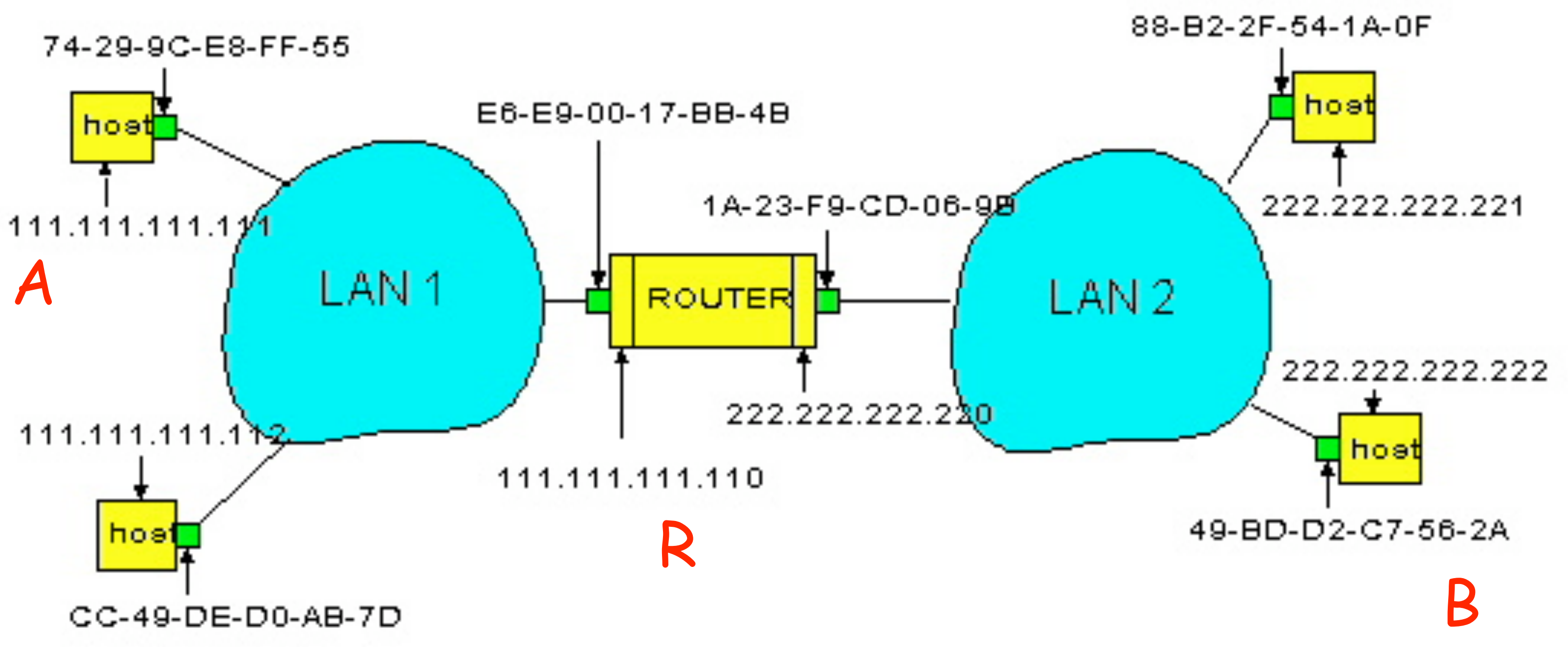
Basic Steps

- Host A must learn the IP address of B via DNS
- Host A uses gateway R to reach external hosts
- Host A sends the frame to R's MAC address
- Router R forwards IP packet to outgoing interface
- Router R learns B's MAC address and forwards frame



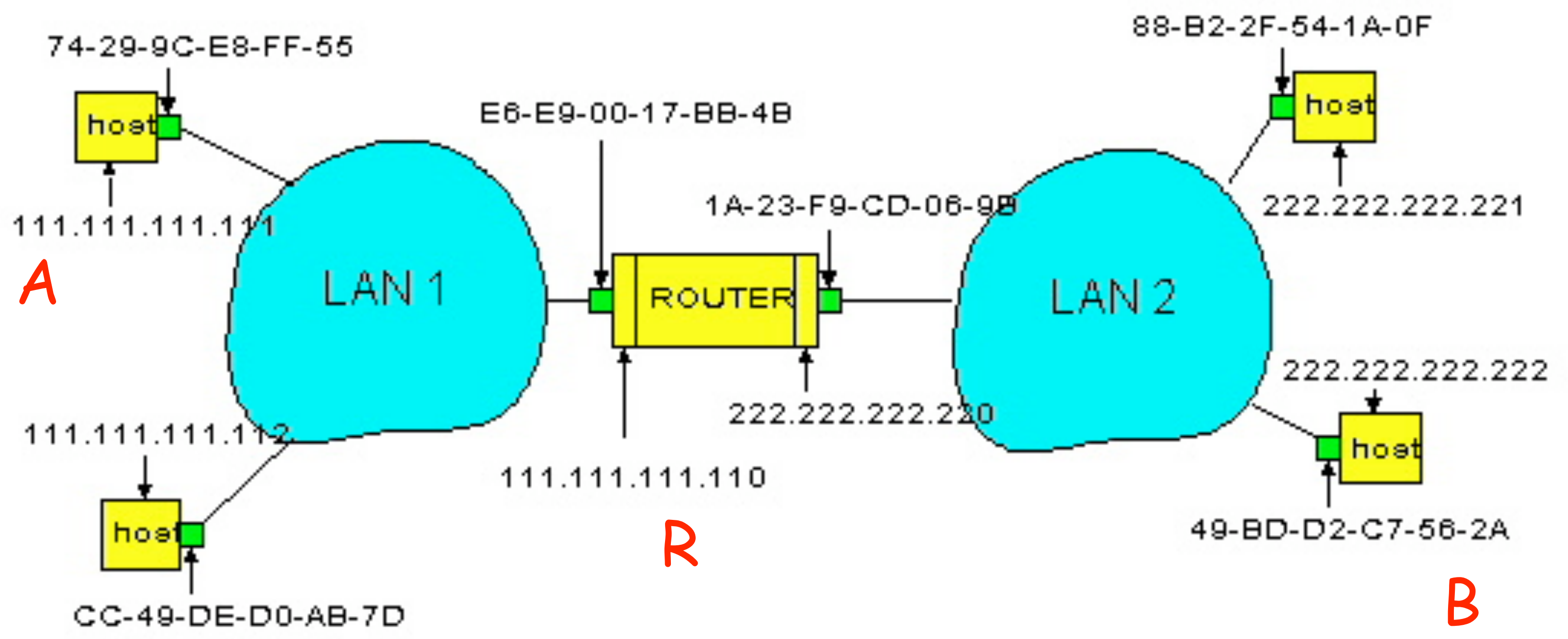
Host A Learns the IP Address of B

- Host A does a DNS query to learn B's address
  - Suppose gethostbyname() returns 222.222.222.222
- Host A constructs an IP packet to send to B
  - Source 111.111.111.111, dest 222.222.222.222



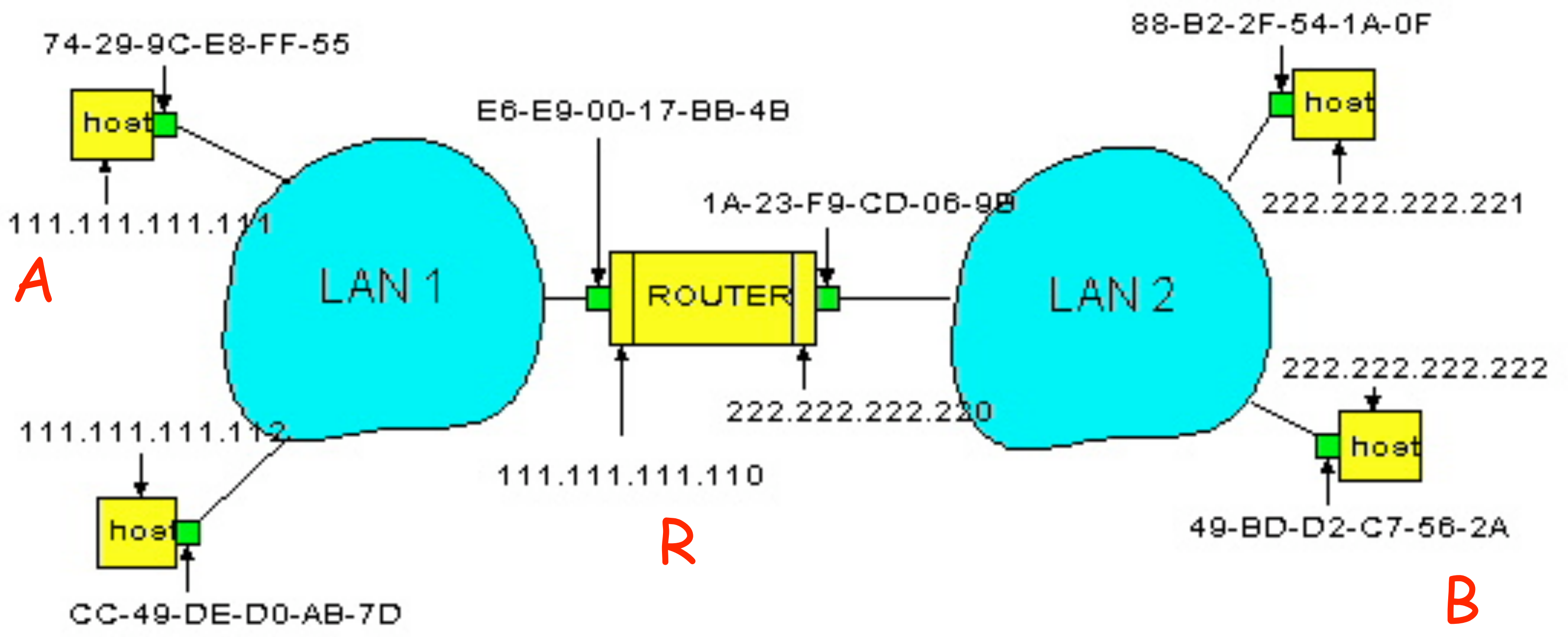
Host A Learns the IP Address of B

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From A: 74-29-9C-E8-FF-55
  - To gateway: ????



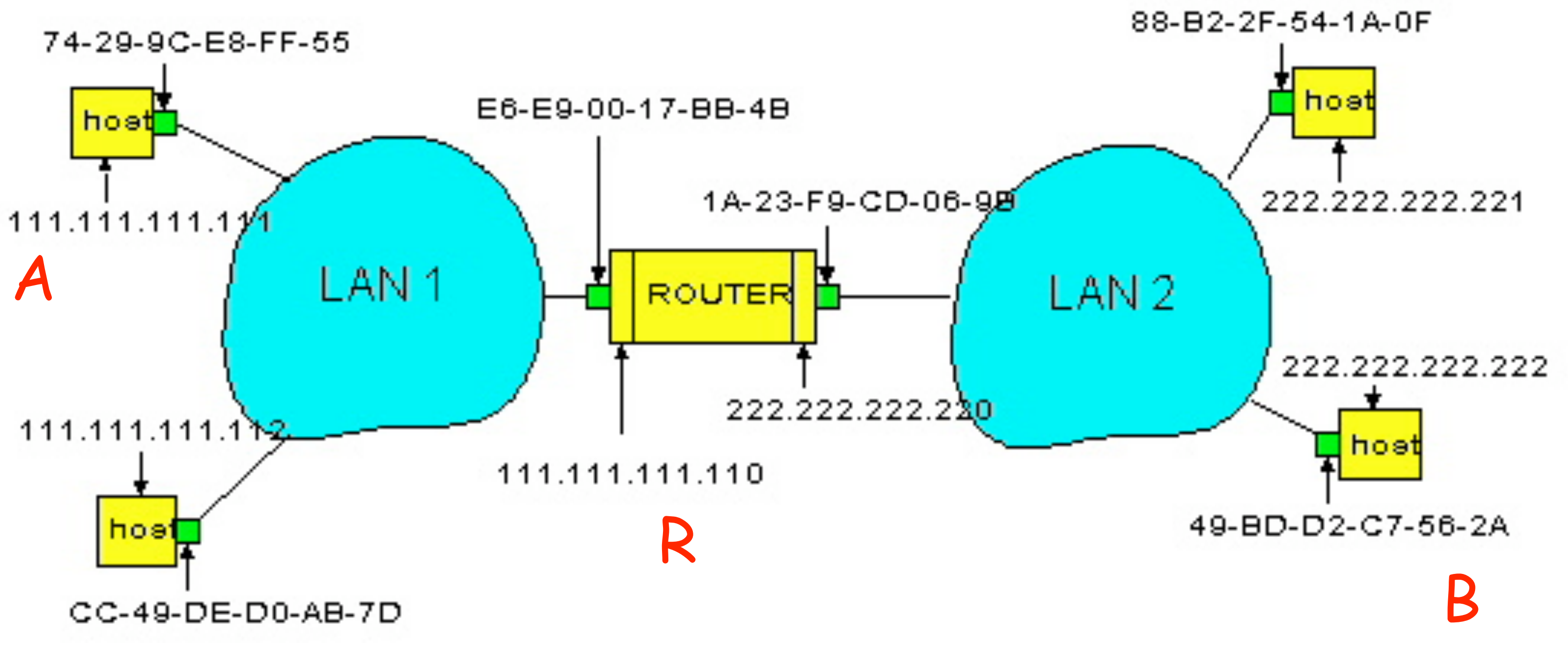
Host A Decides to Send Through R

- Host A has a gateway router R
  - Used to reach dests outside of 111.111.111.0/24
  - Address 111.111.111.110 for R learned via DHCP
- But, what is the MAC address of the gateway?



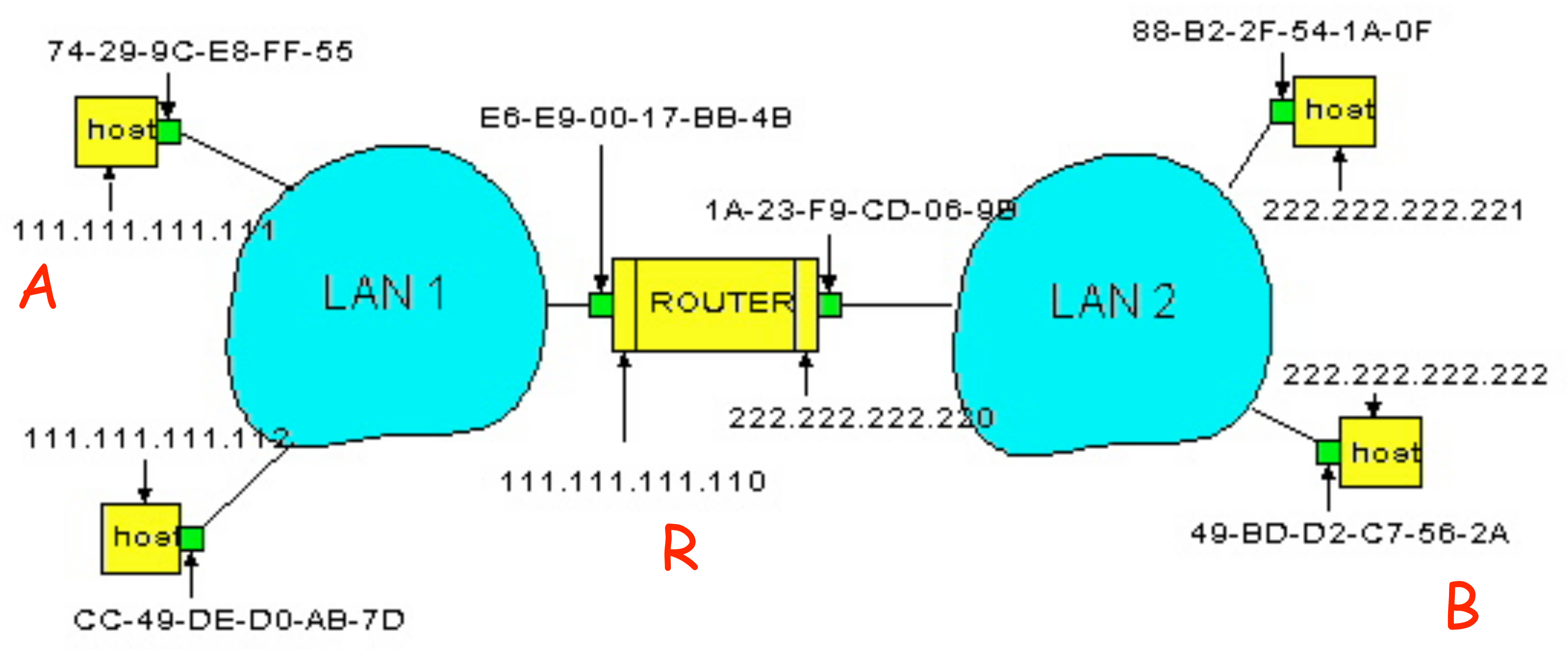
Host A Sends Packet Through R

- Host A learns the MAC address of R's interface
  - ARP request: broadcast request for 111.111.111.110
  - ARP response: R responds with E6-E9-00-17-BB-4B
- Host A encapsulates the packet and sends to R



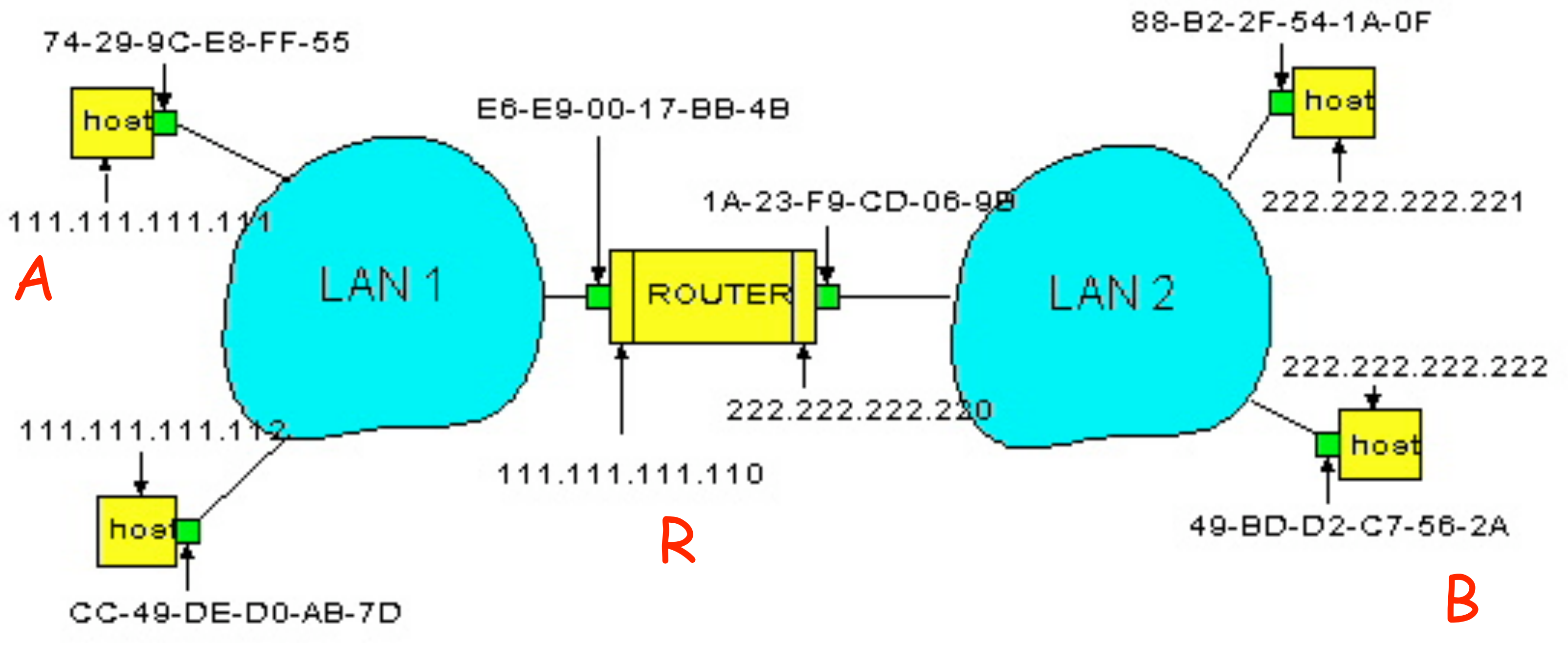
Host A Sends Packet Through R

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From A: 74-29-9C-E8-FF-55
  - To R: E6-E9-00-17-BB-4B



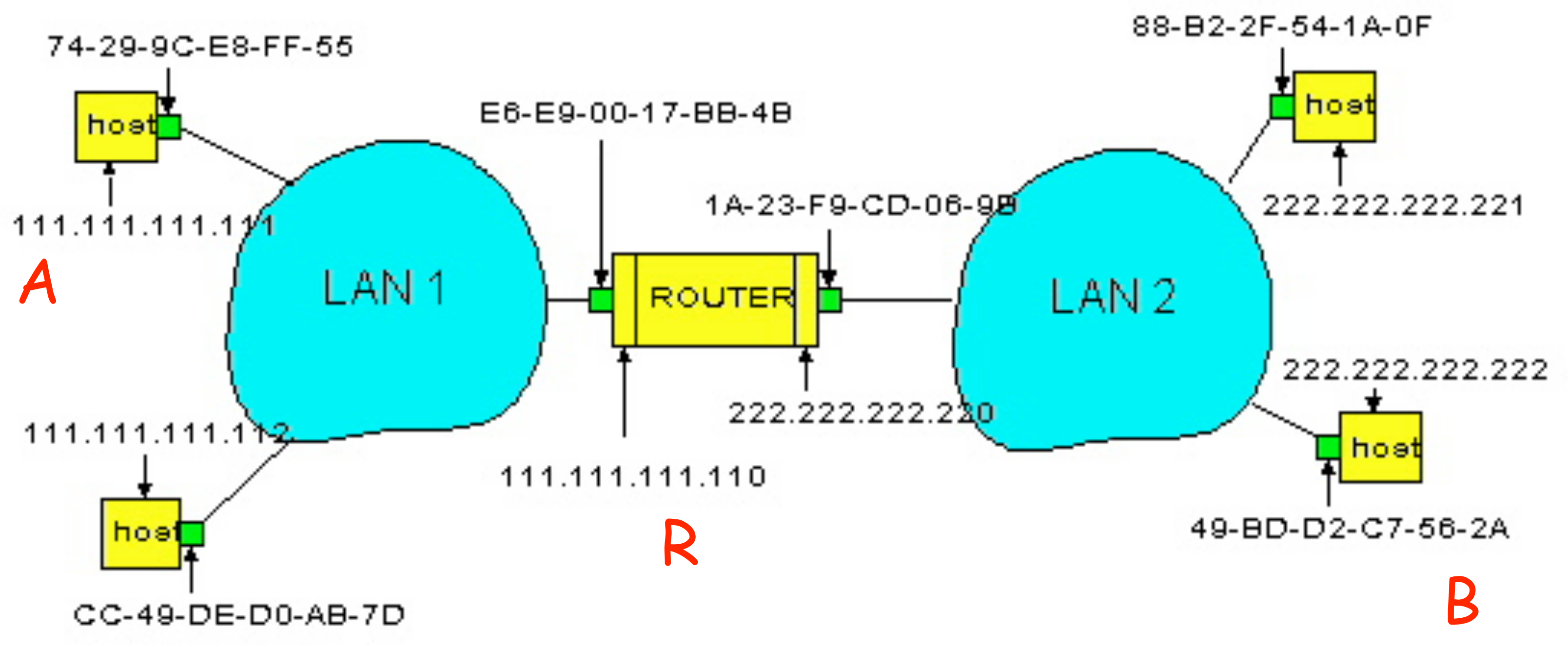
R Decides how to Forward Packet

- Router R's adapter receives the packet
  - R extracts the IP packet from the Ethernet frame
  - R sees the IP packet is destined to 222.222.222.222
- Router R consults its forwarding table
  - Packet matches 222.222.222.0/24 via other adapter



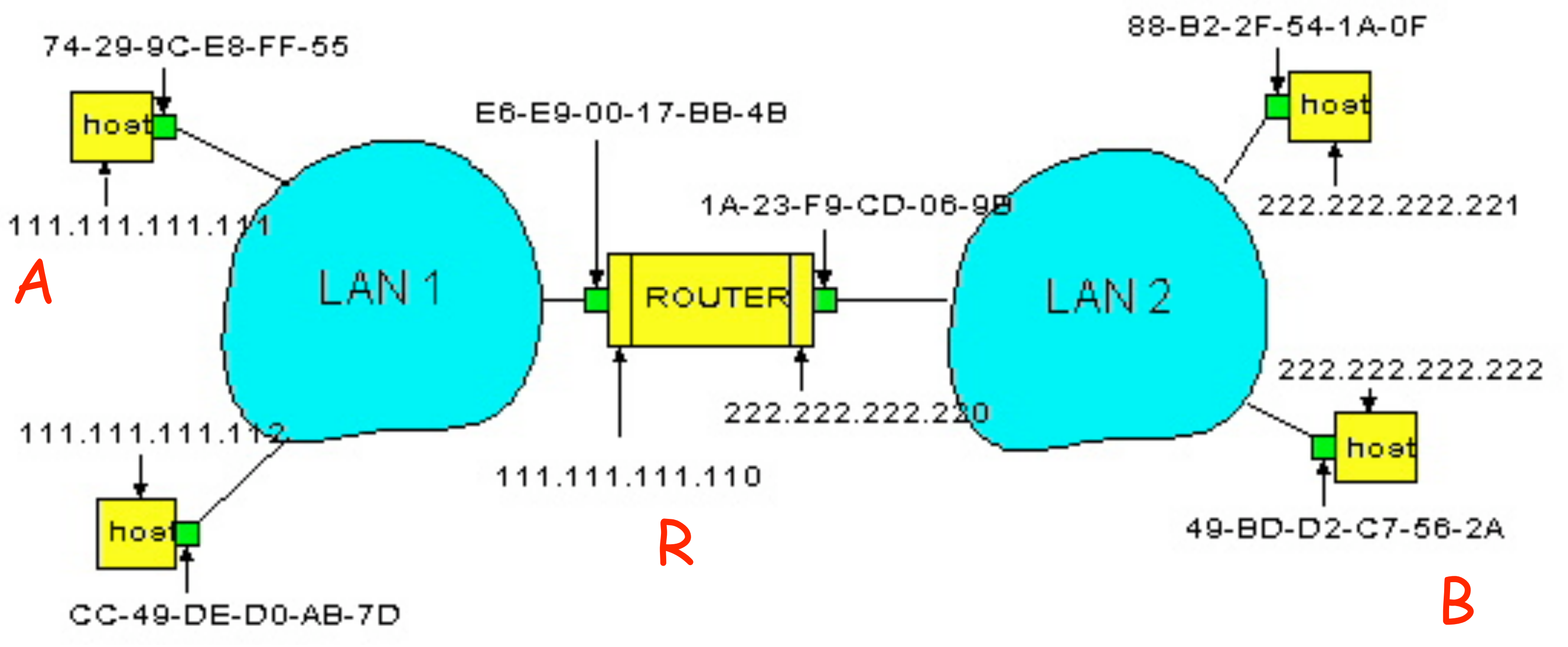
Router R Wants to Forward Packet

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From R: 1A-23-F9-CD-06-9B
  - To B: ???



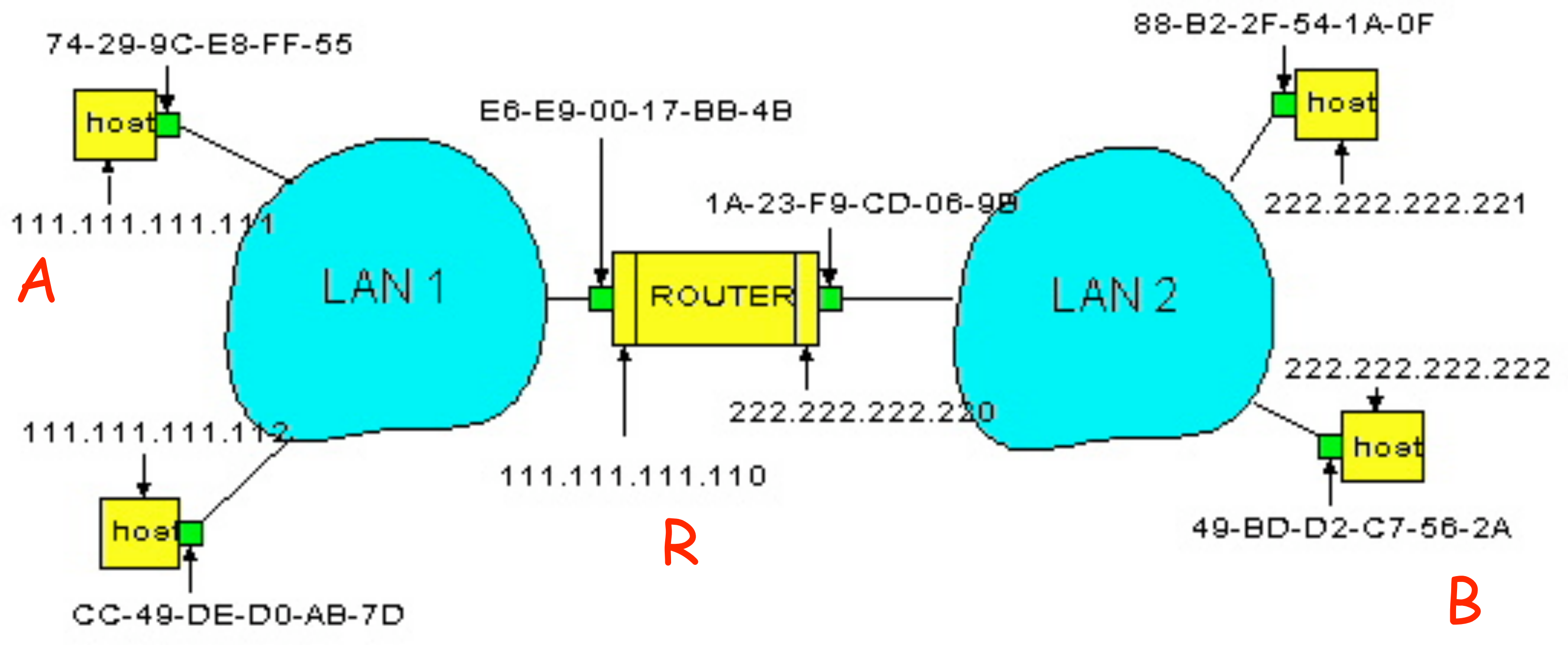
R Sends Packet to B

- Router R's learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-56-2A
- Router R encapsulates the packet and sends to B



Router R Wants to Forward Packet

- IP header
  - From A: 111.111.111.111
  - To B: 222.222.222.222
- Ethernet frame
  - From R: 1A-23-F9-CD-06-9B
  - To B: 49-BD-D2-C7-56-2A



Conclusion

- Domain Name System
  - Distributed, hierarchical database
  - Distributed collection of servers
  - Caching to improve performance
- Bootstrapping an end host
  - Dynamic Host Configuration Protocol (DHCP)
  - Address Resolution Protocol (ARP)
- Next class: middleboxes
  - Reading: Section 8.4 (for Wednesday) and Ch. 2
  - Network Address Translator (NAT)
  - Firewalls