CS 425 / ECE 428 Distributed Systems Fall 2014

Indranil Gupta (Indy)

Lecture 14: Networking and Routing

Our Distributed System Definition

Focus of this lecture

A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and communicating through an unreliable communication medium.

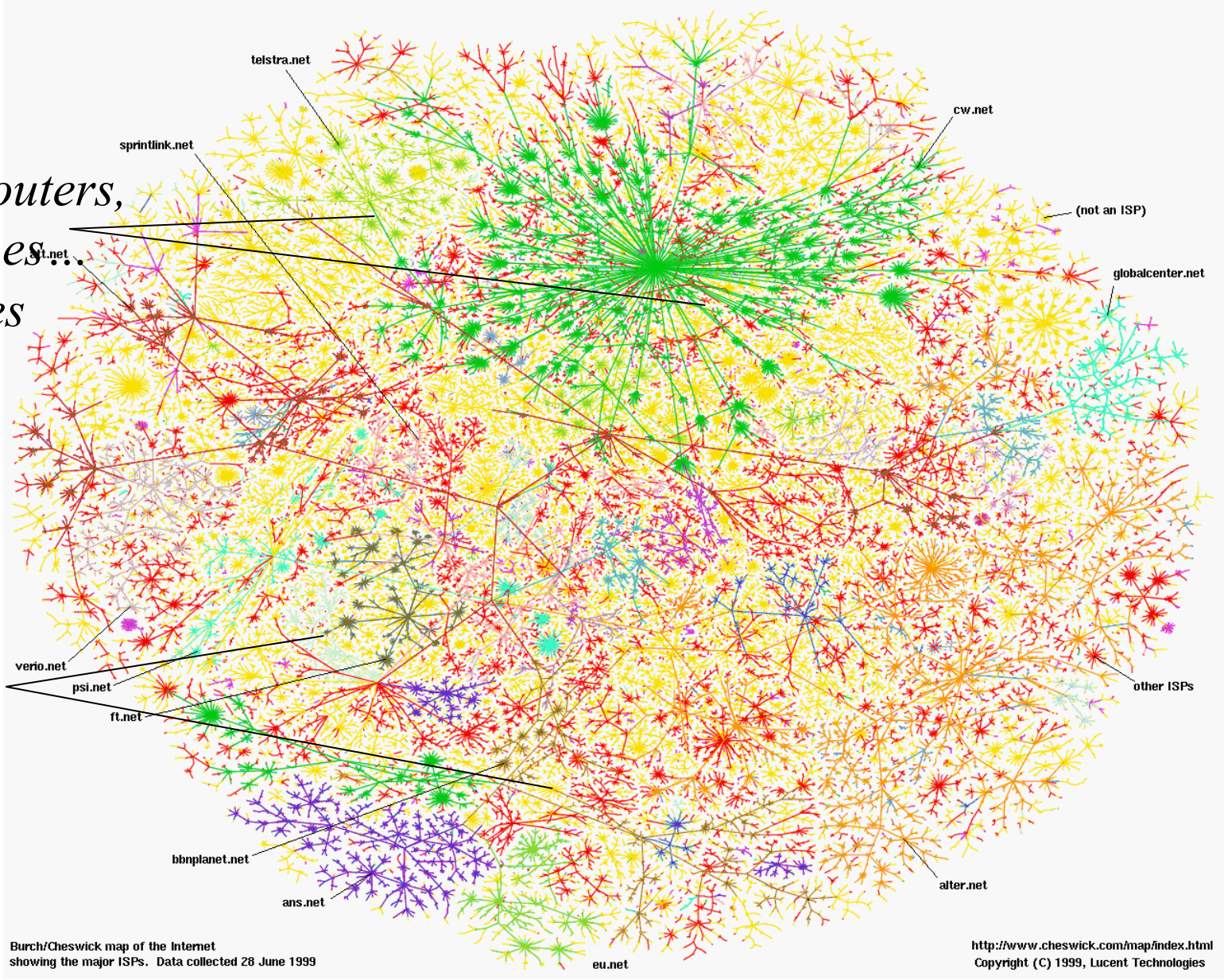
- **Our interest in distributed systems involves**
  - **design and implementation, maintenance, study, algorithmics**
- **Entity=a process on a device (PC, PDA)**
- **Communication Medium=Wired or wireless network**

So far…

- Abstract distributed system – collection of processes over a communication medium
- Protocols/algorithms for synchronization, snapshots, multicast, election, mutual exclusion, failure detectors, p2p, Hadoop
- Intended to work in *any* distributed group of processes
  1. E.g., Group of processes on computer hosts
  2. E.g., Group of processes on mobile devices
- For most of this course, we'll focus on (1): computer hosts over the Internet

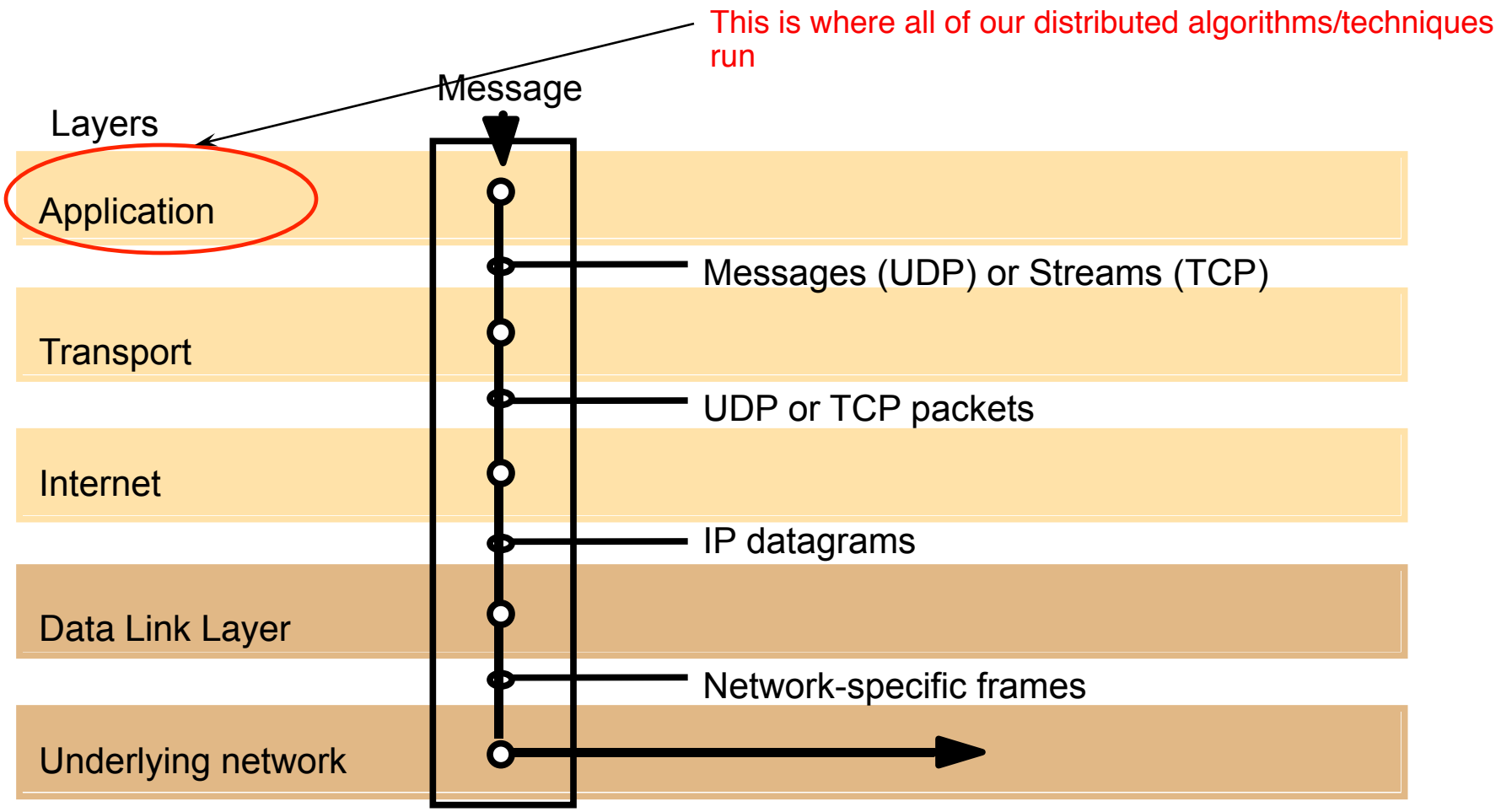
PCs,routers, switches… =nodes

links= *edges*



The Internet (Internet Mapping Project, color coded by ISPs)

Internet 5-Layer Networking Stack



Routing Algorithms

- Nodes connected in some topology. Routing algorithm runs at network layer in each node.
- Goal of routing algorithm:
  - given the destination IP address in packet, determine the "next hop"
  - thus determine the route for each packet as it travels through the net
  - dynamically update routing information to reflect failures, changes (e.g., router joins and leaves) and congestion (overloaded router)
- Two approaches:
  - Distance-vector (e.g., RIP)
    - Every node knows, for each possible destination LAN, the next-hop
  - Link-state (e.g., OSPF)
    - Every node knows status of every "link" in the network

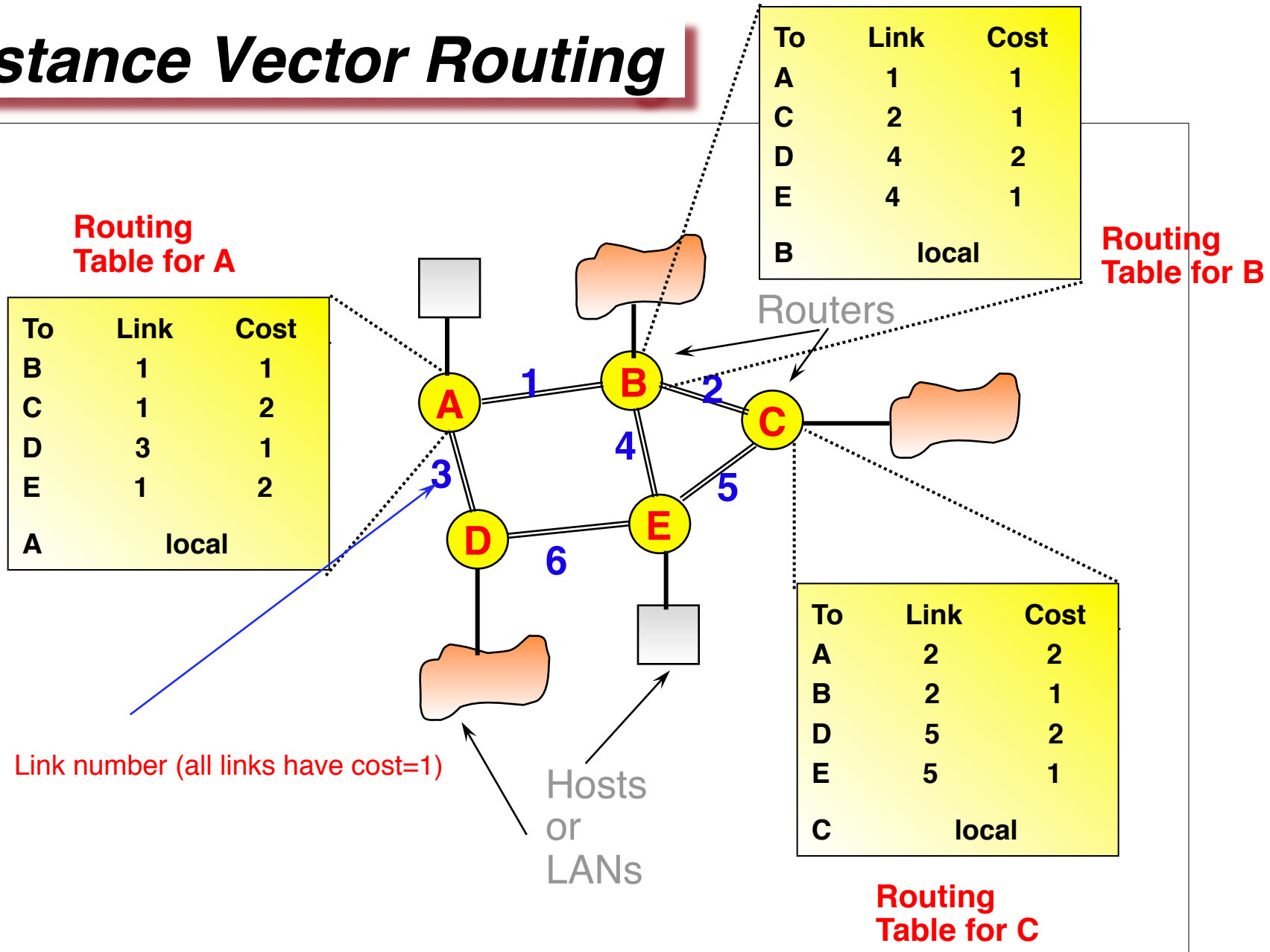
  In both, information maintained as a table

  Tables updated either
  - Proactively – periodically, or
  - Reactively – when a neighbor/some link status changes

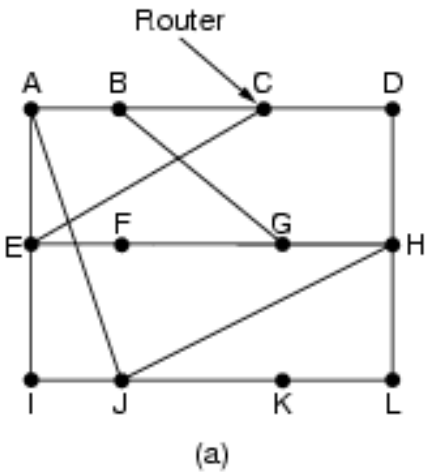
Distance Vector Routing

- **Also termed as distributed Bellman-Ford algorithm or Ford-Fulkerson algorithm, included in RIP (Routing Information Protocol), AppleTalk, and Cisco routers.**
  - **Each node/router maintains a table indexed by each destination node. Entry gives (best known distance to destination, best next-hop for destination)**
  - **Once every T seconds, each router sends to each neighbor its own entire table. Neighbor uses this to update its own table. (proactive)**

Distance Vector Routing



Distance Vector Routing: Node J



(a)

Costs only (next hop omitted)

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	JA delay is	JI delay is	JH delay is	JK delay is	New routing table for J
	8	10	12	6	

+, then min across neighbors

Vectors received from J's four neighbors

(b)

Pseudo-Code for RIP

Send: Each *t* seconds or when *Tl* changes, send *Tl* on each non-faulty outgoing link.
Receive: Whenever a routing table *Tr* is received on link *n*:

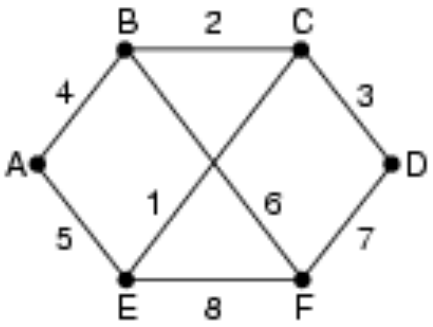
```
for all rows Rr in Tr {
    if (Rr.link not equal n) {
        Rr.cost = Rr.cost + 1;
        Rr.link = n;
        if (Rr.destination is not in Tl) add Rr to Tl;
         // add new destination to Tl
        else for all rows Rl in Tl {
            if (Rr.destination = Rl.destination and
                    (Rr.cost < Rl.cost or Rl.link = n)) Rl = Rr;
            // Rr.cost < Rl.cost : remote node has better route
            // Rl.link = n : remote node is more authoritative
        }
    }
}
```

Link State Routing

- **Each router must**
  1. **Discover its neighbors and learn their network addresses**
     - **When a router is booted up, it learns who its neighbors are by sending a special Hello packet on each point-to-point link.**
     - **The router on the other end sends back a reply.**
  2. **Measure the delay or cost to each of its neighbors**
     - **A router sends a special Echo packet over the link that the other end sends back immediately. By measuring the round-trip time, the sending router gets an RTT estimate.**
  3. **Construct a packet telling all it has just learned.**
     - **Broadcast this packet**

Link State Routing

- A router broadcasts a link-state-advertisement (LSA) packet after booting, as well as periodically (or upon topology change). Packet forwarded only once, TTL-restricted
- Initial TTL is very high, since need it to get to every router



(a)

Link State Packets

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

(b)

Link State Routing

4. **Broadcast the LSA packet to all other routers.**
   - Each packet contains a sequence number that is incremented for each new LSA packet sent by that source router.
   - Each router keeps track of all the (source router, sequence) pairs it sees. When a new LSA packet comes in, it is checked against these pairs. If the received packet is new, it is forwarded on all the links except the one it arrived on. Old LSA packets are dropped.
   - The age of each packet, being stored after reception, is decremented once per time unit. When the age hits zero, the information is discarded. Initial age = high. Such state is often called soft state.
5. **For routing a packet, since the source knows the entire network graph, it simply computes the shortest path (actual sequence of nodes) locally using the Dijkstra's algorithm. It can include the path in the packet, and intermediate nodes simply follow this route to decide their next hop for the packet.**

Transport Layer =Transmission Control Protocol

- **Function 0 – provide an application with a connection-oriented view of the network (IP is connectionless)**
- **Function 1 (Message decomposition and reassembly): Breaks messages into packets at the transmitting end and reassembles packets into messages at the receiving end.**
  - **E.g., using identification and fragment offset fields in IPv4 header**
- **Function 2 (Multiplexing and demultiplexing): Multiplexes several lower-rate sessions, all from the same source and all going to the same destination, into one session at the network layer.**
- **Function 3 (Reliable communication): Provides reliability to the application by acks + retransmissions in an end to end manner**
- **Function 4 (End-to-end congestion/flow control): Reduces rate at which data is sent when congestion is detected in the network. (TCP-friendliness)**
- **Function 5 (Ordering): Of packets within a stream -- FIFO**
- ***All these functionalities are a part of TCP.***

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., sal.cs.uiuc.edu – human-readable format

Q: given a resource name, how does a client find out the IP address of the service/server?

Domain Name System:

- *distributed database* implemented in a hierarchy of many *name servers*
- *application-layer protocol that is responsible for resolving* names (address/name translation)

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., sal.cs.uiuc.edu – human-readable format

Name: human-readable string

Address: machine-readable string

Name Resolution: Name → Address

E.g.,: given a resource name, how does a client find out the IP address of the service/ server

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., sal.cs.uiuc.edu – human-readable format

Q: given a resource name, how does a client find out the IP address of the service/server?

Domain Name System:

- *distributed database* implemented in a hierarchy of many *name servers*
- *application-layer protocol that is responsible for resolving* names (address/name translation)

DNS Name Servers

Why not have a central DNS server?

- single point of failure
- traffic volume
- may be far
- maintenance difficult

Doesn't *scale!*

(WWW contains several billion pages today)

Alternative

- no single server has all name-to-IP address mappings
- Hierarchy of name servers

authoritative name servers:

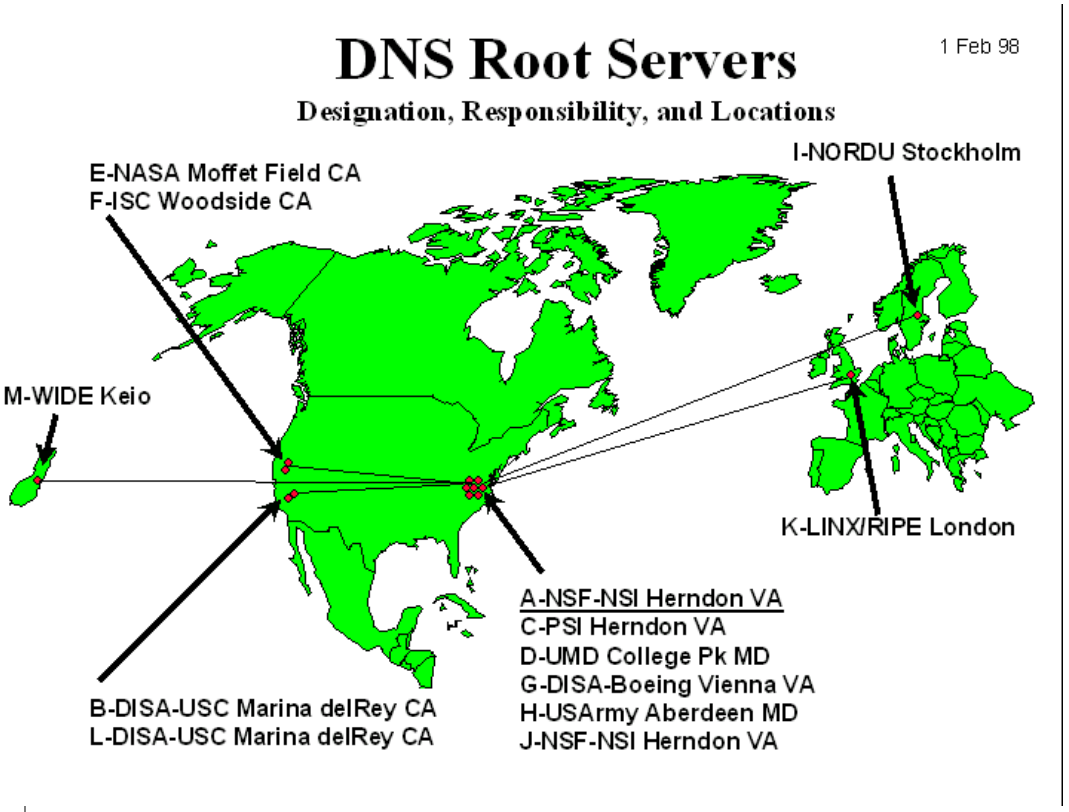
- for a resource, definitively stores the mapped IP address for that resource

local name servers:

- each institution/company/ISP owns a *local (default) name server*
- Receives DNS queries from host
- Caches recently seen name→address mappings, and can answer queries quickly

DNS: Root Name Servers

- contacted by local name server that cannot resolve query
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- ~ 13 root name servers worldwide (as of '12)

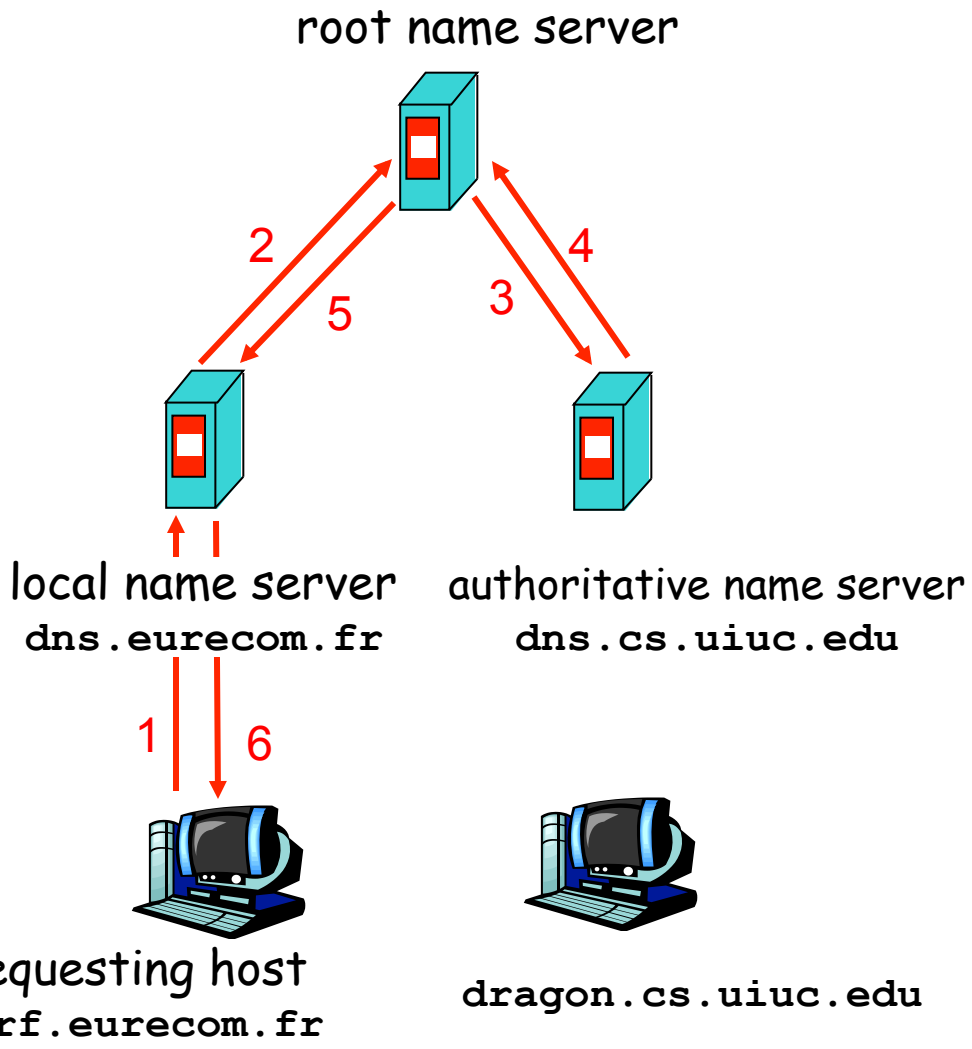


Simple DNS Example

host `surf.eurecom.fr` **wants IP address of** `dragon.cs.uiuc.edu`

1. **Contacts its local DNS server,** `dns.eurecom.fr`
2. `dns.eurecom.fr` **contacts root name server, if necessary**
3. **root name server contacts authoritative name server,** `dns.cs.uiuc.edu`, **if necessary**

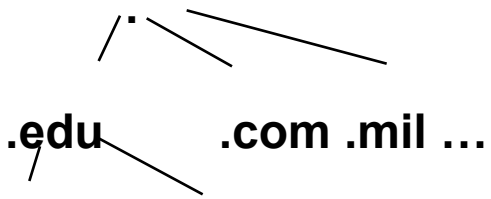
Answer returned by first server that is caching the mapping tuple

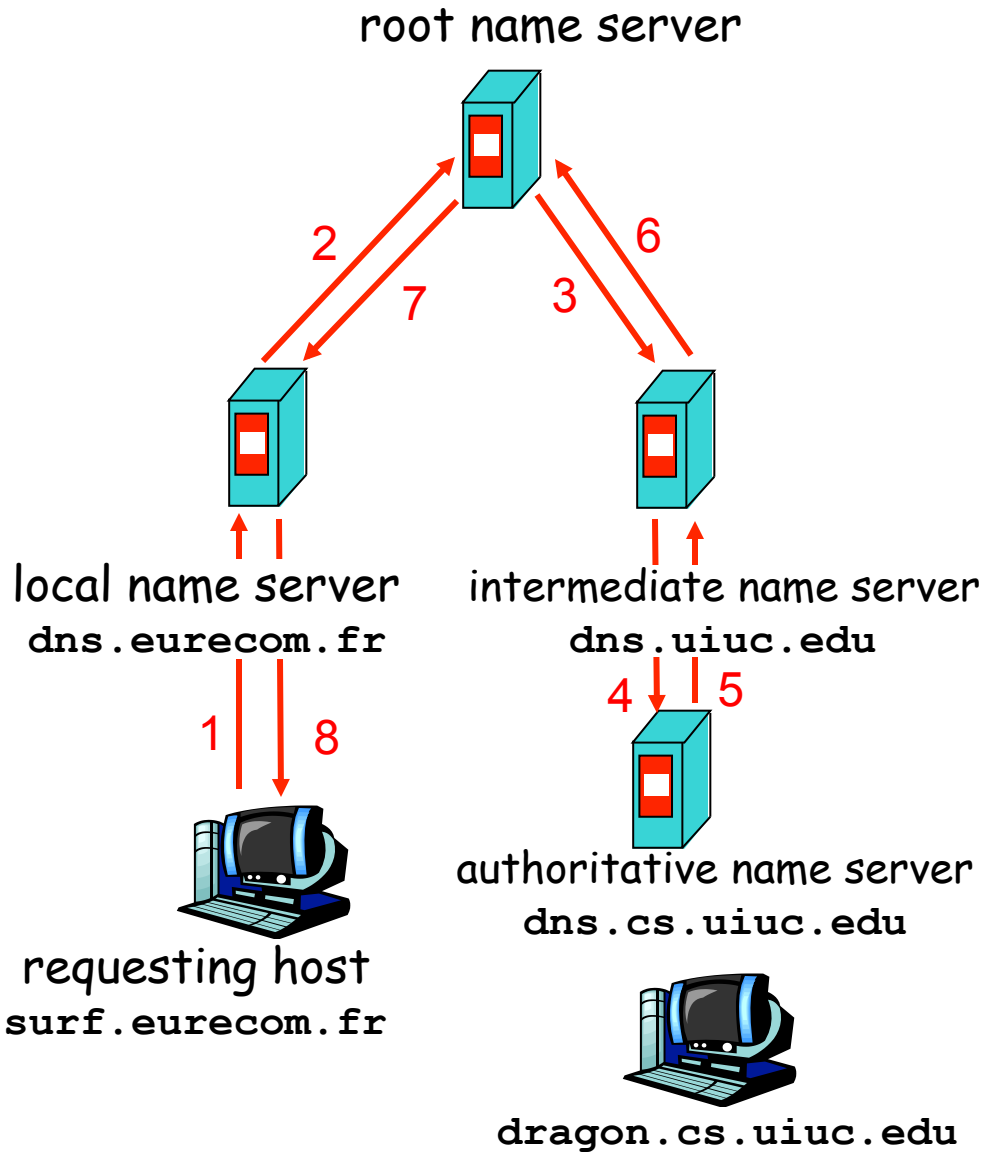


DNS Example

Root name server:

- may not know the authoritative name server
- may know *intermediate name server:* whom to contact to find authoritative name server
- Server Hierarchy parallels URL hierarchy





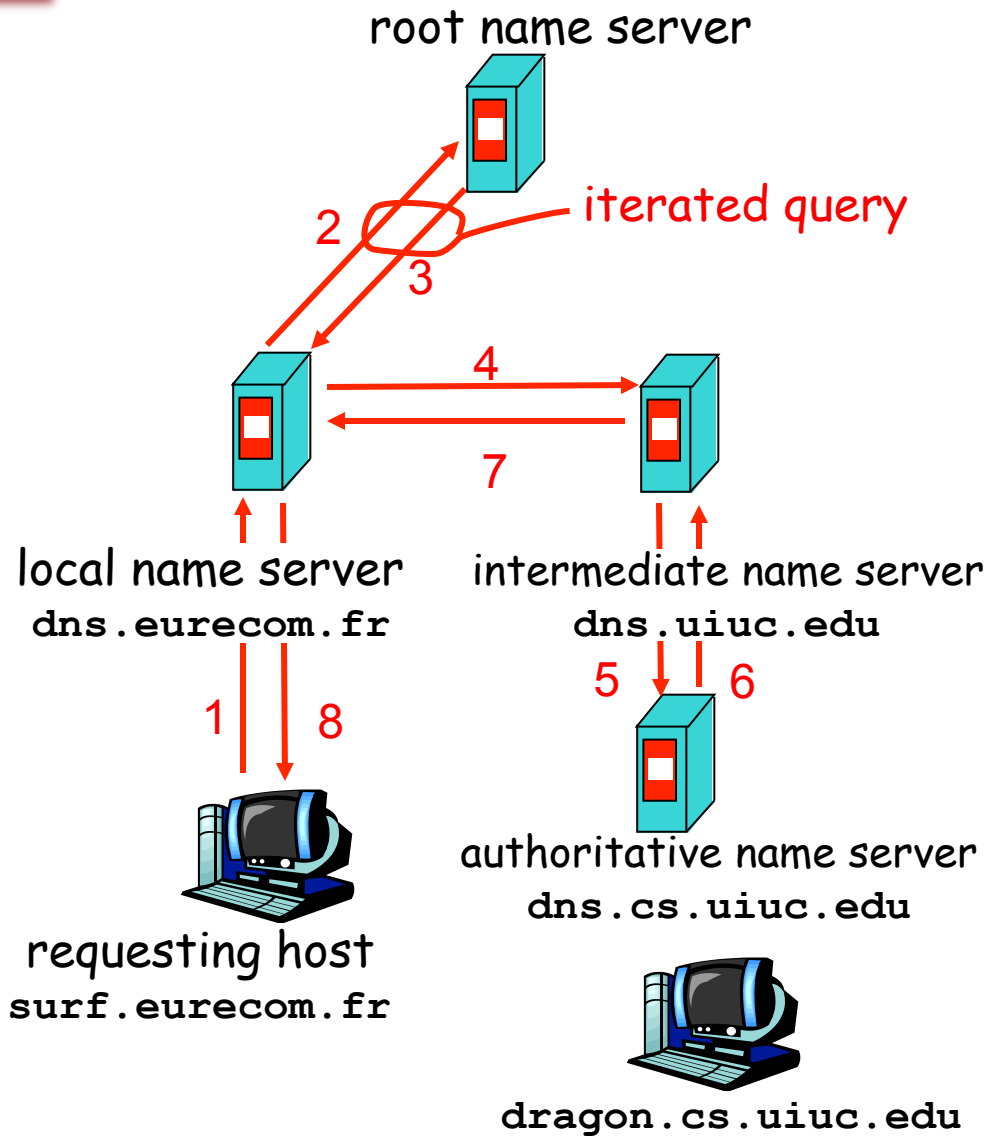
DNS: Iterated Queries

recursive query:

- puts burden of name resolution on servers along the way
- may fail if a server does not know next server to contact, or if intermediate server fails

iterated query:

- contacted server replies with name of server to querying server
- "I don't know this resource name, but ask this other server"
- takes longer (more longer replies) but gives client more control



DNS: Caching and Updating Records

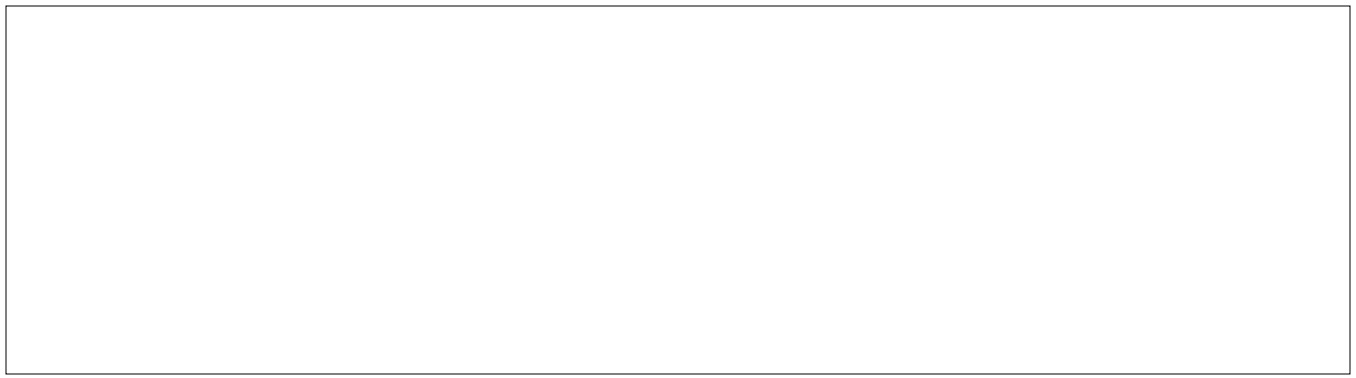
- Once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
- Update/notify mechanisms: insert new DNS entries
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html
  - Akamai/CDNs used this extensively to redirect HTTP requests to nearby caching servers

Summary

- **Midterm next Tuesday October 14th**
  - Location: Here! (1320 DCL)
  - Syllabus: Lectures 1-12, HWs1-2, MP1.
  - Closed book, closed notes. Calculators ok. NO cheatsheets or cellphones or devices.

  1. Multiple choice questions
  2. Big problems: like HW problems, either design or application
- **Practice midterm posted on Website (Assignments page) – no solutions will be posted**
  - Please use our office hours!

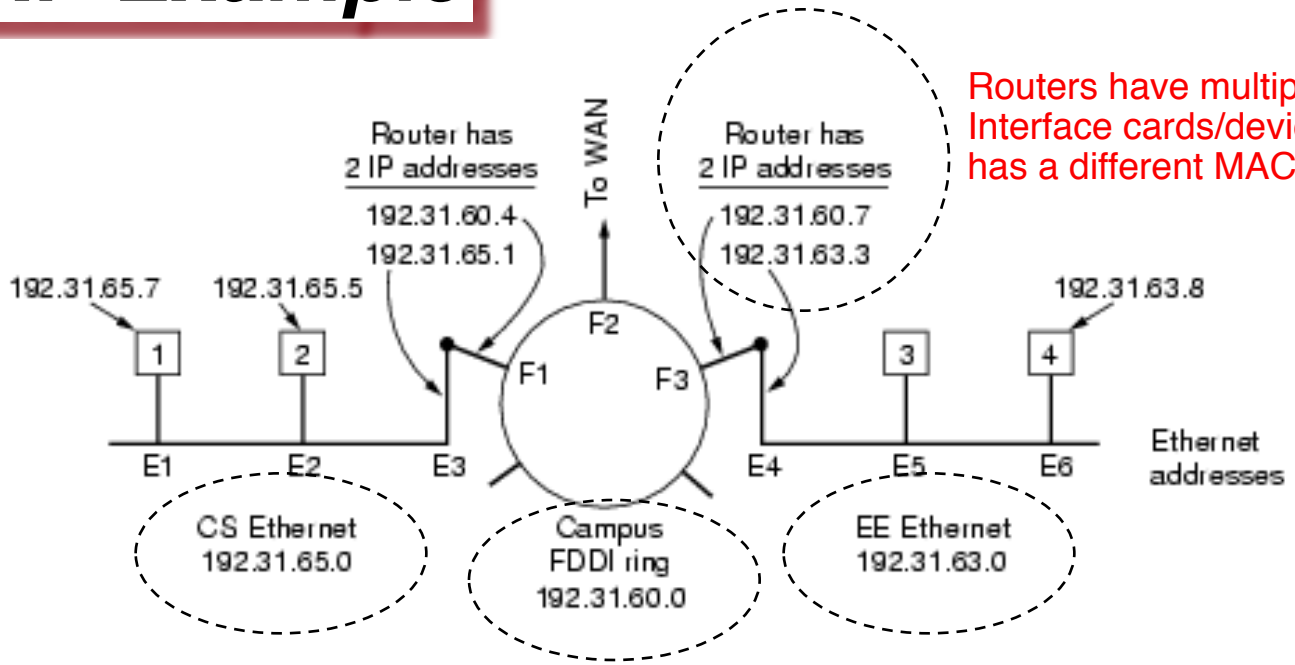
Backup Slides (Not Covered)



ARP: Address Resolution Protocol between IP and Underlying Networks

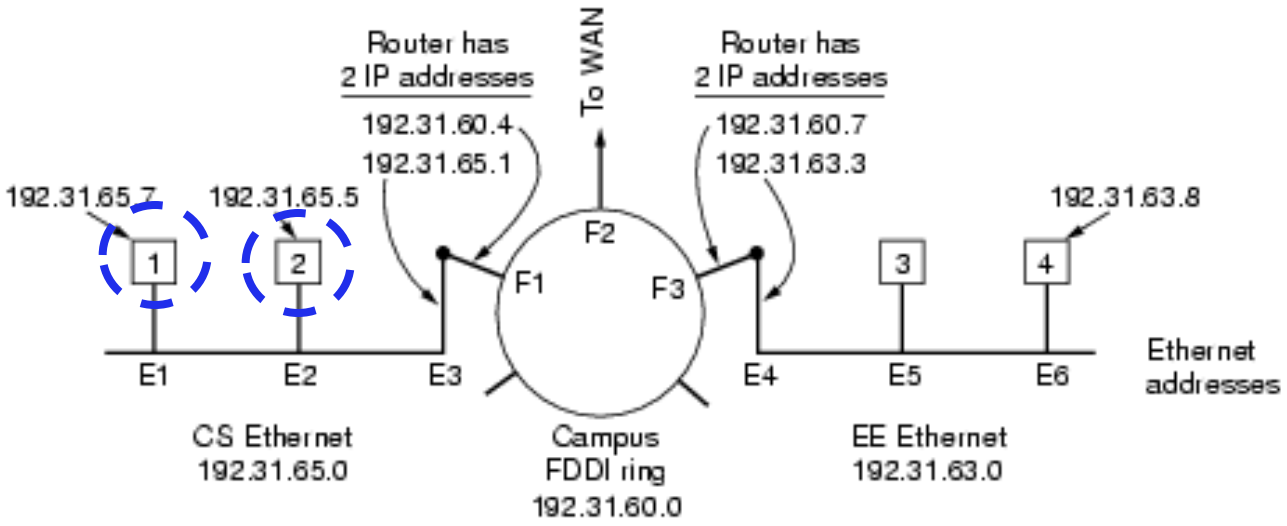
- Most hosts are attached to a LAN by an interface board that only understands LAN addresses. For example, every Ethernet board is equipped with a globally unique 48-bit Ethernet address.
- The boards send and receive frames based on 48-bit Ethernet addresses. They know nothing about the 32-bit IP addresses.
- Address Resolution Protocol (ARP) maps the IP addresses onto data link layer addresses (e.g., Ethernet).

ARP Example



Routers have multiple network Interface cards/devices. Each interface has a different MAC/IP address.

ARP



Suppose host 1's IP layer (192.31.65.7) gets a packet from its transport layer destined for 192.31.65.5 (host 2)?

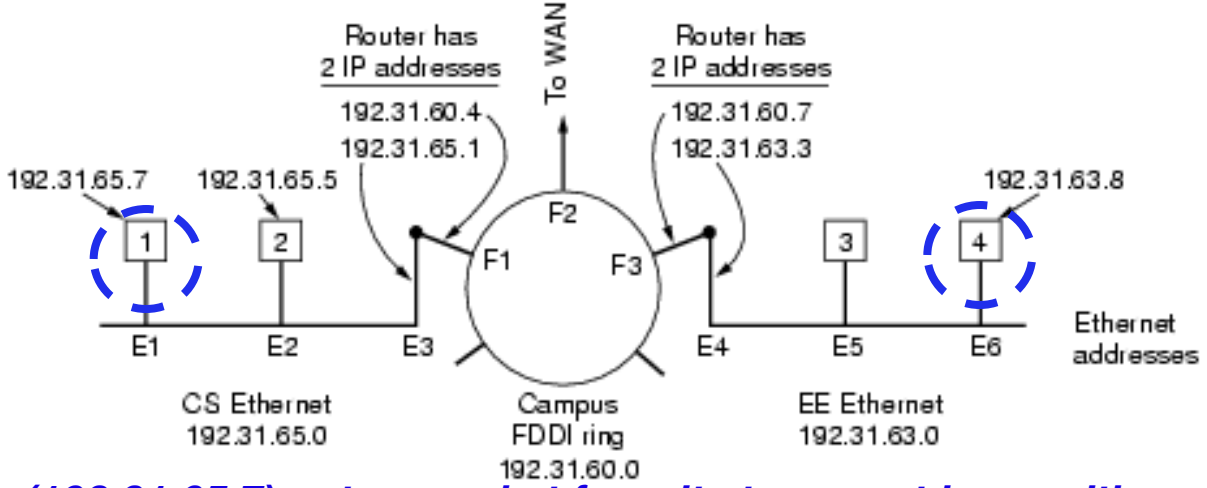
- Host 1's IP layer broadcasts an ARP packet onto the Ethernet asking: ``Who owns IP address 192.31.65.5?''
- Host 2 responds with its Ethernet address (E2).
- The IP layer on host 1 builds an Ethernet frame addressed to E2, puts the IP packet in the payload field, and transmits it on the Ethernet.
- The Ethernet board of host 2 detects this frame and causes an interrupt, to deliver the packet to the IP layer on host 2.

Thus, the packet is transmitted from host 1's IP layer to host 2's IP layer

ARP

- The performance of ARP can be improved by caching the broadcast results.
- Host 1 can include its own IP to Ethernet mapping in the ARP packet.

ARP



Suppose host 1's IP layer (192.31.65.7) gets a packet from its transport layer with destination address set to 192.31.63.8 (host 4)?

- Host 1's IP layer broadcasts an ARP packet onto the Ethernet asking: ``Who owns IP address 192.31.63.8?"
- *Router E3/F1* responds with its Ethernet address (E3).
- The IP layer on host 1 transmits an Ethernet frame addressed to E3
- The E3 Ethernet board of router F1 receives the frame and delivers it to the IP layer on F1
- F1's IP layer ***knows*** from the destination address of 192.31.63.8 in the packet that it has to be next sent to 192.31.60.7.
- F1's IP layer sends an ARP on the FDDI ring for IP address 192.31.60.7.
- Router E4/F3 replies.
- F1's IP layer transmits an FDDI frame (containing the packet) addressed to F3
- The FDDI board of F3 receives the frame, and delivers it to the IP layer.
- F3 ***knows*** from the destination address of 192.31.63.8 in the packet that it has to be next sent out to 192.31.63.8 through the interface E4
- F3 does an ARP on the EE Ethernet for IP address 192.31.63.8
- Host 4 responds with E6
- F3 transmits the packet inside an Ethernet frame addressed to E6, and it reaches host 4's IP layer