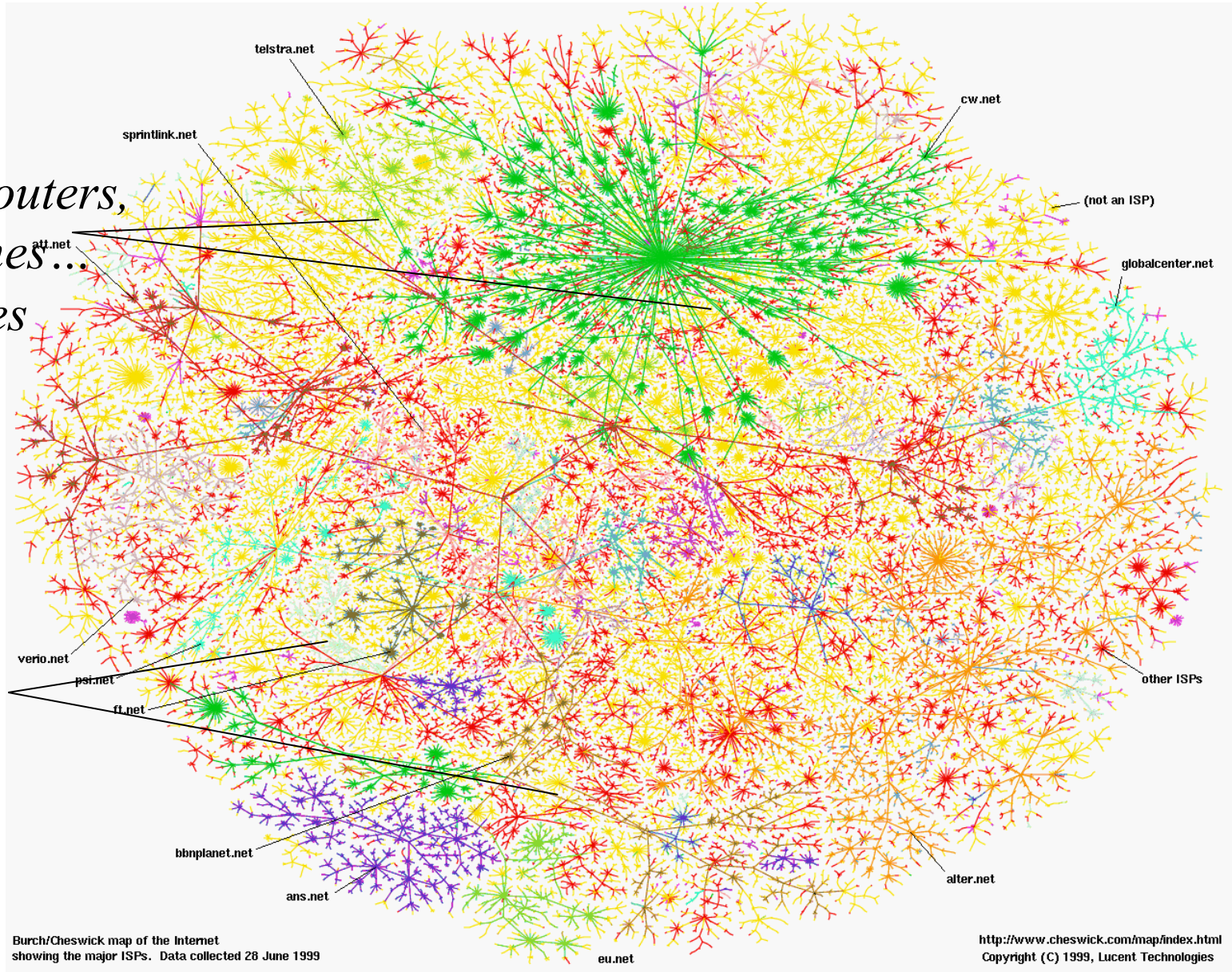
CS425 / CSE424 / ECE428 — Fall 2011 — Distributed Systems

Networking

Some material derived from slides by I. Gupta, K. Nahrstedt, S. Mitra, N. Vaidya, M.T. Harandi, J. Hou

PCs,routers, switches… =nodes

links= *edges*

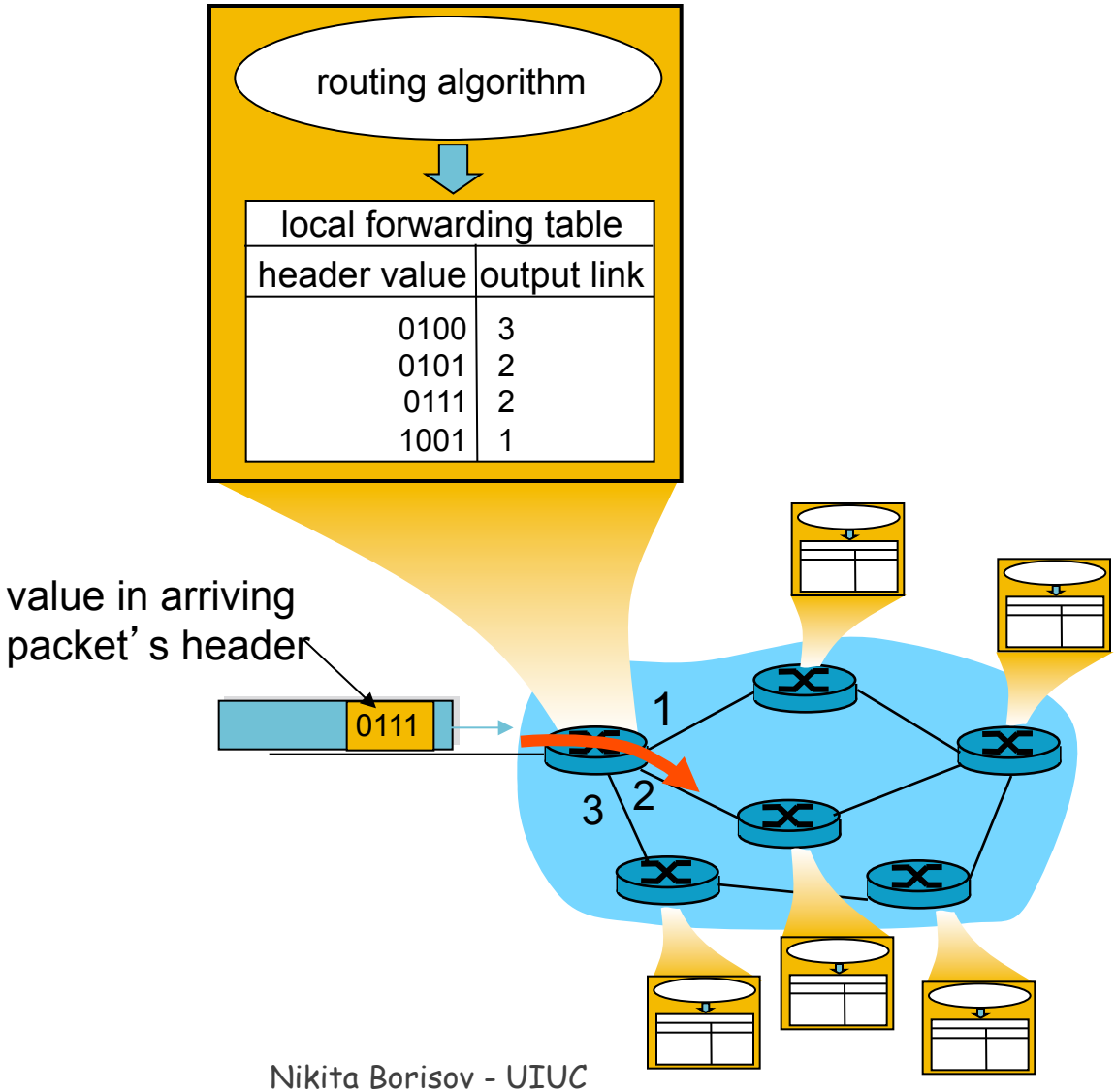


The Internet (Internet Mapping Project, color coded by ISPs)

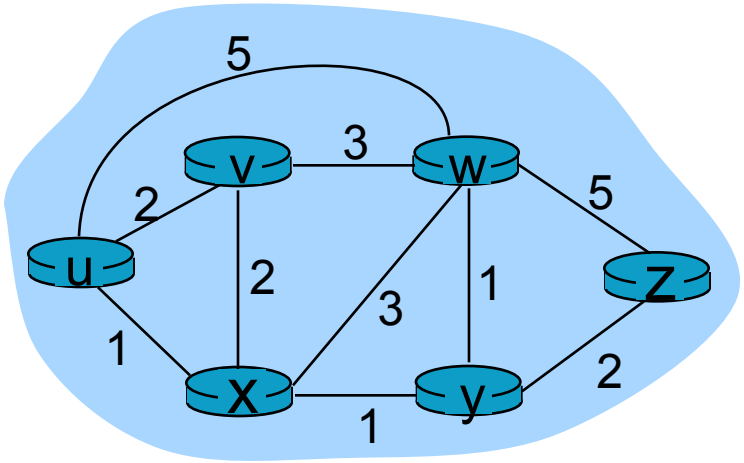
Routing

- Routing algorithms
  - Link state
  - Distance Vector

Interplay between routing and forwarding



Graph abstraction



Graph: G = (N,E)

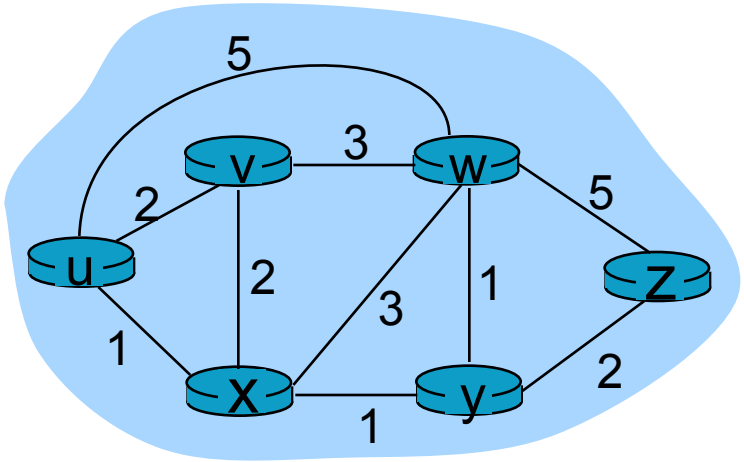
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

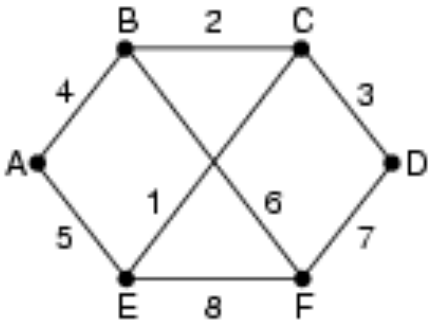
- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

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.



(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

- Each router must
  - 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.
  - 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 a reasonable delay estimate.
  - Construct a packet telling all it has just learned.
    - Broadcast this packet

Link State Routing

- Broadcast the LSA packet to all other routers.
  - Each packet contains a sequence number that is incremented for each new LSA packet sent.
  - Each router keeps track of all the (source router, sequence) pairs it sees. When a new LSA packet comes in, it is checked against the pairs. If the received packet is new, it is forwarded on all the links except the one it arrived on.
  - The age of each packet is included and is decremented once per time unit. When the age hits zero, the information is discarded. Initial age = very high
- 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.

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

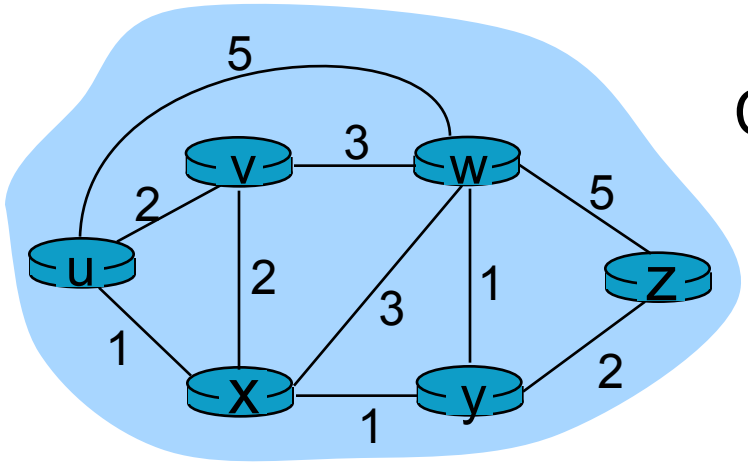
$d_x(y)$:= cost of least-cost path from x to y

Then

$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$

where min is taken over all neighbors of x

Bellman-Ford example (2)



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm (3)

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When node a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

Under minor, natural conditions, the estimate $D_x(y)$ *converge the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

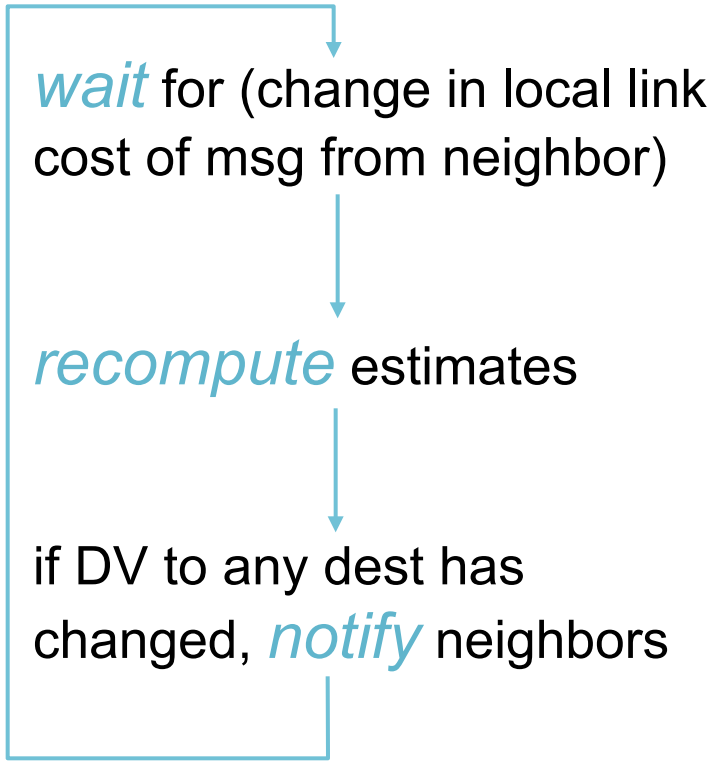
Iterative,
asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



u's DV

u	0
v	2
w	5
x	1
y	∞
z	∞

v's DV

u	2
v	0
w	3
x	2
y	∞
z	∞

w's DV

u	5
v	3
w	0
x	3
y	1
z	5

x's DV

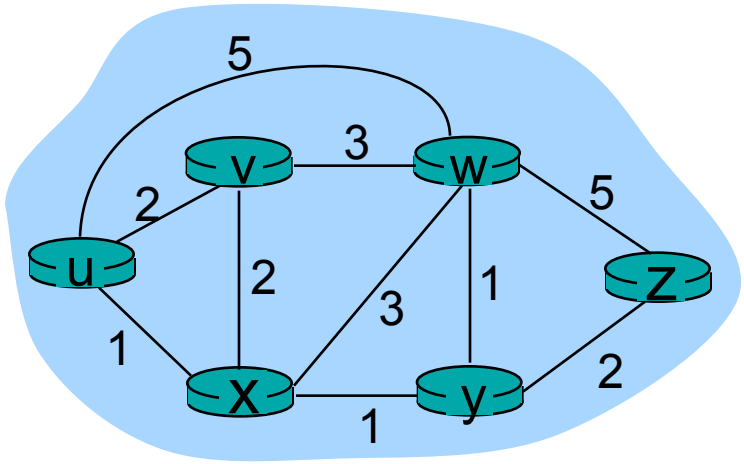
u	1
v	2
w	3
x	0
y	1
z	∞

y's DV

u	∞
v	∞
w	1
x	1
y	0
z	2

z's DV

u	∞
v	∞
w	5
x	∞
y	2
z	0



u's DV

u	0
v	2
w	5
x	1
y	∞
z	∞

v's DV

u	2
v	0
w	3
x	2
y	∞
z	∞

w's DV

u	5
v	3
w	0
x	3
y	1
z	5

x's DV

u	1
v	2
w	3
x	0
y	1
z	∞

y's DV

u	∞
v	∞
w	1
x	1
y	0
z	2

z's DV

u	∞
v	∞
w	5
x	∞
y	2
z	0

2

5

3+2

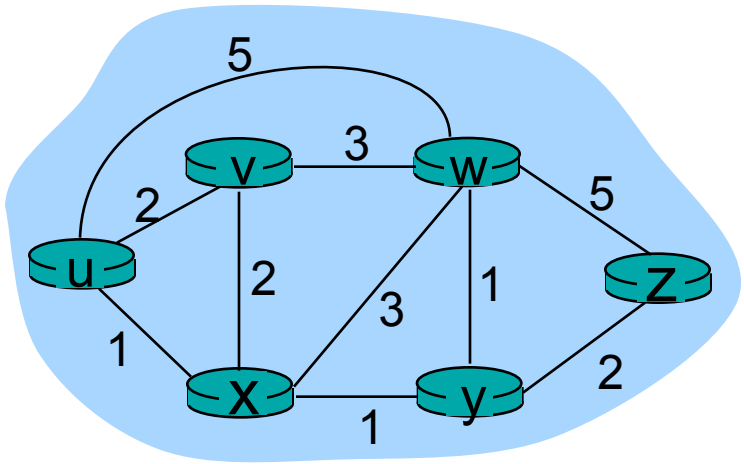
3+5

1+2

1+3

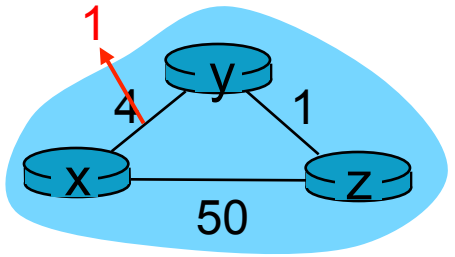
u's updated DV

u	
v	2
w	4
x	
y	
z	



Distance Vector: link cost changes

Link cost changes:
node detects local link cost change
updates routing info, recalculates distance vector
if DV changes, notify neighbors



"good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: link cost changes

Link cost changes:

good news travels fast

bad news travels slow - "count to infinity" problem!

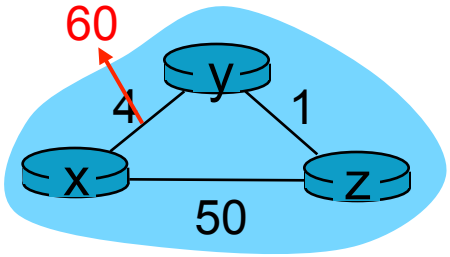
44 iterations before algorithm stabilizes

Poisoned reverse:

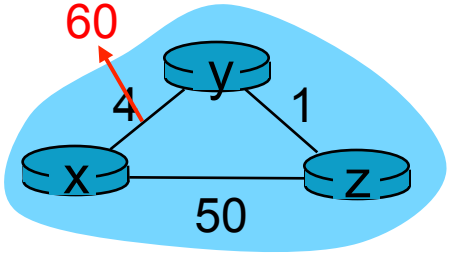
If Z routes through Y to get to X :

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

will this completely solve count to infinity problem?



Count-to-infinity Problem



y's DV

x	4
z	1

dist=60

dist=5+1

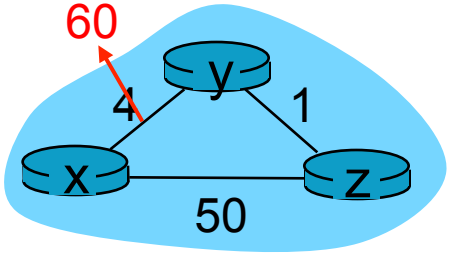
x's DV

y	4
z	5

z's DV

x	5
y	1

Count-to-infinity Problem



y's DV

x	6
z	1

dist=6+1

z's DV

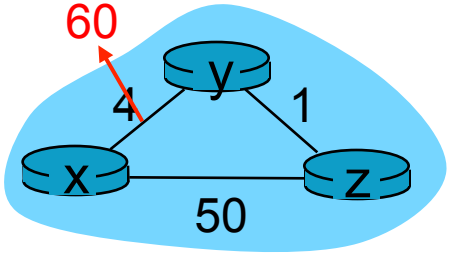
x	5
y	1

x's DV

y	4
z	5

dist=50

Count-to-infinity Problem



y's DV

x	6
z	1

dist=60

dist=7+1

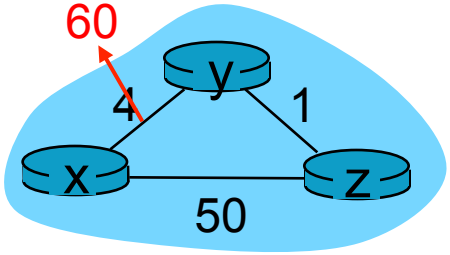
x's DV

y	4
z	5

z's DV

x	7
y	1

Poisoned Reverse



y's DV

x	4
z	1

dist=60

dist=∞+1

x's DV

y	4
z	5

z's DV

x	5
y	1

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

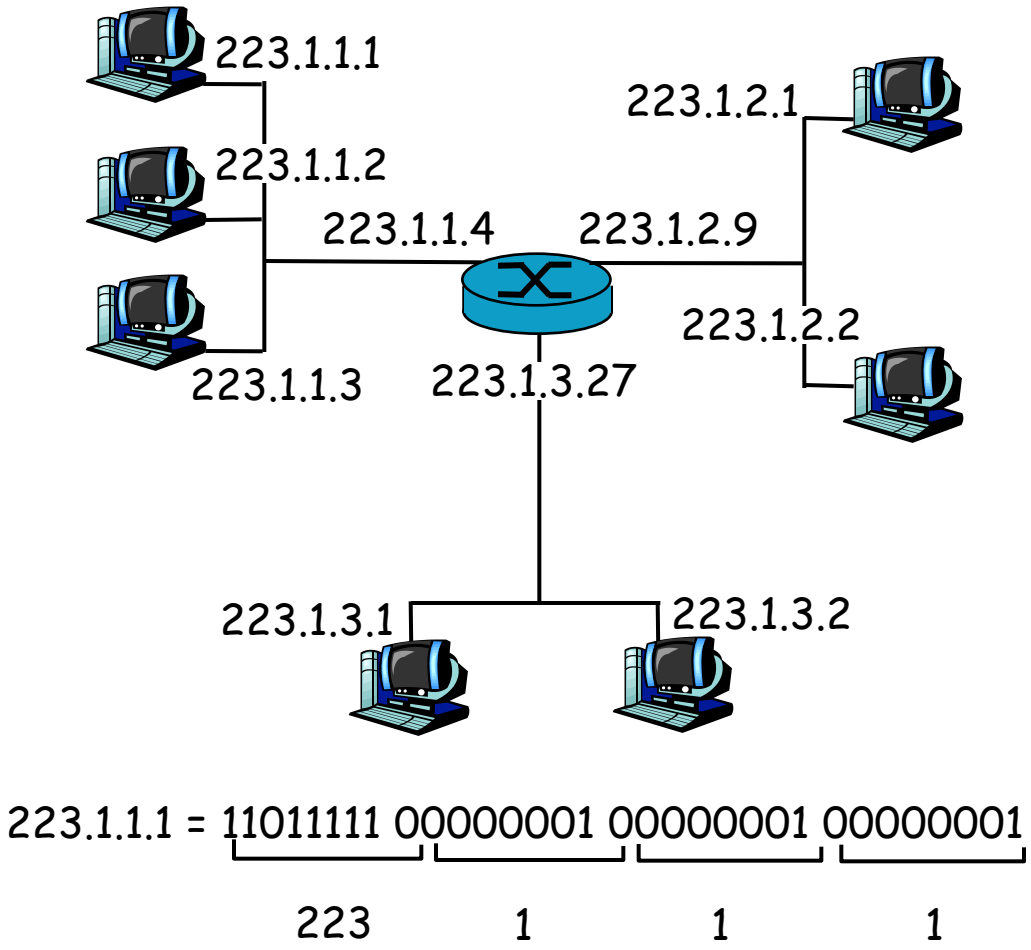
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

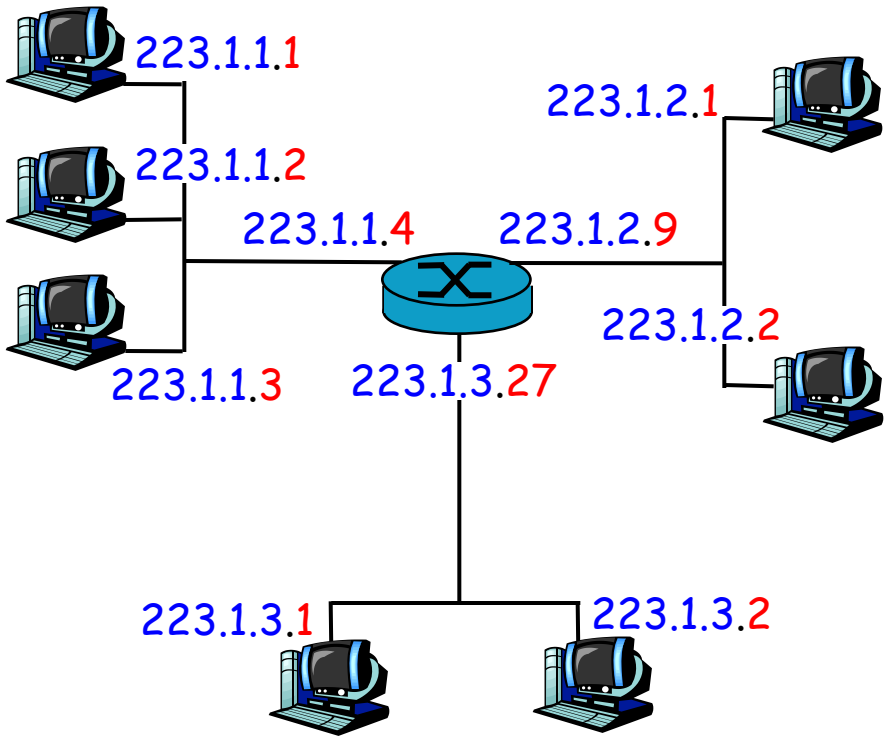
IP Addressing: introduction

- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - routers typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



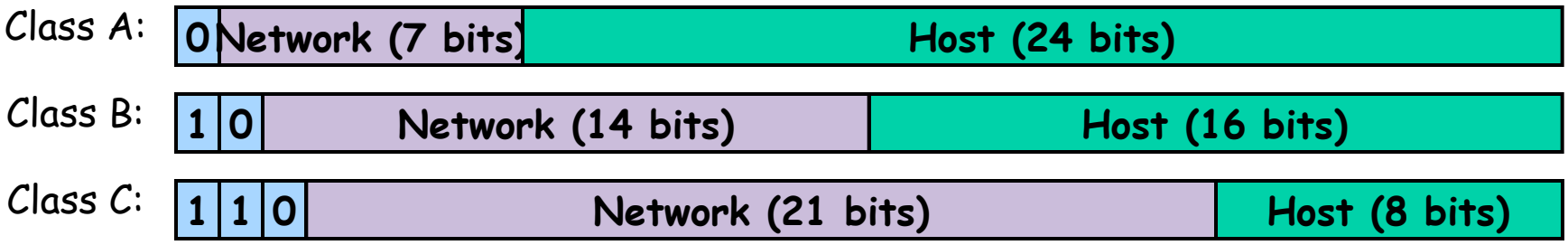
IP networks

- Address has 2 components
  - Network (high-order bits)
  - Host (low-order bits)



IPv4 Address Model

Class	Network ID	Host ID	# of Addresses	# of Networks
A	0 + 7 bit	24 bit	2^{24}-2	126
B	10 + 14 bit	16 bit	65,536 - 2	2^{14}
C	110 + 21 bit	8 bit	256 - 2	2^{21}
D	1110 + Multicast Address		IP Multicast	
E	Future Use			



IP networks

- Class A network: 18.0.0.0 (MIT)
  - www.mit.edu has address 18.7.22.83
- Class B network: 128.174.0.0 (UIUC)
  - www.cs.uiuc.edu has address 128.174.252.84
- Class C network: 216.125.249.0 (Parkland)
  - www.parkland.edu has address 216.125.249.97

CIDR

- 3-class model too inflexible
- CIDR: Classless InterDomain Routing
  - Arbitrary number of bits to specify network
  - Address format: a.b.c.d/x, where x is # bits in network portion

200.23.16.0/23

Classless Domains

- Internet Archive - 207.241.224.0/20
  - 4K hosts
  - 207.241.224.0 - 207.241.239.255
- AT&T - 204.127.128.0/18
  - 16K hosts
  - 204.127.128.0 - 204.127.191.255
- UUNET - 63.64.0.0/10
  - 4M hosts
  - 63.64.0.0 - 63.127.255.255

IP forwarding

- Forwarding table has:
  - Network number
  - Interface
- Avoid having to store 4 billion entries
  - But there are still 2 million class C's
  - …and perhaps more CIDR networks

Hierarchical Routing

Our routing study thus far - idealization
all routers identical
network "flat"
… *not* true in practice

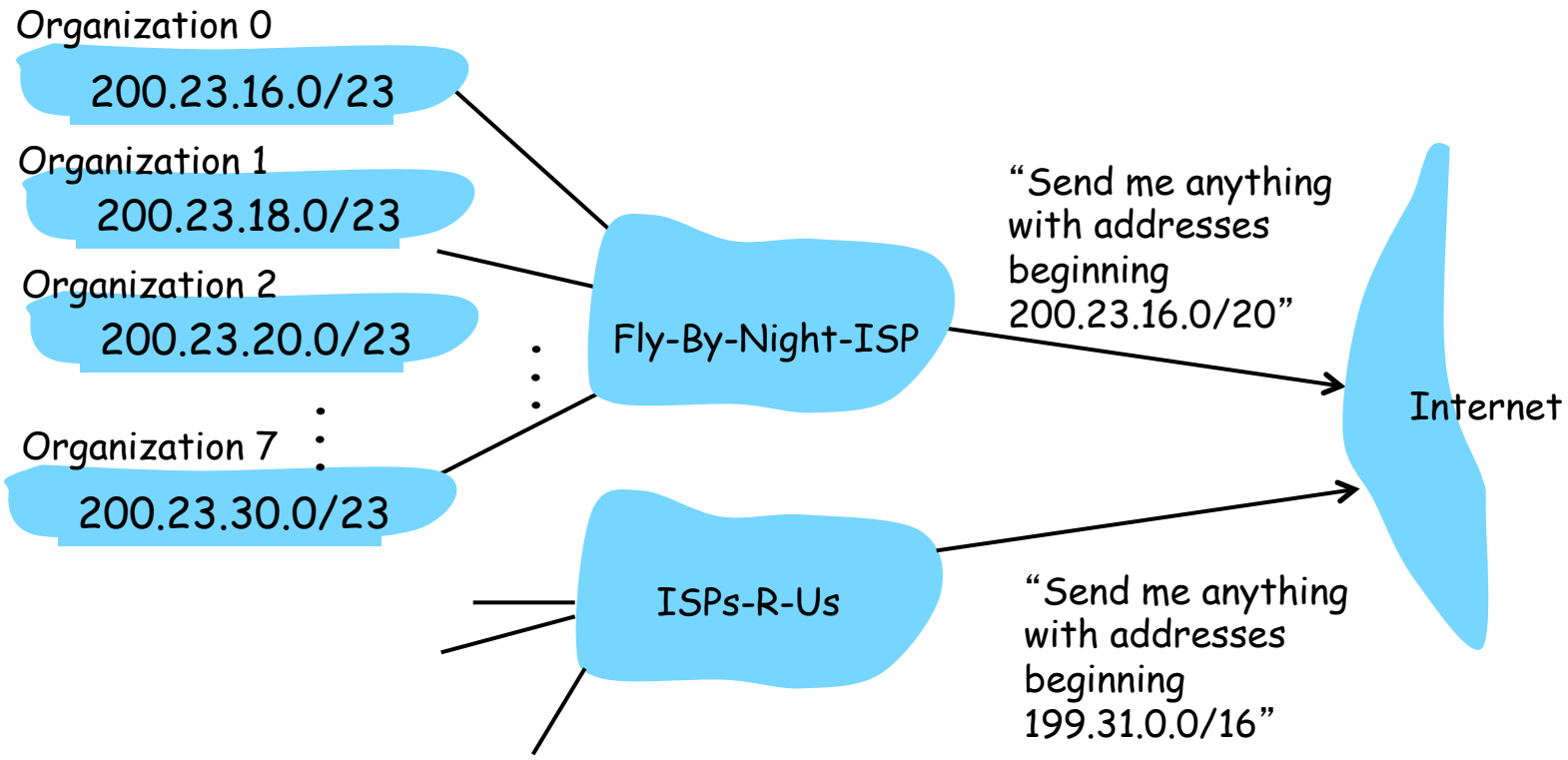
scale: with 200 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Networks

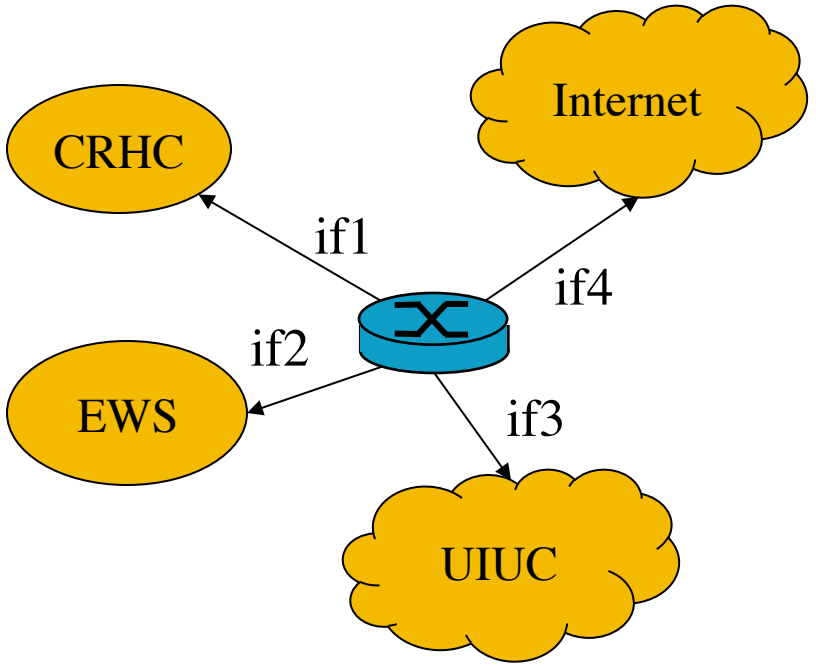


Subnetting

- UIUC - 130.126.0.0/16
  - 130.126.0.0 - 130.126.255.255
- CRHC - 130.126.136.0/21
  - 130.126.136.0 - 130.126.143.255
- EWS - 130.126.160.0/21
  - 130.126.160.0 - 130.126.167.255

Forwarding Tables

130.126.136.0/21	if1
130.126.160.0/21	if2
130.126.0.0/16	if3
0.0.0.0/0	if4



- Most specific rule is used
- Most hosts outside of the core have default rules

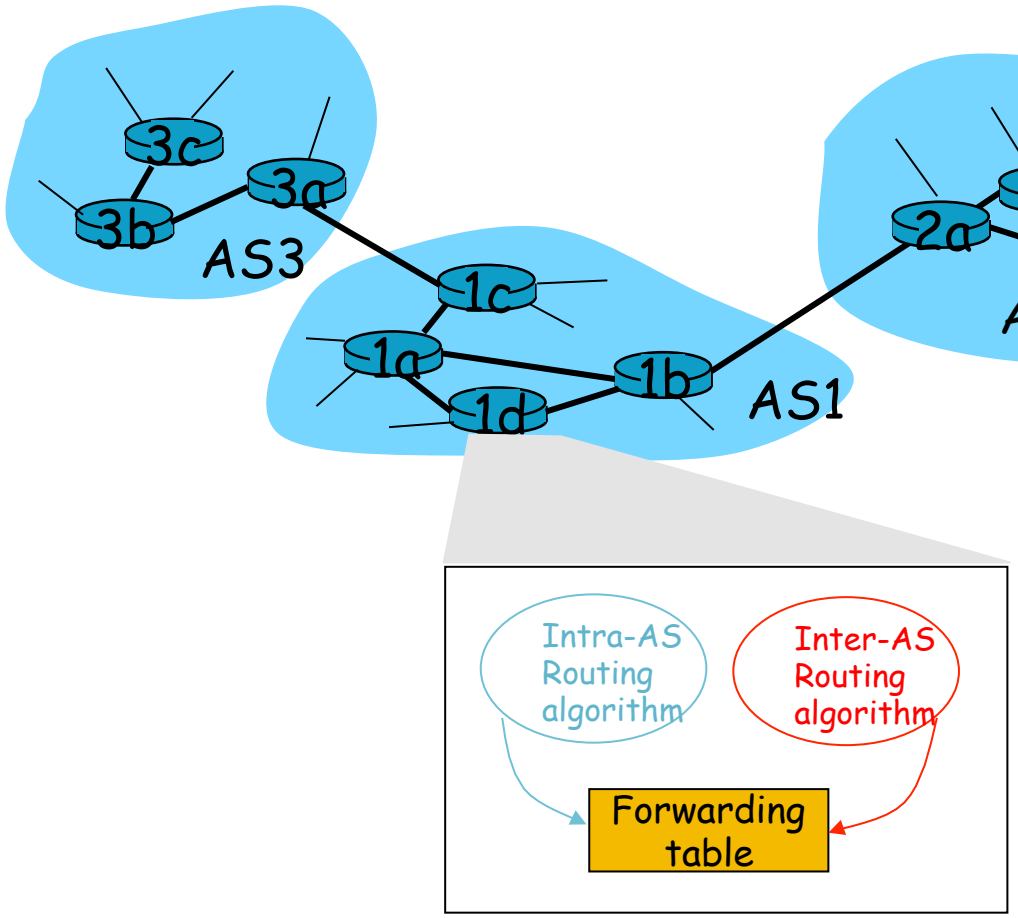
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router

- Direct link to router in another AS

Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

DNS: Domain Name System

People: many identifiers:

- SSN, name, UIN, etc.

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- Resource "name", e.g., URL 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 server has all name-to-IP address mappings
- Hierarchy of name servers

authoritative name servers:

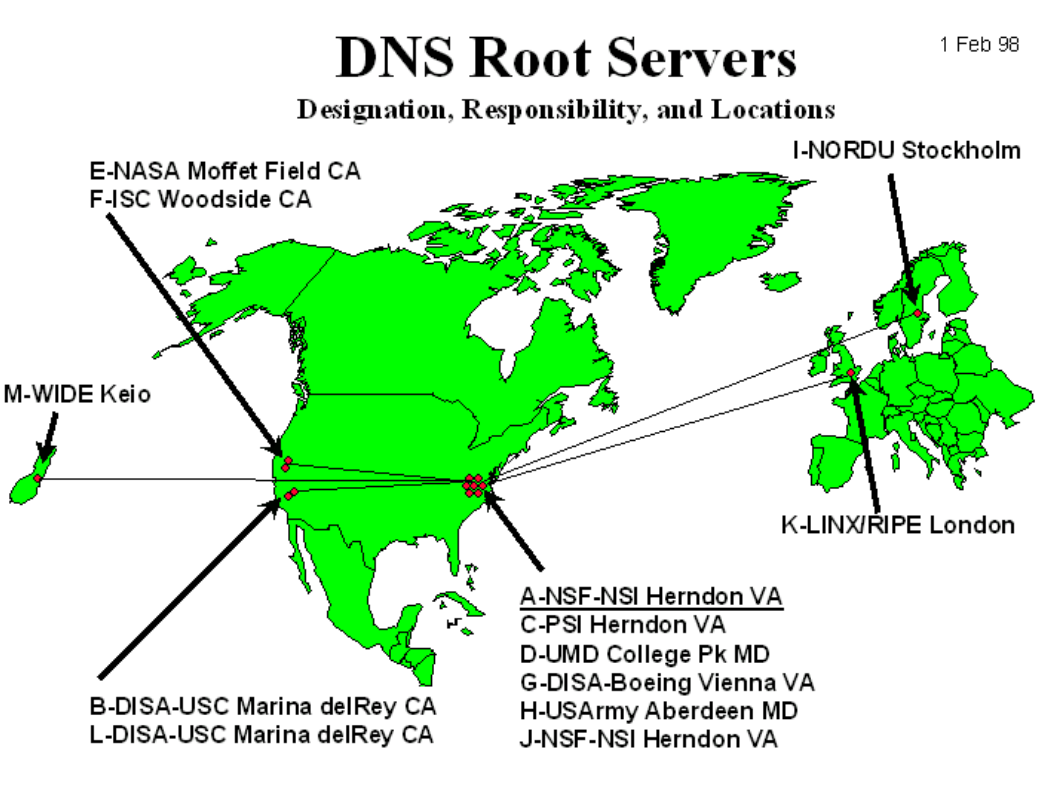
- for a resource, stores the mapped IP address for that resource

local name servers:

- each institution/company/ISP owns a *local (default) name server*
- host DNS query first goes to local name server
- local name server might be caching an answer

DNS: Root Name Servers

- contacted by local name server that can not resolve query
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- ~ dozen root name servers worldwide (as of ’98)

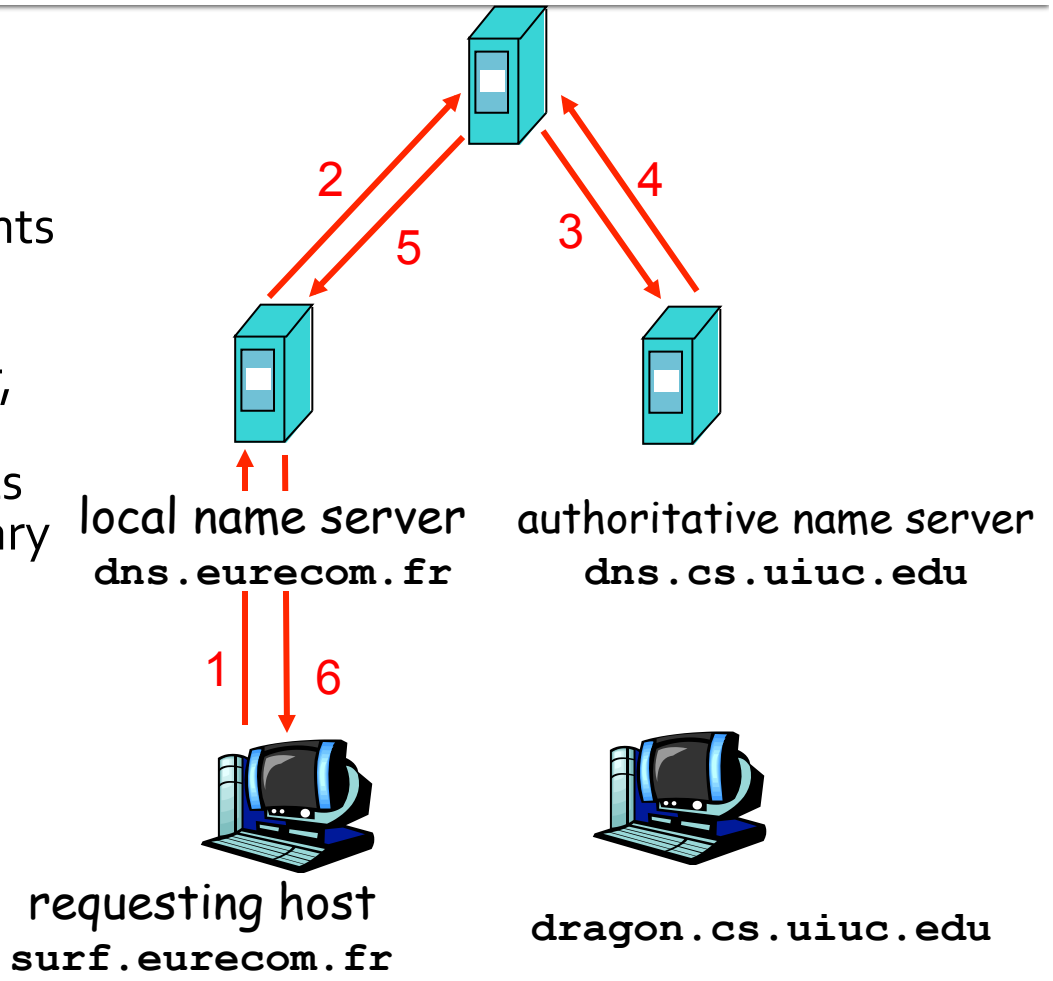


Simple DNS Example

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

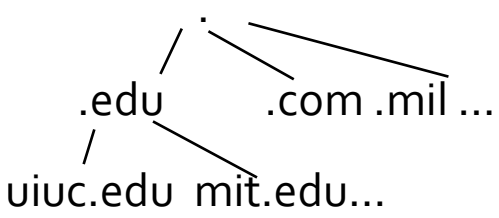
Answer returned by first server that is caching the mapping tuple

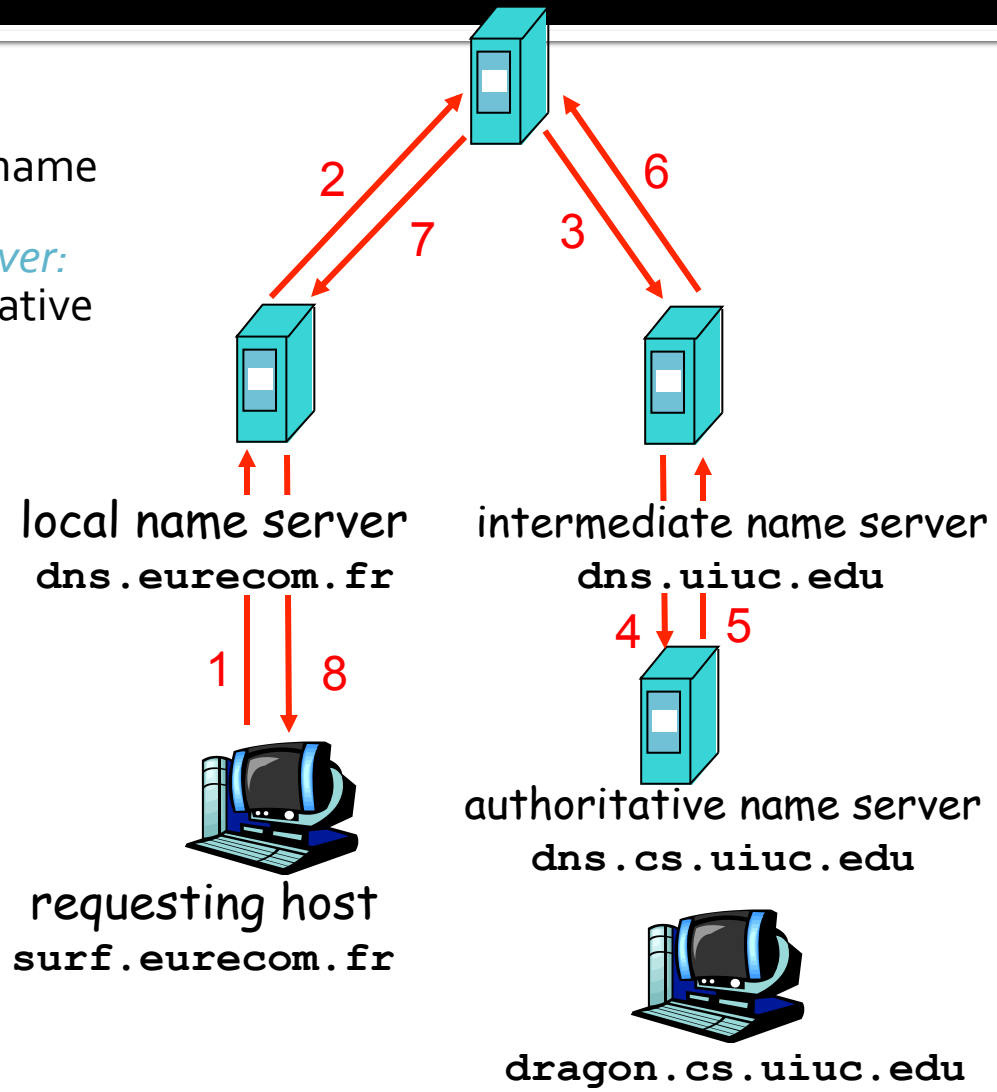


DNS Example

Root name server:

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





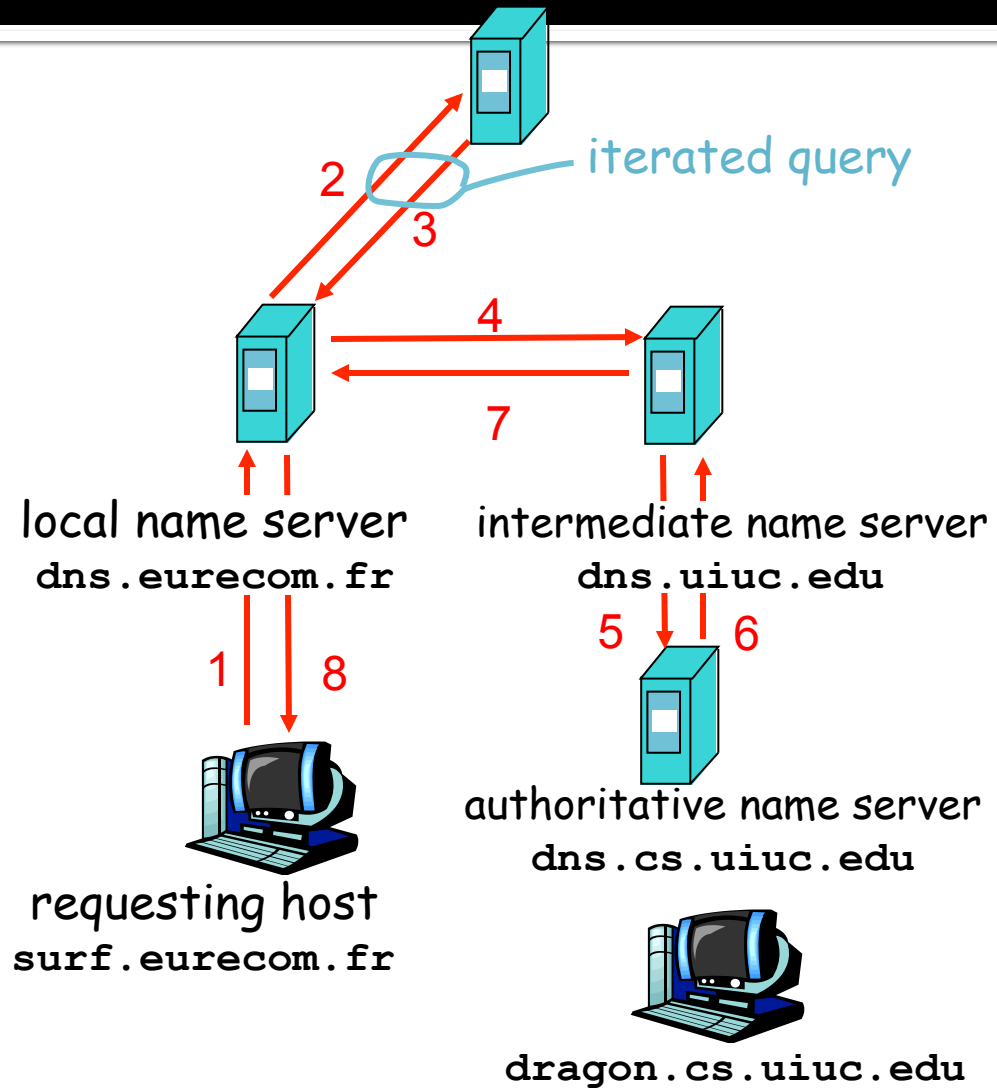
DNS: Iterated Queries

recursive query:

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

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 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
  - Rarely update for most websites
  - Until Akamai realized otherwise (their first version uses update extensively)