Computer Networks

Routing

Adrian Sergiu DARABANT

Lecture 9

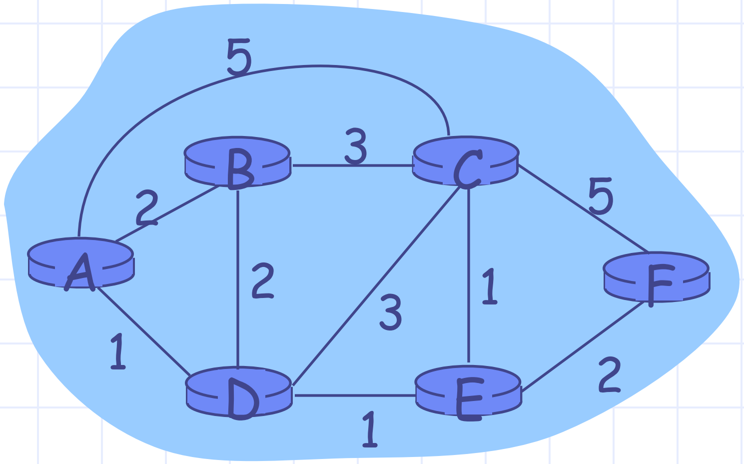
Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level



- "good" path:
  - typically means minimum cost path
  - other def's possible

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

Routing tables - Campus

Destination	Gateway	Genmask	Flags	Metric	Iface
193.226.40.128	*	255.255.255.224	U	0	eth1
192.168.1.0	172.30.5.19	255.255.255.0	UG	0	eth1
192.168.0.0	172.30.1.4	255.255.255.0	UG	0	eth1
193.231.20.0	*	255.255.255.0	U	0	eth0
172.30.0.0	*	255.255.0.0	U	0	eth1
169.254.0.0	*	255.255.0.0	U	0	eth1
127.0.0.0	*	255.0.0.0	U	0	lo
default	193.231.20.9	0.0.0.0	UG	0	eth0

Routing tables (static)

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.25.1	172.30.0.4	255.255.255.255	UGH	0	0	0	Eth1
193.226.40.128	0.0.0.0	255.255.255.224	U	0	0		Eth0
193.0.225.0	0.0.0.0	255.255.255.0	U	0	0		Eth0
193.231.20.0	0.0.0.0	255.255.255.0	U	0	0		Eth0
172.30.0.0	0.0.0.0	255.255.0.0	U	0	0		Eth1
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0		Eth1
0.0.0.0	193.0.225.9	0.0.0.0	UG	0	0		Eth0

The **route** command – (Windows/Linux/other OS)

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

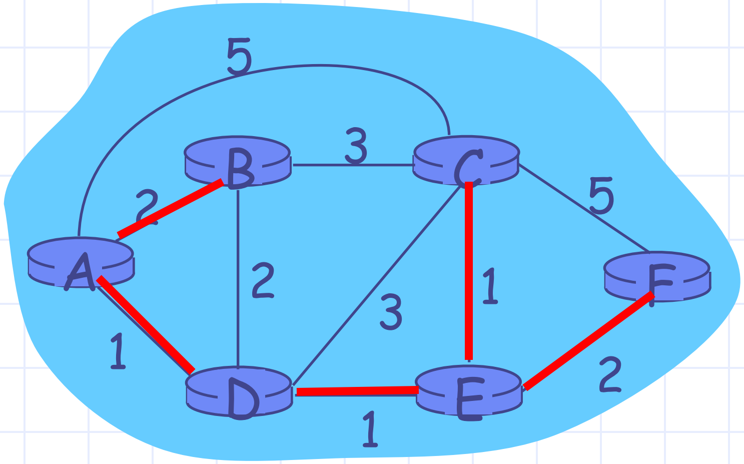
- c(i,j): link cost from node i to j. cost infinite if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

Dijsktra’s Algorithm

```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infinity

Loop
   find w not in N such that D(w) is a minimum
    add w to N
    update D(v) for all v adjacent to w and not in N:
        D(v) = min( D(v), D(w) + c(w,v) )
    /* new cost to v is either old cost to v or known
     shortest path cost to w plus cost from w to v */
  until all nodes in N
```

Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



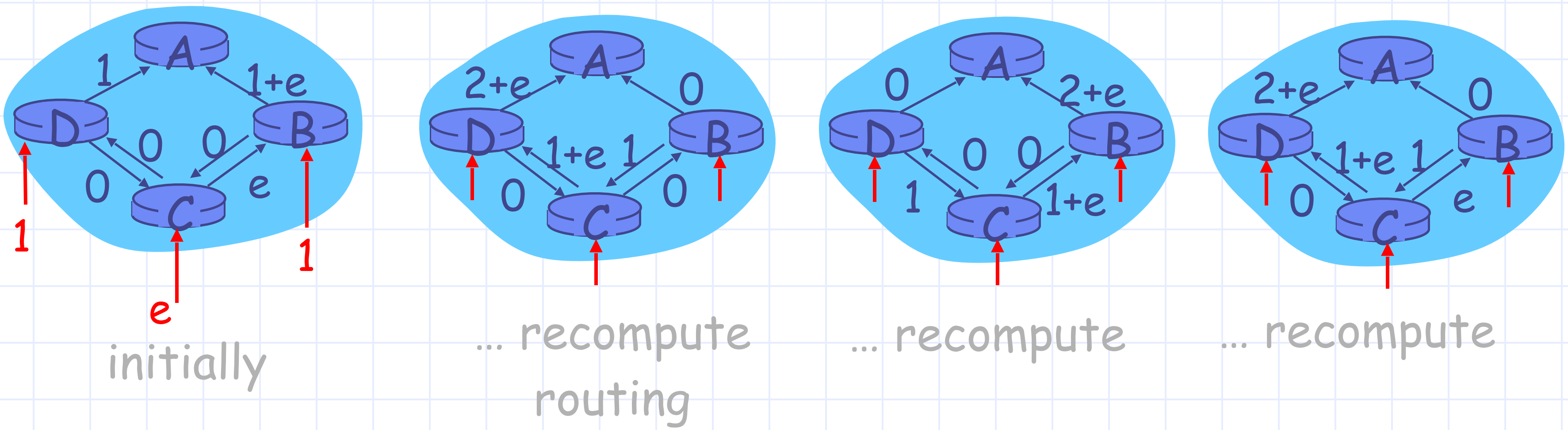
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n*(n+1)/2 comparisons: O(n**2)
- more efficient implementations possible: O(nlogn)

Oscillations possible:

- e.g., link cost = amount of carried traffic



Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- self-terminating: no "signal" to stop

asynchronous:

- nodes need not exchange info/iterate in lock step!

distributed:

- each node communicates only with directly-attached neighbors

Distance Table data structure

- each node has its own

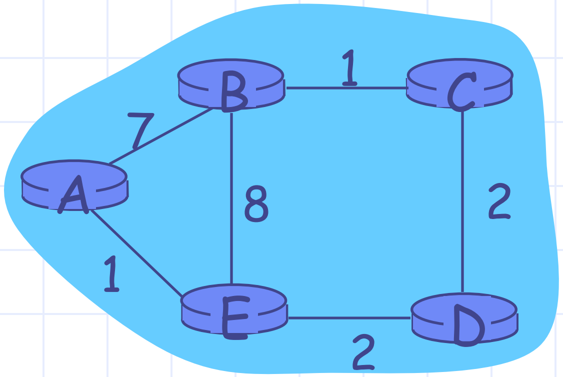
row for each possible destination

- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Distance Table: example



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\}$$
$$= 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\}$$
$$= 2+3 = 5$$ loop!

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\}$$
$$= 8+6 = 14$$ loop!

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

Distance table gives routing table

A	Next Hop	Dist
B	-	7
C	-	∞
D	-	∞
E	-	1

B	Next Hop	Dist
A	-	7
C	-	1
D	-	∞
E	-	8

C	Next Hop	Dist
A	-	∞
B	-	1
D	-	2
E	-	∞

D	Next Hop	Dist
A	-	∞
B	-	∞
C	-	2
E	-	2

E	Next Hop	Dist
A	-	1
B	-	8
C	-	∞
D	-	2

A	Next Hop	Dist
B	-	7
C	B	8
D	E	3
E	-	1

B	Next Hop	Dist
A	-	7
C	-	1
D	C	3
E	-	8

C	Next Hop	Dist
A	B	8
B	-	1
D	-	2
E	D	4

D	Next Hop	Dist
A	E	3
B	C	3
C	-	2
E	-	2

E	Next Hop	Dist
A	-	1
B	-	8
C	D	4
D	-	2

Distance table → Routing table

Distance Vector routing

A	Next Hop	Dist
B	-	7
C	E	5
D	E	3
E	-	1

B	Next Hop	Dist
A	-	7
C	-	1
D	C	3
E	C	5

C	Next Hop	Dist
A	D	5
B	-	1
D	-	2
E	D	4

D	Next Hop	Dist
A	E	3
B	C	3
C	-	2
E	-	2

E	Next Hop	Dist
A	-	1
B	D	5
C	D	4
D	-	2

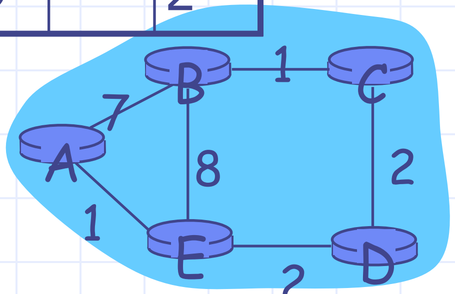
A	Next Hop	Dist
B	E	6
C	E	5
D	E	3
E	-	1

B	Next Hop	Dist
A	C	6
C	-	1
D	C	3
E	C	5

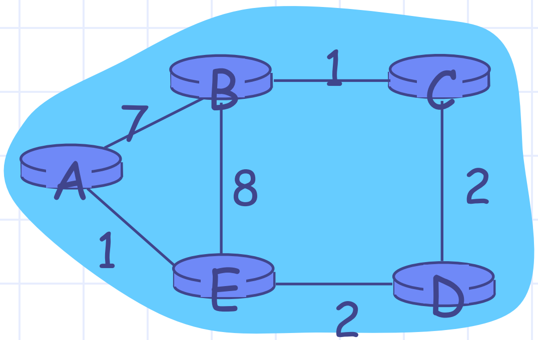
C	Next Hop	Dist
A	D	5
B	-	1
D	-	2
E	D	4

D	Next Hop	Dist
A	E	3
B	C	3
C	-	2
E	-	2

E	Next Hop	Dist
A	-	1
B	D	5
C	D	4
D	-	2



Distance Vector



A	Next Hop	Dist
B	E	**6**
C	E	**5**
D	E	**3**
E	-	**1**

B	Next Hop	Dist
A	C	**6**
C	-	**1**
D	C	**3**
E	C	**5**

C	Next Hop	Dist
A	D	**5**
B	-	**1**
D	-	**2**
E	D	**4**

D	Next Hop	Dist
A	E	**3**
B	C	**3**
C	-	**2**
E	-	**2**

E	Next Hop	Dist
A	-	**1**
B	D	**5**
C	D	**4**
D	-	**2**

Distance Vector Routing: overview

Iterative, asynchronous:
each local iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:

- each node notifies neighbors only when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost of msg from neighbor)

↓

recompute distance table

↓

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm:

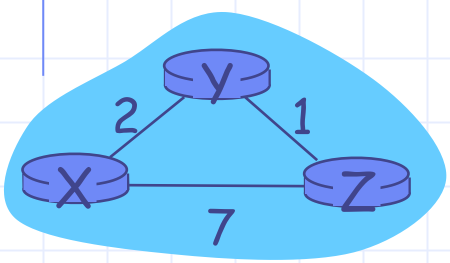
At all nodes, X:

```
Initialization:
 for all adjacent nodes v:
    D^X(*,v) = infinity        /* the * operator means "for all rows" */
    D^X(v,v) = c(X,v)
 for all destinations, y
    send min_w D^X(y,w) to each neighbor  /* w over all X's neighbors */
```

Distance Vector Algorithm (cont.):

```
loop
  wait (until I see a link cost change to neighbor V
        or until I receive update from neighbor V)

  if (c(X,V) changes by d)
    /* change cost to all dest's via neighbor v by d */
    /* note: d could be positive or negative */
    for all destinations y:  DX(y,V) =  DX(y,V) + d

  else if (update received from V wrt destination Y)
    /* shortest path from V to some Y has changed  */
    /* V has sent a new value for its  min_w  DV(Y,w) */
    /* call this received new value is "newval"     */
    for the single destination y: DX(Y,V) = c(X,V) + newval

  if we have a new min_w  DX(Y,w)for any destination Y
     send new value of min_w  DX(Y,w) to all neighbors

forever
```

Distance Vector Algorithm: example



D^X (dest)	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y (dest)	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z (dest)	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X (dest)	cost via Y	cost via Z
Y	2	8
Z	3	7

D^Y (dest)	cost via X	cost via Z
X	2	8
Z	9	1

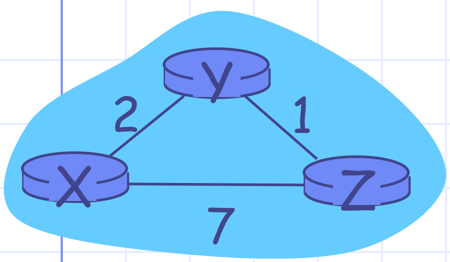
D^Z (dest)	cost via X	cost via Y
X	7	3
Y	9	1

D^X (dest)	cost via Y	cost via Z
Y		
Z		

D^Y (dest)	cost via X	cost via Z
X		
Z		

D^Z (dest)	cost via X	cost via Y
X		
Y		

Distance Vector Algorithm: example



D^X	cost via Y	Z
dest Y	2	∞
dest Z	∞	7

D^Y	cost via X	Z
dest X	2	∞
dest Z	∞	1

D^Z	cost via X	Y
dest X	7	∞
dest Y	∞	1

D^X	cost via Y	Z
dest Y	2	8
dest Z	3	7

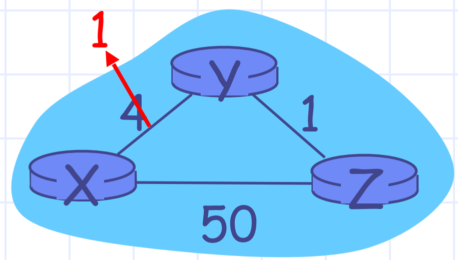
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w\{D^Y(Z,w)\}$$
$$= 2+1 = 3$$

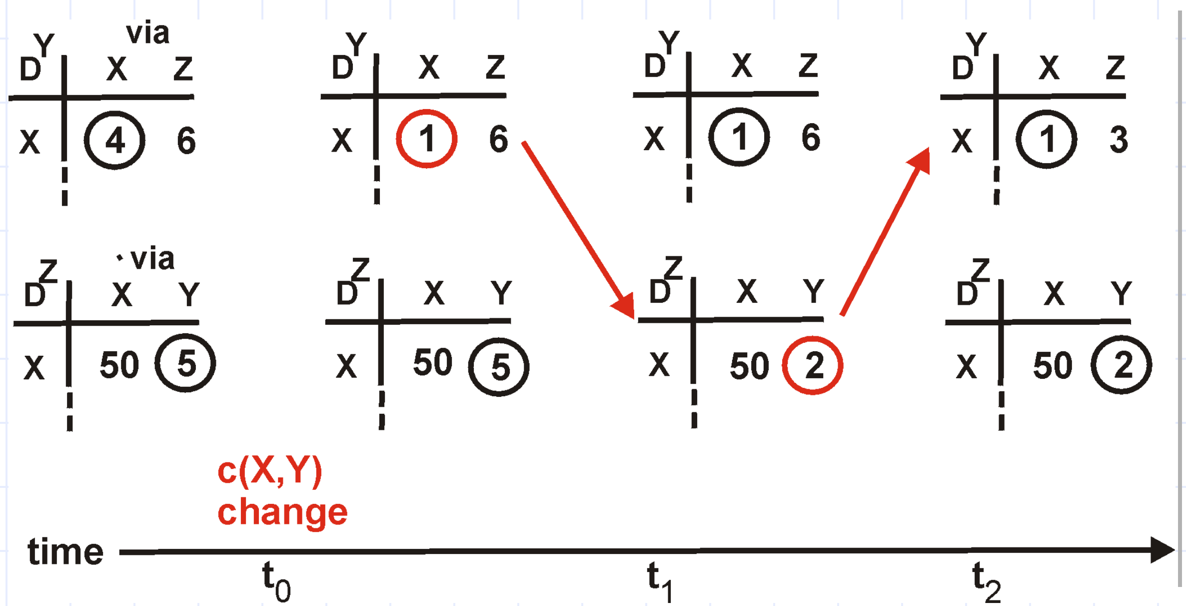
Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)



"good news travels fast"

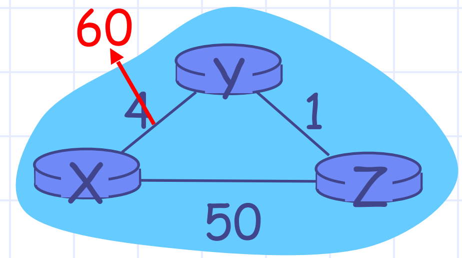


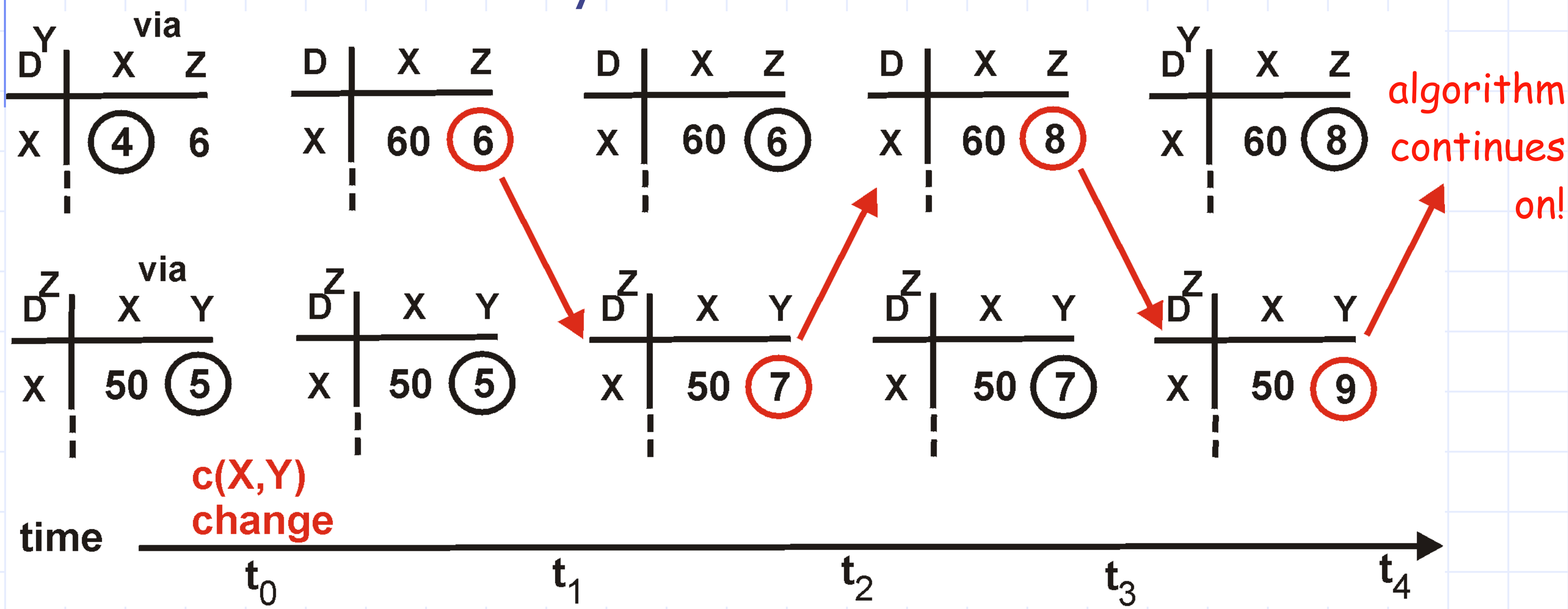
algorithm terminates

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity"

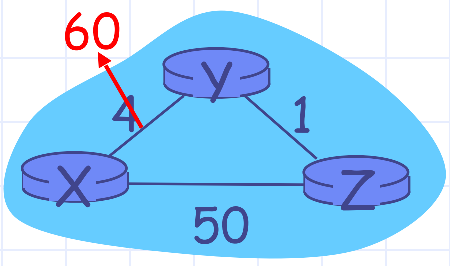


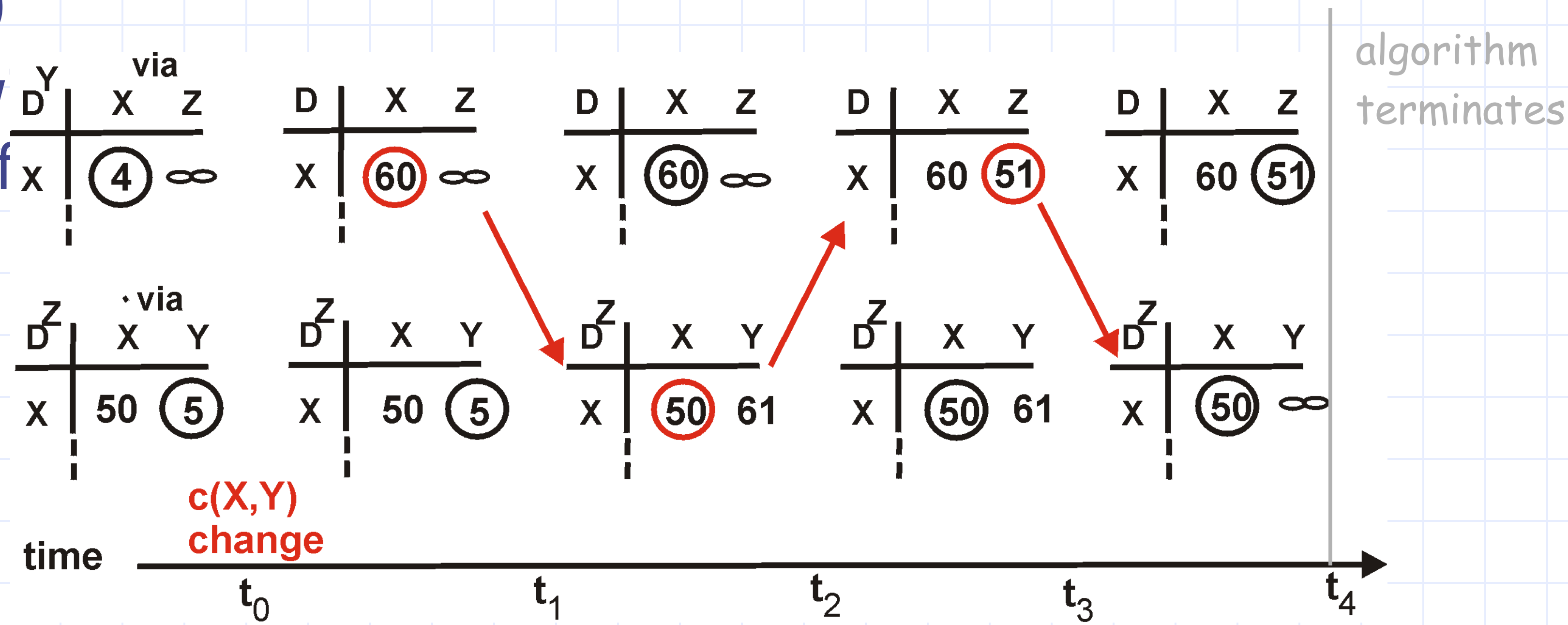


Distance Vector: poisoned reverse

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)
- w
  inf





Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent each
- 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:

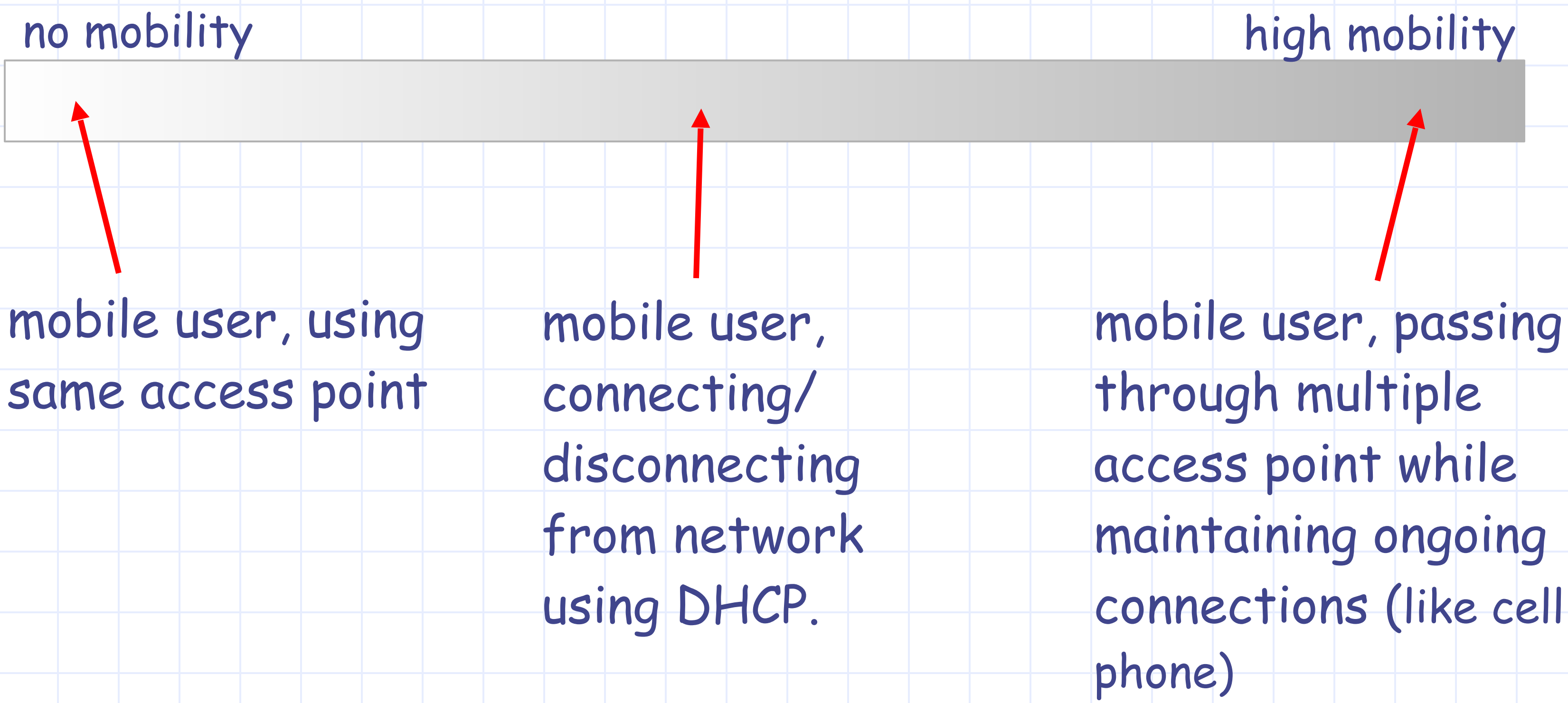
- node can advertise incorrect link cost
- each node computes only its own table

DV:

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

What is mobility?

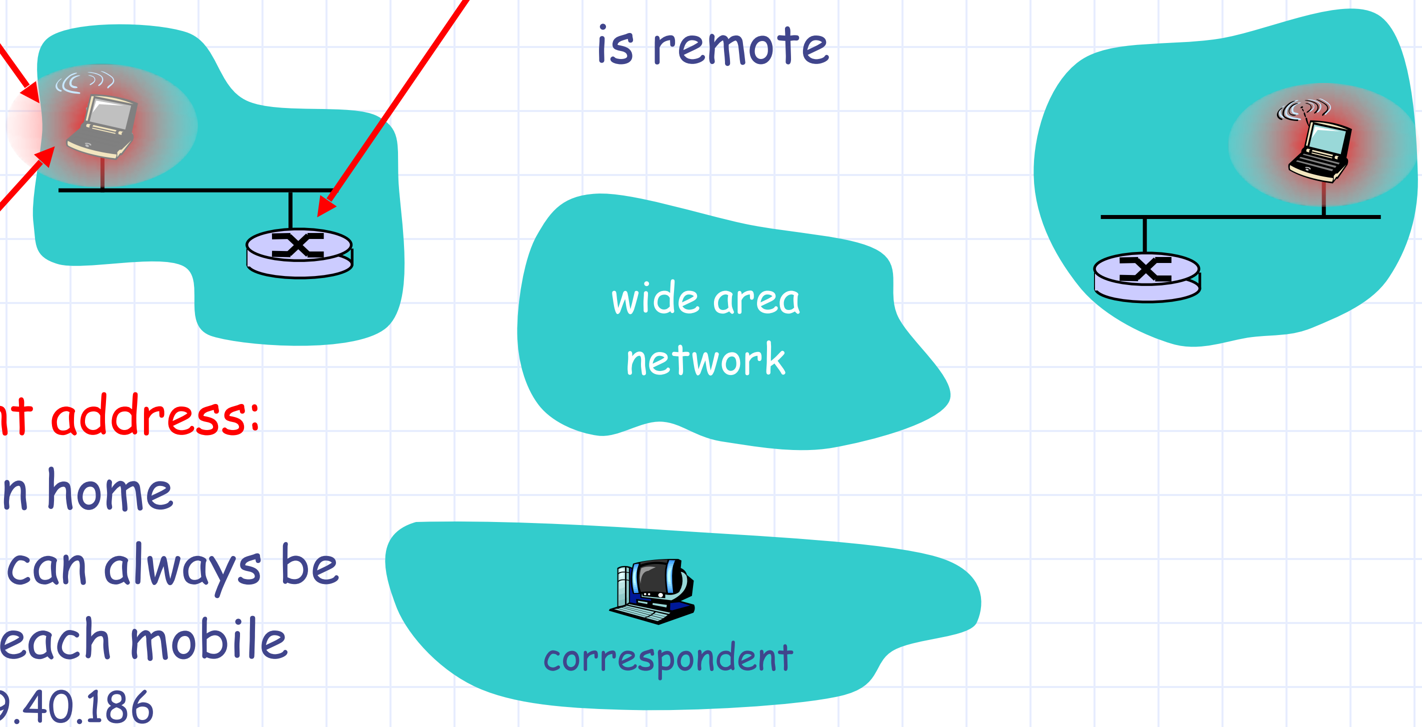
- spectrum of mobility, from the network perspective:



Mobility: Vocabulary

home network: permanent "home" of mobile (e.g., 128.119.40/24)

home agent: entity that will perform mobility functions on behalf of mobile, when mobile is remote



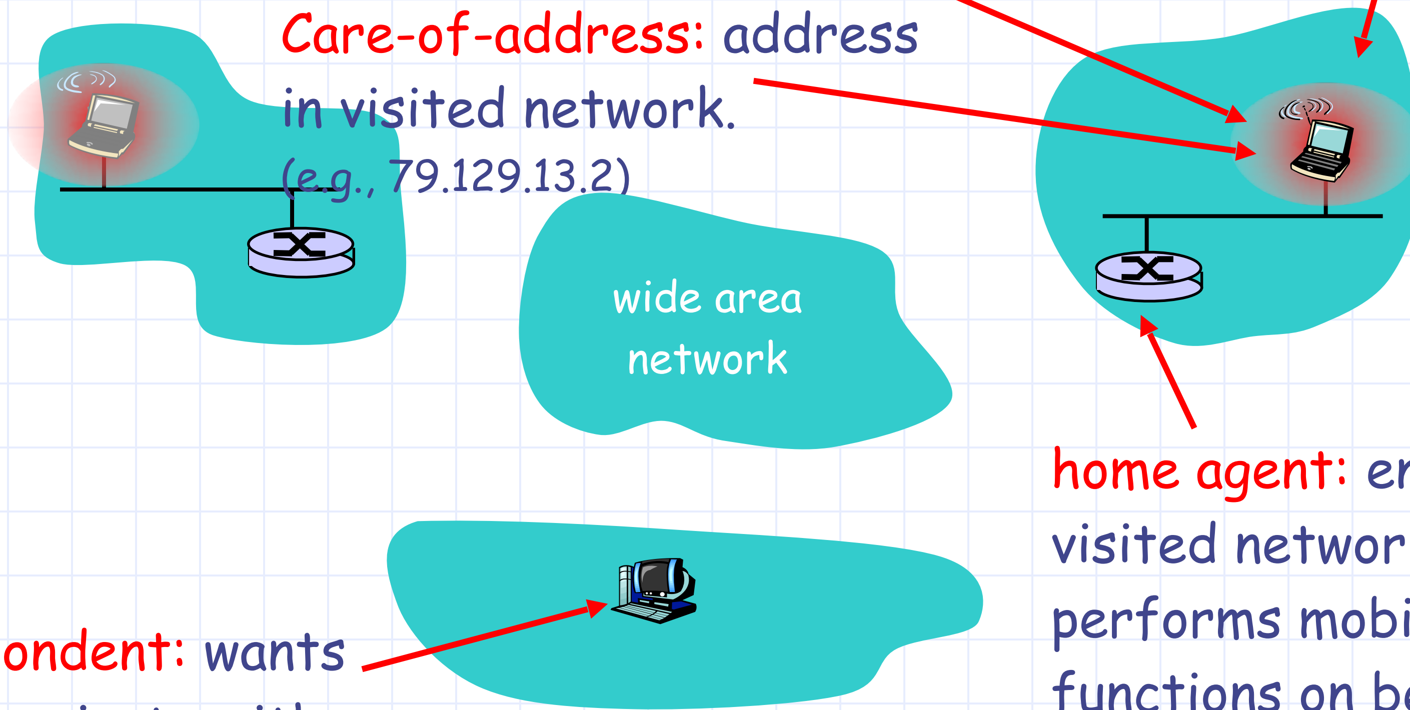
Permanent address: address in home network, can always be used to reach mobile e.g., 128.119.40.186

Mobility: more vocabulary

Permanent address: remains constant (e.g., 128.119.40.186)

visited network: network in which mobile currently resides (e.g., 79.129.13/24)

Care-of-address: address in visited network. (e.g., 79.129.13.2)



home agent: entity in visited network that performs mobility functions on behalf of mobile.

correspondent: wants to communicate with mobile

How do you contact a mobile friend:

Consider friend frequently changing addresses, how do you find her?

- search all phone books?
- call her parents?
- expect her to let you know where he/she is?

Mobility: approaches

- Let routing handle it: routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- Let end-systems handle it:
  - indirect routing: communication from correspondent to mobile goes through home agent, then forwarded to remote
  - direct routing: correspondent gets foreign address of mobile, sends directly to mobile

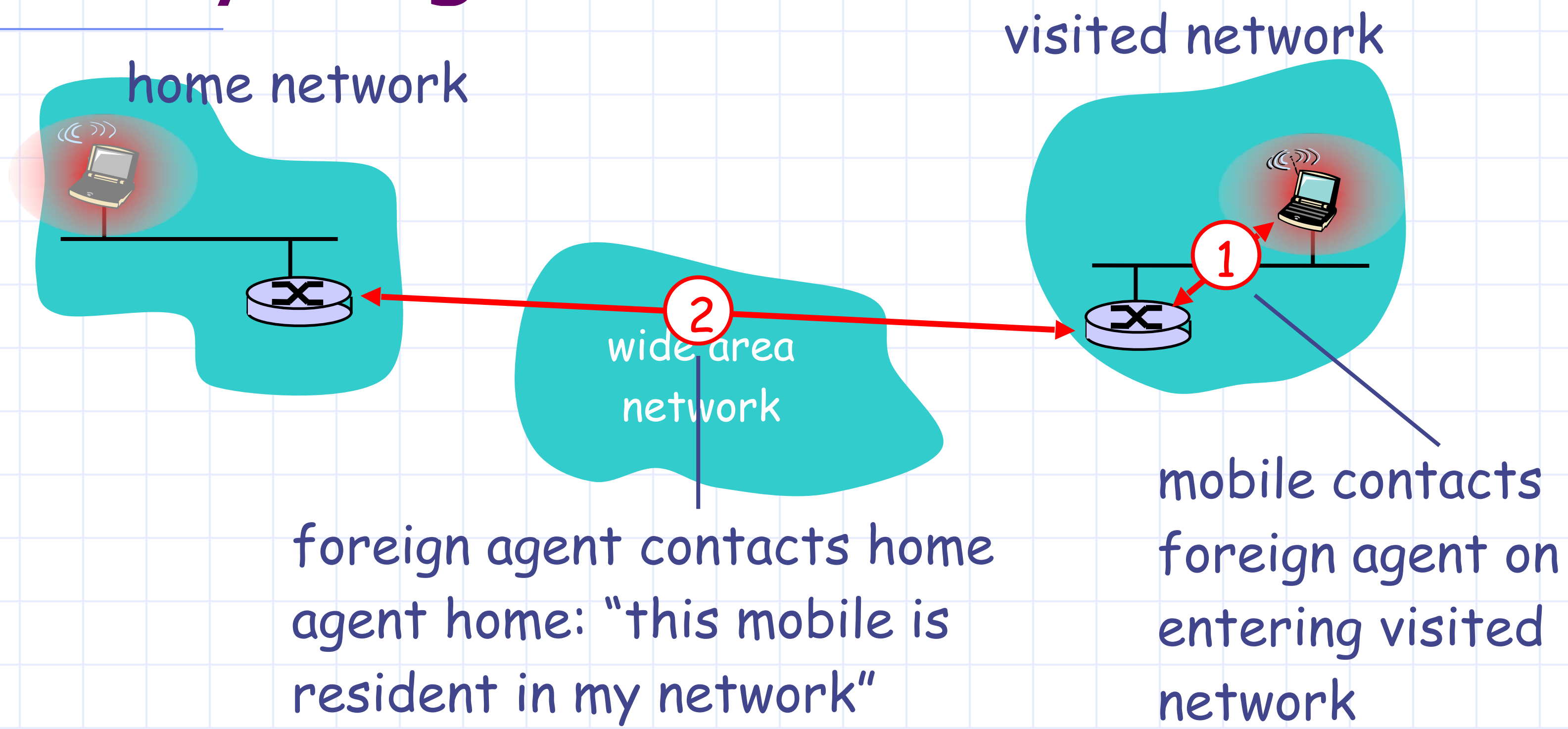
Mobility: approaches

- Let routing handle it… ters advertise permanent address of mobile… sidence via usual routing table exchange.
  - routing tables… e each mobile located
  - no changes to e… ms

not scalable to millions of mobiles

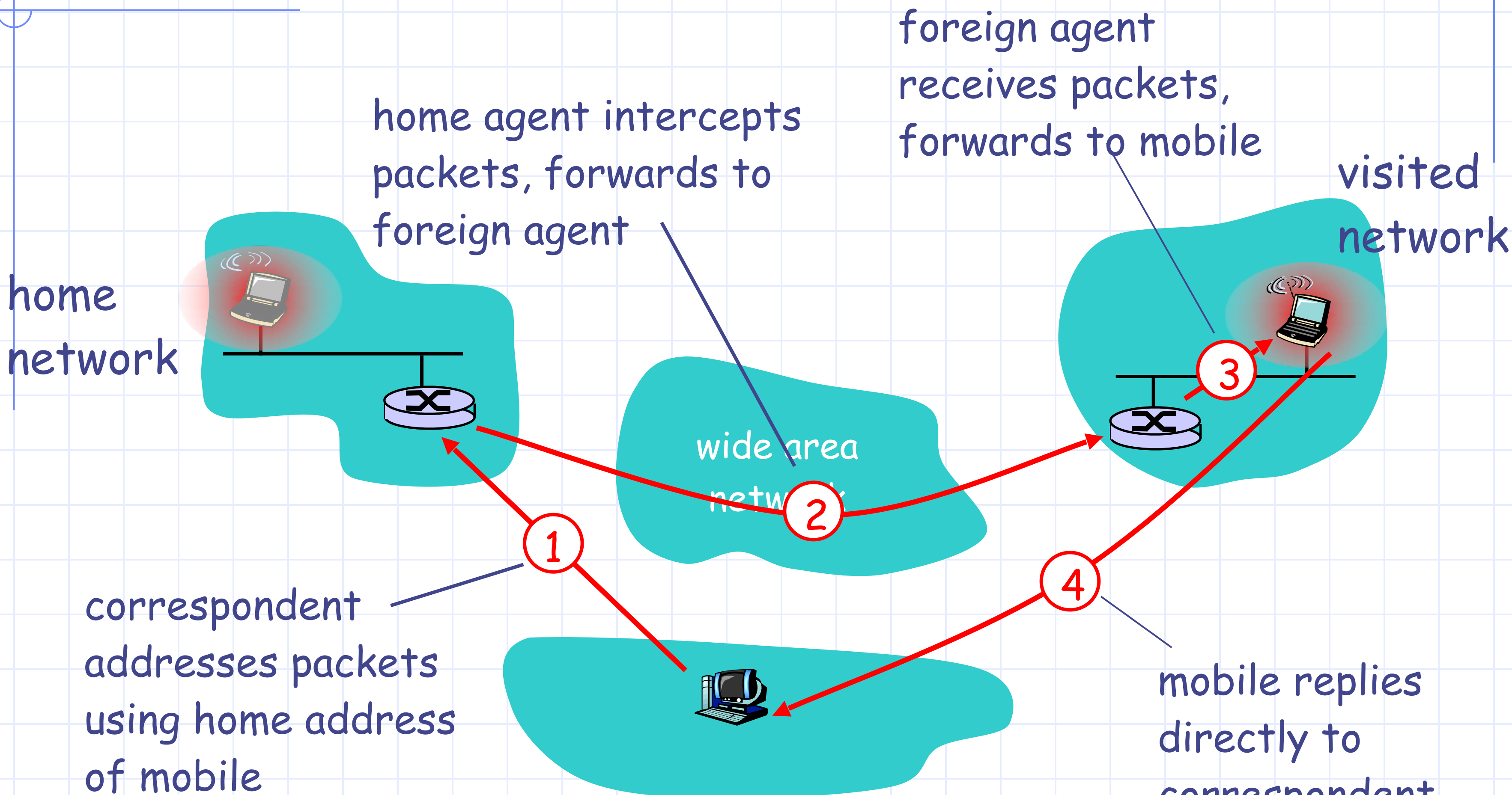
- let end-systems handle it:
  - indirect routing: communication from correspondent to mobile goes through home agent, then forwarded to remote
  - direct routing: correspondent gets foreign address of mobile, sends directly to mobile

Mobility: registration



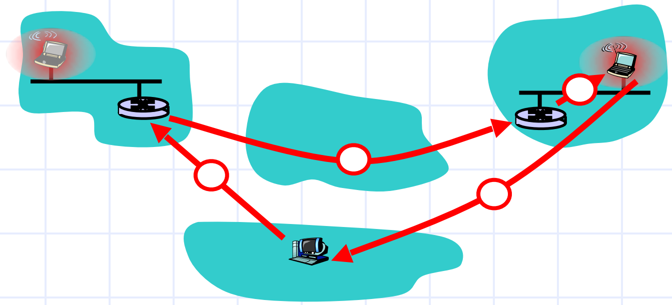
End result:
- Foreign agent knows about mobile
- Home agent knows location of mobile

Mobility via Indirect Routing

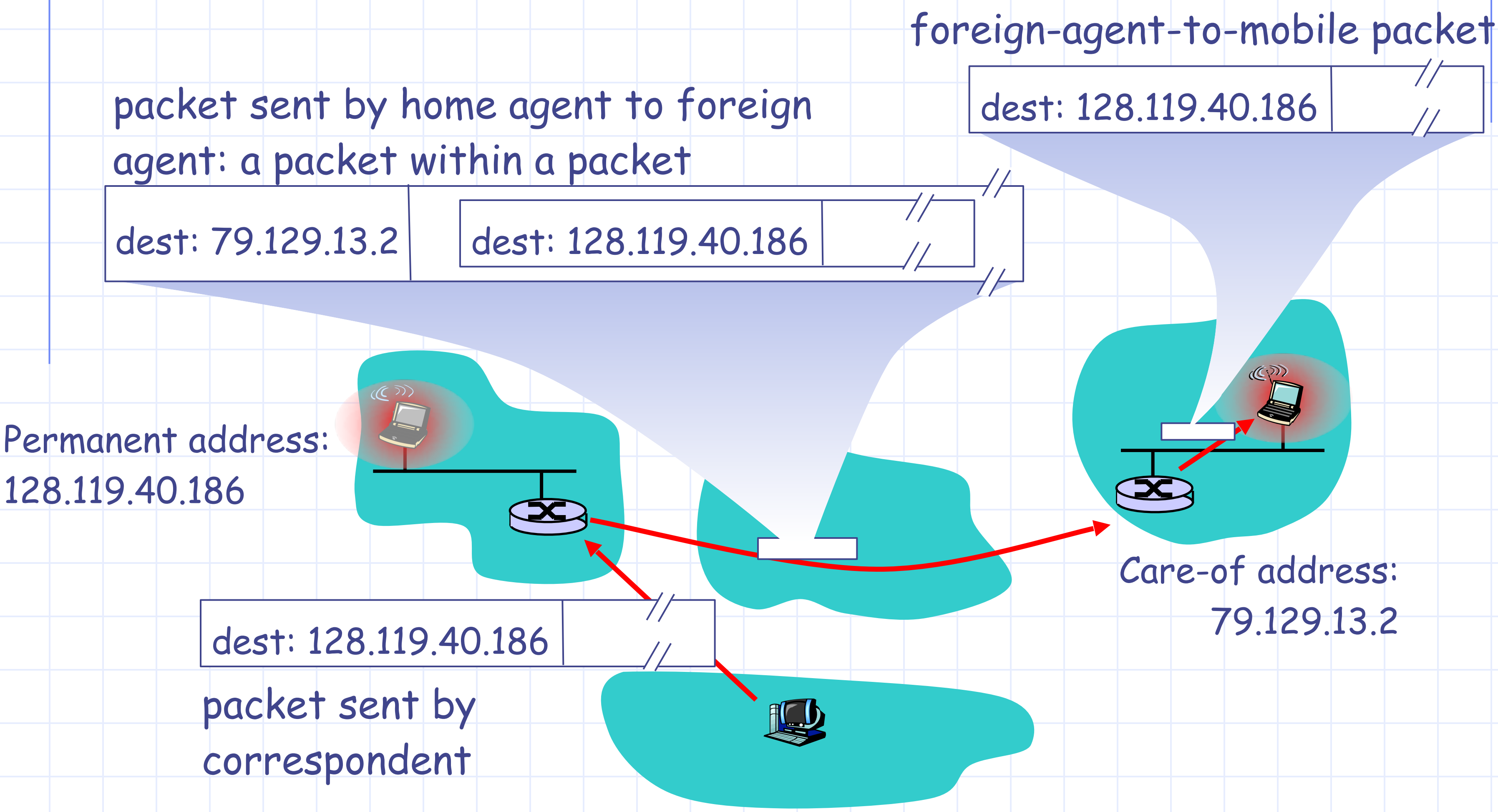


Indirect Routing: comments

- Mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is transparent to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network



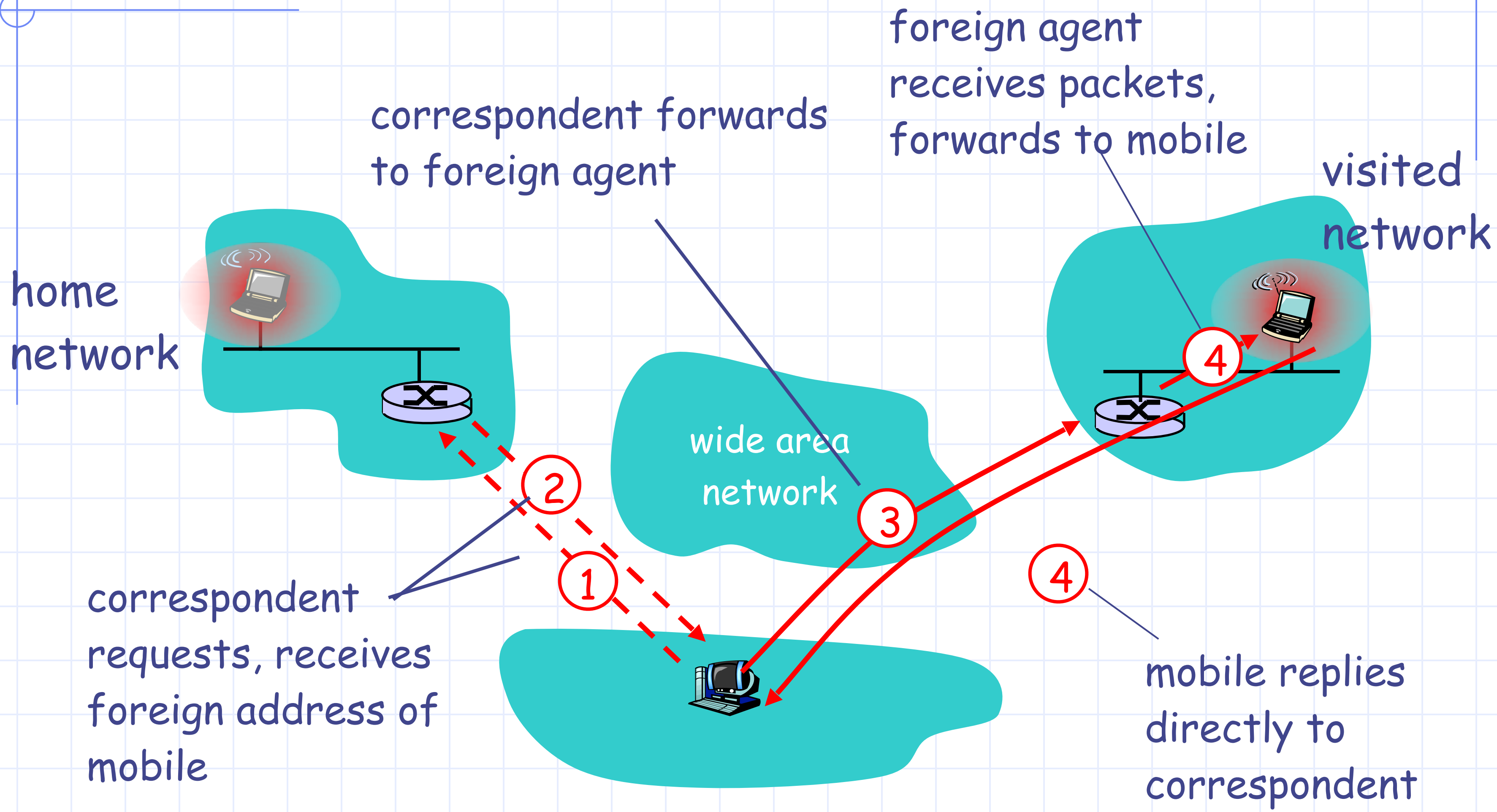
Forwarding datagrams to remote mobile



Indirect Routing: moving between networks

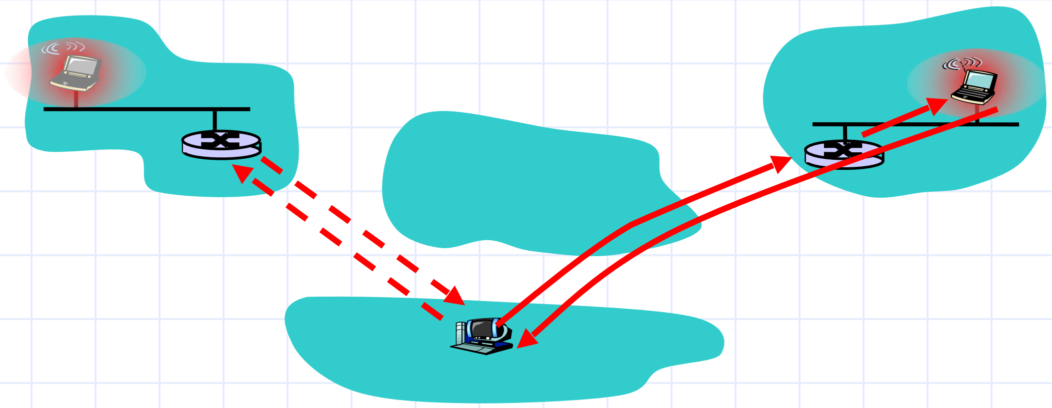
- suppose mobile user moves to another network
  - registers with new foreign agent
  - new foreign agent registers with home agent
  - home agent update care-of-address for mobile
  - packets continue to be forwarded to mobile (but with new care-of-address)
- Mobility, changing foreign networks transparent: on going connections can be maintained!

Mobility via Direct Routing



Mobility via Direct Routing: comments

- overcome triangle routing problem
- non-transparent to correspondent: correspondent must get care-of-address from home agent
  - What happens if mobile changes networks?

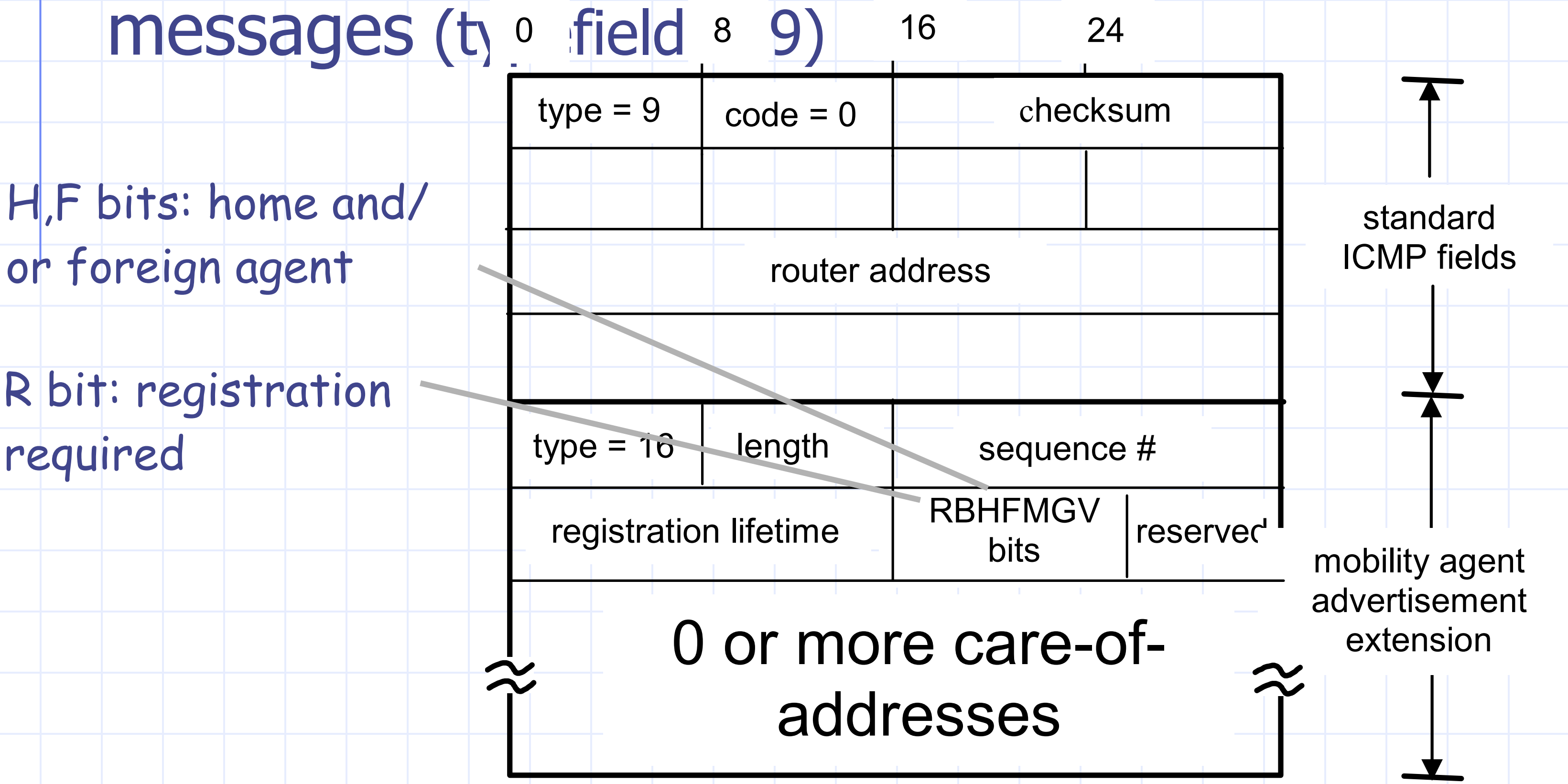


Mobile IP

- RFC 3220
- has many features we’ve seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - agent discovery
  - registration with home agent
  - indirect routing of datagrams

Mobile IP: agent discovery

- agent advertisement: foreign/home agents advertise service by broadcasting ICMP messages (type field = 9)



Mobile IP: registration example