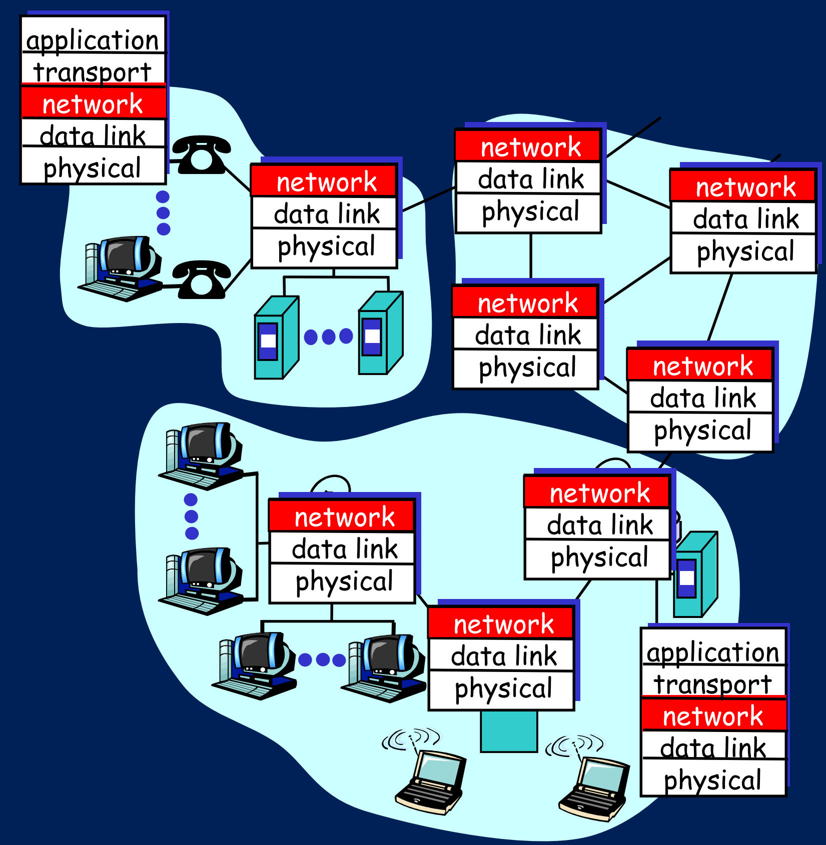
ECSE-4730: Computer Communication Networks (CCN)

Network Layer (Routing)

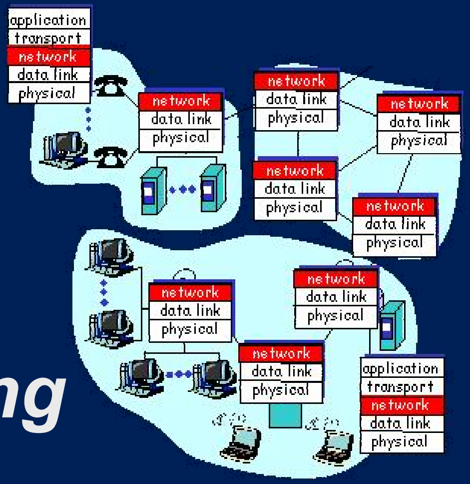
Shivkumar Kalyanaraman: shivkuma@ecse.rpi.edu
Biplab Sikdar: sikdab@rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Network layer functions - 1

- transport packet from sending to receiving hosts
- network layer protocols in *every* host, router



Network layer functions - 2



<u>three important functions:</u>

- ***<u>path determination:</u>*** route taken by packets from source to dest. *Routing algorithms*
- ***<u>Switching (forwarding)</u>:*** move packets from router's input to appropriate router output
- ***<u>call setup</u>:*** (optional) some network architectures require router call setup along path before data flows

Network service model

service abstraction

Q: What *service model* for "channel" transporting packets from sender to receiver?

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

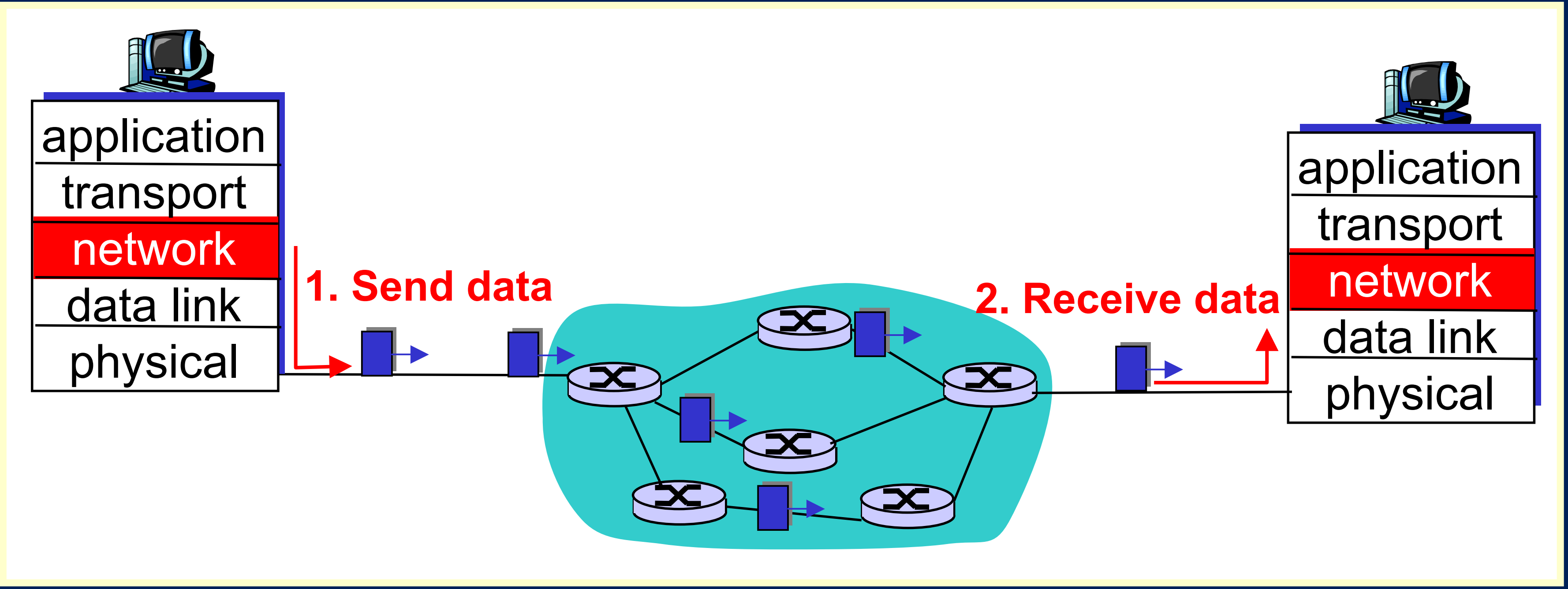
The most important abstraction provided by network layer:

virtual circuit
or
datagram?

Datagram networks: the Internet model - 1

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths

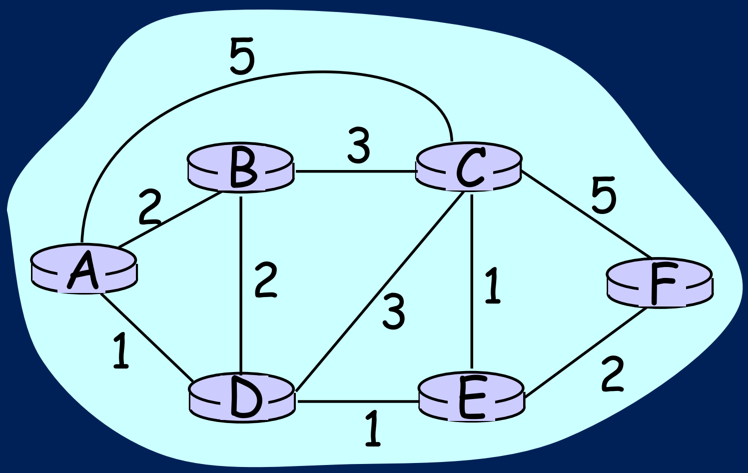
Datagram networks: the Internet model - 2



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 - 1

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 partial info with neighbors
- "**distance vector**" algorithms

Routing Algorithm classification - 2

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

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

A Link-State Routing Algorithm - 1

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

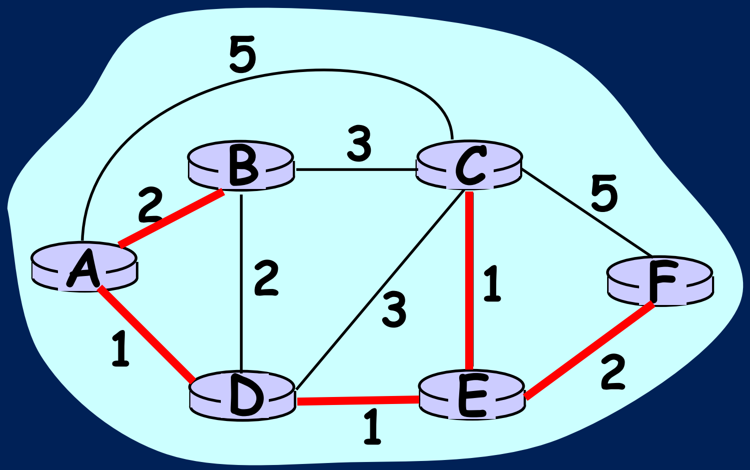
A Link-State Routing Algorithm - 2

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 (neighbor of v) along path from source to v
- **N:** set of nodes whose least cost path definitively known

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					



Dijsktra's Algorithm

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

 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: 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 - 1

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 Vector Routing Algorithm - 2

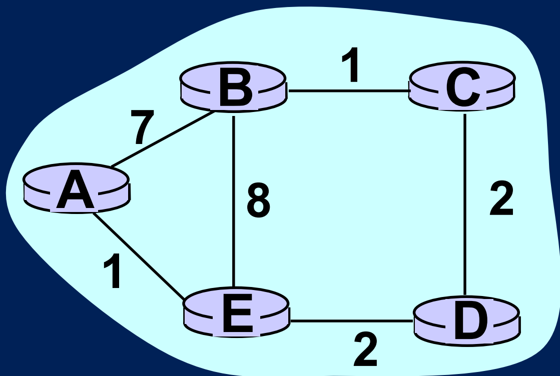
Distance Table data structure

- 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



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

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

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

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

Distance table gives routing table

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

destination	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,4

Distance table → Routing table

Distance Vector Routing: overview - 1

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

Distance Vector Routing: overview - 2

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 - 1

<u>At all nodes, X</u>:

Initialization:

for all adjacent nodes v:

$D^X(*,v)$ = infty /* the * operator means "for all rows" */

$D^X(v,v) = c(X,v)$

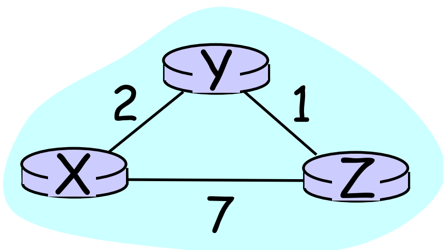
for all destinations, y

send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Distance Vector Algorithm - 2

```
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:  D^X(y,V) =  D^X(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: D^X(Y,V) = c(X,V) + newval

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

forever
```

Distance Vector Algorithm: example - 1



D^X — cost via

dest	Y	Z
Y	2	∞
Z	∞	7

D^Y — cost via

dest	X	Z
X	2	∞
Z	∞	1

D^Z — cost via

dest	X	Y
X	7	∞
Y	∞	1

D^X — cost via

dest	Y	Z
Y	2	8
Z	3	7

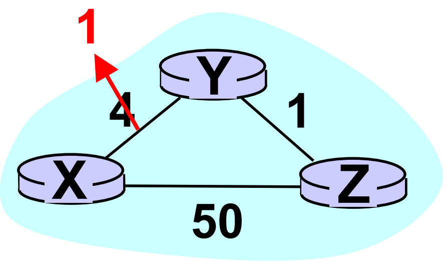
$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

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

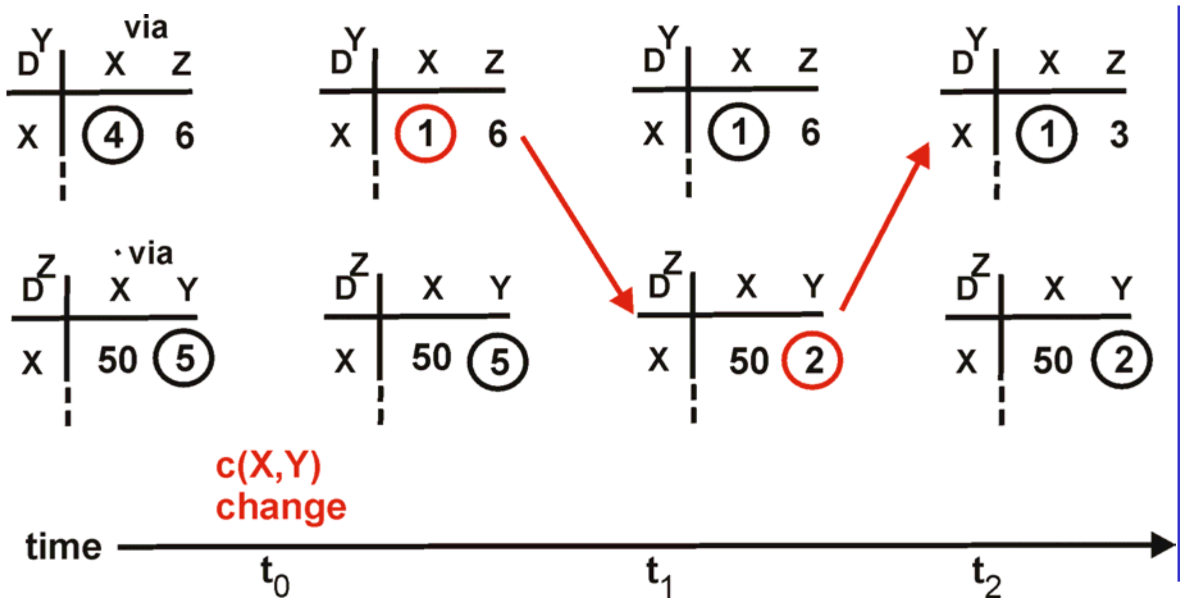
Distance Vector: link cost changes - 1

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"



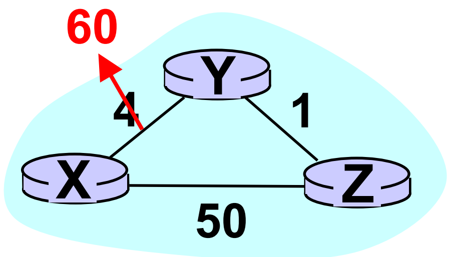
algorithm terminates

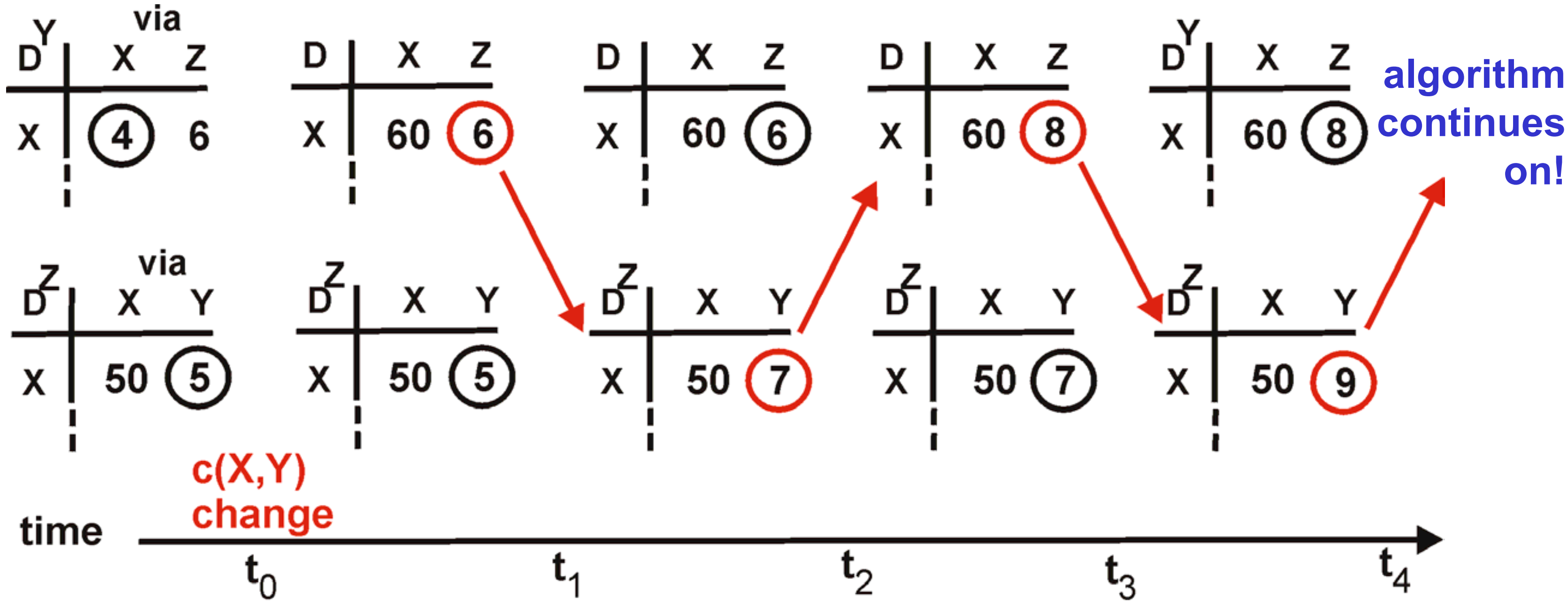
Distance Vector: link cost changes - 2

Link cost changes:

good news travels fast

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



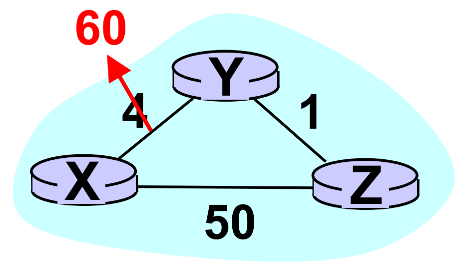


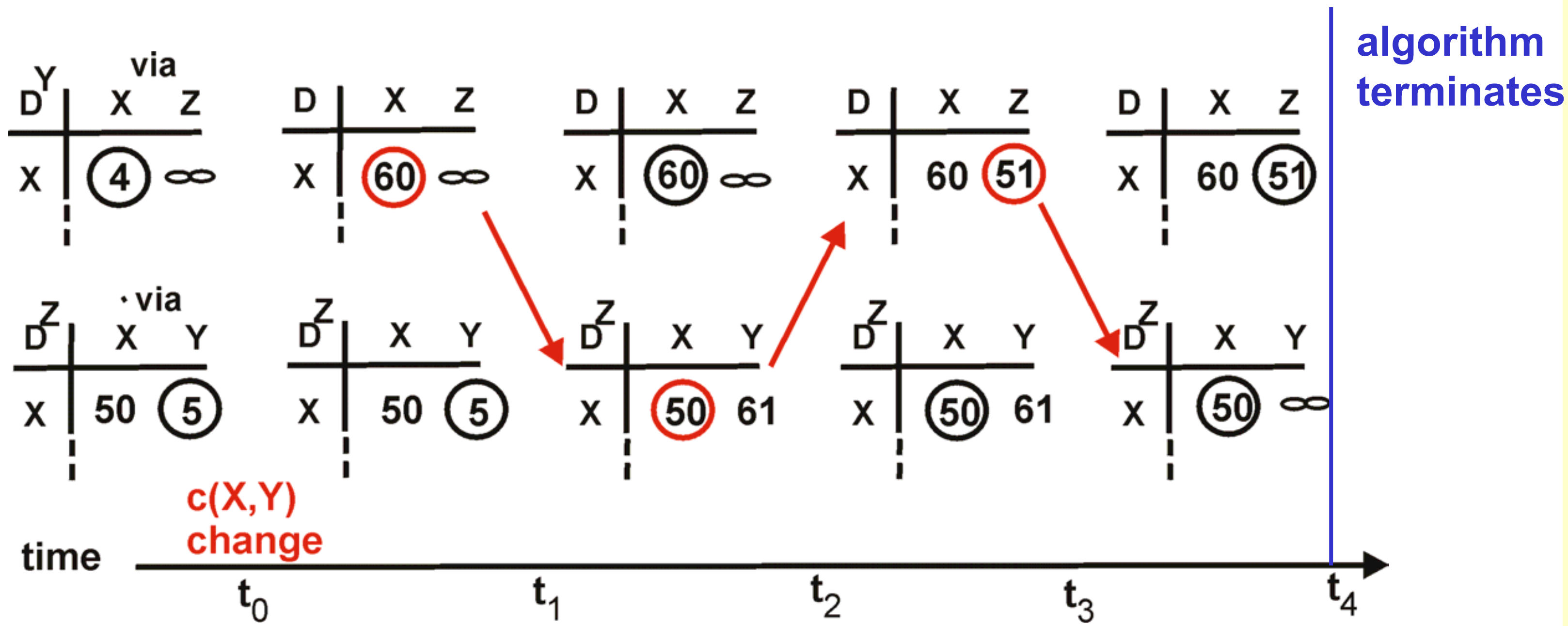
Distance Vector: poisoned reverse

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)

will this completely solve count to infinity problem?





Comparison of LS and DV algorithms - 1

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

Comparison of LS and DV algorithms - 1

Robustness: what happens if router malfunctions?

LS:

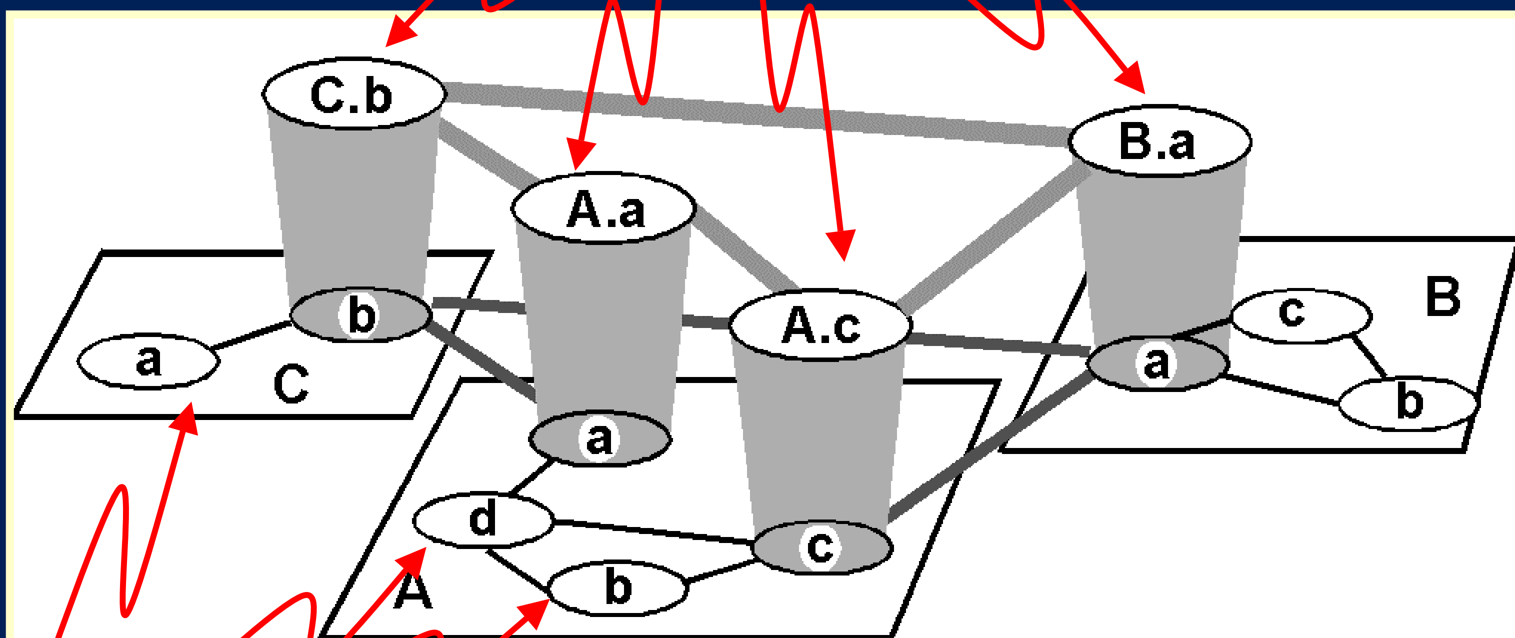
- node can advertise incorrect ***link*** cost
- each node computes only its ***own*** table

DV:

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

Internet AS Hierarchy

Intra-AS border (exterior gateway) routers



Inter-AS interior (gateway) routers

Intra-AS Routing

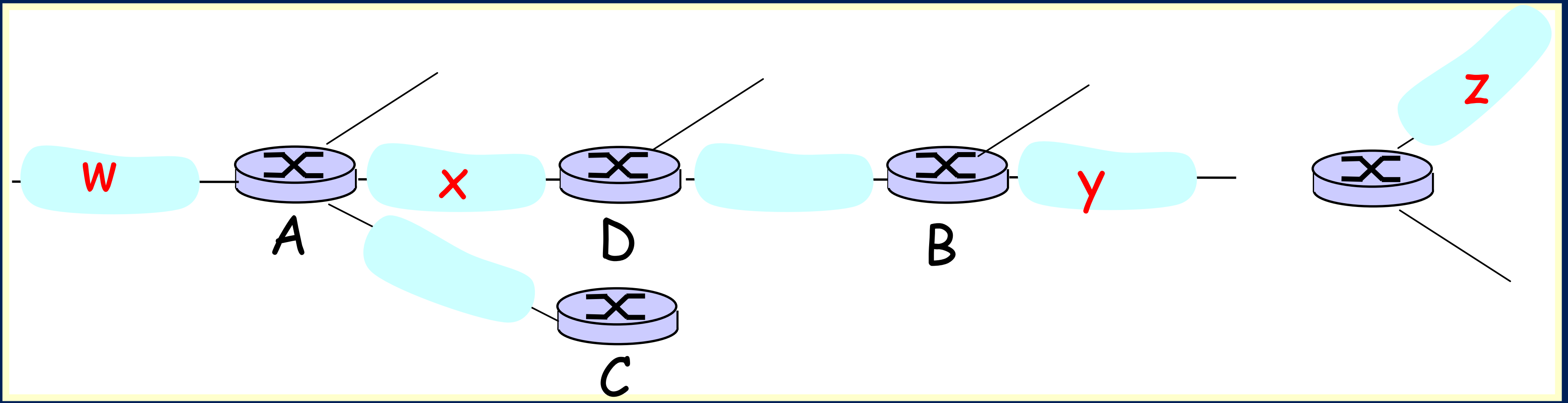
- Also known as Interior Gateway Protocols (IGP)
- Most common IGPs:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco propr.)

RIP
(Routing Information Protocol) - 1

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)

- Distance vectors: exchanged every 30 sec via Response Message (also called advertisement)
- Each advertisement: route to up to 25 destination nets

RIP
(Routing Information Protocol) - 2



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

Routing table in D

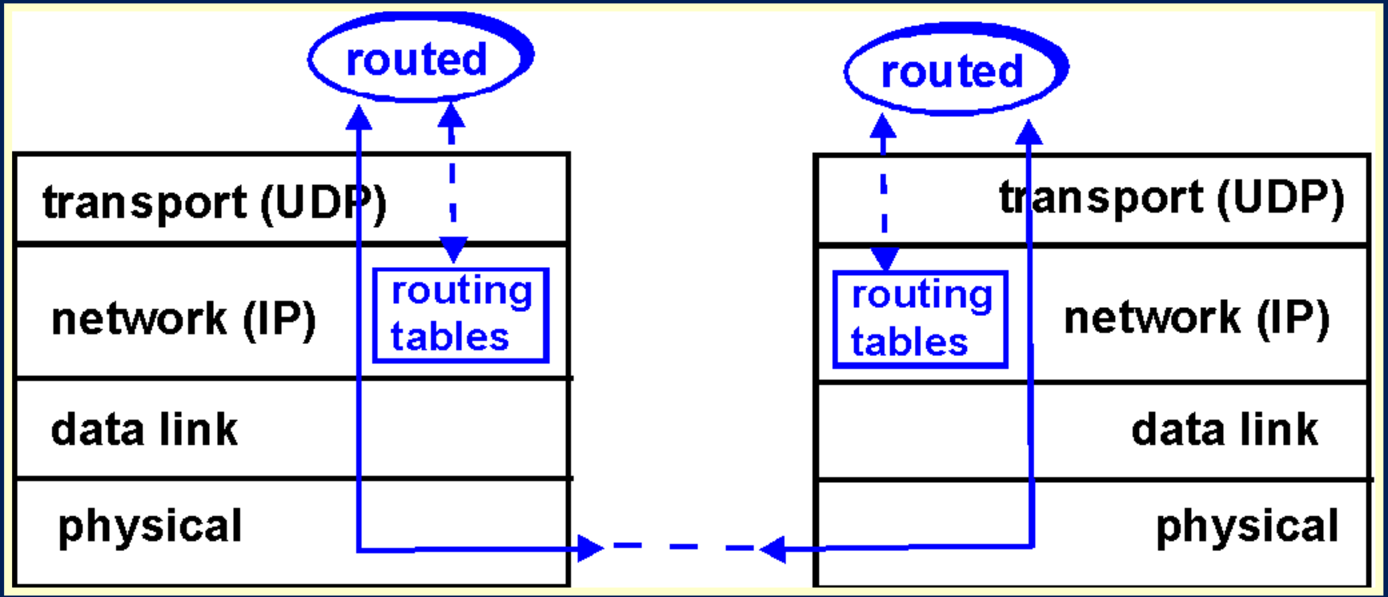
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing - 1

- **RIP routing tables managed by** application-level **process called route-d (daemon)**
- **advertisements sent in UDP packets, periodically repeated**



RIP Table processing - 2

Router: *giroflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

Three attached class C networks (LANs)
Router only knows routes to attached LANs
Default router used to "go up"
Route multicast address: 224.0.0.0
Loopback interface (for debugging)

OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)