Università
degli Studi
della Campania
Luigi Vanvitelli

Reti di Calcolatori e Cybersecurity

Routing Distance Vector

Ing. Vincenzo Abate

Algoritmo Distance Vector

Ogni router mantiene una tabella di tutti gli instradamenti noti

- inizialmente, solo le reti a cui è connesso direttamente

Ogni entry della tabella indica:

- una rete raggiungibile
- il next hop
- il numero di hop necessari per raggiungere la destinazione

Periodicamente, ogni router invia a tutti i vicini (due router sono vicini se sono collegati alla stessa rete fisica):

- un messaggio di aggiornamento contenente tutte le informazioni della propria tabella (vettore delle distanze – **distance vector**)

I router che ricevono tale messaggio aggiornano la tabella nel seguente modo:

- eventuale modifica di informazioni relative a cammini già noti
- eventuale aggiunta di nuovi cammini
- eventuale eliminazione di cammini non più disponibili

Algoritmo Distance Vector

Destin.	Dist.	Route
net 1	0	direct
net 2	0	direct
net 4	8	router L
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	2	router A

Tabella del router B

Costo come hop.

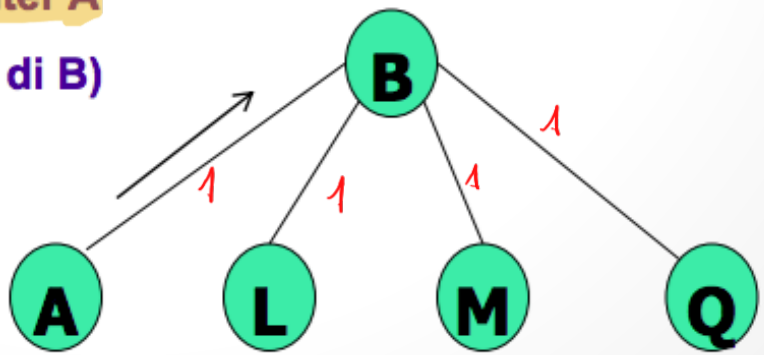
Destin.	Dist.
net 1	2
net 4	3
net 17	6
net 21	4
net 24	5
net 30	10
net 42	3

Messaggio di aggiornamento del router A

(vicino di B)

Destin.	Dist.	Route
net 1	0	direct
net 2	0	direct
net 4	4	router A
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	4	router A
net 21	5	router A

Tabella aggiornata del router B



Algoritmo Distance Vector

Il calcolo consiste nella fusione di tutti i distance vector delle linee attive

Un router ricalcola le sue tabelle se:

- cade una linea attiva
- riceve un distance vector, da un nodo adiacente, diverso da quello memorizzato

Se le tabelle risultano diverse da quelle precedenti:

- invia ai nodi adiacenti un nuovo distance vector

Algoritmo Distance Vector

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

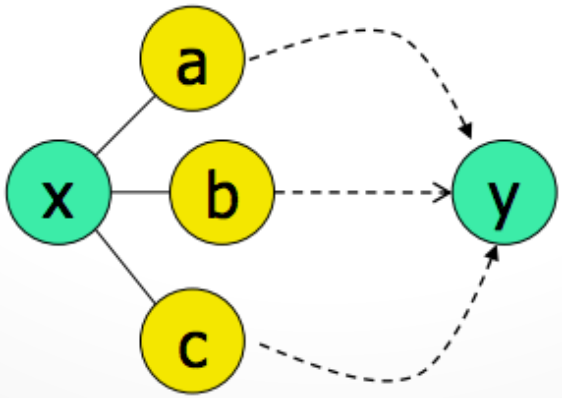
Let $D_x(y)$: cost of least-cost path from x to y.
Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

v's estimated least-cost-path to y

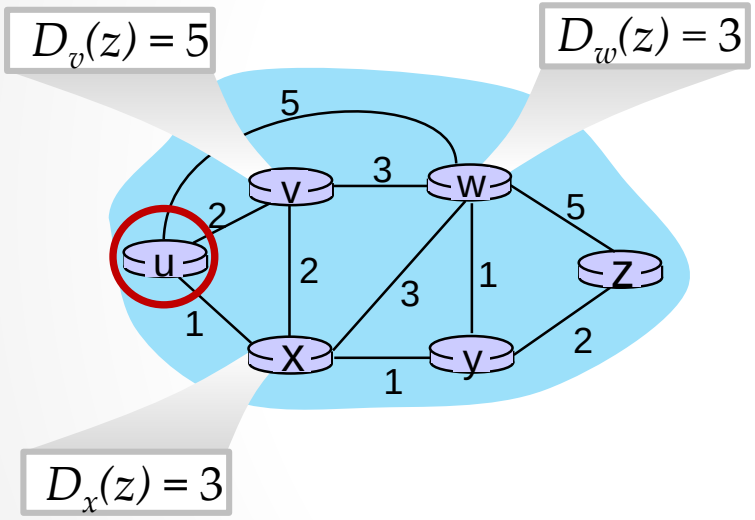
min taken over all neighbors v of x

direct cost of link from x to v



Bellman-Ford Example

Suppose that *u*'s neighboring nodes, *x,v,w*, know that for destination *z*:



Bellman-Ford 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 achieving minimum (x) is next hop on estimated least-cost path to destination (z)

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when *x* receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

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

Distance vector algorithm

Define

- $D_x(y)$ = 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)

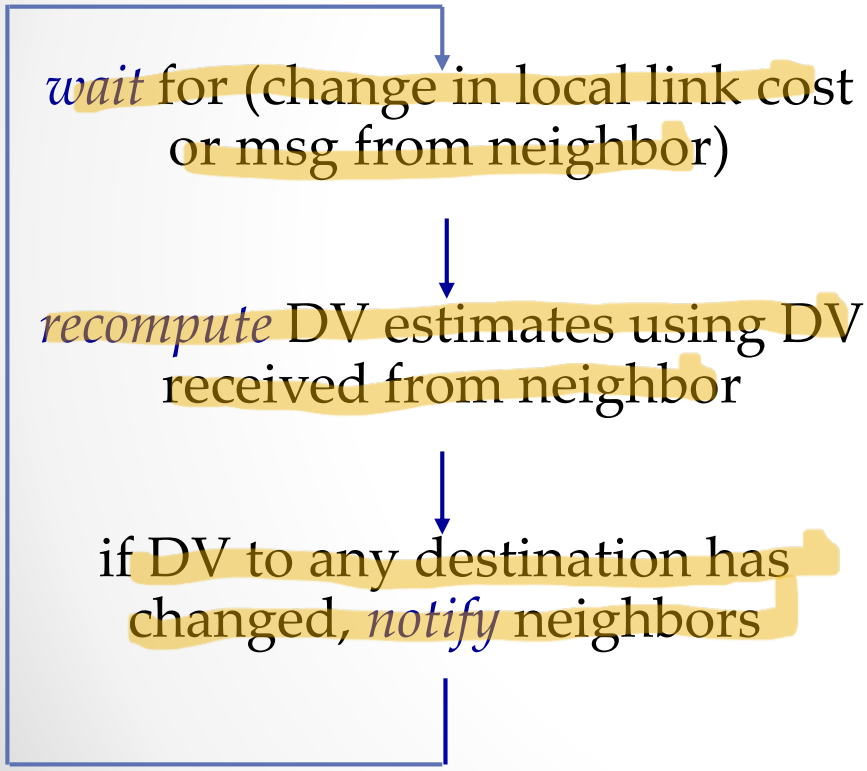
Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$

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:

each node:



iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Algoritmo Distance Vector

Ad ogni nodo, x:

1 *Inizializzazione:*
2 *per tutti i nodi adiacenti v:*
3 $D^X(*,v) = infinito$ *{il simbolo * significa "per ogni riga"}*
4 $D^X(v,v) = c(x,v)$
5 *per tutte le destinazioni, y*
6 *manda* $min_w D(y,w)$ *a ogni vicino*

tutti i nodi che v può raggiungere

ogni nodo adiacente

Algoritmo Distance Vector

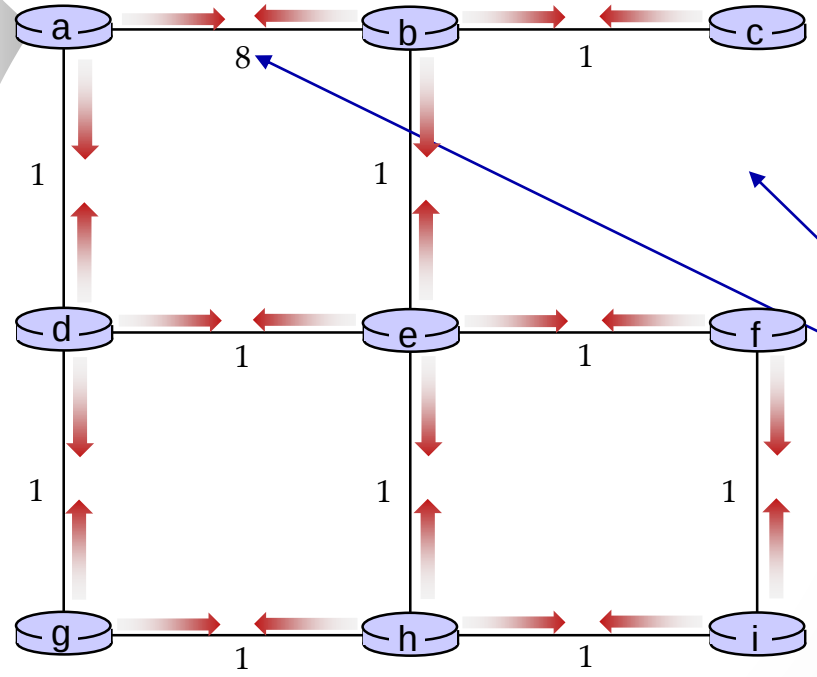
```
8  loop
9    aspetta  (fino a  quando vedo una modifica nel costo di un
10           collegamento  oppure ricevo un messaggio da un vicino v)
11
12   if (c(x,v) cambia di  d)
13     { cambia il costo a tutte le dest. via vicino v di d }
14     { nota: d puo' essere  positivo o  negativo  }
15     per tutte le  destinazioni  y:  D^X(y,v) =  D^X(y,v) + d
16
17   else if (ricevo mess. aggiornamento da v verso destinazione y)
18     { cammino minimo da v a y e' cambiato }
19     { V ha mandato un nuovo valore per il suo  min_W D^V(y,w) }
20     { chiama questo valore  "newval"  }
21     per la singola destinazione y: D^X(y,v) = c(x,v) + newval
22
23   if hai un nuovo  min_W D^X(y,w) per una qualunque destinazione  y
24     manda il nuovo valore di  min_W D^X(y,w) a tutti i vicini
25
26 forever
```

Distance vector: example

t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



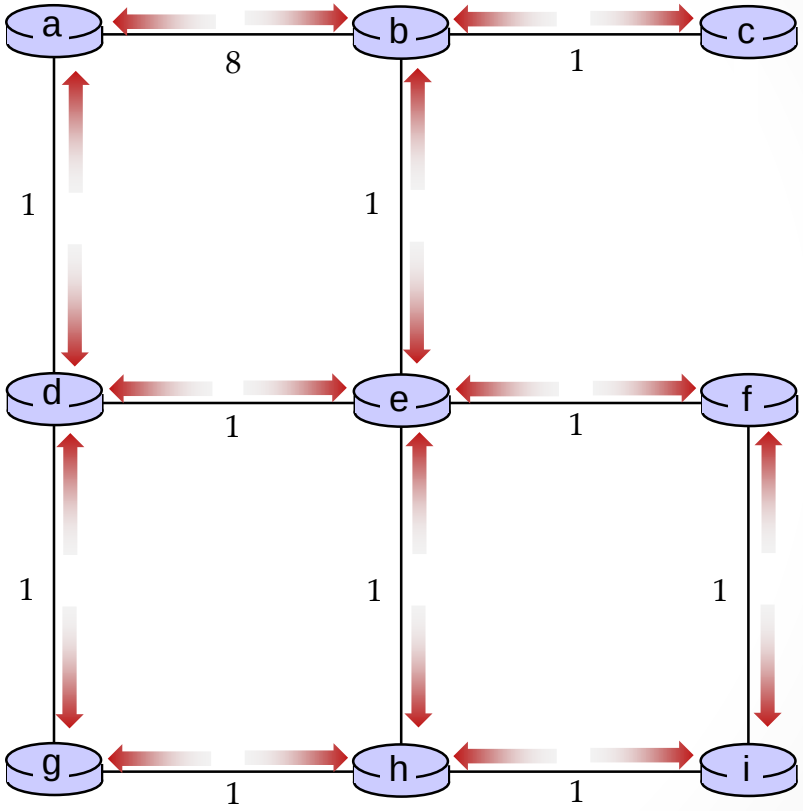
A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

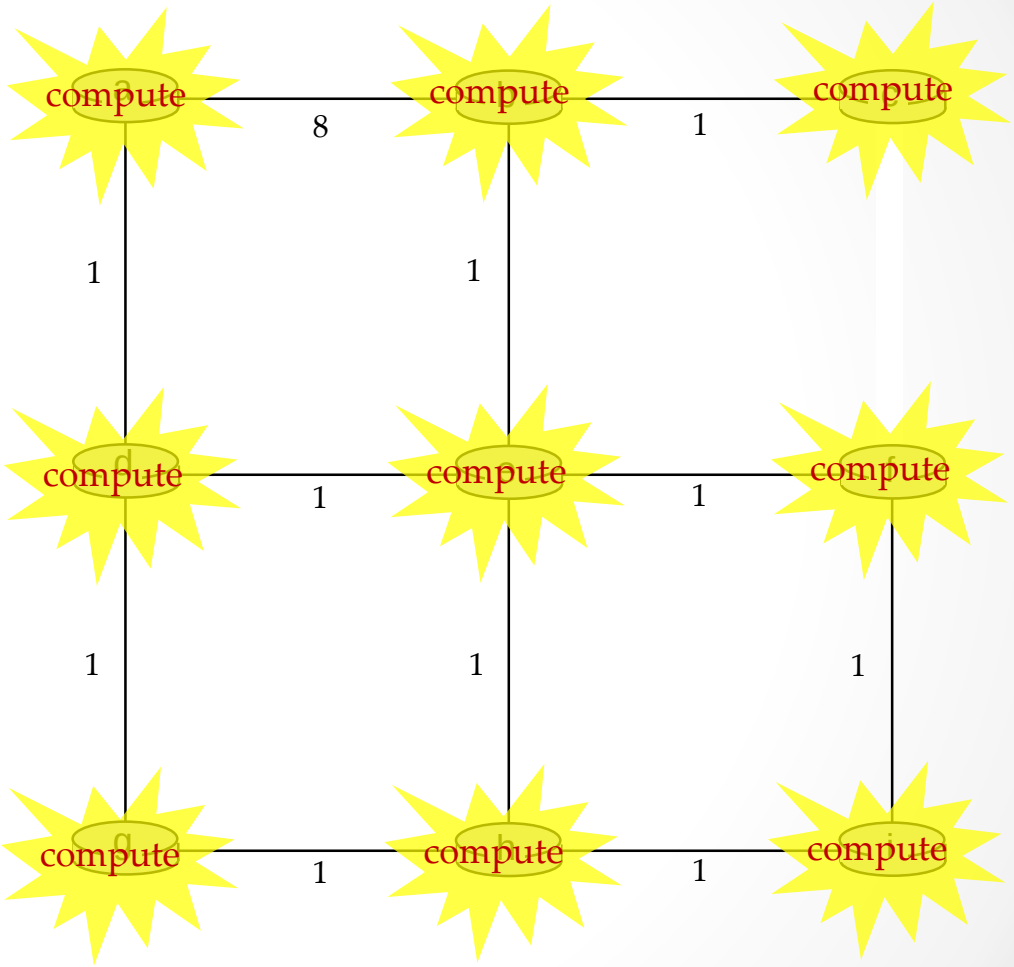


Distance vector example: iteration

t=1

All nodes:

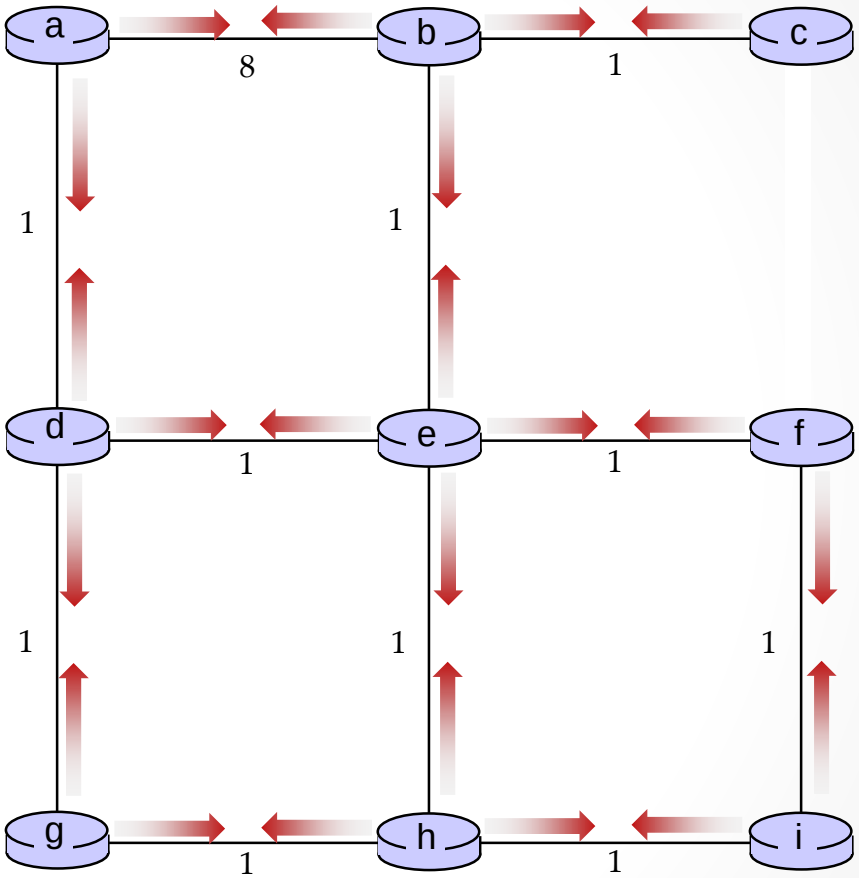
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

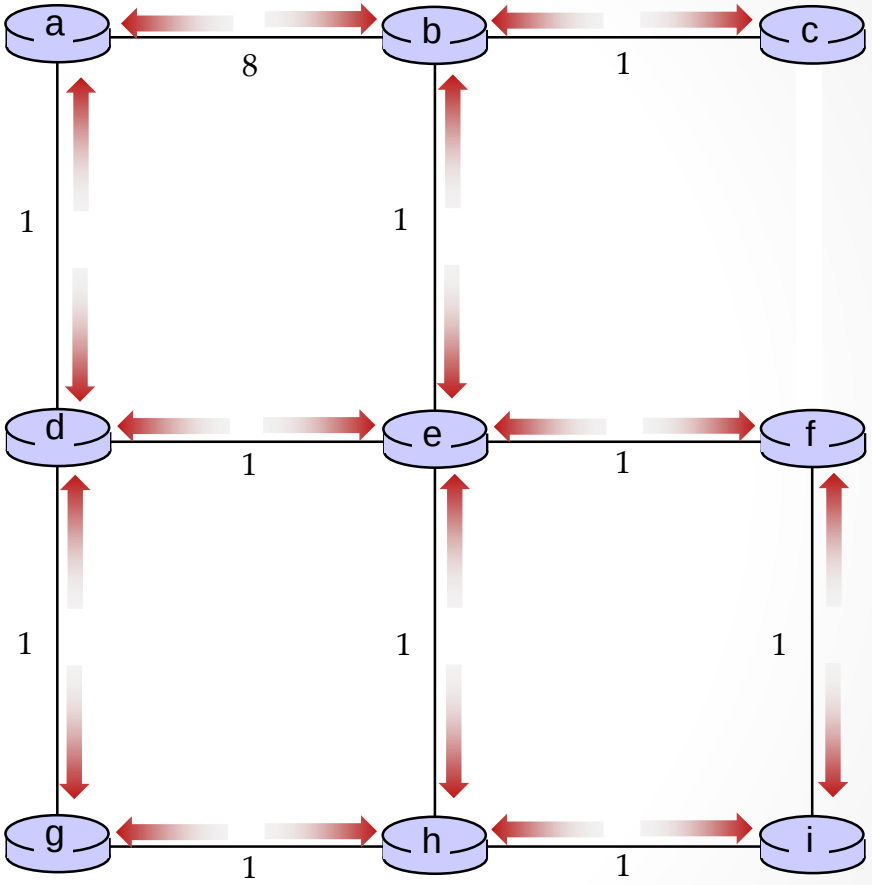


Distance vector example: iteration

t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

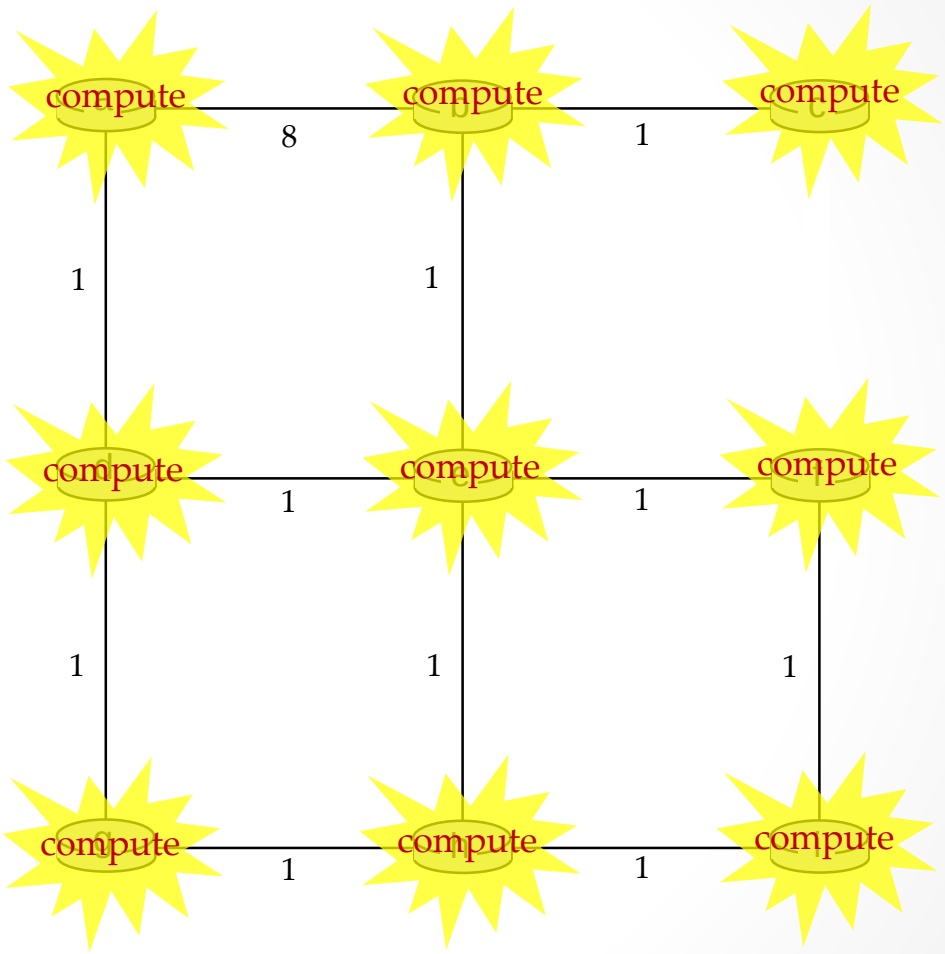


Distance vector example: iteration

t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

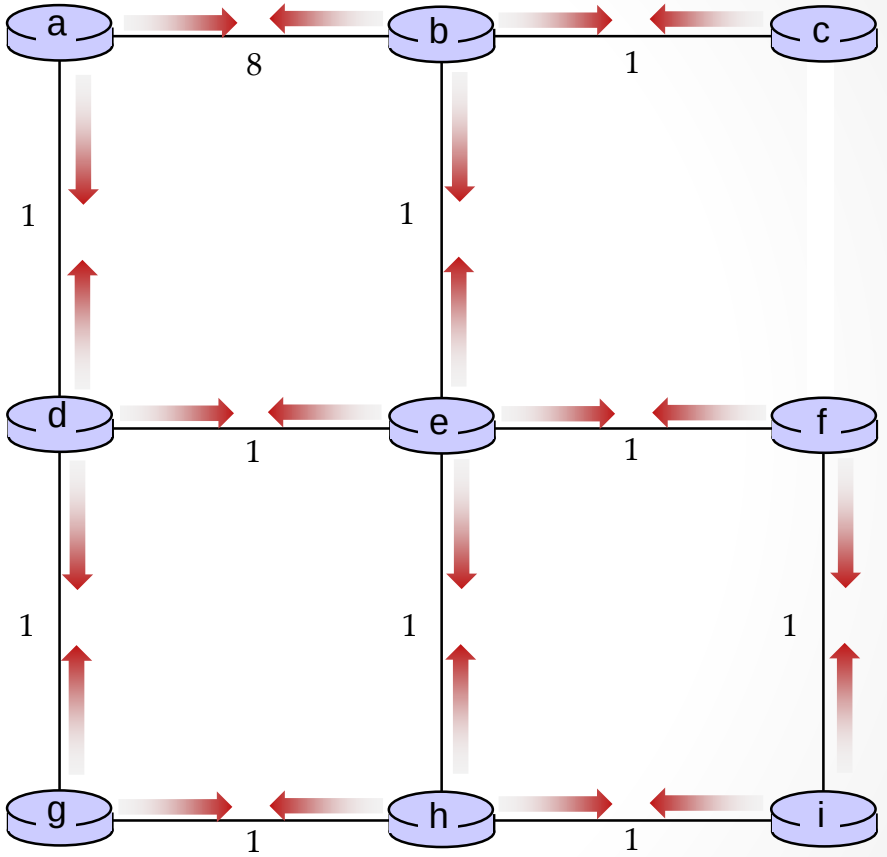


Distance vector example: iteration

t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

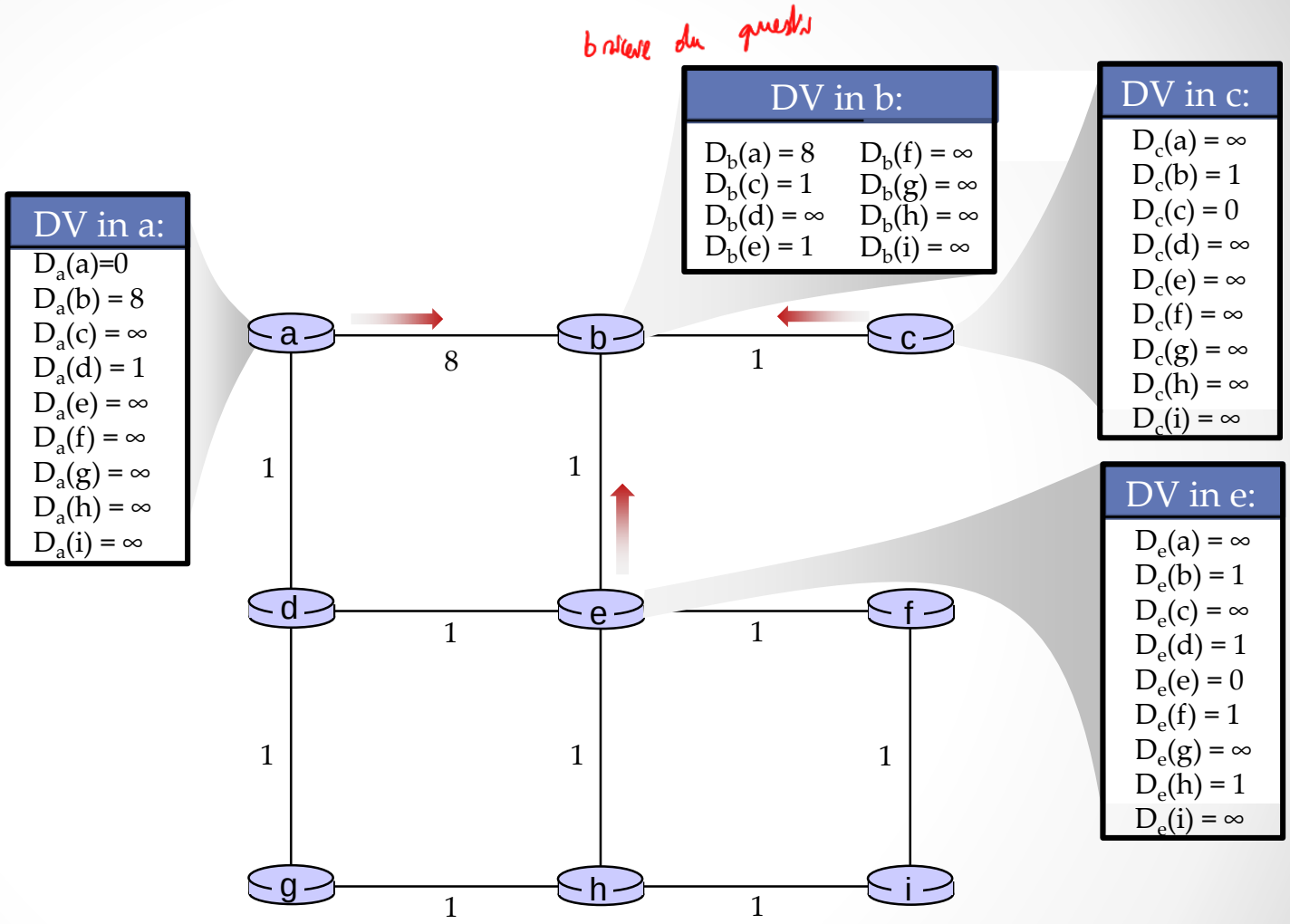
…. and so on

Let's next take a look at the iterative *computations* at nodes

Distance vector example: computation

t=1

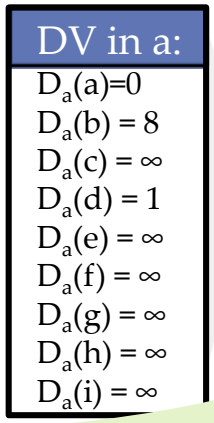
- b receives DVs from a, c, e

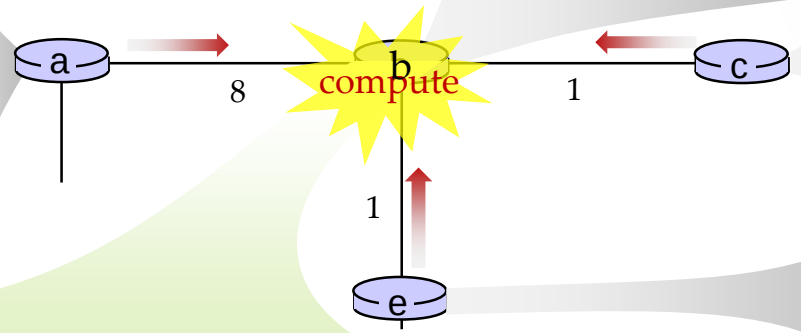


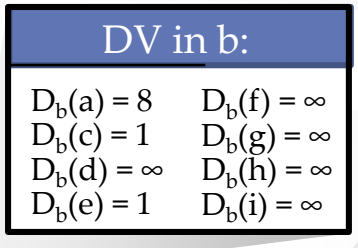
Distance vector example: computation

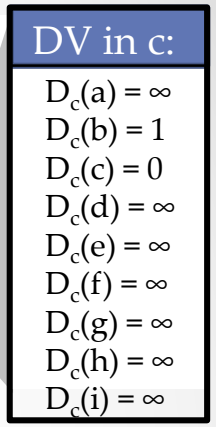
t=1

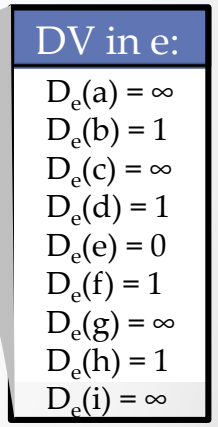
- b receives DVs from a, c, e, computes:











$D_b(a) = \min\{c_{b,a}+D_a(a),\ c_{b,c}+D_c(a),\ c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c),\ c_{b,c}+D_c(c),\ c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d),\ c_{b,c}+D_c(d),\ c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$

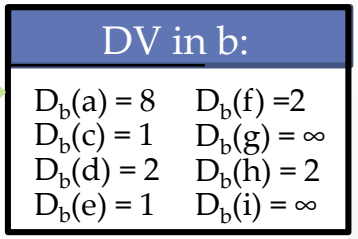
$D_b(e) = \min\{c_{b,a}+D_a(e),\ c_{b,c}+D_c(e),\ c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

$D_b(f) = \min\{c_{b,a}+D_a(f),\ c_{b,c}+D_c(f),\ c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g),\ c_{b,c}+D_c(g),\ c_{b,e}+D_e(g)\} = \min\{\infty,\infty,\infty\} = \infty$

$D_b(h) = \min\{c_{b,a}+D_a(h),\ c_{b,c}+D_c(h),\ c_{b,e}+D_e(h)\} = \min\{\infty,\infty,2\} = 2$

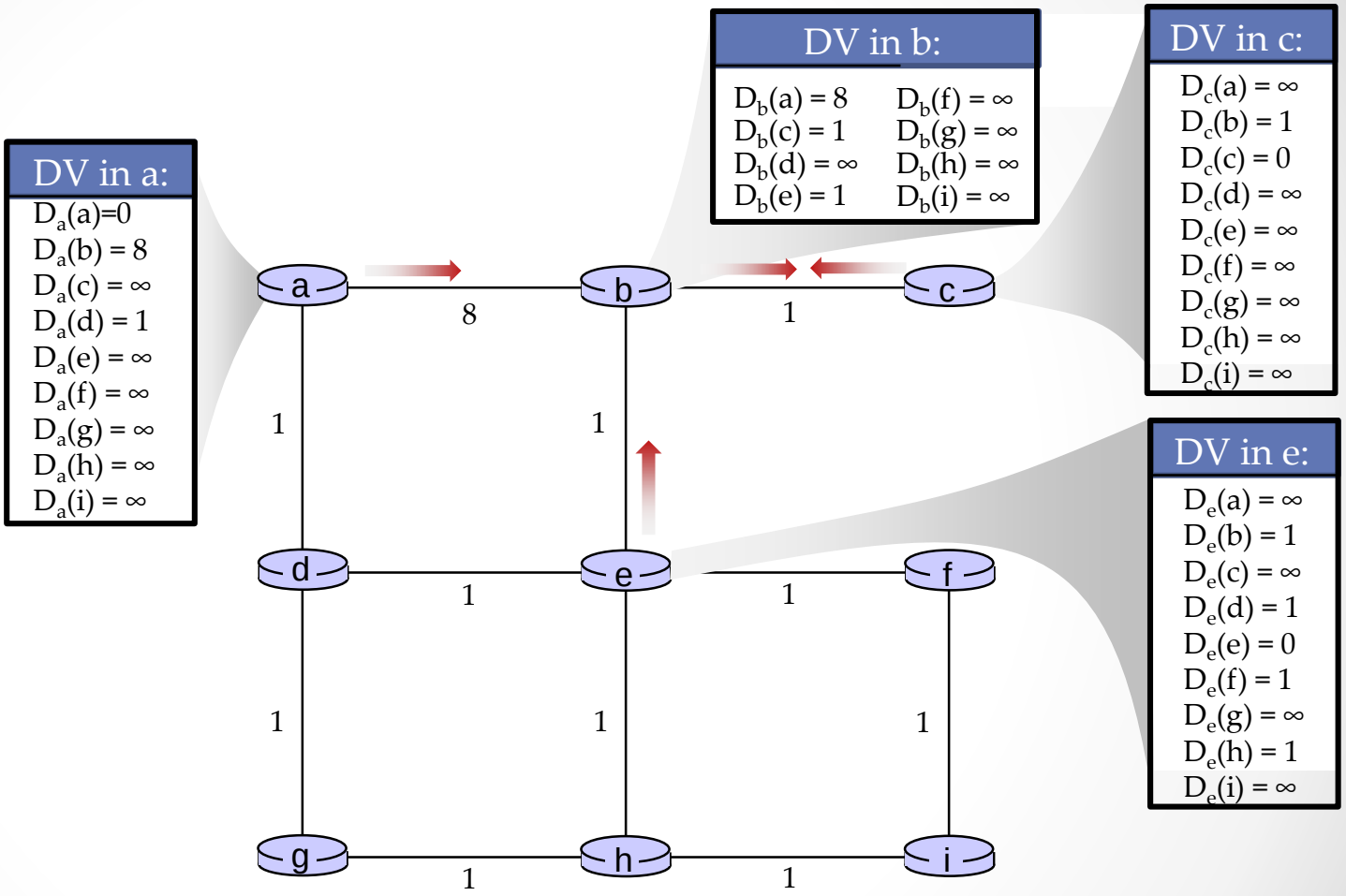
$D_b(i) = \min\{c_{b,a}+D_a(i),\ c_{b,c}+D_c(i),\ c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$



Distance vector example: computation

t=1

- c receives DVs from b

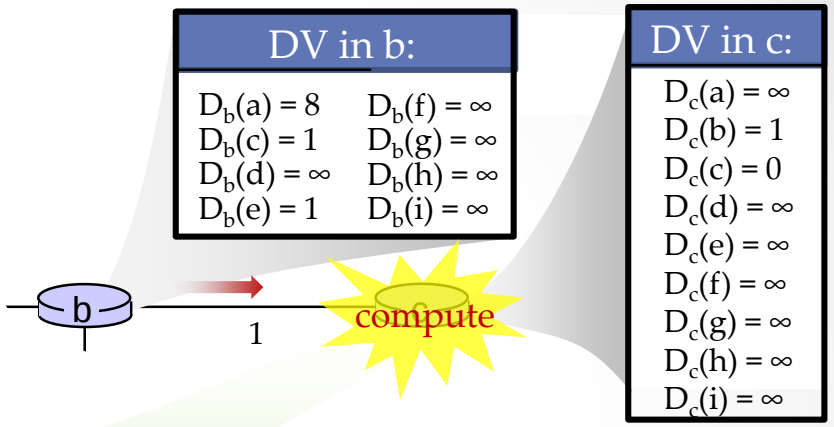


Distance vector example: computation

b manda prima steady aggiornamento

t=1

- c receives DVs from b computes:



$D_c(a) = \min\{c_{c,b}+D_b(a)\} = 1 + 8 = 9$

$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$

$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$

$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$

$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$

$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty = \infty$

$D_c(h) = \min\{c_{bc,b}+D_b(h)\} = 1+ \infty = \infty$

$D_c(i) = \min\{c_{c,b}+D_b(i)\} = 1+ \infty = \infty$

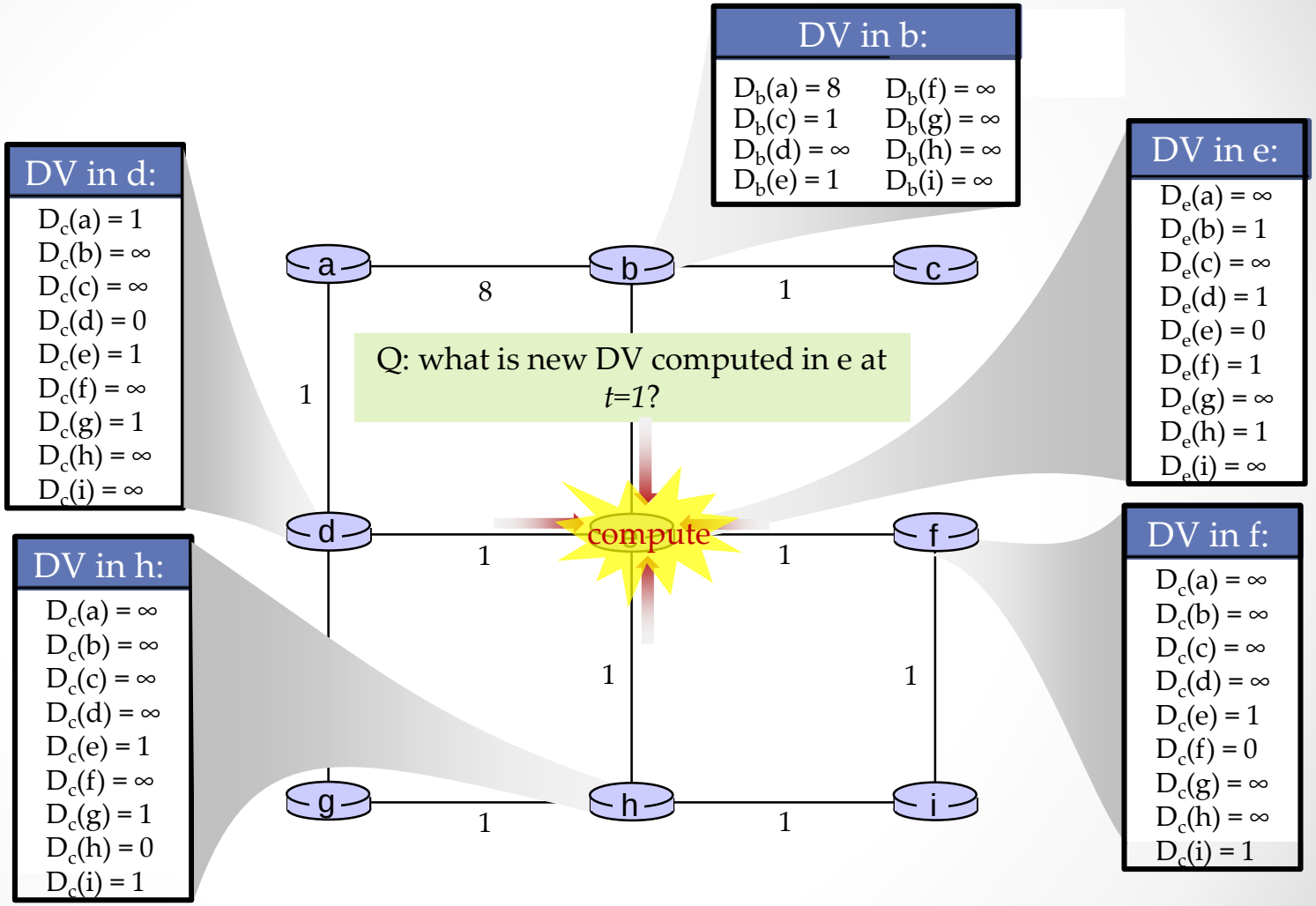
DV in c:

$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

Distance vector example: computation

t=1

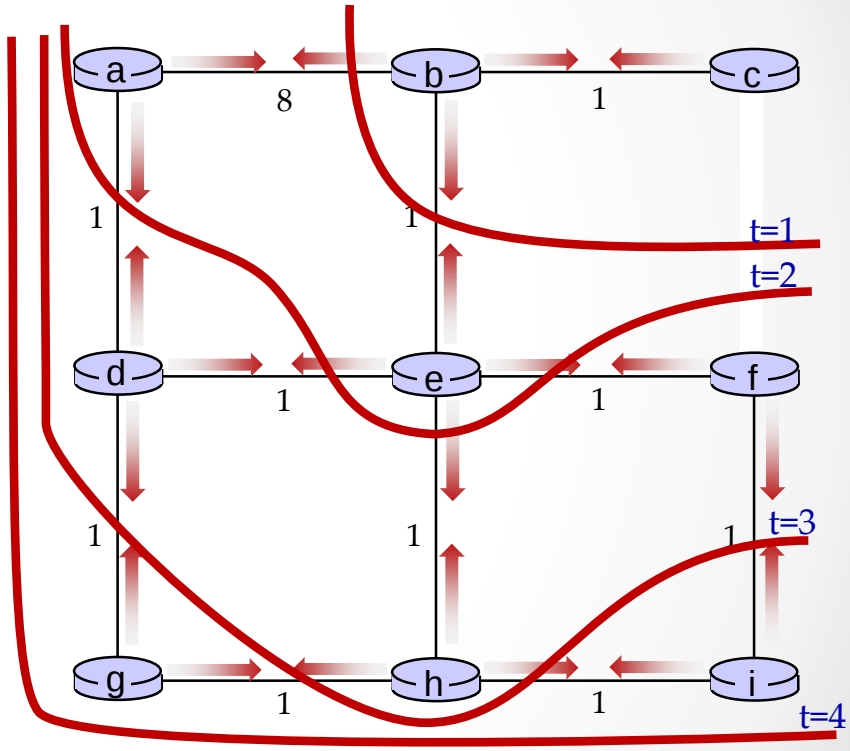
- e receives DVs from b, d, f, h



Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

- t=0 c's state at t=0 is at c only
- t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
- t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
- t=3 c's state at t=0 may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well
- t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Distance Vector: considerazioni

Vantaggi:

- facile da implementare

Svantaggi

- ogni messaggio contiene un'intera tabella di routing
- lenta propagazione delle informazioni sui cammini:
  - converge alla velocità del router più lento
- se lo stato della rete cambia velocemente, le rotte possono risultare inconsistenti
  - possono innescarsi dei loop a causa di particolari variazioni della topologia
- difficile capirne e prevederne il comportamento su reti grandi
  - nessun nodo ha una mappa della rete!

Distance Vector: considerazioni

How fast the routers learn about link-status change in the network

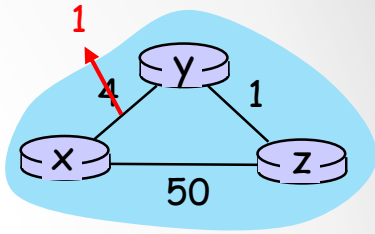
With distance vector routing

- **Good news travels fast**
- **Bad news travels slow**

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors



"good news travels fast"

t_0 : y detects link-cost change, updates its DV, informs its neighbors.

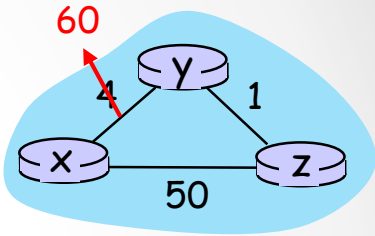
t_1 : z receives update from y, updates its table, computes new least cost to x , sends its neighbors its DV.

t_2 : y receives z's update, updates its distance table. y's least costs do *not* change, so y does *not* send a message to z.

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- "bad news travels slow" – count-to-infinity problem:

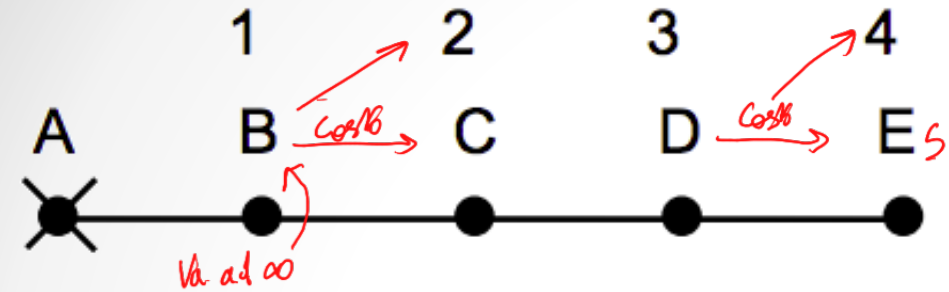


- *y* sees direct link to *x* has new cost 60, but z has said it has a path at cost of 5. So *y* computes "my new cost to x will be 6, via z); notifies *z* of new cost of 6 to *x*.
- *z* learns that path to *x* via *y* has new cost 6, so *z* computes "my new cost to *x* will be 7 via y), notifies *y* of new cost of 7 to *x*.
- *y* learns that path to *x* via *z* has new cost 7, so *y* computes "my new cost to *x* will be 8 via y), notifies *z* of new cost of 8 to *x*.
- *z* learns that path to *x* via *y* has new cost 8, so *z* computes "my new cost to *x* will be 9 via y), notifies *y* of new cost of 9 to *x*.

…

- ***Possibile soluzione al problema?***

Count to Infinity



1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges

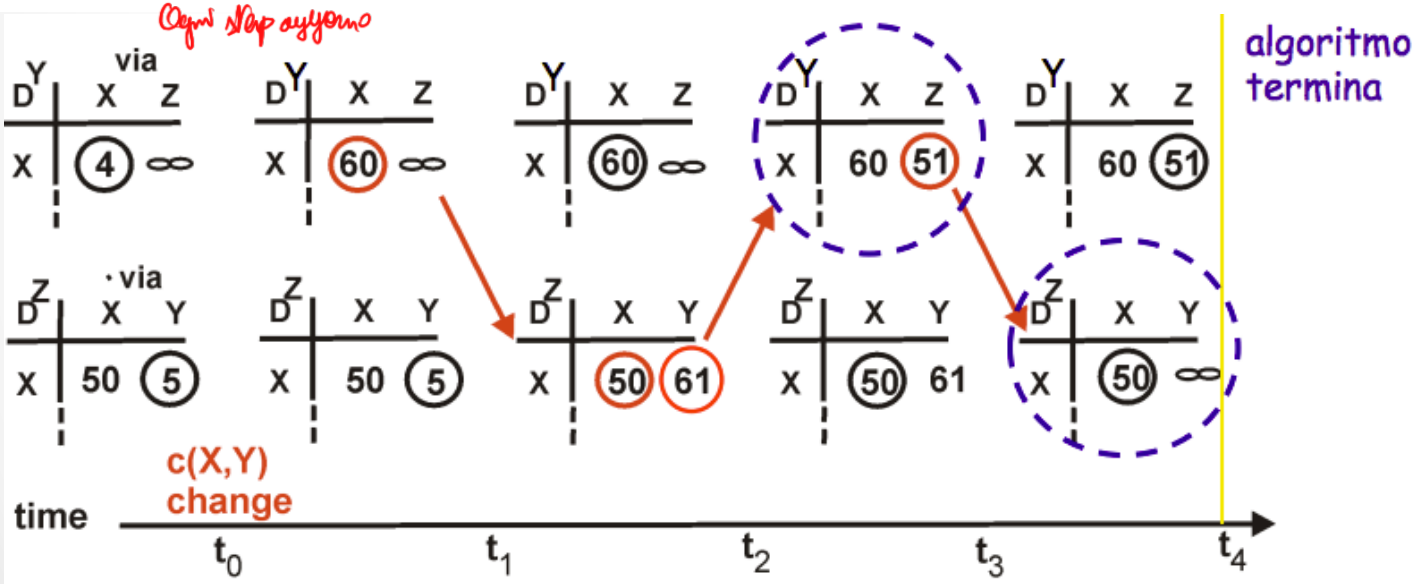
etc… to infinity

- ***Possibile soluzione al problema?***

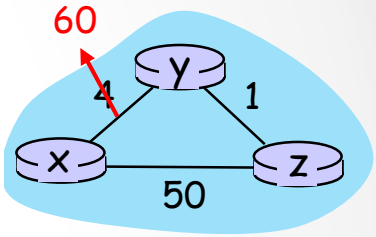
Distance vector: poisoned reverse

Se z raggiunge x tramite y:

- z dice a y che la sua distanza per x è infinita (così y non andrà a x attraverso z)
- Viene risolto completamente il problema?



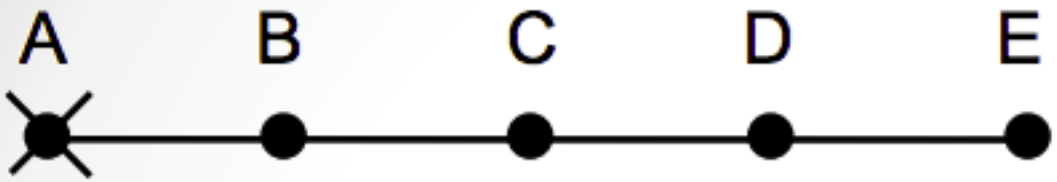
algoritmo termina



…

- ***Problema risolto?***

Count to Infinity



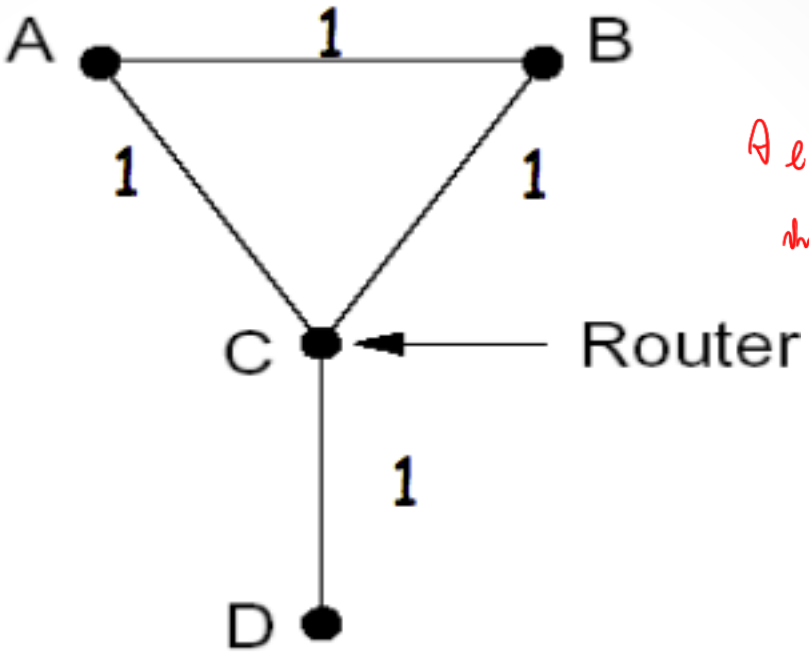
Stanno tutti che è calcolato

inf.	2	3	4	B learns A is dead
inf.	inf. → 2	3	4	B reports to C that A's metric is inf.
inf.	inf.	3	4	After 1 exchange
inf.	inf.	inf.	4	After 2 exchanges
inf.	inf.	inf.	inf.	After 3 exchanges

- ***Problema risolto?***


NO

Fallimento Poisoned Reverse



A e B si passeggiano informazione di avere su D.

- Quando il link tra C e D si interrompe, C "setterà" la sua distanza da D ad ∞
- Però, A userà B per andare a D e B userà A per andare a D.
- Dopo questi update, sia A che B riporteranno un nuovo percorso da C a D (diverso da ∞)

Comparison of LS and DV algorithms

message complexity

LS: n routers, $O(n^2)$ messages sent

DV: exchange between neighbors; convergence time varies

speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages

- may have oscillations

DV: convergence time varies

- may have routing loops
- count-to-infinity problem

robustness: what happens if router malfunctions, or is compromised?

LS:

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

DV:

- DV router can advertise incorrect *path* cost ("I have a *really* low cost path to everywhere"): black-holing
- each router's table used by others: error propagate thru network