

Università degli Studi di Roma "Tor Vergata"  
Laurea in Informatica

Sistemi Operativi e Reti  
(modulo Reti)  
a.a. 2023/2024

# Livello di rete: piano di controllo (parte1)

dr. Manuel Fiorelli

[manuel.fiorelli@uniroma2.it](mailto:manuel.fiorelli@uniroma2.it)

<https://art.uniroma2.it/fiorelli>

Basate sulle slide del libro di testo:

[https://gaia.cs.umass.edu/kurose\\_ross/ppt.php](https://gaia.cs.umass.edu/kurose_ross/ppt.php)

# Piano di controllo del livello di rete: obiettivi

- comprendere i principi alla base del piano di controllo:
  - algoritmi di instradamento tradizionali
  - controller SDN
  - gestione e configurazione della rete
- istanziamento, implementazione in Internet:
  - OSPF, BGP
  - OpenFlow, ODL e controller ONOS
  - Internet Control Message Protocol: ICMP
  - SNMP, YANG/NETCONF

# Livello di rete: tabella di marcia del “piano di controllo”

- **introduzione**
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento interno al sistema autonomo: OSPF
- instradamento tra sistemi autonomi: BGP
- piano di controllo SDN
- Internet Control Message Protocol



- gestione e configurazione della rete
  - SNMP
  - NETCONF/YANG

# Funzioni del livello di rete

- **inoltro:** spostare i pacchetti dall'ingresso del router all'uscita del router appropriata
- **instradamento:** determinare il percorso seguito dai pacchetti dalla sorgente alla destinazione

*piano dei dati*

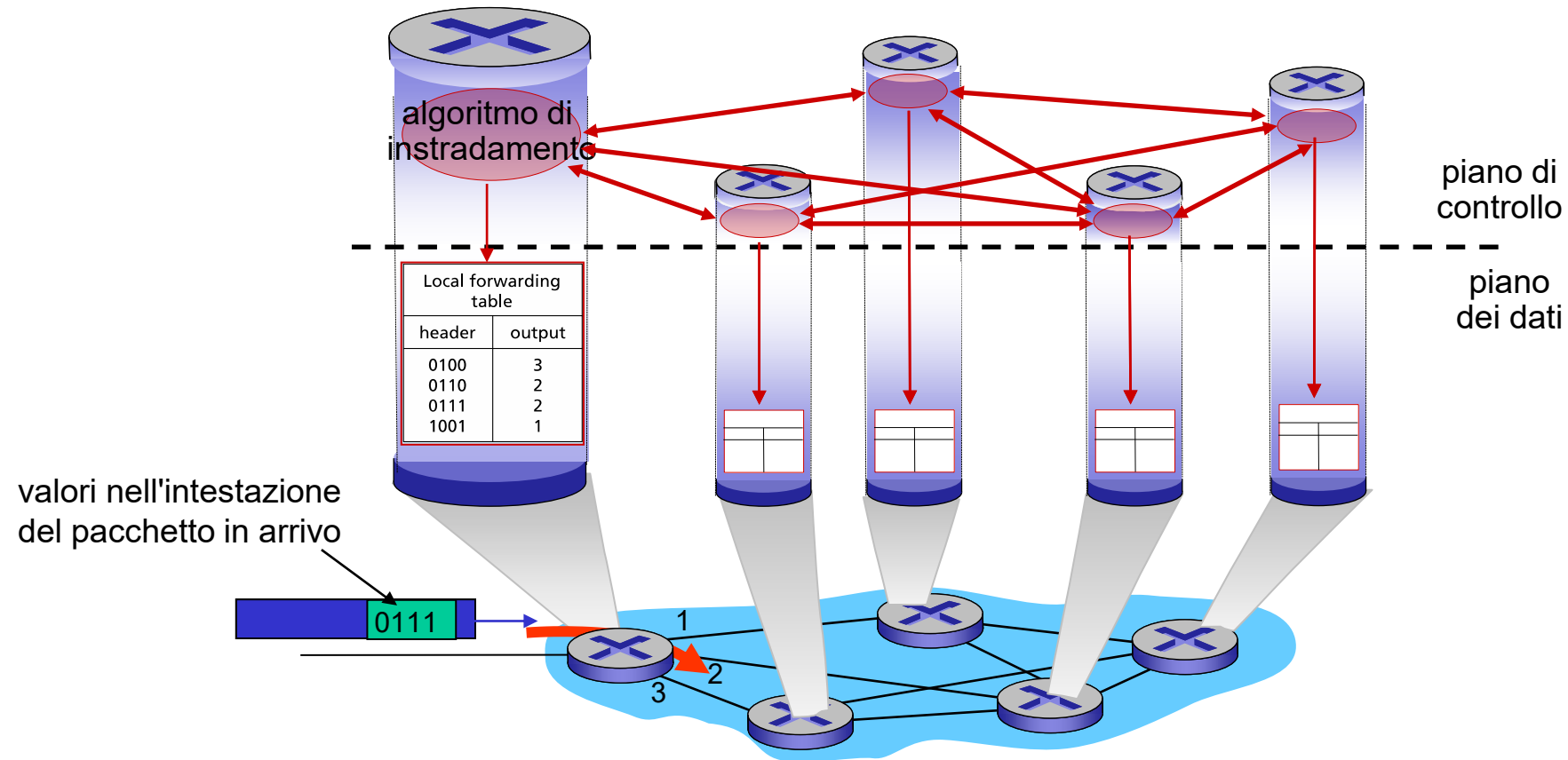
*piano di controllo*

## Due approcci alla strutturazione del piano di controllo della rete:

- controllo per router (tradizionale)
- controllo logicamente centralizzato (software defined networking)

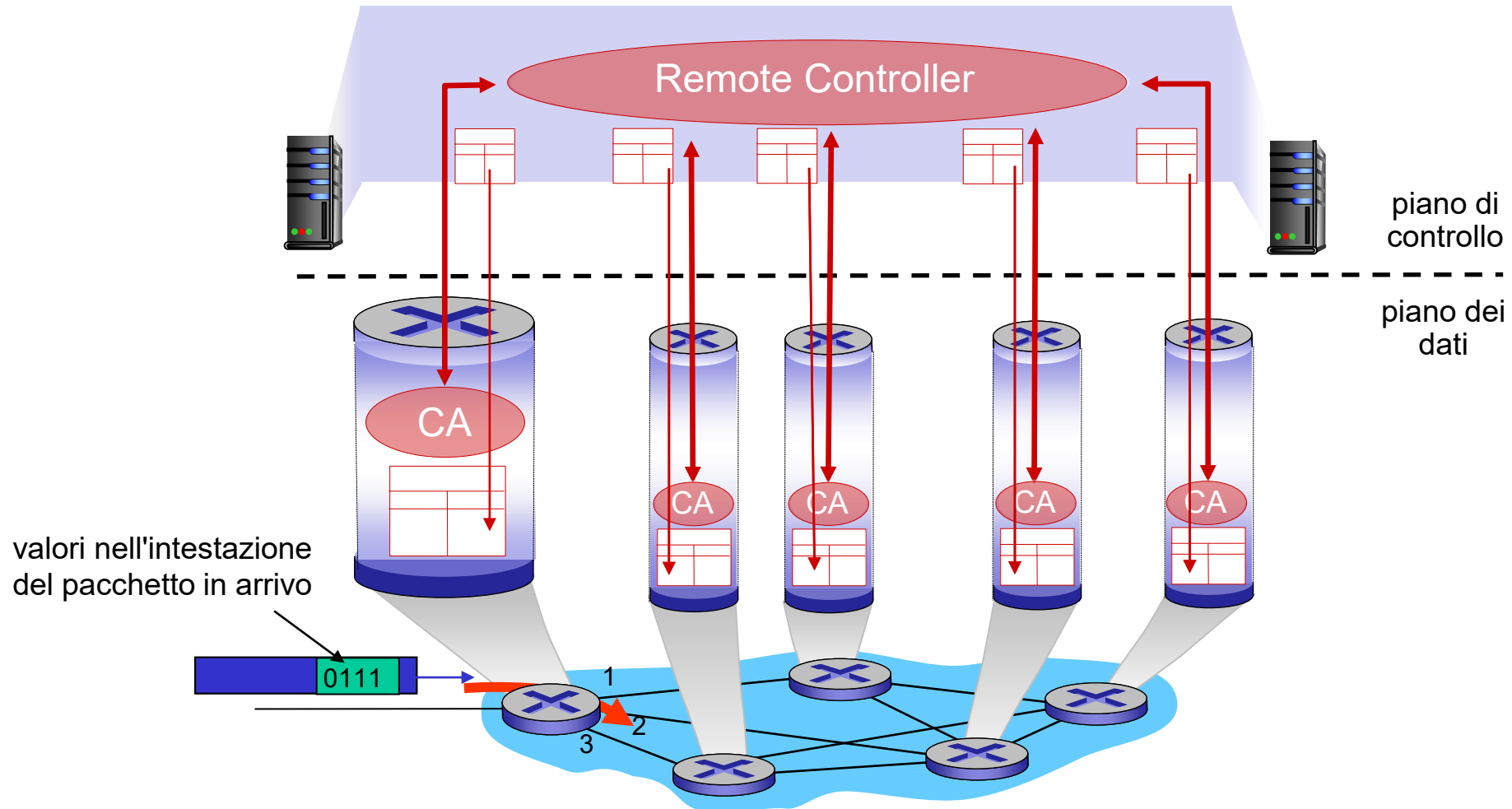
# Piano di controllo per router

I singoli componenti dell'algoritmo di instradamento *in ogni router* interagiscono nel piano di controllo.

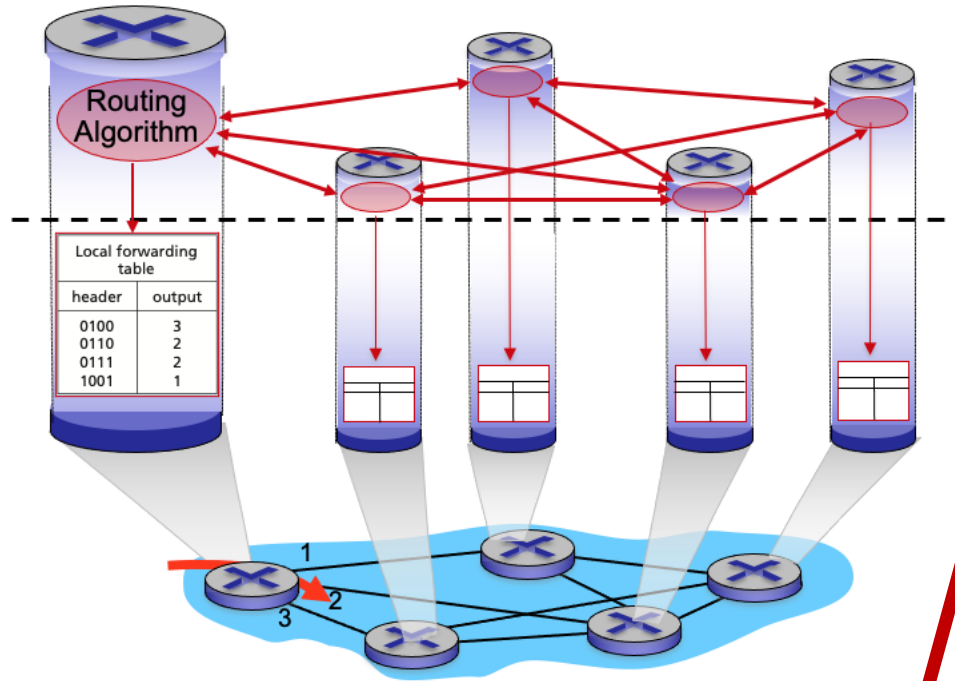


# Piano di controllo Software-Defined Networking (SDN)

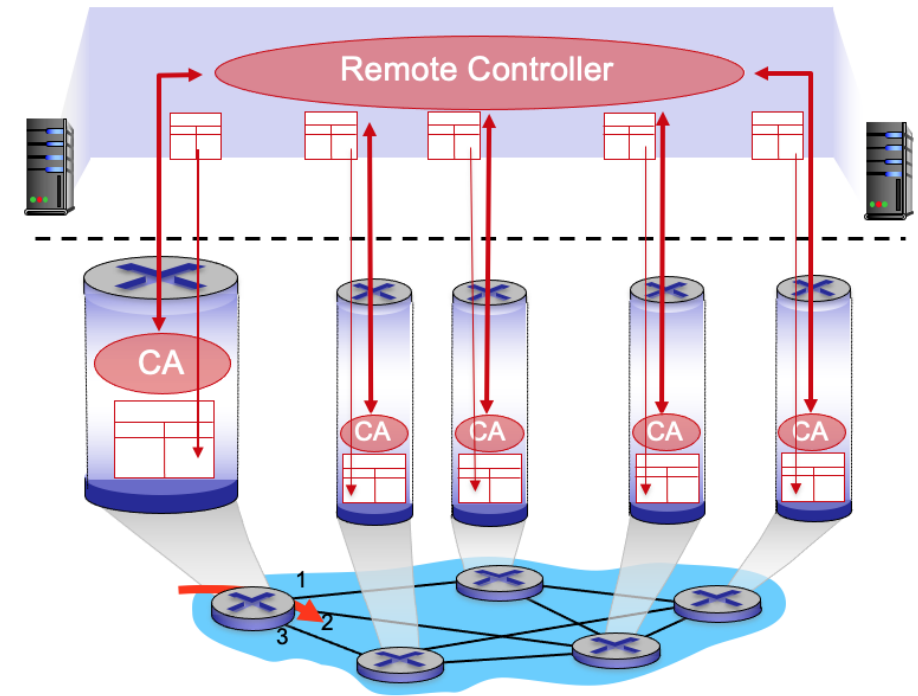
Un controllore remoto calcola le tabelle di inoltro e le installa nei router



# Piano di controllo per router



# Piano di controllo SDN



# Livello di rete: tabella di marcia del “piano di controllo”

- introduzione
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento interno al sistema autonomo: OSPF
- instradamento tra sistemi autonomi: BGP
- piano di controllo SDN
- Internet Control Message Protocol



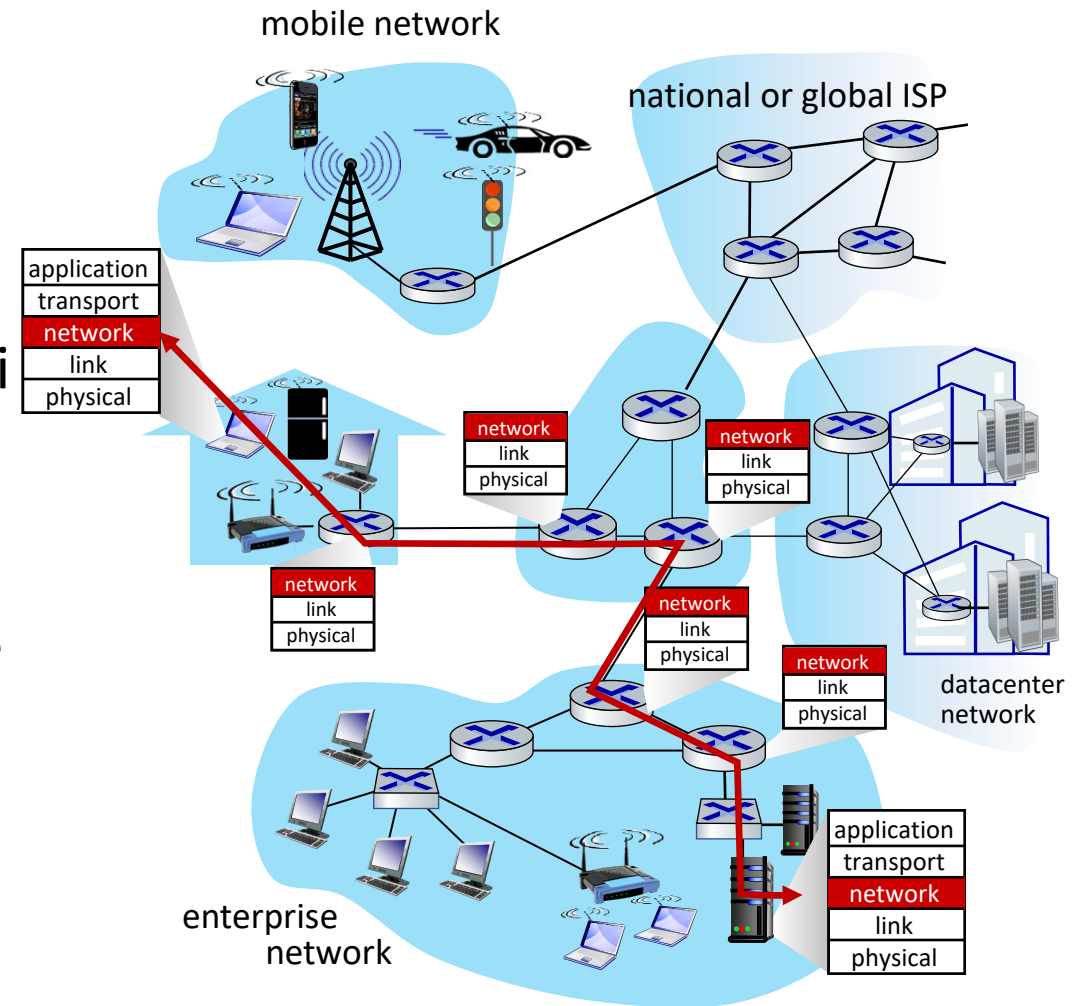
- gestione e configurazione della rete
  - SNMP
  - NETCONF/YANG



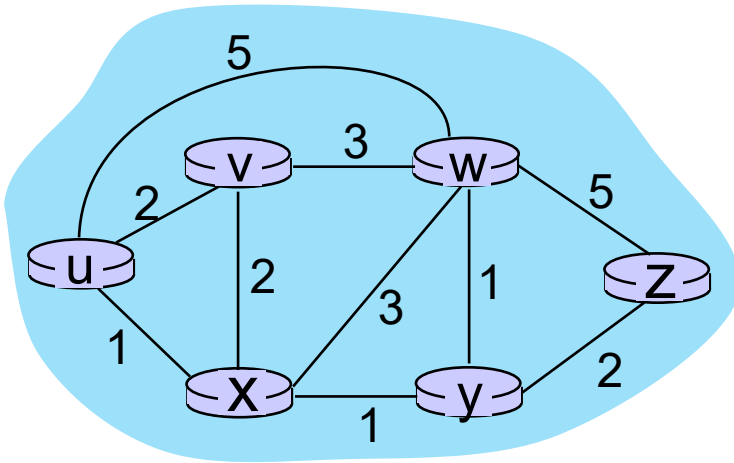
# Algoritmi di instradamento (*routing algorithms*)

**Obiettivo degli algoritmi di instradamento:** determinare percorsi o cammini "buoni" tra le sorgenti e i destinatari attraverso la rete di router

- **percorso:** sequenza di router che i pacchetti attraversano dall'host di origine all'host di destinazione (ovviamente, presupponendo l'attraversamento di un collegamento tra un nodo e il successivo)
- **"buono":** "costo" minimo (anche se occorre considerare ulteriori vincoli, spesso determinati da *policy* degli amministratori delle reti)
- instradamento: uno dei "primi 10" problemi nelle reti!



# Astrazione: grafo



grafo:  $G = (N, E)$

$N$ : insieme di router =  $\{ u, v, w, x, y, z \}$

$E$ : insieme di collegamenti =  $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

$c_{a,b}$ : costo del collegamento *diretto* che connette  $a$  e  $b$

es.,  $c_{w,z} = 5, c_{u,z} = \infty$

$z$  è adiacente o vicino  
(neighbor)

costo definito dall'operatore di rete: potrebbe essere sempre 1, o proporzionale alla lunghezza fisica di un collegamento (ritardo di propagazione), o inversamente correlato alla larghezza di banda, o inversamente correlato alla congestione

Il costo di un *percorso* è uguale alla somma dei costi dei collegamenti attraversati.

# Classificazione degli algoritmi di instradamento



# Classificazione degli algoritmi di instradamento

- **sensibili al carico:** il costo dei collegamenti riflette il livello corrente di congestione (es. correlato al ritardo di accodamento)
- **insensibili al carico:** il costo dei collegamenti non riflette il livello corrente (o recente) di congestione

A causa di difficoltà sperimentate nell'uso di algoritmi sensibili al carico ai tempi di ARPAnet, oggi si preferiscono algoritmi insensibili al carico.

# Livello di rete: tabella di marcia del “piano di controllo”

- introduzione
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento interno al sistema autonomo: OSPF
- instradamento tra sistemi autonomi: BGP
- piano di controllo SDN
- Internet Control Message Protocol



- gestione e configurazione della rete
  - SNMP
  - NETCONF/YANG

# Instradamento "link-state": algoritmo di Dijkstra

- **centralizzato**: la topologia della rete e il costo dei collegamenti sono noti a *tutti* i nodi
  - Informazioni ottenute attraverso un algoritmo di "link state broadcast"
  - tutti i nodi hanno le stesse informazioni
- calcola i percorsi di costo minimo da un nodo ("sorgente") a tutti gli altri nodi
  - dà la *tabella di inoltro* per quel nodo
- **iterativo**: dopo  $k$  iterazioni, conosce il cammino di costo minimo verso  $k$  destinazioni

## notazione

- $c_{x,y}$ : costo del collegamento diretto dal nodo  $x$  al nodo  $y$ ;  
 $= \infty$  se non sono vicini diretti
- $D(v)$ : stima *corrente* del costo minimo del percorso dalla sorgente alla destinazione  $v$
- $p(v)$ : immediato predecessore di  $v$  lungo il percorso a costo minimo dall'origine a  $v$
- $N'$ : sottoinsieme di nodi contenente tutti (e solo) i nodi  $v$  per cui il percorso a costo minimo dall'origine a  $v$  è *definitivamente* noto

# Instradamento "link-state": algoritmo di Dijkstra

1 *Inizializzazione:*

2  $N' = \{u\}$  */\* calcola il percorso di minor costo da u a tutti gli altri nodi \*/*

3 per tutti i nodi  $v$

4 se  $v$  è adiacente a  $u$  */\* inizialmente conosce il costo del percorso diretto solo per i vicini diretti \*/*

5 allora  $D(v) = c_{u,v}$  */\* ma potrebbe non essere di costo minimo \*/*

6 altrimenti  $D(v) = \infty$

7



8 *Ciclo*

9 determina un  $w$  non in  $N'$  tale che  $D(w)$  sia minimo

10 aggiungi  $w$  a  $N'$

11 aggiorna  $D(v)$  per ciascun nodo  $v$  adiacente a  $w$  e non in  $N'$  :

12  **$D(v) = \min ( D(v), D(w) + c_{w,v} )$**

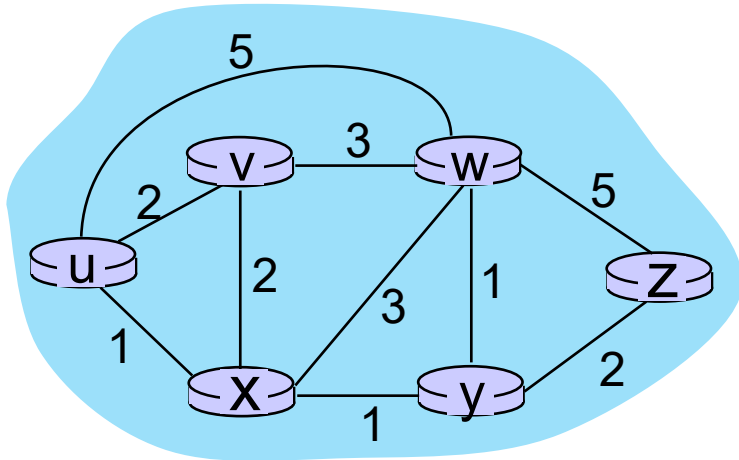
13 */\* il nuovo costo verso v è il vecchio costo verso v oppure il costo del percorso minimo*

14 *noto verso w più il costo da w a v \*/*

15 *Ripeti il ciclo finché non si verifica che  $N' = N$*

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1						
2						
3						
4						
5						



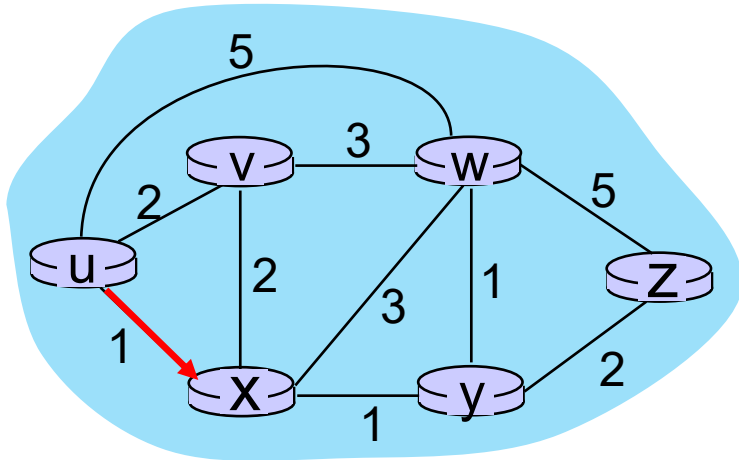
Inizializzazione (passo 0):

Per ogni  $a$ : se  $a$  è adiacente a  $u$  allora  $D(a) = c_{u,a}$



# Algoritmo di Dijkstra: un esempio

Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	$\infty$	$\infty$
1	ux					
2						
3						
4						
5						



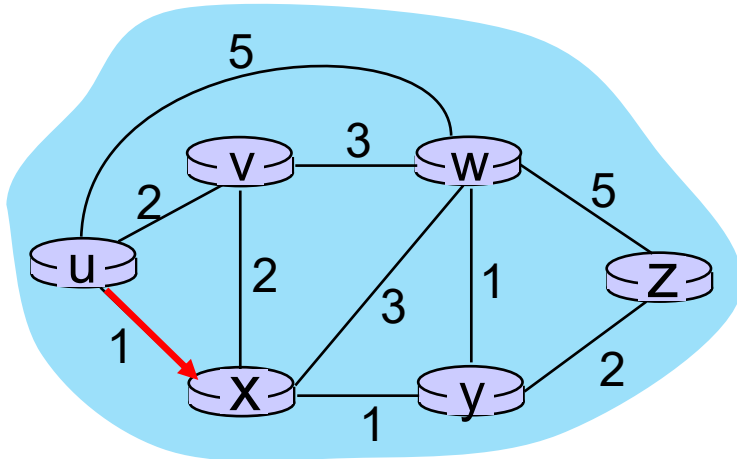
8 *Ciclo*

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2						
3						
4						
5						



8 **Ciclo**

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

11 aggiorna  $D(b)$  per ogni  $b$  adiacente a  $a$  e non in  $N'$ :

$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

$$D(v) = \min ( D(v), D(x) + c_{x,v} ) = \min(2, 1+2) = 2$$

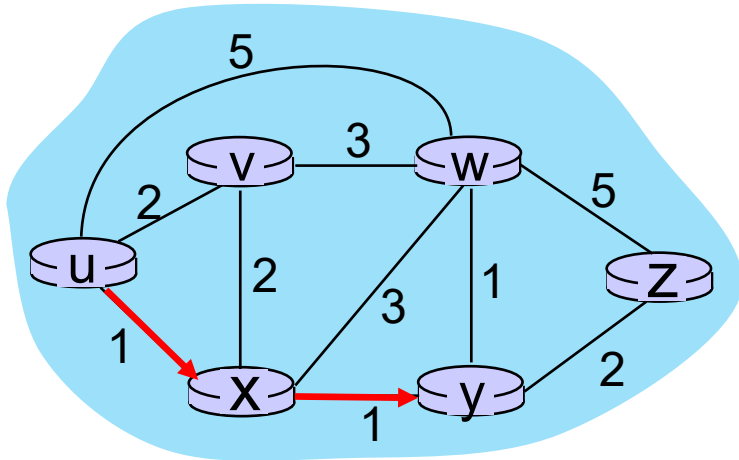
$$D(w) = \min ( D(w), D(x) + c_{x,w} ) = \min(5, 1+3) = 4$$

$$D(y) = \min ( D(y), D(x) + c_{x,y} ) = \min(\infty, 1+1) = 2$$



# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	$\infty$	$\infty$
1	ux	2, u	4, x		2, x	$\infty$
2	uxy					
3						
4						
5						



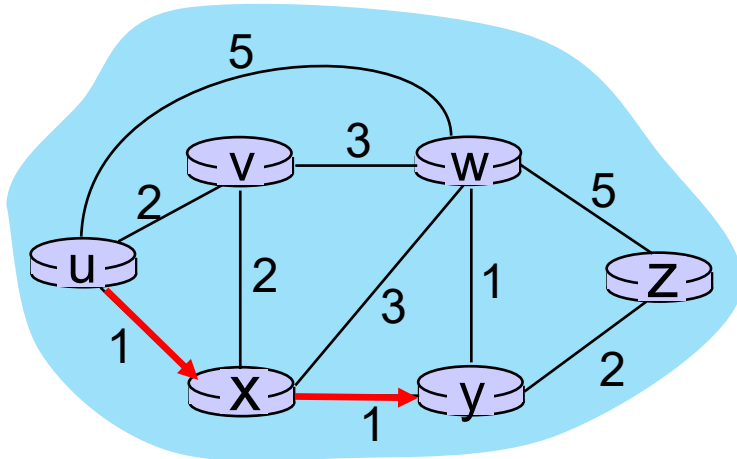
## 8 Ciclo

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3						
4						
5						



8 *Ciclo*

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

11 aggiorna  $D(b)$  per ogni  $b$  adiacente a  $a$  e non in  $N'$ :

$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

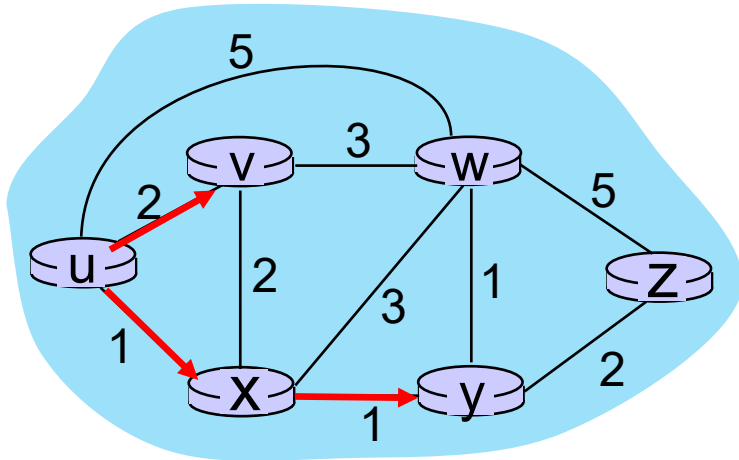
$$D(w) = \min ( D(w), D(y) + c_{y,w} ) = \min ( 4, 2+1 ) = 3$$

$$D(z) = \min ( D(z), D(y) + c_{y,z} ) = \min ( \infty, 2+2 ) = 4$$



# Algoritmo di Dijkstra: un esempio

Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						



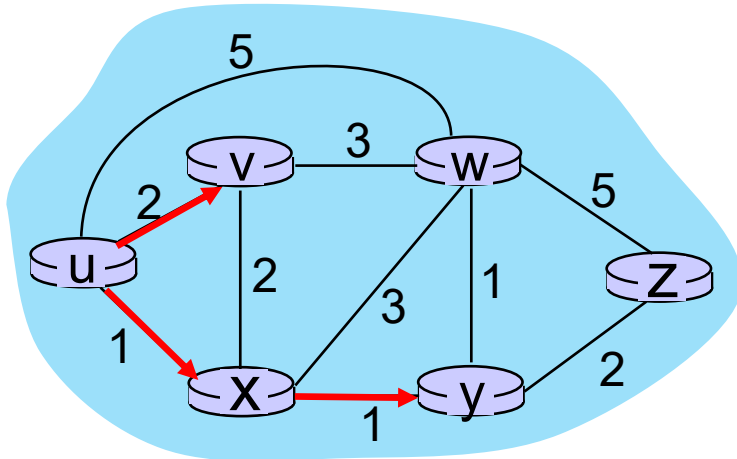
## 8 Ciclo

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x	2,x	$\infty$	$\infty$
2	uxy	2,u	3,y		4,y	
3	uxyv		3,y		4,y	
4						
5						



8 **Ciclo**

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

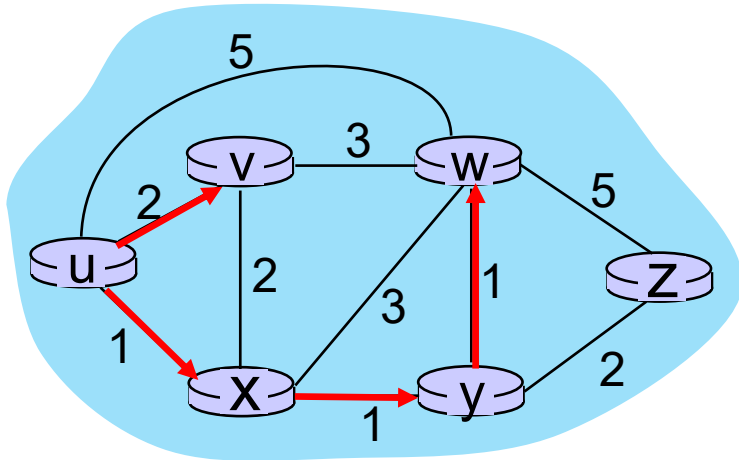
11 aggiorna  $D(b)$  per ogni  $b$  adiacente a  $a$  e non in  $N'$ :

$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

$$D(w) = \min ( D(w), D(v) + c_{v,w} ) = \min ( 3, 2+3 ) = 3$$

# Algoritmo di Dijkstra: un esempio

Step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x	2,x	$\infty$	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5						



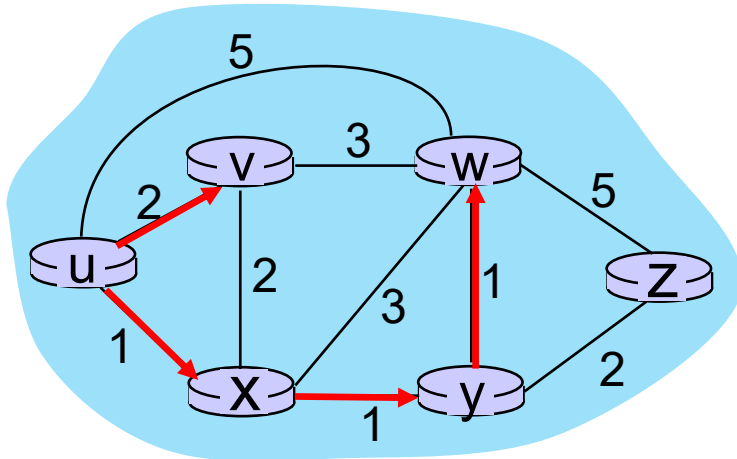
## 8 Ciclo

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5						



8 **Ciclo**

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

11 aggiorna  $D(b)$  per ogni  $b$  adiacente a  $a$  e non in  $N'$ :

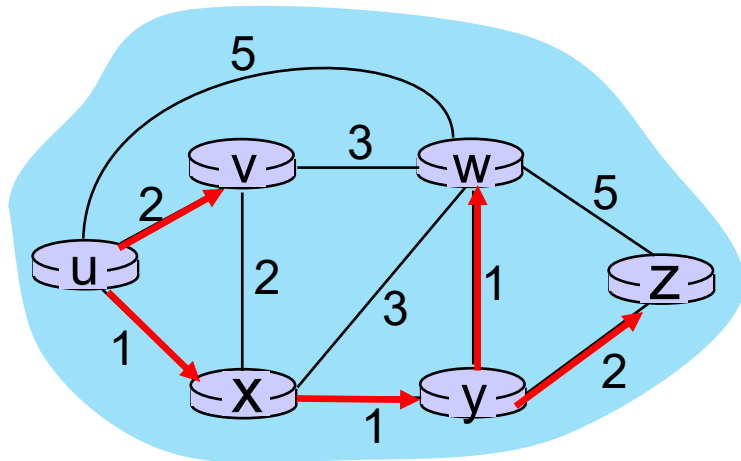
$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

$$D(z) = \min ( D(z), D(w) + c_{w,z} ) = \min ( 4, 3+5 ) = 4$$



# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	$\infty$	$\infty$
1	ux	2, u	4, x		2, x	$\infty$
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					



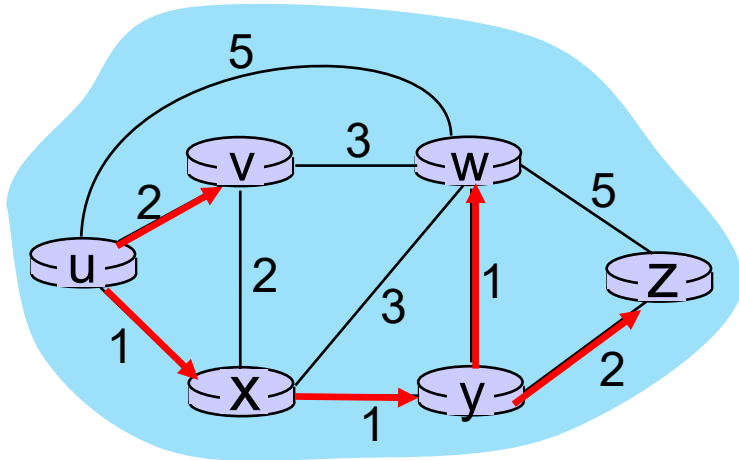
## 8 Ciclo

9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo

10 aggiungi  $a$  a  $N'$

# Algoritmo di Dijkstra: un esempio

		<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>
Passo	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

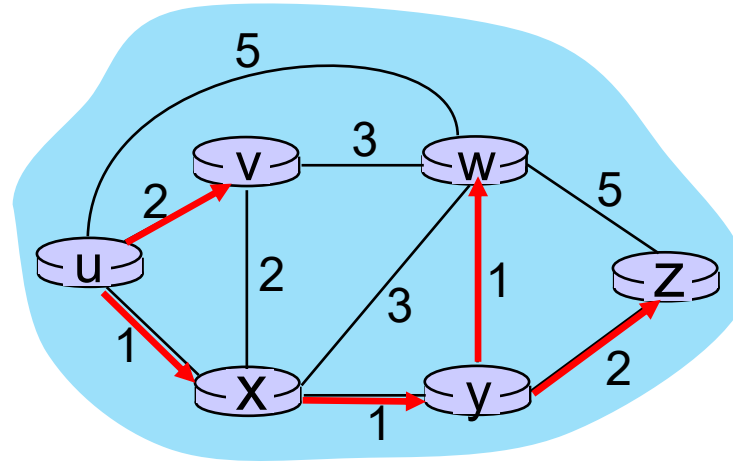


## 8 Ciclo

- 9 determina  $a$  non in  $N'$  tale che  $D(a)$  sia minimo
- 10 aggiungi  $a$  a  $N'$
- 11 aggiorna  $D(b)$  per ogni  $b$  adiacente a  $a$  e non in  $N'$ :  

$$D(b) = \min ( D(b), D(a) + c_{a,b} )$$

# Algoritmo di Dijkstra: un esempio



Albero dei cammini minimi di  $u$  :

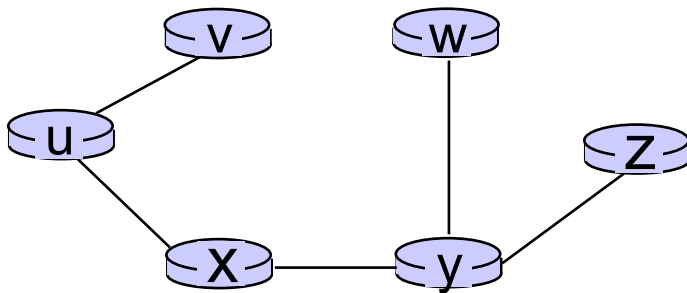


tabella di inoltro risultante in  $u$

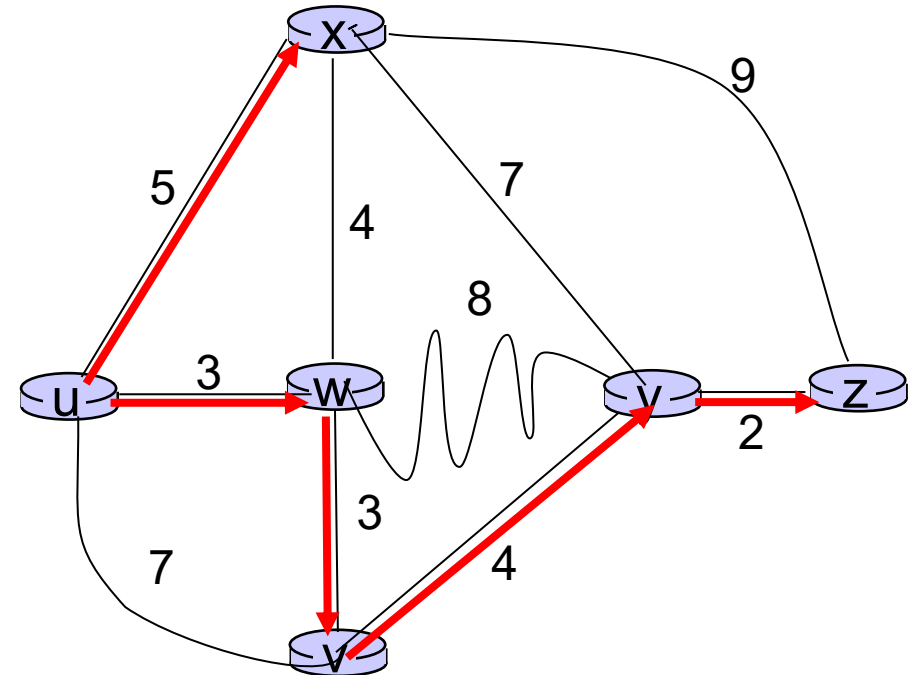
destinazione	collegamento di uscita
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

instrada da  $u$  a  $v$  direttamente

instrada da  $u$  a tutte le altre destinazioni attraverso  $x$

# Algoritmo di Dijkstra: un altro esempio

Passo	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwvx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



note:

- costruisce l'albero dei cammini minimi tenendo traccia del predecessore
- ci possono essere pareggi (possono essere risolti arbitrariamente)

# Algoritmo di Dijkstra: discussione

**Complessità algoritmica:**  $n$  nodi (senza contare l'origine)

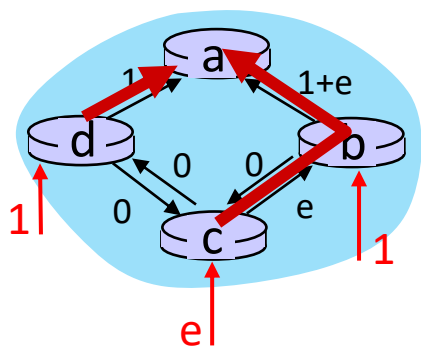
- ciascuna delle  $n$  iterazioni: deve controllare tutti i nodi,  $w$ , non in  $N$  per determinare quello avente il costo minimo:  $n$  nodi,  $n - 1$  nodi, ..., 1
- complessità  $O(n^2)$
- sono possibili implementazioni più efficienti:  $O(n \log n)$  usando uno *heap*

**Complessità dei messaggi:**  $n$  nodi

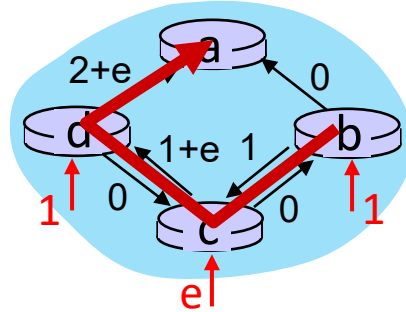
- ogni router deve *trasmettere in broadcast* le proprie informazioni sullo stato dei collegamenti agli altri  $n$  router
- algoritmi di broadcasting efficienti (e interessanti!):  $O(n)$  attraversamenti dei collegamenti per diffondere un messaggio di broadcasting da una sorgente
- il messaggio di ogni router attraversa  $O(n)$  collegamenti: complessità dei messaggi complessiva:  $O(n^2)$

# Algoritmo di Dijkstra: discussione: oscillazioni

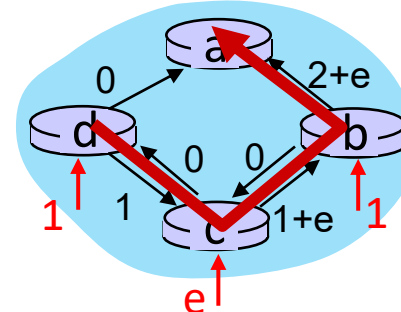
- quando i costi dei collegamenti dipendono dal volume di traffico, sono possibili **oscillazioni dei percorsi**
- scenario di esempi:
  - instradamento verso  $a$ : i nodi  $b$ ,  $d$  e  $c$  trasmettono rispettivamente con tasso 1, 1, e ( $<1$ )
  - Il costo dei collegamenti è direzionale dipendente dal traffico



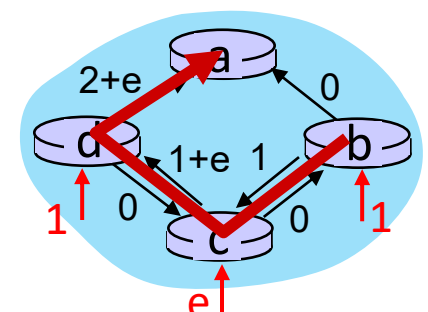
instradamento  
iniziale



dati questi costi,  
trovano un nuovo  
instradamento....  
producendo nuovi costi



dati questi costi,  
trovano un nuovo  
instradamento....  
producendo nuovi costi



dati questi costi,  
trovano un nuovo  
instradamento....  
producendo nuovi costi

# Livello di rete: tabella di marcia del “piano di controllo”

- introduzione
- algoritmi di instradamento
  - link state
  - **distance vector**
- instradamento interno al sistema autonomo: OSPF
- instradamento tra sistemi autonomi: BGP
- piano di controllo SDN
- Internet Control Message Protocol



- gestione e configurazione della rete
  - SNMP
  - NETCONF/YANG

# Algoritmo distance vector

Basato sulla equazione di *Bellman-Ford* (BF) (programmazione dinamica):

Equazione di Bellman-Ford

Sia  $d_x(y)$ : il costo del percorso di costo minimo da  $x$  a  $y$ .

Allora:

$$d_x(y) = \min_v \{ c_{x,v} + d_v(y) \}$$

costo del cammino minimo da  $v$  a  $y$

costo diretto del collegamento da  $x$  a  $v$

$\min$  calcolato su tutti i vicini  $v$  di  $x$



# Algoritmo distance vector

Basato sulla equazione di *Bellman-Ford* (BF) (programmazione dinamica):

Equazione di Bellman-Ford

Sia  $D_x(y)$ : *una stima del costo del percorso di costo minimo da x a y.*

Allora:

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

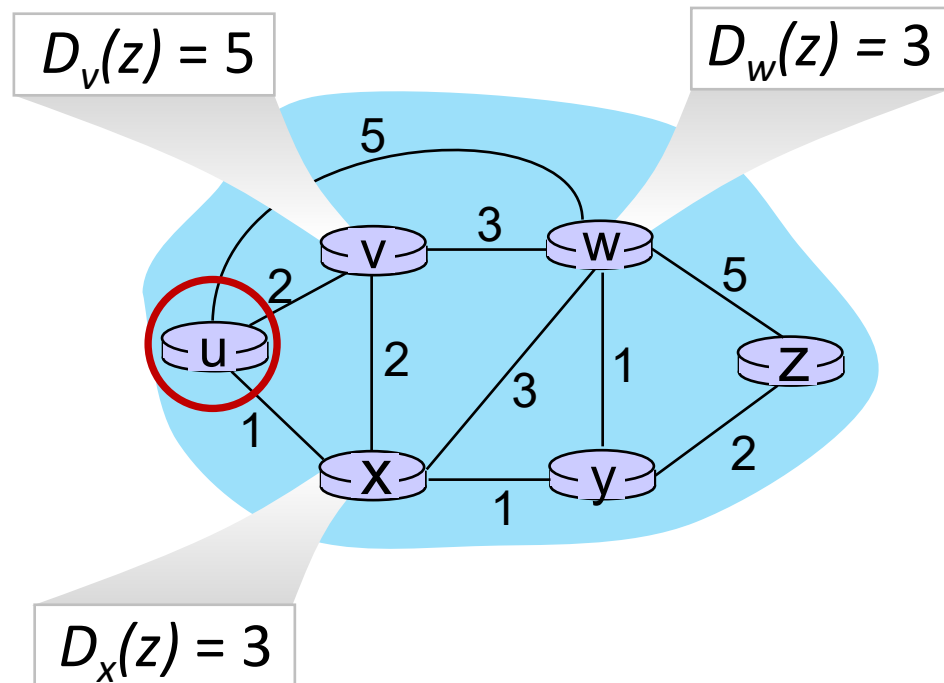
costo stimato del cammino minimo da v a y

costo diretto del collegamento da x a v

*min* calcolato su tutti i vicini v di x

# Bellman-Ford: esempio

Si supponga che i nodi vicini di  $u$ ,  $x, v, w$ , sappiano che per la destinazione  $z$ :



L'equazione di Bellman-Ford dice:

$$\begin{aligned} D_u(z) &= \min \{ c_{u,v} + D_v(z), \\ &\quad c_{u,x} + D_x(z), \\ &\quad c_{u,w} + D_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

*il nodo che raggiunge il minimo ( $x$ )  
è l'hop successivo sul percorso a  
costo minimo stimato verso la  
destinazione ( $z$ )*

# Algoritmo distance vector

## idea chiave:

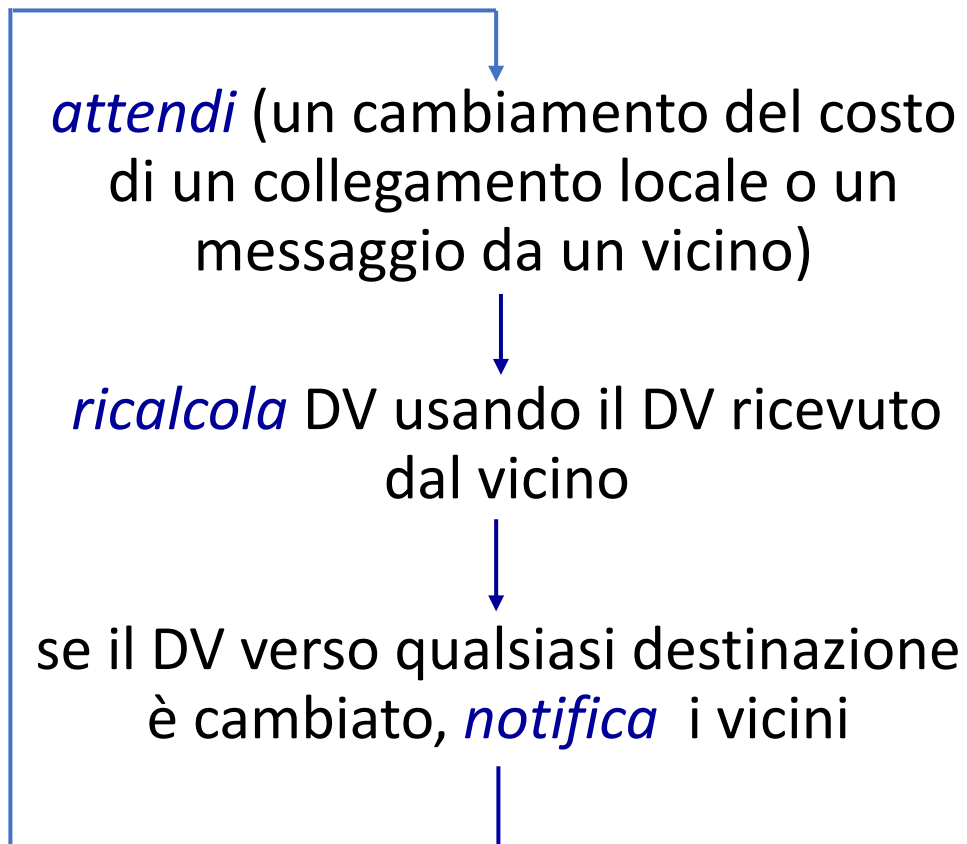
- di tanto in tanto, ogni nodo invia ai vicini il proprio vettore delle distanze (stimate), *distance vector* in inglese
- quando  $x$  riceve un DV da un qualsiasi vicino, aggiorna la propria DV utilizzando l'equazione B-F:

$$D_x(y) \leftarrow \min_v \{c_{x,v} + D_v(y)\} \text{ per ogni nodo } y \in N$$

- sotto certe condizioni minori e naturali, la stima  $D_x(y)$  converge verso l'effettivo costo minimo  $d_x(y)$

# Algoritmo distance vector

## ciascun nodo:



**iterativo, asincrono:** ciascuna iterazione locale causata:

- cambiamento del costo del collegamento locale
- messaggio di aggiornamento del DV da un vicino

## **distribuito, auto-terminante:**

ciascun nodo notifica i vicini *solo* quando la sua DV cambia

- i vicini notificano i loro vicini - *solo se necessario*.
- nessuna notifica ricevuta, nessuna azione intrapresa!

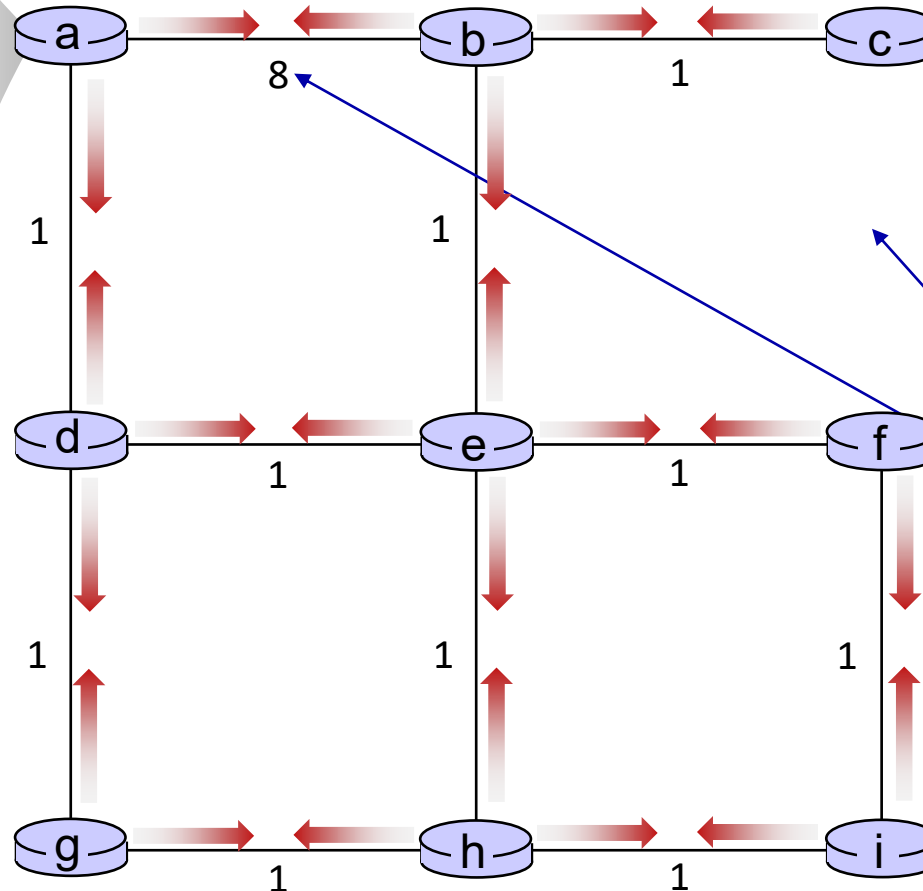
# Distance vector: esempio



t=0

- Tutti i nodi hanno stime della distanza dai vicini più prossimi (solo)
- Tutti i nodi inviano il proprio vettore di distanza locale ai propri vicini

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



Alcune asimmetrie:  
■ collegamento mancante  
■ costo più grande

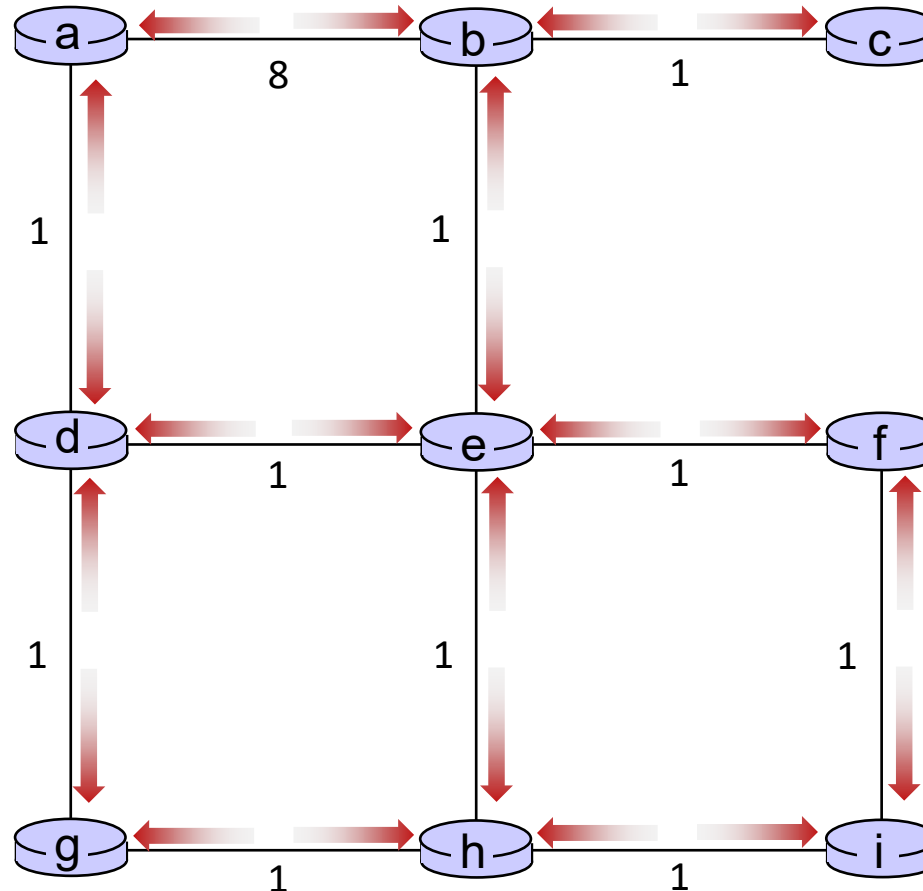
# Distance vector: esempio: iterazione



$t=1$

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviano il loro nuovo vettore delle distanze locale ai vicini



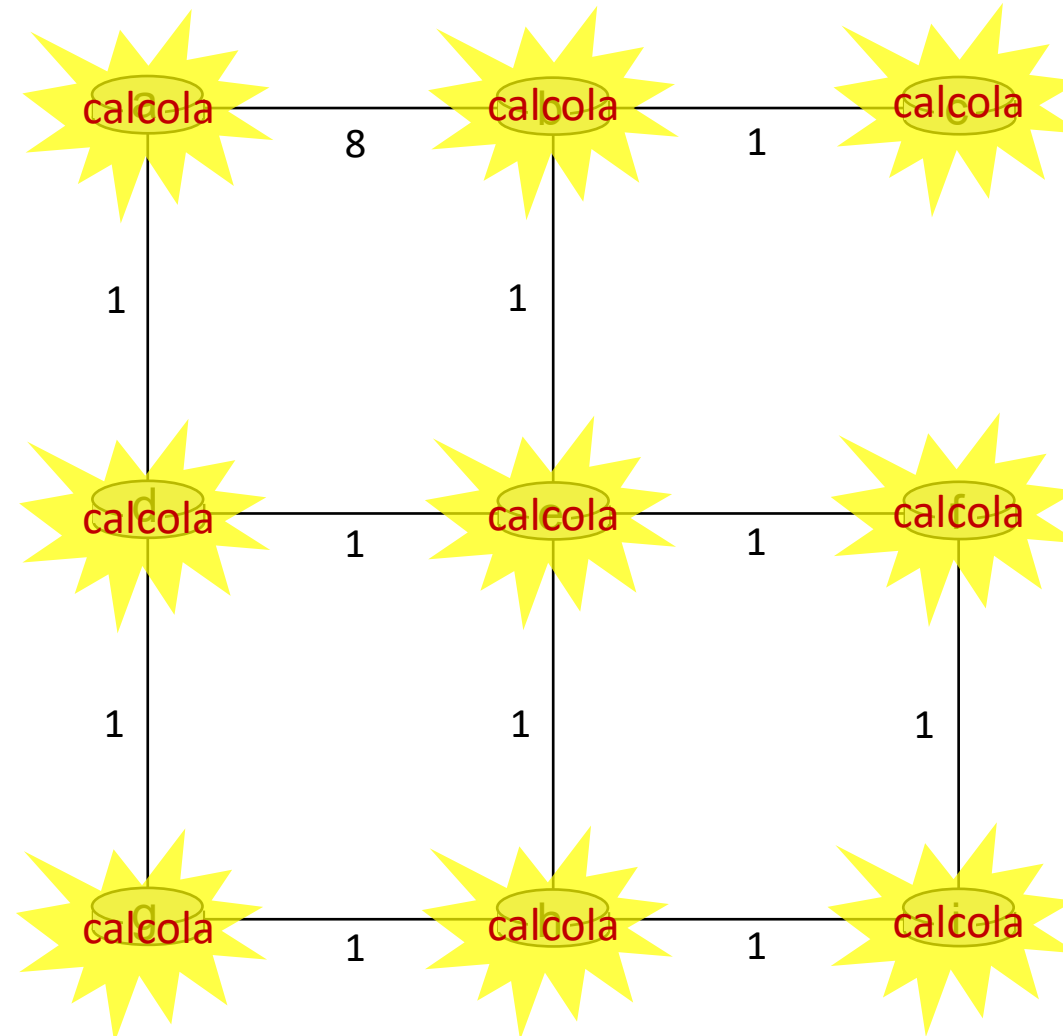
# Distance vector: esempio: iterazione



**t=1**

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviano il loro nuovo vettore delle distanze locale ai vicini



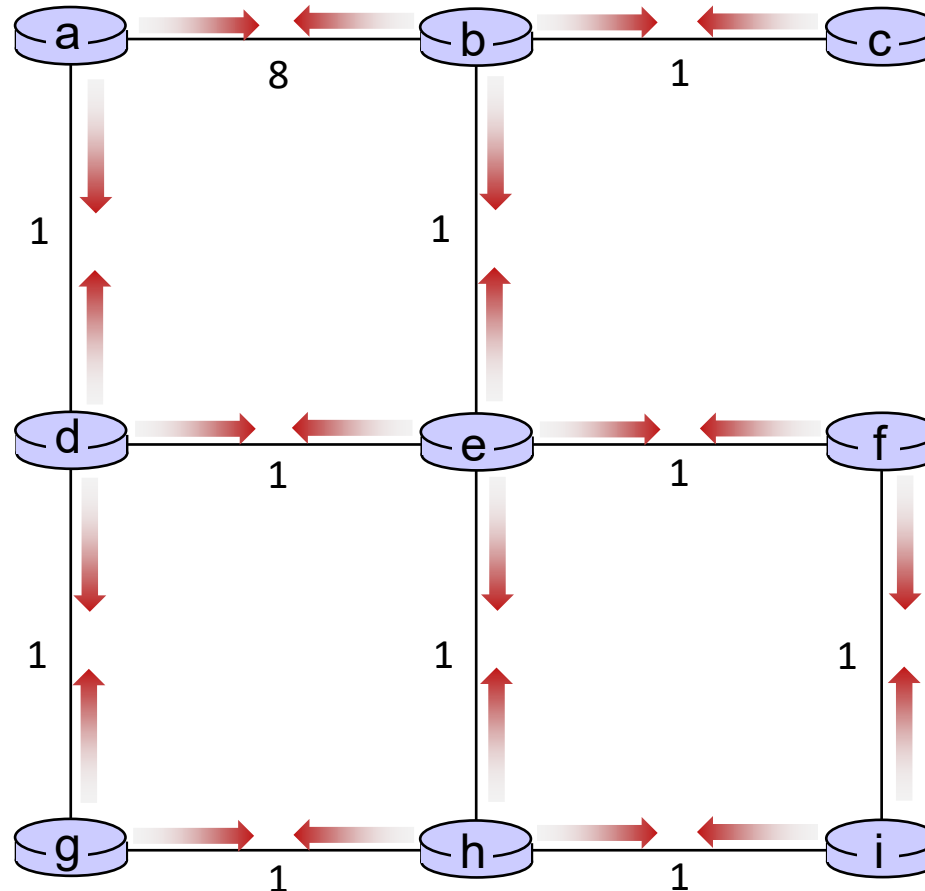
# Distance vector: esempio: iterazione



$t=1$

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviano il loro nuovo vettore delle distanze locale ai vicini





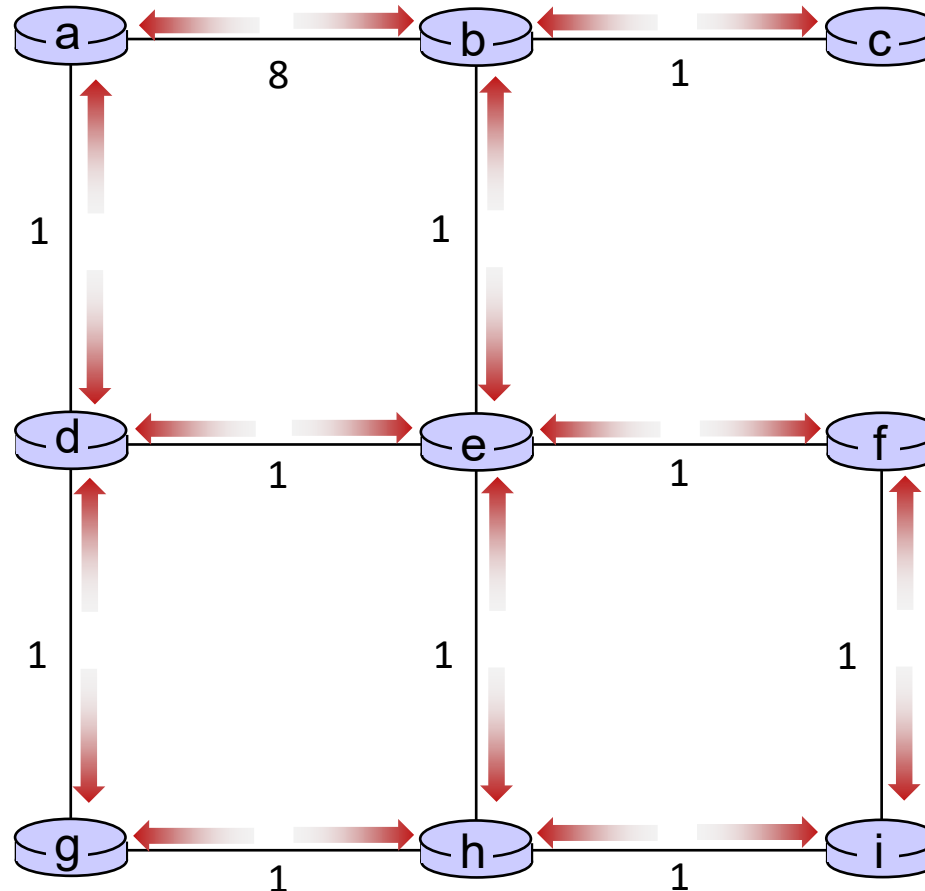
# Distance vector: esempio: iterazione



t=2

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviano il loro nuovo vettore delle distanze locale ai vicini



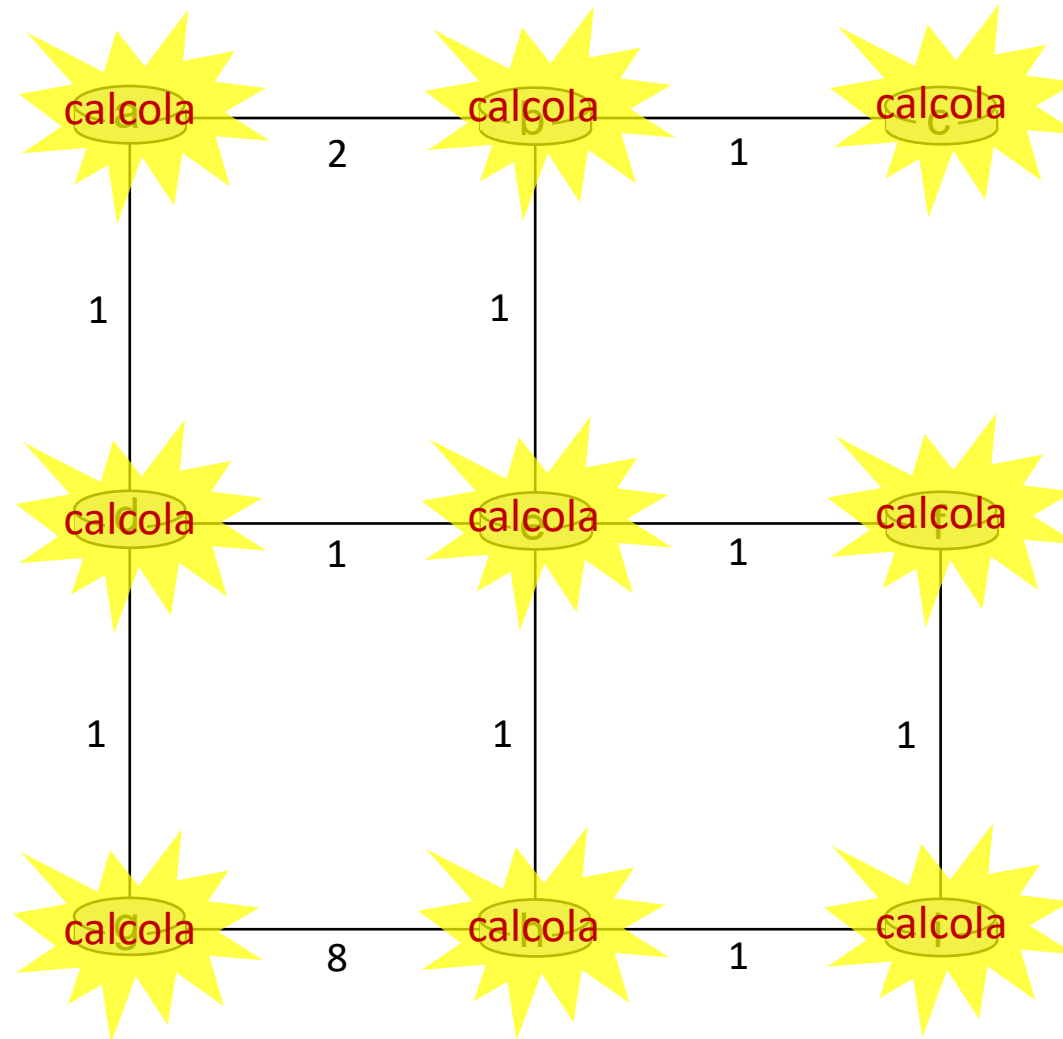
# Distance vector: esempio: iterazione



t=2

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviano il loro nuovo vettore delle distanze locale ai vicini



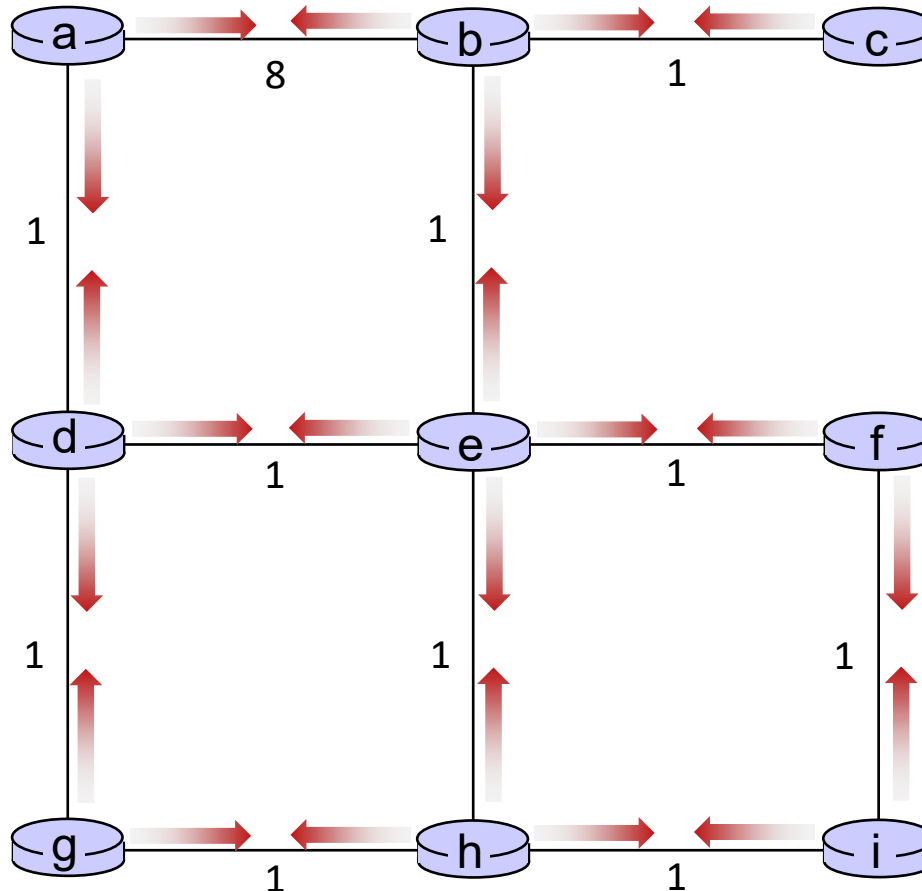
# Distance vector: esempio: iterazione



t=2

Tutti i nodi:

- ricevono i vettori delle distanze dai vicini
- calcolano il loro nuovo vettore delle distanze locale
- inviando il loro nuovo vettore delle distanze locale ai vicini



# Distance vector: esempio: iterazione

.... e così via

Vediamo ora le *computazioni* iterative ai nodi

# Distance vector: esempio:



**t=1**

- *b* i riceve i DV da *a*, *c*, *e*

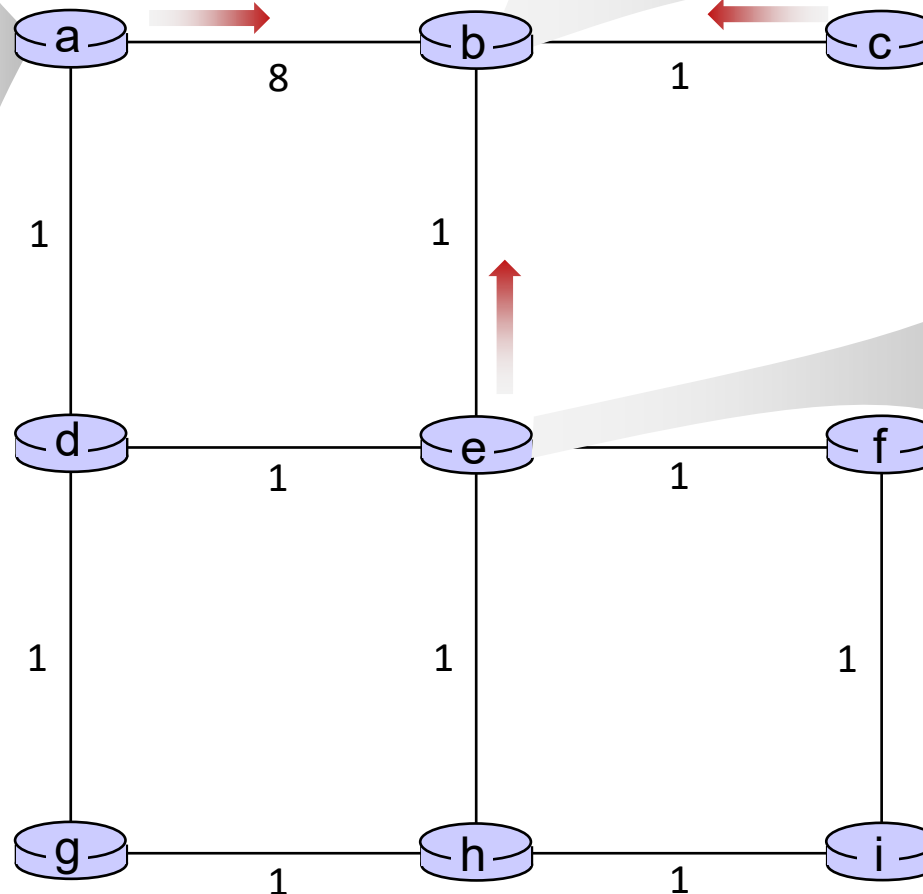
DV in a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

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

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



# Distance vector: esempio:



**t=1**

- *b* i riceve i DV da *a*, *c*, *e*, calcola:

## 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

8

calcola

1

e

## DV in b:

$D_b(a)=8$     $D_b(f)=\infty$   
 $D_b(c)=1$     $D_b(g)=\infty$   
 $D_b(d)=\infty$     $D_b(h)=\infty$   
 $D_b(e)=1$     $D_b(i)=\infty$

## DV in c:

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

## DV in e:

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

## DV in b:

$D_b(a)=8$     $D_b(f)=2$   
 $D_b(c)=1$     $D_b(g)=\infty$   
 $D_b(d)=2$     $D_b(h)=2$   
 $D_b(e)=1$     $D_b(i)=\infty$

$D_b(a) = \min\{c_{b,a}+D_a(a), c_{b,c}+D_c(a), c_{b,e}+D_e(a)\} = \min\{8, \infty, \infty\} = 8$   
 $D_b(c) = \min\{c_{b,a}+D_a(c), c_{b,c}+D_c(c), c_{b,e}+D_e(c)\} = \min\{\infty, 1, \infty\} = 1$   
 $D_b(d) = \min\{c_{b,a}+D_a(d), c_{b,c}+D_c(d), c_{b,e}+D_e(d)\} = \min\{9, 2, \infty\} = 2$   
 $D_b(e) = \min\{c_{b,a}+D_a(e), c_{b,c}+D_c(e), c_{b,e}+D_e(e)\} = \min\{\infty, \infty, 1\} = 1$   
 $D_b(f) = \min\{c_{b,a}+D_a(f), c_{b,c}+D_c(f), c_{b,e}+D_e(f)\} = \min\{\infty, \infty, 2\} = 2$   
 $D_b(g) = \min\{c_{b,a}+D_a(g), c_{b,c}+D_c(g), c_{b,e}+D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty$   
 $D_b(h) = \min\{c_{b,a}+D_a(h), c_{b,c}+D_c(h), c_{b,e}+D_e(h)\} = \min\{\infty, \infty, 2\} = 2$   
 $D_b(i) = \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$

# Distance vector: esempio:



**t=1**

- c riceve il DV da b

## DV in a:

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

## DV in b:

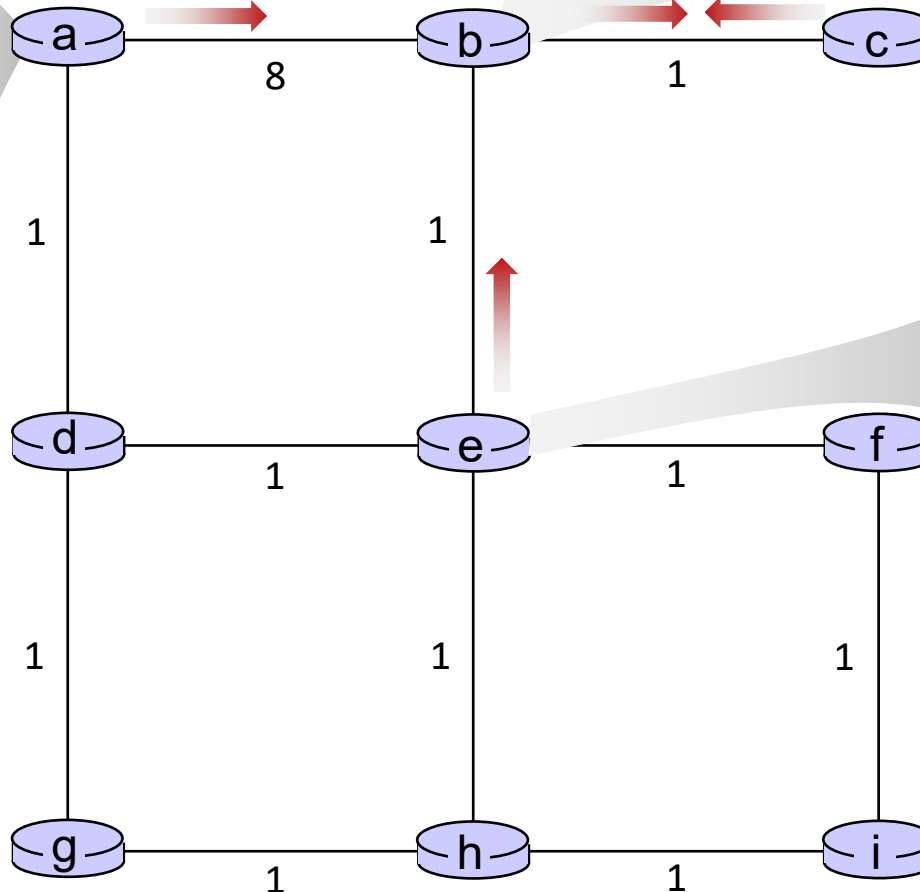
$D_b(a)=8$     $D_b(f)=\infty$   
 $D_b(c)=1$     $D_b(g)=\infty$   
 $D_b(d)=\infty$     $D_b(h)=\infty$   
 $D_b(e)=1$     $D_b(i)=\infty$

## DV in c:

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

## DV in e:

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

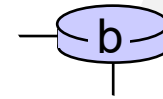


# Distance vector: esempio:



t=1

- c riceve il DV da b, calcola



1

calcola

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:

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

$$\begin{aligned} D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\ D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\ D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\ D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\ D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\ D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\ D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\ D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty \end{aligned}$$

DV in c:

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

\* Check out the online interactive exercises for more examples:  
[http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)



# Distance vector: esempio:

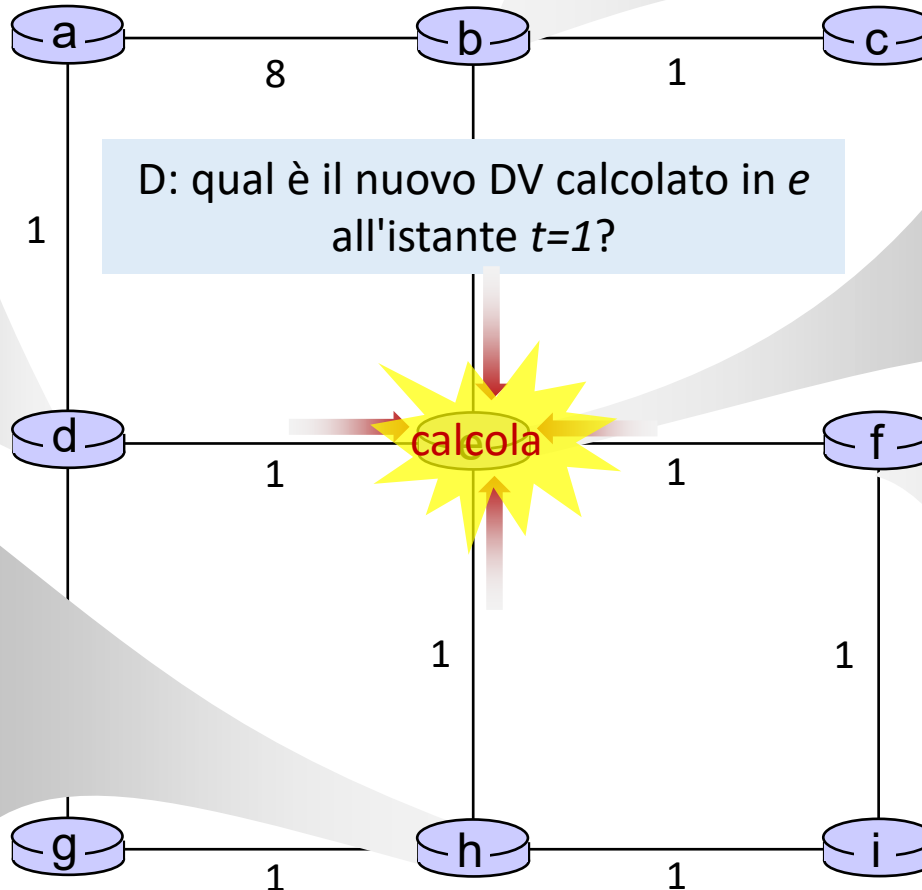


**t=1**

- e riceve i DV da b, d, f, h

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

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



D: qual è il nuovo DV calcolato in e all'istante  $t=1$ ?

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






e

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

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

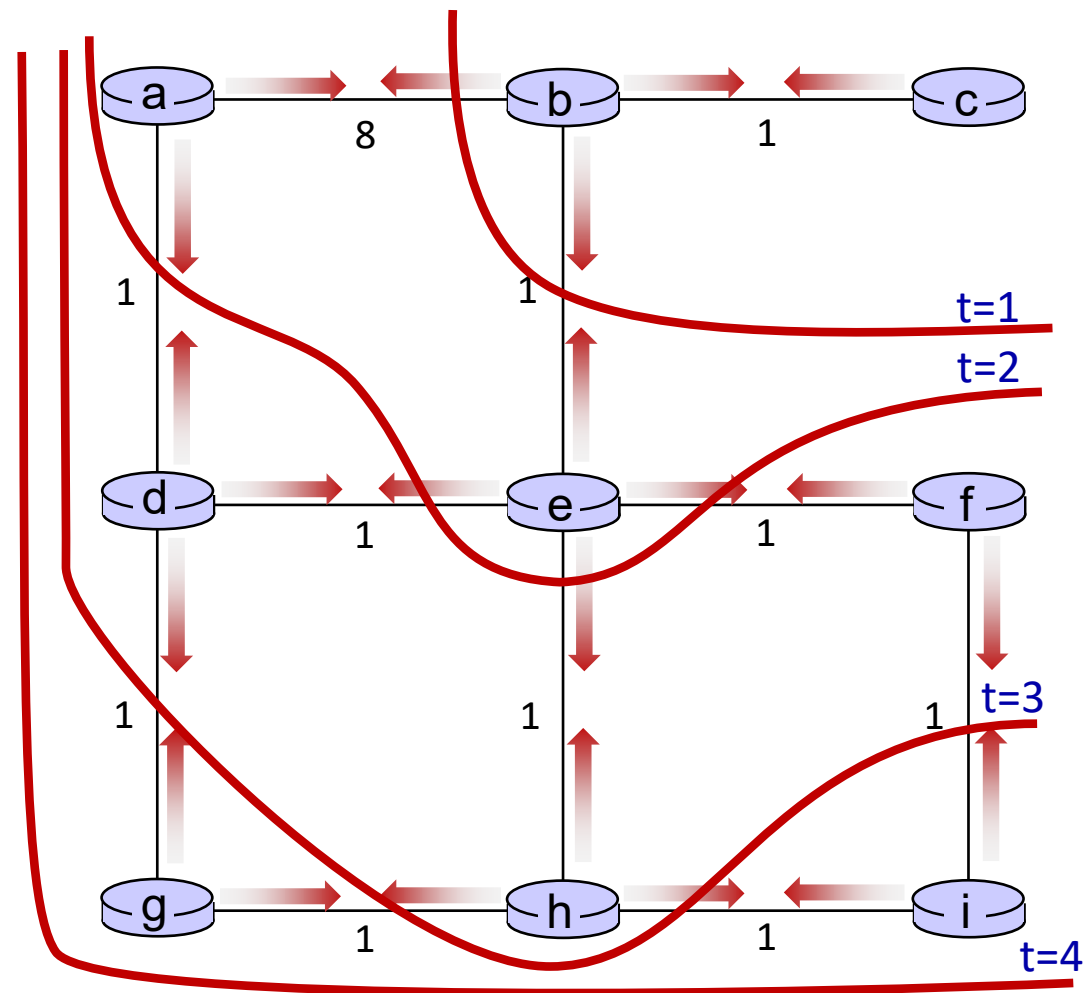
# Vettore di distanza: diffusione di informazioni sullo stato

La comunicazione iterativa, le fasi di calcolo diffondono le informazioni attraverso la rete:

-   $t=0$  lo stato di  $c$  a  $t=0$  è solo a  $c$
-   $t=1$  lo stato di  $c$  a  $t=0$  si è propagato a  $b$  e può influenzare i calcoli del vettore distanza fino a **1** hop di distanza, cioè a  $b$
-   $t=2$  lo stato di  $c$  a  $t=0$  può ora influenzare i calcoli del vettore distanza fino a **2** hop di distanza, cioè a  $b$  e ora anche a  $a, e$
-   $t=3$  lo stato di  $c$  a  $t=0$  può influenzare i calcoli del vettore distanza fino a **3** hop di distanza, cioè a  $d, f, h$
-   $t=4$  lo stato di  $c$  a  $t=0$  può influenzare i calcoli del vettore distanza fino a **4** hop di distanza, cioè a  $g, i$

Diametro della rete

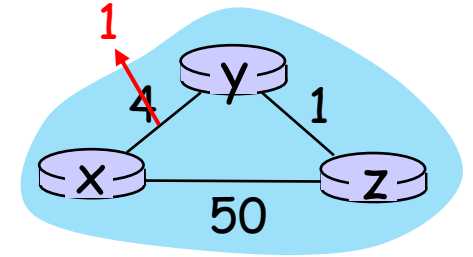
(massima lunghezza dei cammini tra tutti i nodi)



# Vettore delle distanze: cambiamento del costo dei collegamenti

## cambiamento del costo dei collegamenti:

- un nodo rileva la modifica del costo del collegamento locale
- aggiorna le informazioni di instradamento, ricalcola il DV locale
- se il DV cambia, avvisa i vicini



$t_0$ : y rileva la variazione del costo del collegamento, aggiorna il suo DV, informa i suoi vicini.

$t_1$ : z riceve l'aggiornamento da y, aggiorna la sua DV, calcola il nuovo costo minimo per x e invia ai suoi vicini la sua DV.

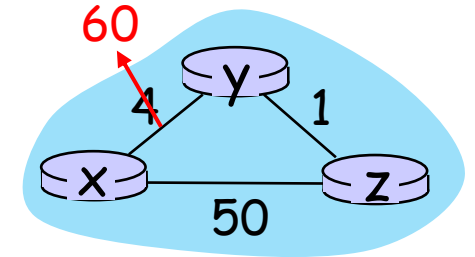
$t_2$ : y riceve l'aggiornamento di z e aggiorna il proprio DV. I costi minimi di y non cambiano, quindi y *non* invia un messaggio a z.

“Le buone notizie viaggiano velocemente”

# Vettore delle distanze: cambiamento del costo dei collegamenti

## cambiamento del costo dei collegamenti:

- un nodo rileva il cambiamento del costo del collegamento locale
- **“le cattive notizie viaggiano lentamente”** – problema di conteggio all'infinito

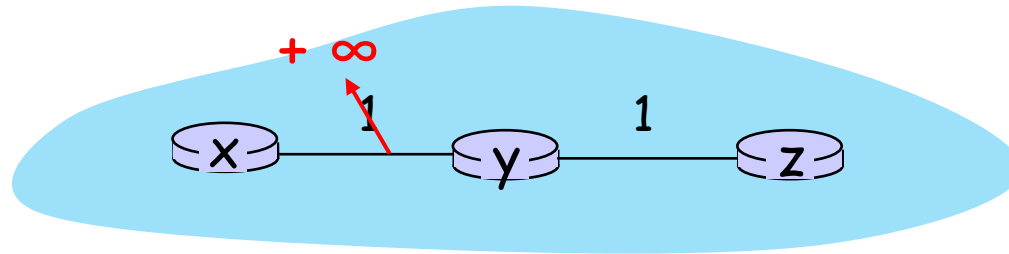


instradamento  
ciclico

- $t_0$ : y vede che il collegamento diretto con x ha un nuovo costo 60, ma z ha detto che ha un percorso di costo 5. Pertanto, y calcola “il mio nuovo costo verso x sarà 6, tramite z”; notifica z del nuovo costo 6 verso x.
  - $t_1$ : z apprende che il percorso verso x tramite y ha il nuovo costo 6, pertanto z calcola “il mio nuovo costo verso x sarà 7 tramite y”, notifica y del nuovo costo di 7 verso x.
  - $t_2$ : y apprende che il percorso verso x tramite z ha il nuovo costo 7, pertanto y calcola “il mio nuovo costo verso x sarà 8 tramite y”, notifica z del nuovo costo di 8 verso x.
  - $t_3$ : z apprende che il percorso verso x tramite y ha il nuovo costo 8, pertanto z calcola “il mio nuovo costo verso x sarà 9 tramite y”, notifica y del nuovo costo di 9 tramite x.
- ... così fino a  $t_{45}$  quando z non consideri il collegamento diretto con x più conveniente del percorso tramite y, rompendo l'istradamento ciclico e nell'iterazione successiva anche la stima della distanza di y converge

# Vettore delle distanze: conteggio all'infinito

Consideriamo questo caso in cui z instrada verso x tramite y



$t_0$ : il nodo y rileva che  $c_{yx} = +\infty$  (collegamento diretto interrotto); sapendo che  $D_z(x) = 2$ , y decide di instradare verso x tramite z e calcola  $D_y(x) = 3$ . y informa z del cambiamento del suo DV.

$t_1$ : il nodo z viene a sapere che  $D_y(x) = 3$ , quindi calcola il nuovo costo del percorso verso x tramite y  $D_z(x) = 4$  e lo comunica a y

$t_2$ : il nodo y viene a sapere che  $D_z(x) = 4$ , quindi calcola il nuovo costo del percorso verso x tramite z  $D_y(x) = 5$  e lo comunica a z

$t_3$ : il nodo z viene a sapere che  $D_y(x) = 5$ , quindi calcola che il nuovo costo del percorso verso x tramite y  $D_z(x) = 6$  e lo comunica a y

...

così via a causa dell'instradamento ciclico venutosi a formare.

# Vettore delle distanze: inversione avvelenata (*poisoned reverse*)

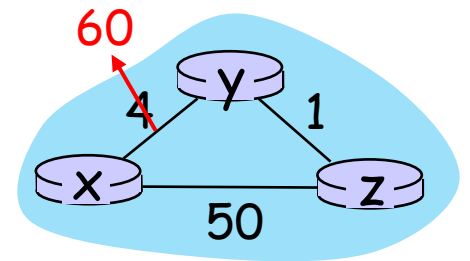
Se  $z$  instrada tramite  $y$  (cioè è il *next hop*) per giungere alla destinazione  $x$ , allora  $z$  avvertirà  $y$  che la sua distanza verso  $x$  è infinita, ossia  $z$  comunicherà a  $y$  che  $D_z(x) = +\infty$  (nonostante  $z$  sappia che  $D_z(x) = 5$ )

$t_0$ :  $y$  vede che il collegamento diretto con  $x$  ha un nuovo costo 60, ma continua a instradare attraverso il collegamento diretto perché per effetto dell'inversione avvelenata pensa che  $D_z(x) = +\infty$ ; informa  $z$  del nuovo costo del percorso verso  $x$

$t_1$ :  $z$  vede che il costo del percorso verso  $x$  tramite  $y$  è aumentato a 61, pertanto lo cambia in favore del collegamento diretto di costo 50; visto che ora non è più sul percorso verso  $x$  comunica a  $y$  che  $D_z(x) = 50$

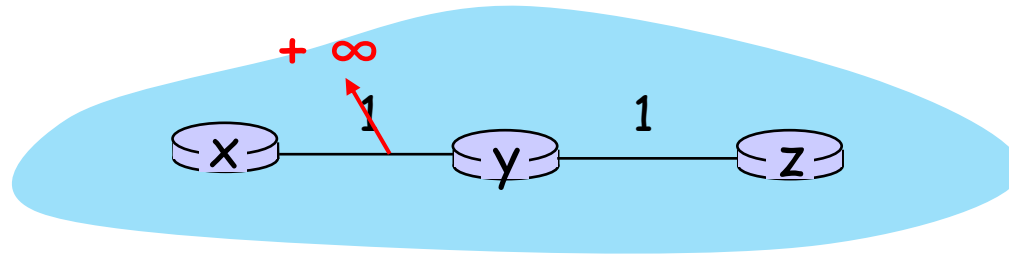
$t_2$ :  $y$  riceve l'aggiornamento da  $z$  e cambia l'instradamento verso  $x$  passando per  $z$ , pertanto  $D_y(x) = 51$ . Poiché  $z$  è ora sul percorso verso  $x$ ,  $y$  avvelena il percorso inverso inviando a  $z$   $D_y(x) = +\infty$

L'inversione avvelenata risolve il problema del conteggio all'infinito solo nel caso di cicli che riguardano nodi adiacenti!



# Vettore delle distanze: inversione avvelenata (*poisoned reverse*)

Ritorniamo sul questo caso in cui  $z$  instrada verso  $x$  tramite  $y$

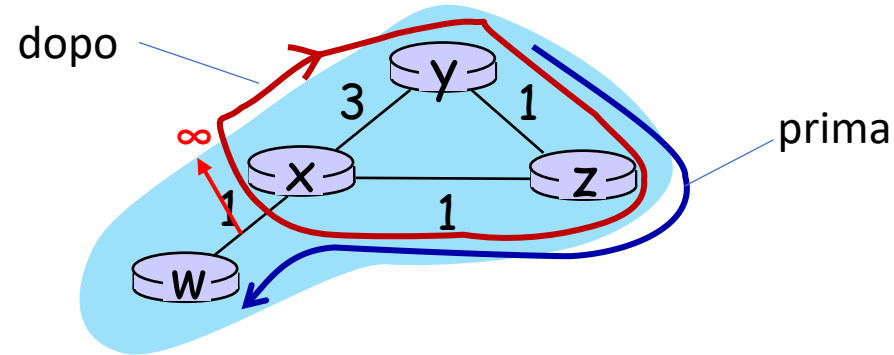


L'inversione avvelenata risolve anche questo scenario. Infatti,  $y$  ora crede che  $D_z(x) = +\infty$ , pertanto realizza subito che  $x$  non è più raggiungibile e quindi comunica a  $z$  che  $D_y(x) = +\infty$ . Ricevuta questa notifica,  $z$  realizza anch'esso che  $x$  è irraggiungibile, cioè  $D_z(x) = +\infty$ .

# Vettore delle distanze: inversione avvelenata (poisoned reverse)

Come abbiamo anticipato, l'inversione avvelenata non risolve tutti i casi di conteggio all'infinito.

Inizialmente,  $y$  instrada verso  $w$  tramite  $z$ , il quale instrada tramite  $x$ , che sfrutta un collegamento diretto con  $w$ .



- $x$  non può instradare tramite  $z$  (inversione avvelenata), ma decide di instradare tramite  $y$ , con costo  $D_x(w) = 3 + D_y(w) = 6$ , annuncia il nuovo costo
- $y$  riceve annuncio avvelenato da  $x$ , continua a instradare tramite  $z$
- $z$  riceve annuncio da  $x$ , calcola  $D_z(w) = 7$ , annuncia il nuovo costo
- $y$  riceve annuncio da  $z$ , calcola  $D_y(w) = 8$ , annuncia il nuovo costo
- $x$  riceve annuncio da  $y$ , calcola  $D_x(w) = 9$ , annuncia il nuovo costo
- $z$  riceve annuncio da  $x$ , calcola  $D_z(w) = 10$ , annuncia il nuovo costo

gli annunci si ripetono ciclicamente, preservando l'instradamento ciclico venutosi a creare, il cui costo però aumenta progressivamente.



# Vettore delle distanze: rappresentazione dell'infinito

Nella pratica il conteggio all'infinito viene interrotto rappresentando l'infinito con un valore (finito!) scelto preventivamente.

RIP (RFC 1058) usa costi unitari per i collegamenti diretti e 16 per la rappresentazione dell'infinito. A tal riguardo si dice:

Ora si dovrebbe capire perché “infinito” è stato scelto per essere il più piccolo possibile. Se una rete diventa completamente inaccessibile, vogliamo che il conteggio all'infinito venga interrotto il prima possibile. L'infinito deve essere abbastanza grande da impedire che un percorso reale sia così grande. Ma non dovrebbe essere più grande del necessario. La scelta dell'infinito è quindi un compromesso tra le dimensioni della rete e la velocità di convergenza nel caso in cui si verifichi il conteggio all'infinito. I progettisti di RIP ritenevano che il protocollo non fosse pratico per le reti con un diametro superiore a 15.

# Confronto tra gli instradamenti LS e DV

## Complessità dei messaggi

LS:  $n$  router,  $O(n^2)$  messaggi inviati

DV: scambio di messaggi tra router adiacenti; tempo necessario per la convergenza variabile

## Velocità di convergenza

LS: algoritmo  $O(n^2)$ , che richiede  $O(n^2)$  messaggi

- può avere oscillazioni

DV: può convergere molto lentamente

- può avere instradamenti ciclici
- problema del conteggio all'infinito

robustezza: che succede se un router si guasta o è compromesso?

LS:

- Un router può comunicare in broadcast un costo sbagliato per uno dei suoi *collegamenti*
- ciascun router calcola solo la *propria* tabella

DV:

- un router DV può comunicare *percorsi a costo minimo* errati (“ho un cammino di costo bassissimo verso qualsiasi destinazione”): *buco nero*
- Il DV di ciascun router è usato dagli altri: gli errori si propagano attraverso la rete