Reti di Elaboratori

Livello di Rete: Introduzione al control plane

Capitolo 5

SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Checco

alessandro.checco@uniroma1.it

Piano di controllo a livello di rete: obiettivi

- comprendere i principi alla base del piano di controllo della rete:
  - algoritmi di routing tradizionali
  - Controller SDN
  - gestione della rete, configurazione
- istanziazione e implementazione di:
  - OSPF, BGP
  - Controller OpenFlow, ODL e ONOS
  - SNMP, YANG/NETCONF

Livello di rete – piano di controllo: sommario

- introduzione
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento intra-ISP: OSPF
- instradamento tra ISP: BGP
- piano di controllo SDN
- gestione della rete, configurazione
  - SNMP
  - NETCONF/YANG

Funzioni del livello di rete

- forwarding (inoltro): sposta i pacchetti dall'input del router all'output del router appropriato

data plane

- routing (instradamento): determina il percorso seguito dai pacchetti dalla sorgente alla destinazione

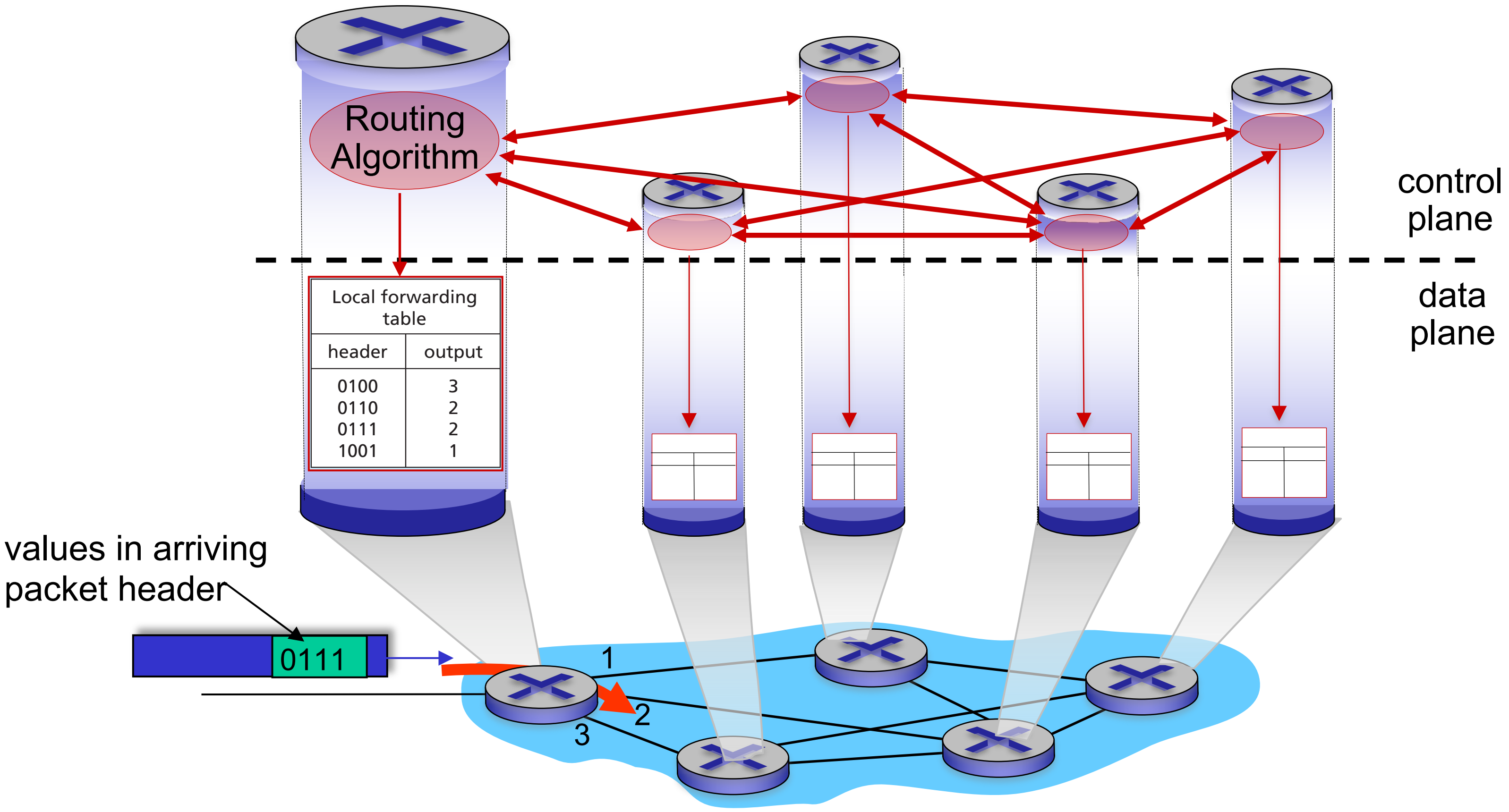
control plane

Due approcci alla strutturazione del piano di controllo della rete:

- tradizionale: controllo in ogni router
- SDN: controllo “centralizzato” (dal punto di vista logico)

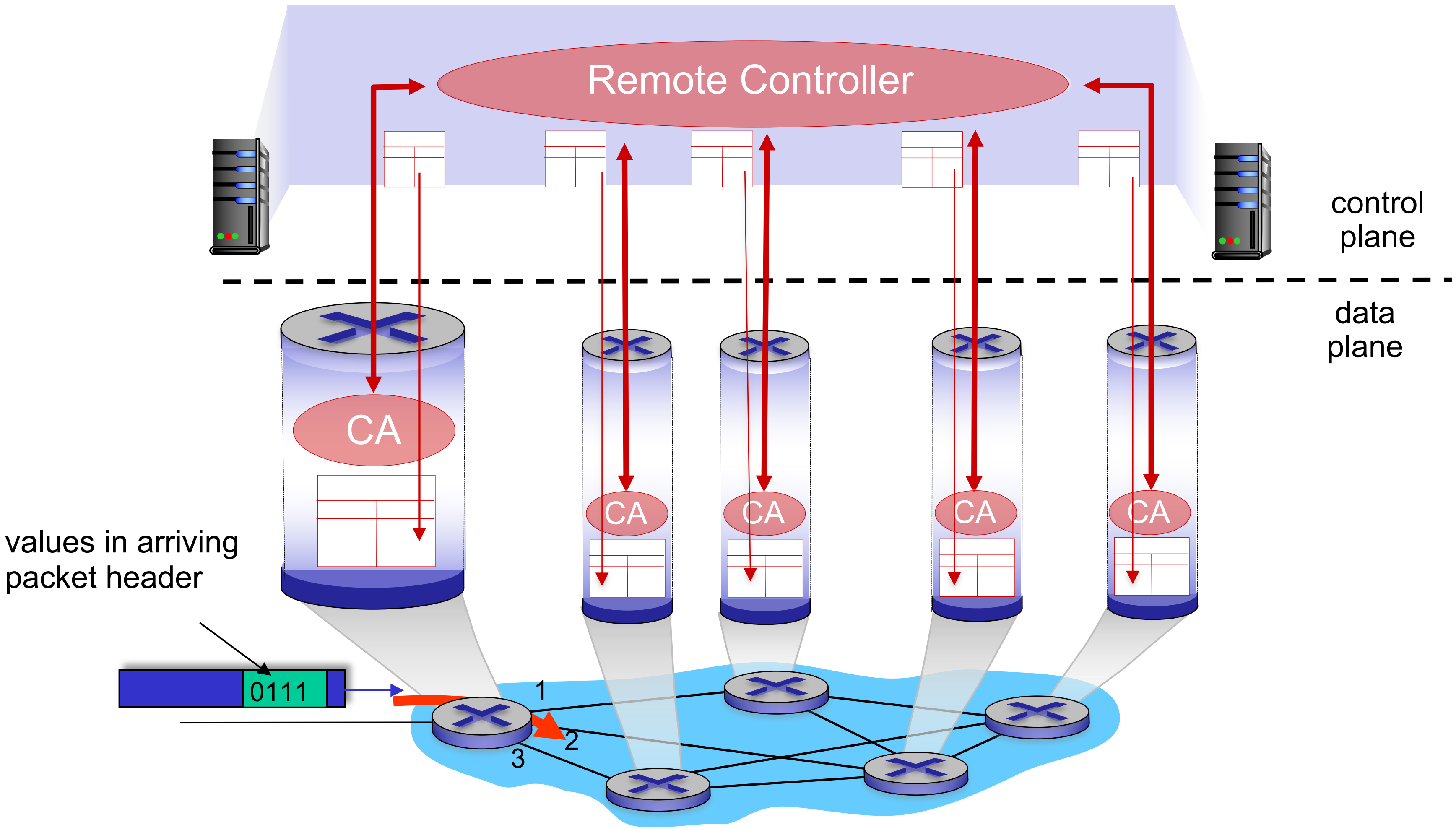
Per-router control plane

I singoli componenti dell'algoritmo di routing *in ogni singolo router* interagiscono nel piano di controllo

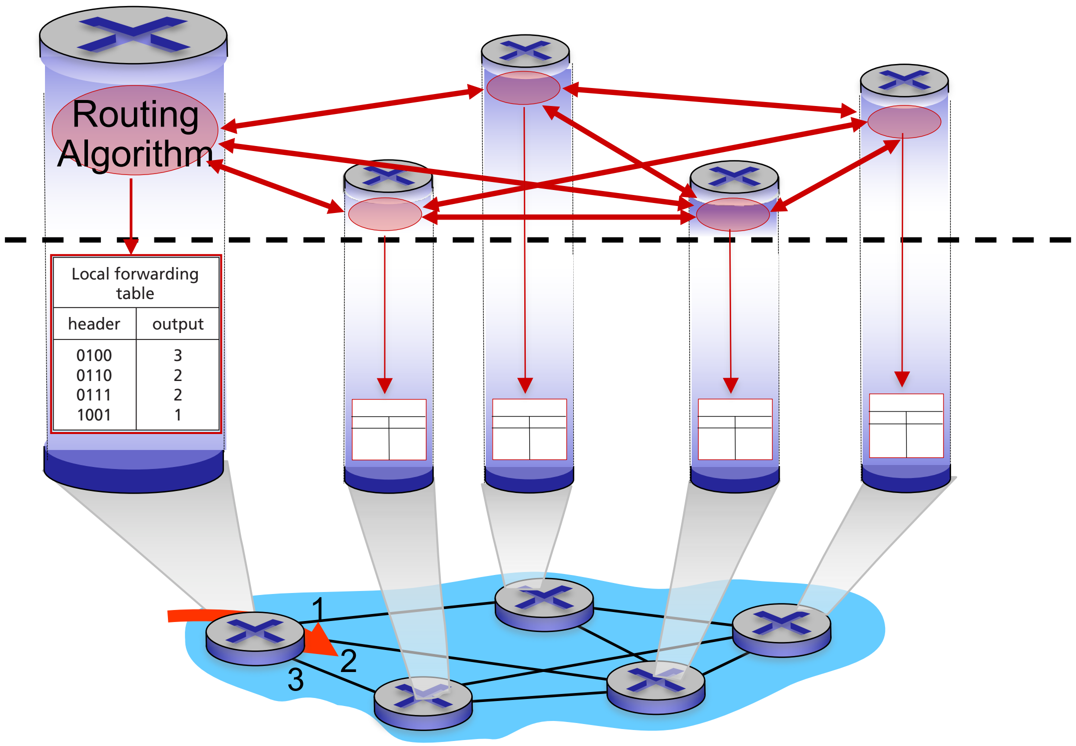


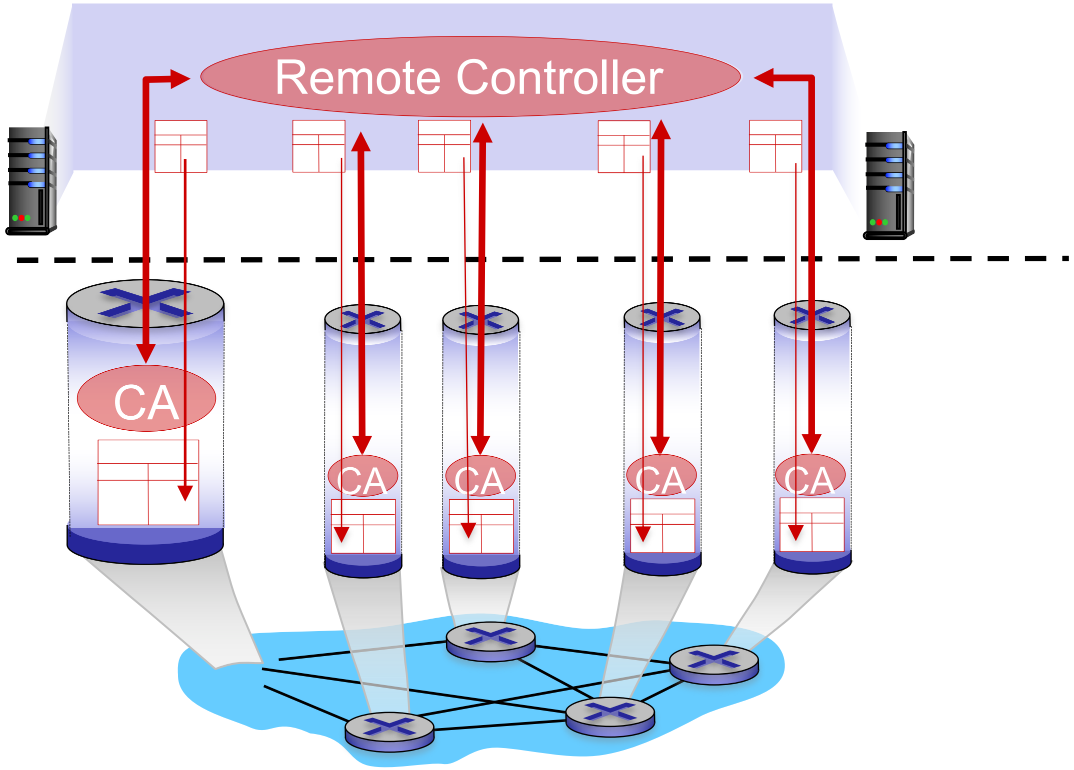
Piano di controllo Software-Defined Networking (SDN)

Il controller remoto calcola e installa le tabelle di inoltro nei router



Potenzialmente stesso algoritmo, diversa implementazione





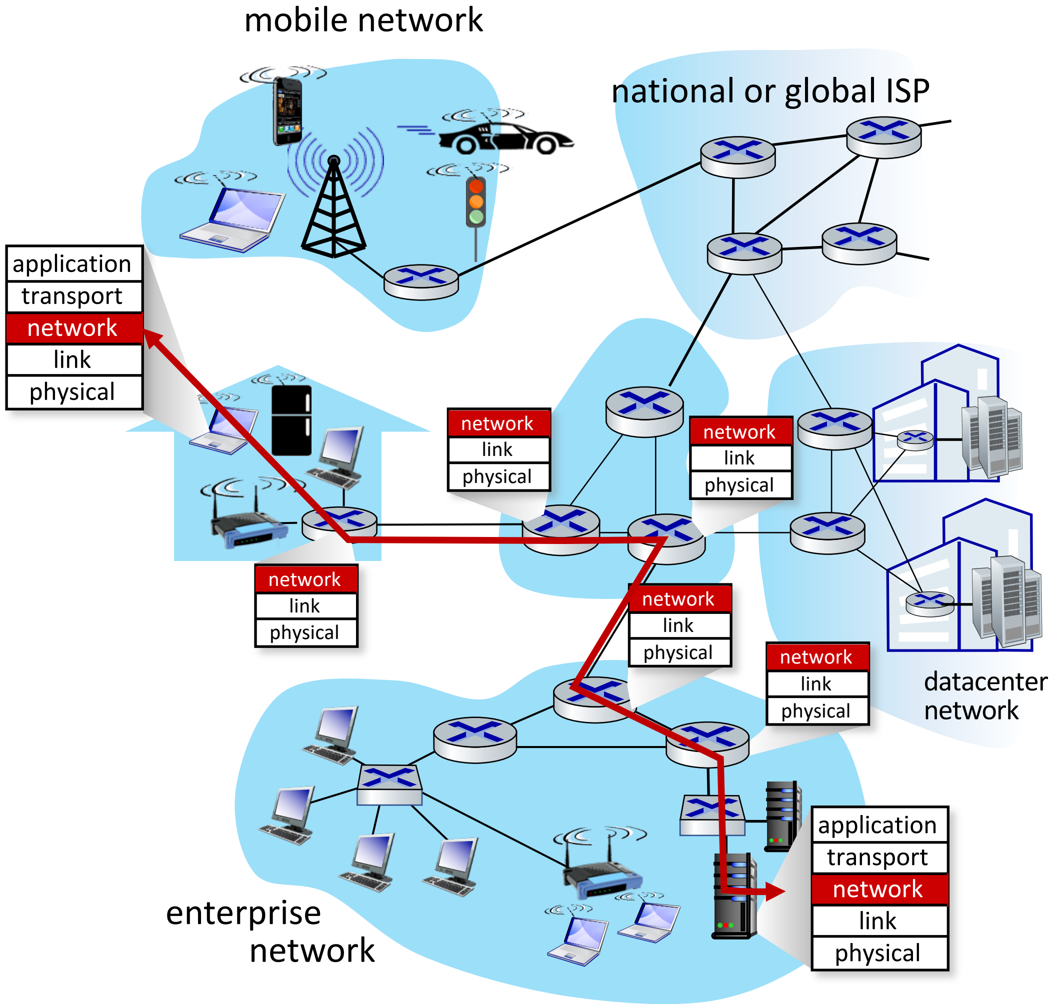
Livello di rete – piano di controllo: sommario

- introduzione
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento intra-ISP: OSPF
- instradamento tra ISP: BGP
- piano di controllo SDN
- gestione della rete, configurazione
  - SNMP
  - NETCONF/YANG

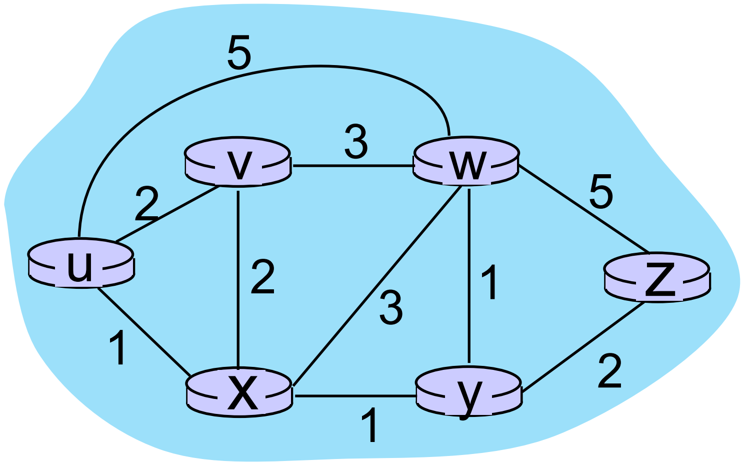
Protocolli di instradamento

Obiettivo del protocollo di instradamento: determinare percorsi (path o routes) "buoni", da sorgente a destinazione, attraverso la rete di router

- percorso: sequenza di router che i pacchetti attraversano da un host di origine all'host di destinazione
- “buoni”: minor “costo”, “più veloce”, “meno congestionato”
- routing: a “top-10” networking challenge!



Grafo: costo dei link



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

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

costo definito dall'operatore di rete: ad es. fisso a 1, o inversamente correlato alla larghezza di banda, o inversamente correlato alla congestione

grafo: $G = (N,E)$

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

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

Classificazione degli algoritmi di instradamento

globale: tutti i router conoscono completamente la topologia e il costo dei link

- algoritmi "link state".

Quanto velocemente cambiano?

statico: percorsi cambiano lentamente nel tempo

dinamico: i percorsi cambiano più rapidamente

Aggiornamenti periodici o in risposta a variazioni dei costi dei collegamenti

decentralizzato: scambio di informazioni tra router vicini che non conoscono l'intero stato della rete

- algoritmi "distance vector"

centralizzazione dell'informazione

Algoritmo di instradamento link-state di Dijkstra

- centralizzata: topologia di rete, costi di collegamento noti a *tutti i* nodi
  - realizzato tramite "link state broadcast"
  - tutti i nodi hanno le stesse informazioni
- calcola i percorsi a minor costo da un nodo ("sorgente") a tutti gli altri nodi
  - fornisce una *tabella di inoltro* per **quel nodo**
- iterativo: dopo *k* iterazioni, fornisce il percorso di costo minimo verso le *k* destinazioni più vicine

notazione

- $c_{x,y}$: costo del link diretto tra *a* e *b*; è ∞ se non sono vicini
- *D(v):* *stima corrente* del costo del percorso a minor costo dalla sorgente alla destinazione *v*
- *p(v):* nodo predecessore lungo il percorso dalla sorgente a v
- *N':* insieme di nodi il cui percorso meno costoso *è definitivamente* noto

Algoritmo di instradamento link-state di Dijkstra

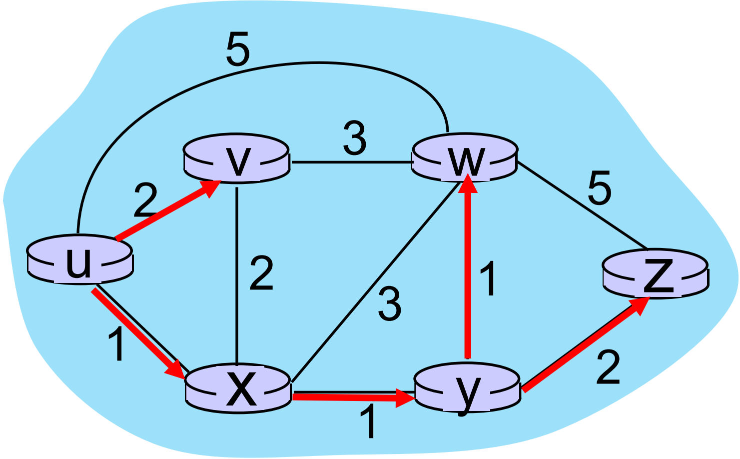
Initialization:
  $N' = \{u\}$ /* compute least cost path from u to all other nodes */
  for all nodes v
    if v adjacent to u /* u initially knows direct-path-cost only to direct neighbors */
      then $D(v) = c_{u,v}$ /* but may not be *minimum* cost! */
    else $D(v) = \infty$

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 least-path-cost to v is either old least-cost-path to v or known
  least-cost-path to w plus direct-cost from w to v */
until all nodes in N'

Algoritmo di Dijkstra: 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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

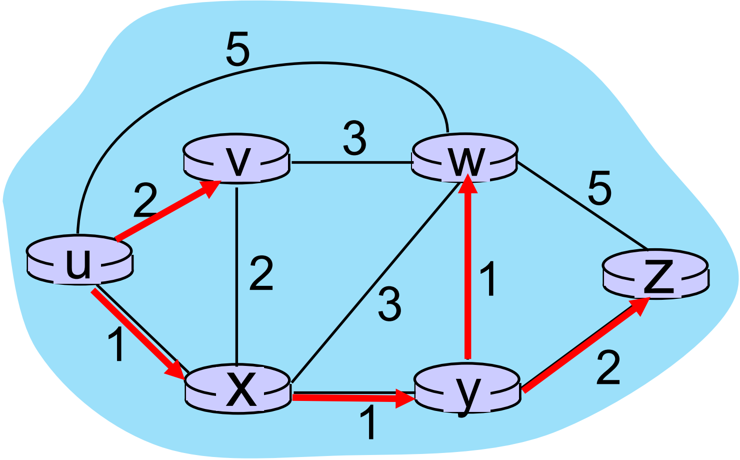
Initialization (step 0): For all a: if *a* adjacent to u then $D(a) = c_{u,a}$



find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

Algoritmo di Dijkstra: esempio



percorso a minor costo da *u*

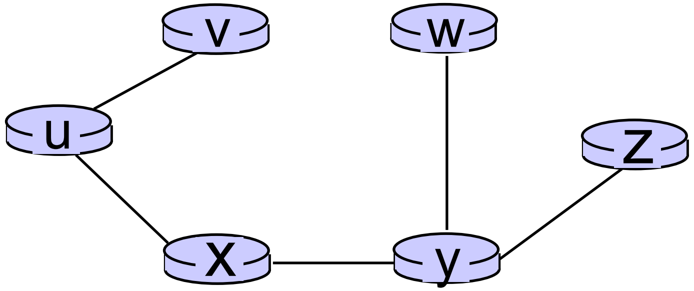


tabella di inoltro:

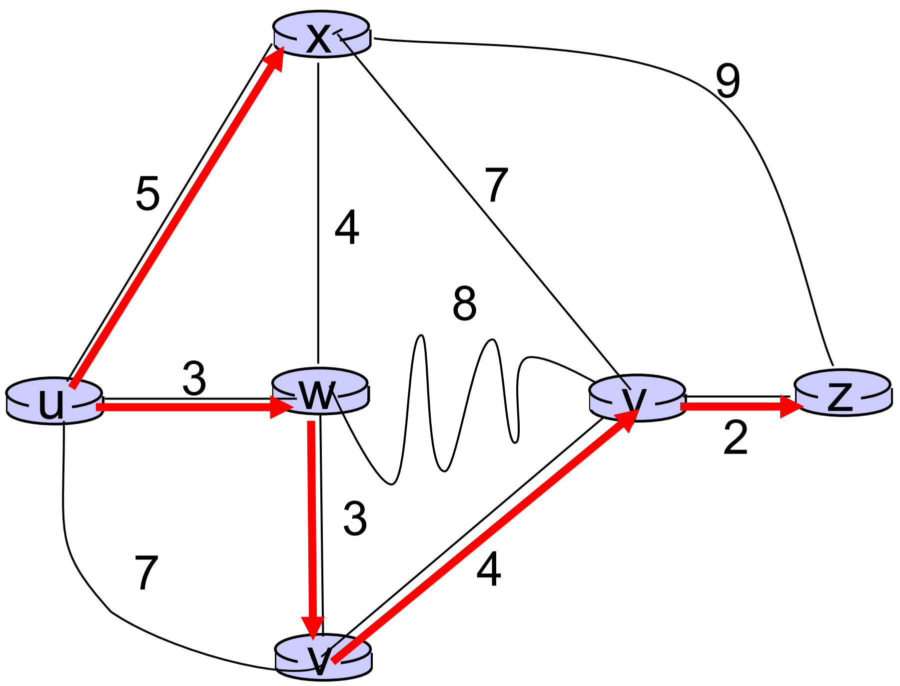
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

percorso diretto da *u* a *v*

percorso da u a tutte le altre destinazioni via *x*

Algoritmo di Dijkstra: un altro 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	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



note:

- costruire l'albero del percorso di costo minimo tracciando i nodi predecessori
- può essere necessario scegliere tra costi uguali (tie break)

Algoritmo di Dijkstra: discussione

complessità dell'algoritmo: con *n* nodi

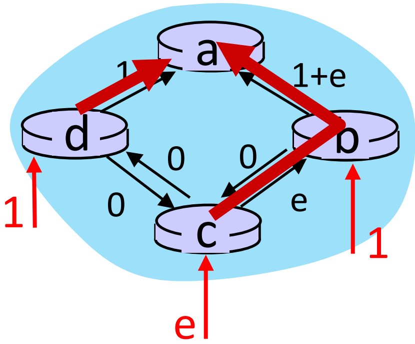
- per ciascuna delle *n* iterazioni è necessario controllare tutti i nodi *w* non in *N*
- $n(n+1)/2$ confronti: complessità $O(n^2)$
- possibili implementazioni più efficienti: $O(n \log n)$

complessità di comunicazione (traffico):

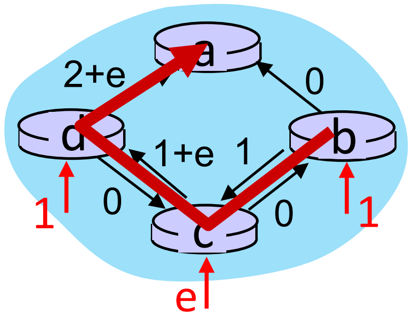
- ogni router deve trasmettere (*broadcast)* il suo stato dei costi (link state) a tutti gli altri router
- esistono algoritmi di broadcast intelligenti: bastano $O(n)$ attraversamenti di link per disseminare un messaggio broadcast da una sorgente
- il messaggio di ogni router attraversa $O(n)$ link: complessità complessiva di traffico: $O(n^2)$

Caso patologico: oscillazioni

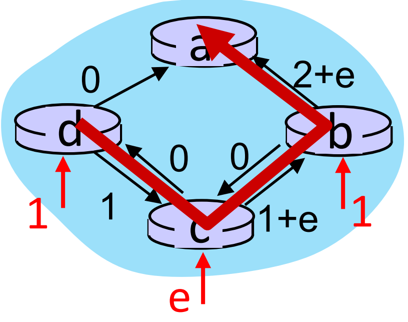
- quando i costi di collegamento dipendono dal volume di traffico, sono possibili **oscillazioni** del percorso
- scenario di esempio:
  - routing verso a, traffico entrante da d (1), c(e<<1), b(1)
  - il costo dei link dipende dal traffico (archi diretti = non simmetrici)



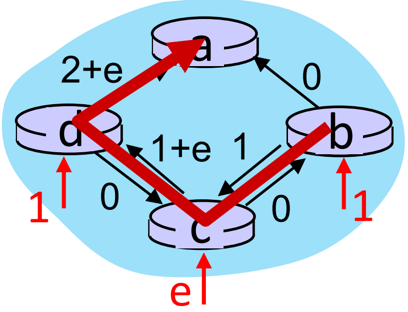
initially



given these costs,
find new routing….
resulting in new costs



given these costs,
find new routing….
resulting in new costs



given these costs,
find new routing….
resulting in new costs

Livello di rete – piano di controllo: sommario

- introduzione
- algoritmi di instradamento
  - link state
  - distance vector
- instradamento intra-ISP: OSPF
- instradamento tra ISP: BGP
- piano di controllo SDN
- gestione della rete, configurazione
  - SNMP
  - NETCONF/YANG

Distance vector algorithm

Basata sull'equazione di *Bellman-Ford* (BF):

Bellman-Ford equation

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

Allora:

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

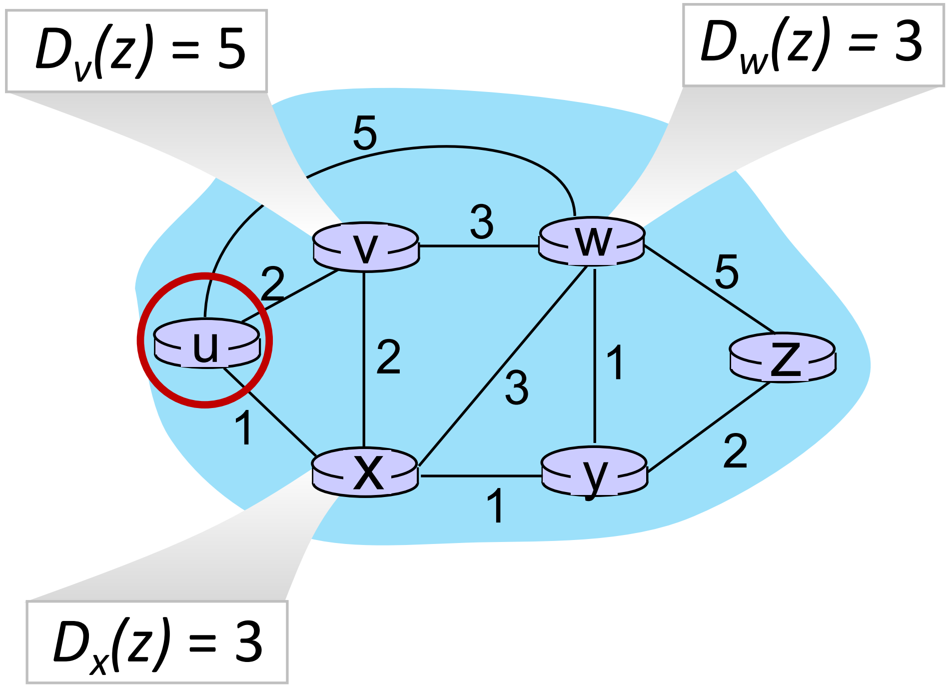
stima del percorso di minor costo da v a y

min su tutti i vicini *v di x*

costo diretto da x a v

Bellman-Ford: esempio

Supponiamo che i vicini di *u: x,v,w* sappiano che per la destinazione *z*:



Bellman-Ford:

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

l'argmin (x) definisce il percorso u->x e il costo stimato 4 per raggiungere z

Algoritmo distance vector

idea chiave:

- di tanto in tanto (vedi dopo), ogni nodo invia la propria stima del vettore di distanza ai vicini
- quando *x* riceve una nuova stima DV da qualsiasi vicino, aggiorna il proprio DV utilizzando l'equazione BF:

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

- sotto certe condizioni ragionevoli, la stima $D_x(y)$ *converge al minimo costo effettivo* per raggiungere y

Algoritmo distance vector:

ogni nodo:

attende per (cambio del costo diretto locale o messaggio da un vicino)

↓

ricalcola le stima del DV per ogni y usando i nuovi valori

↓

se DV a qualsiasi destinazione è cambiato, *avvisa* i vicini

iterativo, asincrono: ogni iterazione locale causata da:

- cambio del costo del collegamento locale
- messaggio di aggiornamento DV dal vicino

distribuito, self-stopping e responsive: ogni nodo avvisa i vicini *solo* quando il suo DV cambia

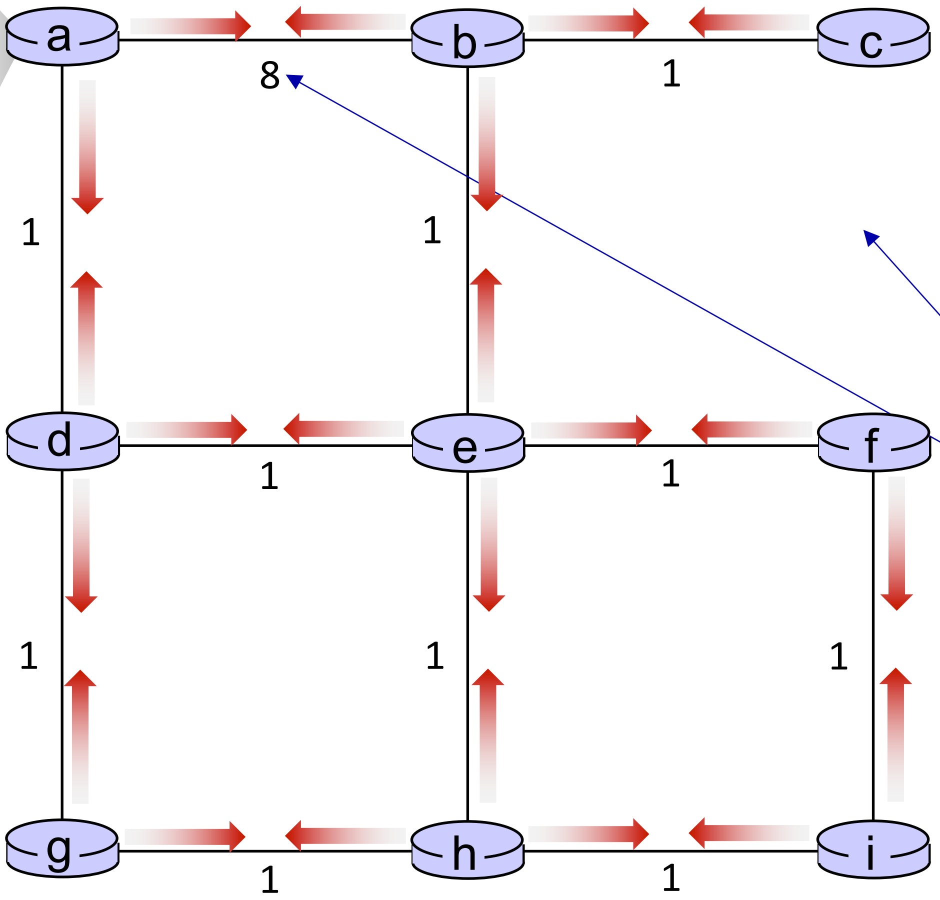
- i vicini quindi avvisano i loro vicini, *solo se necessario*
- se nessuna notifica ricevuta, nessuna azione intrapresa!

Distance vector: esempio

t=0

- Ogni nodo conosce (solo) la stima della distanza dai vicini
- Ogni nodo comunica la propria distanza ai vicini

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

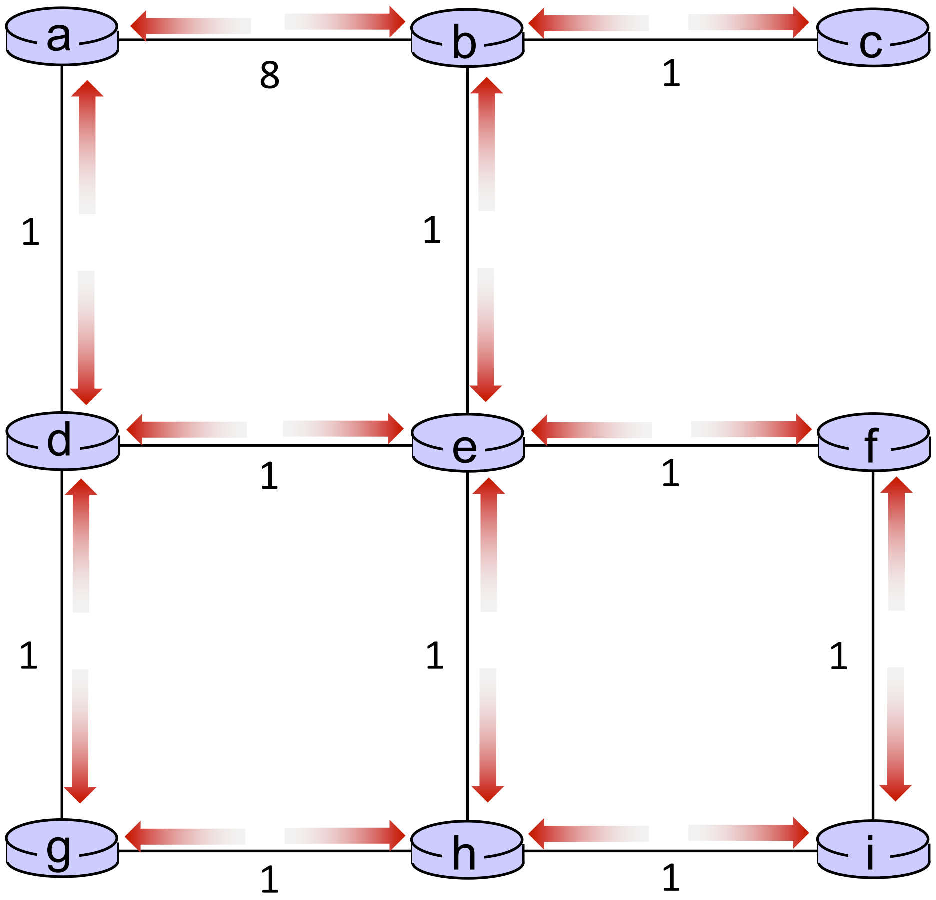


piccole differenze nella griglia
- missing link
- larger cost

Distance vector: iterazione

ogni nodo:

- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)

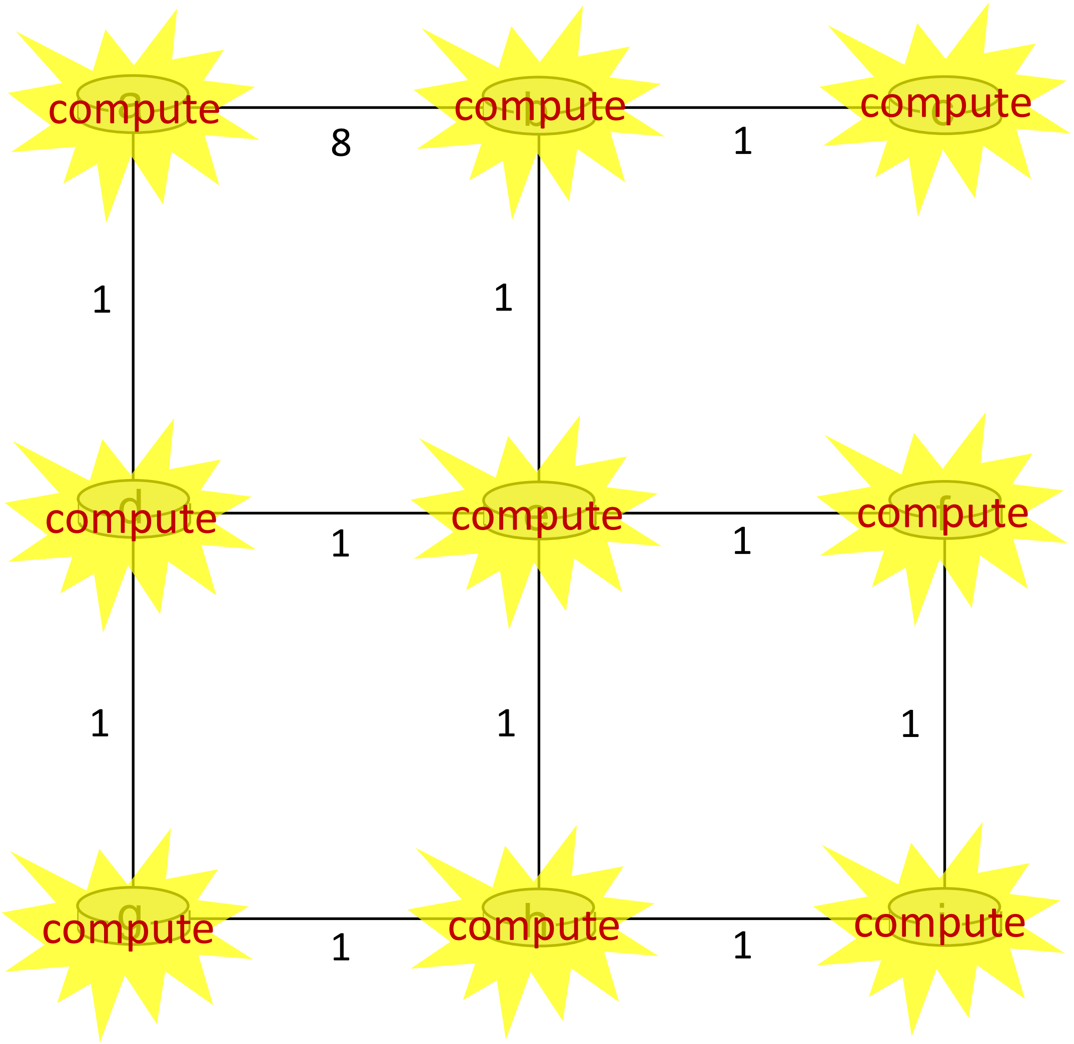


Distance vector: iterazione

t=1

ogni nodo:

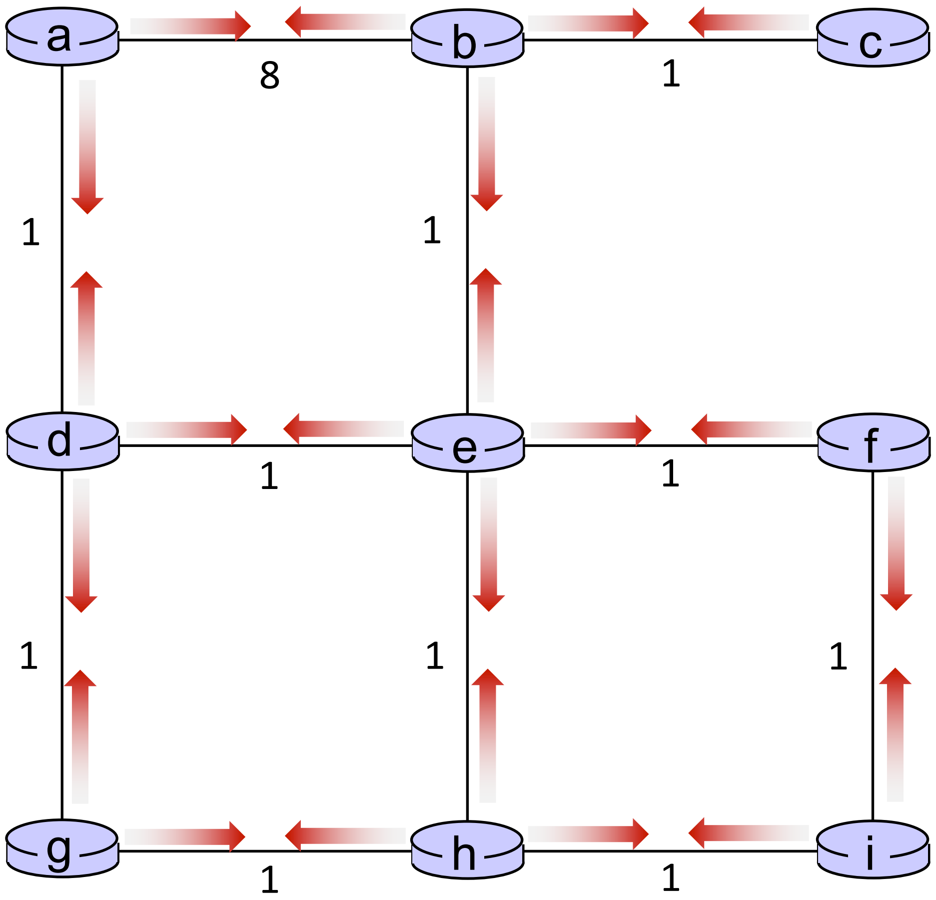
- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)



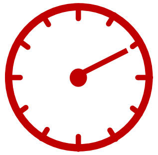
Distance vector: iterazione

t=1

ogni nodo:
- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)



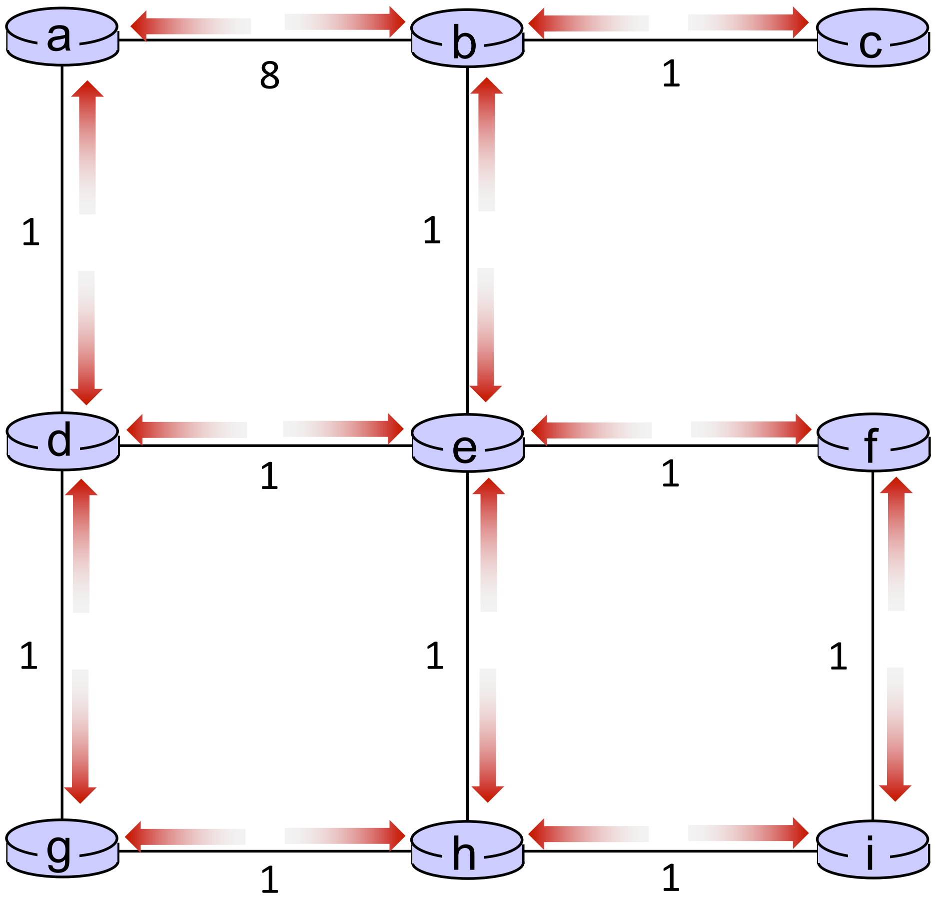
Distance vector: iterazione



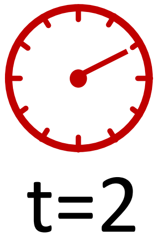
t=2

ogni nodo:

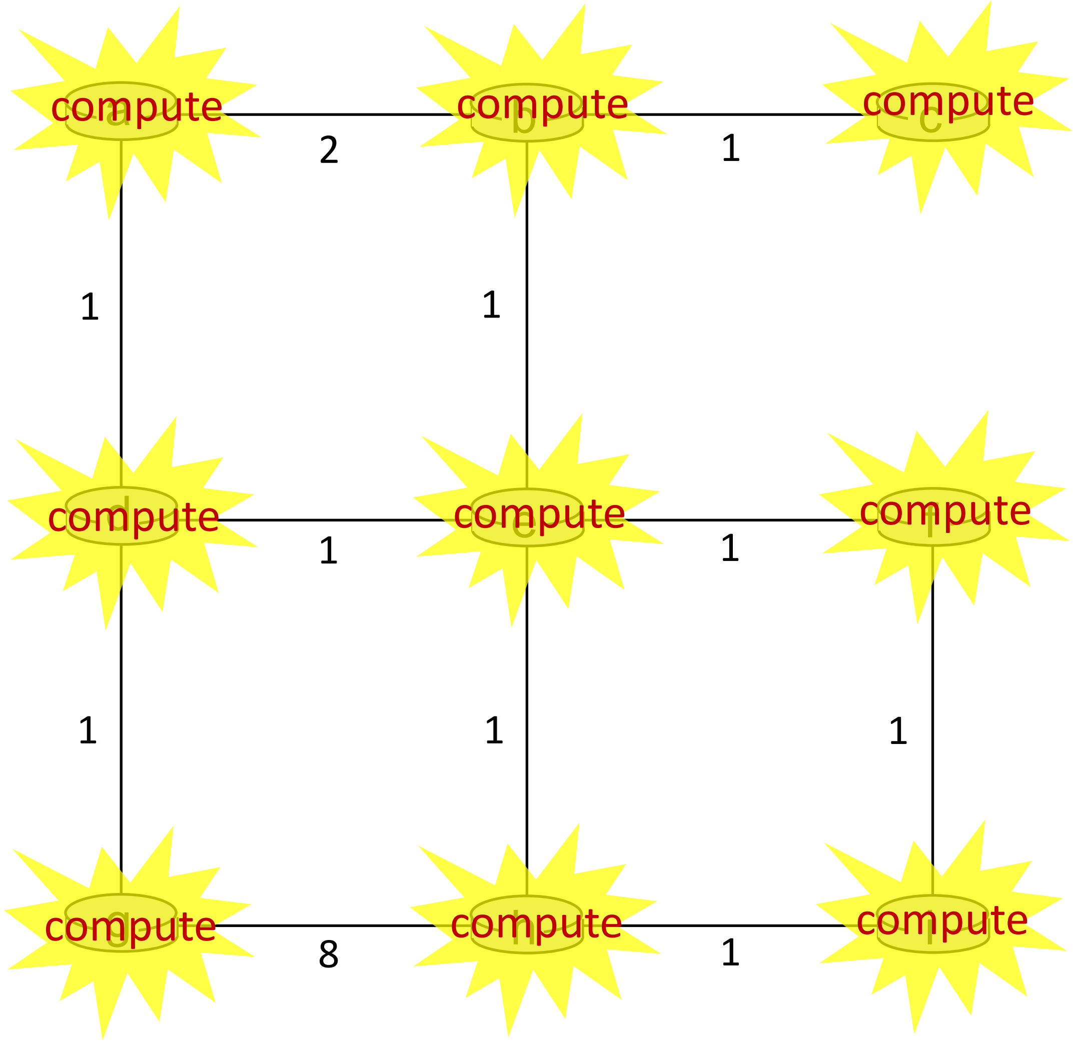
- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)



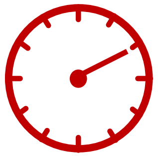
Distance vector: iterazione



ogni nodo:
- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)



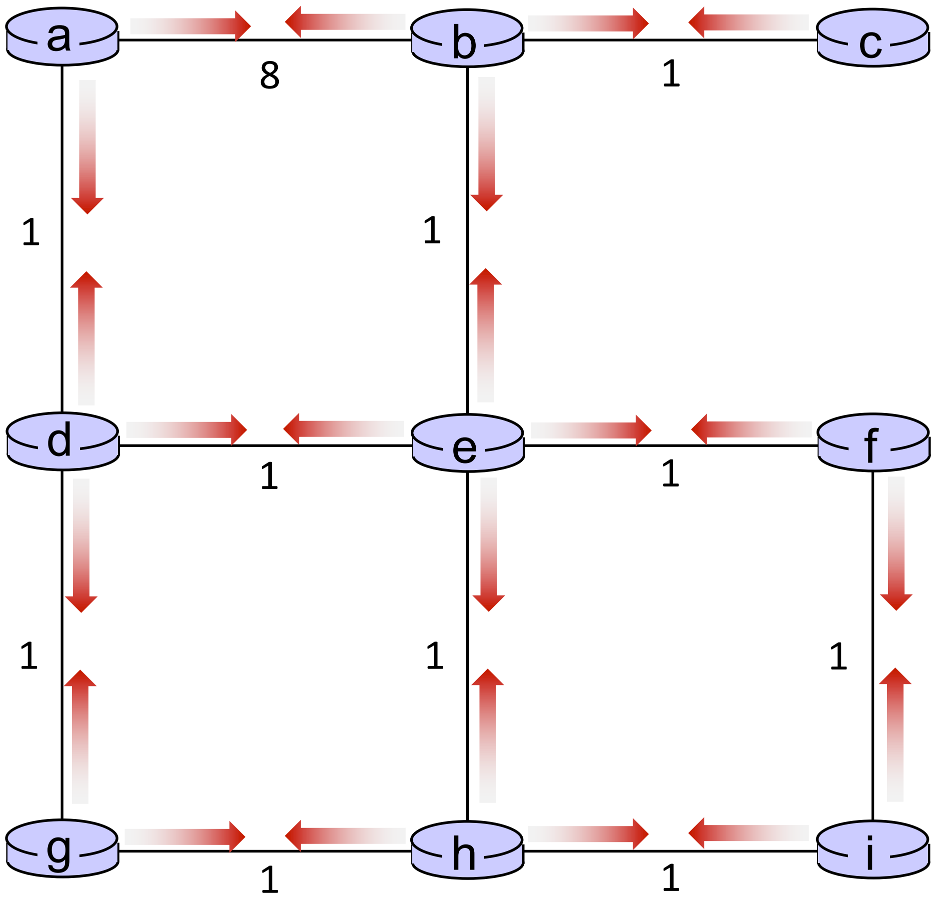
Distance vector: iterazione



t=2

ogni nodo:

- riceve i distance vector dai vicini
- calcola il proprio nuovo vettore locale
- manda il nuovo vettore ai vicini (se è cambiato)

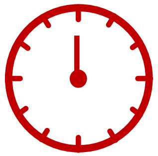


Distance vector: iterazione

…. e così via

vediamo nel dettaglio i calcoli all'interno dei nodi

Distance vector: esempio



t=1

- b receives DVs from a, c, e

DV in a:

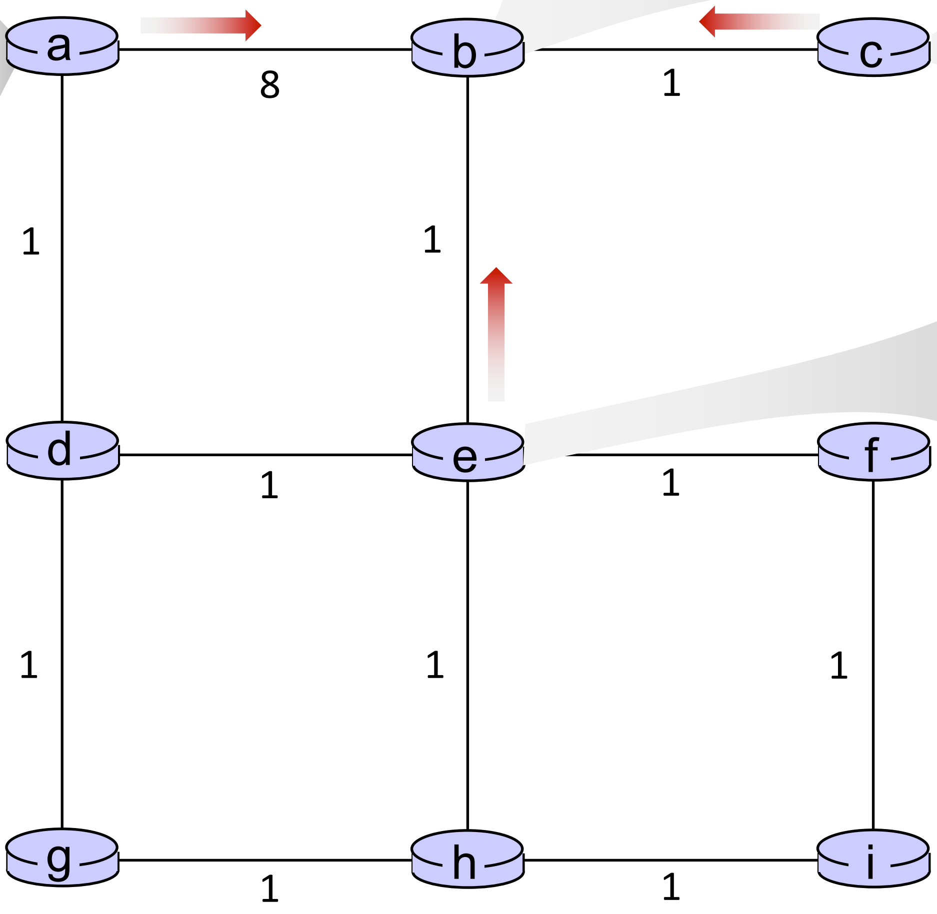
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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:

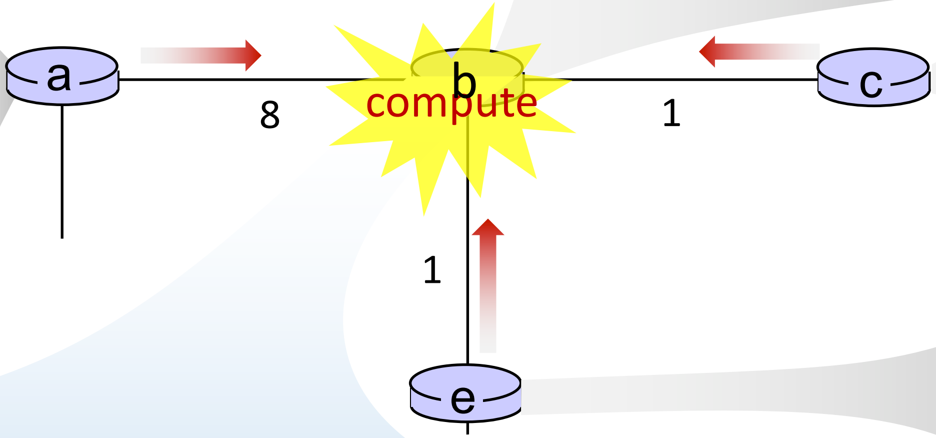
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

Distance vector: esempio

t=1

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

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

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

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

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

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

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

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

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

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

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

Distance vector: esempio

t=1

- c receives DVs from b

DV in a:

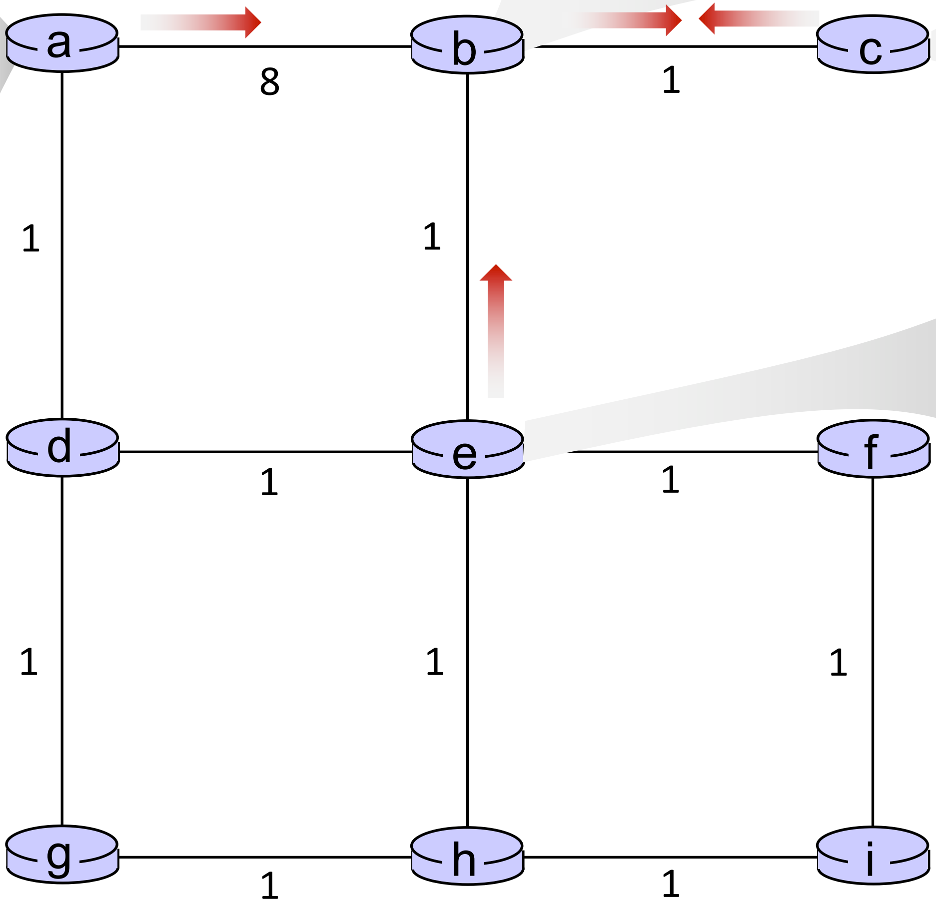
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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:

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

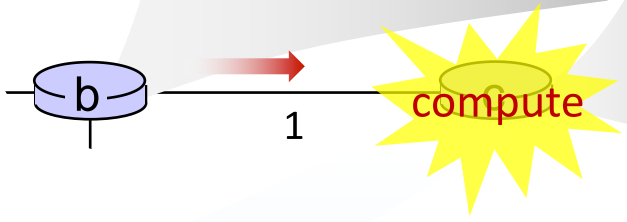
Distance vector: esempio

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:

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



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:

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

Distance vector: esempio

t=1

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

DV in b:

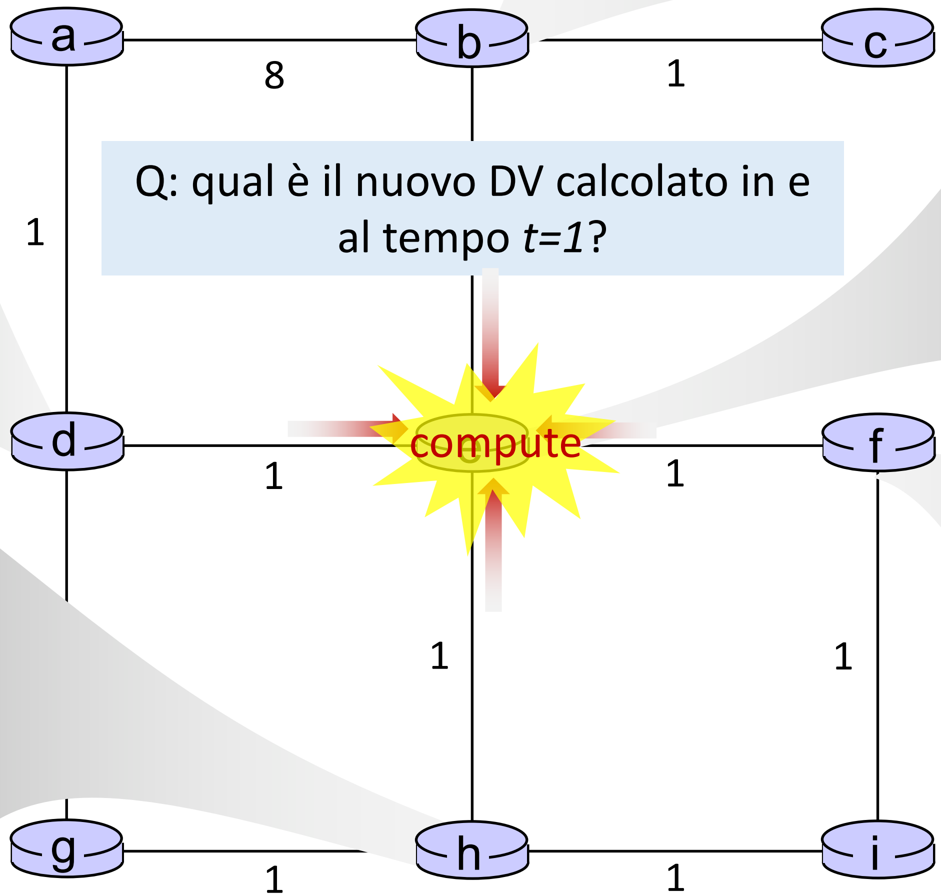
$D_b(a) = 8$ $D_b(f) = \infty$
$D_b(c) = 1$ $D_b(g) = \infty$
$D_b(d) = \infty$ $D_b(h) = \infty$
$D_b(e) = 1$ $D_b(i) = \infty$

DV in 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 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 h:

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

DV in f:

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

Distance vector: propagazione dell'informazione di stato

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

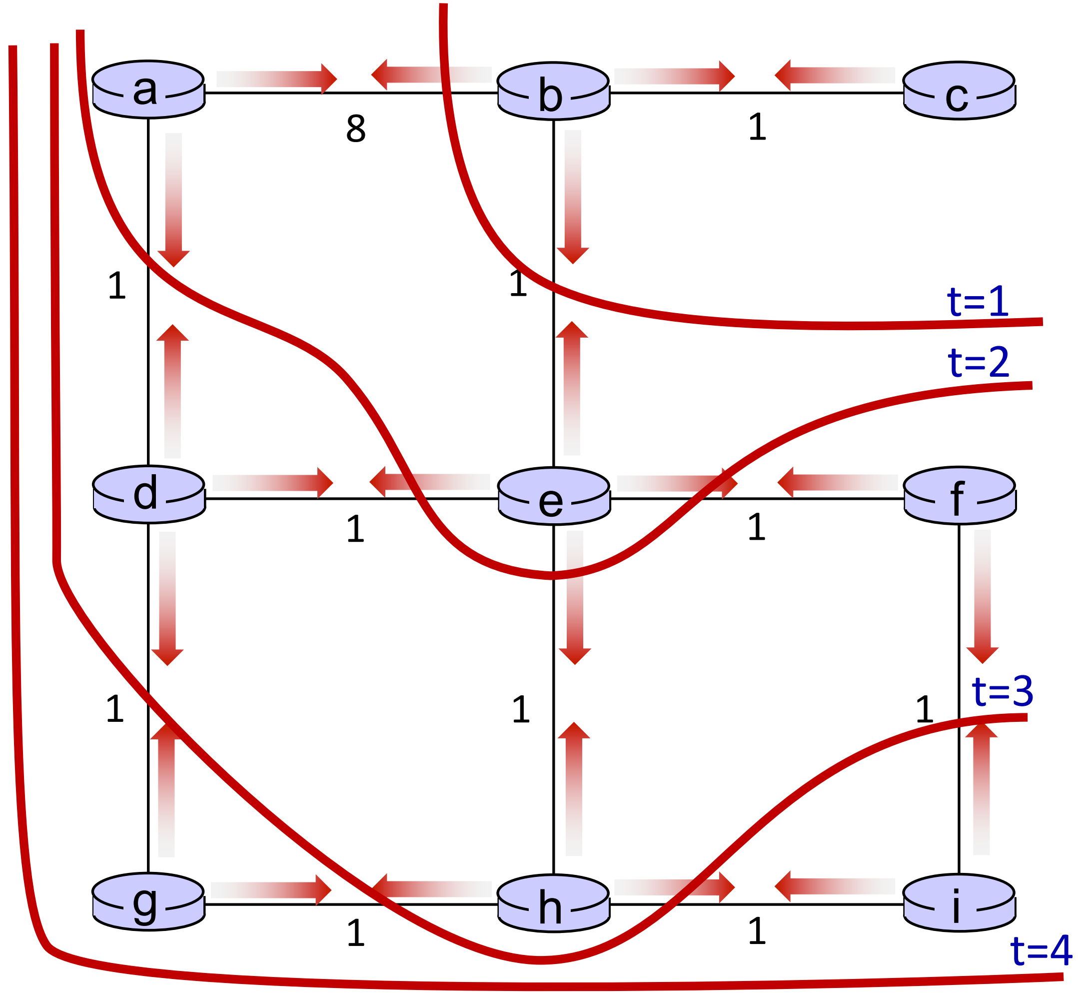
t=0 c's state at t=0 is at c only

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

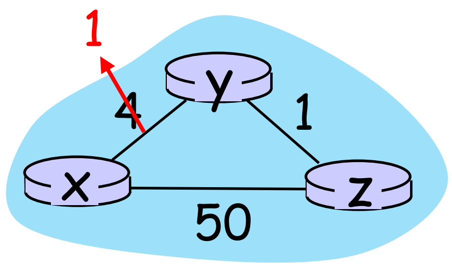
t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well

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: cambio del costo locale



costo diretto del link cambia:

- il nodo rileva la variazione del costo del link locale
- aggiorna le informazioni di routing, ricalcola il DV locale
- se DV cambia, avvisa i vicini

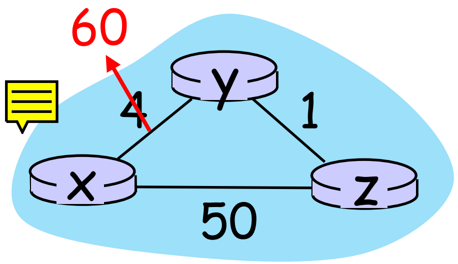
"le buone notizie viaggiano veloci"

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: cambio del costo locale



costo diretto del link cambia:

- node detects local link cost change
- "le cattive notizie viaggiano lentamente" – problema di velocità del conteggio all'infinito (se link sparisce ad esempio)

  - *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.*
    …

- *Gli algoritmi distribuiti possono avere comportamenti patologici*

Conteggio all'infinito: soluzioni

Split horizon

- Invece di inviare la tabella attraverso ogni interfaccia, ciascun nodo invia solo una parte della sua tabella tramite le interfacce
- Il nodo y omette nell'advertising ad x, informazioni apprese proprio da x (L'informazione è arrivata da x e quindi la conosce già)
- Nell'esempio y elimina la riga di x dalla tabella prima di inviarla ad x
- Cerca di prevenire rotte cicliche evitando di mandare indietro informazioni stantie

Poisoned reverse (inversione avvelenata)

- Si pone a ∞ il valore del costo del percorso che passa attraverso il vicino a cui si sta inviando il vettore
- Nell'esempio y pone a ∞ il costo verso x quando invia il vettore ad x
- Serve a propagare l'informazione (negativa) su una rotta quando diventa ciclica

Confronto tra Link State e Distance Vector

complessità di messaggio

LS: *n* router, O(n^2) messaggi

DV: propagazione del messaggio O(n)

velocità di convergenza

LS: O(n^2) computazioni

- può avere oscillazioni (dipende dal tipo dicosto)

DV: dipende

- può avere instradamento ciclico
- conteggio all'infinito

robustezza: che succede in caso di malfunzionamento o attacco a un router?

LS:

- il router può pubblicizzare un costo di *link* errato
- ogni router calcola solo la propria tabella

DV:

- Il router DV può pubblicizzare un costo di *percorso errato* ("Ho un *percorso* a basso costo per ovunque"): black-holing
- ogni router table viene utilizzata da altri: l'errore si propaga nella rete