

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

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

Livello di rete: piano di controllo (parte1)

dr. Manuel Fiorelli

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

Introduction: 1-1

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
- istanziazione, 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

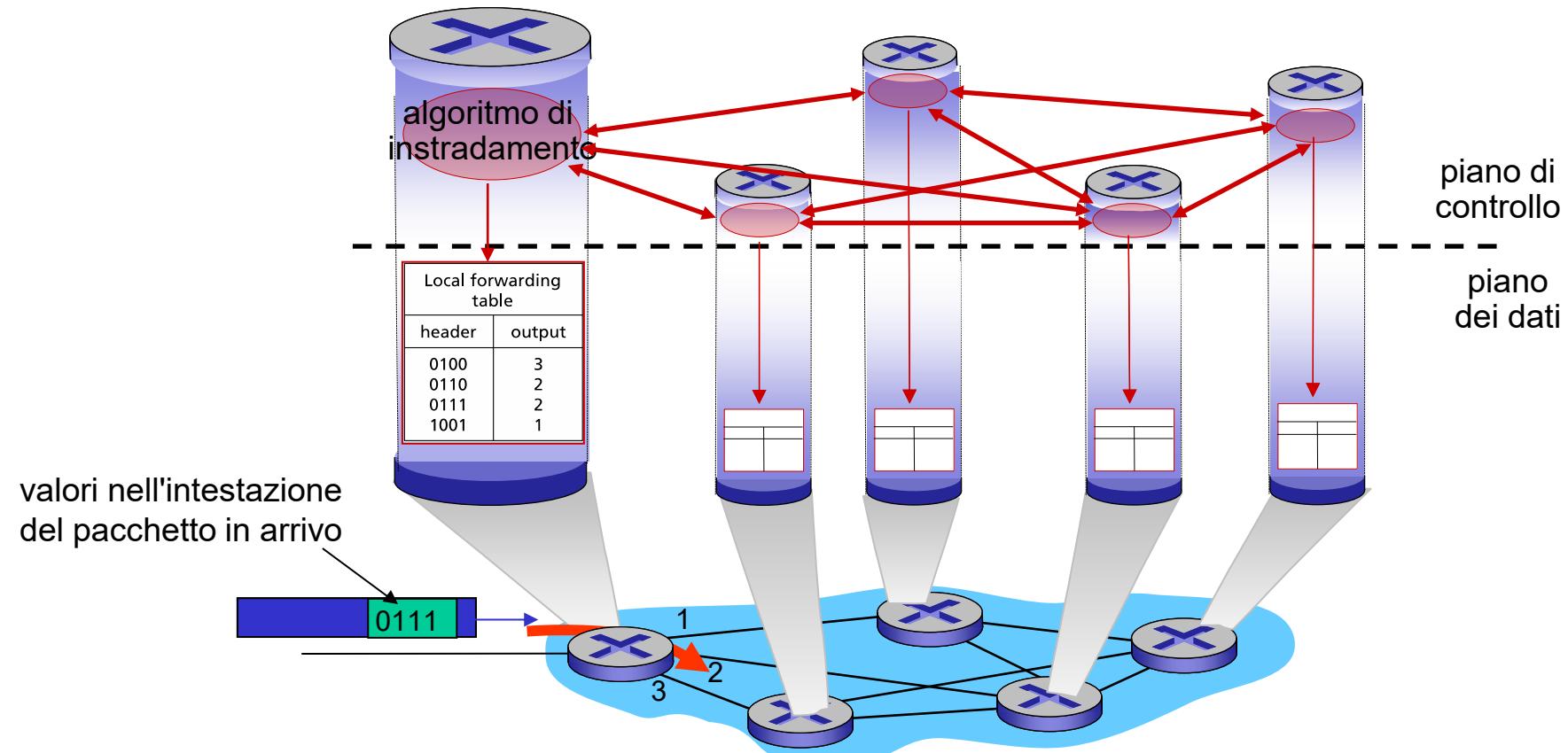
piano di controllo

Due approcci alla strutturazione del piano di controllo della rete:

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

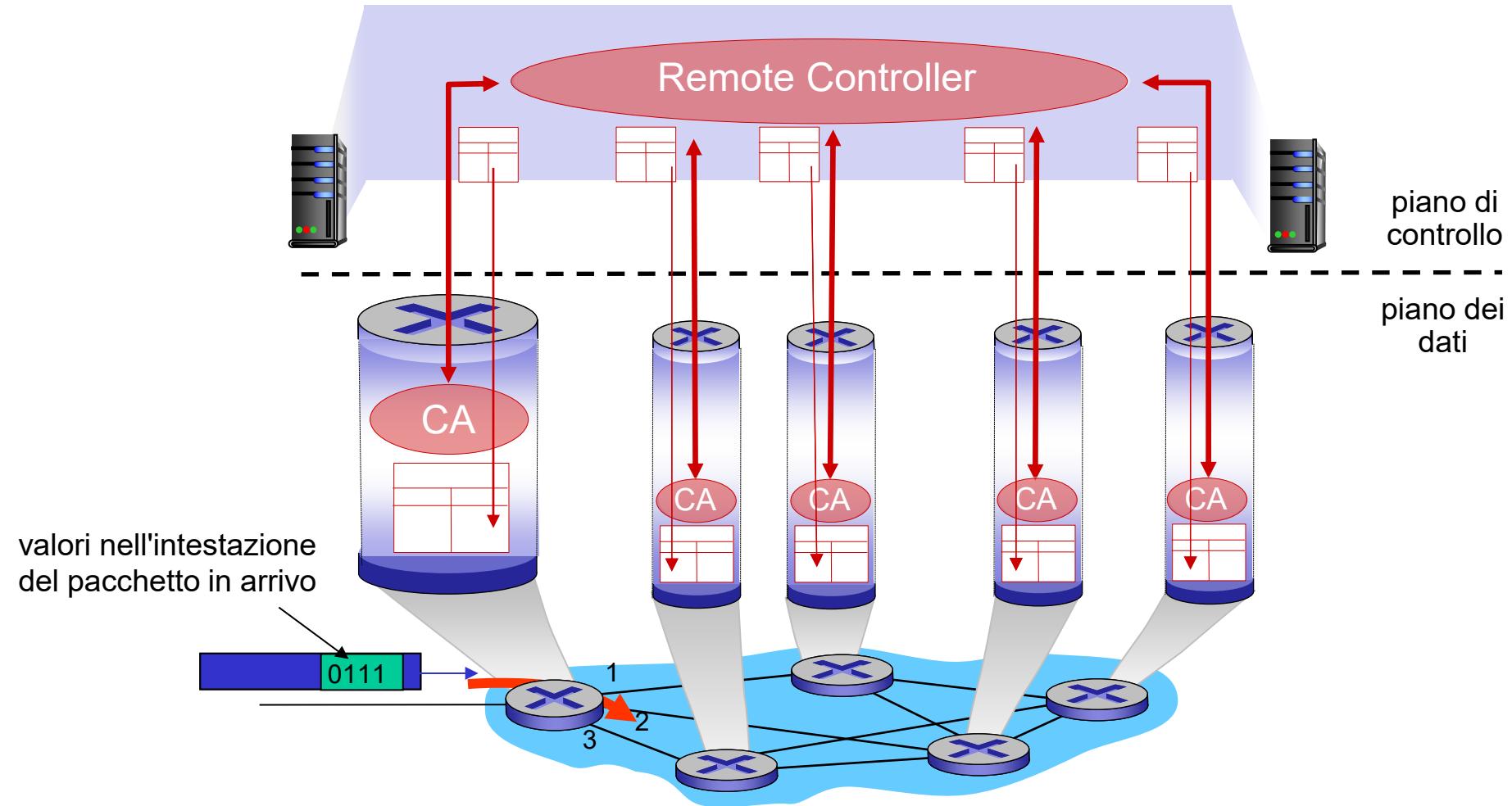
Piano di controllo per router

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

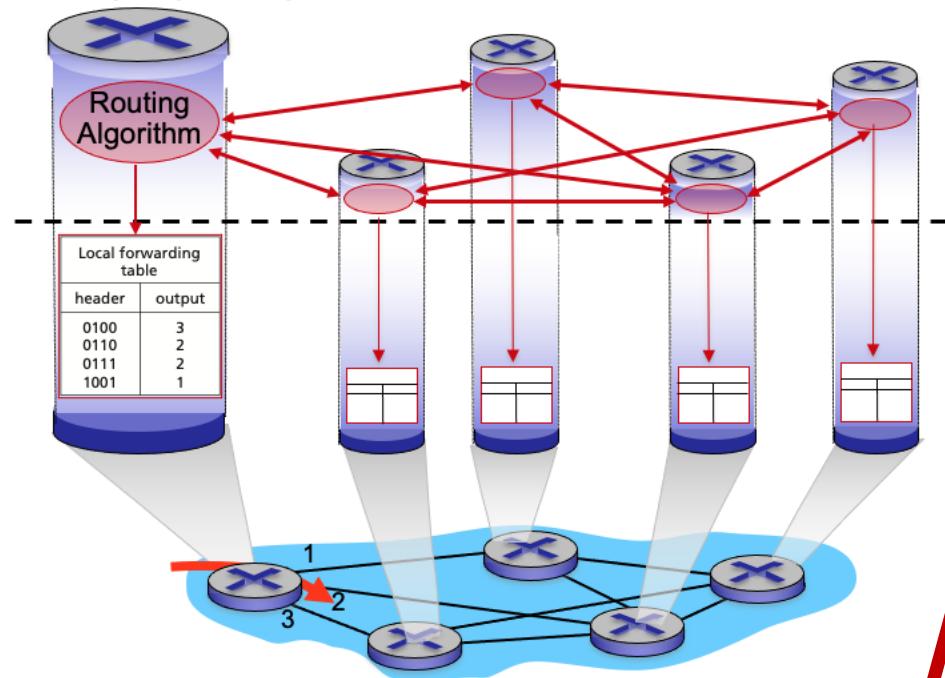


Piano di controllo Software-Defined Networking (SDN)

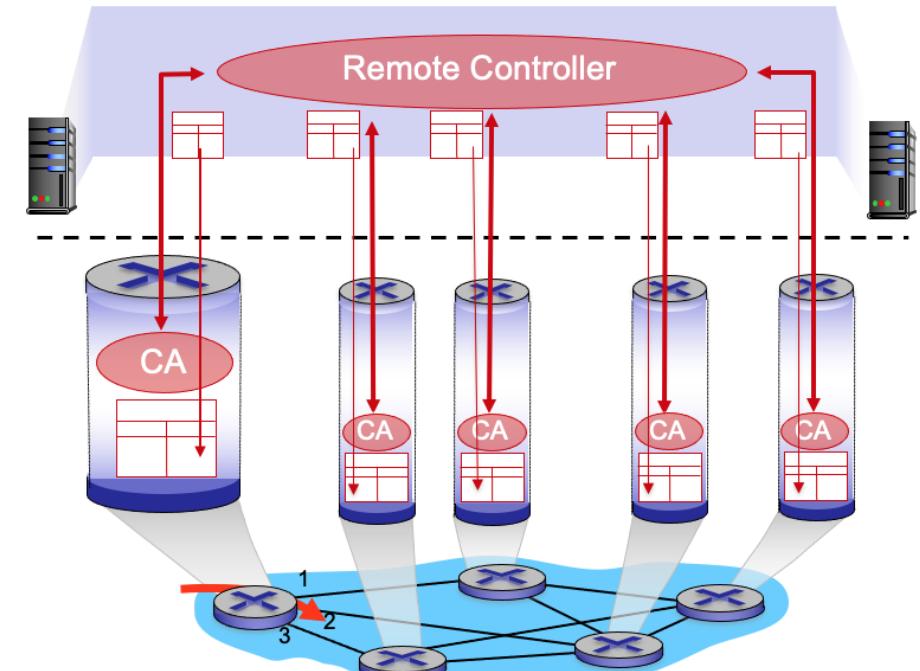
Un controllore remoto calcola le tabelle di inoltro e le installa nei router



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

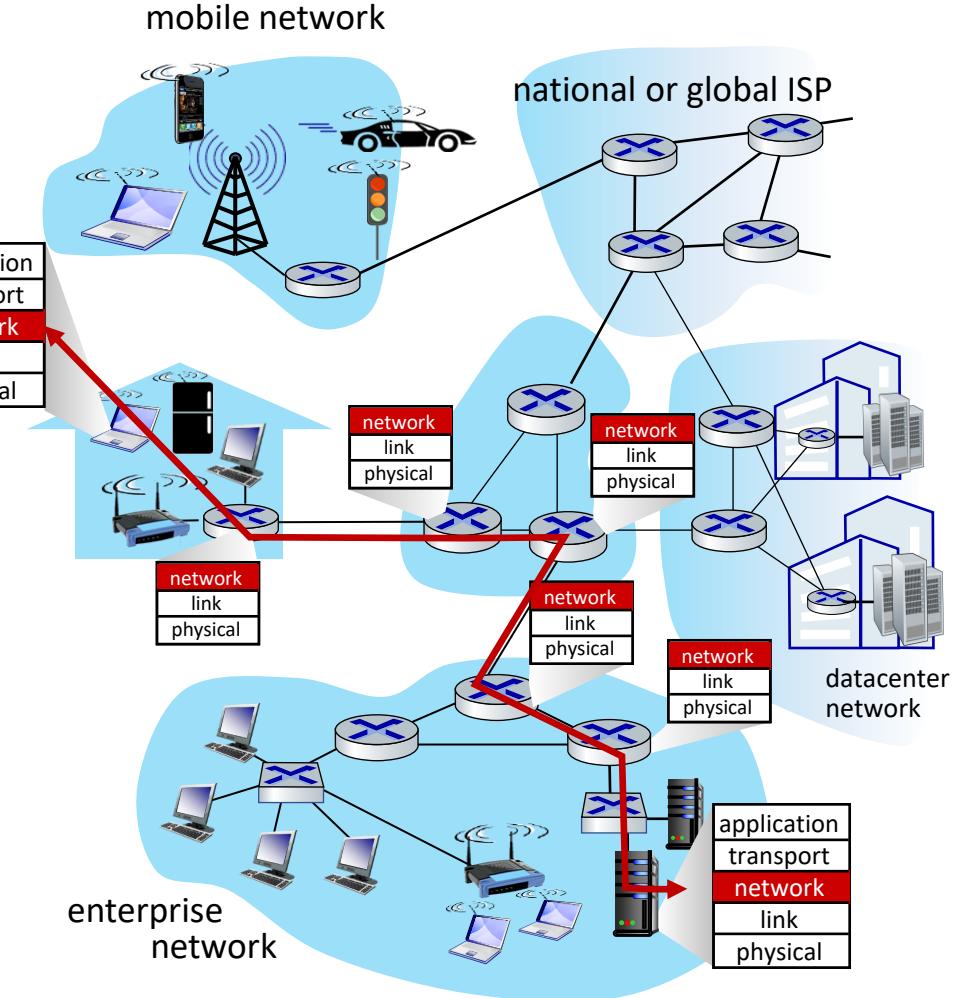


- gestione e configurazione della rete
 - SNMP
 - NETCONF/YANG

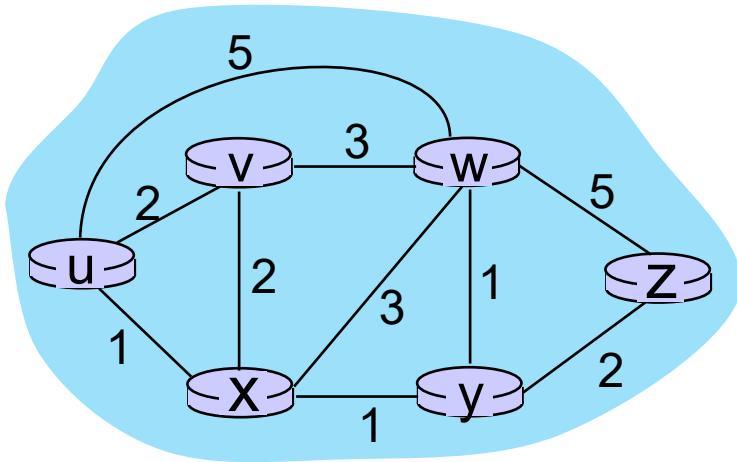
Algoritmi di instradamento (*routing algorithms*)

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 proporzionale alla congestione

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

Classificazione degli algoritmi di instradamento

Quanto velocemente cambiano i percorsi?

statici: i percorsi cambiano lentamente nel tempo (magari con un intervento umano)

globali: calcolo logicamente centralizzato o replicato sui tutti router, basato sulla conoscenza completa della topologia e del costo dei collegamenti

- algoritmi “link state”

decentralizzati: processo di calcolo iterativo, basato sullo scambio di informazioni tra vicini

- inizialmente i router conoscono solo il costo dei collegamenti ai loro vicini
- algoritmi “distance vector”

dinamici: i percorsi cambiano più velocemente

- aggiornamenti periodici o in risposta a cambiamenti del costo dei collegamenti (e della topologia)

Informazioni globali o decentralizzate?

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, oggigiorno 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

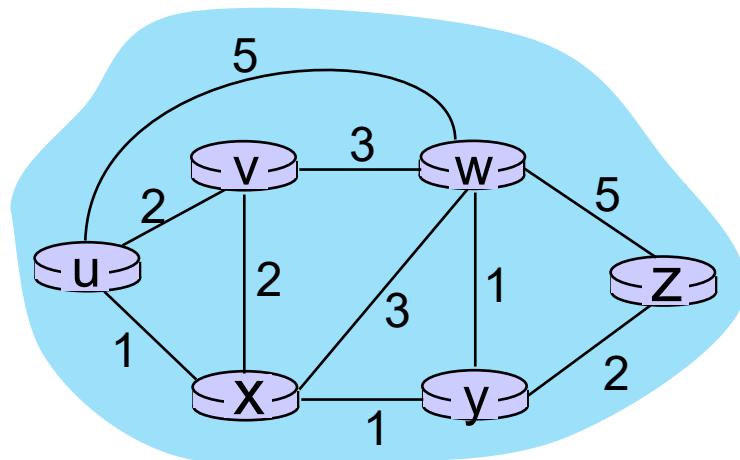
- $c_{x,y}$: costo del collegamento diretto dal nodo x al nodo y ; $= \infty$ se non sono vicini diretti
- $D(v)$: stima *corrente* del costo minimo del percorso dalla sorgente alla destinazione v
- $p(v)$: immediato predecessore di v lungo il percorso a costo minimo dall'origine a v
- N' : sottoinsieme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

Passo	N'	V D(v),p(v)	W D(w),p(w)	X D(x),p(x)	Y D(y),p(y)	Z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1						
2						
3						
4						
5						

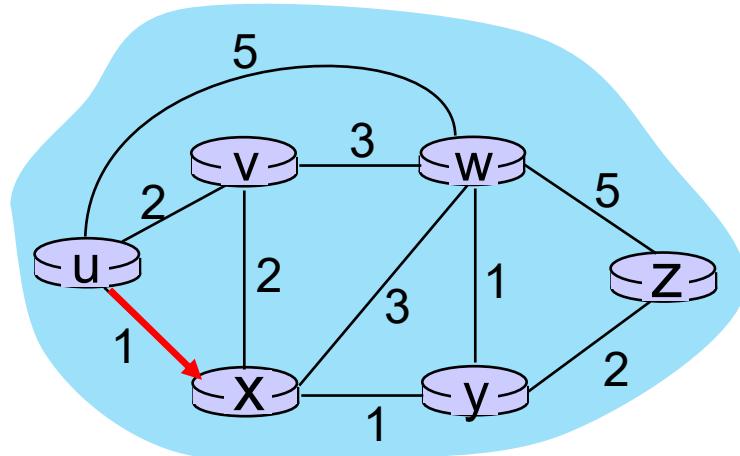


Inizializzazione (passo 0):

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	∞	∞
1	ux					
2						
3						
4						
5						

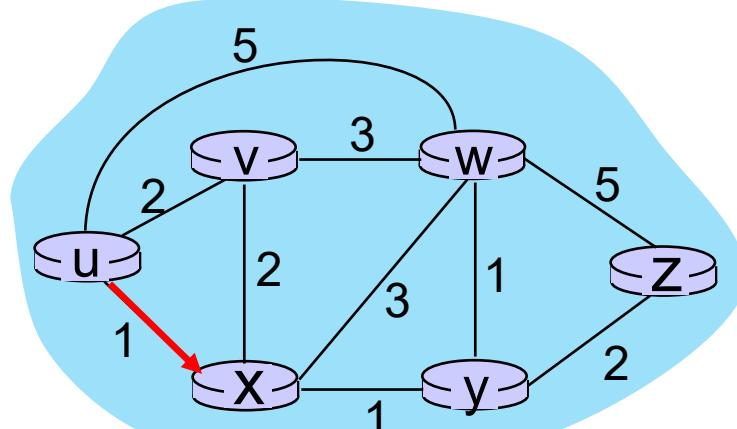


8 Ciclo

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

Algoritmo di Dijkstra: un esempio

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2						
3						
4						
5						

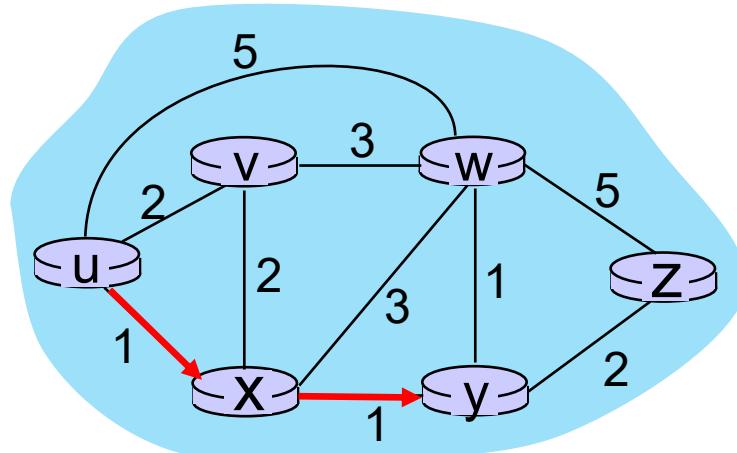


- 8 Ciclo
- 9 determina a non in N' tale che $D(a)$ sia minimo
- 10 aggiungi a a N'
- 11 aggiorna $D(b)$ per ogni b adiacente a a e non in N' :
- $$D(b) = \min (D(b), D(a) + c_{a,b})$$
- $D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$
- $D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$
- $D(y) = \min (D(y), D(x) + c_{x,y}) = \min(\infty, 1+1) = 2$

NEW!

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy					
3						
4						
5						

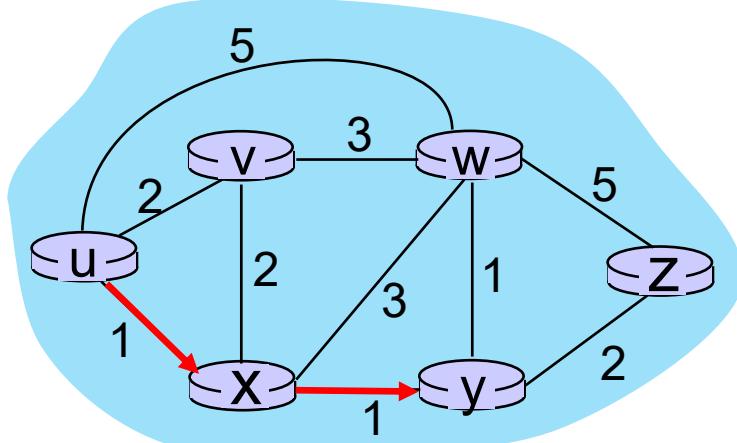


8 Ciclo

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

Algoritmo di Dijkstra: un esempio

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	∞	∞
1	ux	2,u	4,x		2,x	∞
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' :

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

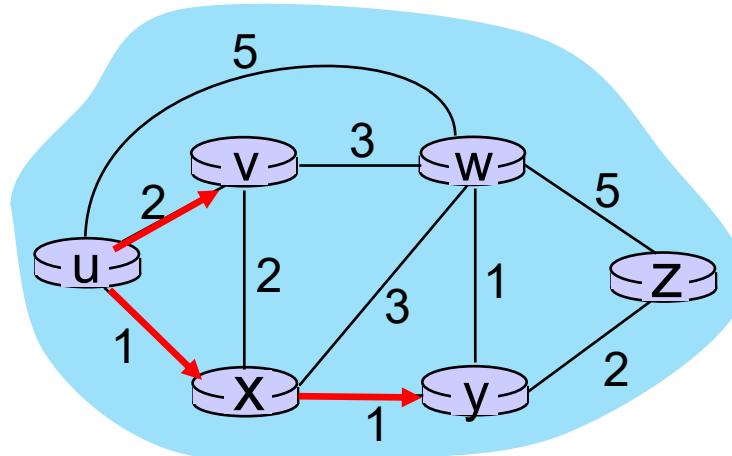
$$D(w) = \min (D(w), D(y) + c_{y,w}) = \min (4, 2+1) = 3$$

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

NEW!

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						

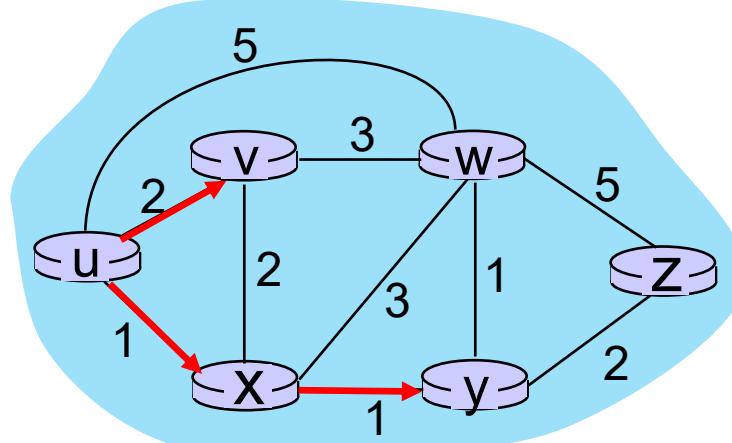


8 Ciclo

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

Algoritmo di Dijkstra: un esempio

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4						
5						



8 Ciclo

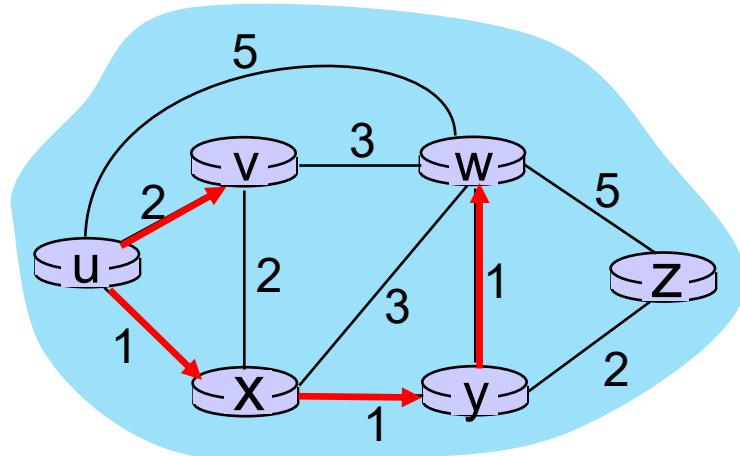
- 9 determina a non in N' tale che $D(a)$ sia minimo
 10 aggiungi a a N'
 11 aggiorna $D(b)$ per ogni b adiacente a a e non in N' :

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

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

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyy		3,y			4,y
4	uxyvw					
5						

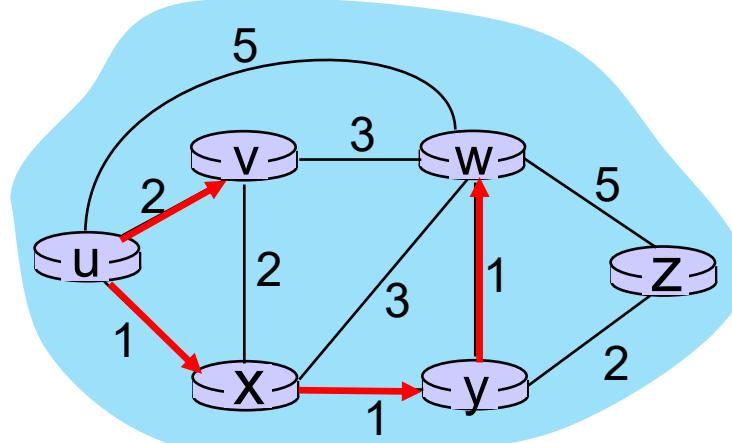


8 Ciclo

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

Algoritmo di Dijkstra: un esempio

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					4,y
5						



8 Ciclo

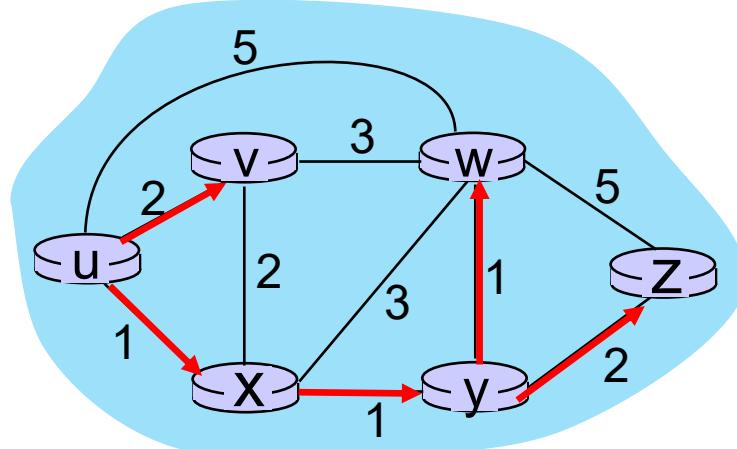
- 9 determina a non in N' tale che $D(a)$ sia minimo
 10 aggiungi a a N'
 11 aggiorna $D(b)$ per ogni b adiacente a a e non in N' :

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

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

Algoritmo di Dijkstra: un esempio

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

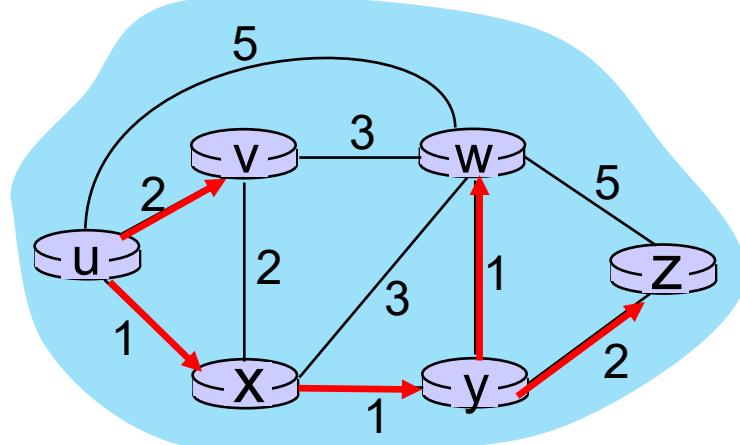


8 Ciclo

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

Algoritmo di Dijkstra: un esempio

Passo	N'	V D(v),p(v)	W D(w),p(w)	X D(x),p(x)	Y D(y),p(y)	Z 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	uxyyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

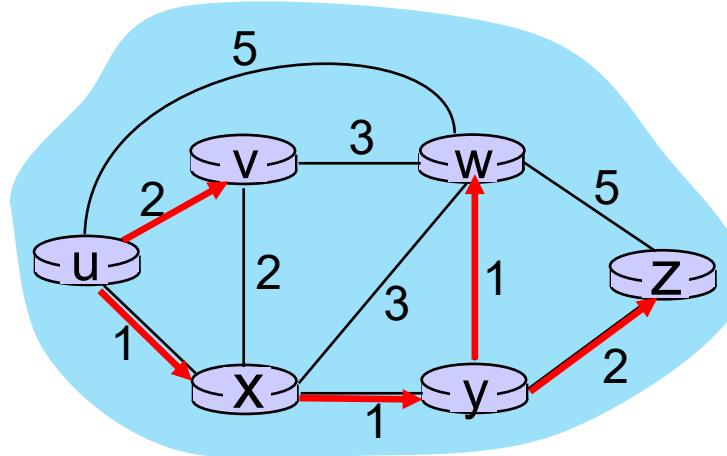


8 Ciclo

- 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' :

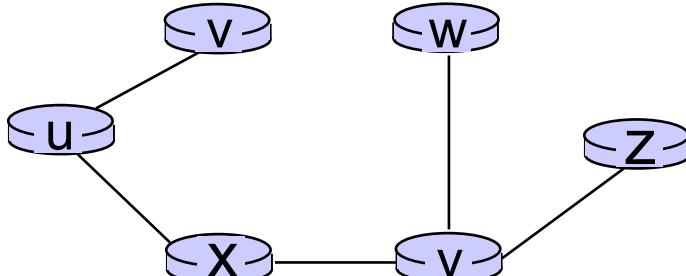
$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Algoritmo di Dijkstra: un esempio



Albero dei cammini minimi di u :

tabella di inoltro risultante in u



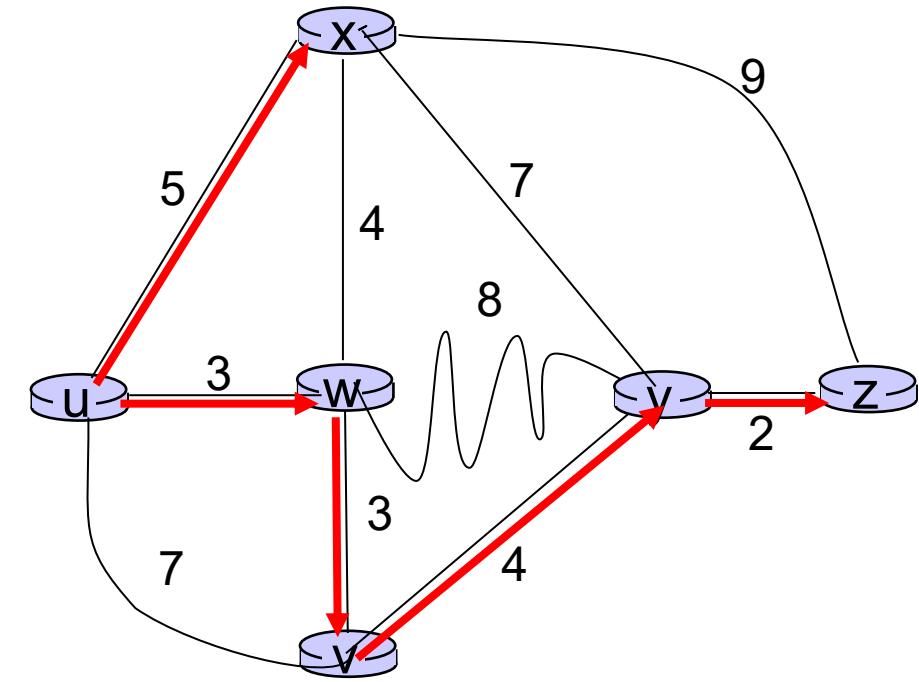
destinazione	collegamento di uscita
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

instrada da u a v direttamente

instrada da u a tutte le altre destinazioni attraverso x

Algoritmo di Dijkstra: un altro esempio

Passo	N'	v	w	x	y	z
0	u	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
1	uw	$7, u$	$3, u$	$5, u$	∞	∞
2	uwx	$6, w$	$5, u$	$11, w$	∞	∞
3	$uwxv$	$6, w$	$10, v$	$11, w$	$14, x$	∞
4	$uwxy$	$10, v$	$12, y$	$14, x$	∞	∞
5	$uwxyz$	$12, y$	∞	∞	∞	∞



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)

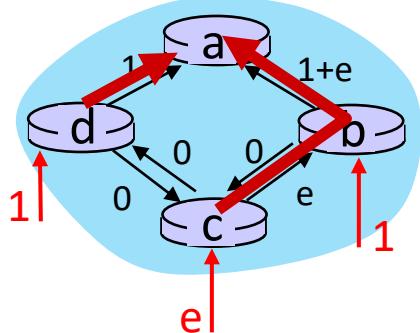
- ciascuna delle n iterazioni: deve controllare tutti i nodi, w , non in N' per determinare quello avente il costo minimo: n nodi, $n - 1$ nodi, ..., 1
- complessità $O(n^2)$
- sono possibili implementazioni più efficienti: $O(n \log n)$ usando uno *heap*

Complessità dei messaggi: n nodi

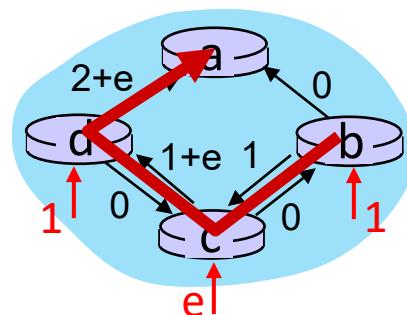
- ogni router deve *trasmettere in broadcast* le proprie informazioni sullo stato dei collegamenti agli altri n router
- algoritmi di broadcasting efficienti (e interessanti!): $O(n)$ attraversamenti dei collegamenti per diffondere un messaggio di broadcasting da una sorgente
- il messaggio di ogni router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

Algoritmo di Dijkstra: discussione: oscillazioni

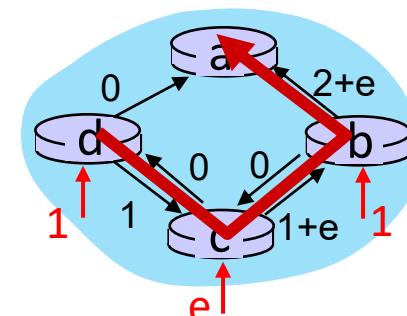
- quando i costi dei collegamenti dipendono dal volume di traffico, sono possibili **oscillazioni dei percorsi**
- scenario di esempi:
 - instradamento verso a : i nodi b , d e c trasmettono rispettivamente con tasso 1 , 1 , e (<1)
 - Il costo dei collegamenti è direzionale dipendente dal traffico



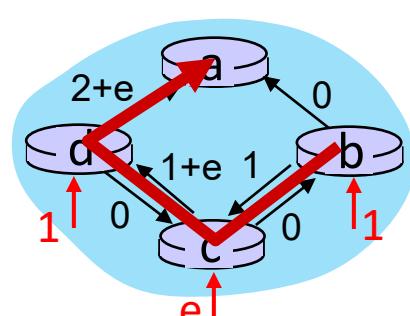
instradamento iniziale



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:

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

\min calcolato su tutti i vicini v di x

costo del cammino minimo da v a y
costo diretto del collegamento da x a v

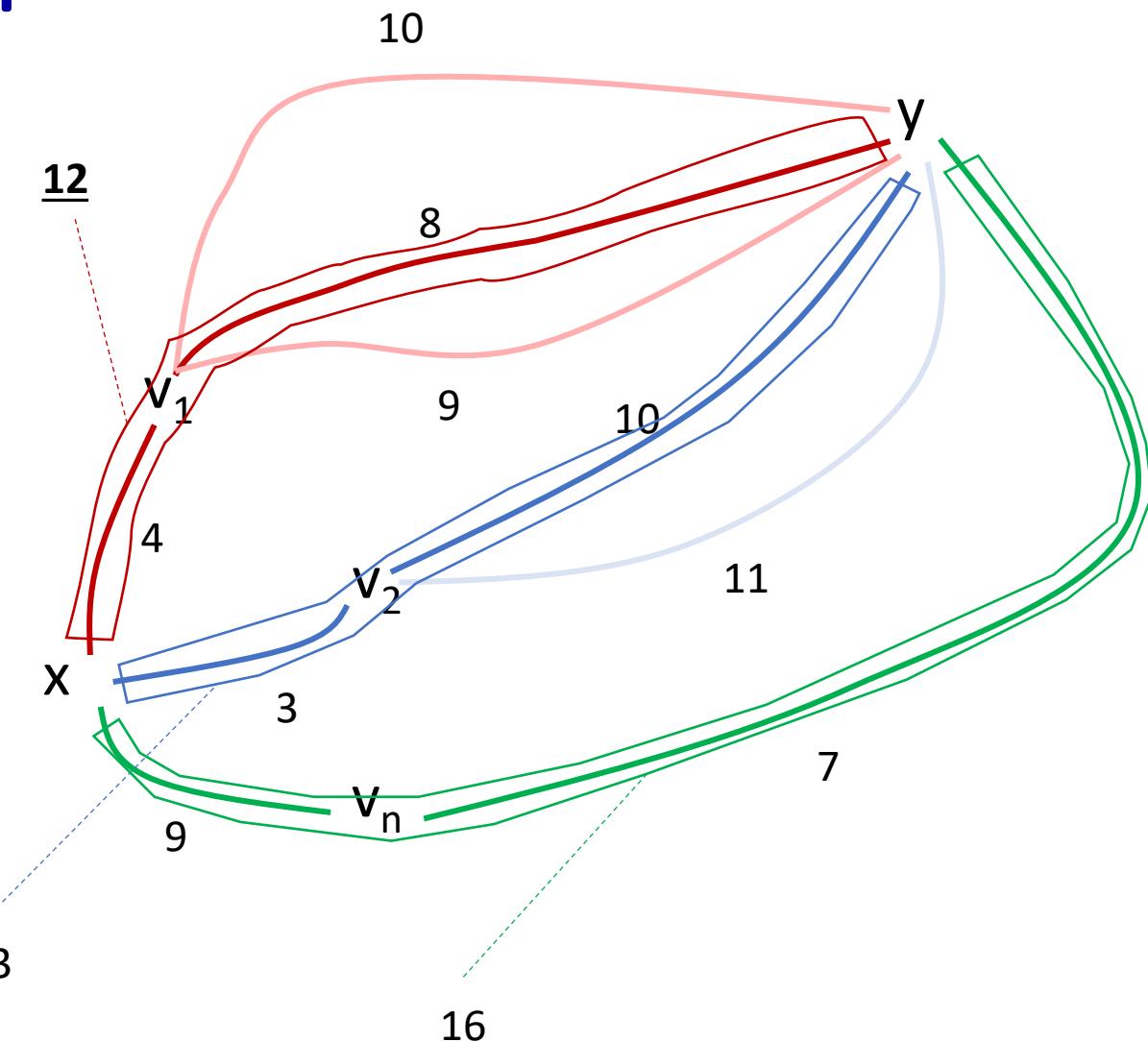
Algoritmo distance vector

Intuizione sull'equazione di BF:

Il secondo nodo lungo qualsiasi cammino da x a y è necessariamente un vicino v_i di x .

Il costo del primo arco è c_{x,v_i} mentre la parte restante del cammino non può costare meno di $d_{v_i}(y)$ cioè il costo del cammino minimo da v_i a y , per un totale di $c_{x,v_i} + d_{v_i}(y)$.

Per trovare il cammino di costo minimo da x a y , è quindi sufficiente trovare il vicino di x che minimizza questa quantità.



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:

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

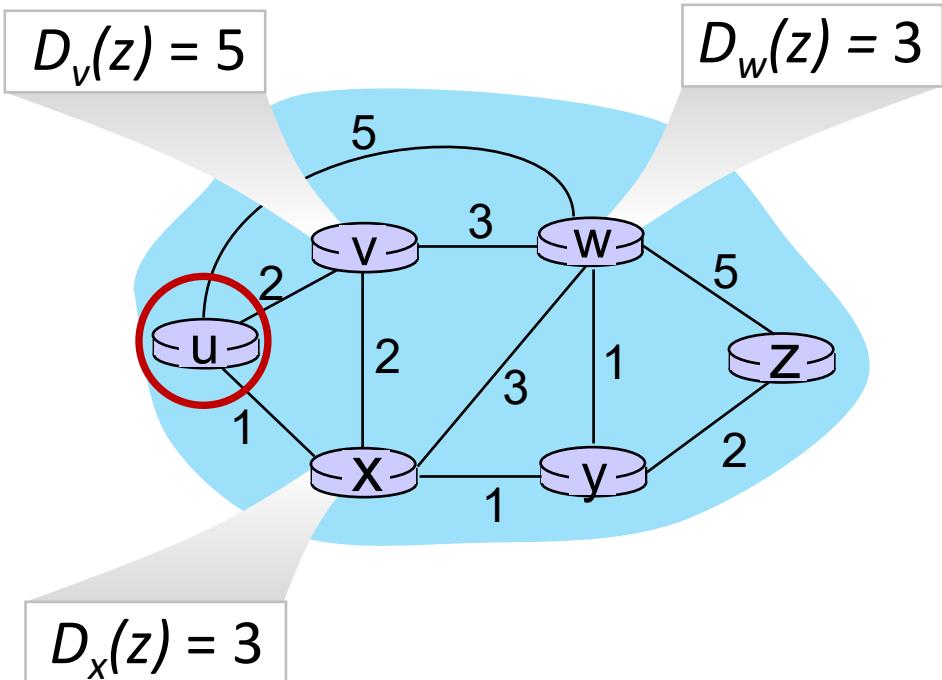
\min calcolato su tutti i vicini v di x

costo stimato del cammino minimo da v a y

costo diretto del collegamento da x a v

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:

- attendi* (un cambiamento del costo di un collegamento locale o un messaggio da un vicino)
- ↓
- ricalcola* DV usando il DV ricevuto dal vicino
- ↓
- se il DV verso qualsiasi destinazione è cambiato, *notifica* i vicini

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!

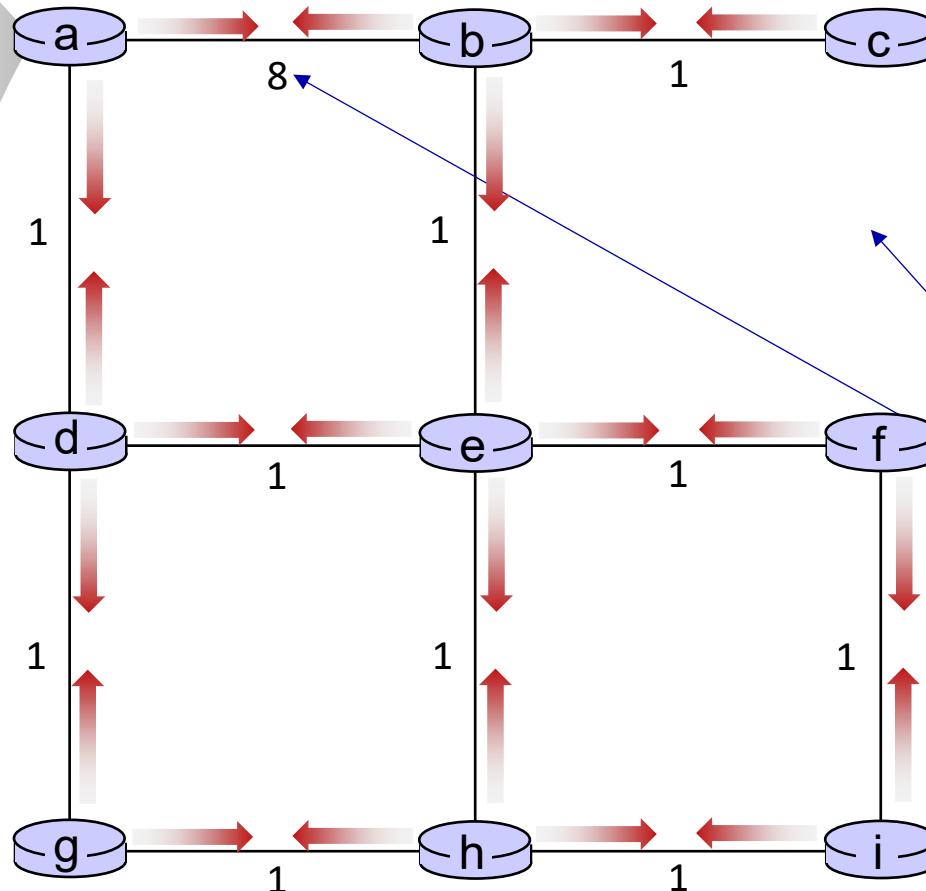
Distance vector: esempio



$t=0$

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



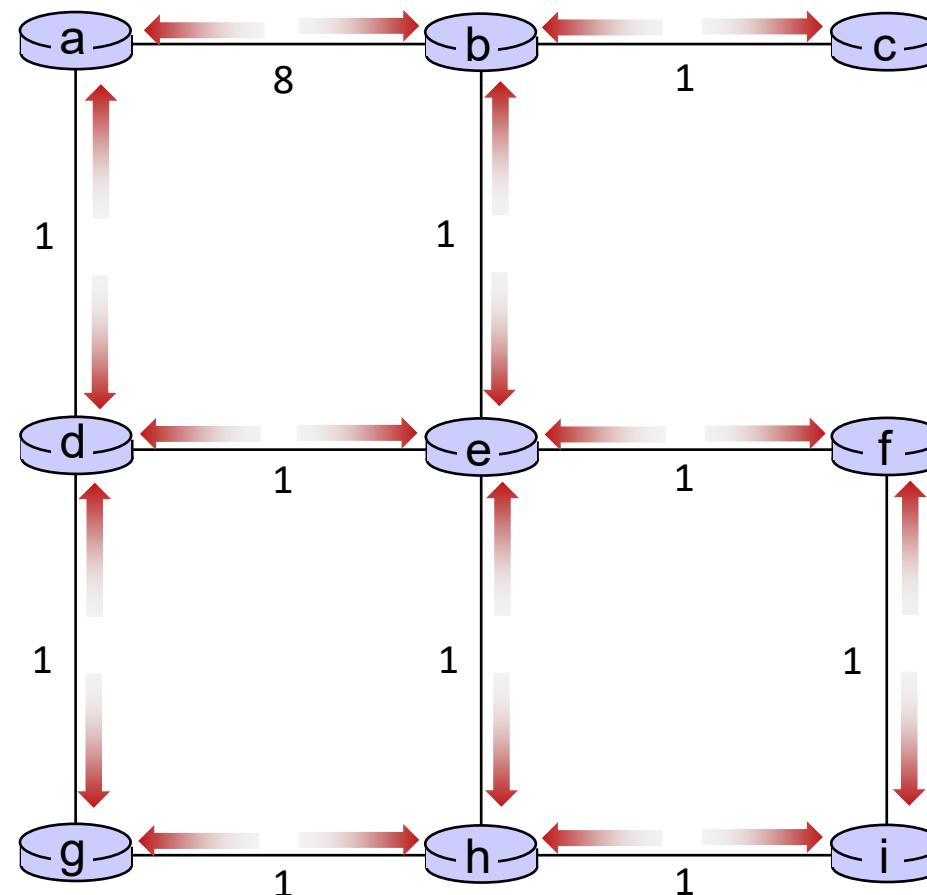
Distance vector: esempio: iterazione



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



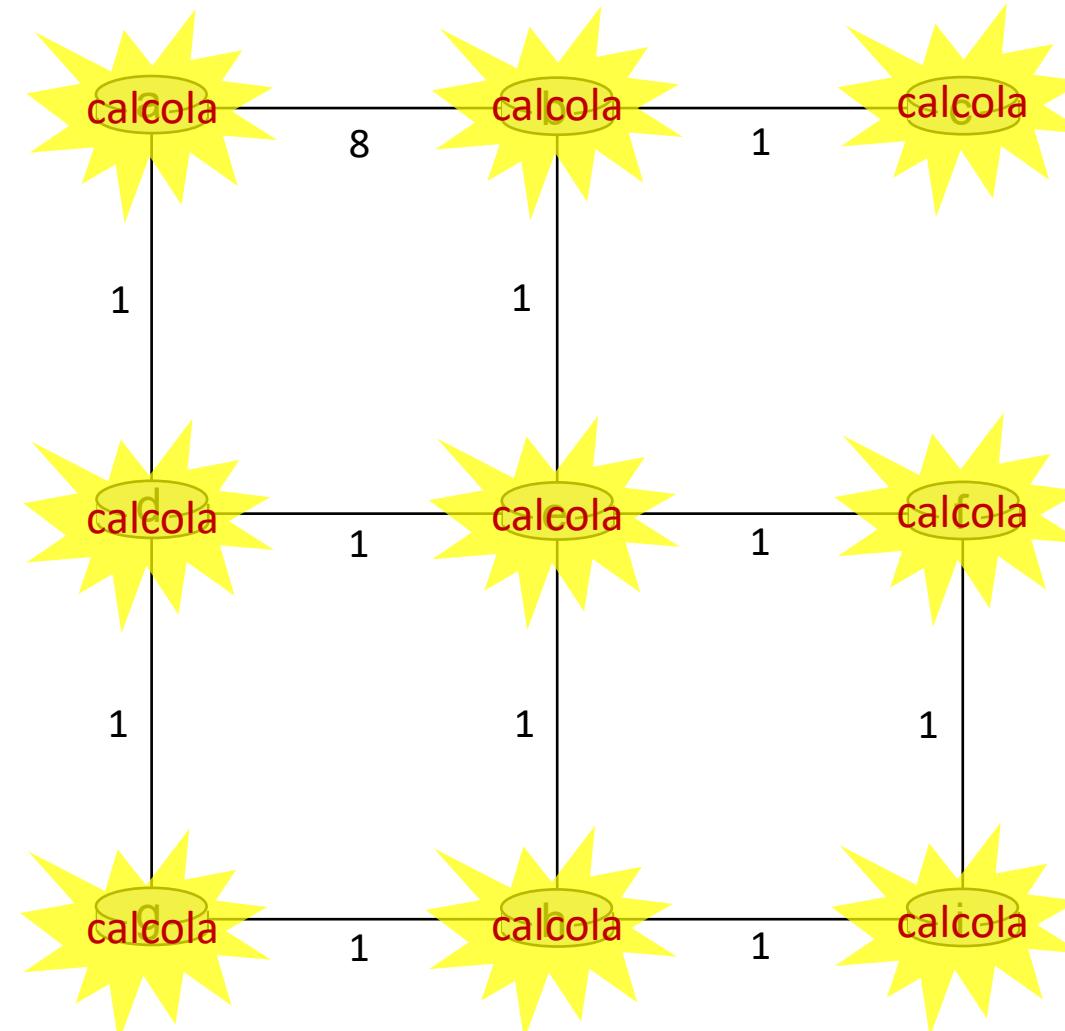
Distance vector: esempio: iterazione



$t=1$

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



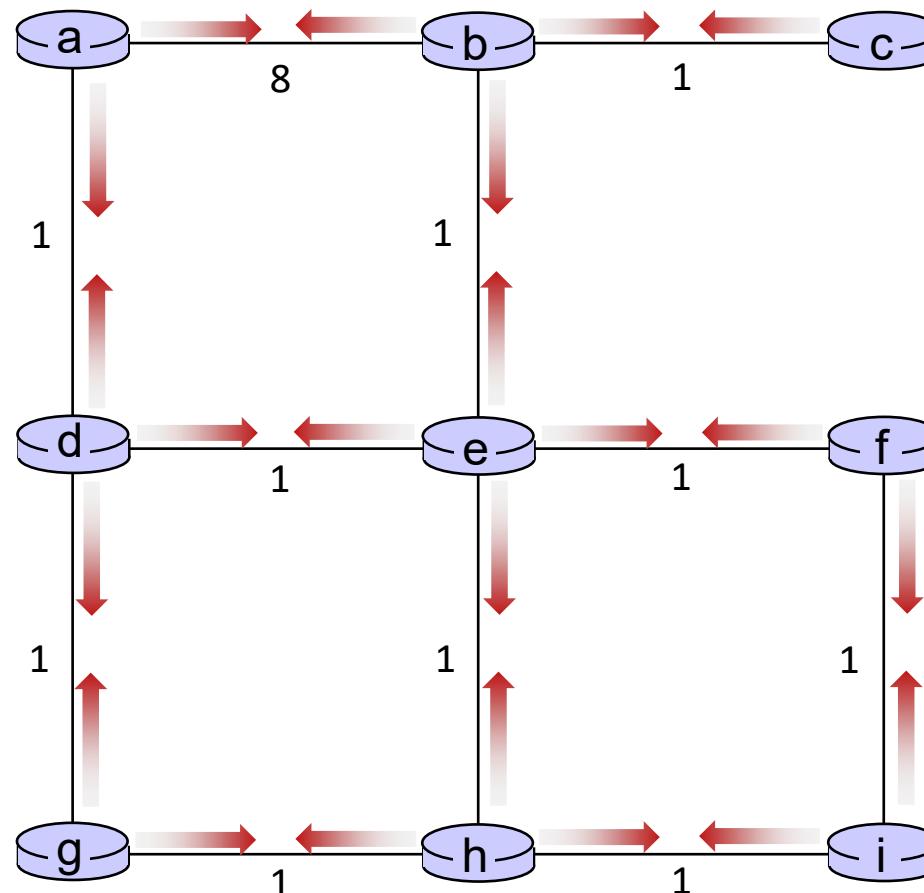
Distance vector: esempio: iterazione



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



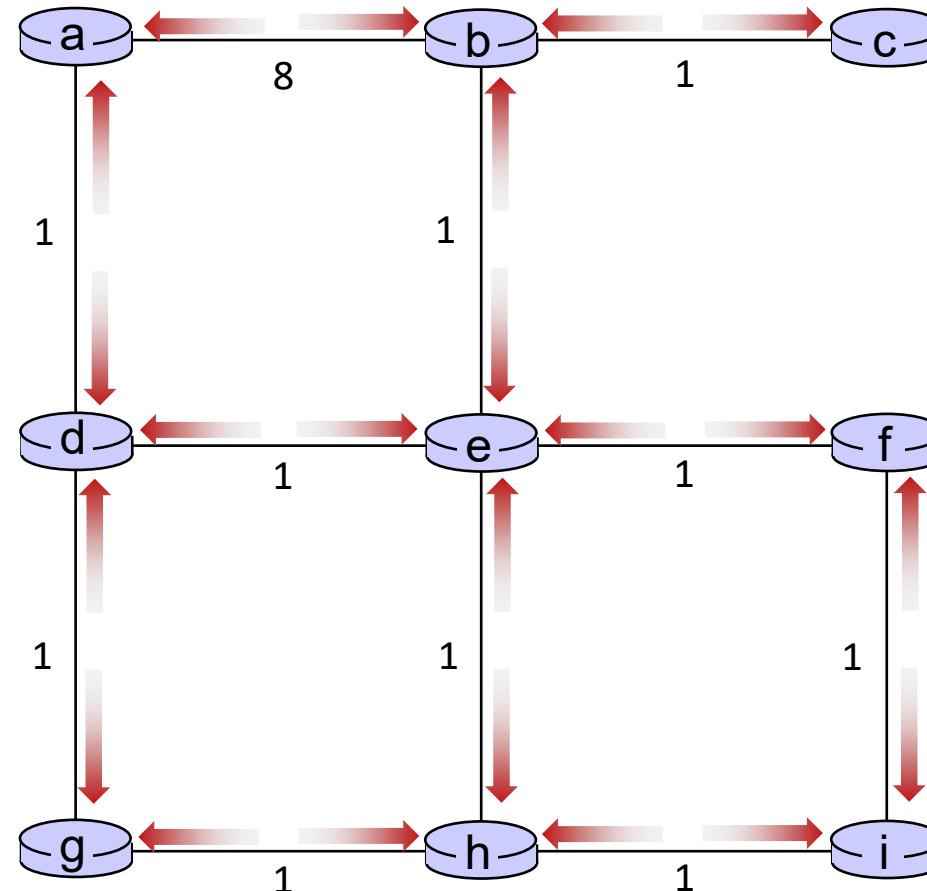
Distance vector: esempio: iterazione



$t=2$

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



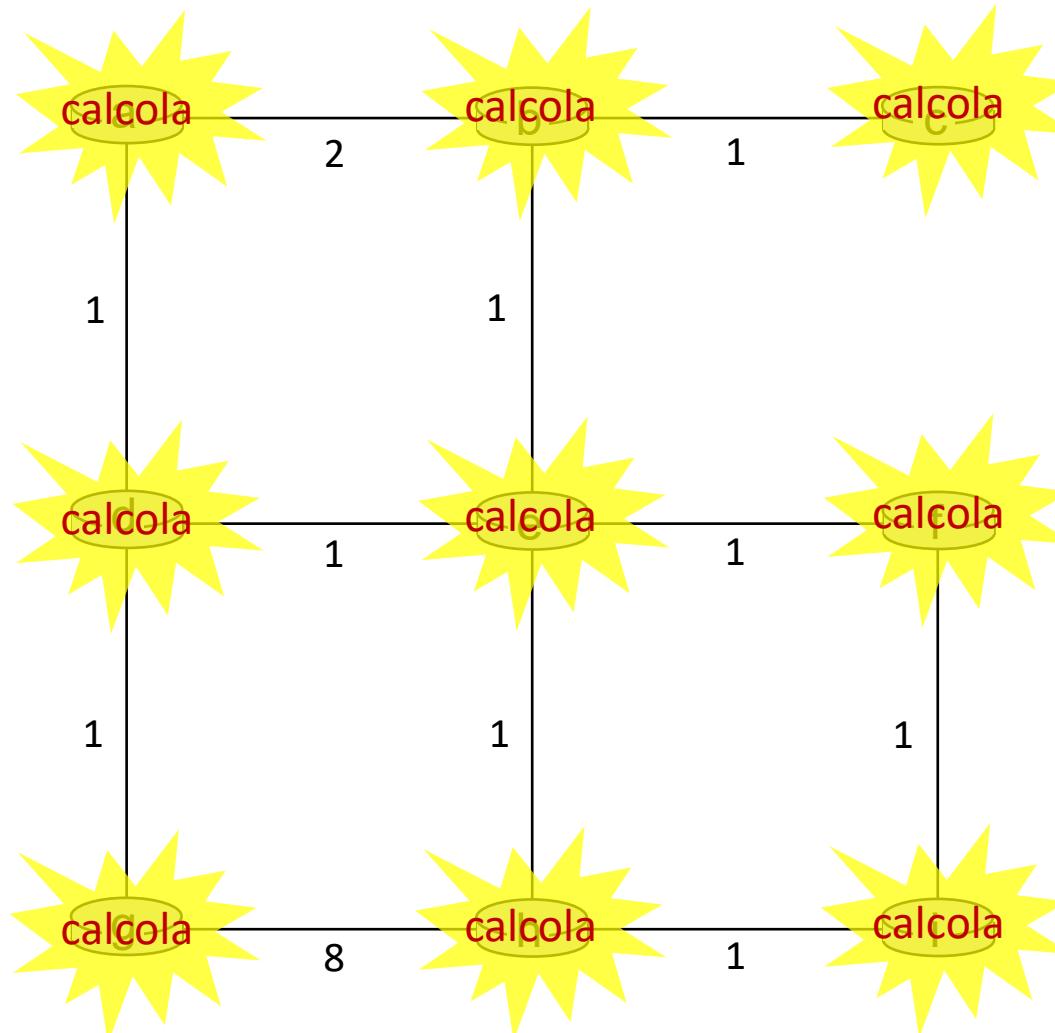
Distance vector: esempio: iterazione



$t=2$

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



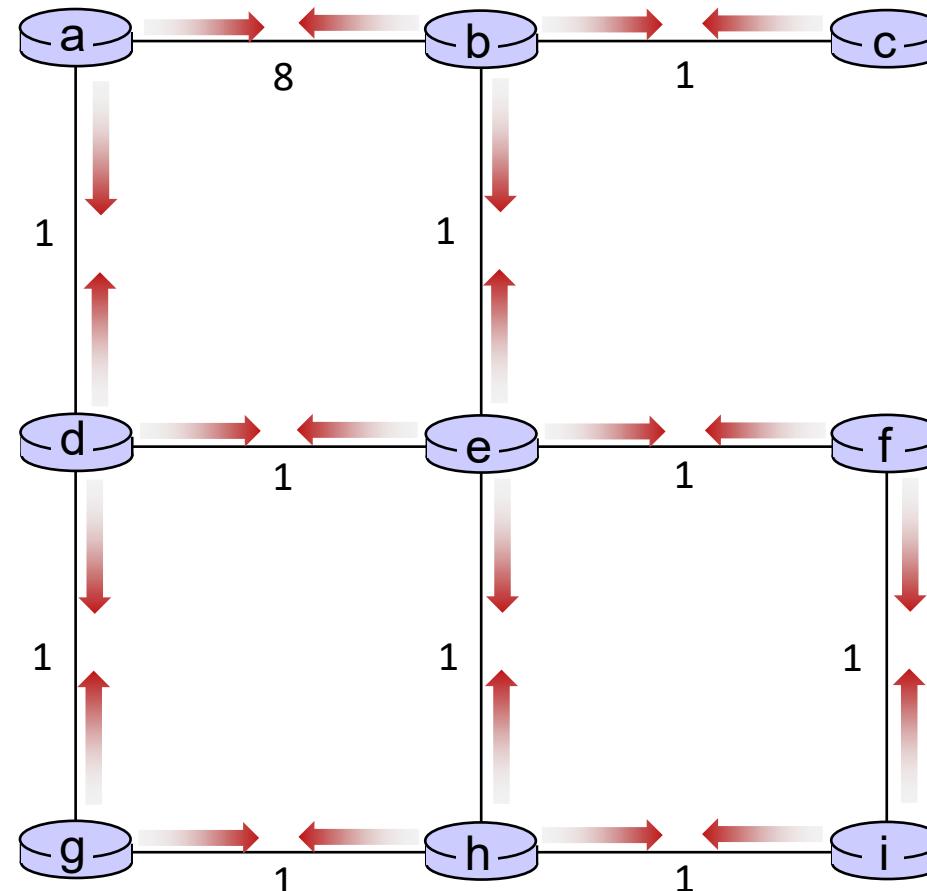
Distance vector: esempio: iterazione



t=2

Tutti i nodi:

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



Distance vector: esempio: iterazione

.... e così via

Vediamo ora le *computazioni* iterative ai nodi

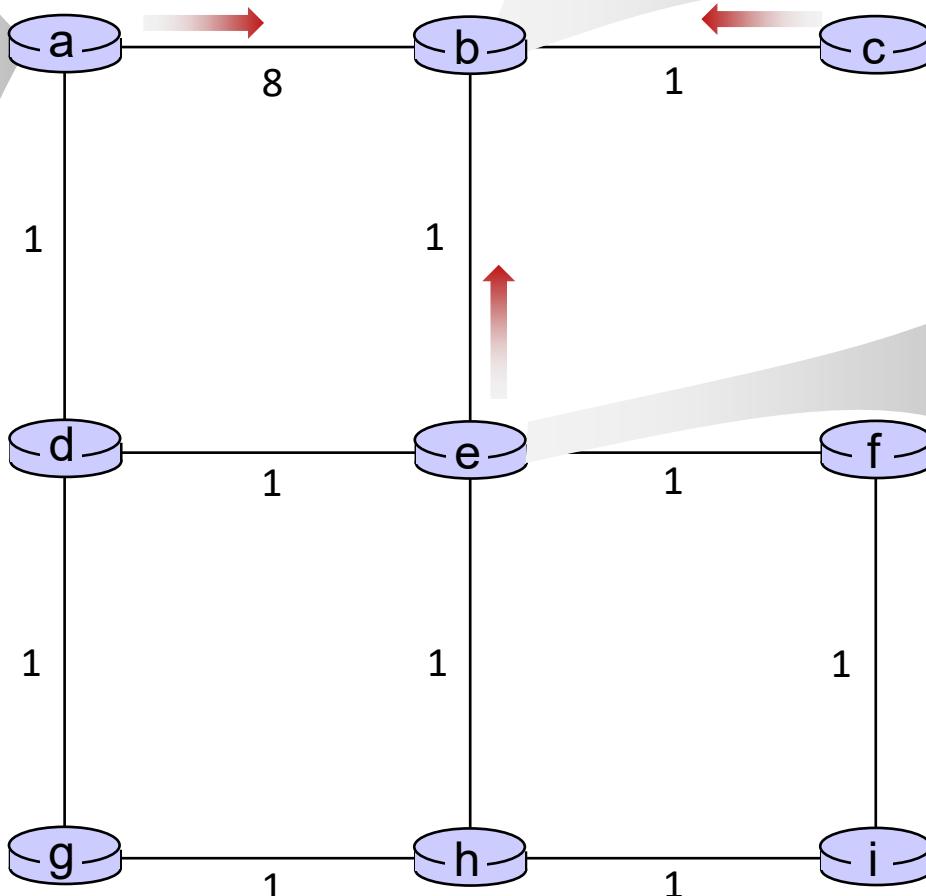
Distance vector: esempio:



$t=1$

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

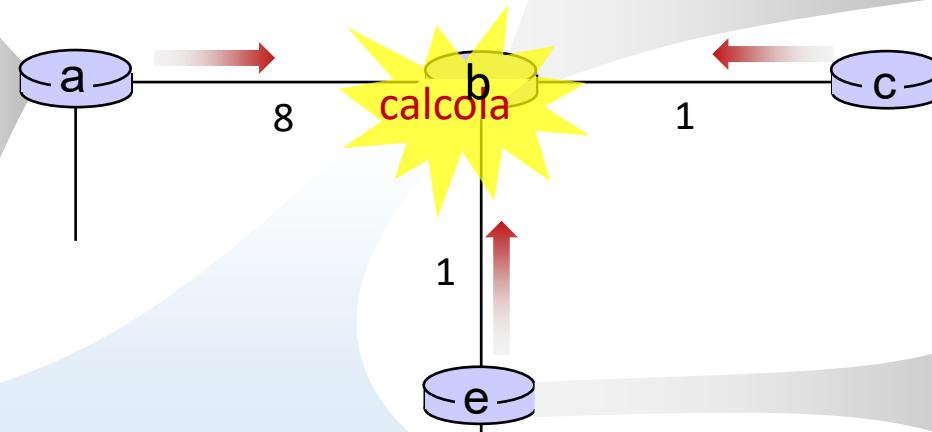
Distance vector: esempio:



t=1

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

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



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

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

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

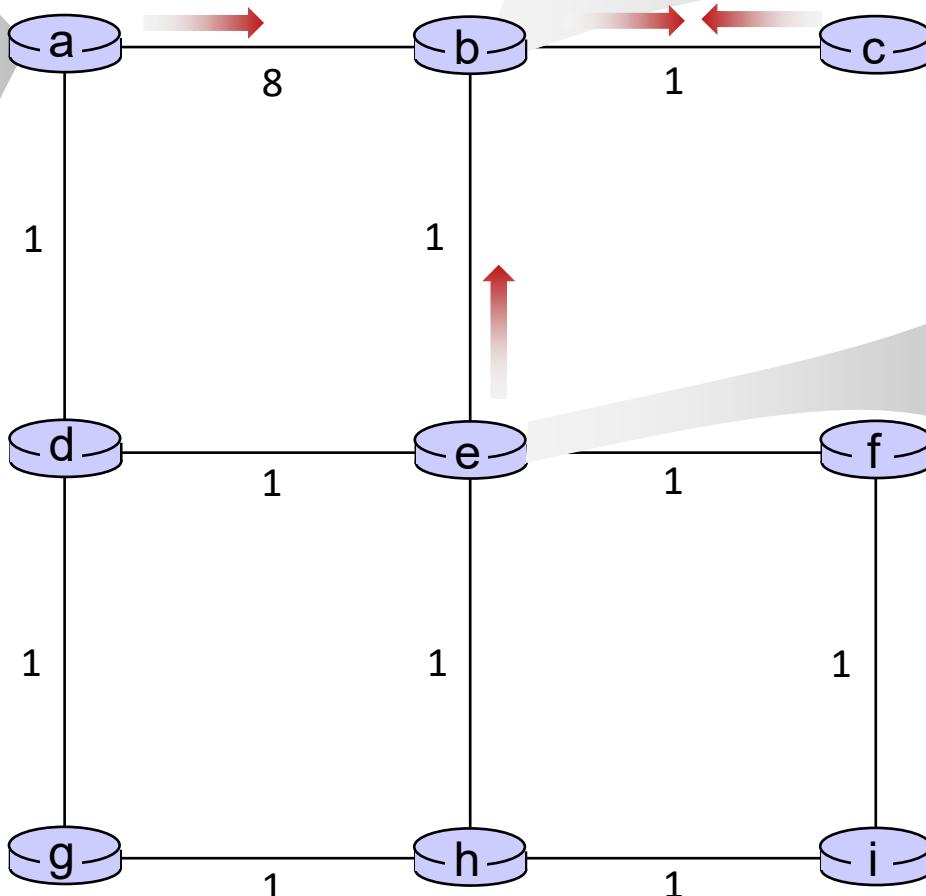
Distance vector: esempio:



t=1

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

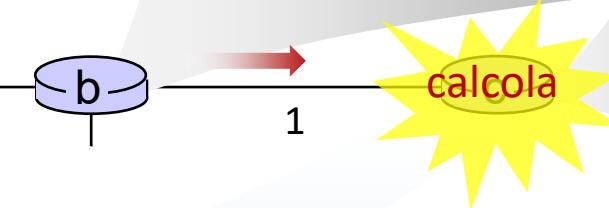
Distance vector: esempio:



$t=1$

- c riceve il DV da b ,
calcola

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/

Distance vector: esempio:

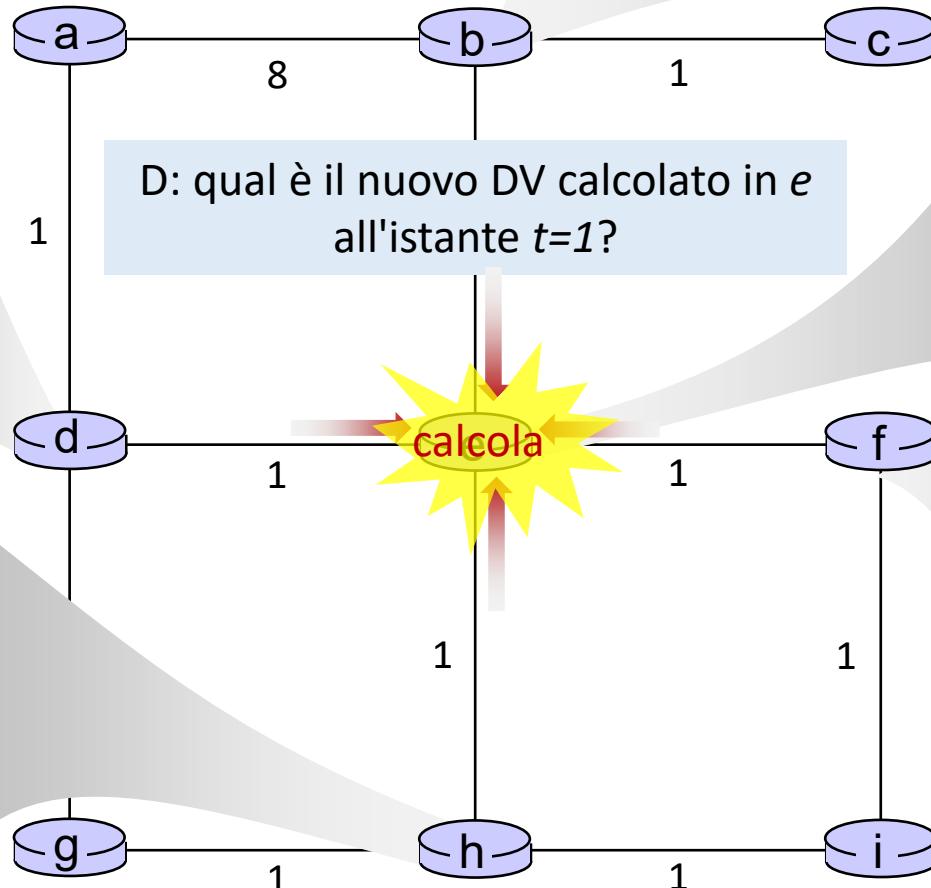


t=1

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



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

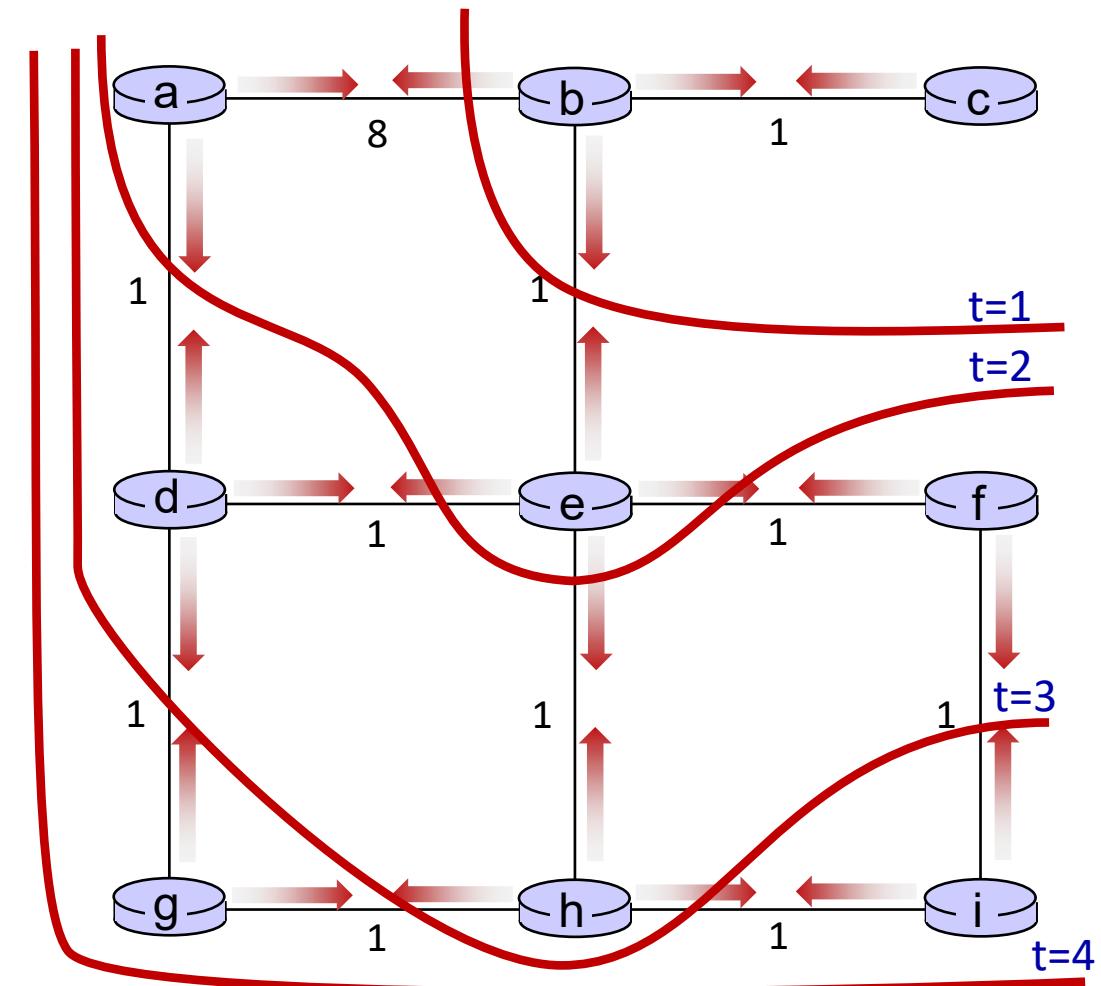
DV in f:
$D_c(a) = \infty$
$D_c(b) = \infty$
$D_c(c) = \infty$
$D_c(d) = \infty$
$D_c(e) = 1$
$D_c(f) = 0$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = 1$

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

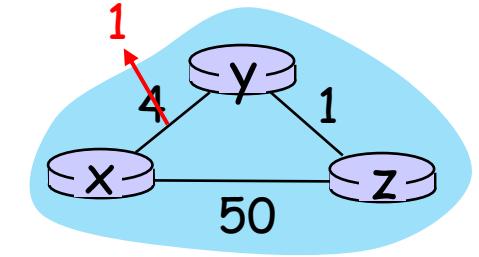


(lunghezza massima del cammino più breve (come numero di hop) tra *qualsiasi* coppia di nodi)

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.

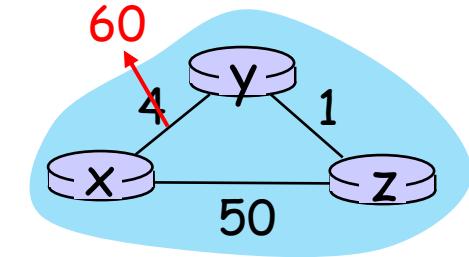
“Le buone notizie viaggiano velocemente”

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
 - 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.

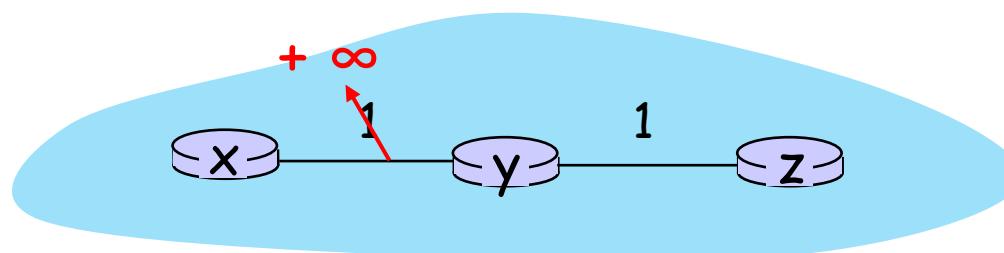
instradamento ciclico



... così fino a t_{45} quando z non considererà il collegamento diretto con x più conveniente del percorso tramite y, rompendo l’instradamento 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 $c_{y,x} = +\infty$ (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.

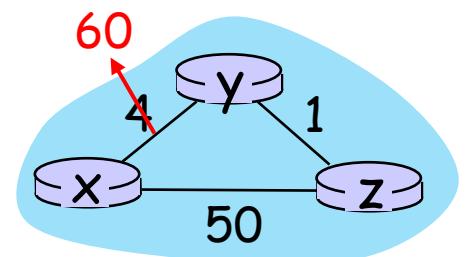
Vettore delle distanze: inversione avvelenata (poisoned reverse)

Se z intrada 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 $D_z(x) = 50$

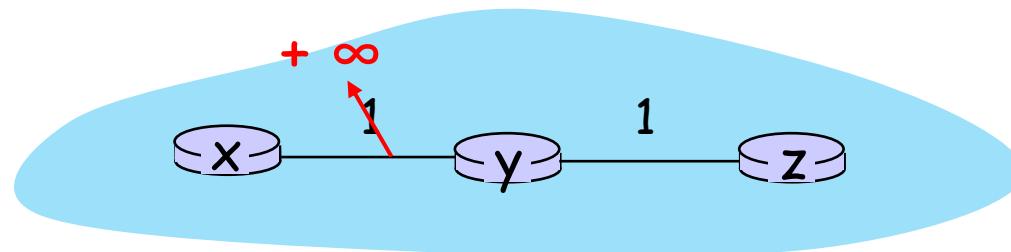
t_2 : y riceve l'aggiornamento da z e cambia l'instradamento verso x passando per z, pertanto $D_y(x) = 51$. Poiché z è ora sul percorso verso x, y avvelena il percorso inverso inviando a z $D_y(x) = +\infty$



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

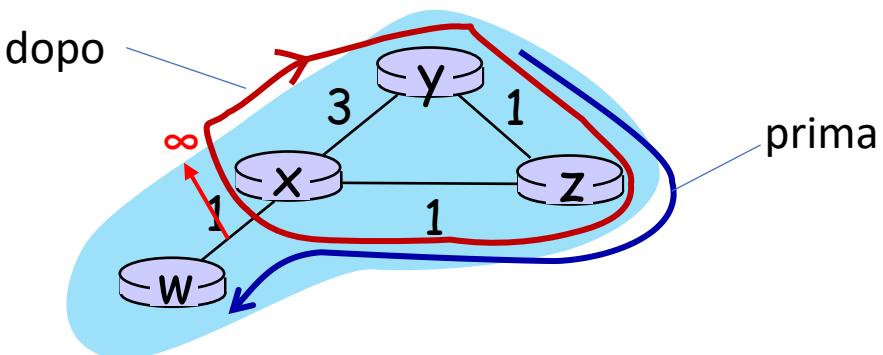


L'inversione avvelenata risolve anche questo scenario. Infatti, y ora crede che $D_z(x) = + \infty$, pertanto realizza subito che x non è più raggiungibile e quindi comunica a z che $D_y(x) = + \infty$. Ricevuta questa notifica, z realizza anch'esso che x è irraggiungibile, cioè $D_z(x) = + \infty$.

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 $D_x(w) = 11$, annuncia il nuovo costo
- z riceve annuncio da x , calcola $D_z(w) = 12$, annuncia il nuovo costo

gli annunci si ripetono ciclicamente, preservando l'instradamento circolare 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”): *bucco nero*
- Il DV di ciascun router è usato dagli altri: gli errori si propagano attraverso la rete

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

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

Livello di rete: piano di controllo (parte2)

dr. Manuel Fiorelli

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

Introduction: 1-61

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

Rendere l'instradamento scalabile

il nostro studio del routing fino ad ora - idealizzato

- tutti i router sono identici
- la rete è “piatta”

... non è vero nella pratica

scalabilità: miliardi di destinazioni:

- non può memorizzare tutte le destinazioni nelle tabelle di instradamento!
- l'invio (in broadcast) degli aggiornamenti sullo stato dei link ingolferebbe i collegamenti!
- gli algoritmi *distance vector* impiegherebbero un tempo enorme per convergere!

autonomia amministrativa:

- Internet: una rete di reti
- ogni amministratore di rete può voler controllare l'instradamento nella propria rete o nascondere dettagli della sua struttura interna

Approccio di Internet al routing scalabile

aggregare i router in regioni note come “*sistemi autonomi*” (AS, *autonomous system*) (anche detti “*domini*”): di solito formati da router sottoposti alla stessa amministrazione.

Un ISP può costituire un unico AS oppure essere partizionato in più AS.

AS identificati da un Autonomous System Number (ASN) [allocati dallo IANA tramite 5 Registri Regionali].

intra-AS (o “intra-domain”):

instradamento *interno al sistema autonomo* (“rete”)

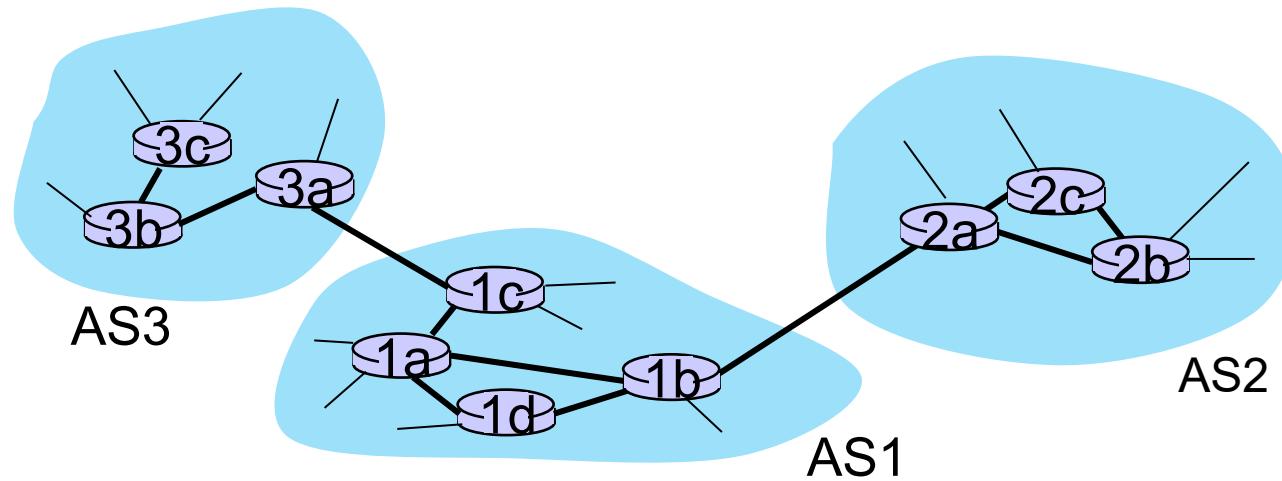
- tutti i router nell'AS devono eseguire lo stesso *protocollo di instradamento interno al sistema autonomo*
- AS differenti possono eseguire differenti *protocolli di instradamento interno al sistema autonomo*
- **router gateway:** sul “bordo” (*edge*) del proprio AS, connesso a uno o più router in altri AS

inter-AS (o “inter-domain”):

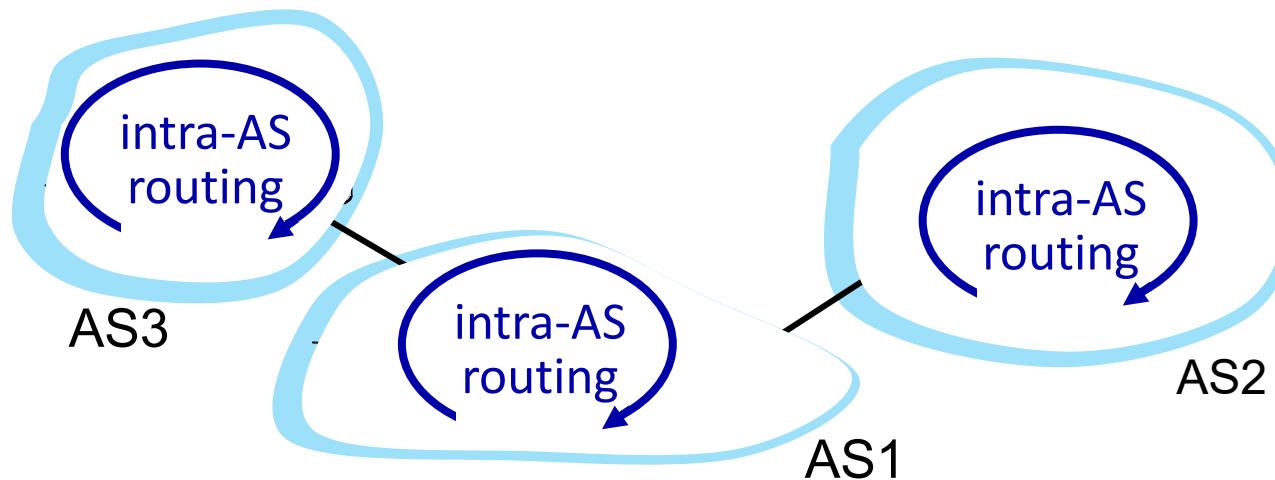
instradamento *tra* sistemi autonomi

- i gateway effettuano l'intradamento inter-AS (come pure l'intradamento intra-AS)

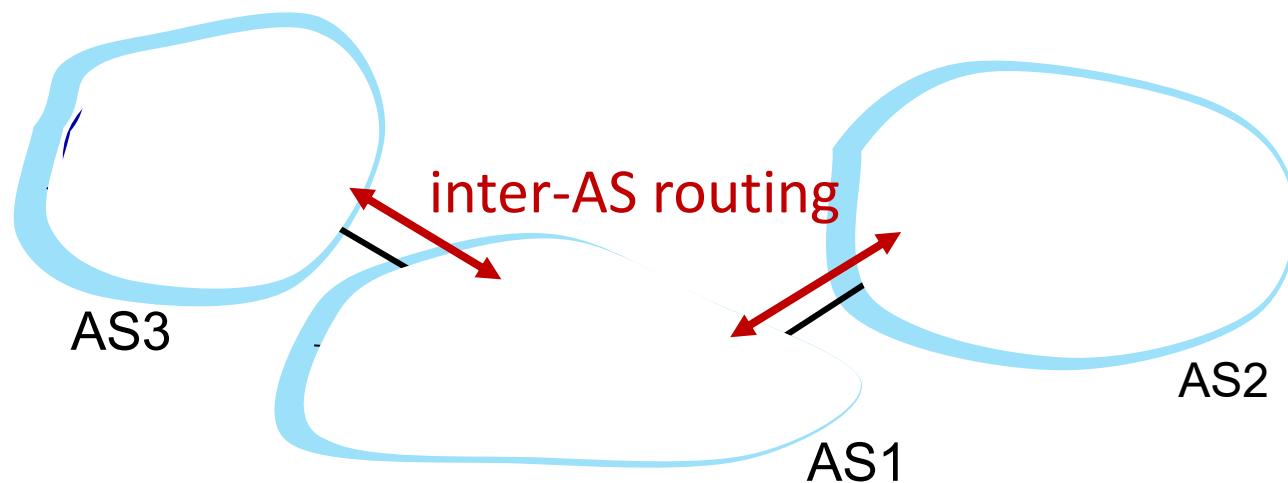
Sistemi autonomi (AS) interconnessi



Sistemi autonomi (AS) interconnessi



Sistemi autonomi (AS) interconnessi



Sistemi autonomi (AS) interconnessi

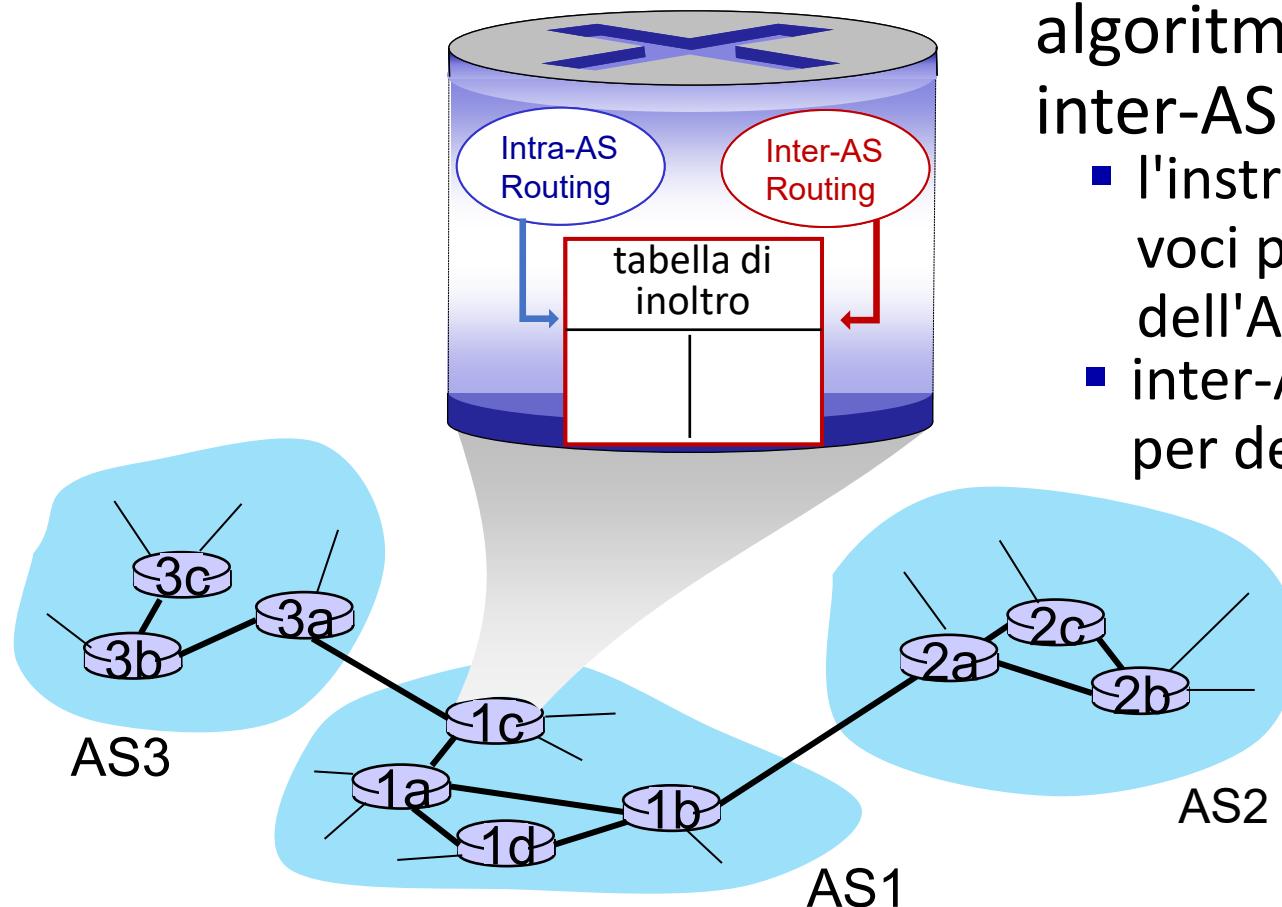
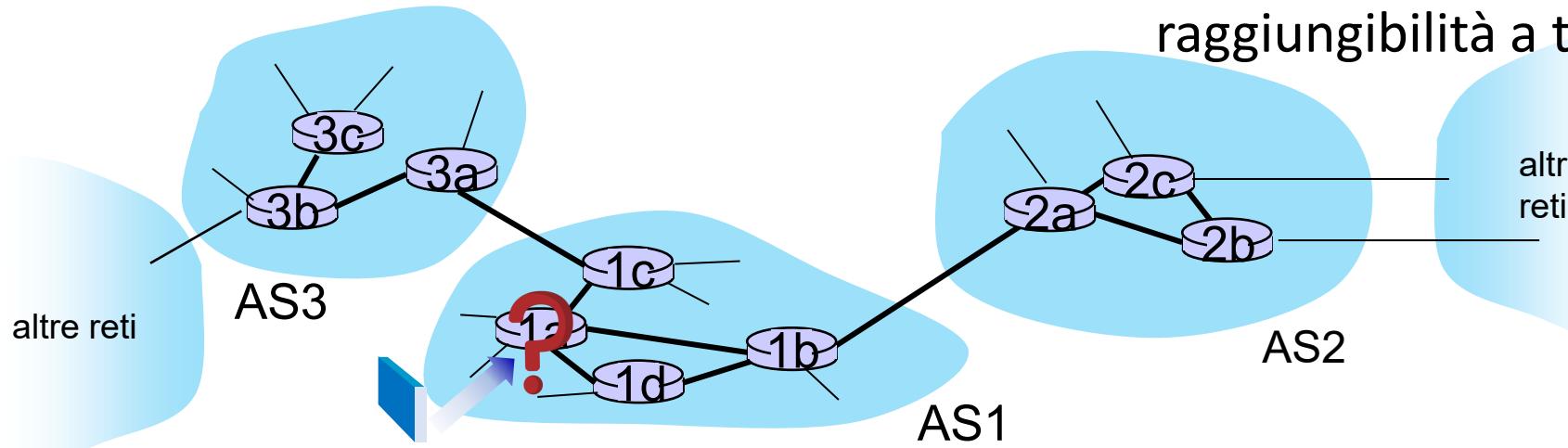


tabella di inoltro configurata dagli algoritmi di instradamento intra- e inter-AS

- l'instradamento intra-AS determina le voci per le destinazioni all'interno dell'AS
- inter-AS & intra-AS determinano le voci per destinazioni esterne

Instradamento inter-AS: un ruolo nell'inoltro intra-dominio

- Si supponga che un router dentro AS1 riceva un datagramma destinato al di fuori di AS1:
 - Il router dovrebbe inoltrare il pacchetto a un router gateway in AS1, ma quale?



**Istradamento inter-AS in AS1
deve:**

1. imparare quali destinazioni sono raggiungibili attraverso AS2 e quali attraverso AS3
 2. propagare queste informazioni di raggiungibilità a tutti i router in AS1

Instradamento intra-AS: instradamento interno al AS

protocolli di instradamento intra-AS più comuni:

- **RIP: Routing Information Protocol [RFC 1723]**
 - DV classico: DV scambiati ogni 30 secondi
 - non più largamente usato
- **EIGRP: Enhanced Interior Gateway Routing Protocol**
 - basato su DV
 - precedentemente di proprietà di Cisco per decenni (è diventato aperto nel 2013 [RFC 7868])
- **OSPF: Open Shortest Path First [RFC 2328]**
 - instradamento link-state
 - Protocollo IS-IS (ISO standard, non standard RFC) essenzialmente identico a OSPF

OSPF (Open Shortest Path First)

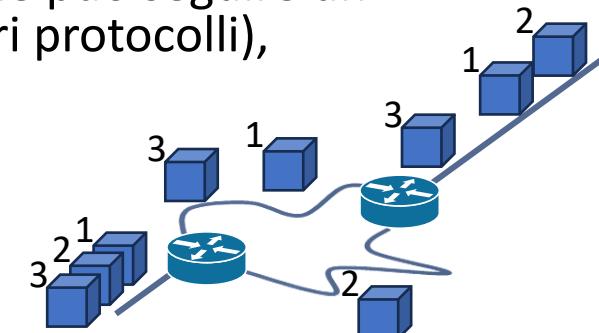
- “aperto”: disponibile pubblicamente
- classico link-state
 - ciascun router utilizza il *flooding* (inondazione) per inviare in broadcast le informazioni circa lo stato dei collegamenti (direttamente su IP invece di utilizzare TCP/UDP) a tutti gli altri router nell'intero AS
 - costo dei collegamenti: inversamente proporzionale alla larghezza di banda
 - ogni router dispone di una topologia completa, utilizza l'algoritmo di Dijkstra per calcolare la tabella di inoltro
- *sicurezza*: tutti i messaggi OSPF sono autenticati (per prevenire intrusioni dannose)

Equal-cost multi-path (ECMP) Routing

L'**instradamento ECMP** consente di instradare i pacchetti verso una stessa destinazione usando molteplici percorsi di uguale costo (verso un certo prefisso), *aumentando la larghezza di banda*.

Un router che deve inoltrare un pacchetto fa *load balancing* tra i possibili *next hop*:

- **per flusso** (es. può determinare il next hop calcolando un hash dei campi di intestazione che definiscono un flusso, quali indirizzo IP di origine e quello di destinazione, nonché i numeri di porta UDP/TCP, violando la separazione tra i livelli): ne segue che tutti i pacchetti che appartengono al medesimo flusso attraversano lo stesso percorso
- **per destinazione** (es. usando come input della funzione di hash solo l'indirizzo IP di destinazione): ne segue che i pacchetti che hanno lo stesso indirizzo IP di destinazione attraversano lo stesso percorso (anche se hanno origine in host differenti)
- **per pacchetto**: ne segue che ogni pacchetto per una stessa destinazione può seguire un percorso differente. Può creare problemi a TCP (e analogamente ad altri protocolli), a causa di:
 - consegna fuori ordine (interpretata come un evento di perdita, innescando ritrasmissioni e riduzione della finestra di congestione)
 - variabilità del ritardo (RTT "ballerino", aumento del DevRTT, timeout più lunghi, minore reattività alle perdite reali)
 - variabilità della MTU minima (interferisce con il meccanismo di PMTUD)



OSPF gerarchico

- gerarchia a due livelli: area locale, dorsale (*backbone*).

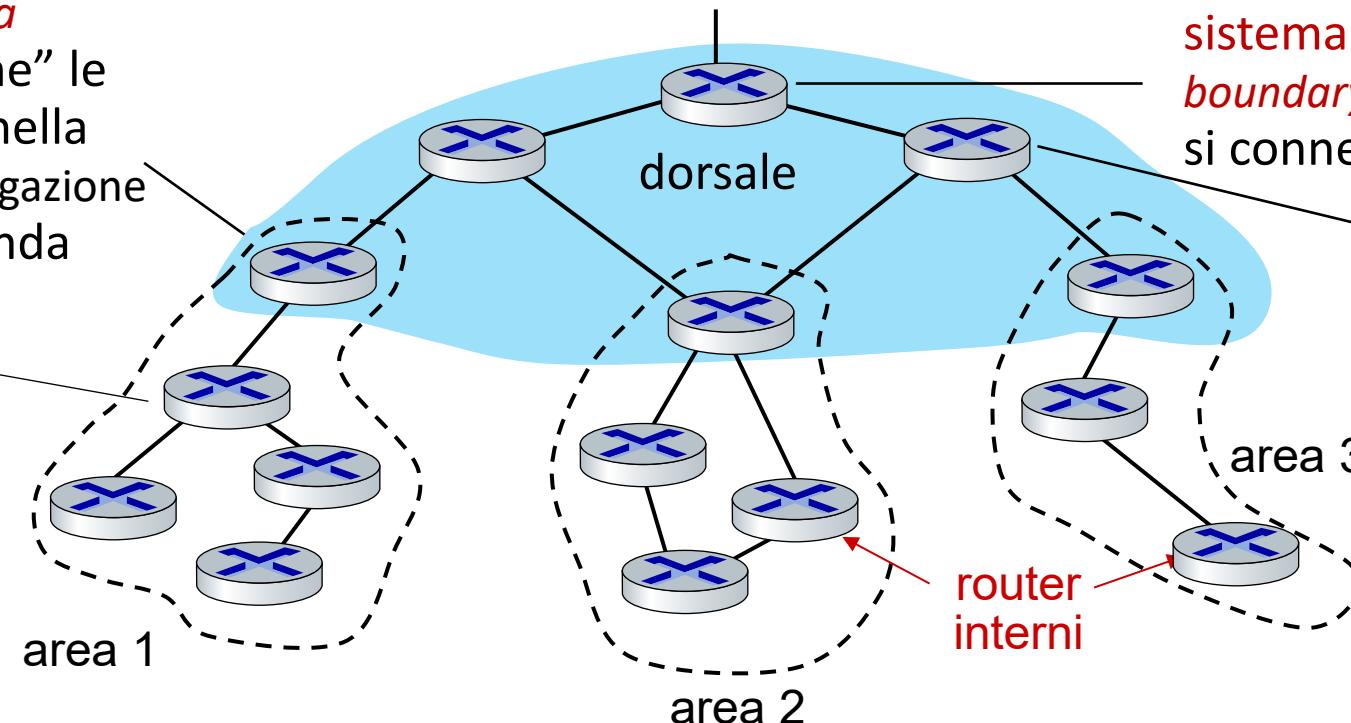
- annunci link-state inondati solo in area o dorsale
- ogni nodo ha una topologia dettagliata dell'area; conosce solo la direzione per raggiungere altre destinazioni

router di confine d'area (*area border routers, ABR*):

“riassume” le distanze verso destinazioni nella propria area (in realtà, l'aggregazione degli indirizzi è opzionale), manda annunci nella dorsale

router locali (*local routers*):

- inviano le informazioni di stato solo nell'area
- calcolano l'instradamento dentro l'area
- inoltrano i pacchetti all'esterno attraverso i router di confine d'area



Router di confine del sistema autonomo(AS) (*boundary router, ASBR*): si connette ad altri AS

Router di dorsale (*backbone router*): esegue OSPF limitatamente alla area 3 dorsale

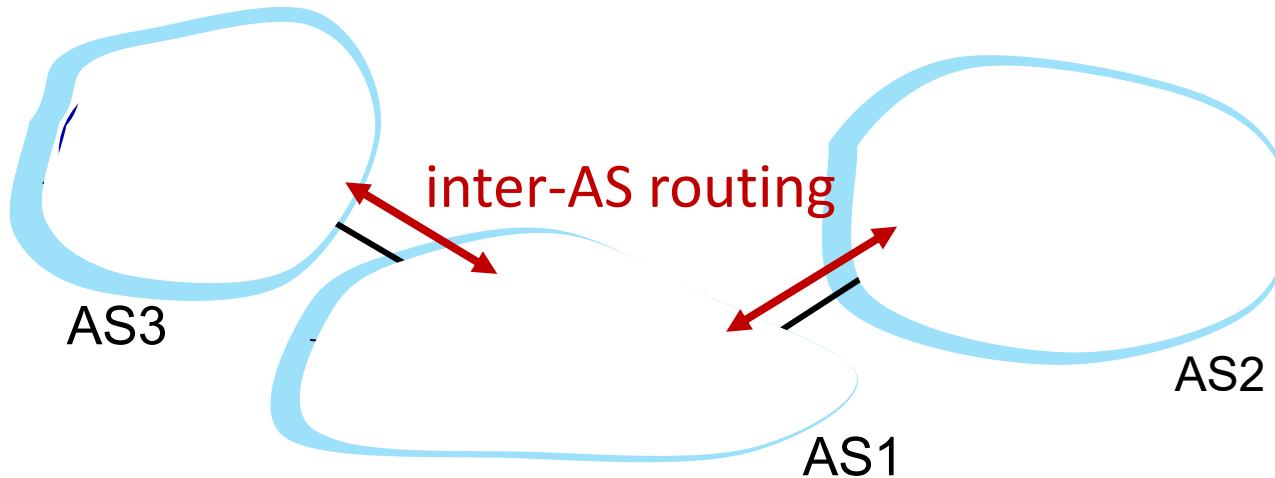
Livello di rete: tabella di marcia del “piano di controllo”

- 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

Sistemi autonomi (AS) interconnessi

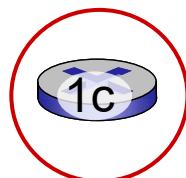
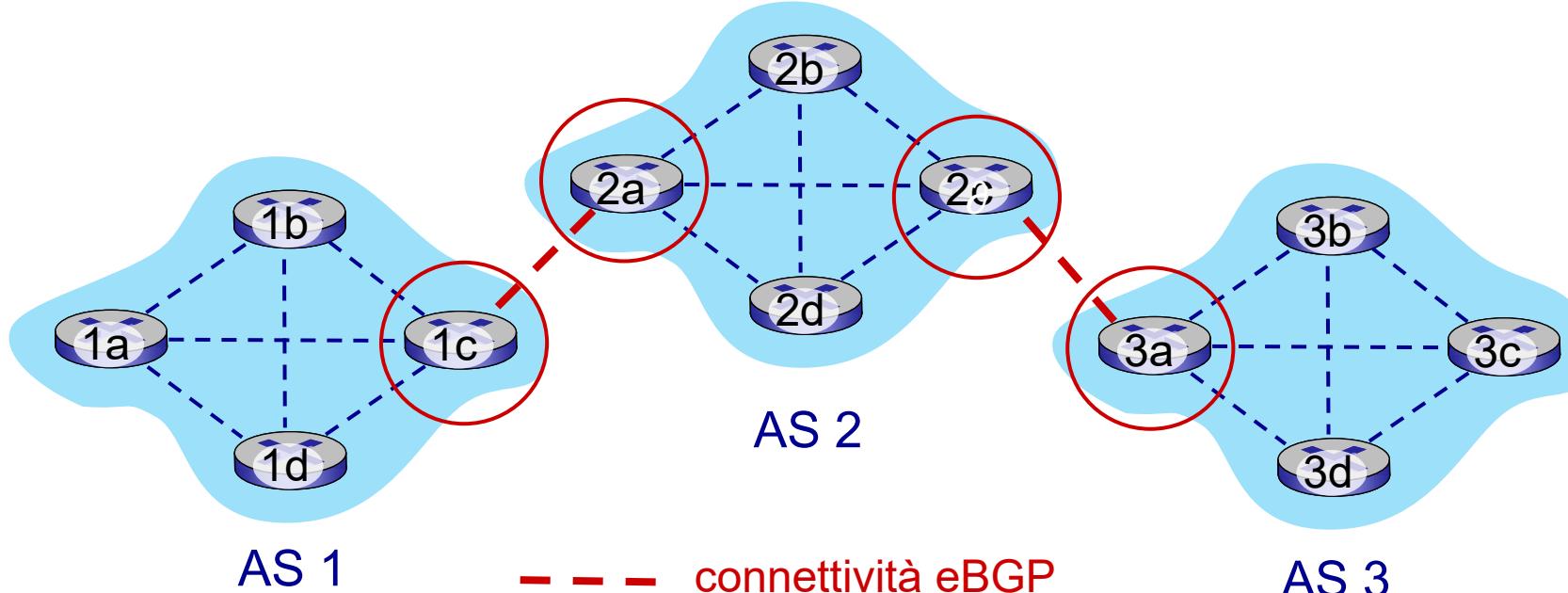


- ✓ intra-AS (o “intra-domain”): instradamento tra router *dentro lo stesso AS (“rete”)*
- inter-AS (o “inter-domain”): instradamento *tra AS*

Instradamento Internet inter-AS: BGP

- **BGP (Border Gateway Protocol)**: *il protocollo di fatto per l'instradamento inter-domain*
 - “colla che tiene insieme Internet”
- permette alla sottorete di pubblicizzare la sua esistenza e le destinazioni che può raggiungere al resto di Internet: *“Io sono qui, ecco chi posso raggiungere e come”*
- BGP fornisce a ciascun AS un mezzo per:
 - ottenere informazioni sulla raggiungibilità dei prefissi di sottorete da parte dei sistemi confinanti (**eBGP**)
 - determinare le rotte verso altre reti sulla base delle informazioni di raggiungibilità e di *politiche (policy)*
 - propagare le informazioni di raggiungibilità a tutti i router interni all'AS (**iBGP**)
 - **annunciare** (alle reti confinanti) le informazioni sulla raggiungibilità delle destinazioni

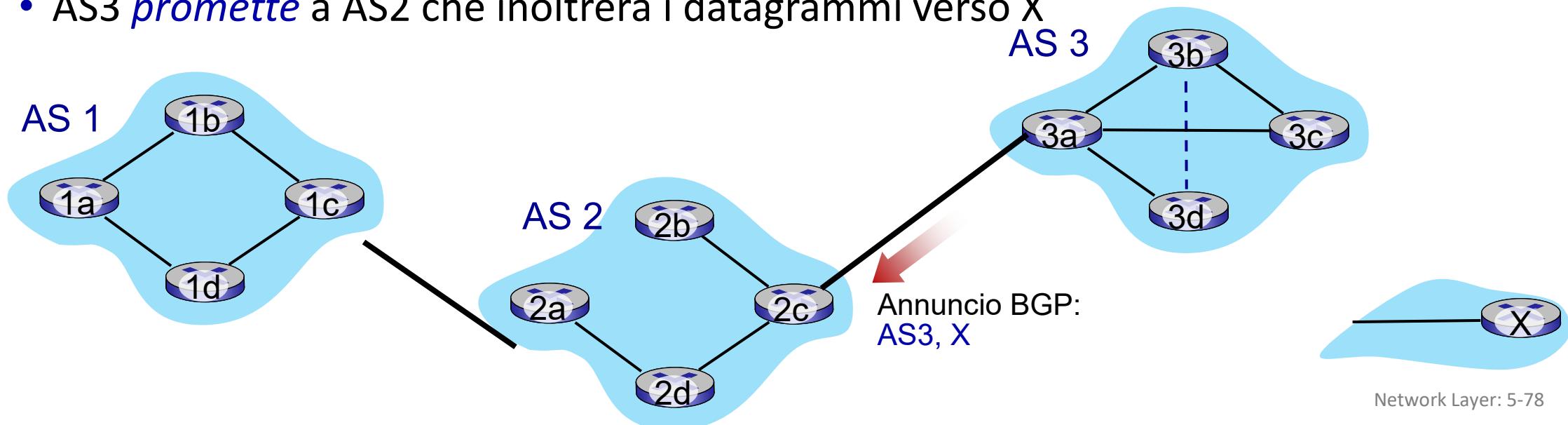
Connessioni eBGP, iBGP



router gateway eseguono sia il protocollo eBGP sia il protocollo iBGP

Nozioni di base su BGP

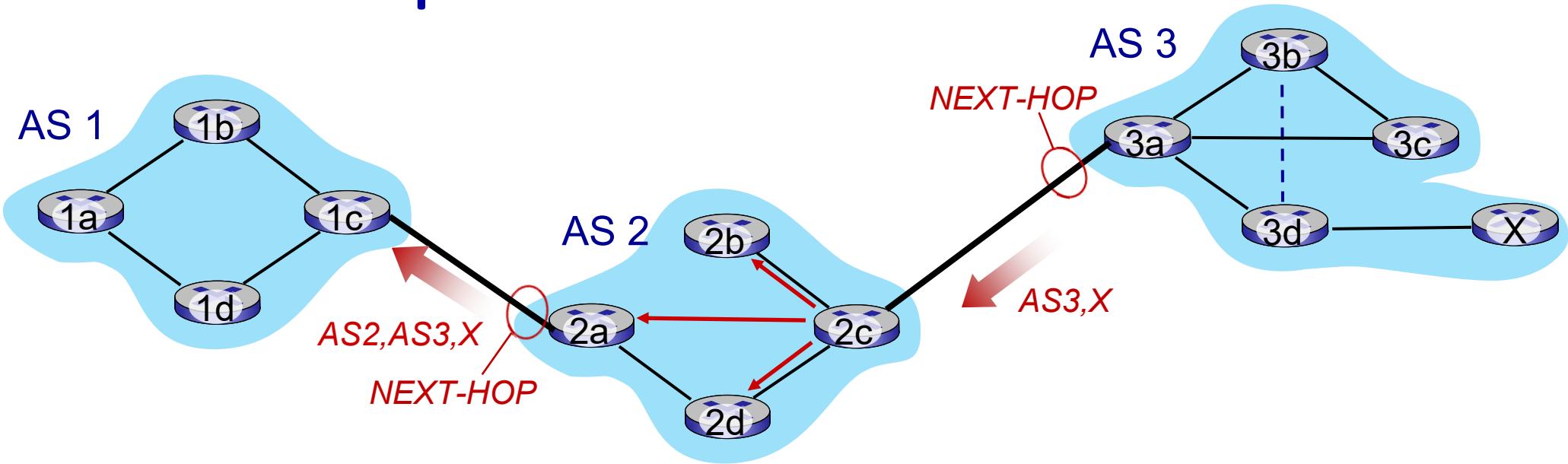
- **Sessione BGP:** due router BGP (“peers”) si scambiano messaggi BGP attraverso un connessione TCP semi-permanente:
 - annunciare *percorsi* verso diversi prefissi di rete (BGP è un protocollo “path vector”; ciò permette di evitare cicli, perché un AS non accetterà un percorso che lo vede già coinvolto)
- Quando il gateway 3a di AS3 annuncia il **percorso AS3,X** al gateway 2c di AS2:
 - AS3 *promette* a AS2 che inoltrerà i datagrammi verso X



Attributi dei percorsi e rotte BGP

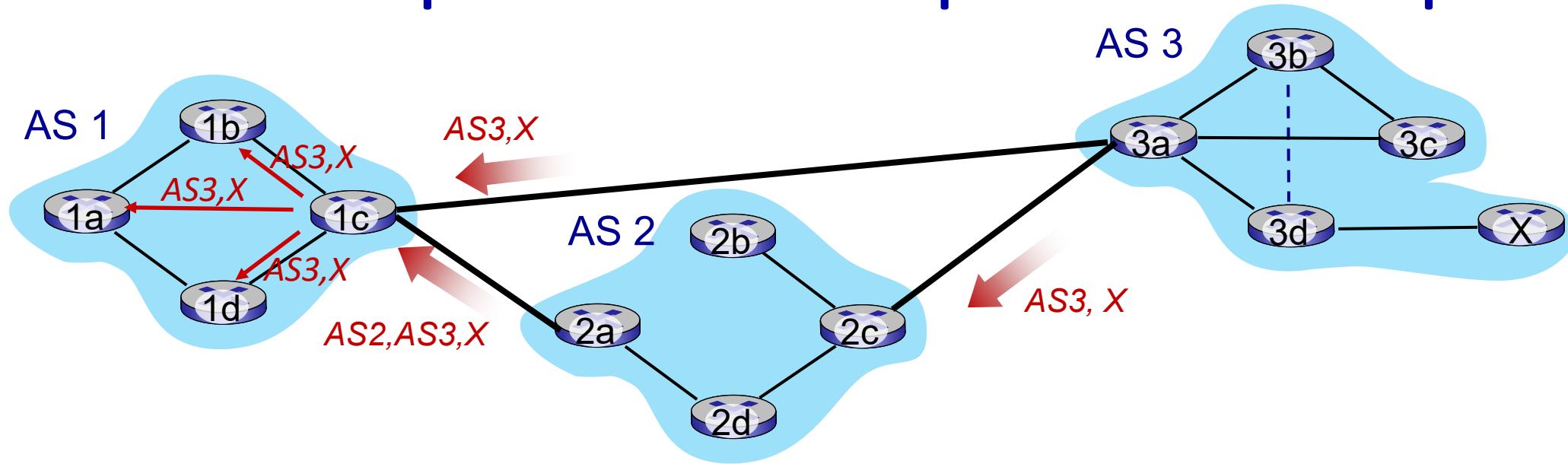
- Rotta (*route*) annunciata da BGP: prefisso + attributi
 - prefisso: la destinazione che viene annunciata
 - due attributi importanti:
 - AS-PATH: elenco degli AS attraverso i quali è passato l'annuncio del prefisso
 - NEXT-HOP: indirizzo IP dell'interfaccia del router che inizia l'AS-PATH
- **instradamento basato su politiche:**
 - Un gateway che riceve un annuncio di percorso usa una *import policy* per accettare/declinare il percorso (es., mai instradare attraverso AS Y).
 - Le politiche dell'AS determinano anche se *annunciare* un percorso a altri AS vicini

Annuncio di percorso BGP



- il router 2c in AS2 riceve l'annuncio del percorso **AS3,X** (attraverso eBGP) dal router 3a in AS3
- sulla base delle politiche di AS2, il router 2c in AS2 accetta il percorso AS3,X, e lo propaga (attraverso iBGP) a tutti i router in AS2
- sulla base delle politiche di AS2, il router 2a in AS2 annuncia (attraverso eBGP) il percorso **AS2, AS3, X** al router 1c in AS1

Annuncio di percorso BGP: percorsi multipli

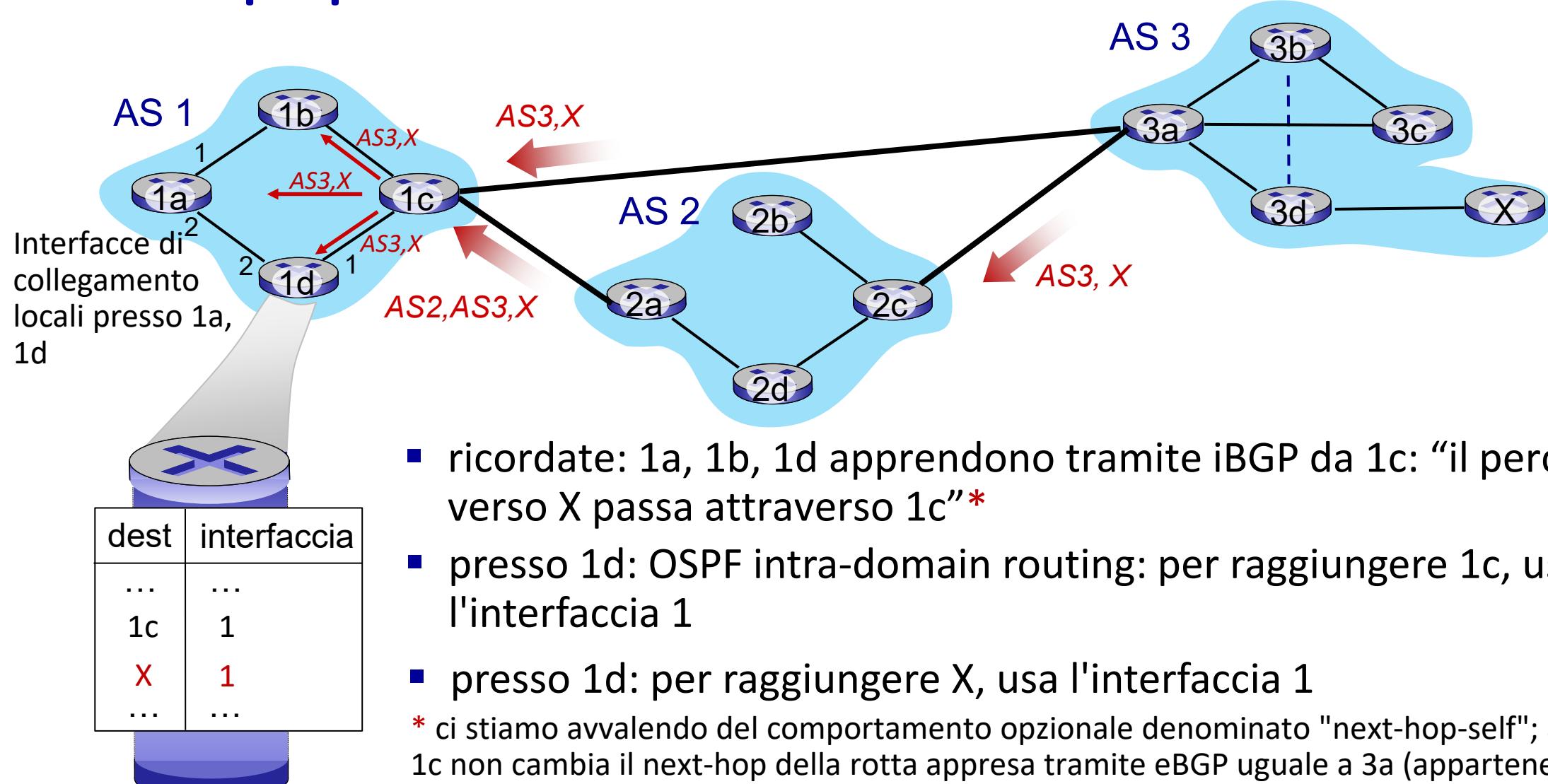


un **router gateway** potrebbe venire a conoscenza di percorsi molteplici verso una certa destinazione:

- il router gateway 1c di AS1 apprende il percorso **AS2,AS3,X** da 2a
- il router gateway 1c di AS1 apprende il percorso **AS3,X** a 3a
- sulla base di **politiche**, il router gateway 1c in AS1 sceglie il percorso **AS3,X** e annuncia il percorso dentro l'AS attraverso iBGP

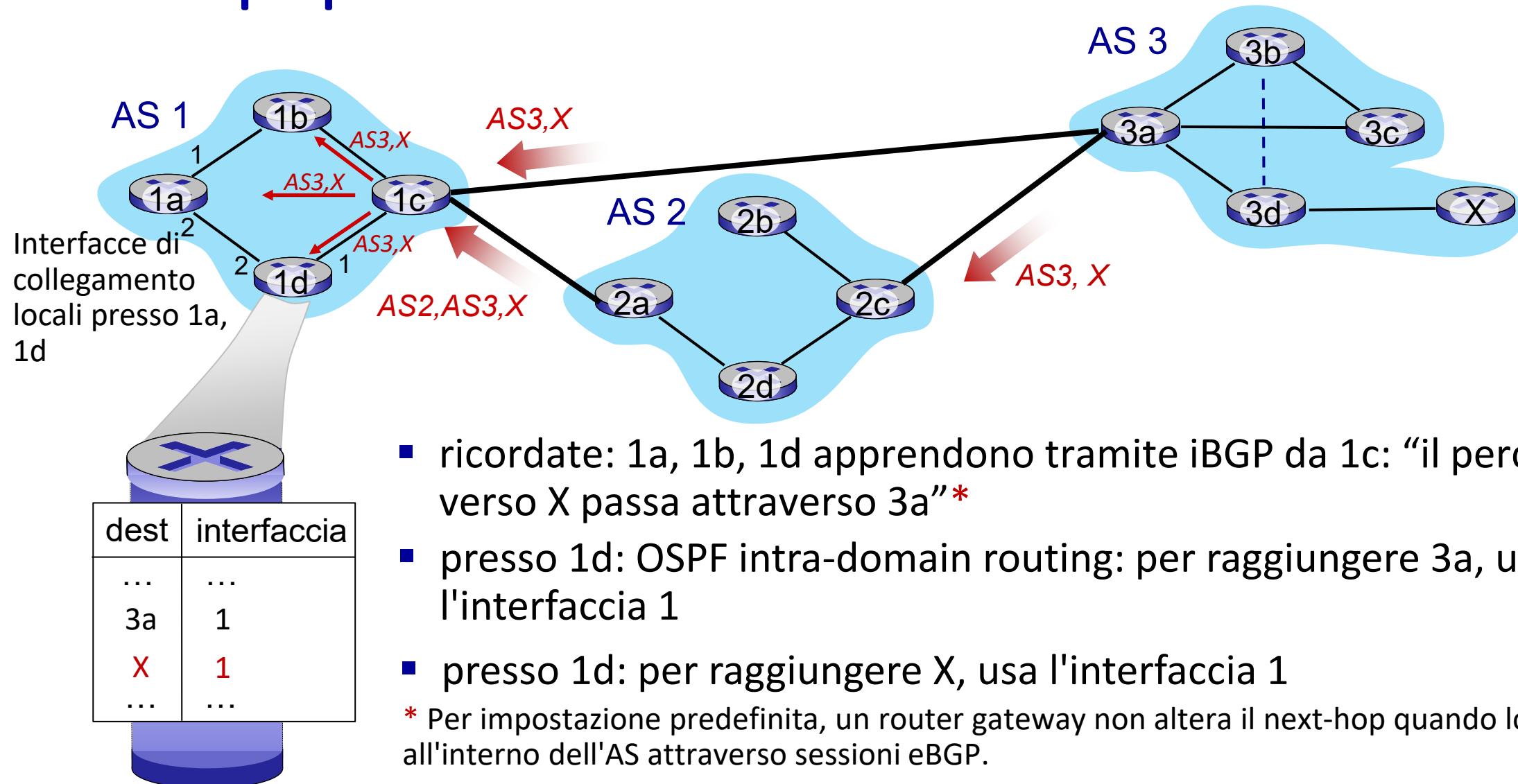
BGP: popolare le tabelle di inoltro

(semplificato con next-hop-self)



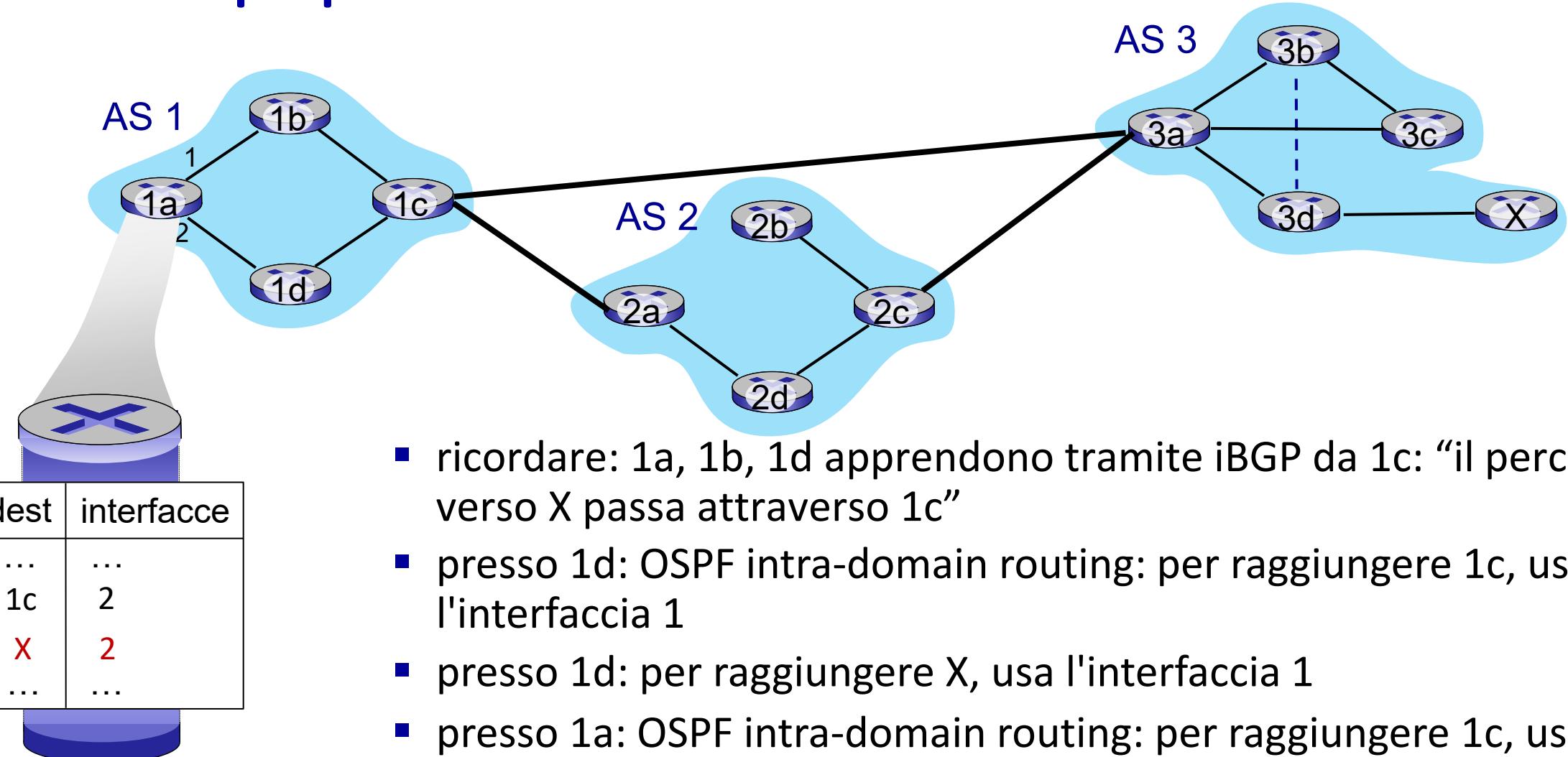
- ricordate: 1a, 1b, 1d apprendono tramite iBGP da 1c: “il percorso verso X passa attraverso 1c”*
 - presso 1d: OSPF intra-domain routing: per raggiungere 1c, usa l'interfaccia 1
 - presso 1d: per raggiungere X, usa l'interfaccia 1
- * ci stiamo avvalendo del comportamento opzionale denominato "next-hop-self"; altrimenti, 1c non cambia il next-hop della rotta appresa tramite eBGP uguale a 3a (appartenente a una sottorete esterna, ma direttamente connessa ad AS1), affidandosi al fatto che l'instradamento interno a AS1 supporti l'instradamento verso quest'ultima (vedi esempio dopo).

BGP: popolare le tabelle di inoltro



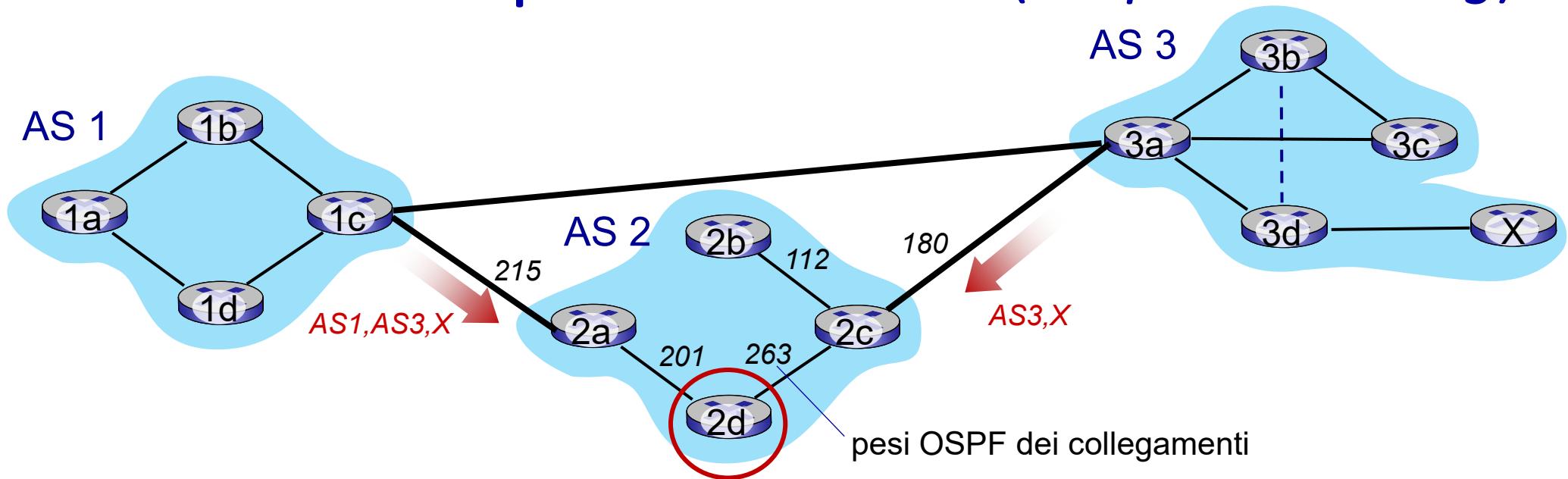
BGP: popolare le tabelle di inoltro

(semplificato con next-hop-self)



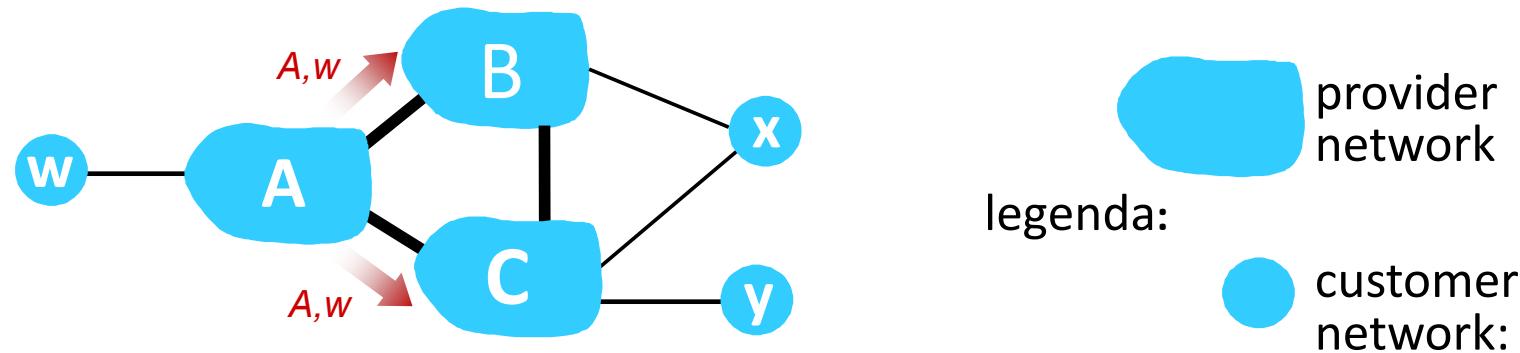
- ricordare: 1a, 1b, 1d apprendono tramite iBGP da 1c: “il percorso verso X passa attraverso 1c”
- presso 1d: OSPF intra-domain routing: per raggiungere 1c, usa l'interfaccia 1
- presso 1d: per raggiungere X, usa l'interfaccia 1
- presso 1a: OSPF intra-domain routing: per raggiungere 1c, usa l'interfaccia 2
- presso 1a: per raggiungere X, usa l'interfaccia 2

Instradamento a patata bollente (*hot potato routing*)



- 2d apprende (tramite iBGP) che può instradare verso X via 1c o 3a*
- **instradamento a patata bollente**: sceglie la rotta con router NEXT-HOP che ha il minimo costo *intra-AS* (es., 2d sceglie 1c, nonostante il maggior numero di hop verso X): non si preoccupa del costo complessivo del percorso!

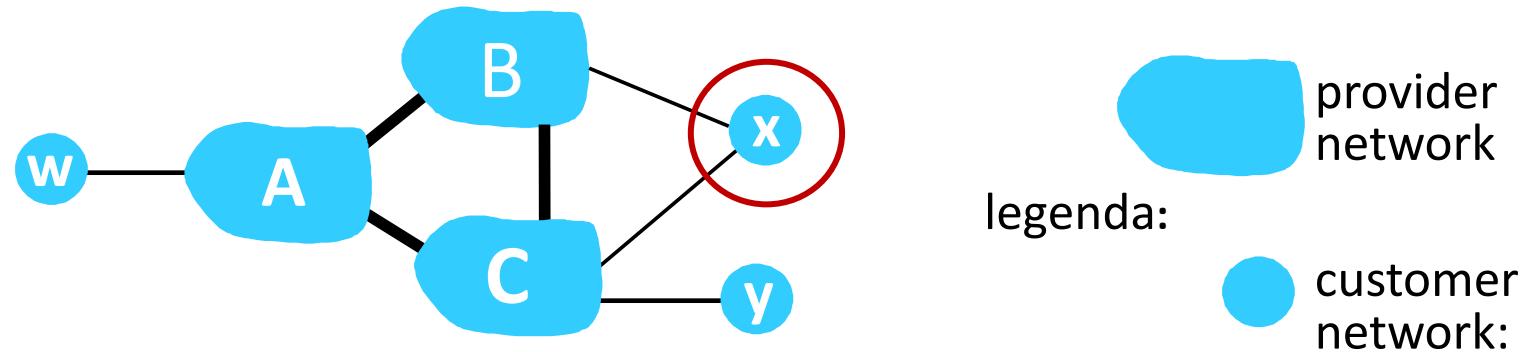
BGP: implementare le politiche attraverso gli annunci



L'ISP vuole instradare il traffico solo verso/da le reti dei propri clienti (non vuole trasportare il traffico di transito tra altri ISP - una politica tipica del “mondo reale”)

- A annuncia il percorso Aw a B e a C
- B *scegli di non annunciare BAw a C!*
 - B non riceve alcuna “entrata” per l'instradamento CBAw, visto che né C, A, w sono clienti di B
 - C *non* viene a conoscenza del percorso CBAw
- C instraderà CAw (non usando B) per raggiungere w

BGP: implementare le politiche attraverso gli annunci (+)



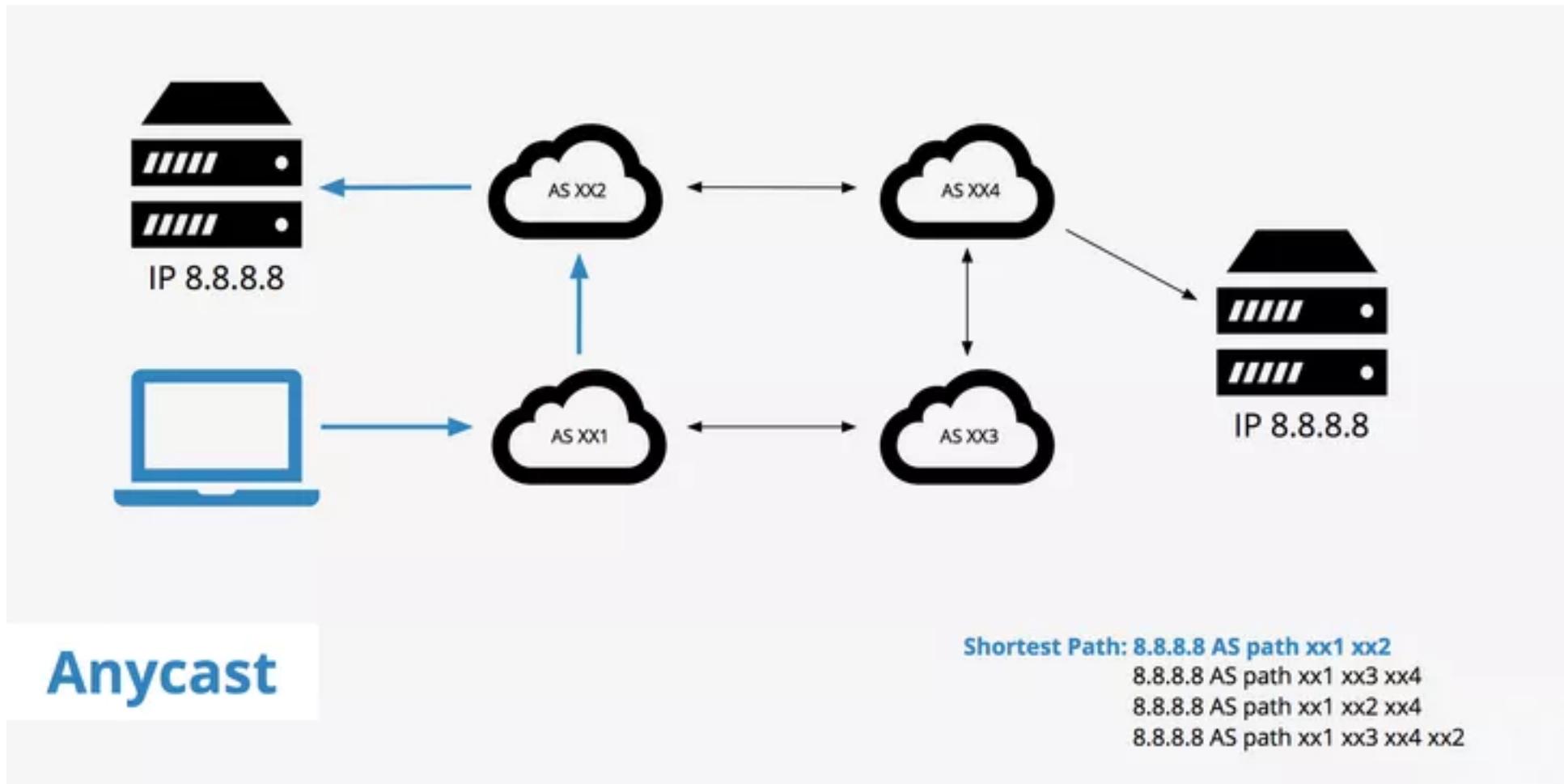
L'ISP vuole instradare il traffico solo verso/da le reti dei propri clienti (non vuole trasportare il traffico di transito tra altri ISP - una politica tipica del “mondo reale”)

- A,B,C sono **provider network**
- x,w,y sono **customer** (delle provider networks)
- x è **dual-homed**: connessa a due reti
- **politica da applicare**: x non vuole instradare da B a C attraverso x
 - .. quindi x non annuncerà a B un percorso verso C

Selezione delle rotte BGP

- Un **router** può conoscere più di un percorso verso l'AS di destinazione, seleziona il percorso in base a:
 1. valore dell'attributo di **preferenza locale**: decisione politica
 2. AS-PATH più breve
 3. router NEXT-HOP più vicino: instradamento a patata bollente
 4. identificatori BGP

IP Anycast



Fonte: <https://www.keycdn.com/support/anycast>



As of 2025-04-30T00:19:56Z, the root server system consists of 1936 instances operated by the 12 independent root server operators.

The 13 root name servers are operated by 12 independent organisations.

You can find more information about each of these organisations by visiting their homepage as found in the 'Operator' field below.

Technical questions about the Root Server System as a whole can be directed to the [Ask RSSAC e-mail address](#).

Visualisations produced from RSSAC002 data submitted by the root server operators can be viewed at [rssac002.root-servers.org](#)

Perché diversi instradamenti Intra- e Inter-AS?

politiche:

- inter-AS: l'amministratore vuole avere il controllo sul modo in cui viene instradato il suo traffico **e** su chi passa attraverso la sua rete
- intra-AS: singolo amministratore, quindi le politiche sono meno rilevanti

scalabilità:

- routing gerarchico (basato sulla distinzione tra instradamento intra-AS e inter-AS): limita l'ambito delle informazioni topologiche dettagliate al singolo AS (in realtà, a una singola area se si considera OSPF gerarchico)
- instradamento BGP verso prefissi per supportare un gran numero di destinazioni (sfruttando anche l'aggregazione degli indirizzi)

prestazioni:

- intra-AS: può concentrarsi sulle prestazioni
- inter-AS: le politiche sono dominanti rispetto alla prestazioni

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



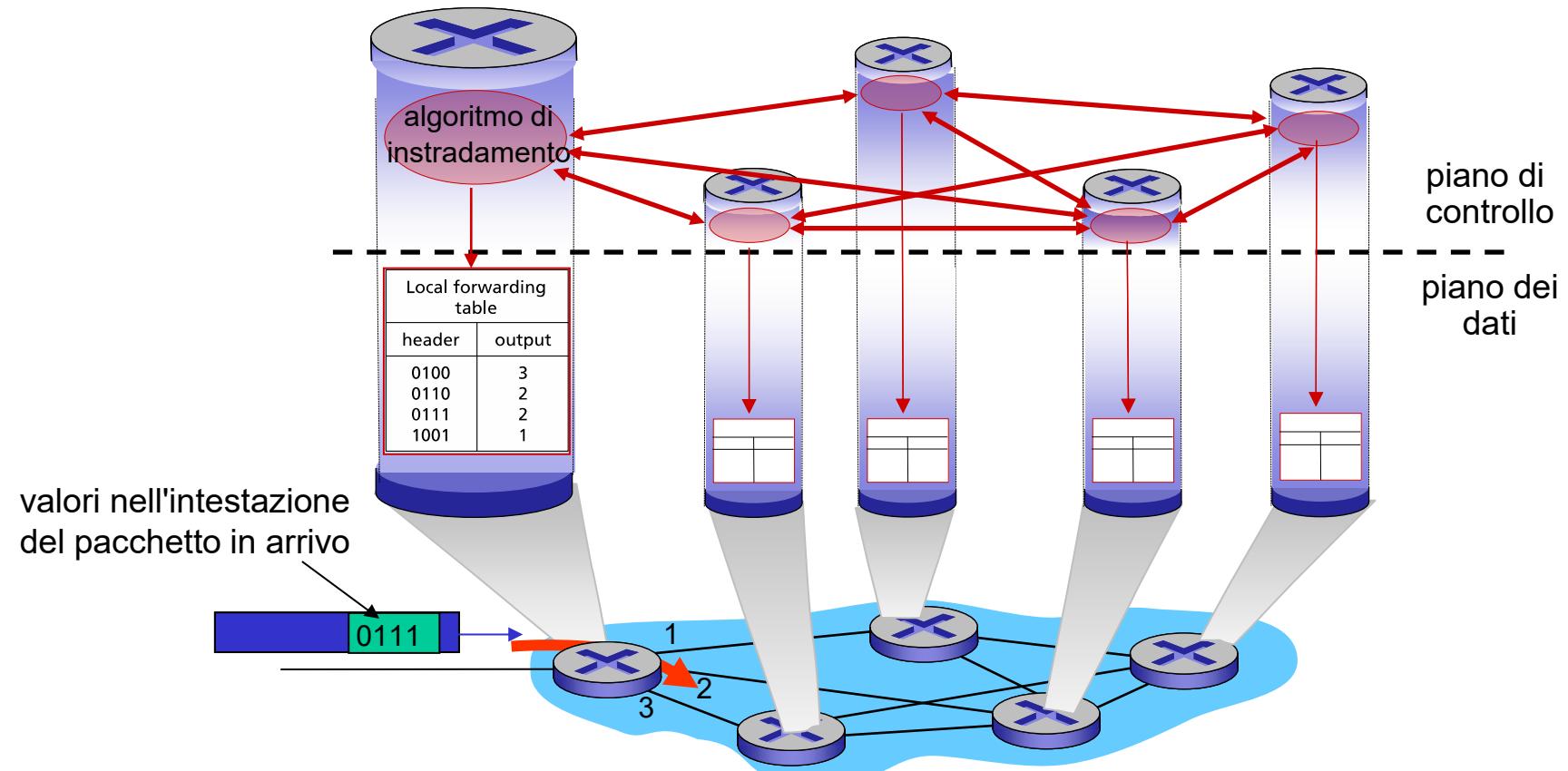
- gestione e configurazione della rete
 - SNMP
 - NETCONF/YANG

Software defined networking (SDN)

- Livello di rete di Internet: storicamente implementato tramite un approccio di controllo distribuito e per router:
 - Un *router monolitico* contiene l'hardware di commutazione (*switching*), esegue una implementazione proprietaria dei protocolli standard di Internet (IP, RIP, IS-IS, OSPF, BGP) in un sistema operativo proprietario specializzato per dispositivi di rete (es. Cisco IOS)
 - “middlebox” differenti per differenti funzioni del livello di rete: firewalls, load balancers, NAT, ..
- ~2005: rinnovato interesse nel ripensare il piano di controllo della rete

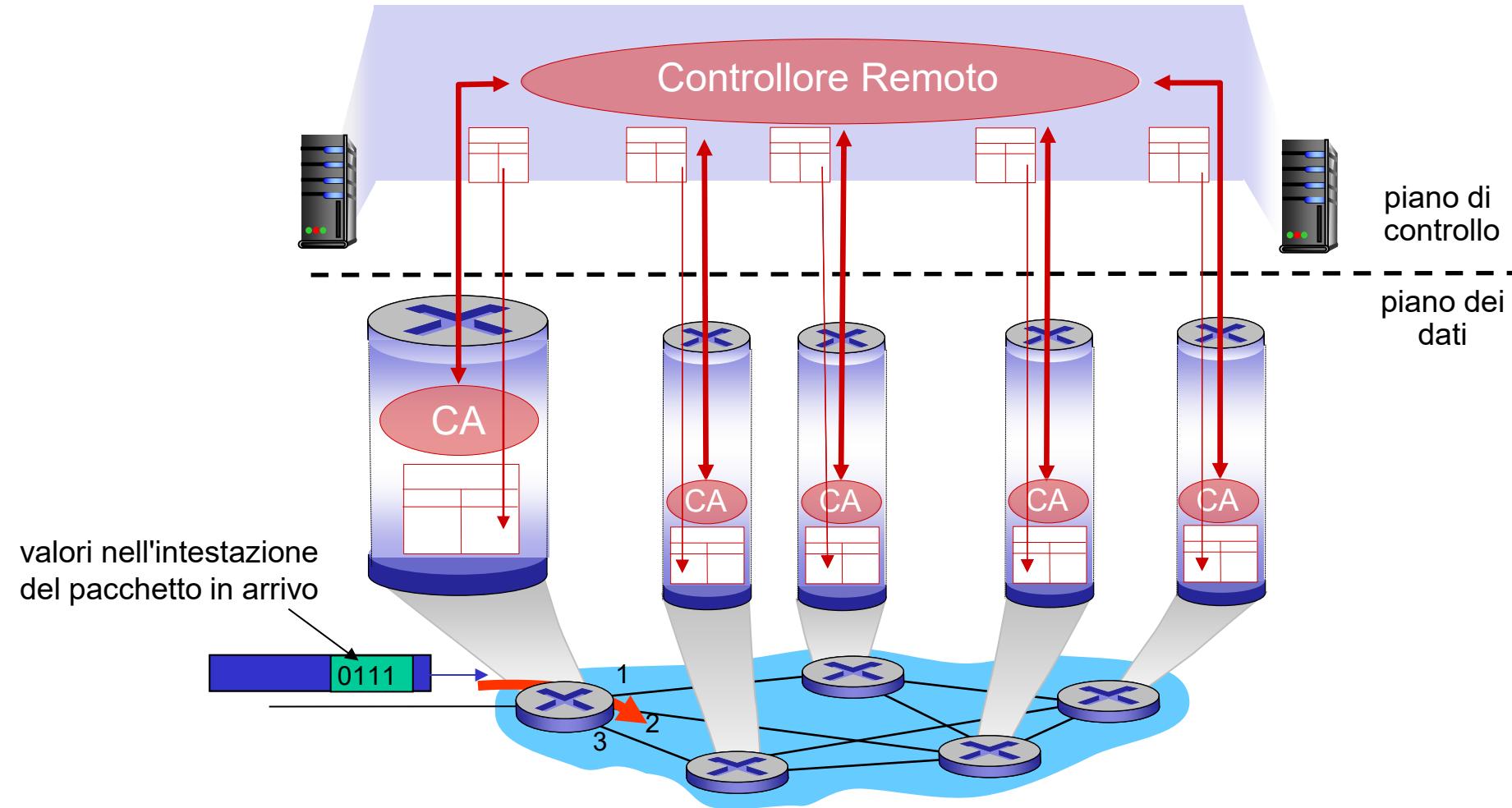
Piano di controllo per rotuer

I singoli componenti dell'algoritmo di instradamento *in ogni 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.



Software defined networking (SDN)

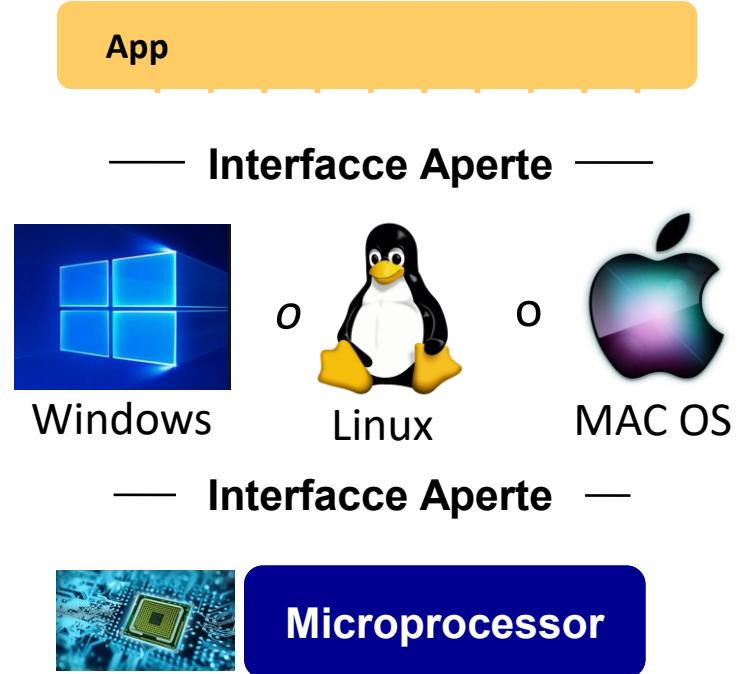
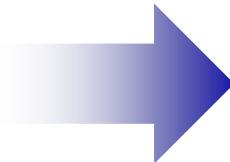
Perché un piano di controllo *logicamente centralizzato*?

- gestione più semplice della rete: evitare errori di configurazione dei router, maggiore flessibilità dei flussi di traffico
- inoltro basato su tabelle dei flussi (ricordate OpenFlow API) permette la "programmazione" dei router
 - la "programmazione" centralizzata è più semplice: calcola le tabelle centralmente e poi distribuisce
 - la "programmazione" distribuita è più difficile: calcolo delle tabelle come risultato di un algoritmo (protocollo) distribuito implementato in ogni singolo router
- implementazione aperta (non proprietaria) del piano di controllo
 - promuovere l'innovazione

Analogia con l'SDN: dal mainframe alla rivoluzione del PC



Integrato verticalmente
chiuso, proprietario
Innovazione lenta
Una piccola industria



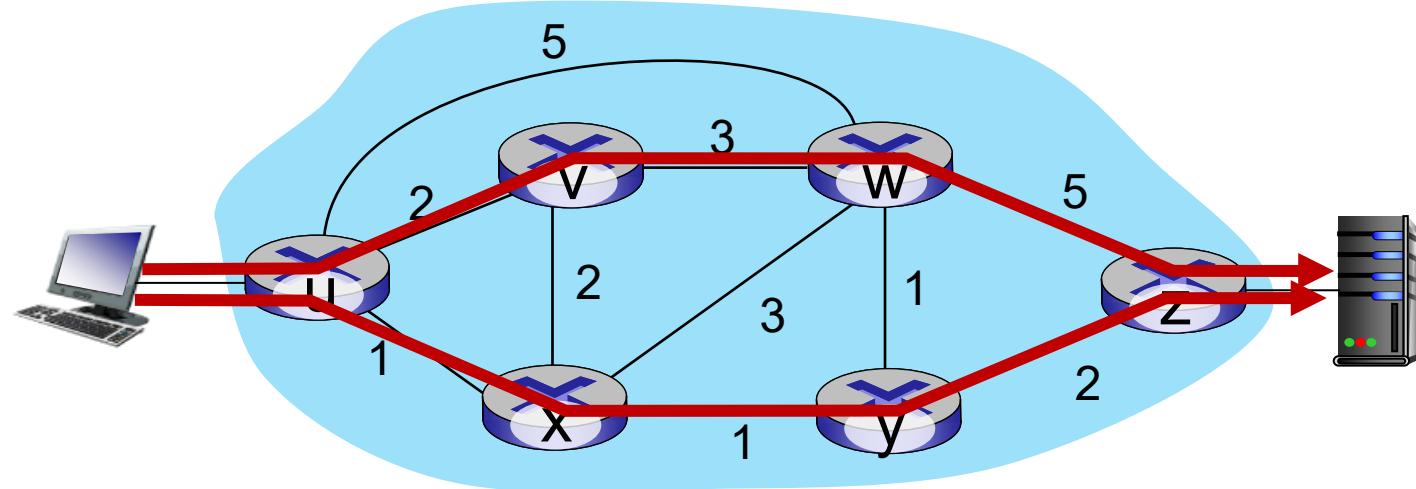
Orizzontale
Interfacce aperte
Innovazione rapida
Un'industria enorme

Ingegneria del traffico

L'**ingegneria del traffico** (TE) si occupa dell'ottimizzazione delle prestazioni delle reti in esercizio. In generale, comprende l'applicazione della tecnologia e dei principi scientifici alla misurazione, alla modellazione, alla **caratterizzazione e** al **controllo del traffico** Internet, e l'applicazione di tali conoscenze e tecniche per **raggiungere specifici obiettivi di prestazione**.

Traduzione da RFC 2702

Ingegneria del traffico: difficile con il routing tradizionale

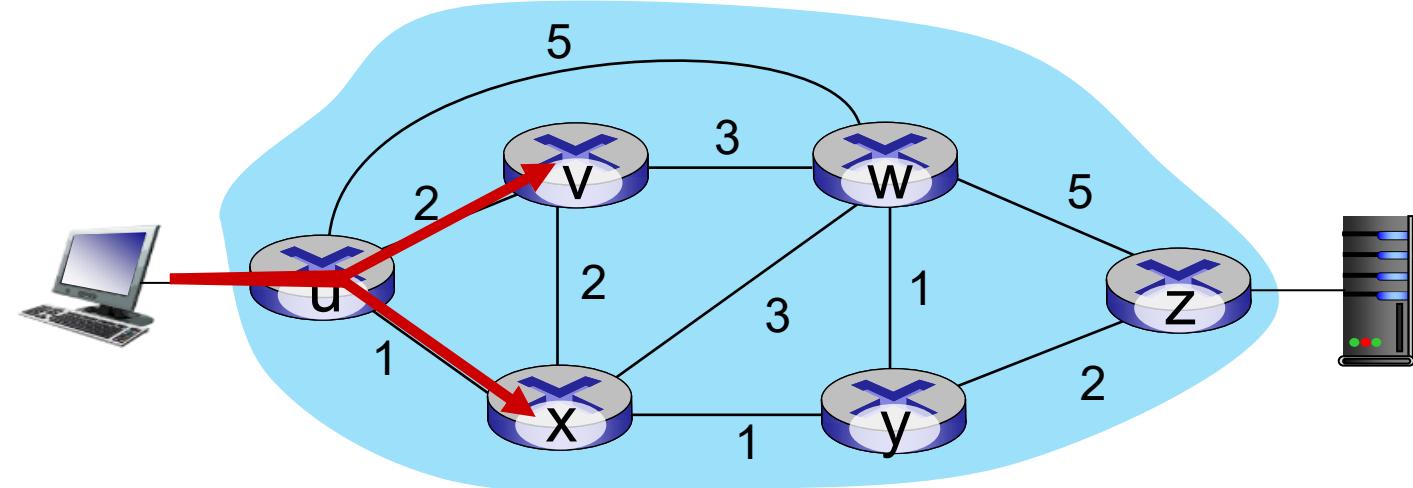


D: cosa succede se l'operatore di rete vuole che il traffico da u a z fluisca lungo $uvwz$, anziché $uxyz$?

R: è necessario ridefinire i pesi dei collegamenti in modo che l'algoritmo di instradamento del traffico calcoli le rotte di conseguenza (o necessitiamo di un nuovo algoritmo di instradamento)!

I pesi dei collegamenti sono le solo “manopole” di controllo: non c'è molto controllo!

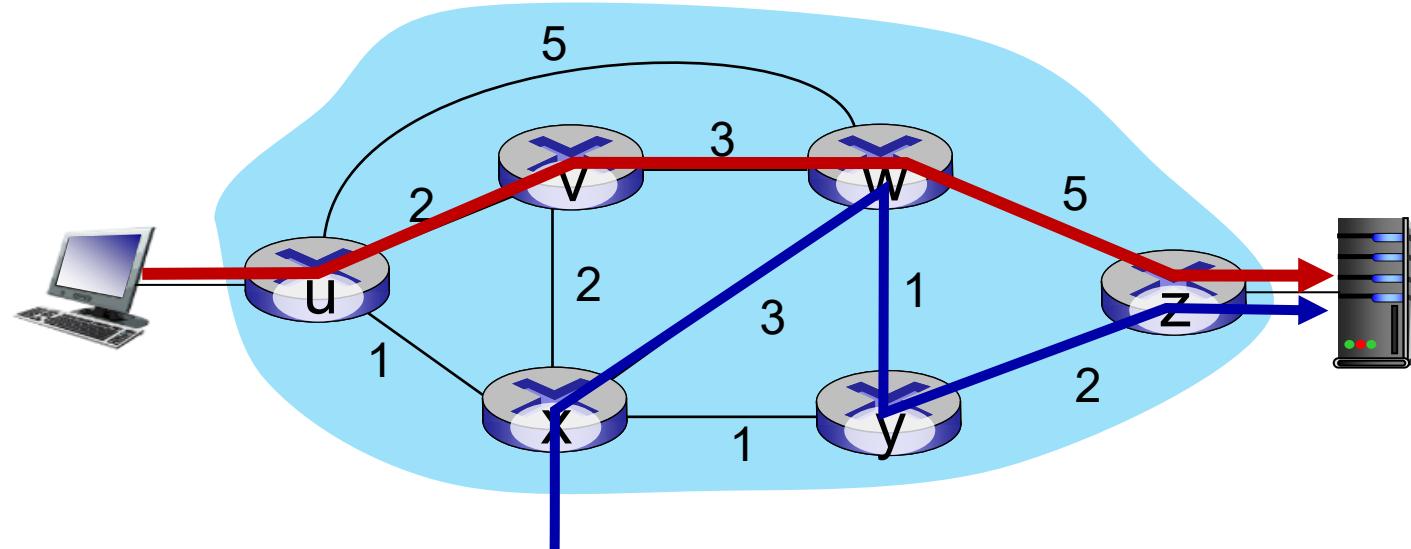
Ingegneria del traffico: difficile con il routing tradizionale



D: cosa succede se l'operatore di rete vuole dividere il traffico u -to- z lungo $uvwz$ e $uxyz$ (bilanciamento del carico)?

R: non può farlo (o ha bisogno di un nuovo algoritmo di routing)

Ingegneria del traffico: difficile con il routing tradizionale



D: e se w volesse instradare il traffico blu e rosso in modo diverso da w a z ?

R: non può farlo (con l'inoltro basato sulla destinazione e l'instradamento LS e DV).

Abbiamo appreso che l'inoltro generalizzato e l'SDN possono essere usati per raggiungere *qualsiasi* instradamento si desideri

Ingegneria del traffico: estensioni in OSPF

OSPF è stato esteso con la possibilità di annunciare informazioni utili al traffic engineering (TE), quali banda disponibile, ritardo, jitter, perdita, etc.

Va però sottolineato che OSPF, di per sé, non sfrutta questi dati per ricalcolare i propri percorsi.

In particolare, tecnologie come MPLS (multiprotocol label switching) – che studieremo più avanti – possono utilizzare queste informazioni. Anziché limitarsi all'inoltro tradizionale basato solo sull'indirizzo di destinazione (secondo i percorsi OSPF), MPLS-TE permette di instradare i flussi su cammini alternativi. In questo modo si tiene conto della congestione di rete e si massimizzano le prestazioni.

Software defined networking (SDN)

4. applicazioni di controllo programmabili

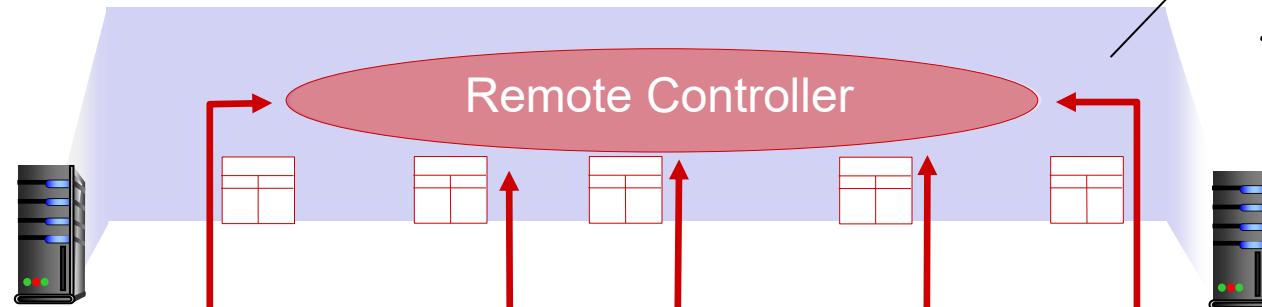
routing

controllo d'accesso

...

bilanciamento del carico

3. Funzioni di controllo di rete: esterne agli switch del piano dei dati



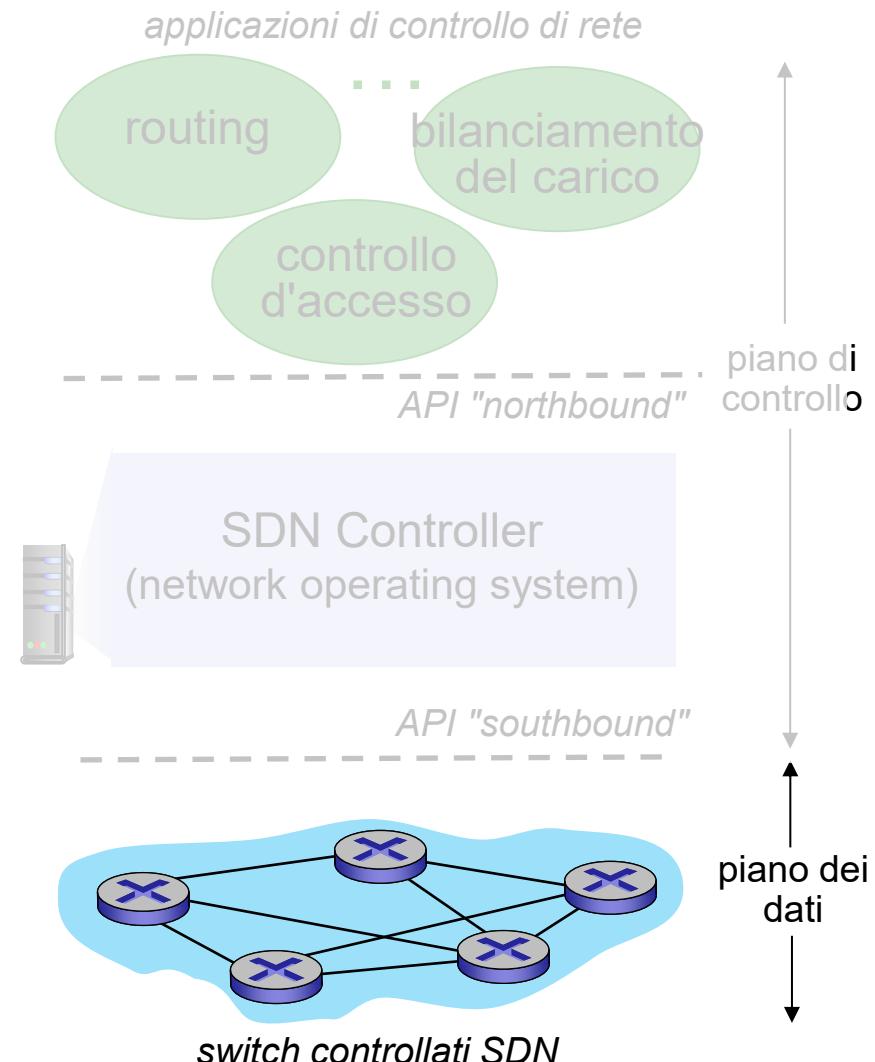
1: inoltro generalizzato "basato sui flussi" (es. OpenFlow)

2. separazione del piano dei dati e del piano di controllo

Software defined networking (SDN)

Switch del piano dei dati :

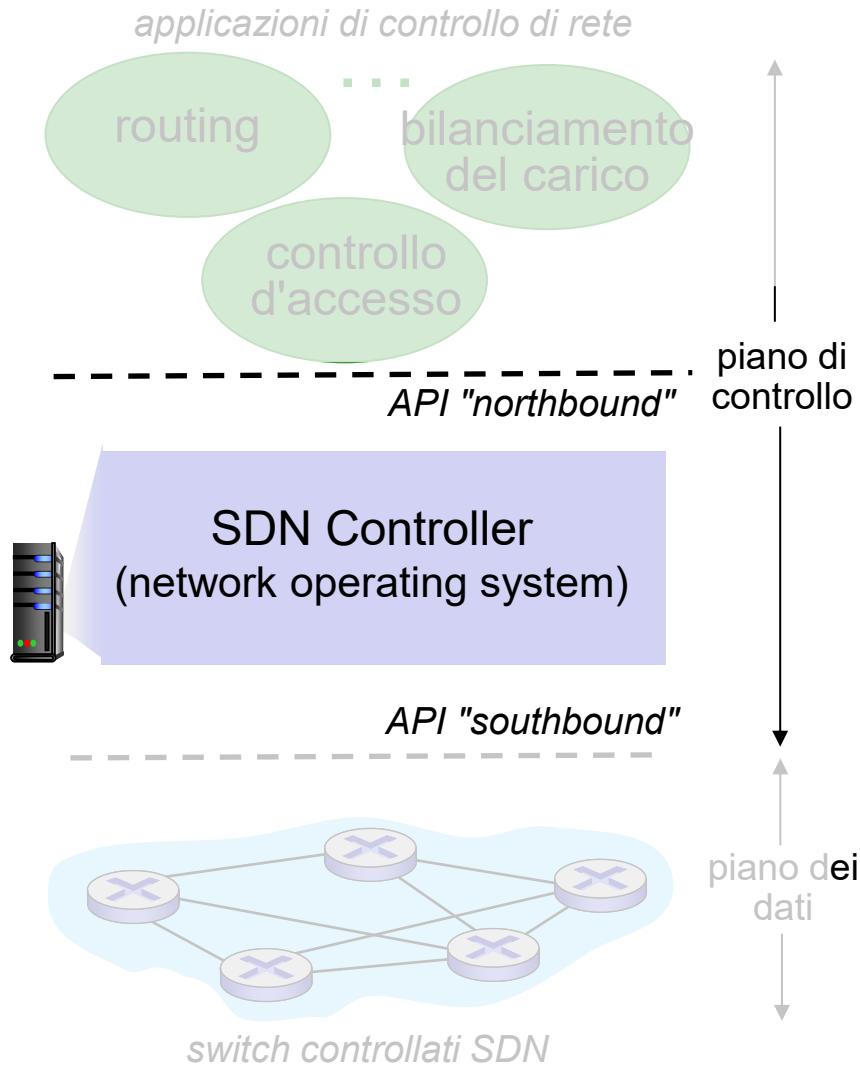
- switch veloci e semplici che implementano l'inoltro generalizzato del piano dei dati in hardware
- tabella dei flussi (inoltro) calcolata, installata sotto la supervisione del controllore
- API per il controllo degli switch basato su tabelle (es., OpenFlow)
 - definisce ciò che è controllabile e ciò che non lo è
- protocollo di comunicazione con il controllore (es. OpenFlow)



Software defined networking (SDN)

SDN controller (network OS):

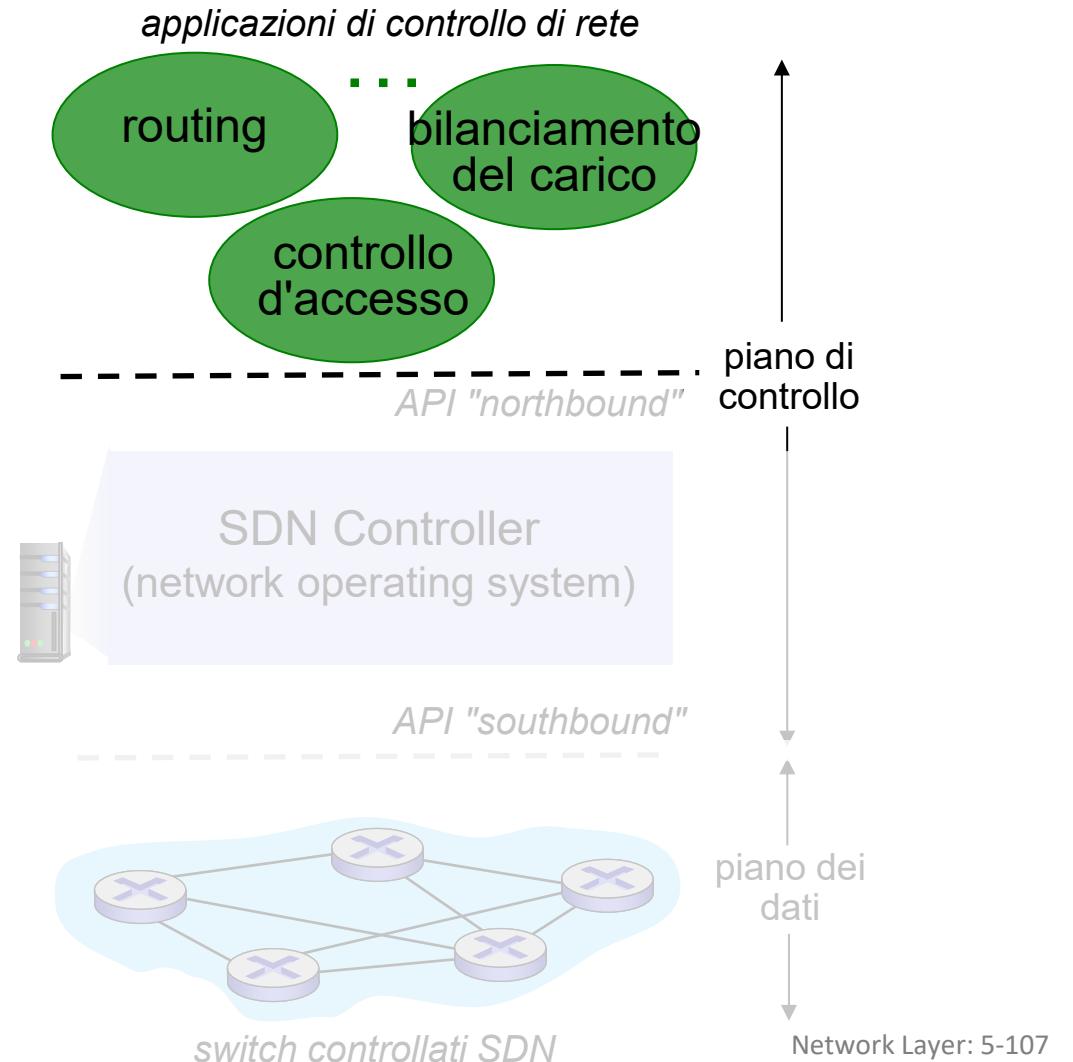
- mantiene le informazioni sullo stato della rete
- interagisce con le applicazioni di controllo della rete “in alto” tramite API “northbound”
- interagisce con gli switch di rete “in basso” tramite API “southbound”
- implementato come sistema distribuito per garantire prestazioni, scalabilità, tolleranza ai guasti, robustezza e sicurezza.



Software defined networking (SDN)

Applicazioni di controllo di rete:

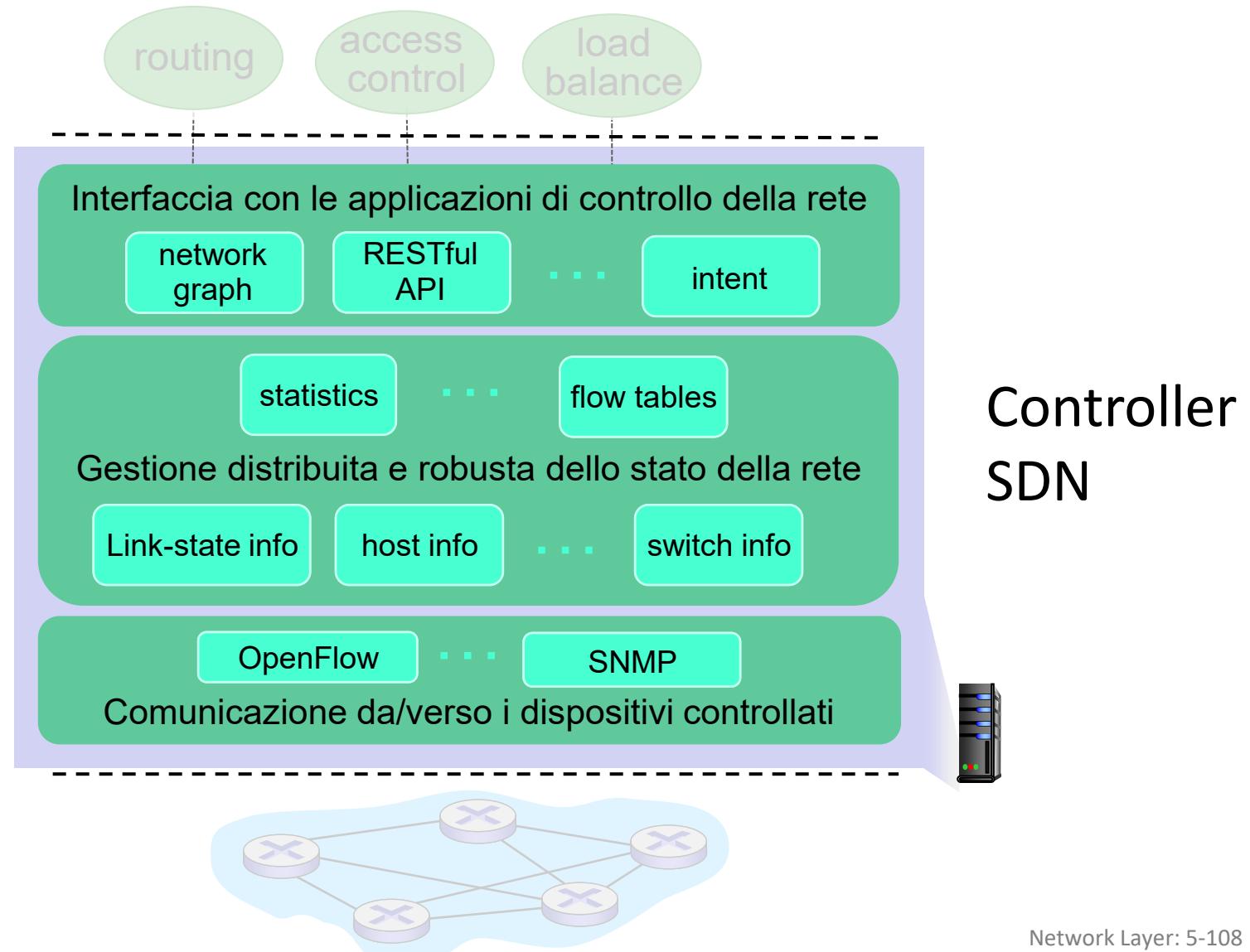
- “cervelli” di controllo: implementano le funzioni di controllo utilizzando servizi di livello inferiore attraverso API fornite dal controller SDN
- *scorporate*: può essere fornito da terzi: distinto dal fornitore di routing o dal controller SDN



Componenti di un Controller SDN

livello di interfaccia con le applicazioni di controllo della rete: astrazioni/API
gestione dello stato della rete: stato dei collegamenti di rete, degli switch, dei servizi: un *database distribuito*

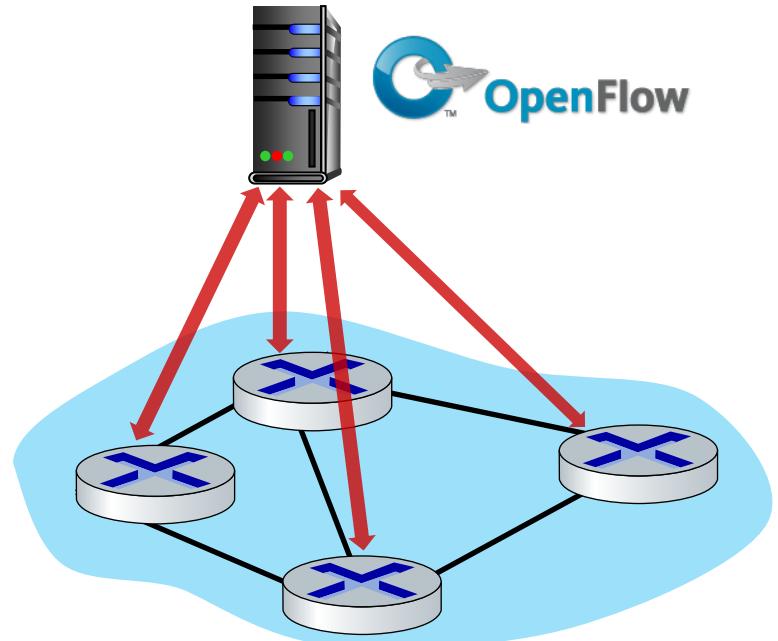
comunicazione: comunicazione tra il controller SDN e gli switch controllati



Protocollo OpenFlow

- opera tra controllore e switch
- TCP utilizzato per lo scambio di messaggi
 - crittografia opzionale
- tre classi di messaggi OpenFlow:
 - controller-to-switch
 - asynchronous (switch to controller)
 - symmetric (misc.)
- distinta dall'API OpenFlow
 - API utilizzata per specificare azioni di inoltro generalizzate

Controller OpenFlow

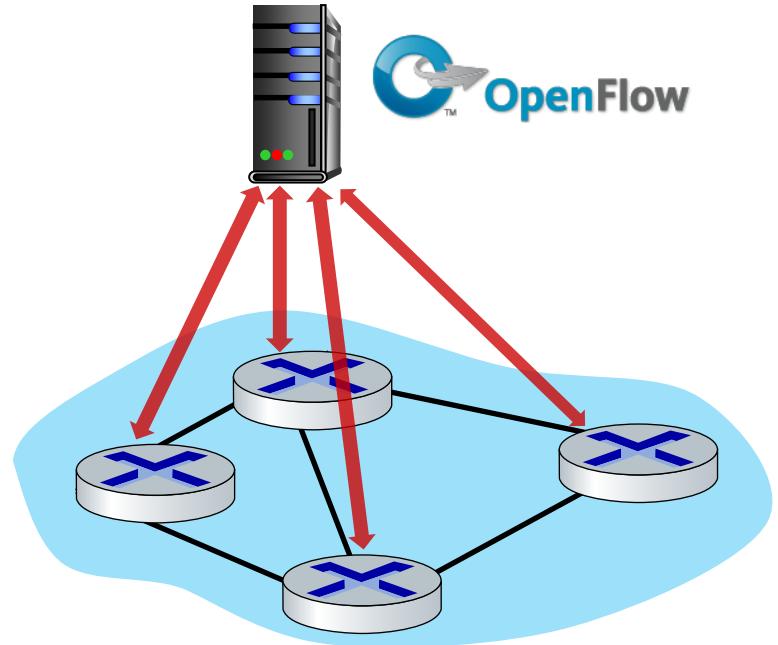


OpenFlow: messaggi controller-to-switch

Messaggi chiave controller-to-switch

- *features*: il controllore interroga le caratteristiche dello switch, lo switch risponde
- *configure*: il controllore interroga/imposta i parametri di configurazione dello switch
- *modify-state*: aggiungere, eliminare, modificare voci di flusso nelle tabelle
- *packet-out*: Il controllore può inviare questo pacchetto da una specifica porta dello switch

Controller OpenFlow

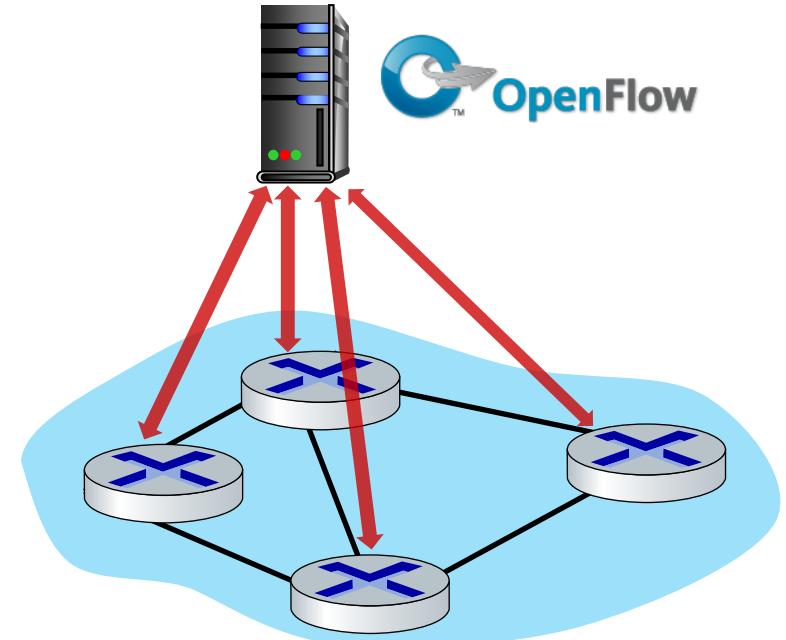


OpenFlow: messaggi switch-to-controller

Messaggi chiave switch-to-controller

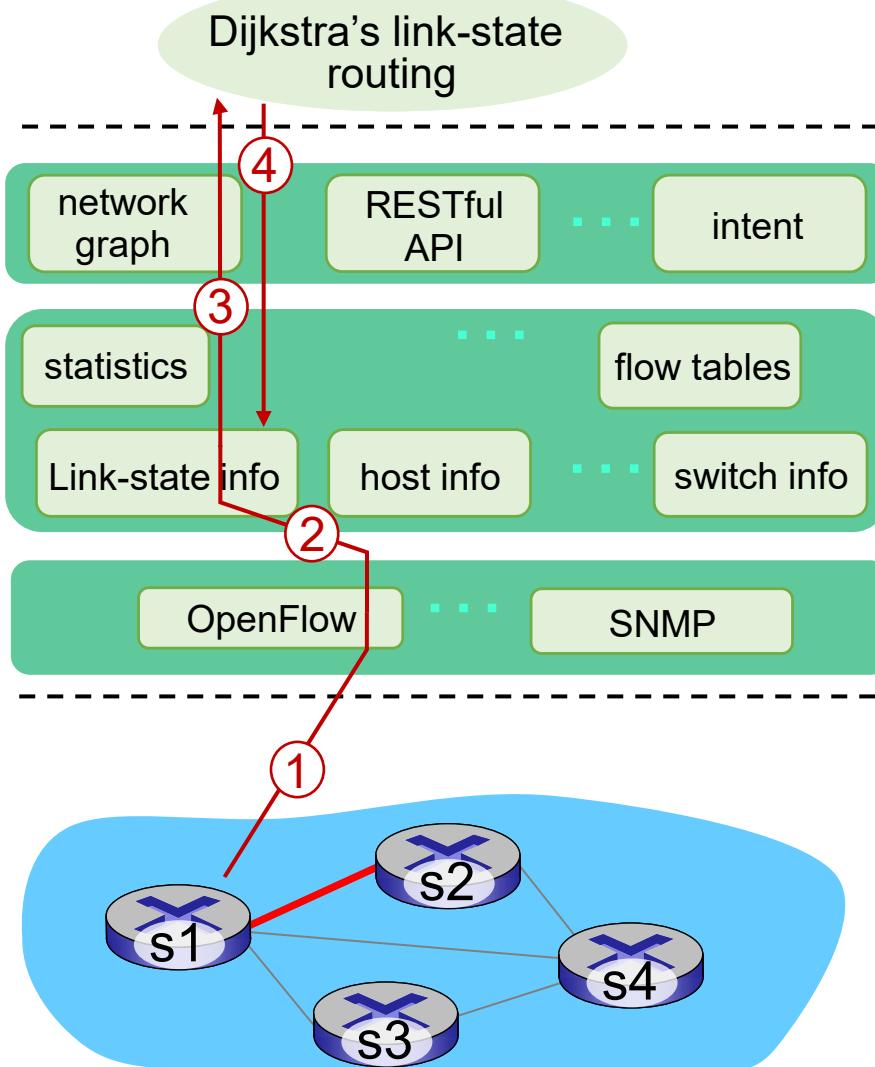
- *packet-in*: trasferire il pacchetto (e il relativo controllo) al controllore. Vedere il messaggio packet-out dal controllore
- *flow-removed*: voce della tabella di flusso cancellata nello switch
- *port status*: informare il controllore di una modifica su una porta.

Controller OpenFlow



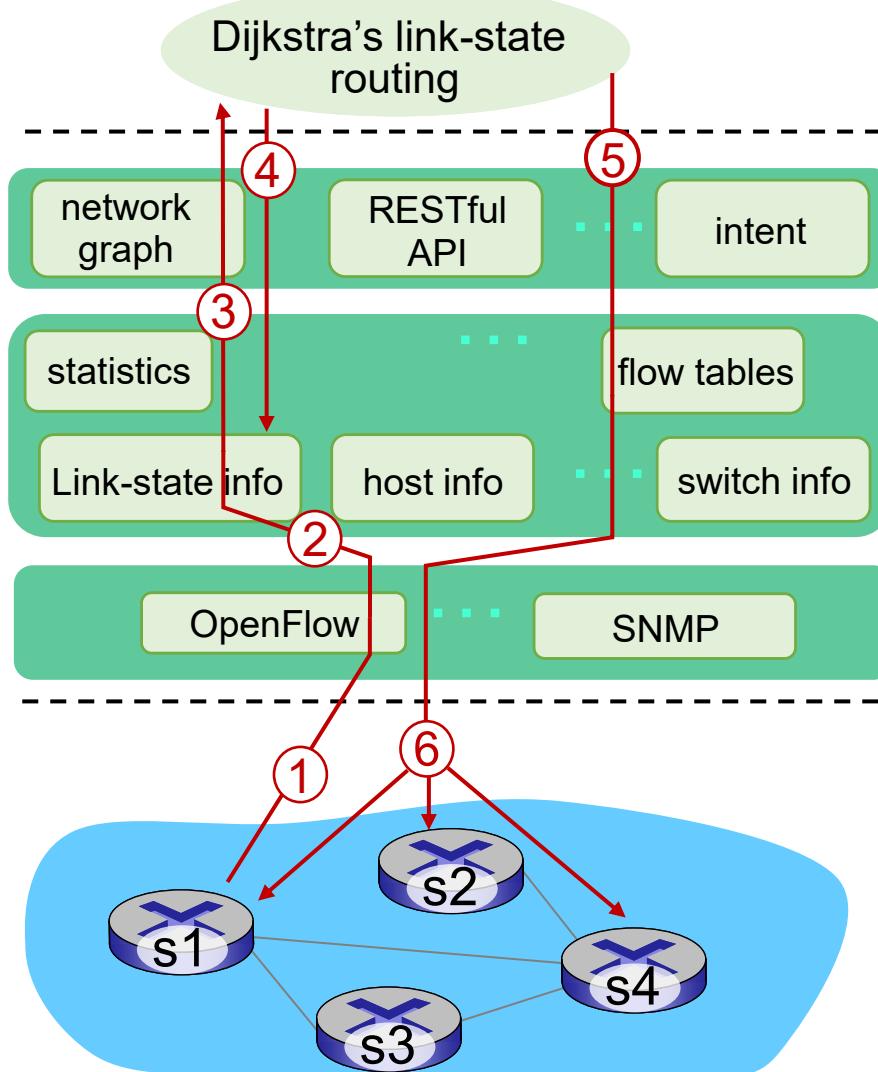
Fortunatamente, gli operatori di rete non “programmano” gli switch creando/inviando direttamente messaggi OpenFlow. Utilizzano invece un’astrazione di livello superiore a livello di controller

SDN: Esempio di interazione tra piano dei dati e piano di controllo



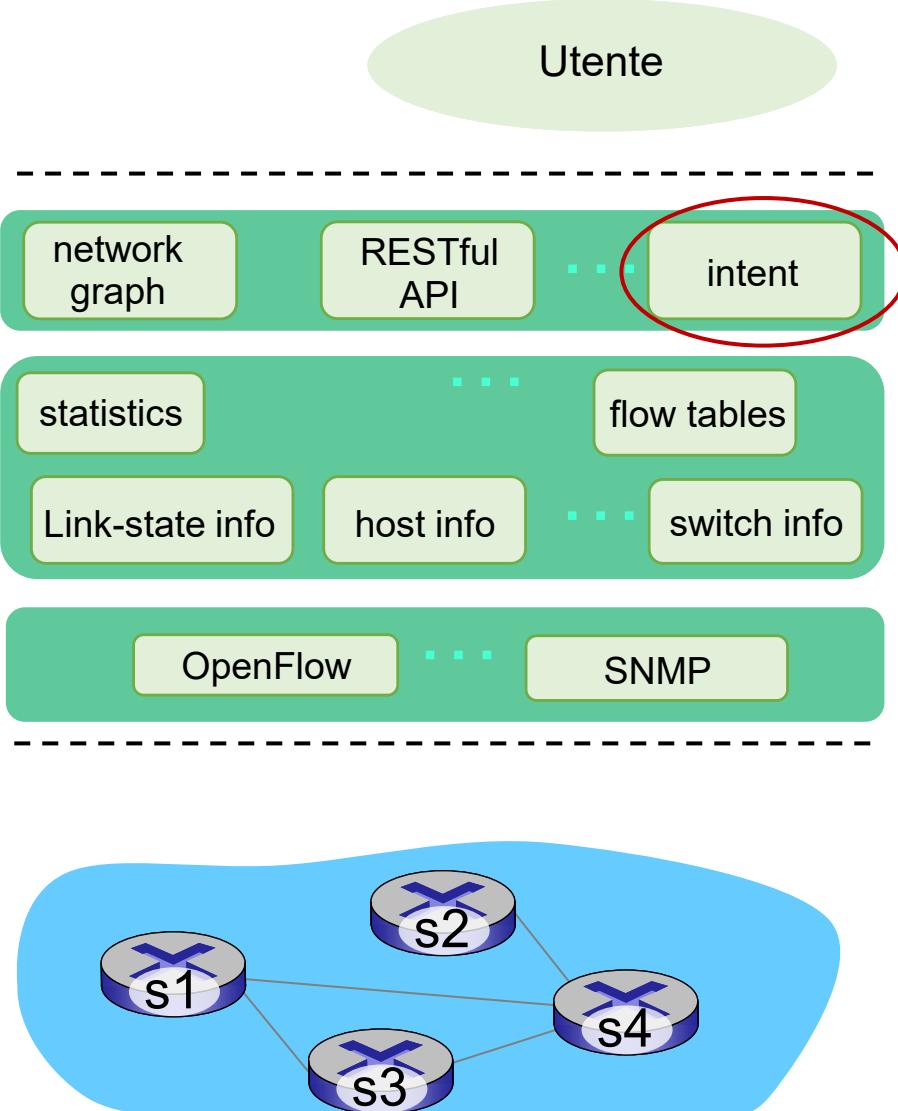
- ① S1, a causa di un guasto del collegamento, utilizza il messaggio di stato della porta OpenFlow per notificare il controllore.
- ② Il controller SDN riceve il messaggio OpenFlow, aggiorna le informazioni sullo stato del collegamento
- ③ L'applicazione dell'algoritmo di routing di Dijkstra si è registrata in precedenza per essere richiamata quando lo stato dei collegamenti cambia. Viene chiamata.
- ④ L'algoritmo di routing di Dijkstra accede alle informazioni sul grafo della rete, alle informazioni sullo stato dei collegamenti nel controllore e calcola nuovi percorsi.

SDN: Esempio di interazione tra piano dei dati e piano di controllo



- ⑤ l'applicazione di link state routing interagisce con il componente flow-table-computation del controller SDN, che calcola le nuove tabelle di flusso necessarie.
- ⑥ il controllore utilizza OpenFlow per installare nuove tabelle negli switch che necessitano di un aggiornamento

SDN: Intent-based networking (IBN)



1. L'utente (es. un amministratore della rete) esprime *in forma dichiarativa* un obiettivo di alto livello (il "cosa"), cioè l'**intento**. Per esempio:
 - "Garantire latenza < 5 ms tra data-center A e B"
 - "Isolare il traffico VoIP dal resto della rete"
2. Il sistema determina "come" realizzare l'obiettivo richiesto, attraverso l'opportuna allocazione e configurazione delle risorse
3. Il sistema può garantire l'obiettivo rimanga soddisfatto nel tempo, attraverso il monitoraggio della rete e interventi correttivi automatici

SDN: sfide selezionate

- Hardening del piano di controllo: sistema distribuito *dependable**, scalabile nelle prestazioni e sicuro
 - robustezza ai guasti: sfruttare la teoria forte dei sistemi distribuiti affidabili per il piano di controllo
 - *dependability*, sicurezza: “incorporati” fin dal primo giorno?
- reti, protocolli che soddisfano i requisiti specifici di missione
 - es., tempo reale, ultra-affidabilità, ultra-sicurezza
- Estensione oltre un singolo AS
- L'SDN è fondamentale per le reti cellulari 5G

**dependability* (predicibilità) è definita in termini di *availability* (disponibilità: percentuale di tempo in cui un sistema è operativo e accessibile quando necessario), *reliability* (affidabilità: la capacità del sistema di svolgere costantemente le sue funzioni senza interruzioni su un determinato periodo di tempo), *safety* (sicurezza da incidenti) e *security* (sicurezza da intrusioni o accessi indesiderati)

SDN e il futuro dei protocolli di rete tradizionali

- Tabelle di inoltro calcolate da SDN rispetto a quelle calcolate da router:
 - solo un esempio di calcolo logicamente centralizzato rispetto al calcolo protocollare
- si potrebbe immaginare un controllo della congestione calcolato da SDN:
 - il controllore imposta le velocità dei mittenti in base ai livelli di congestione segnalati dal router (al controllore)



Come evolverà l'implementazione delle funzionalità di rete (SDN rispetto ai protocolli)?

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

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

Livello di rete: piano di controllo (parte3)

dr. Manuel Fiorelli

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

Introduction: 1-120

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

ICMP: internet control message protocol

- utilizzato da host e router per comunicare informazioni a livello di rete
 - segnalazione di errori: host, rete, porta, protocollo non raggiungibile
 - richiesta/risposta echo (usato da ping)
- livello di rete “sopra” l'IP:
 - messaggi ICMP trasportati nei datagrammi IP
 - non viene considerato un protocollo di trasporto perché non usato dalle applicazioni di rete per trasferire i propri messaggi

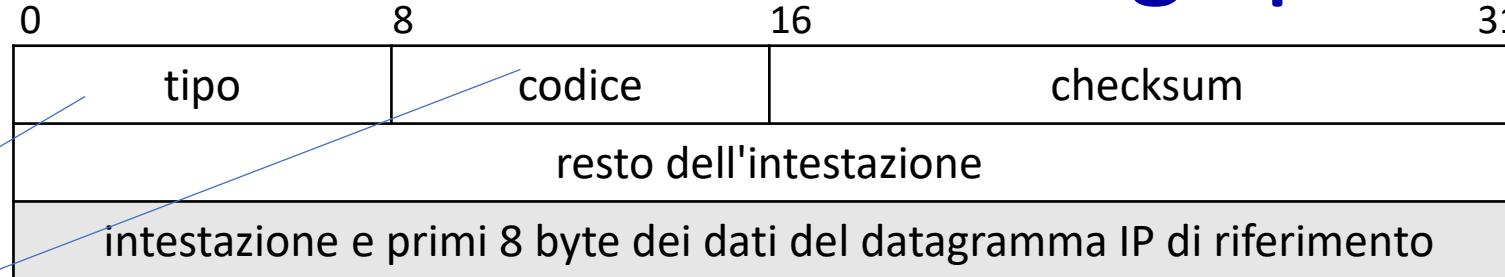
Messaggio ICMP:

la struttura
dipende dal tipo
di messaggio

0	8	16	31
tipo	codice	checksum	
resto dell'intestazione			
dati / intestazione e primi 8 byte dei dati del datagramma IP di riferimento			

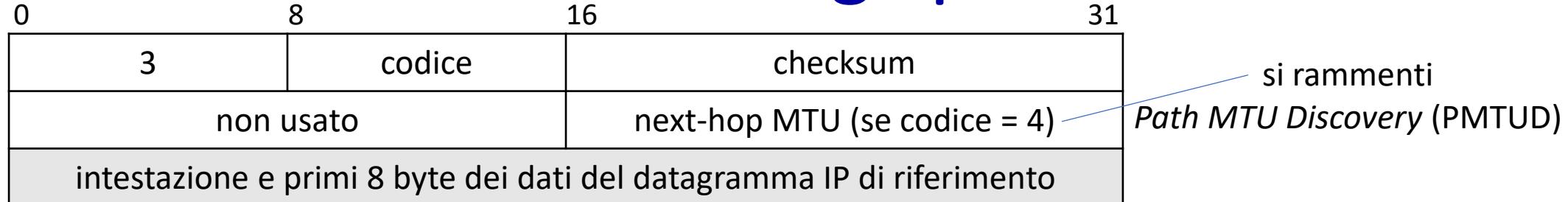
in alcuni casi, usato per determinare il datagramma che ha causato il messaggio e determinare il processo corrispondente (assumendo che i numeri di porta si trovino nei primi 8 byte)

ICMP: internet control message protocol



<u>Tipo</u>	<u>Codice</u>	<u>Descrizione</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	fragmentation required
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

ICMP: internet control message protocol



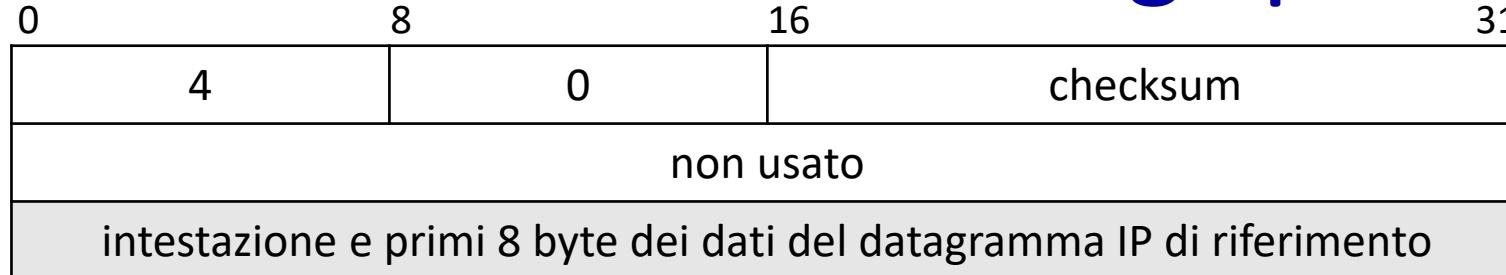
<u>Tipo</u>	<u>Codice</u>	<u>Descrizione</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	fragmentation required
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

possibilmente inviati dai router lungo il percorso. Una rete è irraggiungibile ad esempio se il router sa che la rete di destinazione esiste (cioè ha una voce di routing), ma la ritiene **irraggiungibile** in quel momento: ad esempio perché l'interfaccia di uscita è down oppure il next-hop non risponde. Per un host può derivare dal fallimento della risoluzione ARP. L'essere sconosciuto deriva al non trovare una rotta idonea.

dovrebbero essere inviati dall'host di destinazione, quando il protocollo o la porta non sono attivi. Il protocollo TCP gestisce il secondo caso attraverso l'invio di segmenti RST. Tuttavia, il lato mittente (di qualunque protocollo di trasporto) deve gestire anche l'analogo messaggio ICMP.

I messaggi *destination unreachable* devono essere passati dal livello di rete a quello di trasporto, che dovrebbe farne un uso appropriato (es. riportare il problema all'applicazione)

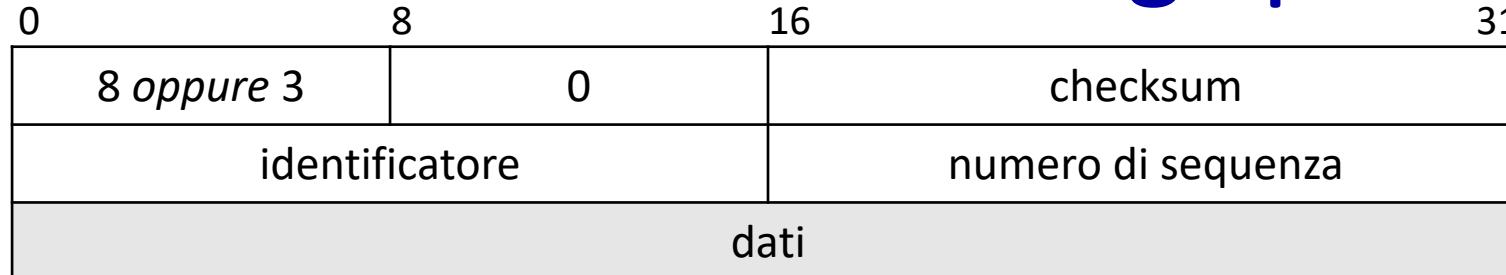
ICMP: internet control message protocol



Tipo	Codice	Descrizione
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	fragmentation required
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Può essere inviato da un router congestionato per forzare l'host mittente a ridurre il tasso di trasmissione. Permette una forma di controllo della congestione informata dalla rete. **Oggi deprecato.** Si ricordi, invece, il meccanismo ECN discusso in precedenza.

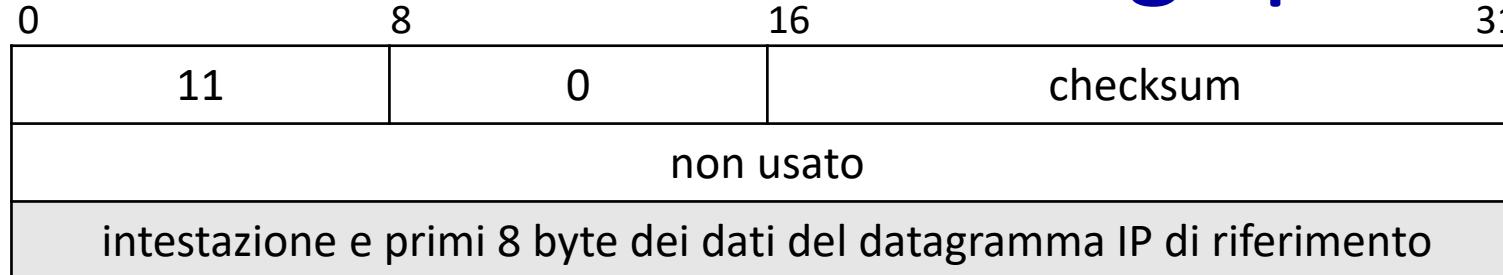
ICMP: internet control message protocol



Tipo	Codice	Descrizione
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	fragmentation required
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

La risposta *ping* contiene gli stessi dati della richiesta *ping*: tra le altre cose può includere un timestamp per calcolare RTT in maniera stateless

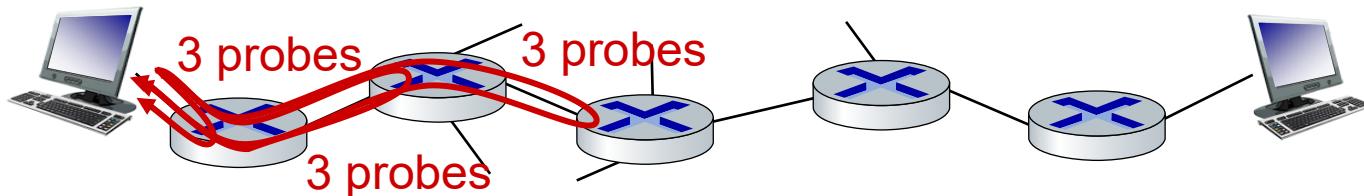
ICMP: internet control message protocol



Tipo	Codice	Descrizione
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	fragmentation required
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

viene usato da *traceroute*

Traceroute e ICMP



- la sorgente invia serie di segmenti UDP alla destinazione (con un numero di porta "improbabile")
 - 1° insieme ha TTL =1, 2° insieme ha TTL=2, ...
 - un datagramma nella n -esimo insieme arriva all' n -esimo router:
 - il router scarta il datagramma e invia il messaggio ICMP alla sorgente (tipo 11, codice 0)
 - l'indirizzo IP del router si trova nel campo indirizzo sorgente dell'intestazione del datagramma che incapsula il messaggio ICMP
 - quando il messaggio ICMP arriva alla sorgente: registrare gli RTT

criteri di arresto:

 - Il segmento UDP arriva eventualmente a destinazione
 - la destinazione restituisce il messaggio ICMP "port unreachable" (tipo 3, codice 3)
 - la sorgente si ferma

ICMP: internet control message protocol

- ICMPv6 per IPv6:
 - nuovi tipi e codici
 - ridefinizione di alcuni tipi e codici esistenti

Per esempio, invece di "*destination unreachable – fragmentation required*" c'è il messaggio "*packet too big*".

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

Cos'è la gestione della rete (*network management*)?

- sistema autonomo (noto altrimenti come "rete"): migliaia di componenti software e hardware che interagiscono tra loro



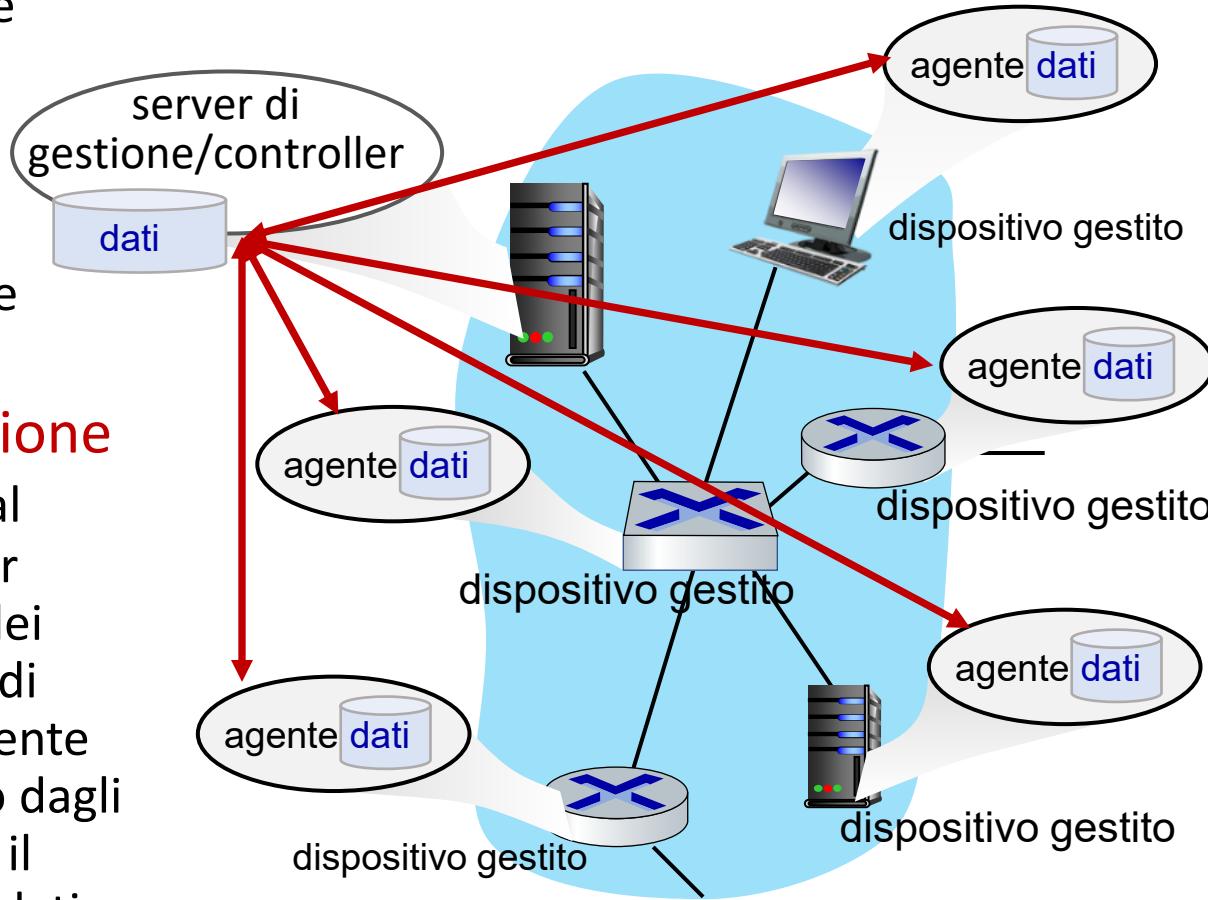
Saydam 1996

"La **gestione della rete** comprende il funzionamento, l'integrazione e il coordinamento di hardware, software e personale tecnico per monitorare, verificare, configurare, analizzare, valutare e controllare le risorse della rete affinché soddisfino le funzionalità in tempo reale e i requisiti di qualità del servizio a un costo accettabile"

Componenti della gestione della rete

Server di gestione:
raccolta, elaborazione
e analisi delle
informazioni, invio di
informazioni e
comandi, in genere
con i gestori della rete
(umani) nel loop

**Protocollo di gestione
di rete:** utilizzato dal
server di gestione per
interrogare lo stato dei
dispositivi e agire su di
essi attraverso un agente
di gestione; utilizzato dagli
agenti per informare il
server di gestione di dati
ed eventi.



Dispositivo di rete gestito:
apparecchiature con
componenti hardware e
software gestibili e configurabili

Dati: “stati” del
dispositivo: dati di
configurazione (assegnati
dall'amministratore, come
indirizzo IP), dati operativi
(acquisiti da dispositivo, come i
vicini OSPF), statistiche

Agente di gestione:
comunica con il server di
gestione, agisce su un
dispositivo gestito

Approcci dell'operatore di rete per gestire la rete

CLI (Command Line Interface)

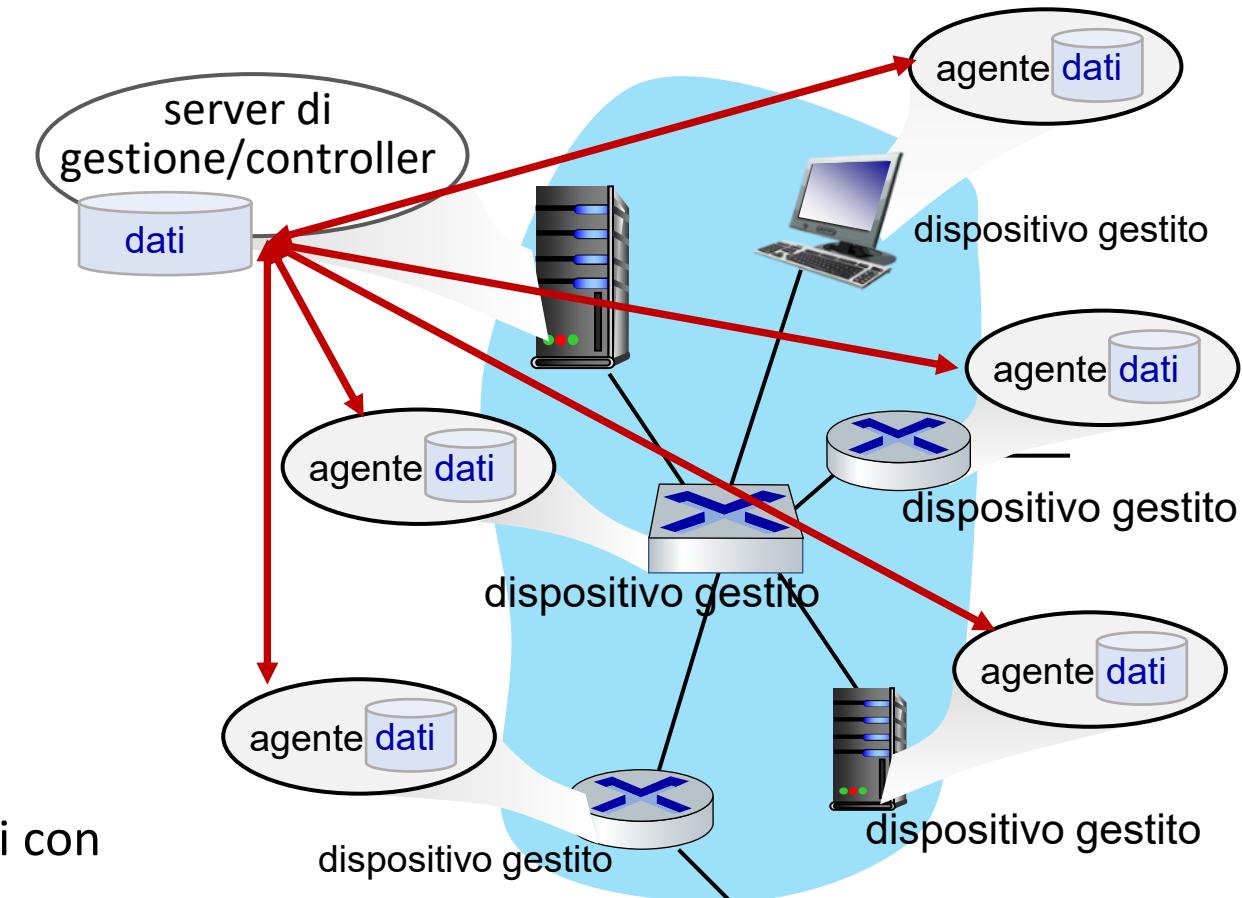
- l'operatore scrive comandi su una console del dispositivo o esegue script da remoto, per esempio, attraverso un ssh
- molti dispositivi hanno anche una UI web

SNMP/MIB

- l'operatore interroga/imposta i dati contenuti negli oggetti MIB (*management information base*) utilizzando il Simple Network Management Protocol (SNMP)

NETCONF/YANG

- più astratto, a livello di rete, olistico
- enfasi sulla gestione della configurazione multidispositivo
- YANG: linguaggio di modellazione dei dati
- NETCONF: comunicare azioni/dati compatibili con YANG a/da/tra dispositivi remoti



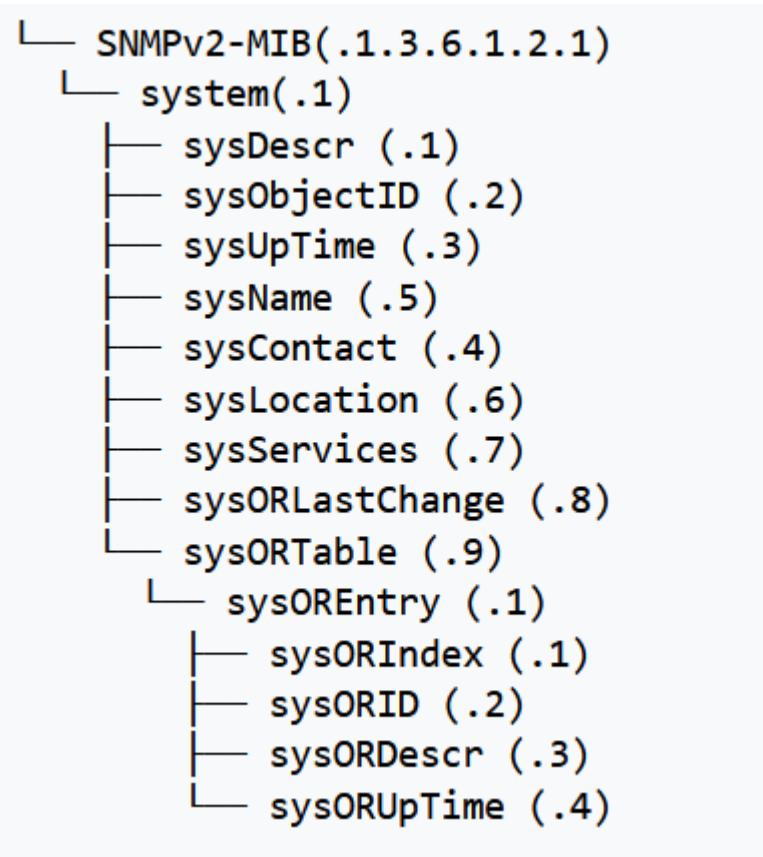
SNMP: Management Information Base (MIB)

- i dati operativi (e alcuni dati di configurazione) del dispositivo gestito sono **modellati** come *managed objects*
- raccolti **moduli MIB**
 - 400 moduli MIB definiti da RFC; molte più moduli MIB specifici del fornitore
- **Structure of Management Information (SMI):** linguaggio di definizione dei dati
- esempio di MIB per il protocollo UDP:



Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPIInDatagrams	32-bit counter	numero totale di datagrammi consegnati
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	numero di datagrammi non consegnabili (nessuna applicazione alla porta)
1.3.6.1.2.1.7.3	UDInErrors	32-bit counter	numero di datagrammi non consegnabili (qualsiasi altra ragione)
1.3.6.1.2.1.7.4	UDPOutDatagrams	32-bit counter	numero totale di datagrammi inviati
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	una voce per ogni porta in uso

SNMP: Management Information Base (MIB)

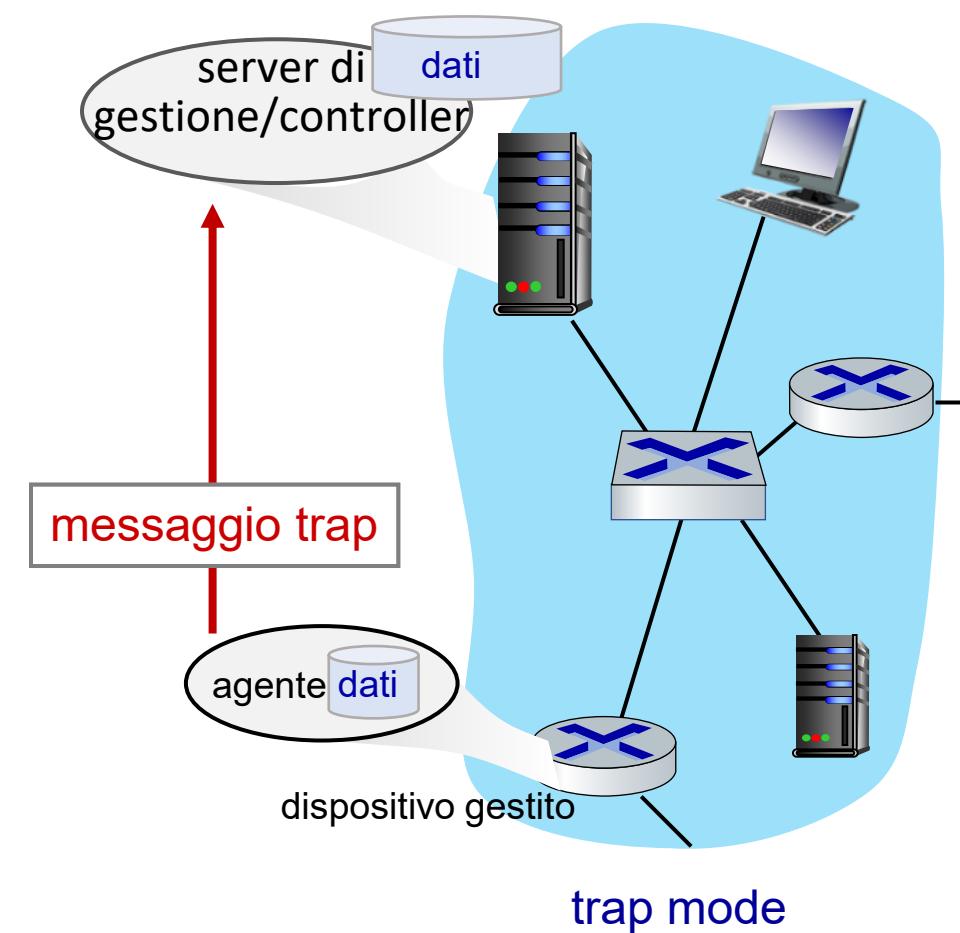
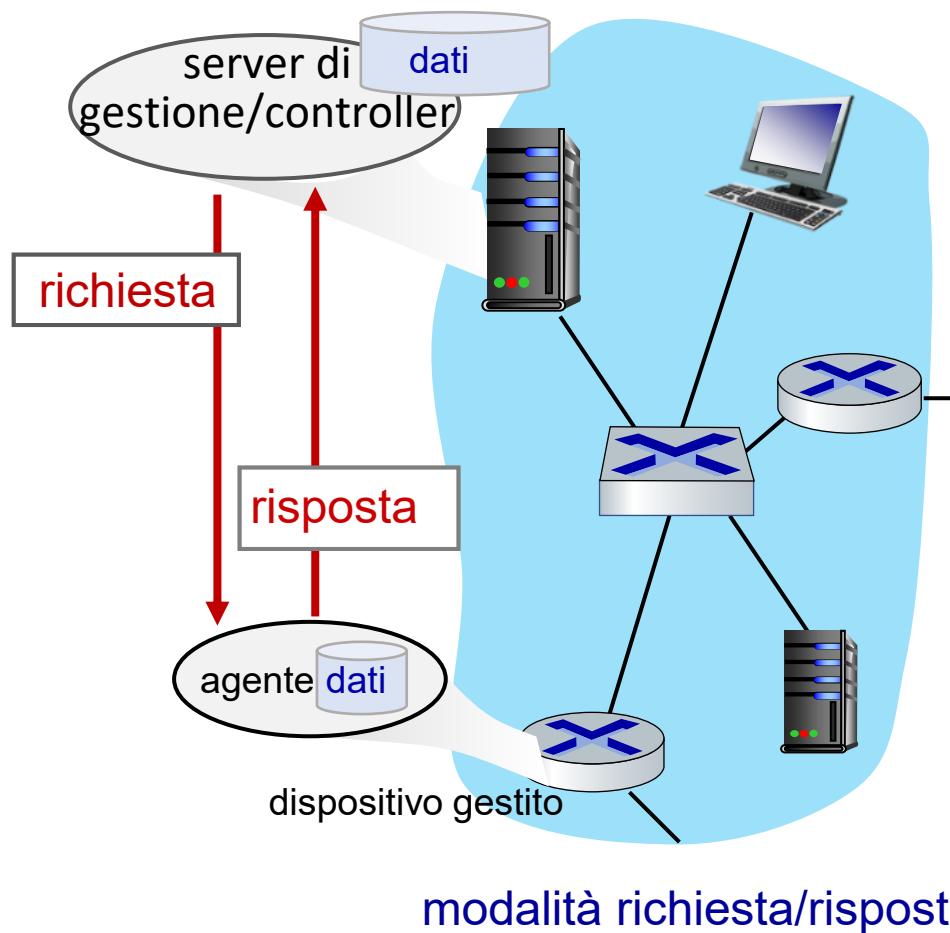


- ogni *managed object* ha un OID (*object identifier*) univoco
- gerarchico
- istanza: uno specifico valore o misurazione di un oggetto MIB in un dato momento
- due tipi di managed object:
 - scalari: singola istanza, identificato come OID.0
 - tabulari: molteplici valori, ciascuno identificato appendendo un indice di riga (a partire da 1) all'OID

Protocollo SNMP

Due modi per trasmettere le informazioni MIB e comandi:

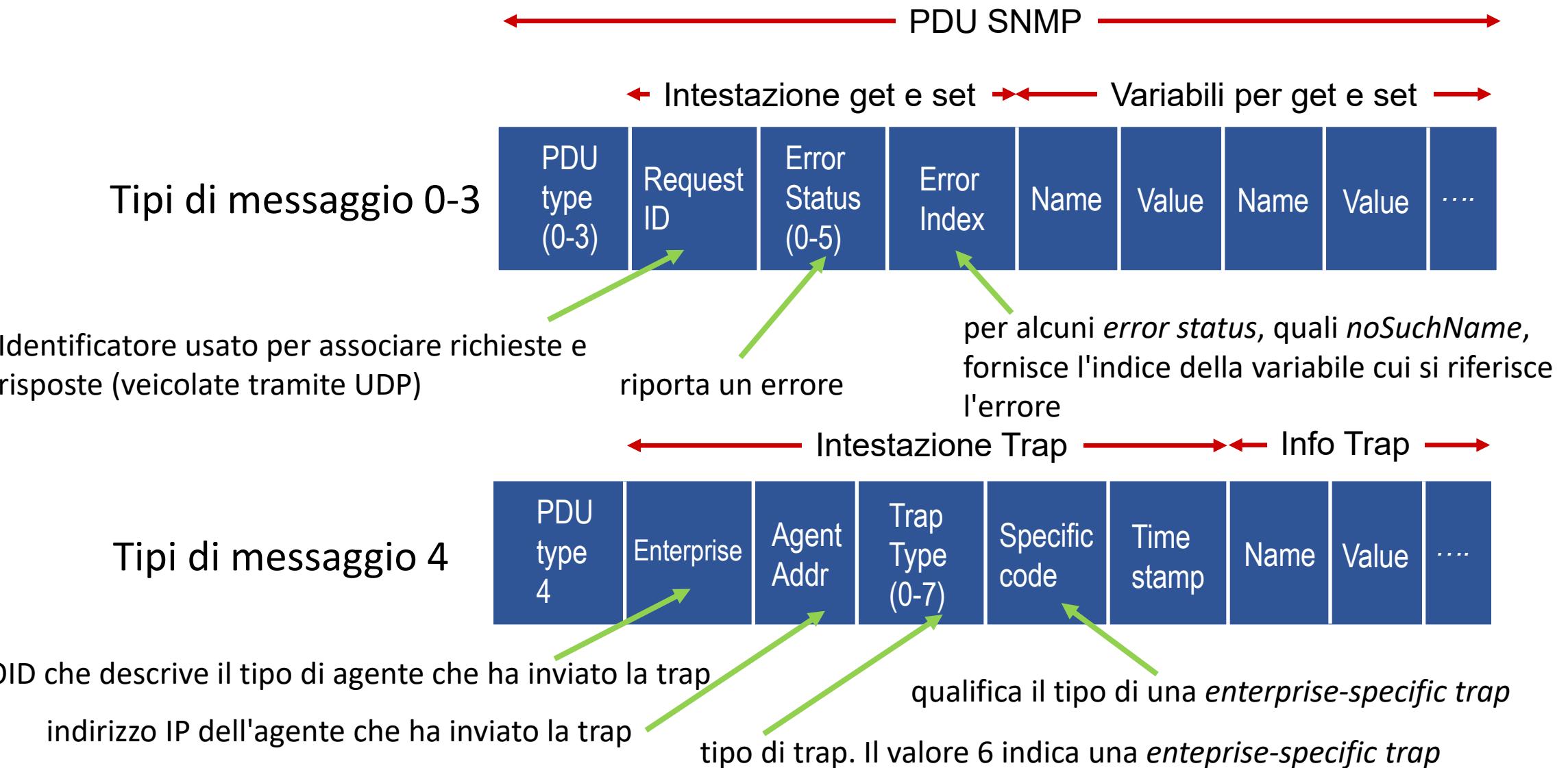
SNMP utilizza in genere il protocollo di traporto UDP



Protocollo SNMP: tipi di messaggio

Tipo di messaggio	Funzione
GetRequest GetNextRequest GetBulkRequest	manager→agente: “dammi dati” (istanze di oggetto, prossima istanza di oggetto in lista o tabella, blocco di dati).
SetRequest	manager→agente: imposta il valore di una o più istanze di oggetti MIB
Response	agente→manager: generato in risposta a una richiesta
Trap	agente→manager: informa il manager di un evento inatteso

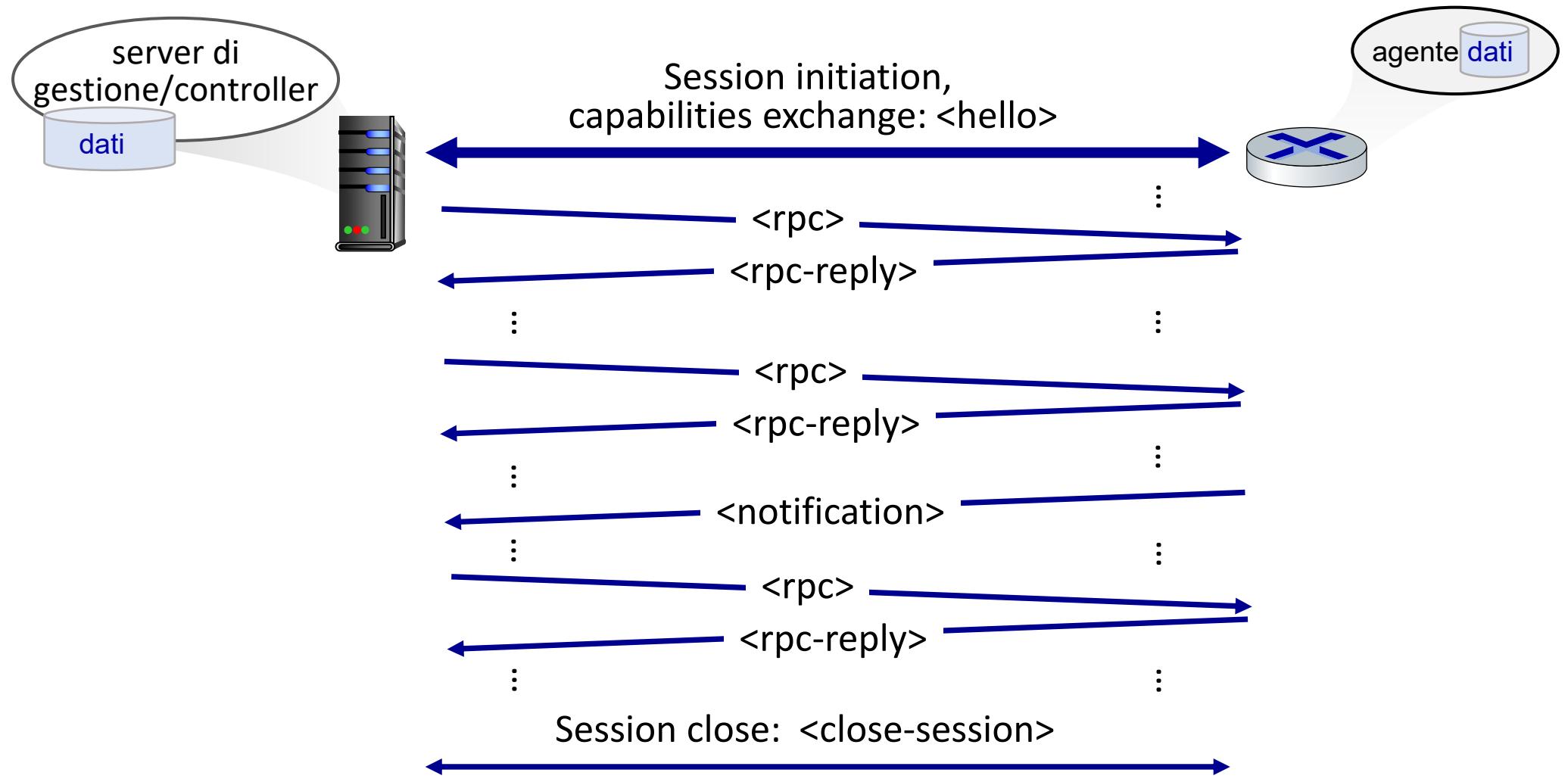
Protocollo SNMP: formati dei messaggi



Panoramica di NETCONF

- **obiettivo:** gestire/**configurare** in maniera attiva dispositivi a sulla rete
- opera tra il server di gestione e i dispositivi di rete gestiti
 - azioni: retrieve, set, modify, activate configurations
 - **commit atomico** di azioni su molteplici dispositivi
 - interrogare i dati operativi e le statistiche
 - sottoscrivere le notifiche dai dispositivi
- Paradigma a chiamata di procedura remota (*remote procedure call*, RPC)
 - messaggi del protocollo NETCONF codificati in XML
 - scambiati attraverso un protocollo di trasporto affidabile e sicuro (e.g., TLS, SSH)

NETCONF: inizializzazione, scambio, chiusura



Operazioni NETCONF selezionate

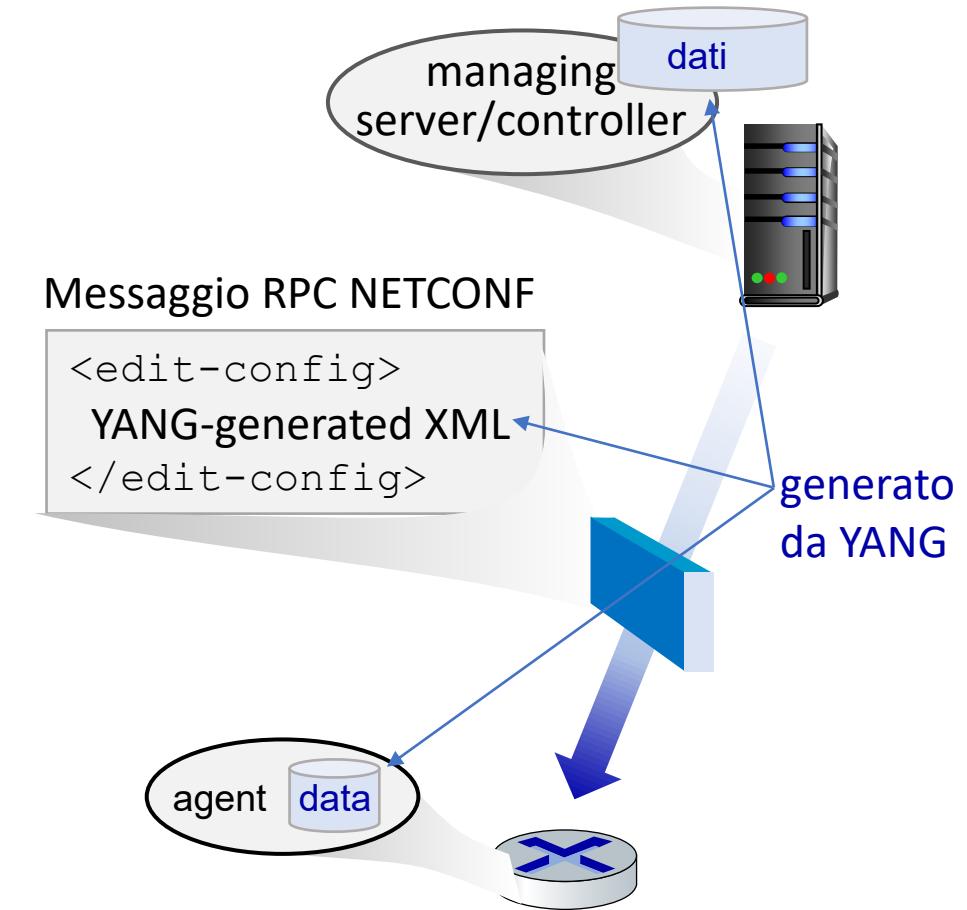
Operazione	Descrizione
<get-config>	Recupera tutta o parte di una data configurazione. Un dispositivo può avere più configurazioni. C'è sempre una configurazione <i>running</i> , che descrive la configurazione corrente (in esecuzione) dei dispositivi.
<get>	Recupera tutti o parte dei dati operativi e della configurazione running.
<edit-config>	Modifica la configurazione (possibilmente in esecuzione) del dispositivo gestito. Quest'ultimo invia un <rpc-reply> contenente <ok>; altrimenti viene inviato un <rpccerror> con rollback.
<lock>, <unlock>	Bloccare (sbloccare) il datastore di configurazione sul dispositivo gestito (per bloccare i comandi NETCONF, SNMP o CLI da altre fonti).
<create-subscription>, <notification>	Abilita la sottoscrizione di notifiche di eventi dal dispositivo gestito

Esempio di messaggio NETCONF RPC

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101" nota l'id del messaggio
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04     <edit-config>    cambia una configurazione
05       <target>
06         <running/>  cambia la configurazione in esecuzione
07       </target>
08     <config>
09       <top xmlns="http://example.com/schema/
10         1.2/config">
11           <interface>
12             <name>Ethernet0/0</name>
13             <mtu>1500</mtu>          cambia la MTU dell'interfaccia Ethernet 0/0 in 1500
14           </interface>
15         </top>
16       </config>
17     </edit-config>
18   </rpc>
```

YANG

- linguaggio di modellazione dei dati utilizzato per specificare la struttura, la sintassi e la semantica dei dati di gestione della rete NETCONF
 - tipi di dati incorporati, come SMI
- documento XML che descrive il dispositivo; può essere generato dalla descrizione YANG
- può esprimere vincoli tra i dati che devono essere soddisfatti da una configurazione NETCONF valida
 - garantire che le configurazioni NETCONF soddisfino i vincoli di correttezza e di coerenza



YANG (continua)

Esempio YANG:

```
container system {  
    container login {  
        leaf message {  
            type string;  
            description "Message given at start of login session";  
        }  
    }  
}
```

Esempio NETCONF:

```
<system>  
    <login>  
        <message>Good morning</message>  
    </login>  
</system>
```

Livello di rete: Riassunto

Abbiamo imparato molto!

- approcci al piano di controllo della rete
 - controllo per router (tradizionale)
 - controllo centralizzato logicamente (software defined networking)
- algoritmi di instradamento tradizionali
 - implementazione in Internet: OSPF , BGP
- controller SDN
 - implementazione in pratica: ODL, ONOS
- Internet Control Message Protocol
- network management

Prossima fermata: livello di collegamento!

Livello di rete: “piano di controllo” Fatto!

- 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