Lo strato di Rete
Architettura di un Router

Reti e Laboratorio III

2023/24

Inoltro vs routing

- *Forwarding (inoltro):*

trasferire i pacchetti sull'appropriato collegamento in uscita

  - Usa la tabella di inoltro

Data Plane

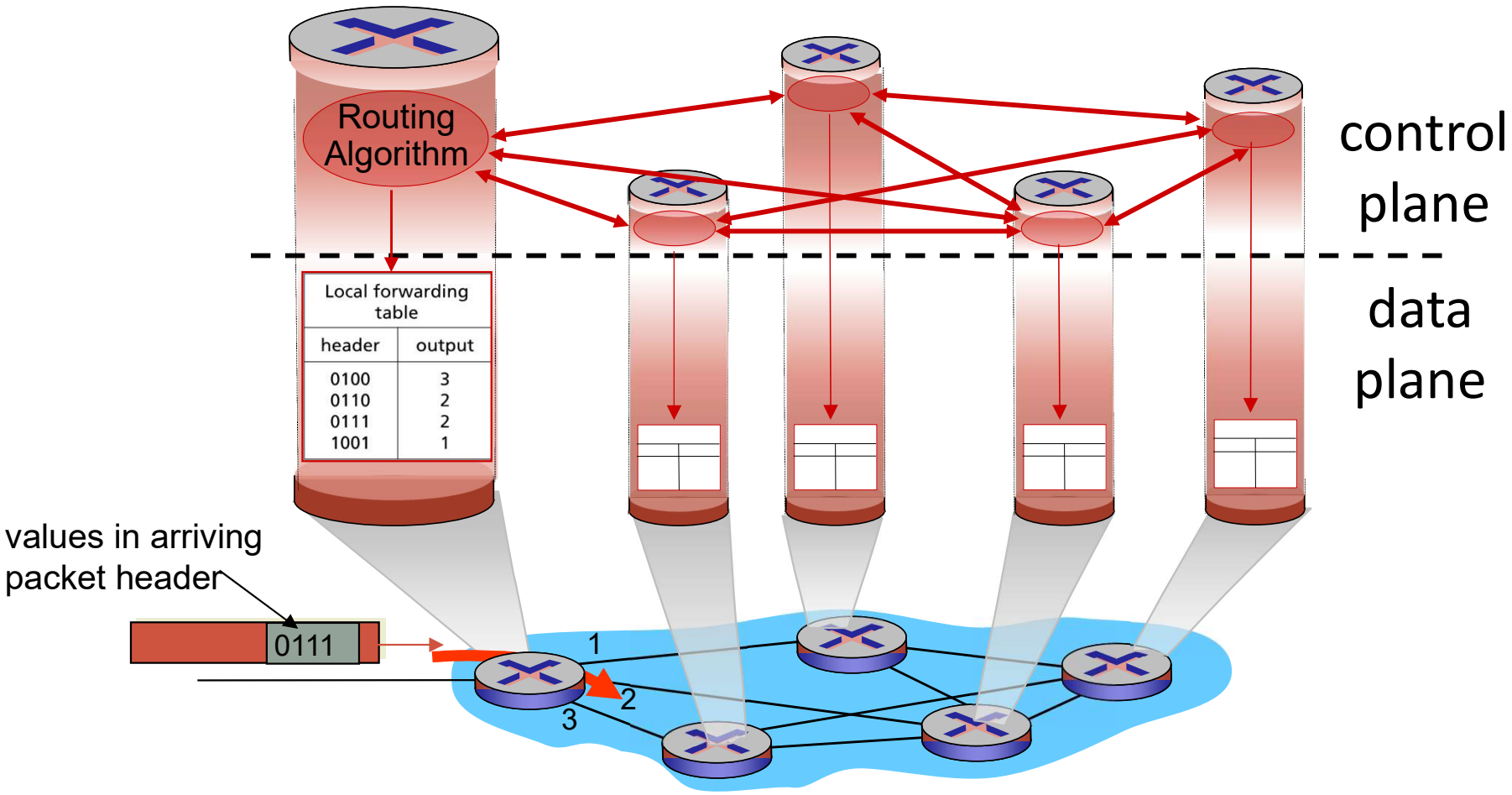
- *Routing (instradamento):*

processo decisionale di scelta del percorso verso una destinazione

  - determina i valori da inserire nella tabella di inoltro
  - tramite algoritmi di *routing*

Control Plane

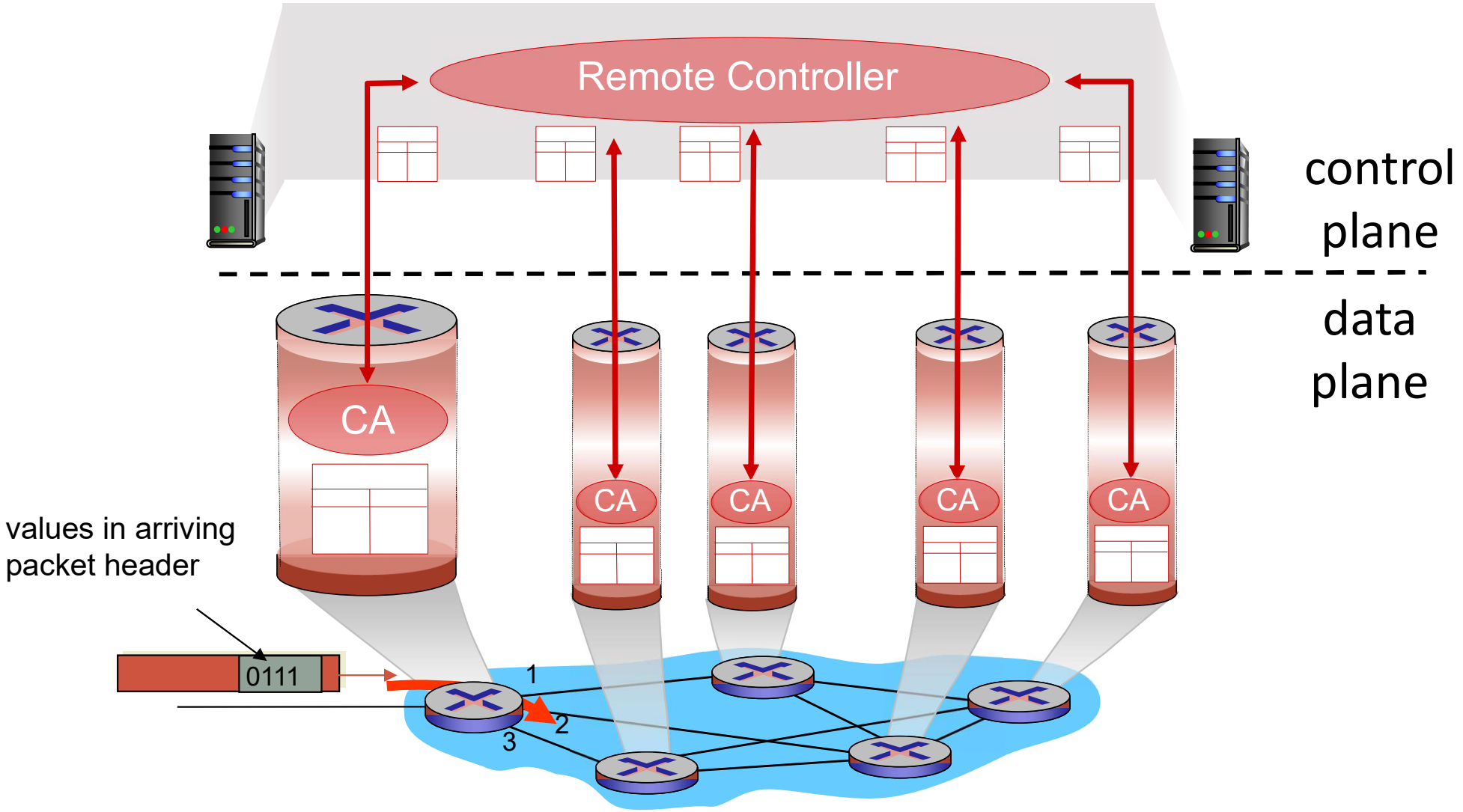
Piano di controllo vs Piano dati



Routing decentralizzato:

- Algoritmo di routing in esecuzione su ciascun router
- I router si scambiano messaggi (protocolli di routing)

Routing logicamente centralizzato



Un controller remoto interagisce con Control Agents (CA) locali:

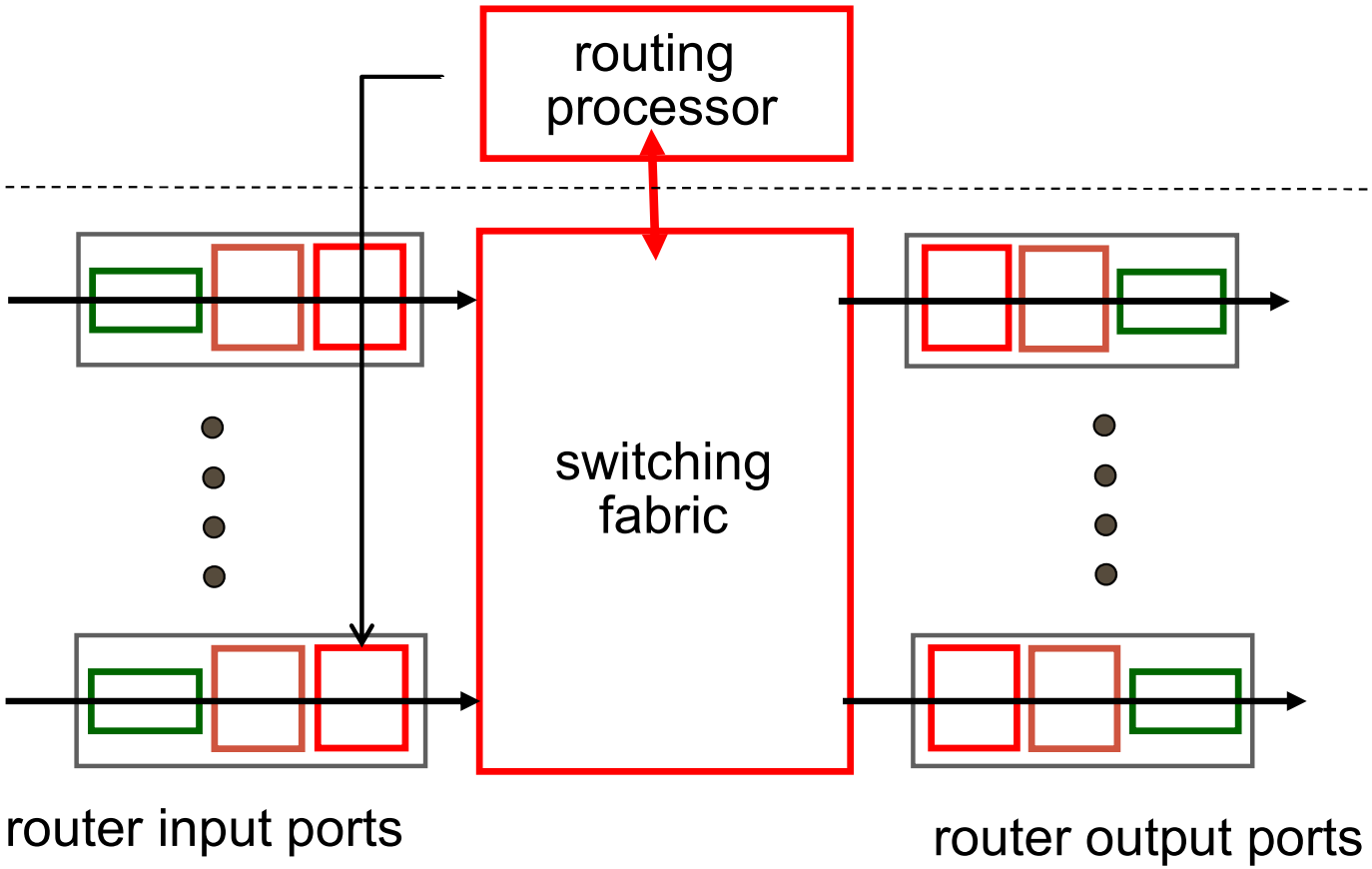
- Riceve dai CA informazioni sui collegamenti e sul traffico
- Invia ai CA i valori da inserire nella tabella di inoltro

Software-defined Networking (SDN)

Architettura di un router

- Nella nostra discussione sull'inoltro e l'instradamento abbiamo rappresentato il router come una scatola nera che accetta pacchetti in entrata da una delle porte di input (interfacce), usa una tabella d'inoltro per trovare la porta di output e da questa invia il pacchetto.
- Componenti
  - Porte di input
  - Porte di output
  - Processore di routing
  - Switching fabric (struttura di commutazione)

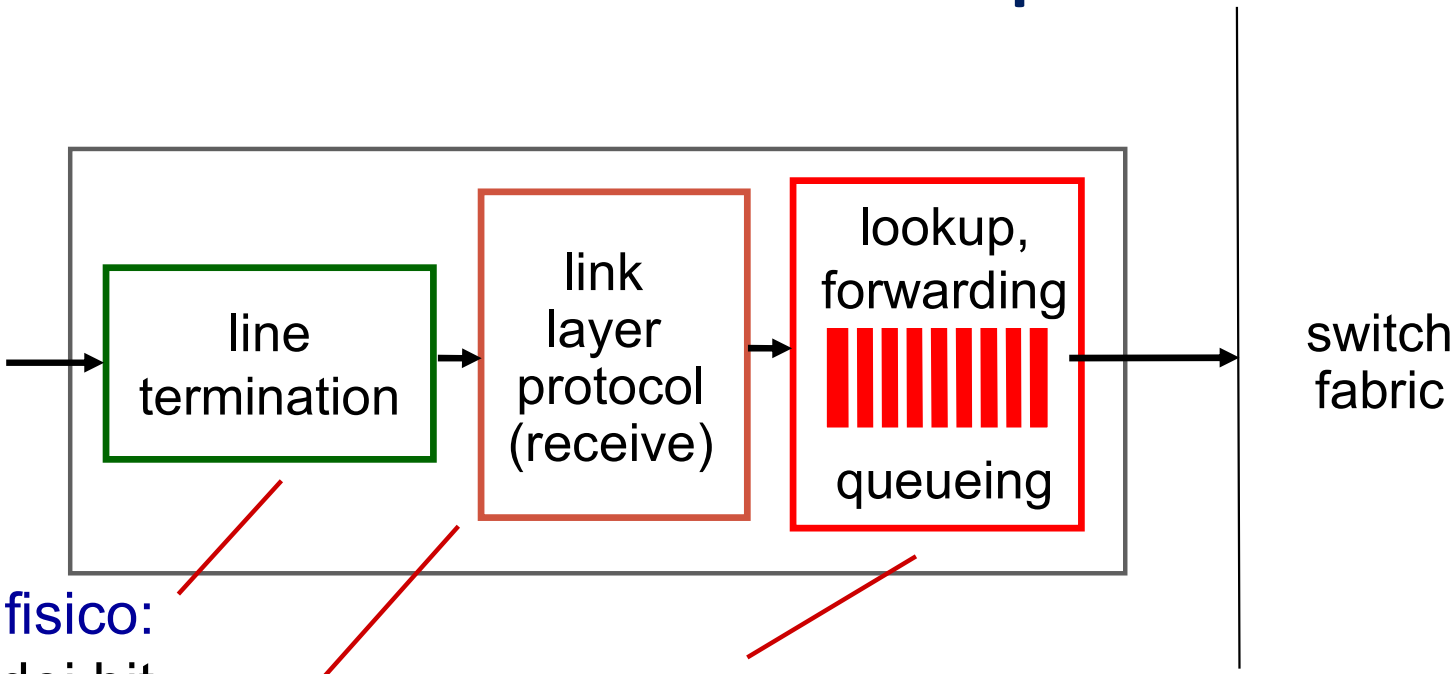
Architettura di un router



routing, management control plane (software)
opera nell'ordine dei millisecondi

forwarding data plane (hardware)
opera nell'ordine dei nanosecondi

Architettura di un router – porte di input



Livello fisico:
Ricezione dei bit

Livello collegamento
es. Ethernet

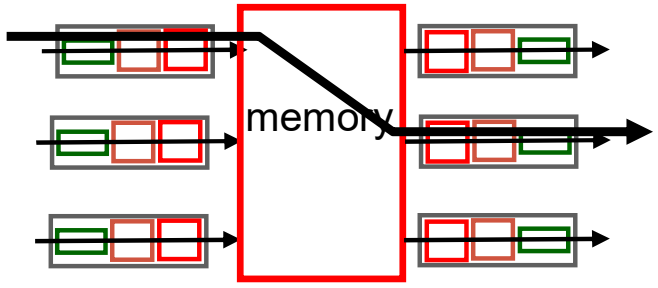
Ricerca, inoltro, accodamento:

- Ha una copia della tabella di inoltro, usa i valori dell'header per determinare la porta di output
- Elaborazione alla velocità "line rate"
- Accodamento: se la velocità con cui arrivano I datagrammi supera la velocità di inoltro nella struttura di commutazione (switching fabric)

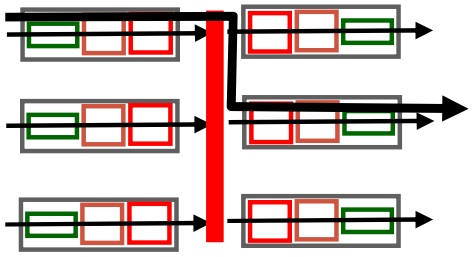
Il numero di porte supportate da un router può variare da un numero relativamente piccolo nei router aziendali, a centinaia di porte a 10 Gbps in un router di bordo di un ISP Il router Juniper MX2020, per esempio, supporta fino a 800 porte Ethernet a 100 Gbps

Switching fabric

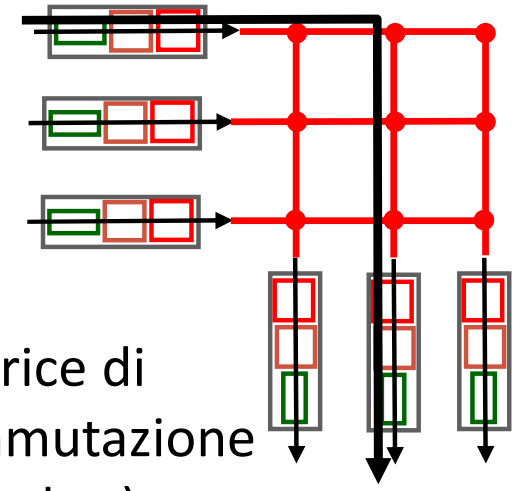
- trasferisce il pacchetto dal buffer di input al buffer di output appropriato
- velocità di commutazione: velocità con cui i pacchetti possono essere trasferiti dagli ingressi alle uscite
  - N input: velocità di commutazione auspicabile N volte la velocità di linea



memory

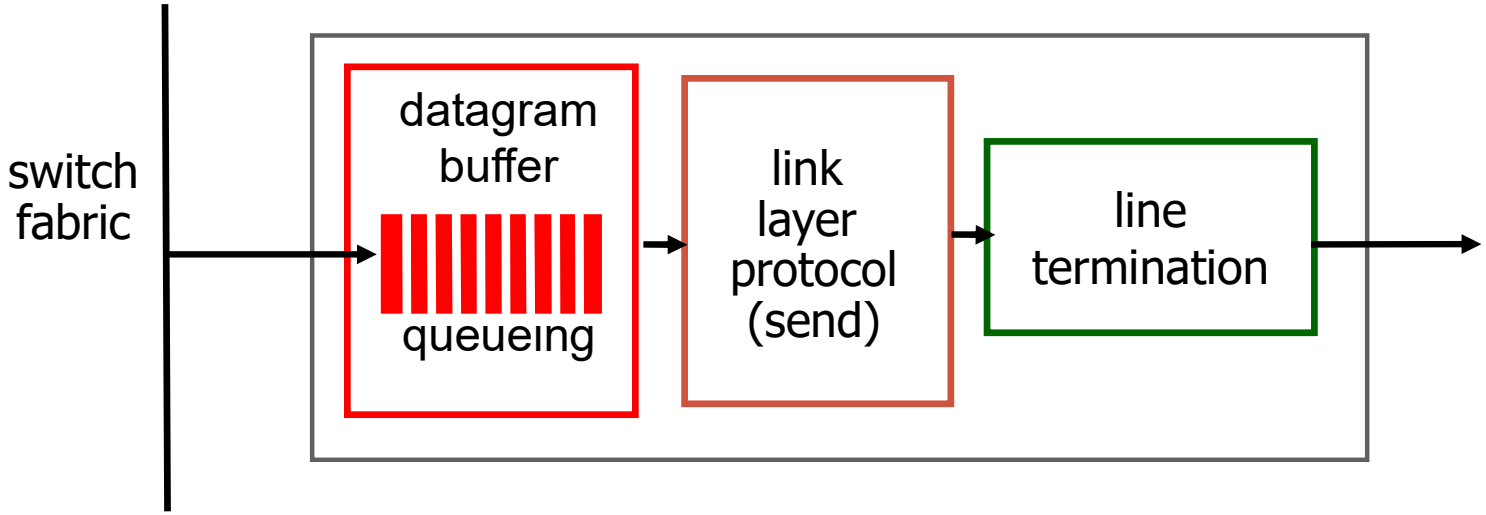


bus



Matrice di commutazione (crossbar)

Architettura di un router – porte di output



- *buffering* richiesto quando i datagrammi arrivano dalla struttura di commutazione ad una velocità maggiore della velocità di trasmissione sul collegamento in uscita
- *Scheduling:* politiche per definire l'ordine di trasmissione dei datagrammi

Ritardi e perdite

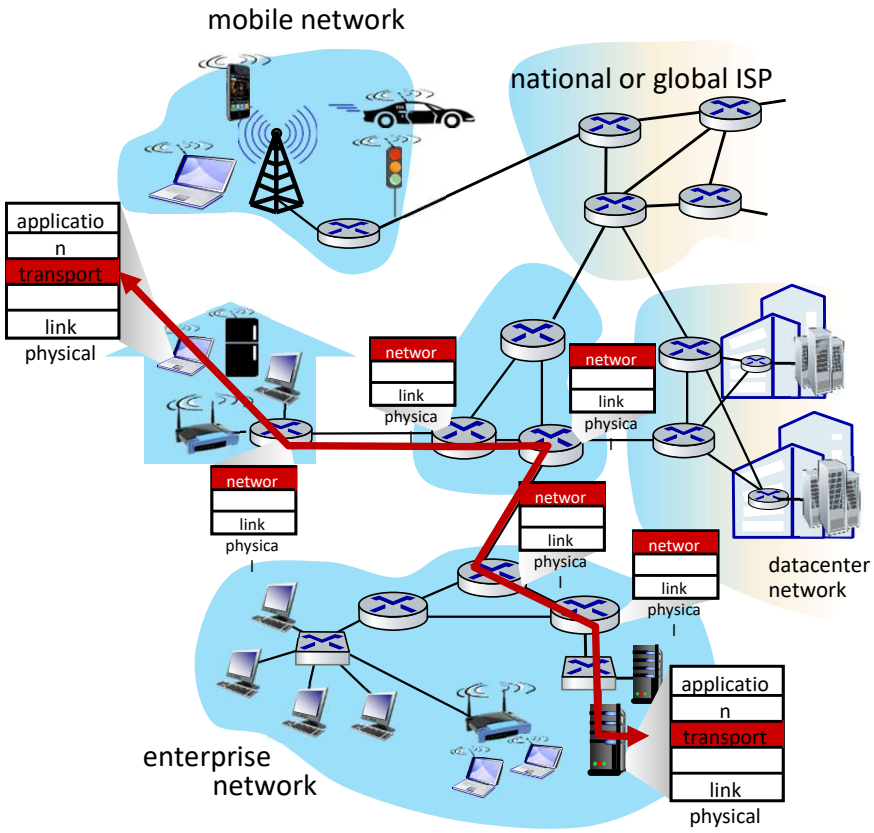
- Si possono formare code di pacchetti sia presso le porte di ingresso sia presso quelle di uscita
- La lunghezza della coda dipende da vari fattori, tra cui: la quantità di traffico di rete, le velocità relative della struttura di commutazione e della linea

Lo strato di Rete
Routing

Protocolli di Routing

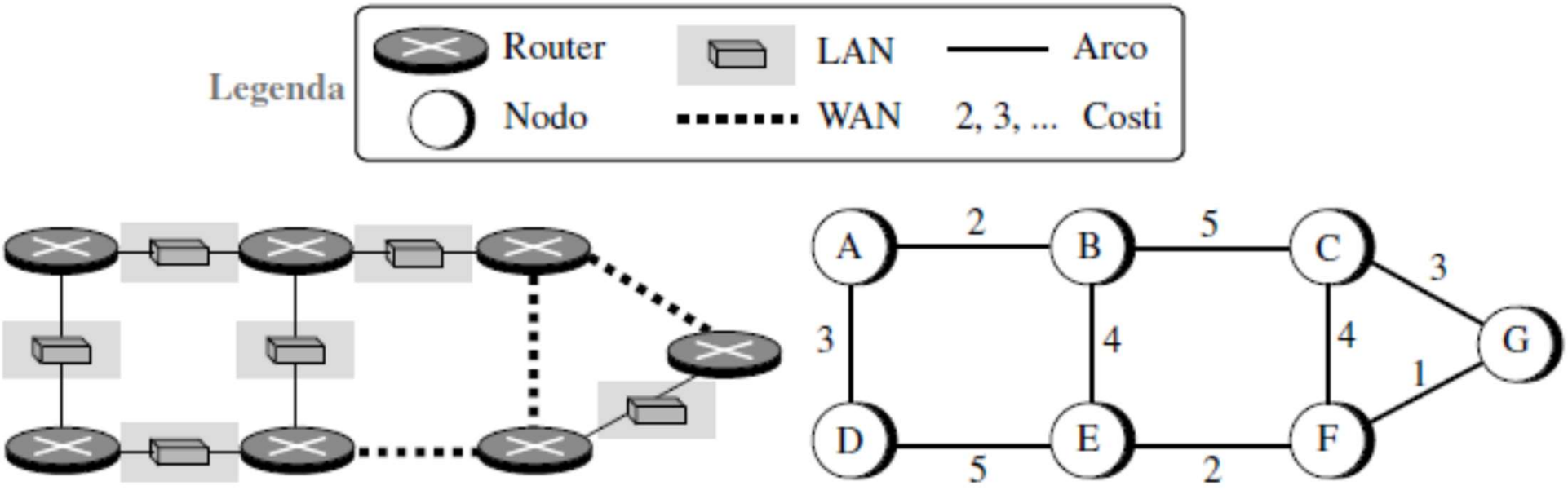
Obiettivo: determinare percorsi “buoni” dagli host mittenti agli host destinatari, attraverso nodi intermedi (router)

- percorso: sequenza di router che i pacchetti attraversano dall’host sorgente alla destinazione finale
- “buono”: costo minore, più veloce, meno congestionato
- routing: una delle principali sfide in ambito networking!



Routing

Rappresentiamo una rete di reti come un grafo

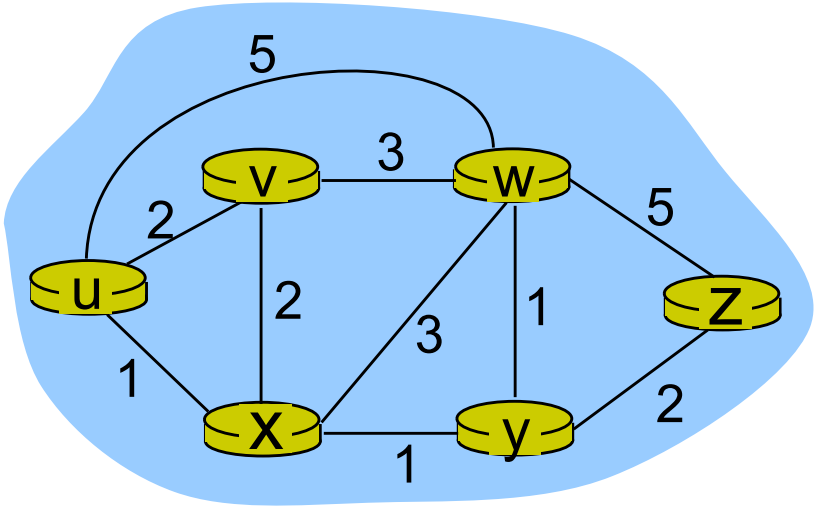


a. Una internet

b. Il grafo pesato

- Nodi: router
- Arco: collegamenti tra router
- Grafo pesato
  - Costo associato all'arco: distanza tra i nodi, numero di hop, distanza, bandwidth disponibile ecc.

Graph abstraction: costi



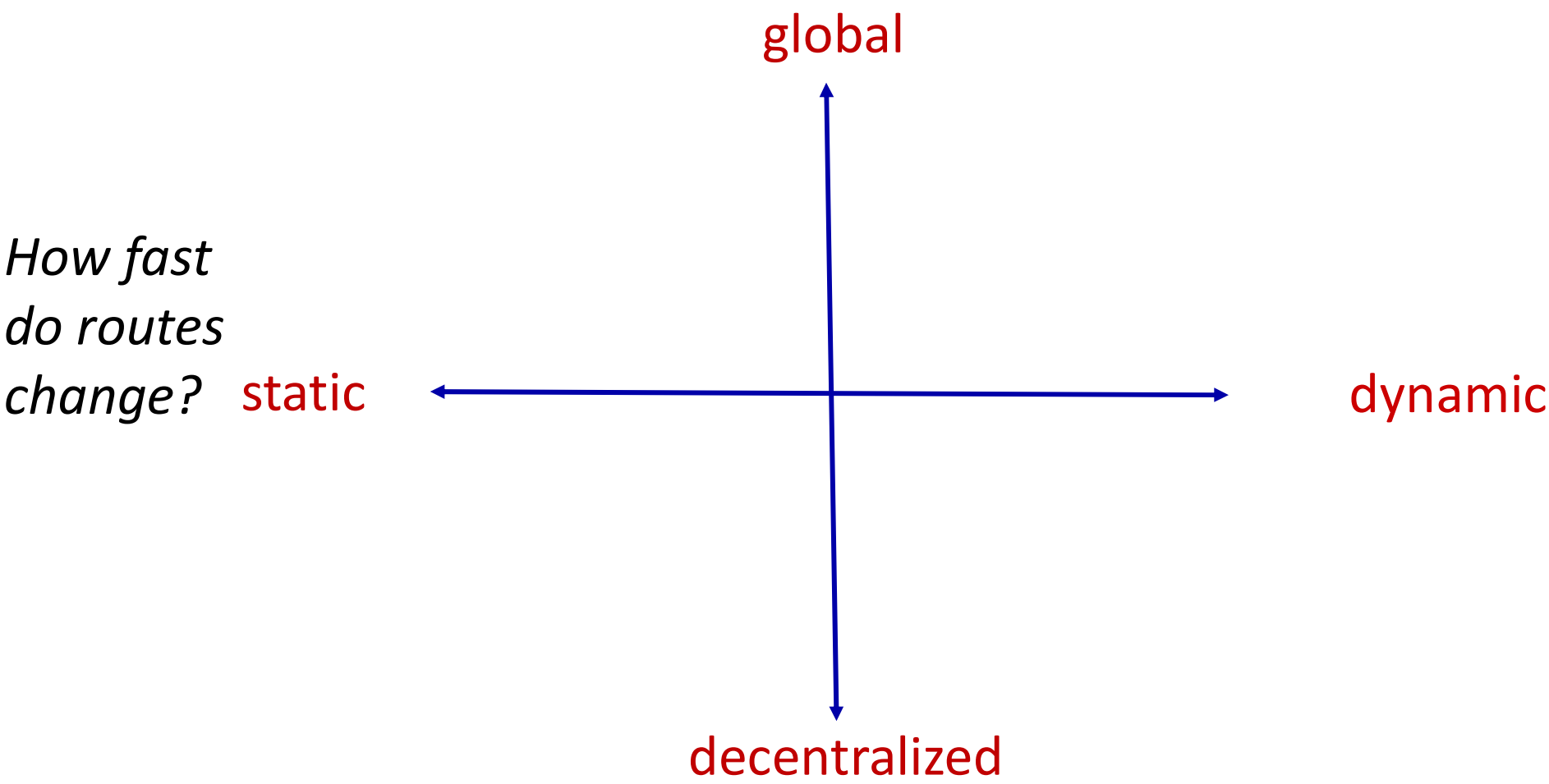
c(x,x') = costo del link (x,x')
e.g., c(w,z) = 5

Costo definite dall'operatore di rete, es.: può essere 1, inversamente proporzionale alla bandwidth o alla congestione

Costo percorso $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: qual è il cammino a costo minimo tra u e z ?
routing algorithm: algoritmo che calcola il cammino a costo minimo

Classificazione algoritmi di Routing



Routing statico e dinamico

- Nel **routing statico** le righe (entry) della tabella vengono configurate manualmente dall'operatore. Tale metodo viene usato per reti di piccole dimensioni e la cui topologia non varia molto, dove è possibile prevedere tutti i possibili percorsi di un pacchetto IP nella rete.

- Nel **routing dinamico** esistono protocolli specifici che provvedono automaticamente ad inserire nella tabella del router le entry relative ai possibili percorsi. Viene usato nelle reti di medie-grandi dimensioni e con topologia variabile (grandi reti locali private e Internet)

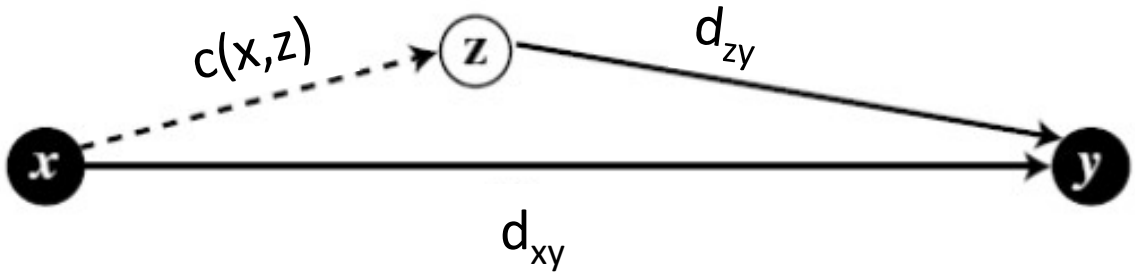
Algoritmi di instradamento globali/decentralizzati

- Gli algoritmi di instradamento si possono classificare in:
- Globali, se si basano sulla conoscenza della topologia di tutta la rete. Il calcolo può essere fatto in un unico sito (es. SDN) o in più nodi, utilizzando informazioni sulla connettività di tutti i nodi e sui costi di tutti i link (es. algoritmo Link State).
  - L'algoritmo riceve in ingresso informazioni su tutti i collegamenti tra i nodi e i loro costi.
- Decentralizzati, quando nessun nodo conosce la topologia di tutta la rete, all'inizio ha informazioni solo sui nodi e link vicini. Il calcolo del percorso è iterativo e distribuito (es. algoritmo Distance Vector).
  - Ogni nodo elabora un vettore di stima dei costi (distanze) verso tutti gli altri nodi nella rete.
  - Il cammino a costo minimo viene calcolato in modo distribuito e iterativo scambiandosi informazioni con i nodi vicini.

Algoritmo Distance Vector

- Distribuito
  - ciascun nodo riceve parte delle informazioni da uno o più *nodi direttamente connessi* (*vicini*), effettua calcoli e comunica i risultati ai vicini
- Iterativo
- Asincrono: non richiede che tutti i nodi operino al passo con gli altri

Algoritmo Distance Vector



Equazione di Bellman-Ford

Formula per trovare il percorso a costo minimo tra un nodo sorgente *x* e un nodo destinazione *y*, tramite dei nodi intermedi (vicini *v*) supponendo che:

- Costo c(x,z) tra nodo sorgente *x* e il vicino *z* sia noto
- Distanza minima tra vicino *z* e destinazione *y* sia nota (d_{zy})

Distance vector algorithm

Bellman-Ford equation

d_{xy} := costo del cammino a costo minimo da x a y

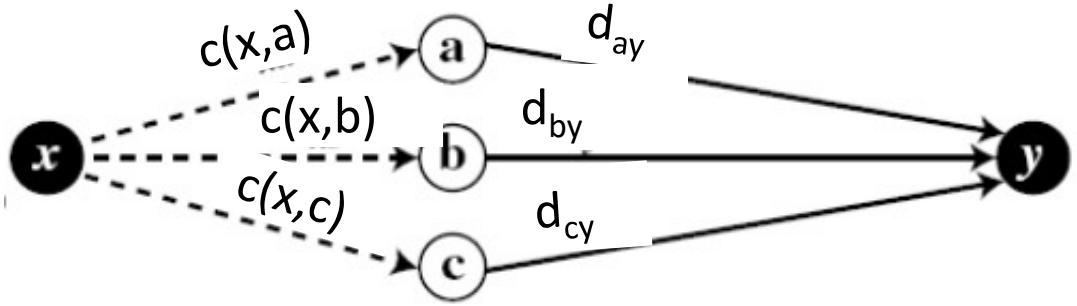
allora

$d_{xy} = min_v \{c(x,v) + d_{vy}\}$

distanza dal vicino v alla destinazione y

costo da x al vicino v

min su tutti i vicini v di x



Distance vector algorithm

- Ciascun nodo x inizia con una stima delle distanze verso tutti i nodi in N
- D_{xy} = stima del costo minimo da *x* a *y* per ogni y ϵ N
- Il nodo *x* mantiene quindi le seguenti informazioni:
  - Conosce il costo del collegamento verso ciascun vicino v: c(x,v)
  - x mantiene il suo vettore distanza $\mathbf{D}_x = [D_{xy} :$ per ogni y ϵ N]
  - Riceve i vettori distanza dei suoi vicini. Per ogni vicino *v*, *x* mantiene
    $\mathbf{D}_v = [D_{vy} :$ per ogni y ϵ N]

Distance vector algorithm

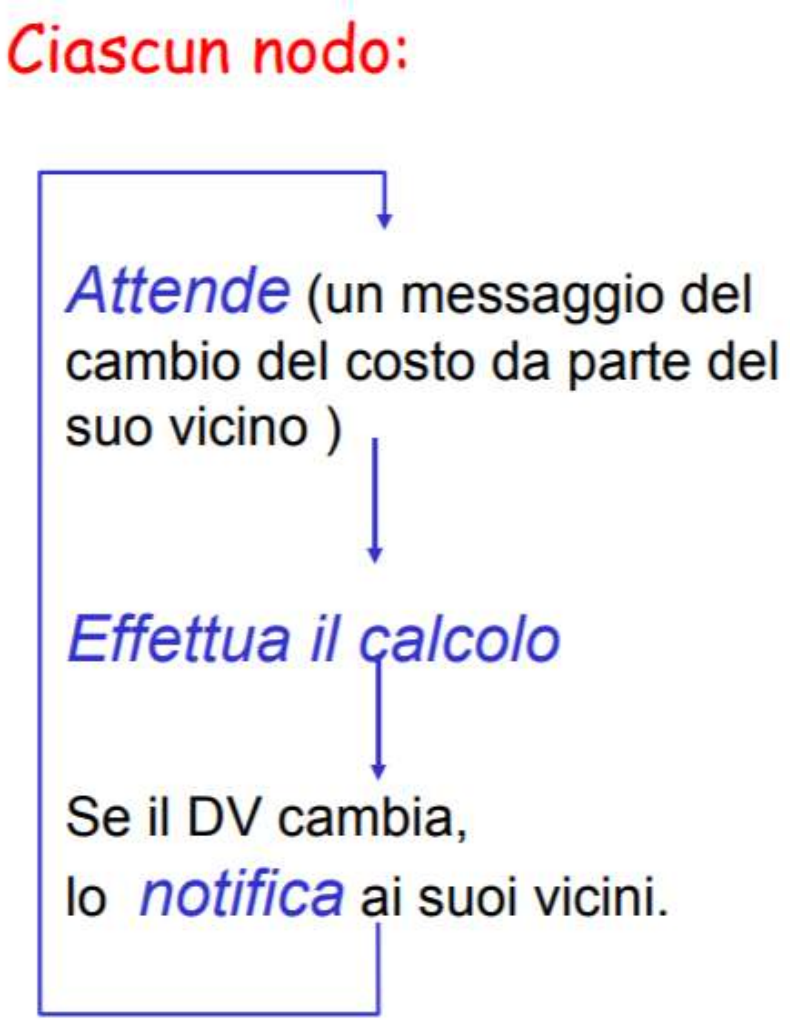
- Idea di base:
- Ogni nodo invia una copia del proprio vettore distanza a ciascuno dei suoi vicini.
- Quando un nodo *x* riceve un nuovo vettore distanza, DV, da qualcuno dei suoi vicini *v*, lo salva e usa l'equazione di Bellman-Ford per aggiornare il proprio vettore distanza

  $D_{xy} = min \{c(x,v) + D_{vy}\}$ per ogni $y \in N$

- Finché tutti i nodi continuano a cambiare i propri DV in maniera asincrona, ciascuna stima dei costi D_{xy} converge all'effettivo costo del percorso a costo minimo d_{xy}

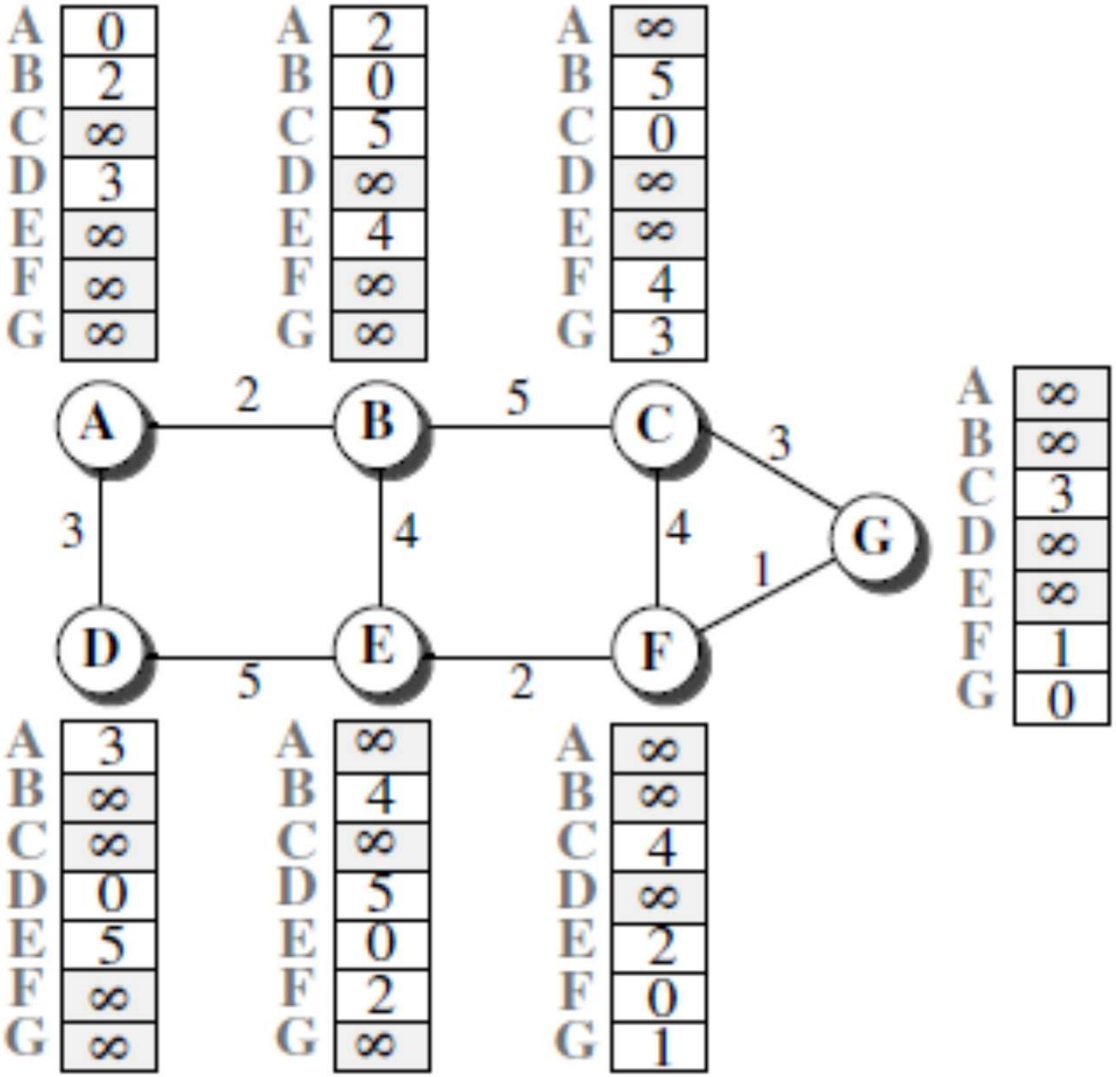
Distance vector algorithm

- Iterativo, asincrono: ogni iterazione locale è causata da:
  - cambio del costo di uno dei collegamenti locali
  - ricezione da qualche vicino di un vettore distanza aggiornato.
- Distribuito:
  - Ogni nodo aggiorna i suoi vicini solo quando il proprio DV cambia.
  - i vicini avvisano i rispettivi vicini solo se necessario.



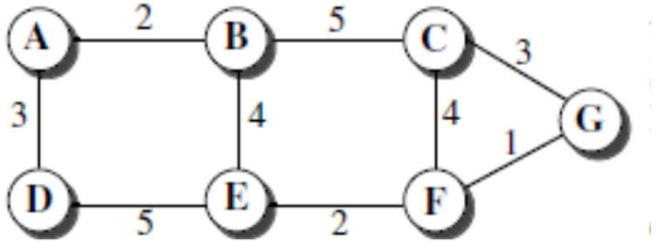
Distance vector routing

Vettori distanza iniziali dei nodi di una rete



Distance vector routing

Aggiornamento dei vettori di distanza



Nuovo B		Vecchio B		A	
A	2	A	2	A	0
B	0	B	0	B	2
C	5	C	5	C	∞
D	**5**	D	∞	D	3
E	4	E	4	E	∞
F	∞	F	∞	F	∞
G	∞	G	∞	G	∞

a. Primo evento: B riceve una copia del vettore di A.

Nuovo B		Vecchio B		E	
A	2	A	2	A	∞
B	0	B	0	B	4
C	5	C	5	C	∞
D	**5**	D	**5**	D	5
E	4	E	4	E	0
F	**6**	F	∞	F	2
G	∞	G	∞	G	∞

b. Secondo evento: B riceve una copia del vettore di E..

Distance vector routing: algoritmo

```
1  Distance_Vector_Routing ( )
2  {
3     // Inizializzazione (creazione dei vettori distanza iniziali del nodo)
4     D[me_stesso] = 0
5     for (y = 1 to N)
6     {
7        if (y è un vicino)
8           D[y] = c[me_stesso][y]
9        else
10          D[y] = ∞
11    }
12    spedisci il vettore {D[1], D[2], …, D[N]} a tutti i vicini
13    // Aggiornamento (usare il vettore ricevuto dal vicino per aggiornare quello locale)
14    repeat (sempre)
15    {
16       wait (un vettore D_w da un vicino w o un qualsiasi cambiamento negli archi)
17       for (y = 1 to N)
18       {
19          D[y] = min_v [ c[me_stesso][v]+D_v[y] ]        Equazione di Bellman-Ford
20       }
21       if (c'è un cambiamento nel vettore)
22          spedisci il vettore {D[1], D[2], …, D[N]} a tutti i vicini
23    }
24 } // Fine dell'algoritmo distance-vector routing
```

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) +$
$D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

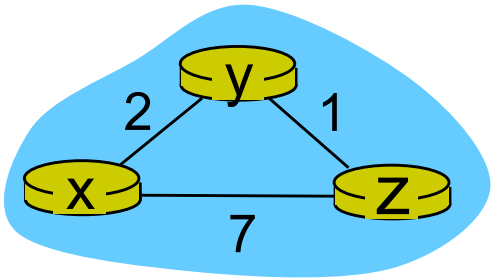
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

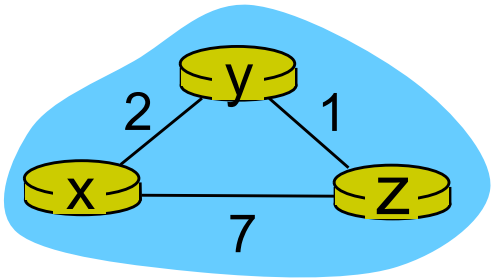
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

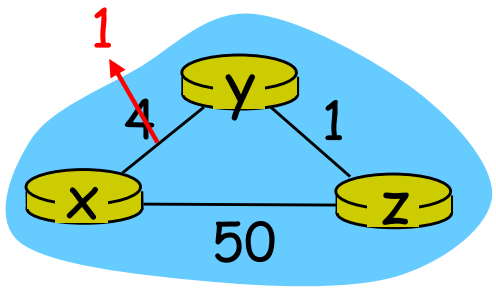


time

Distance vector: link cost changes

Se cambia il costo del collegamento:

- Il nodo rileva il cambiamento del costo
- Ricalcola il vettore distanza
- Se il DV cambia, avvisa i vicini



"good news travels fast"

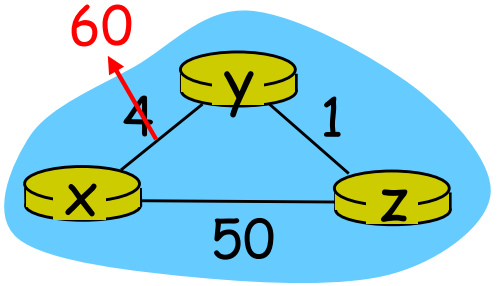
t_0 : *y* si accorge del cambiamento di costo, aggiorna il DV, informa i vicini.

t_1 : *z* riceve il DV di *y*, aggiorna la sua tabella, aggiorna il suo costo minimo verso *x* , invia il suo DV ai suoi vicini.

t_2 : *y* riceve l'aggiornamento di z, aggiorna la tabella. I costi minimi di *y* non cambiano.

Distance vector: link cost changes

- *bad news travels slow* – problema "count to infinity"!
- varie iterazioni prima che l'algoritmo si stabilizzi
- Count-to-infinity problem



y si accorge che il collegamento verso *x* ha un nuovo costo di 60, ma *z* ha detto che la sua distanza da x è 5. Quindi *y* calcola la nuova distanza verso x, 6, e invia una notifica a z.

z apprende che il percorso verso x tramite y ha costo 6, così aggiorna la sua distanza verso x (7) e invia una notifica a y

Il ciclo proseguirà per 44 iterazioni (scambi di messaggi tra y e z), fino a quando z considera il costo del proprio percorso attraverso y maggiore di 50.

A questo punto, z determina che il percorso a costo minimo verso x passa attraverso la connessione diretta a x, e y instraderà verso x passando per z.

DV: count-to-infinity problem

- Split-horizon with Poisoned reverse

  Se nodo X inoltra a V i pacchetti destinati a Z allora

  – X invia a V $D_X[Z]=\infty$

  – le rotte ricevute tramite un'interfaccia devono essere pubblicizzate indietro da quell'interfaccia con una metrica non raggiungibile

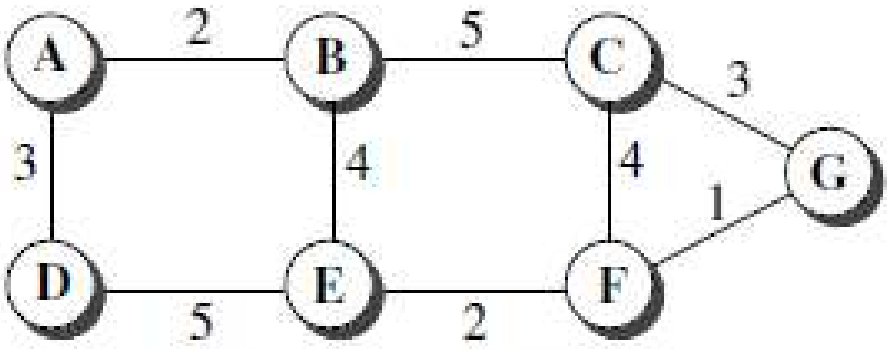
  – *A route learned through an interface will be advertised as unreachable through that same interface*

  – https://tools.ietf.org/html/rfc7868#page-38

Link-state algorithm

- Algoritmo «globale»
- La topologia di rete e tutti i costi dei collegamenti sono noti a tutti i nodi attraverso il “link-state broadcast”.
- Tutti i nodi dispongono delle stesse informazioni
- Calcola il cammino a costo minimo da un nodo (origine) a tutti gli altri nodi della rete.
  - Algoritmo di Dijkstra
  - È iterativo: dopo la k-esima iterazione i cammini a costo minimo verso k nodi di destinazione sono noti.
- Crea una tabella d’inoltro per quel nodo

Link-state database



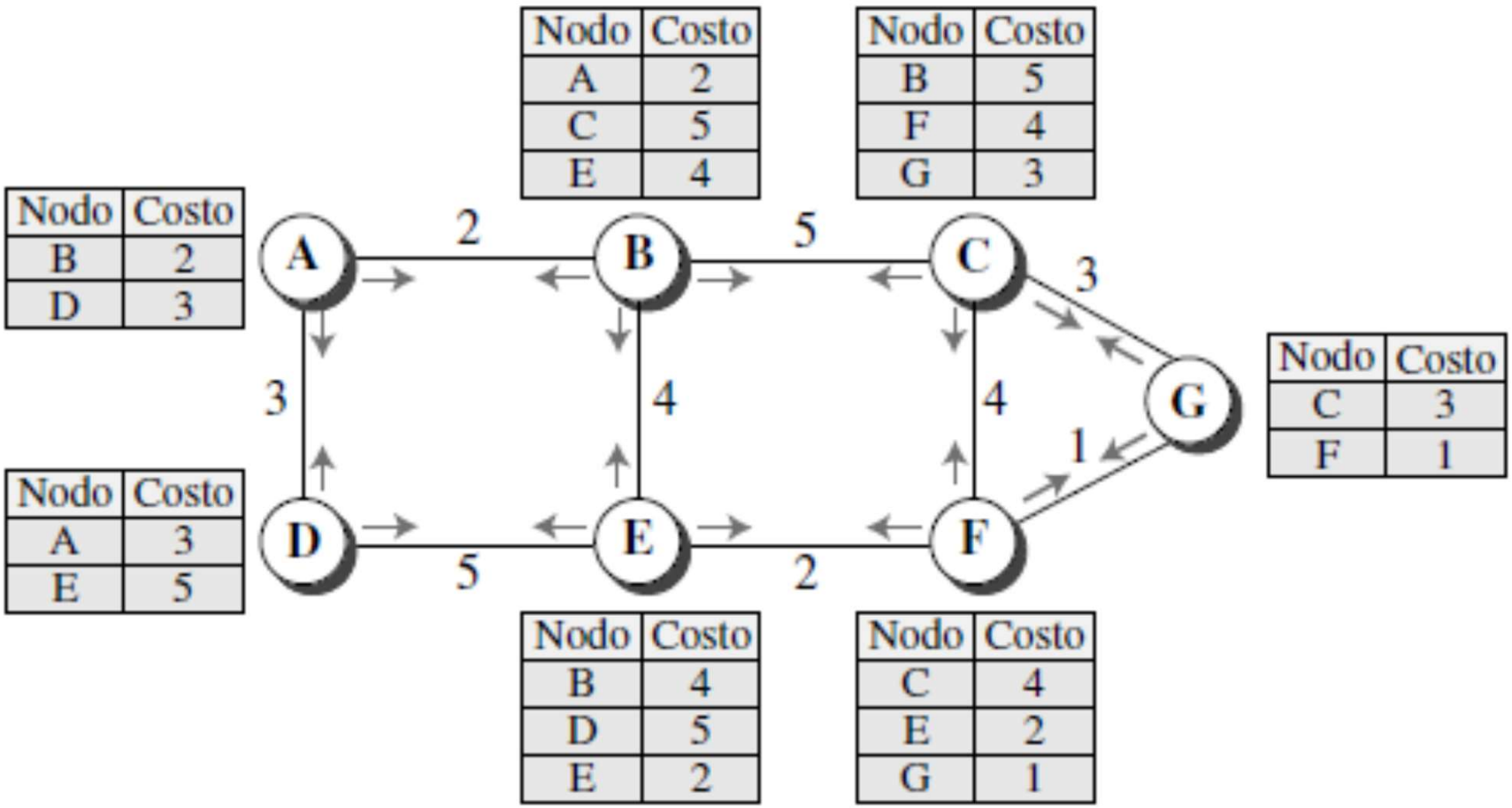
a. Il grafo pesato

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Il link-state database

LSP

LSP (LS Packet) creati ed inviati da ciascun nodo



Notazione

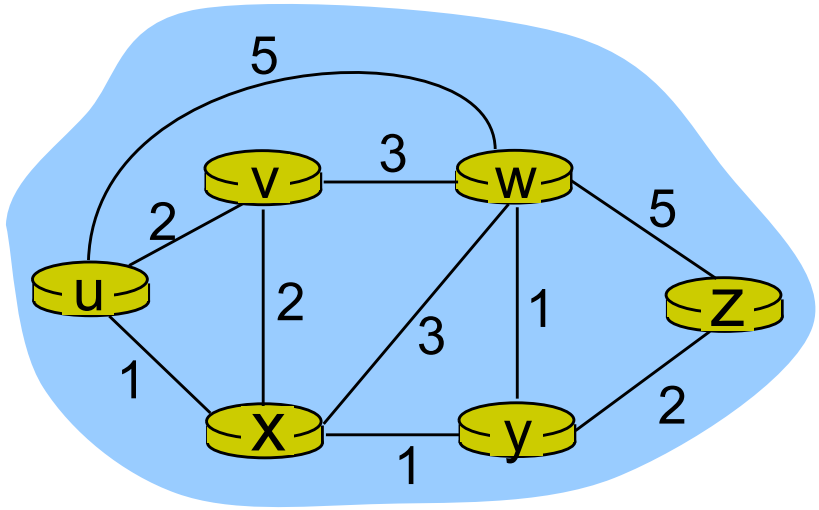
- c(x,y): costo del collegamenti dal nodo x al nodo y
  - ∞ se non sono adiacenti.
- D(v): costo del cammino dal nodo origine alla destinazione v per l'iterazione corrente
- p(v): immediato predecessore di v lungo il cammino
- N': sottoinsieme di nodi per cui il cammino a costo minimo dall'origine è definitivamente noto

Dijsktra's Algorithm

1 ***Inizializzazione del nodo u:***
2 N' = {u}
3 Per tutti i nodi v
4 se v è adiacente a u
5 allora D(v) = c(u,v)
6 altrimenti D(v) = ∞
7
8 ***Loop***
9 trova w non in N' tale che D(w) sia minimo
10 aggiungi w a N'
11 aggiorna D(v) per tutti v adiacenti 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 cammino minimo noto verso w più il costo da w a v */
14 ***Finché tutti i nodi sono in N'***

Algoritmo di Dijkstra: esempio

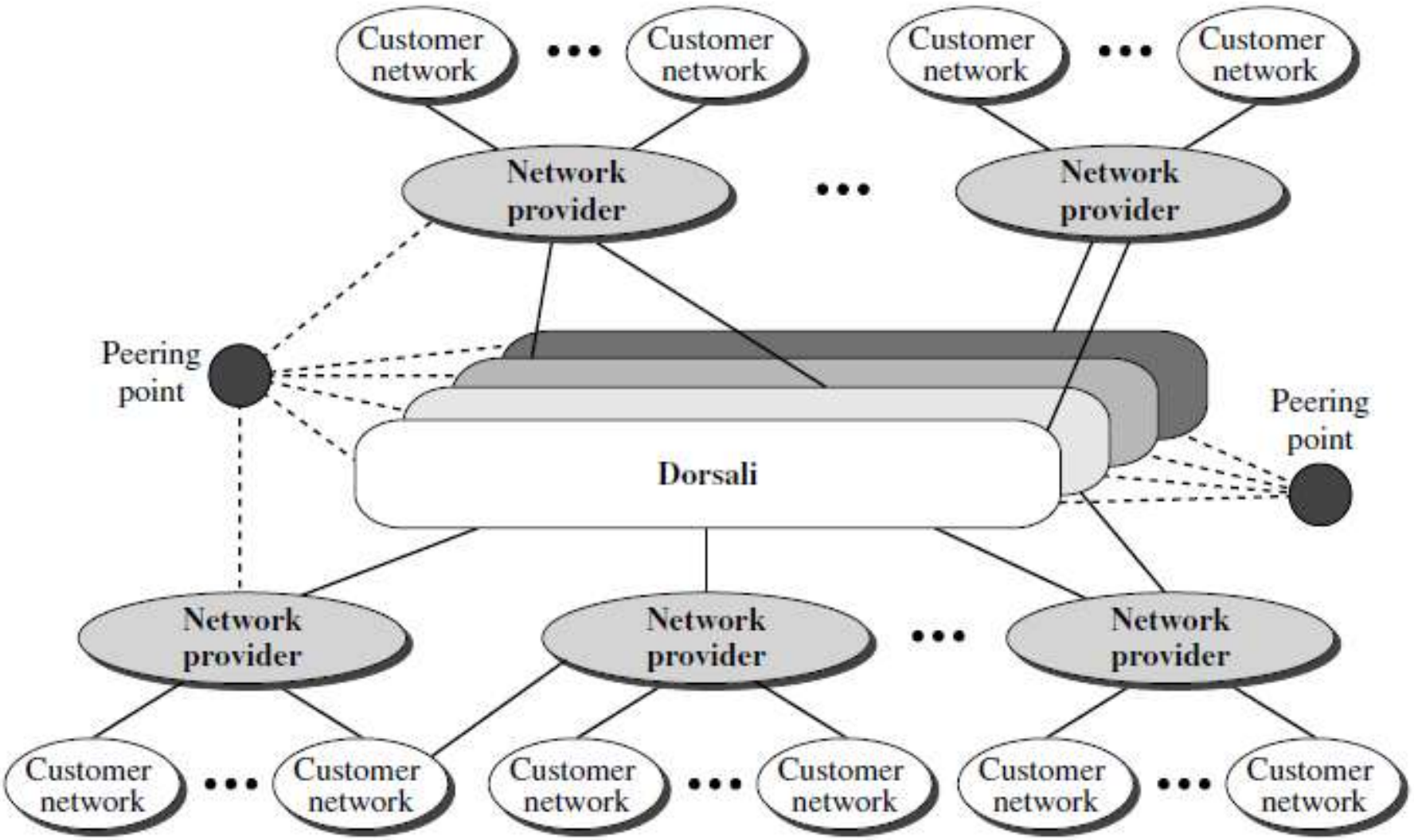
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux					∞
2	uxy					
3	uxyv					
4	uxyvw					
5	uxyvwz					



LS vs DV

- Complessità dei messaggi:
  - LS: con n nodi, E collegamenti, implica l'invio di O(nE) messaggi.
  - DV: richiede scambi tra nodi adiacenti.
- Velocità di convergenza:
  - LS: l'algoritmo $O(n^2)$ richiede O(nE) messaggi.
  - DV: può convergere lentamente. Può presentare cicli d'instradamento. Può presentare il problema del conteggio all'infinito.
- Robustezza: cosa avviene se un router funziona male?
- LS:
  - un router può comunicare via broadcast un costo sbagliato per uno dei suoi collegamenti connessi (ma non per altri).
  - I nodi si occupano di calcolare soltanto le proprie tabelle.
- DV:
  - un nodo può comunicare cammini a costo minimo errati a tutte le destinazioni.
  - la tabella di ciascun nodo può essere usata degli altri.
  - Un calcolo errato si può diffondere per l'intera rete.

Struttura di Internet



Instradamento gerarchico

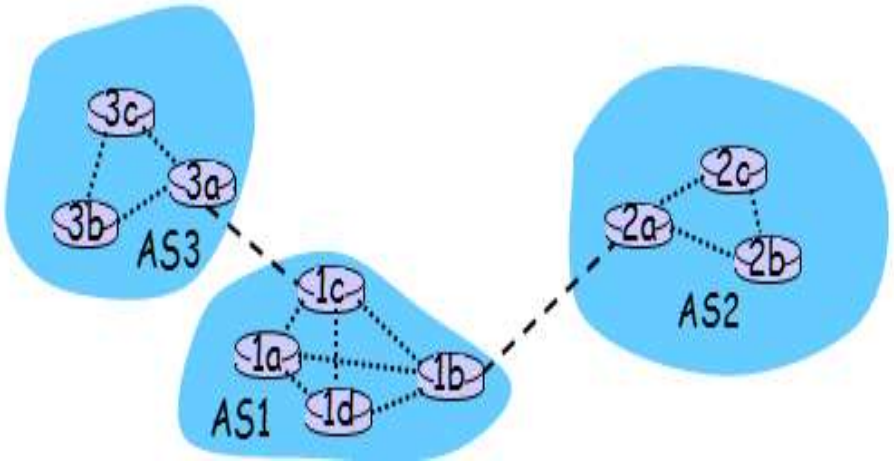
- La visione di una rete costituita da un insieme di router omogenei interconnessi è semplicistica
  - Problemi di scalabilità e autonomia amministrativa
- Nella realtà i router sono organizzati in **sistemi autonomi (AS)**.
  - un AS è un gruppo di router sotto lo stesso controllo amministrativo
    - Es. router e collegamenti di un ISP possono formare un unico AS
    - Un ISP può comunque partizionare la sua rete in più AS
  - gli AS decidono autonomamente i protocolli e le politiche di routing che intendono adottare al loro interno

Instradamento gerarchico

- I protocolli di routing all'interno di un AS sono detti Interior Gateway Protocol (IGP)
- I protocolli di routing fra AS sono detti Exterior Gateway Protocol (EGP)
- **Quindi, all'interno di un sistema autonomo:**
  - **i router sono sotto uno stesso controllo amministrativo**
  - **usano lo stesso protocollo di routing intra-AS (es, RIP, OSPF)**

Protocolli di instradamento

- Internet partizionata in ***sistemi autonomi (AS)***
  - AS stub: collegato solo a un altro AS
  - AS multihomed: collegato a più di un altro AS (ma trasporta – come lo stub - solo traffico di cui è origine o destinazione)
  - AS di transito



Lista di AS italiani:

https://ipinfo.io/countries/it

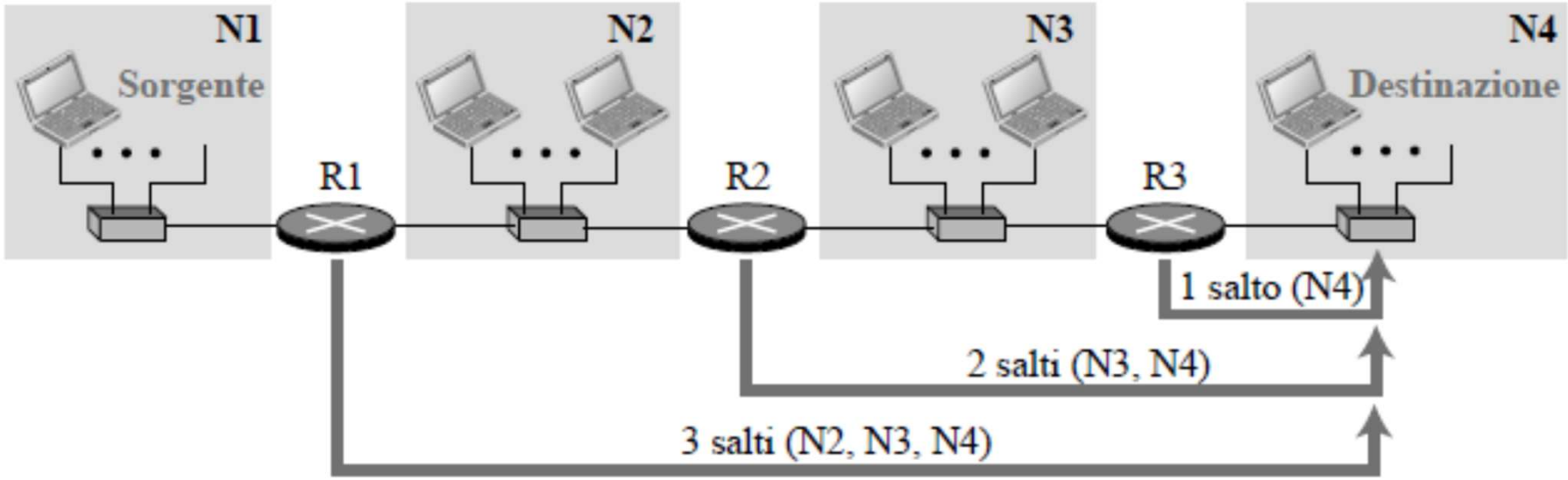
Routing INTRA-AS e routing INTER-AS

- INTRA-AS routing protocol determina (da solo) rotte per le destinazioni **interne** ad AS
  - Routing Information Protocol (RIP) - Algoritmo tipo DV
  - Open Shortest Path First (OSPF) - Algoritmo di tipo LS
- INTRA-AS e INTER-AS routing protocol determinano (insieme) rotte per le destinazioni **esterne** all'AS
- Border gateway protocol BGP
  - INTER-AS routing protocol (standard de facto)
  - Consente di conoscere le destinazioni raggiungibili attraverso sistemi autonomi vicini
  - Propaga le informazioni di raggiungibilità ai router interni del proprio AS
  - Determina percorsi buoni verso le sottoreti di destinazione

Protocolli

RIP (Routing Information Protocol)

- Intra-AS
- Metrica = #sottoreti attraversate (max 15)
  - Inclusa la rete dove si trova la destinazione
- Distance Vector con Poisoned Reverse ("inf"=16)



Esempio

Tabella d'inoltro per R1

Rete di destinazione	Prossimo router	Costo (in hop)
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

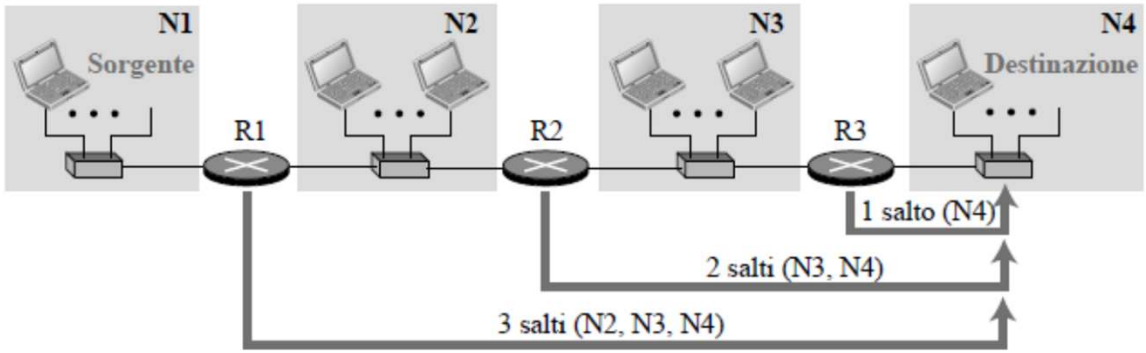


Tabella d'inoltro per R3

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

Tabella d'inoltro per R2

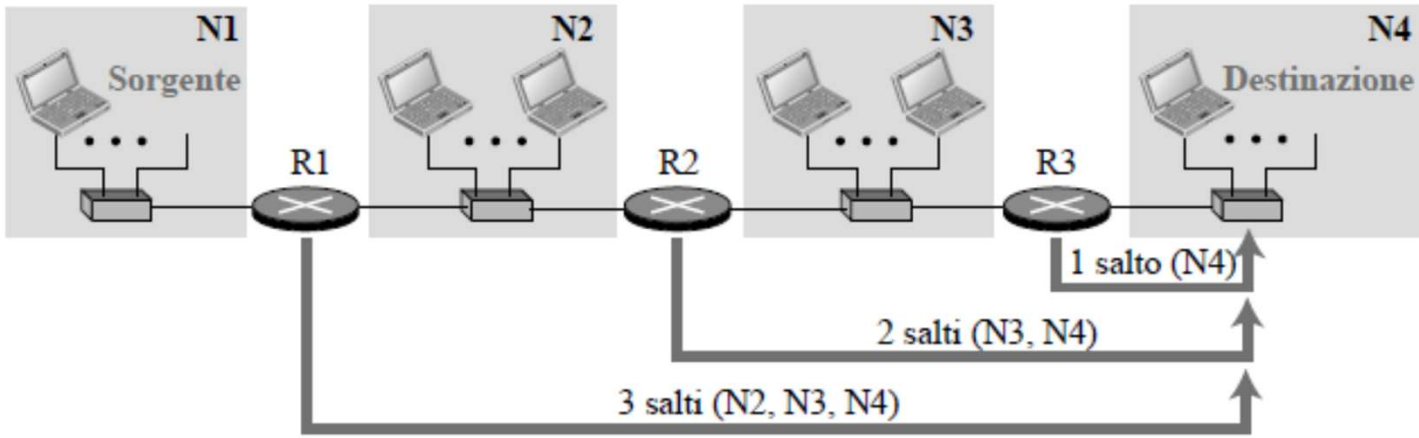
Rete di destinazione	Prossimo router	Costo (in hop)
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

RIP

- Processo demone `routed` (UDP, porta 520)
- Esempio tabella di inoltro:

Tabella d'inoltro per R2

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R1	2
N2	—	1
N3	—	1
N4	R3	2



- Aggiornamenti inviati ogni 30sec o se tabella inoltro cambia

RIP: algoritmo

Un nodo invia la tabella di inoltro ai propri vicini (periodi di circa 30 secondi)

Nodo R riceve advertisement da vicino V con $D_V[Y]$:

$$D_R[Y] = 1 + D_V[Y]$$

Aggiunge un hop (c(x,v)) e considera come next-hop V

Il nuovo percorso da R a Y viene inserito in tabella se:

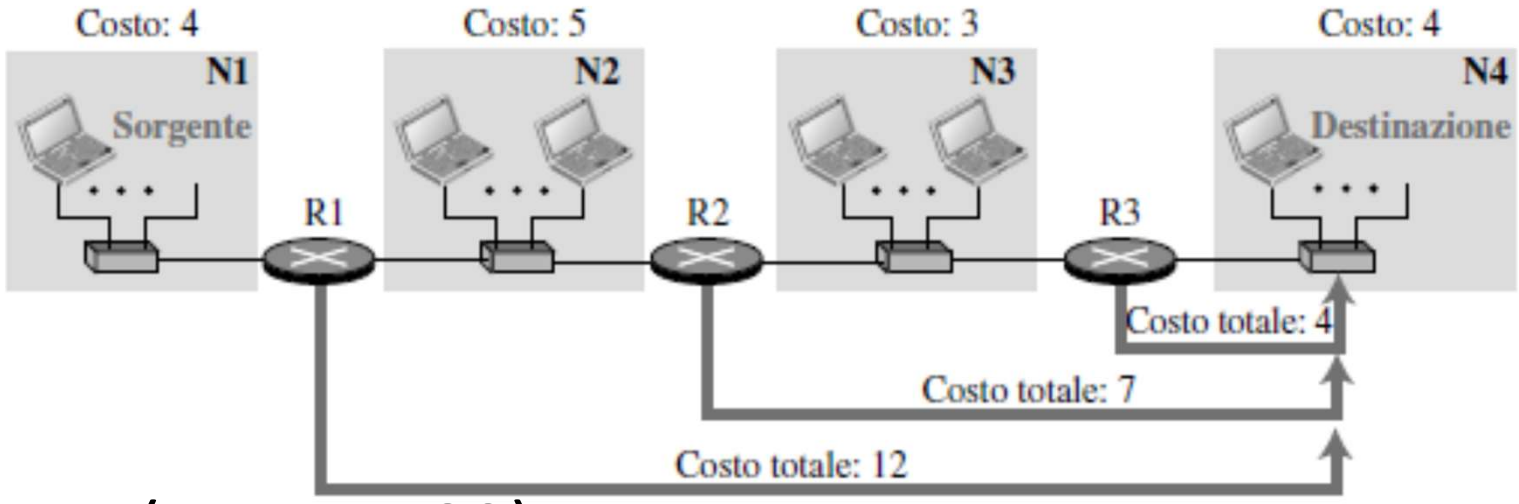
- se è un percorso non presente nella tabella e quindi viene aggiunto
- se $D_V[Y]+1 < D_{Rold}[Y]$ costo ricevuto inferiore a quello del vecchio percorso
- Se V è nextHop del vecchio e nuovo percorso, ma il costo è cambiato (diminuito o incrementato)

OSPF (Open Shortest Path First)

- Intra-AS
- Link State
- metrica: può deciderla l'amministratore: latenza, affidabilità, banda, numero di hop, ecc.

Tabella d'inoltro per R2

Rete di destinazione	Prossimo router	Costo
N1	R1	9
N2	——	5
N3	——	3
N4	R3	7

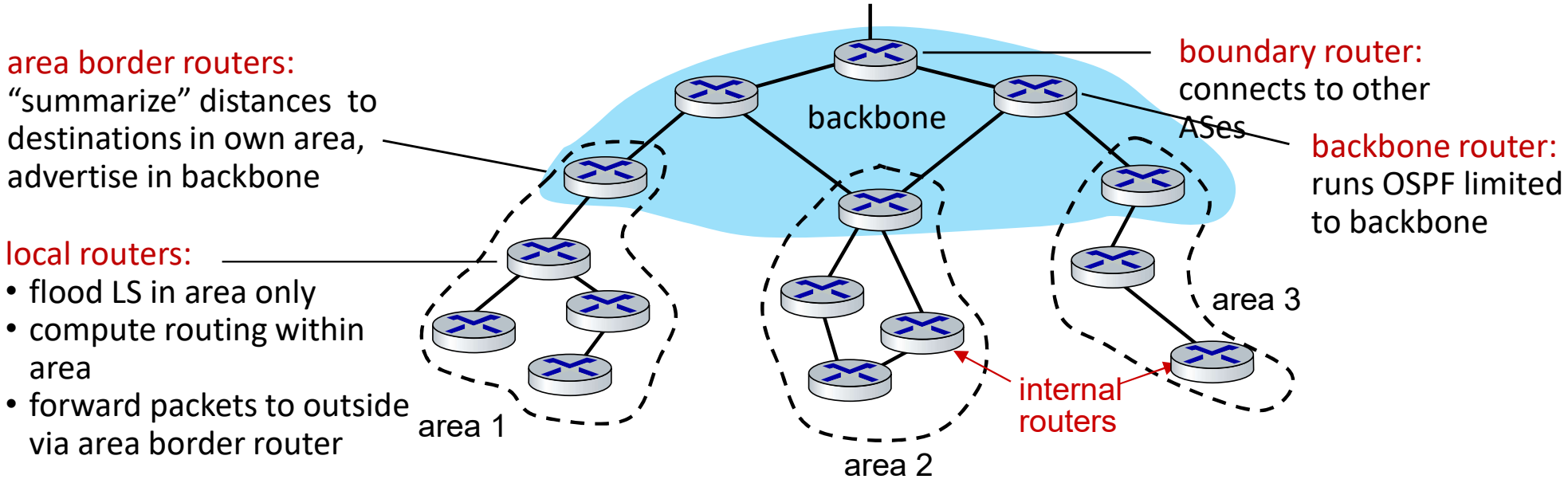


- OSPF usa IP (n. prot. 89)

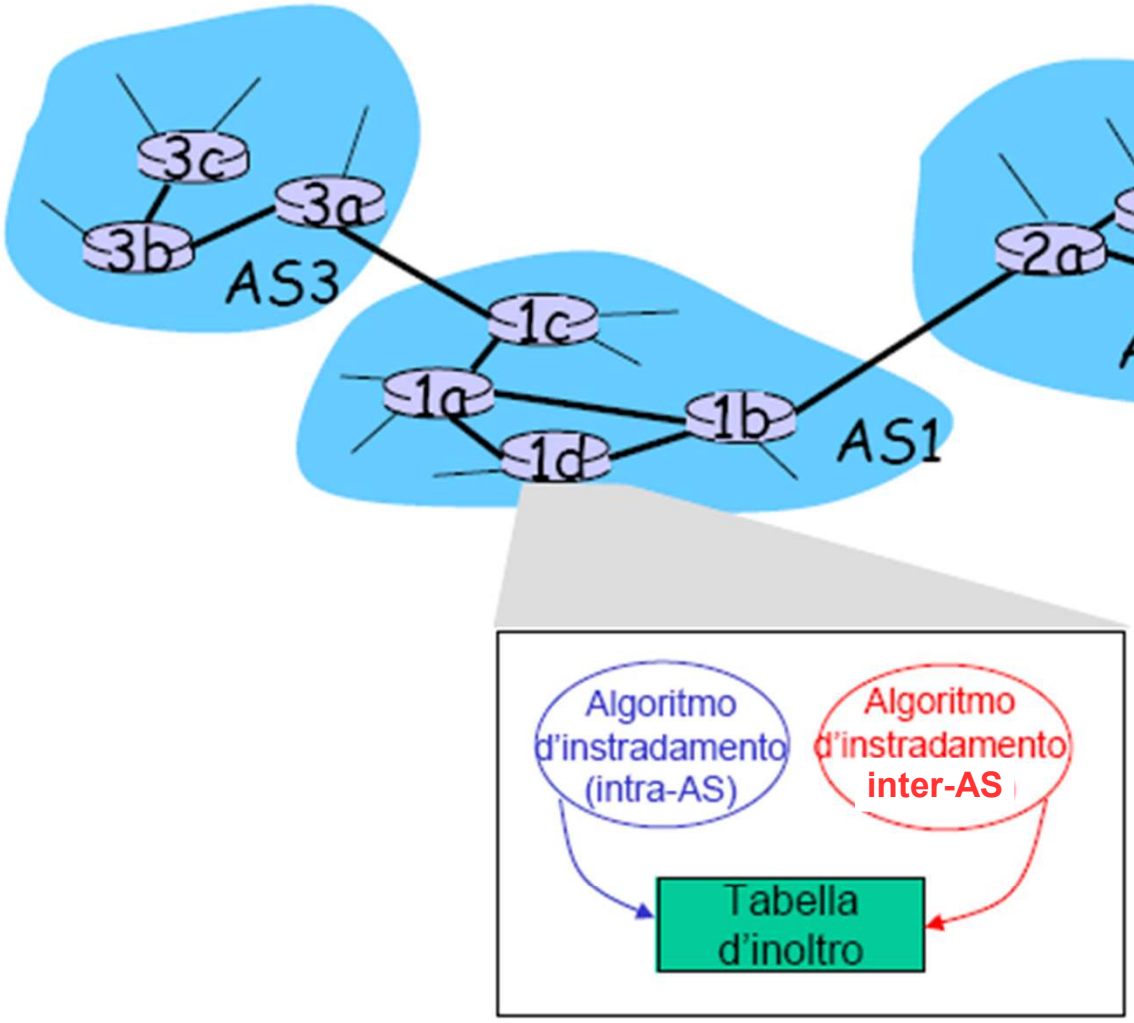
OSPF gerarchico

- AS può essere partizionato in *aree* (per ridurre flooding di Link State Packet), una delle quali fa da dorsale (backbone area)

  - two-level hierarchy: local area, backbone.
    - link-state advertisements flooded only in area, or backbone
    - each node has detailed area topology; only knows direction to reach other destinations

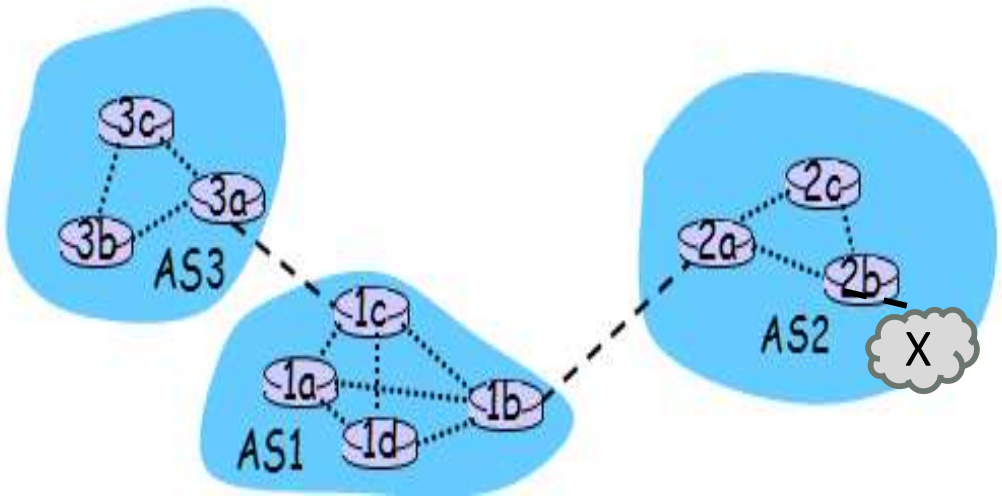


Sistemi autonomi interconnessi



- Ciascun sistema autonomo sa come inoltrare pacchetti lungo il percorso ottimo verso qualsiasi destinazione interna al gruppo
  - I sistemi AS2 e AS3 hanno tre router ciascuno
  - I protocolli d'instradamento dei tre sistemi autonomi non sono necessariamente gli stessi
  - I router 1b, 1c, 2a e 3a sono gateway

Esempio

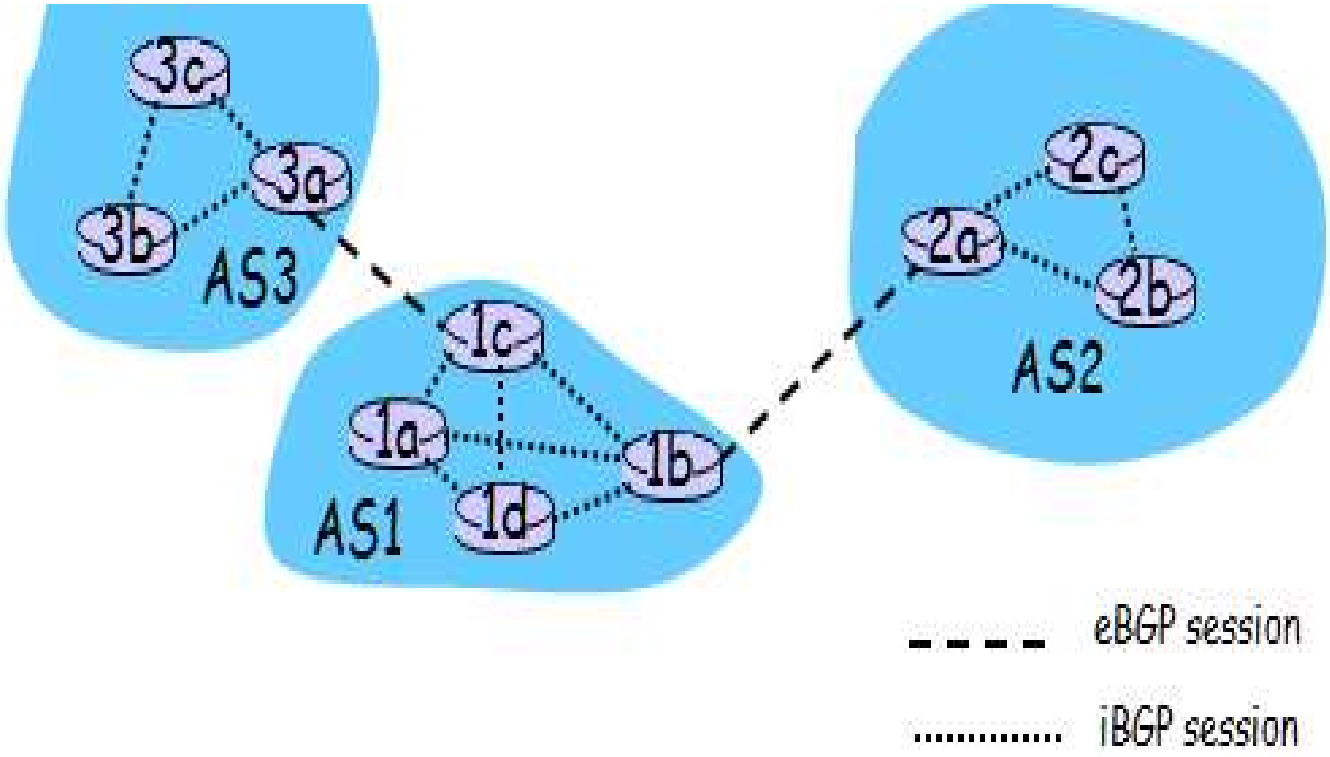


- AS1 scopre (grazie a INTER-AS routing protocol) che una sottorete X è raggiungibile via AS2 (a cui AS1 è collegato mediante il gateway 1b)
- AS1 propaga (con INTER-AS routing protocol) tale informazione al suo interno
- Un router R (es. 1d in figura) di AS1 riceve l'informazione "rete X raggiungibile via AS2":

  Aggiorna (se necessario) la tabella di inoltro

BGP (Border Gateway Protocol)

- BGP4: unico protocollo INTER-AS usato in Internet
- protocollo di tipo distance-vector decentralizzato e asincrono
- Coppie di router si scambiano info su connessioni TCP: sessioni BGP ***esterne*** e sessioni ***interne***



BGP

- Aggregazione indirizzi
  - destinazioni rappresentate da prefissi CIDR
- BGP mette a disposizione di ciascun router un modo per:
  - **Ottenere informazioni sulla raggiungibilità** dei prefissi di sottorete da parte dei sistemi confinanti
    - In particolare, BGP consente a ciascuna sottorete di comunicare la propria esistenza al resto di Internet. Basta che una sottorete invii un annuncio “Esisto, son qui” e BGP si assicura che lo sappiano tutti i router di Internet
  - **Determinare i percorsi “ottimi” verso le sottoreti**.
    - Un router può venire a conoscenza di più cammini verso uno specifico prefisso; per determinare il migliore, esegue localmente BGP, sulla base delle informazioni di raggiungibilità e delle politiche del sistema

BGP

- Pubblicizzare un prefisso significa impegnarsi a instradare pacchetti destinati a reti in quel prefisso

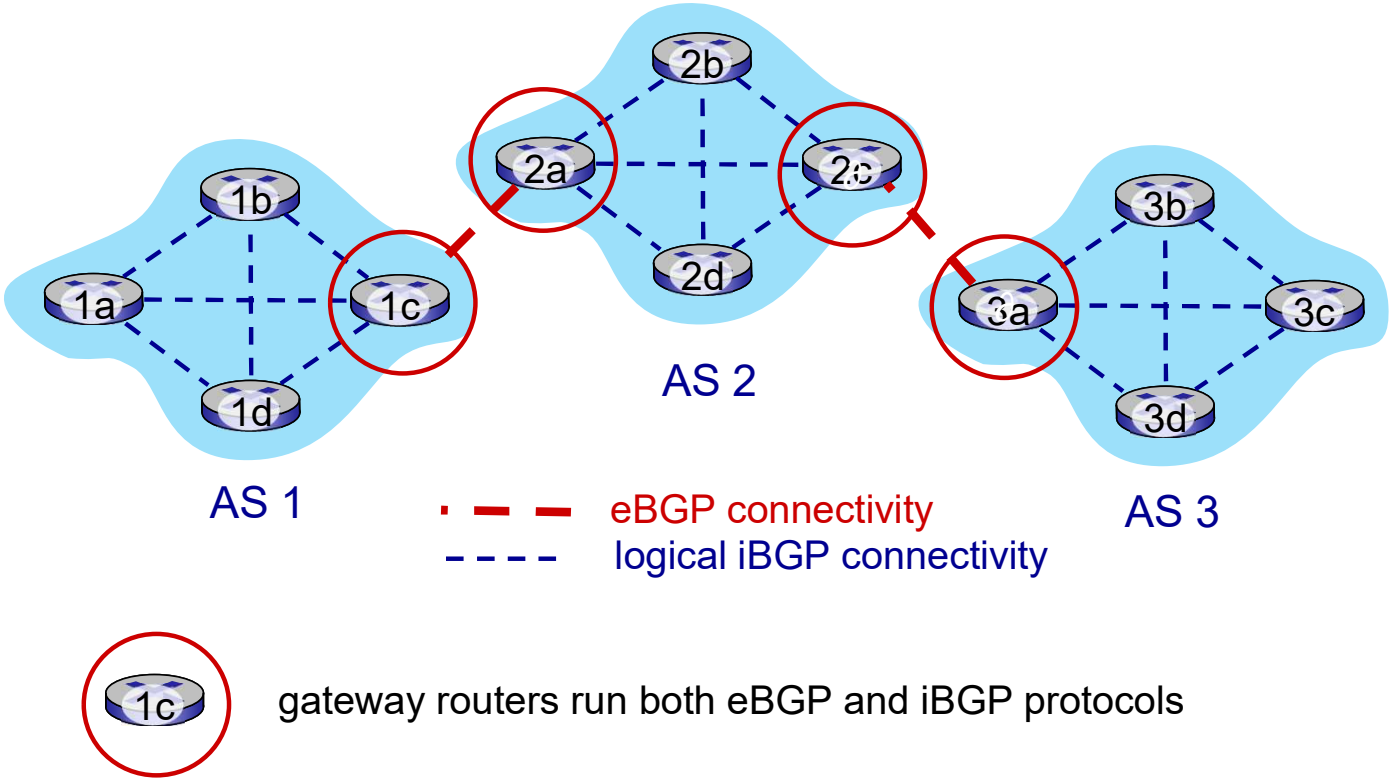
- Distribuzione di informazioni su raggiungibilità:

  eBGP: gateway riceve informazioni da gateway di altri AS su sessione eBGP

  iBGP: gateway distribuisce informazioni ai router interni della propria rete su sessioni iBGP

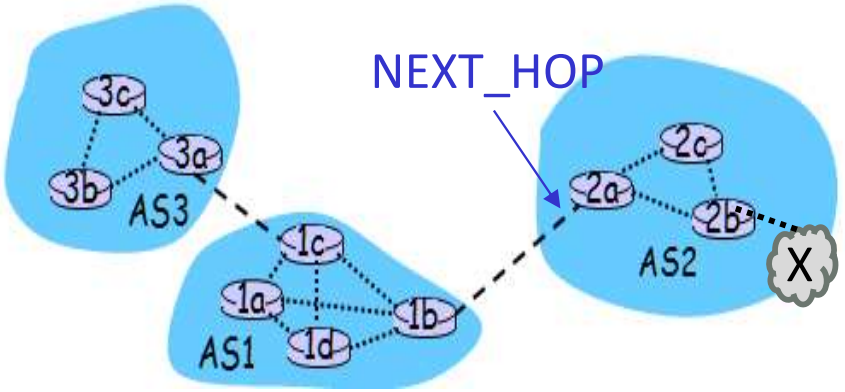
  altro gateway dell'AS può ri-pubblicizzare info con eBGP ...

eBGP, iBGP connections



BGP

- Advertisement (ADV) BGP
  - "route" = prefisso + attributi
  - i due attributi più importanti sono
    - AS_PATH
      - sequenza degli AS attraversati nel path pubblicizzato dall'advertisement
      - usato per
        - scartare advertisement già ricevuti
        - scegliere tra più percorsi per lo stesso prefisso
    - NEXT_HOP
      - L'attributo NEXT-HOP indica l'indirizzo IP del primo router lungo un percorso annunciato (al di fuori dell'AS che riceve l'annuncio) a un dato prefisso di rete



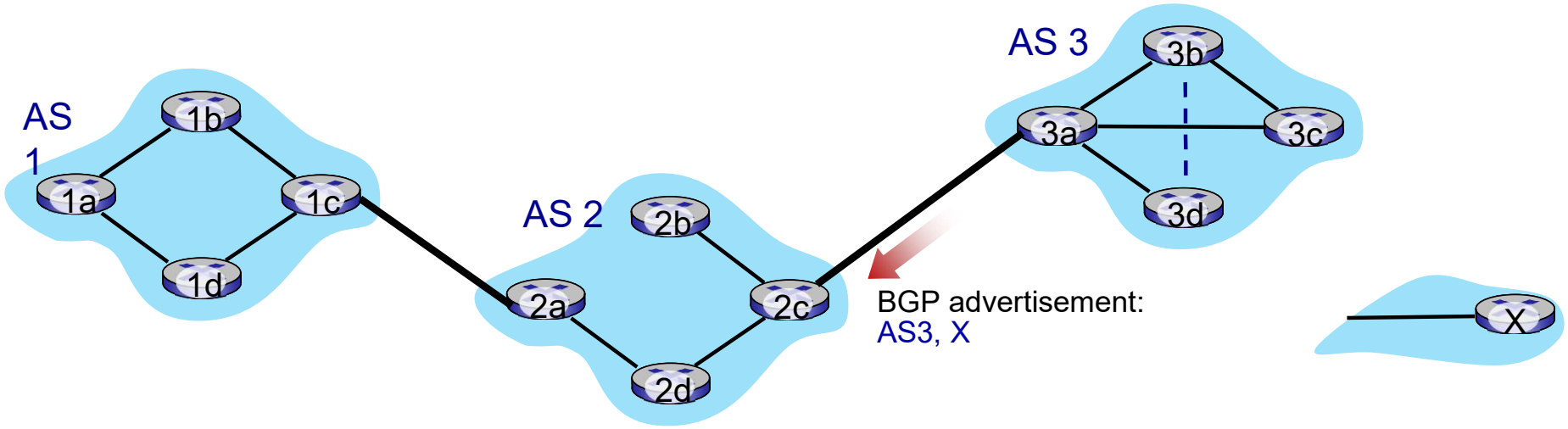
Per AS1 NEXT_HOP verso prefisso X è l'interfaccia di 2a

BGP

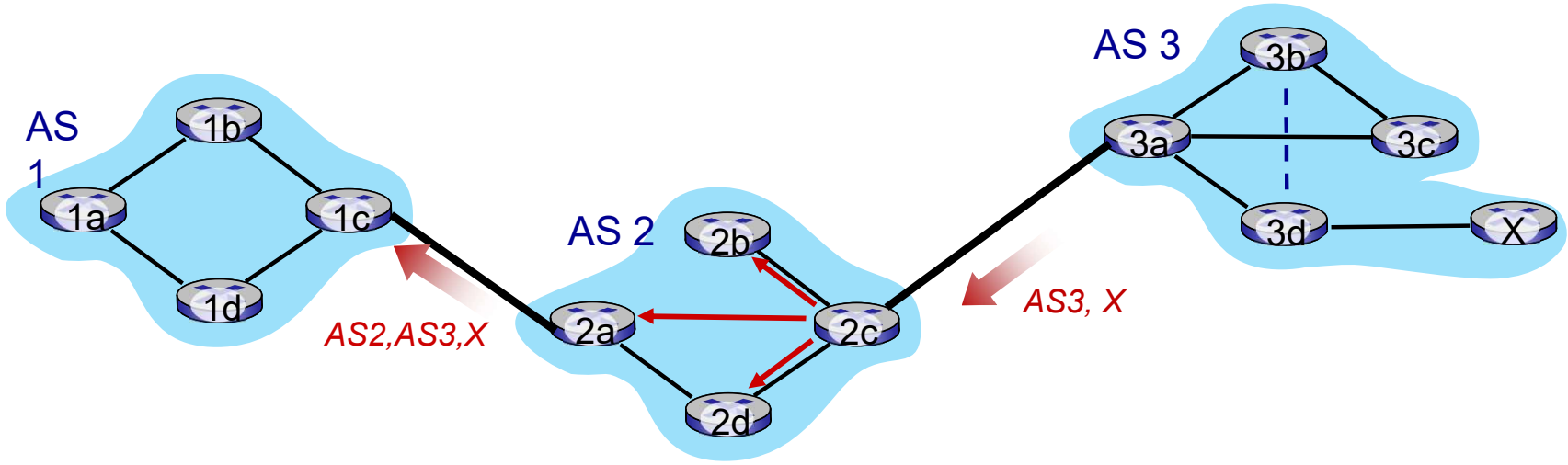
- Policy-based routing
  - Politiche di importazione
  - Il gateway che riceve un route advertisement una *import policy* per accettare/rifiutare il percorso (e.g., never route through AS Y).
  - Si usa l'AS policy anche per determinare se annunciare (*advertise) il percorso* ad altri AS vicini
- Scelta delle rotte
  - router può ricevere più di 1 rotta per lo stesso prefisso
  - Sequenza di regole (principale)
    1. Attributo di "preferenza locale" (LOCAL-PREF scelta da amministratore o impostato dai router dell'AS) → vengono selezionati quelli coi valori più alti
    2. Shortest AS-PATH
    3. Closest NEXT-HOP interface ("hot potato routing")

BGP basics

- BGP session: two router BGP ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X

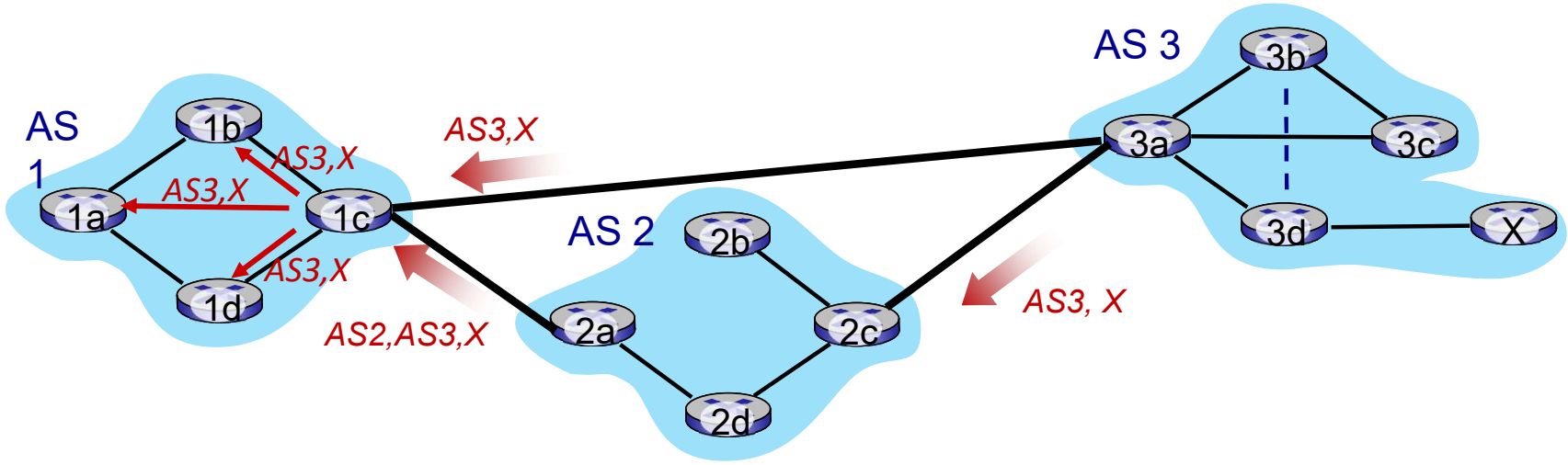


BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

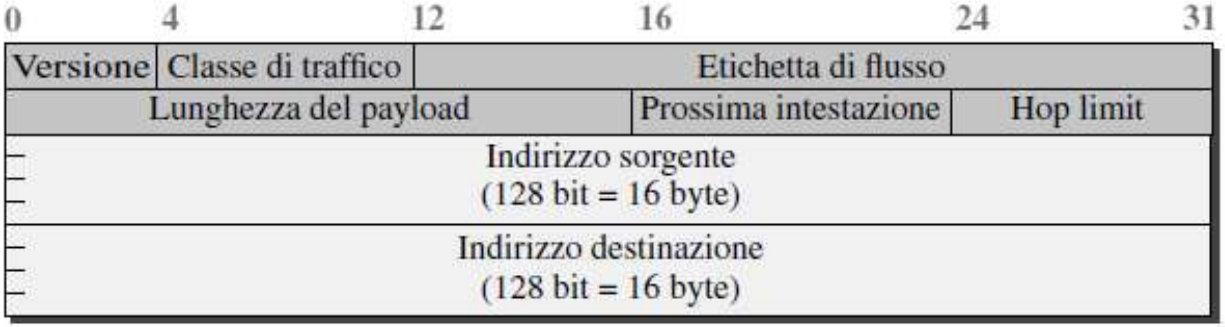
BGP path advertisement (more)



gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- based on *policy,* AS1 gateway router 1c chooses path *AS3,X* and advertises path within AS1 via iBGP

IPv6



Motivazioni (e soluzioni)

- Esaurimento indirizzi a 32 bit

  → indirizzi a 128 bit

- Velocizzare elaborazione e forwarding pacchetti

  → lunghezza fissa (40 byte) dell'header

  → eliminato checksum

- Risparmiare ai router costo frammentazione

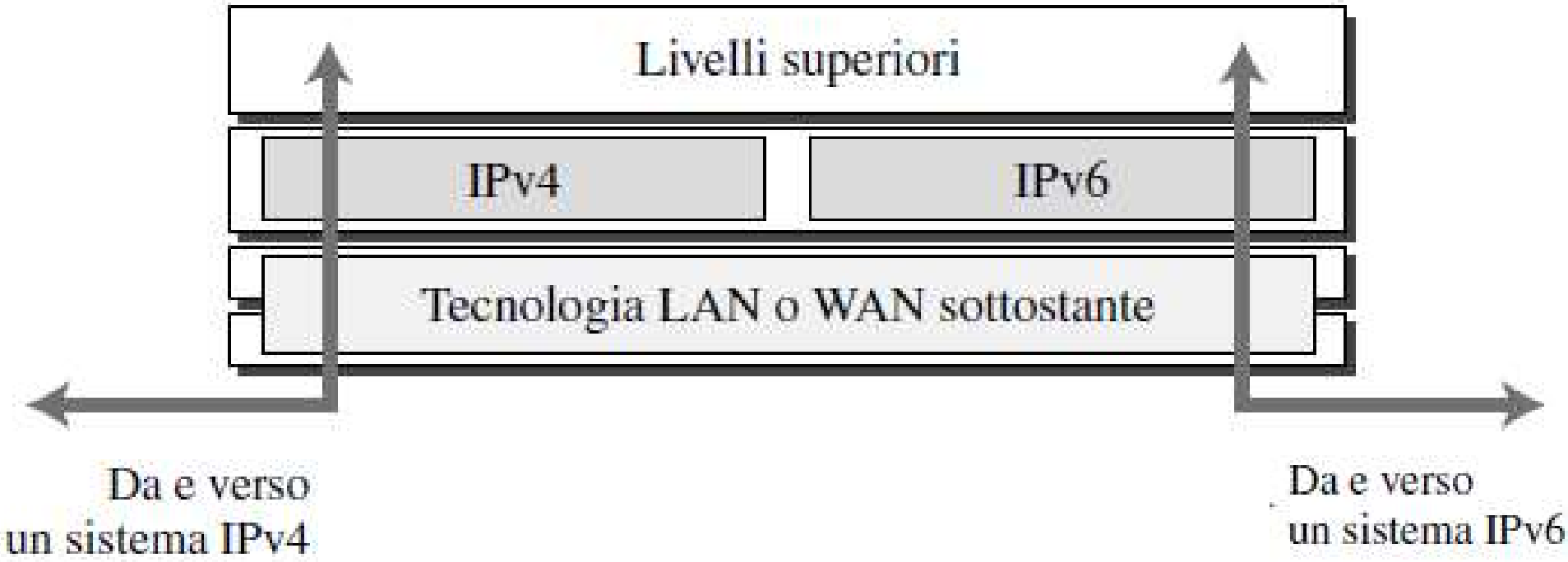
  → ICMPv6: msg "packet too big" (sorgente deve preoccuparsi della frammentazione)

- Facilitare QoS

  → introdotta "flow label" per identificare datagram appartenenti allo stesso "flow"

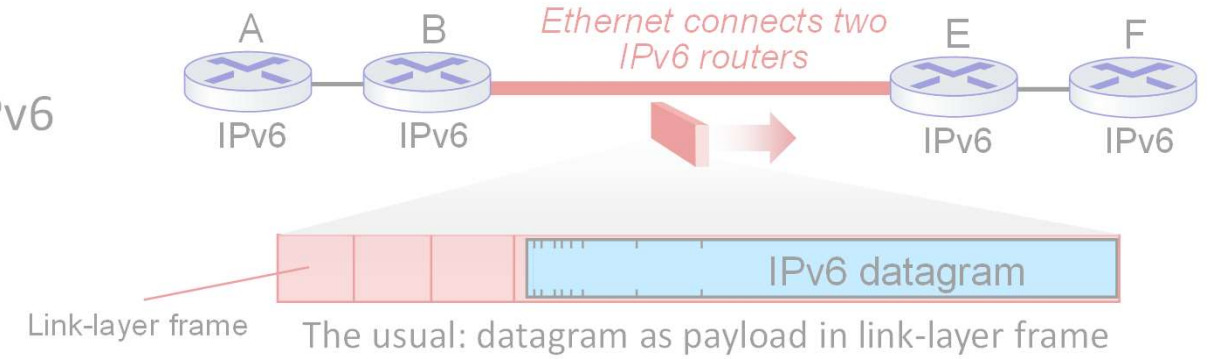
  (p.e. insiemi di pacchetti che richiedono un trattamento speciale come QoS p.e. audio-video, high-priority traffic)

Dual stack

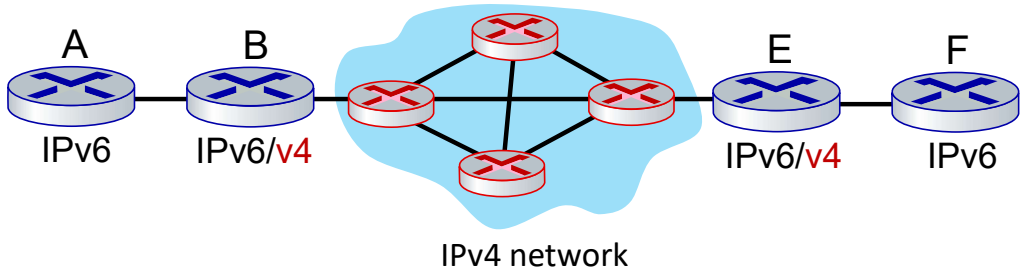

Livelli superiori
IPv4
IPv6
Tecnologia LAN o WAN sottostante
Da e verso
un sistema IPv4
Da e verso
un sistema IPv6

Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

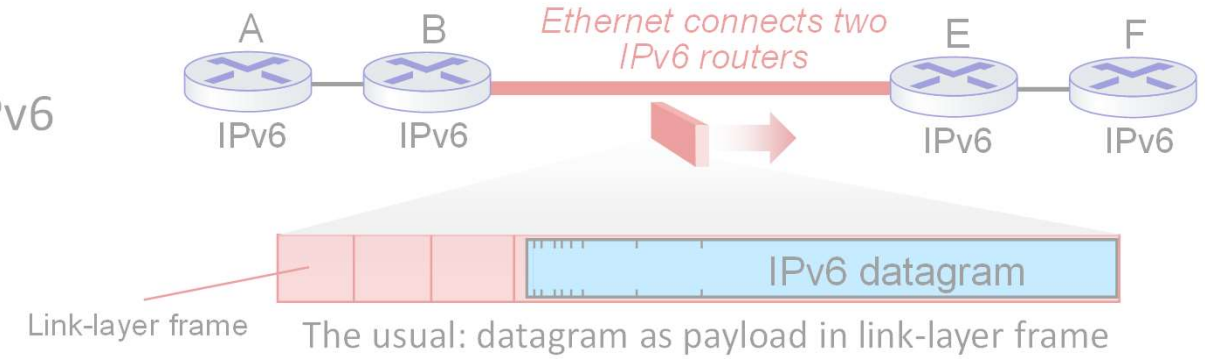


IPv4 network connecting two IPv6 routers

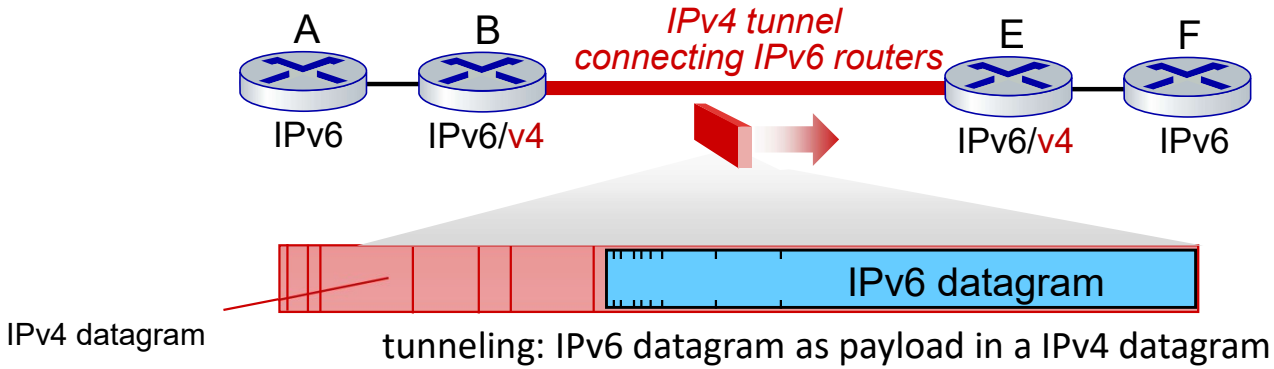


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

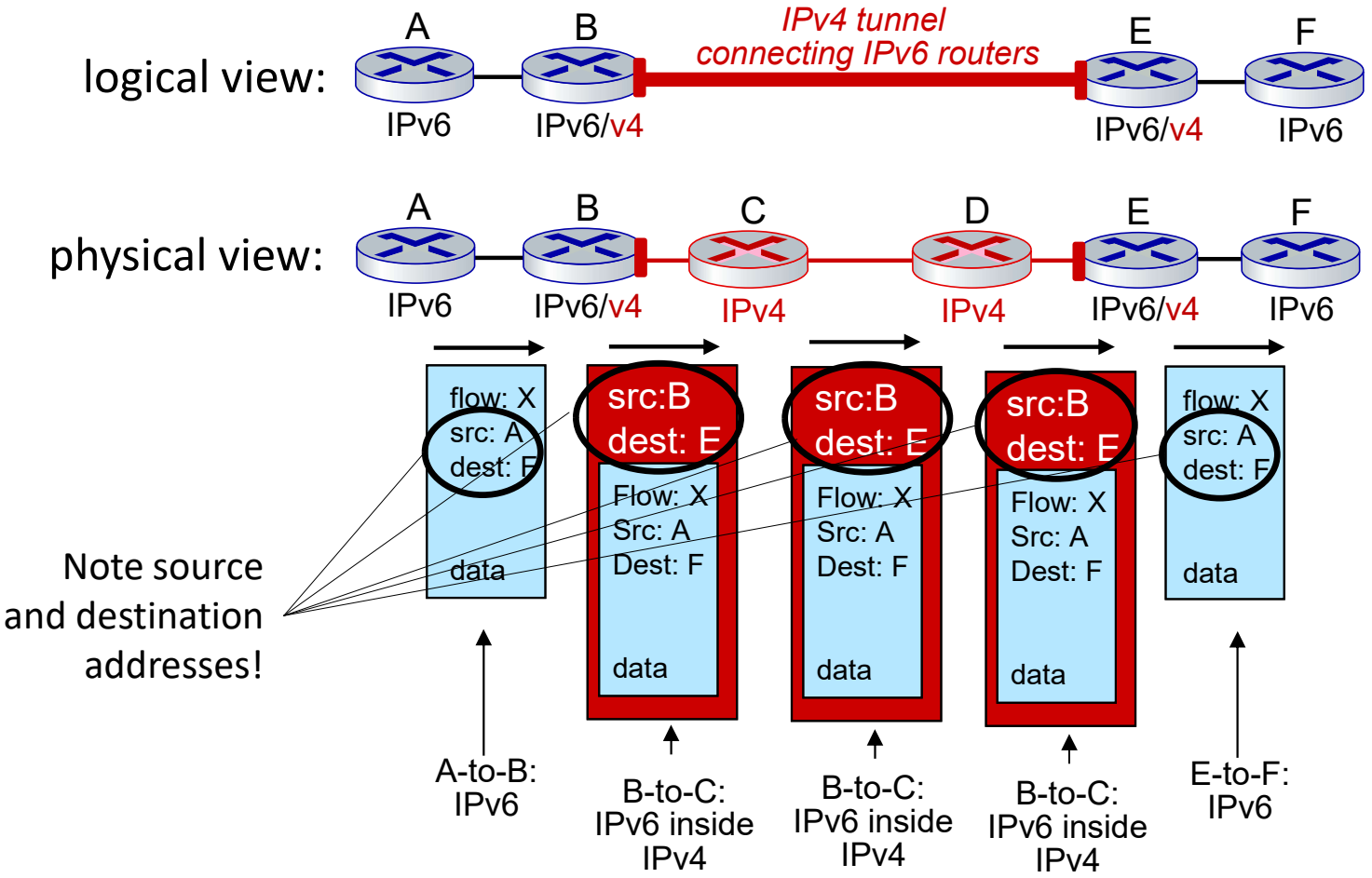


IPv4 tunnel connecting two IPv6 routers

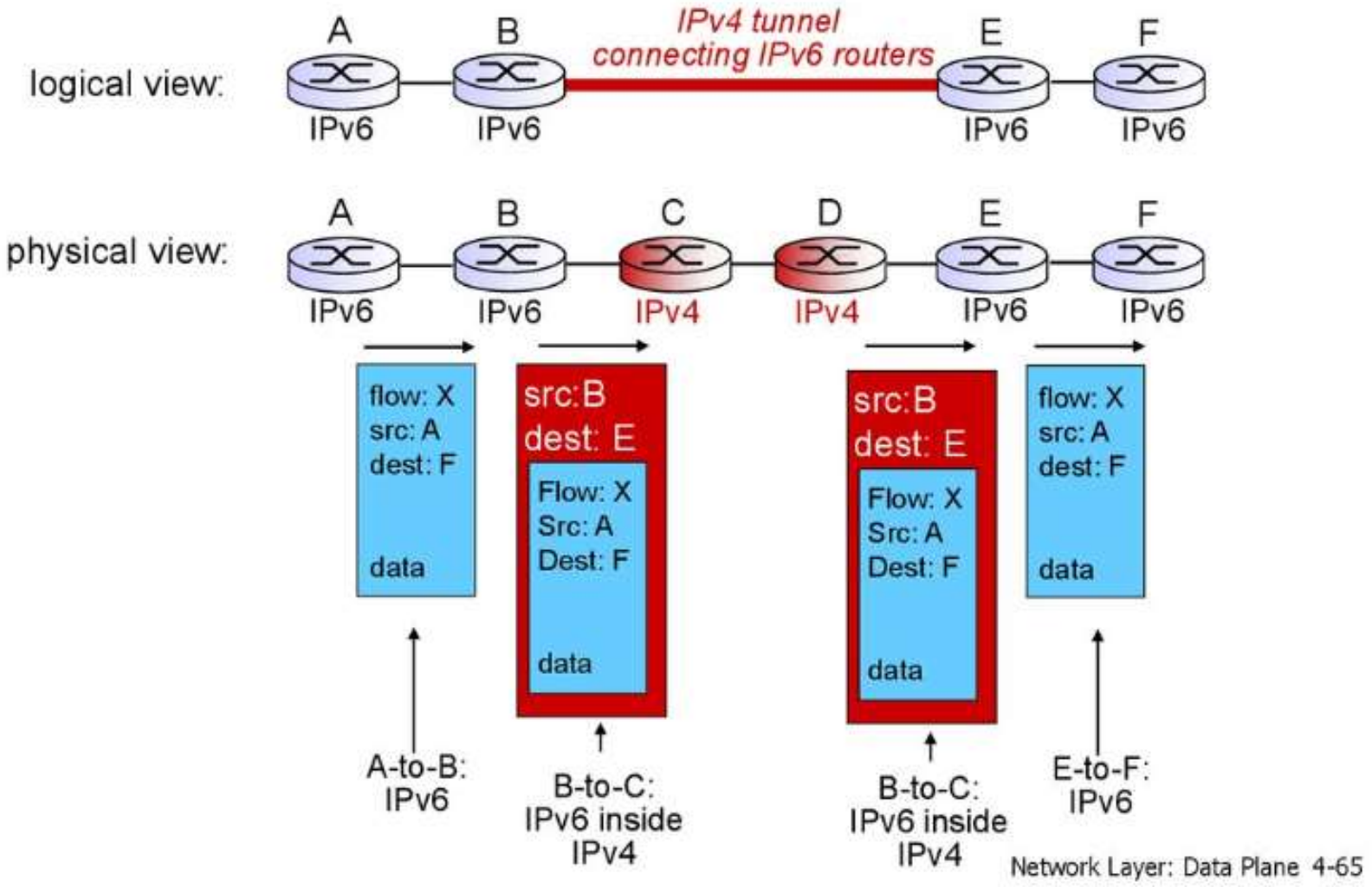


Datagrammi IPv6 come payload di datagrammi tra (interfacce di) router IPv4

Tunneling

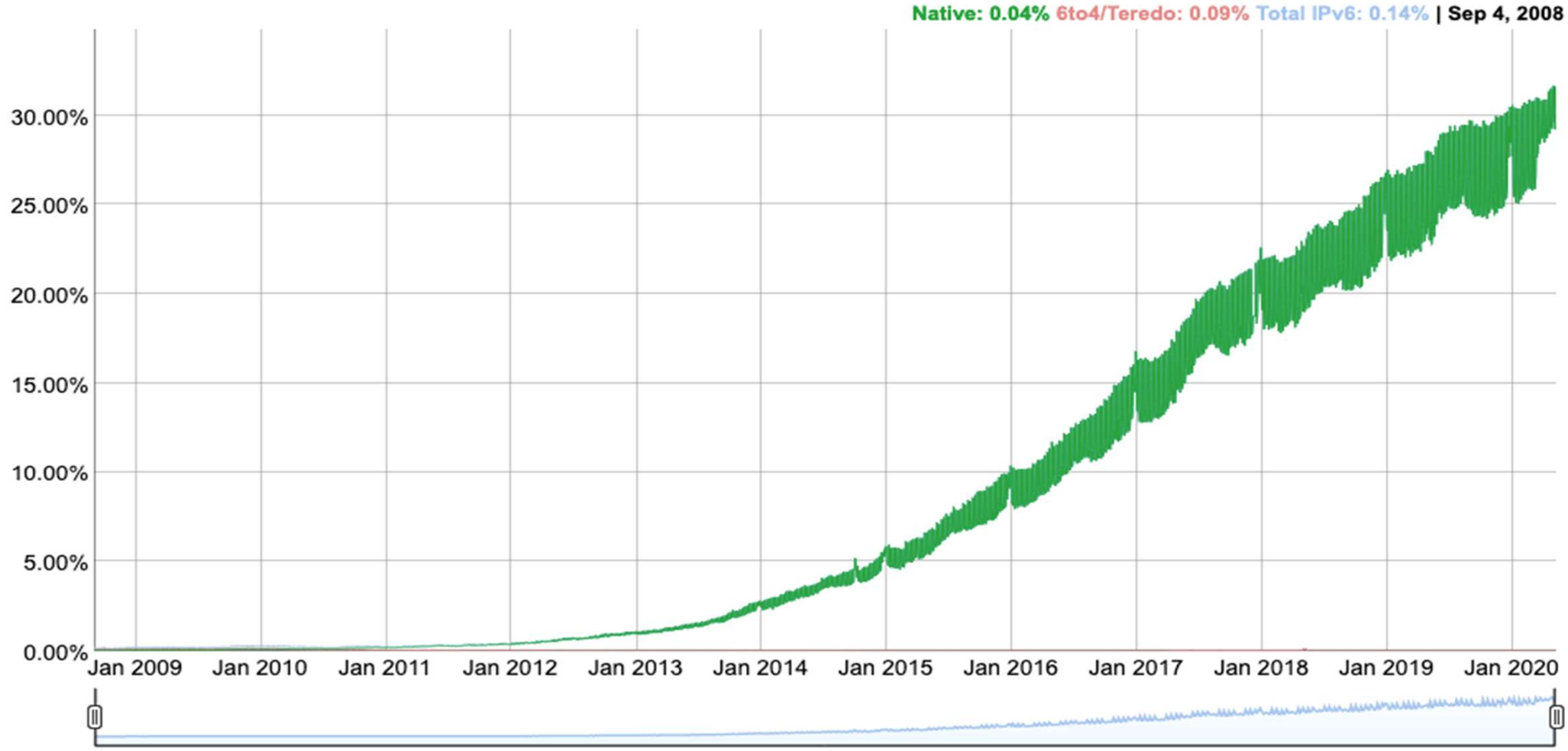

logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
Note source
and destination
addresses!
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.



[1]https://www.google.com/intl/en/ipv6/statistics.html

Riferimenti e immagini da:

- Computer Networks: A Top-Down Approach", B. A. Forouzan, F. Mosharraf, McGraw Hill.
- Jim Kurose and Keith Ross, Computer Networking: A Top-Down Approach, Settima Edizione, Pearson, 2017