

# Capitolo 4

## Livello di rete

### Nota per l'utilizzo:

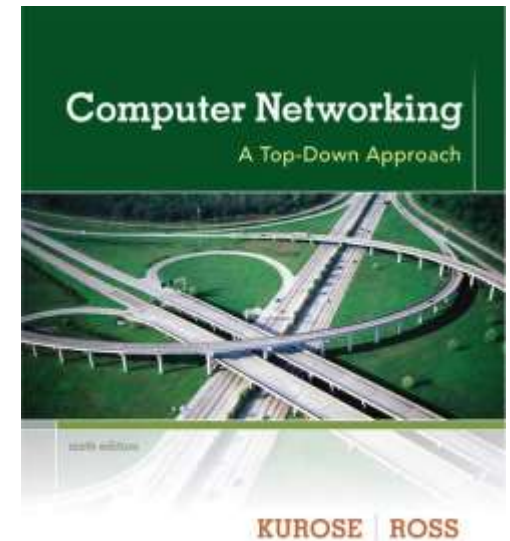
Abbiamo preparato queste slide con l'intenzione di renderle disponibili a tutti (professori, studenti, lettori). Sono in formato PowerPoint in modo che voi possiate aggiungere e cancellare slide (compresa questa) o modificarne il contenuto in base alle vostre esigenze.

Come potete facilmente immaginare, da parte nostra abbiamo fatto *un sacco* di lavoro. In cambio, vi chiediamo solo di rispettare le seguenti condizioni:

- ❑ se utilizzate queste slide (ad esempio, in aula) in una forma sostanzialmente inalterata, fate riferimento alla fonte (dopo tutto, ci piacerebbe che la gente usasse il nostro libro!)
- ❑ se rendete disponibili queste slide in una forma sostanzialmente inalterata su un sito web, indicate che si tratta di un adattamento (o che sono identiche) delle nostre slide, e inserite la nota relativa al copyright.

Thanks and enjoy! JFK/KWR

©All material copyright 1996-2012  
J.F Kurose and K.W. Ross, All Rights Reserved



**Computer  
Networking: A Top  
Down Approach**  
6<sup>th</sup> edition  
Jim Kurose, Keith Ross  
Addison-Wesley  
March 2012

# Capitolo 4: livello di rete

## *obiettivi del capitolo:*

- ❖ capire i principi che stanno dietro i servizi del livello di rete:
  - modelli di servizio del livello di rete
  - forwarding e routing
  - come lavora un router
  - routing (scelta del percorso)
  - broadcast, multicast
- ❖ implementazione in Internet

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

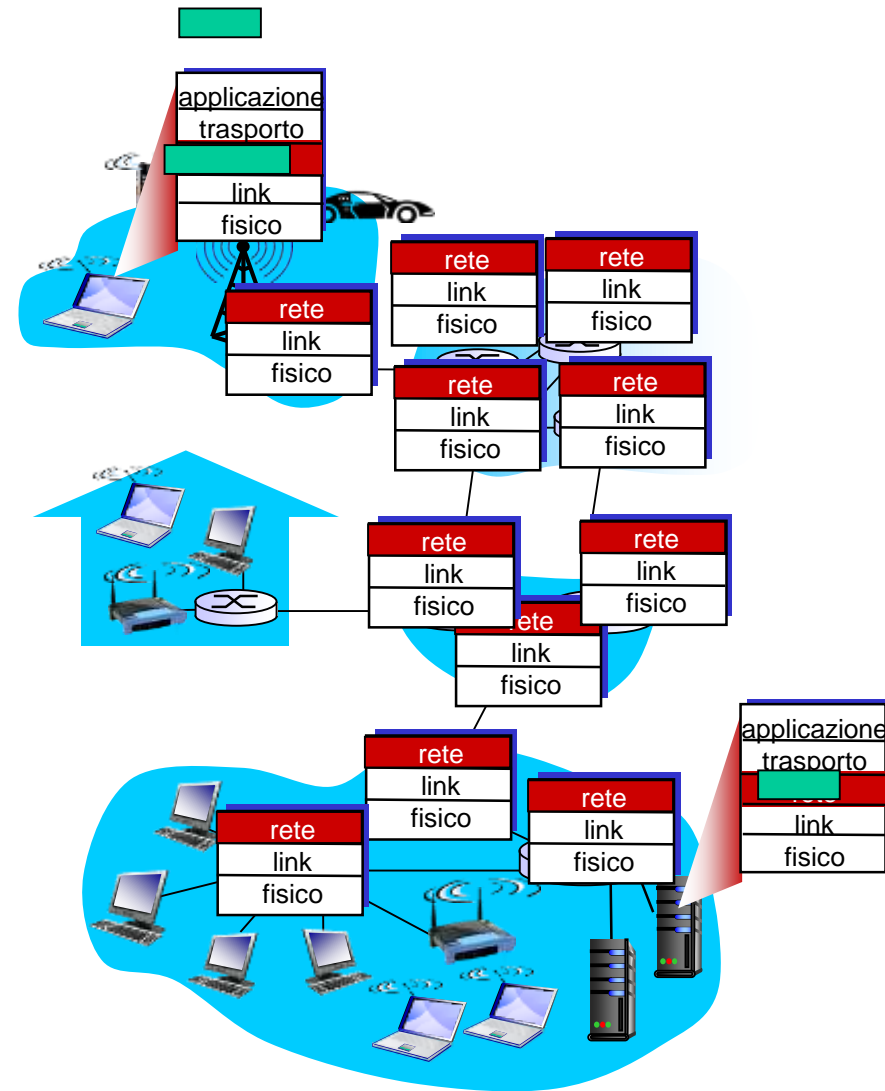
## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Livello di rete

- ❖ trasporta i segmenti dall'host mittente all'host ricevente
- ❖ nel lato mittente incapsula i segmenti in datagrammi
- ❖ nel lato ricevente, consegna i segmenti al livello di trasporto
- ❖ protocolli del livello di rete in *ogni* host e router
- ❖ il router esamina i campi intestazione in tutti i datagrammi IP che lo attraversano



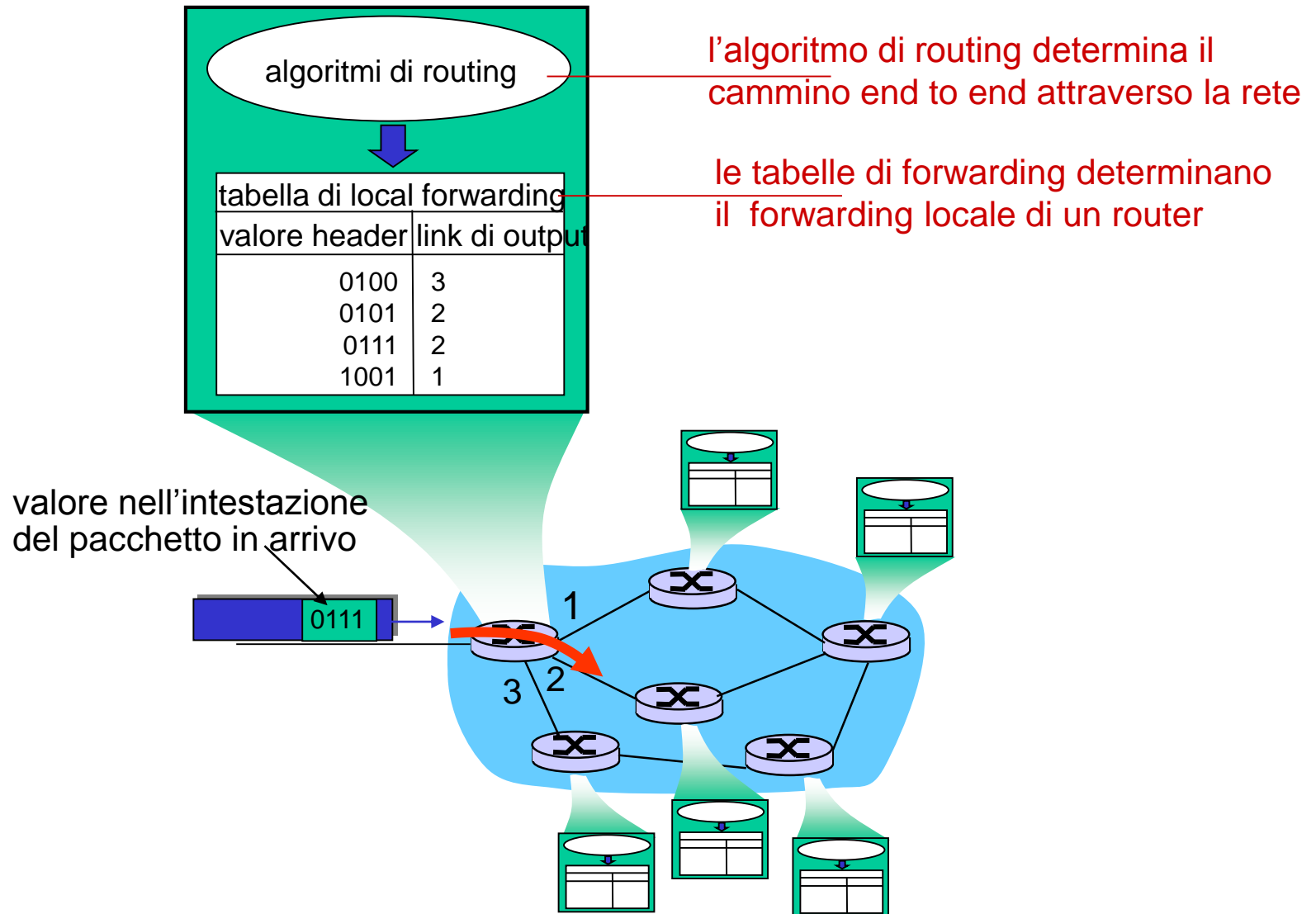
# Due funzioni chiave del livello di rete

- ❖ *forwarding*: muove i pacchetti in ingresso al router verso l'uscita appropriata del router
- ❖ *routing*: determina il percorso preso dai pacchetti dall'ingresso verso la destinazione.
  - *algoritmi di routing*

## *analogia:*

- ❖ *routing*: processo di pianificazione di un viaggio dalla sorgente alla destinazione
- ❖ *forwarding*: processo di attraversamento dei singoli bivi e incroci

# Relazioni tra routing e forwarding



# Impostazione della connessione

- ❖ terza funzione importante in *alcune* architetture a livello di rete:
  - ATM, frame relay, X.25
- ❖ prima che i datagrammi inizino a viaggiare, due end host e i router stabiliscono una connessione virtuale
  - i router vengono coinvolti
- ❖ raffronto tra i servizi di connessione a livello di rete e di trasporto:
  - *rete*: tra due host (può coinvolgere i router interessati nel caso di VC)
  - *trasporto*: tra due processi

# Modello di servizio del livello di rete

**D:** Qual è il **modello di servizio** per il “canale” che trasporta i datagrammi dal mittente al destinatario?

## *servizi per un singolo datagramma:*

- ❖ consegna garantita
- ❖ consegna garantita con un ritardo inferiore a 40 msec

## *servizi per un flusso di datagrammi:*

- ❖ consegna in ordine
- ❖ minima ampiezza di banda garantita
- ❖ restrizioni sul lasso di tempo tra la trasmissione di due pacchetti consecutivi



# Modelli di servizi del livello di rete

Architettura di rete	Modello di servizio	Garanzie?				Feedback sulla congestione
		Bandwidth	Perdite	Ordine	Timing	
Internet	best effort	nessuna	no	no	no	no (dedotta dalle perdite)
ATM	CBR	rate costante	sì	sì	sì	nessuna congestione
ATM	VBR	rate garantito	sì	sì	sì	nessuna congestione
ATM	ABR	minima garantita	no	sì	no	sì
ATM	UBR	nessuna	no	sì	no	no

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Servizi con e senza connessione

- ❖ le reti *a datagramma* offrono solo il servizio *senza connessione*
- ❖ le reti a *circuito virtuale* (VC) mettono a disposizione solo il servizio *con connessione*
- ❖ analogie con i servizi del livello di trasporto TCP/UDP connection-oriented / connectionless, ma:
  - *servizio*: da host a host
  - *non c'è scelta*: la rete offre un tipo o l'altro
  - *implementazione*: nel nucleo (core) della rete

# Virtual circuit (circuito virtuale)

“il percorso tra origine e destinazione si comporta in modo analogo a un circuito telefonico”

- performance tipo
- azioni della rete lungo il percorso source-to-dest

- ❖ avvio e chiusura per ogni chiamata *prima* che i dati comincino a fluire
- ❖ ogni pacchetto ha un identificatore di VC (non un indirizzo dell'host destinazione)
- ❖ ogni router lungo il cammino mantiene lo “stato” per ogni connessione che lo attraversa
- ❖ le risorse di link e router (bandwidth, buffer) possono essere *allocate* per il VC (risorse dedicate = servizio con prestazioni prevedibili)

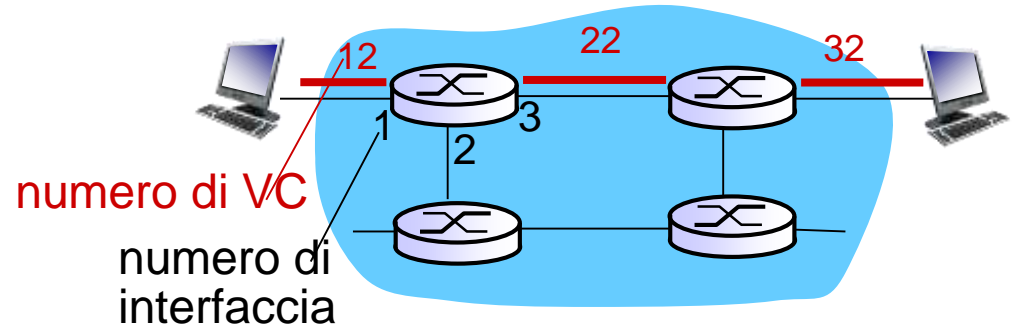
# Implementazione dei VC

*un VC consiste in:*

1. *un percorso* dalla sorgente alla destinazione
  2. *numeri di VC*, un numero per ogni link del percorso
  3. *righe nelle tabelle di forwarding* nei router del percorso
- ❖ un pacchetto di un VC ha un numero di VC (piuttosto che un indirizzo di destinazione)
  - ❖ il numero di VC di un pacchetto può cambiare su ogni link.
    - un nuovo numero di VC viene rilevato dalla tabella d'oltro

# VC forwarding table

*forwarding table:*

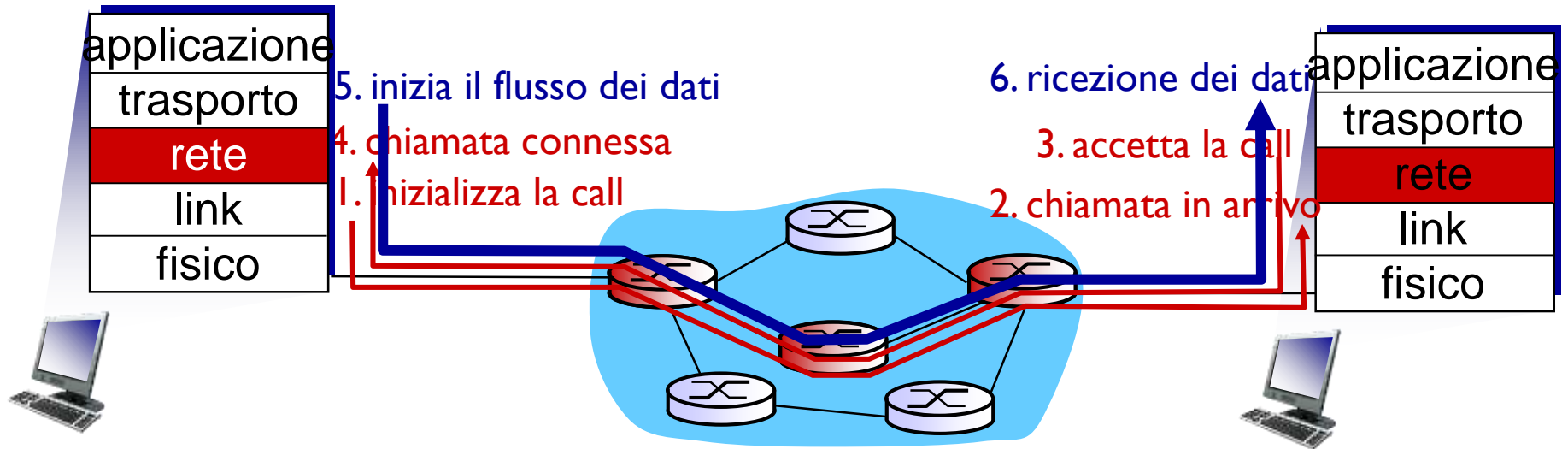


interfaccia di ingresso	# VC di ingresso	interfaccia di uscita	# VC di uscita
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

*I router dei VC mantengono le informazioni sullo stato delle connessioni!*

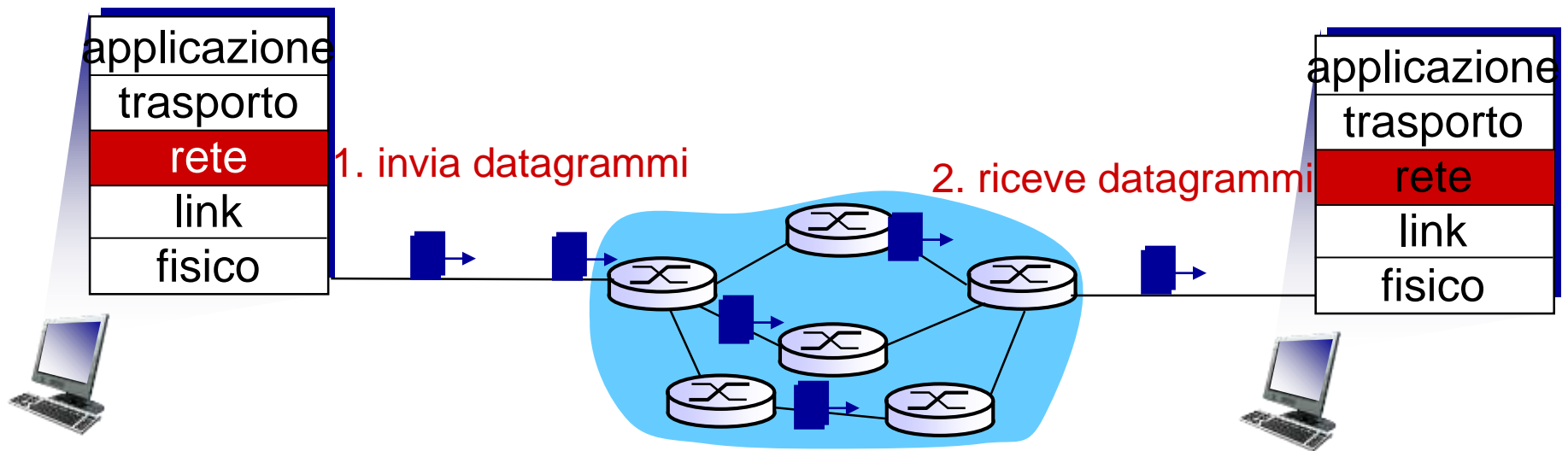
# Virtual circuit: protocolli di segnalazione

- ❖ messaggi inviati dai sistemi terminali per instaurare o smantellare un circuito virtuale
- ❖ usati in ATM, frame-relay, X.25
- ❖ non usati attualmente in Internet



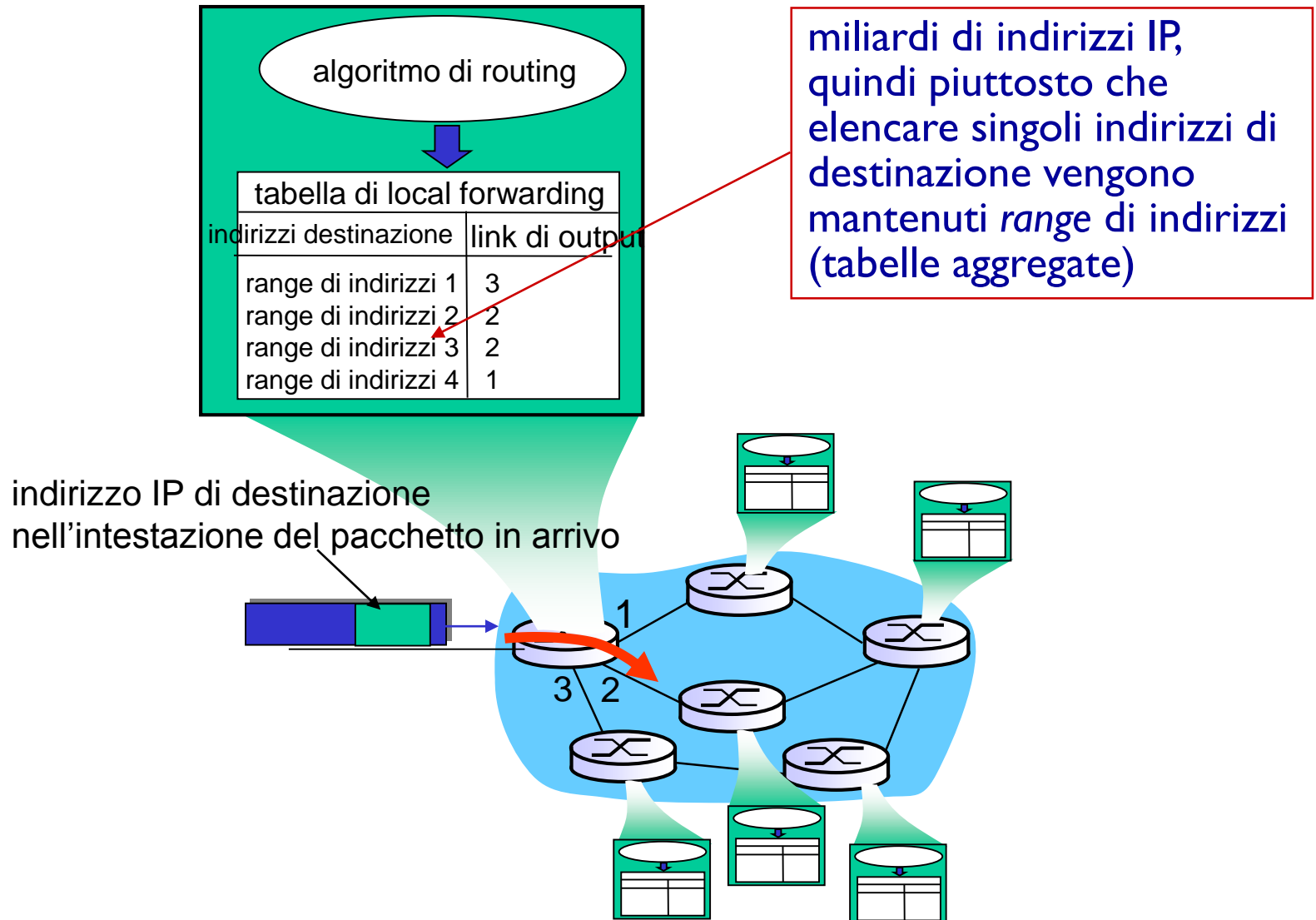
# Reti a datagrammi

- ❖ nessun setup a livello di rete
- ❖ router: nessuno stato riguardo le connessioni end-to-end
  - a livello di rete manca proprio il concetto di “connessione”
- ❖ i pacchetti sono inoltrati usando l'indirizzo dell'host destinazione





# Forwarding table



# Datagram forwarding table

Range di indirizzi di destinazione	Interfaccia del Link
da 11001000 00010111 00010000 00000000 fino a 11001000 00010111 00010111 11111111	0
da 11001000 00010111 00011000 00000000 fino a 11001000 00010111 00011000 11111111	1
da 11001000 00010111 00011001 00000000 fino a 11001000 00010111 00011111 11111111	2
altrimenti	3

**D:** che succede se i range non sono così ben divisi?

# Longest prefix matching

## *longest prefix matching*

quando si consulta la forwarding table per una destinazione, si usa *il più lungo* prefisso dell'indirizzo che coincide con l'indirizzo di destinazione.

Range di indirizzi di destinazione	Interfaccia del Link
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
altrimenti	3

esempi:

Dest: 11001000 00010111 00010110 10100001

quale interfaccia?

Dest: 11001000 00010111 00011000 10101010

quale interfaccia?

# Reti a datagrammi o VC?

## *Internet (datagrammi)*

- ❖ scambio di dati tra differenti calcolatori
  - Servizi “ elastici ”, non vi sono eccessivi requisiti di tempo
- ❖ diversi tipi di link
  - differenti caratteristiche
  - difficile uniformarne il servizio
- ❖ end system “intelligenti” (computer)
  - possono adattarsi, effettuare controlli, rimediare a errori
  - ***semplicità dentro la rete, complessità ai “confini”***

## *ATM (VC)*

- ❖ derivano dal mondo della telefonia
- ❖ conversazione telefonica:
  - requisiti stringenti di tempo e affidabilità
  - necessità di servizi garantiti
- ❖ end system “stupidi”
  - telefoni
  - ***complessità dentro la rete***

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

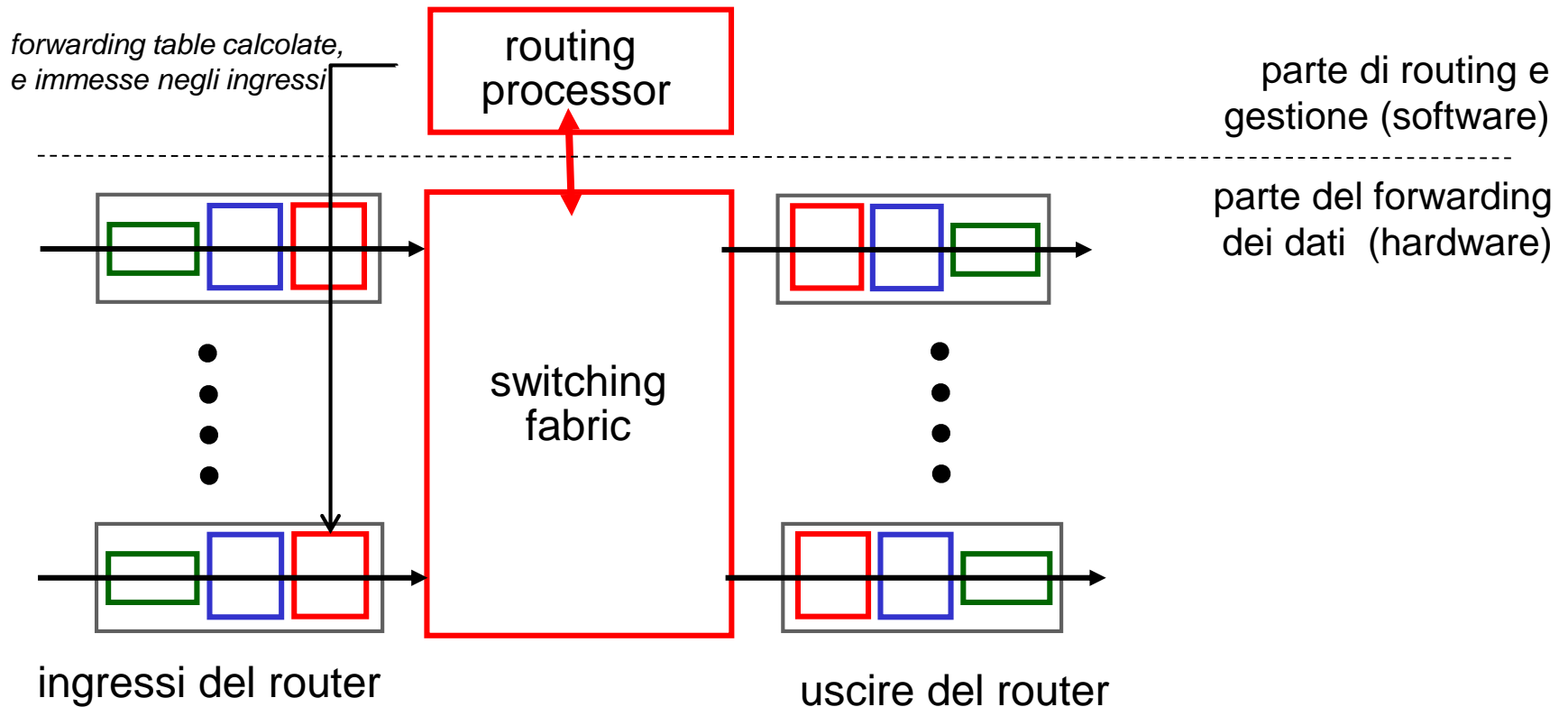
- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

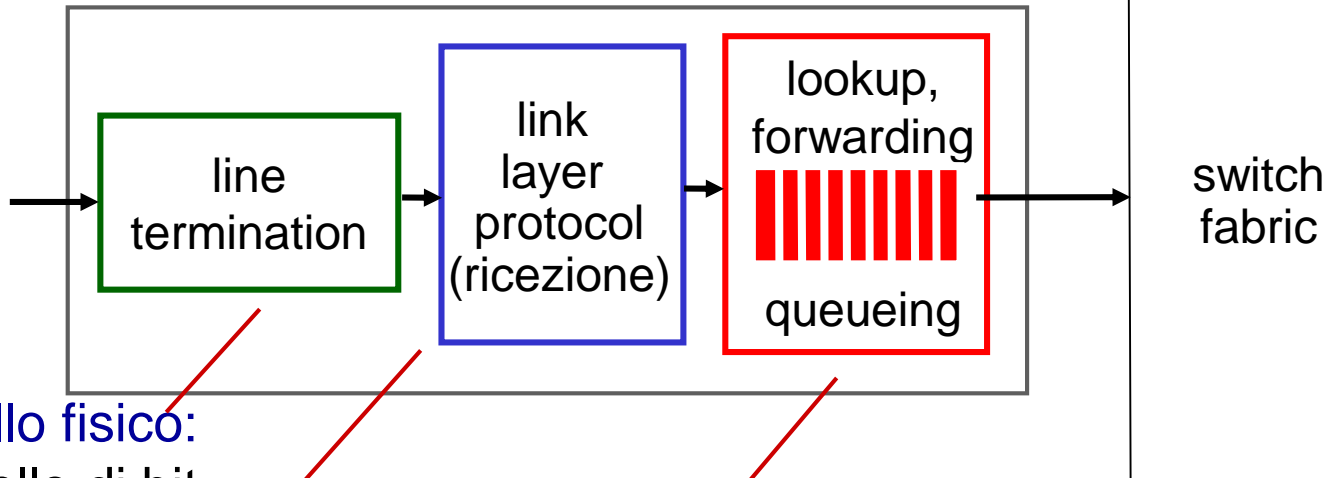
# Architettura dei router

due funzioni chiave dei:

- ❖ eseguire algoritmi e protocolli di routing (RIP, OSPF, BGP)
- ❖ *inoltro* dei datagrammi



# Funzioni nelle interfacce di ingresso



livello fisico:

ricezione a livello di bit

link layer:

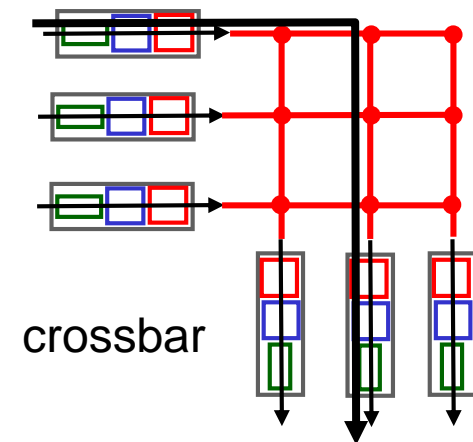
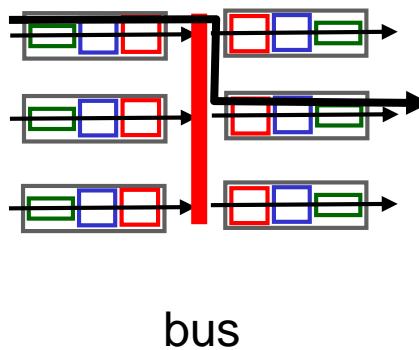
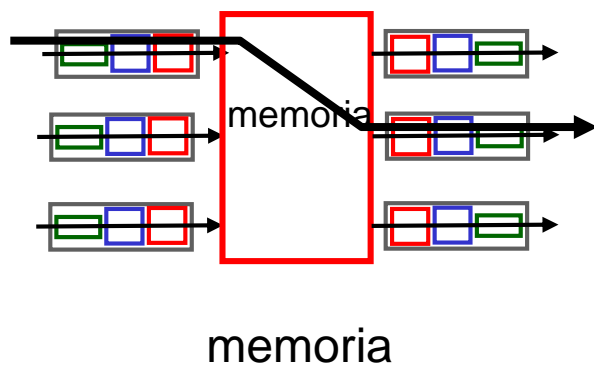
es., Ethernet  
vedi capitolo 5

commutazione decentralizzata:

- ❖ determina la porta d'uscita dei pacchetti utilizzando le informazioni della tabella d'inoltro
- ❖ obiettivo: completare l'elaborazione alla stessa 'velocità della linea'
- ❖ accodamento: se il rate di arrivo dei datagrammi è superiore a quello di inoltro

# Switching fabric

- ❖ trasferisce i pacchetti dal buffer di input al buffer di output appropriato
- ❖ switching rate: rate con il quale i pacchetti possono essere trasferiti dall'input all'output
  - spesso misurato come multiplo del rate di input/output
  - N input: switching rate N volte il line rate
- ❖ tre tipi di strutture di switching

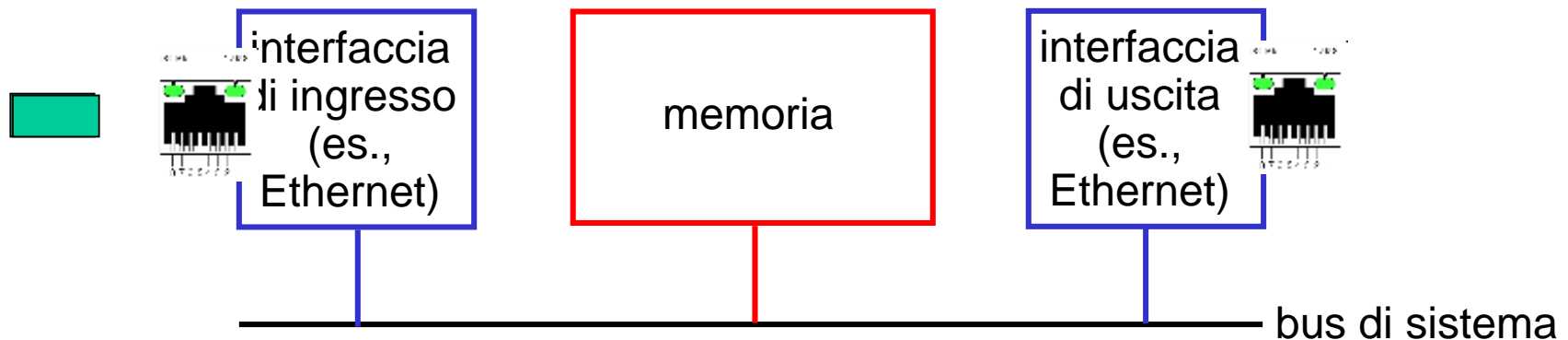




# Switching tramite memoria

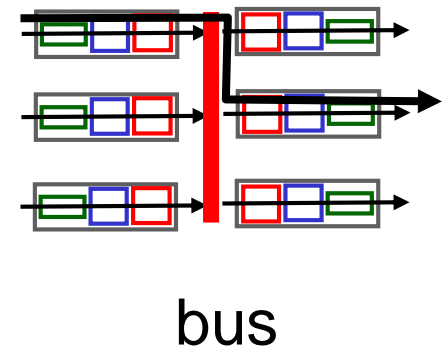
## *router di prima generazione:*

- ❖ computer tradizionali con lo switching sotto il controllo diretto della CPU
- ❖ pacchetto copiato nella memoria del sistema
- ❖ velocità limitata dalla velocità della memoria (2 attraversamenti del bus per datagramma)



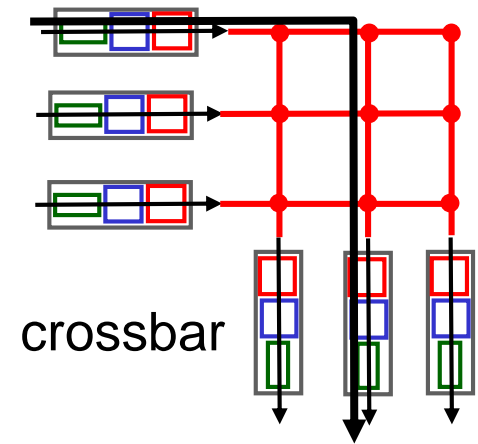
# Switching tramite bus

- ❖ i datagrammi vengono trasferiti dall'ingresso all'uscita attraverso un bus condiviso
- ❖ *bus*: la velocità di switching è limitata dalla bandwidth del bus
- ❖ 32 Gbps bus, Cisco 5600: velocità sufficiente speed per reti d'accesso o aziendali

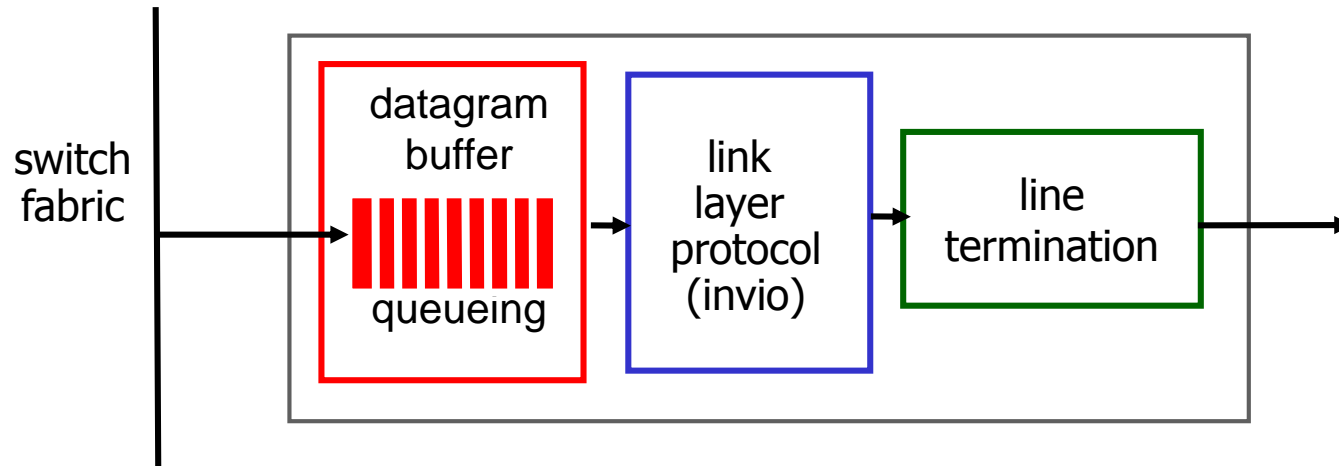


# Switching tramite rete d'interconnessione

- ❖ supera il limite di banda di un singolo bus condiviso
- ❖ crossbar e altre reti d'interconnessione inizialmente sviluppate per connettere processori in multiprocessori
- ❖ design avanzati: i datagrammi sono divisi in celle di lunghezza fissa, e le celle commutate lungo la struttura.
- ❖ Cisco I2000: commuta 60 Gbps attraverso la rete d'interconnessione

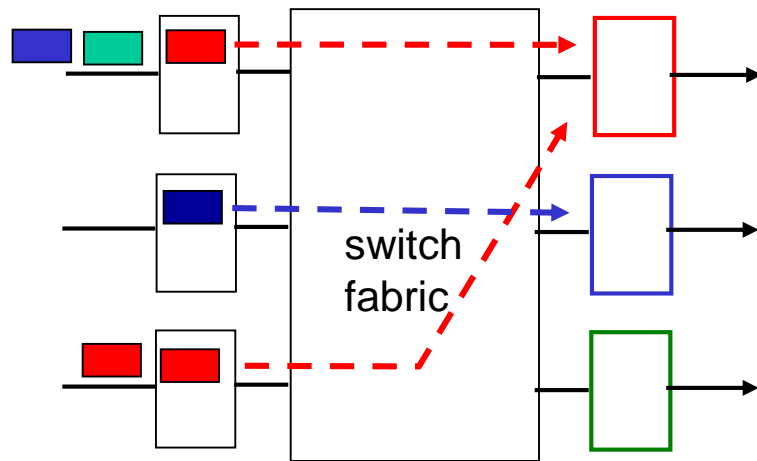


# Accodamento in uscita

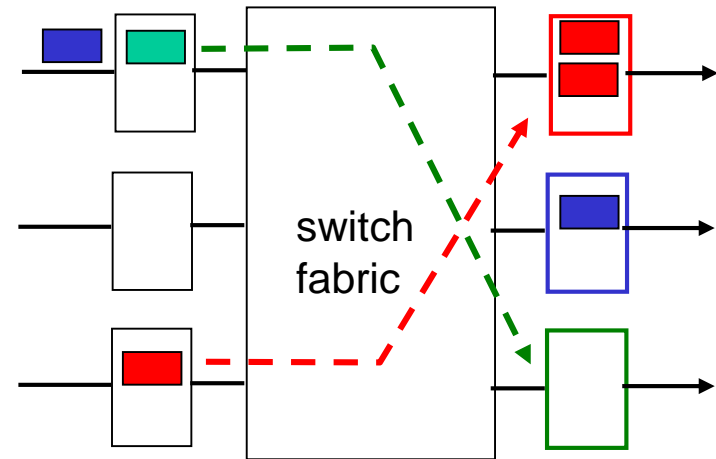


- ❖ *buffering* necessari  
più veloce del rate
  - ❖ *criterio di scheduling*  
accodati qu
- I datagrammi (pacchetti) possono essere persi a causa di congestion, o limite del buffer
- Scheduling – priorità a chi ha le migliori performance (rete 'neutrale')

# Accodamento in uscita



situazione al tempo  $t$



un tempo pacchetto dopo

- ❖ c'è buffering quando il rate di arrivo eccede la velocità di trasmissione in uscita
- ❖ *accodamenti (delay) e perdite dovute a overflow dei buffer di uscita!*

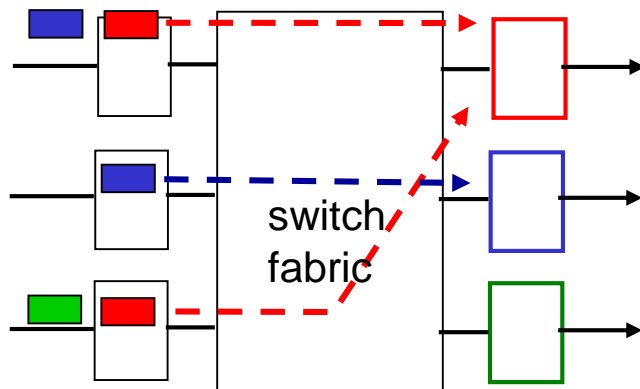
# Quanto buffering?

- ❖ RFC 3439: il buffering medio deve essere uguale al “tipico” RTT (es. 250 msec) per la capacità  $C$  del link
  - es.,  $C = 10$  Gpbs link: 2.5 Gbit buffer
- ❖ recenti direttive: con  $N$  flussi, il buffering deve essere uguale a

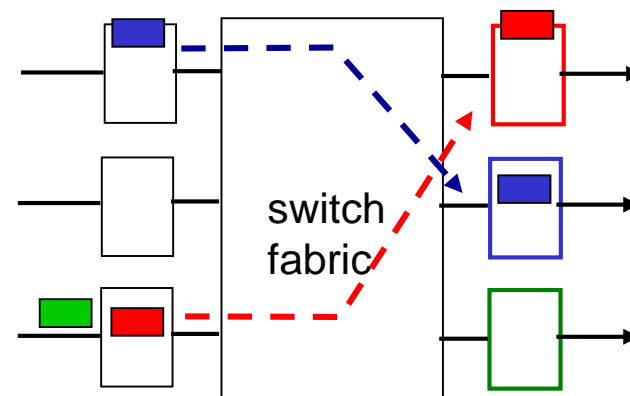
$$\frac{RTT \cdot C}{\sqrt{N}}$$

# Accodamento in ingresso

- ❖ struttura più lenta del rate di ingresso -> può esserci accodamento in ingresso
  - *ritardi di accodamento e perdite per overflow del buffer di ingresso!*
- ❖ **Head-of-the-Line (HOL) blocking:** datagrammi accodati in testa alla coda impediscono ad altri pacchetti di avanzare



al tempo  $t$ :  
solo un datagramma rosso può  
essere trasferito.  
*il pacchetto rosso in basso è bloccato*



un tempo pacchetto dopo:  
il pacchetto verde è vittima  
del HOL blocking (la sua  
destinazione era libera ma  
ha dovuto attendere)

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

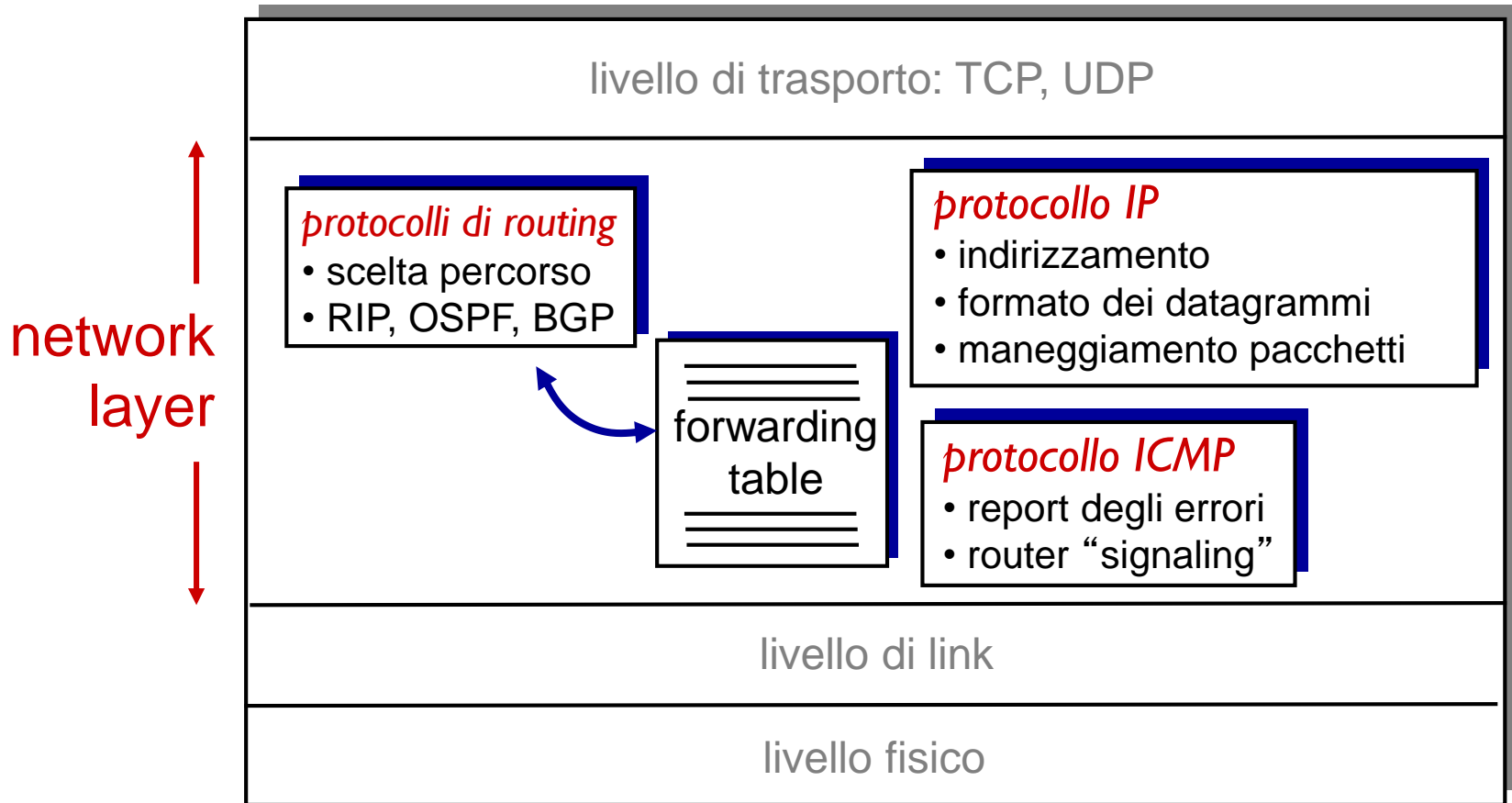
- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

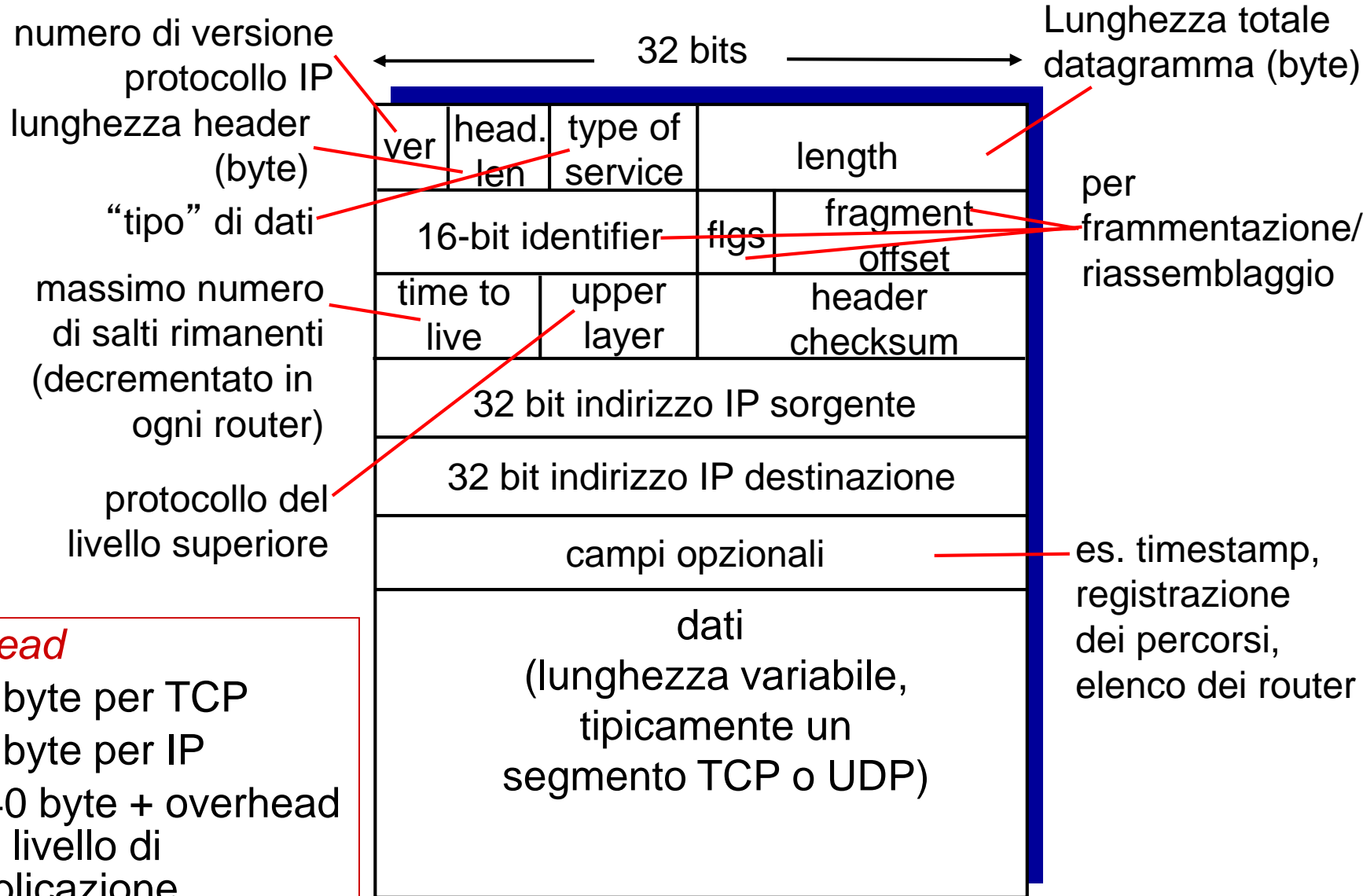


# Livello di rete in Internet

funzioni del livello di rete di host e router:



# Formato dei datagrammi IP



## overhead

- ❖ 20 byte per TCP
- ❖ 20 byte per IP
- ❖ = 40 byte + overhead del livello di applicazione

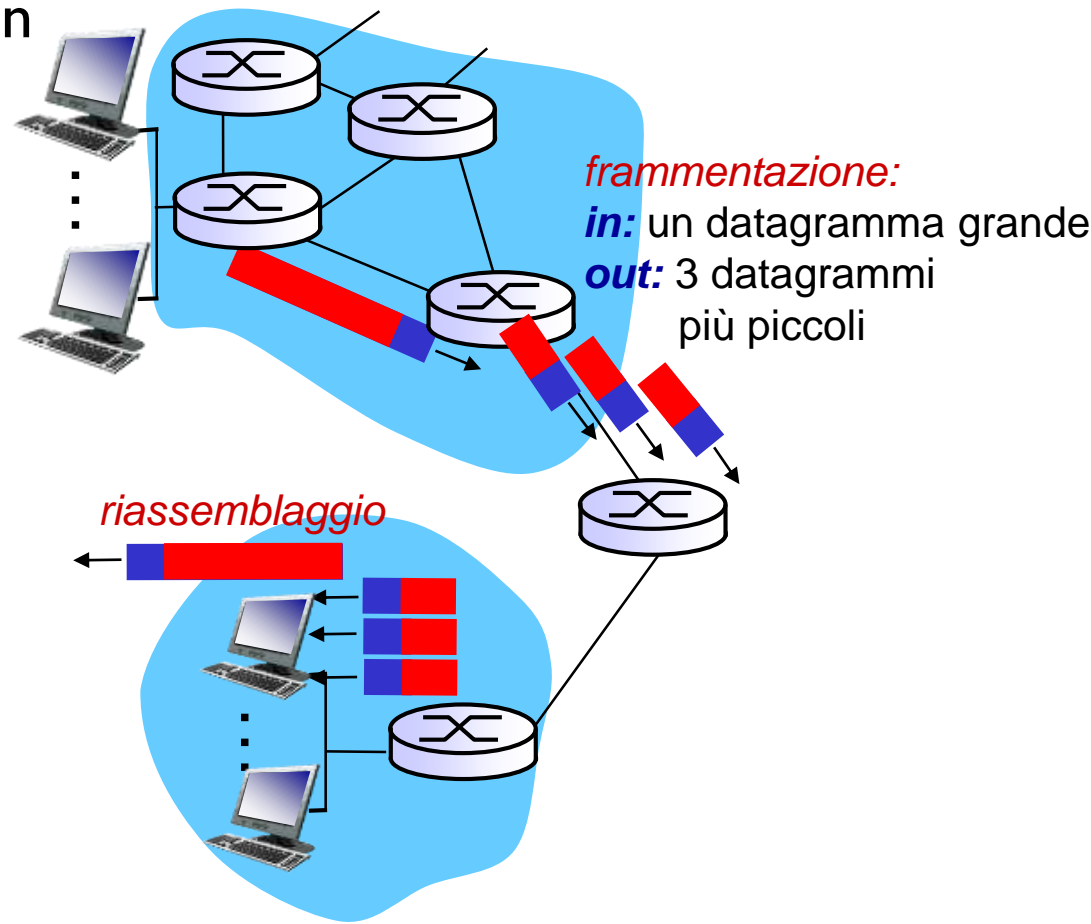
# Frammentazione dei datagrammi IP

❖ i link della rete hanno una MTU (maximum transmission unit) – dimensione massima di un frame

- differenti tipi di link, differenti MTU.

❖ datagrammi IP eccedenti la MTU vengono divisi (“frammentati”)

- un datagramma viene frammentato
- i frammenti saranno riassemblati nell’host di destinazione
- i bit dell’header IP verranno usati per il riordinamento



# Frammentazione dei datagrammi IP

*esempio:*

- ❖ datagramma di 4000 byte
- ❖ MTU = 1500 byte

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

*un datagramma IP grande viene frammentato in datagrammi IP più piccoli*

1480 byte nel  
campo dati

offset =  
1480/8

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=185	

	length	ID	fragflag	offset	
	=1040	=x	=0	=370	

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

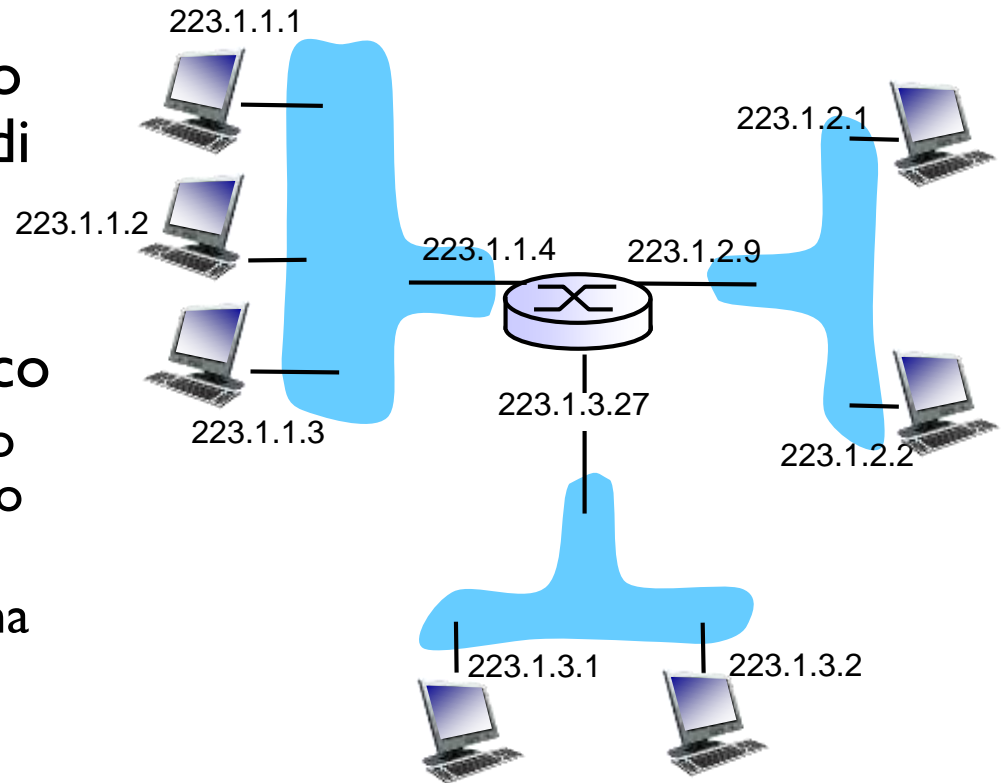
## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Indirizzamento IP

- ❖ **indirizzo IP**: identificativo a 32-bit per le *interfacce* di host e router
- ❖ **interfaccia**: connessione tra host/router e link fisico
  - i router tipicamente hanno interfacce multiple (almeno due!)
  - un host tipicamente ha una o due interfacce (es., Ethernet, wireless 802.11)
- ❖ **ogni interfaccia ha un indirizzo IP associato**



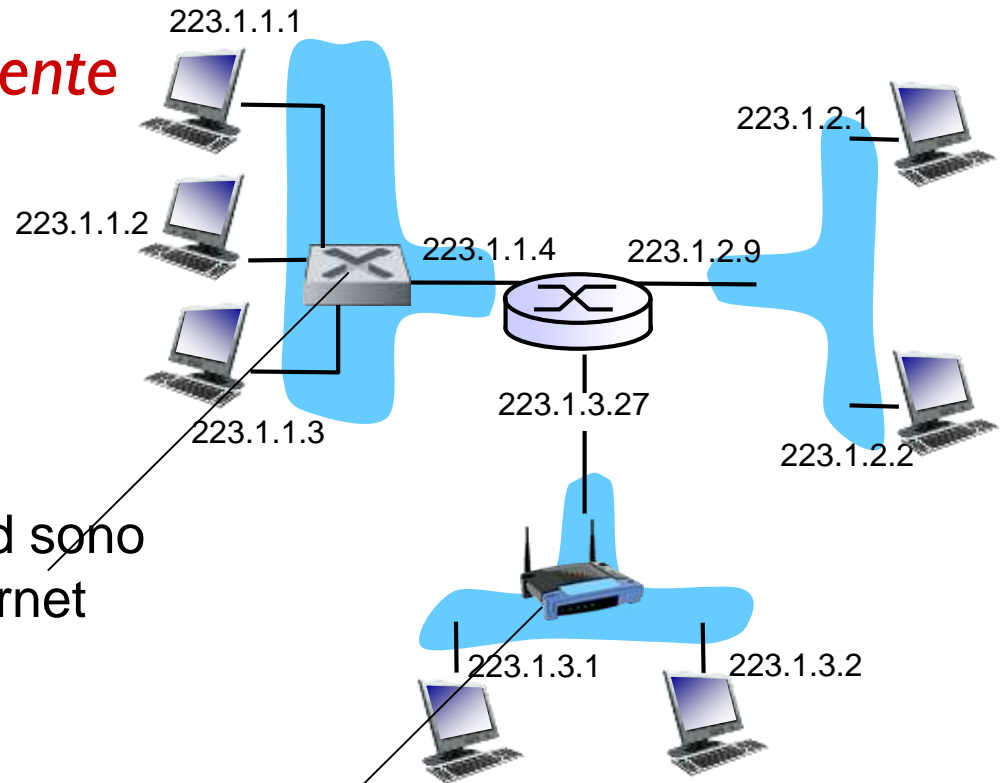
$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

# Indirizzamento IP

**D:** come sono effettivamente connesse le interfacce?

**R:** capitoli 5, 6.

**R:** le interfacce Ethernet wired sono connesse tramite switch Ethernet



**A:** le interfacce wireless WiFi sono connesse tramite base station WiFi

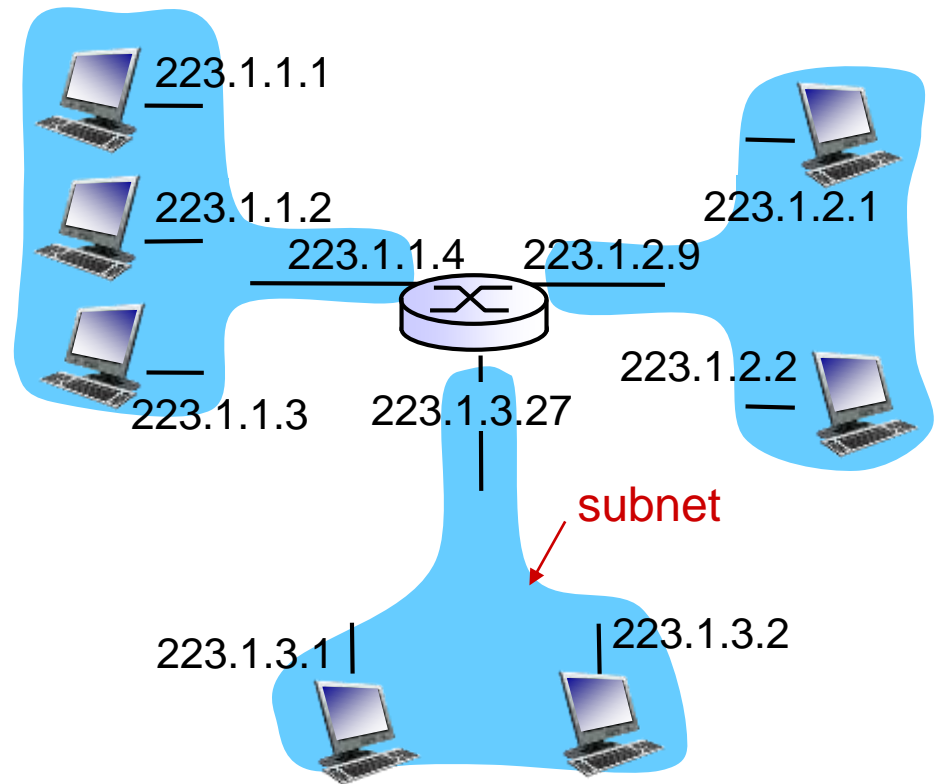
# Sottoreti (subnet)

## ❖ indirizzo IP:

- parte della sottorete - bit più significativi
- parte host - bit meno significativi

## ❖ *cos'è una sottorete?*

- interfacce di dispositivi con la stessa parte di sottorete nell'indirizzo IP
- possono raggiungersi tra di loro *senza l'intervento di un router*

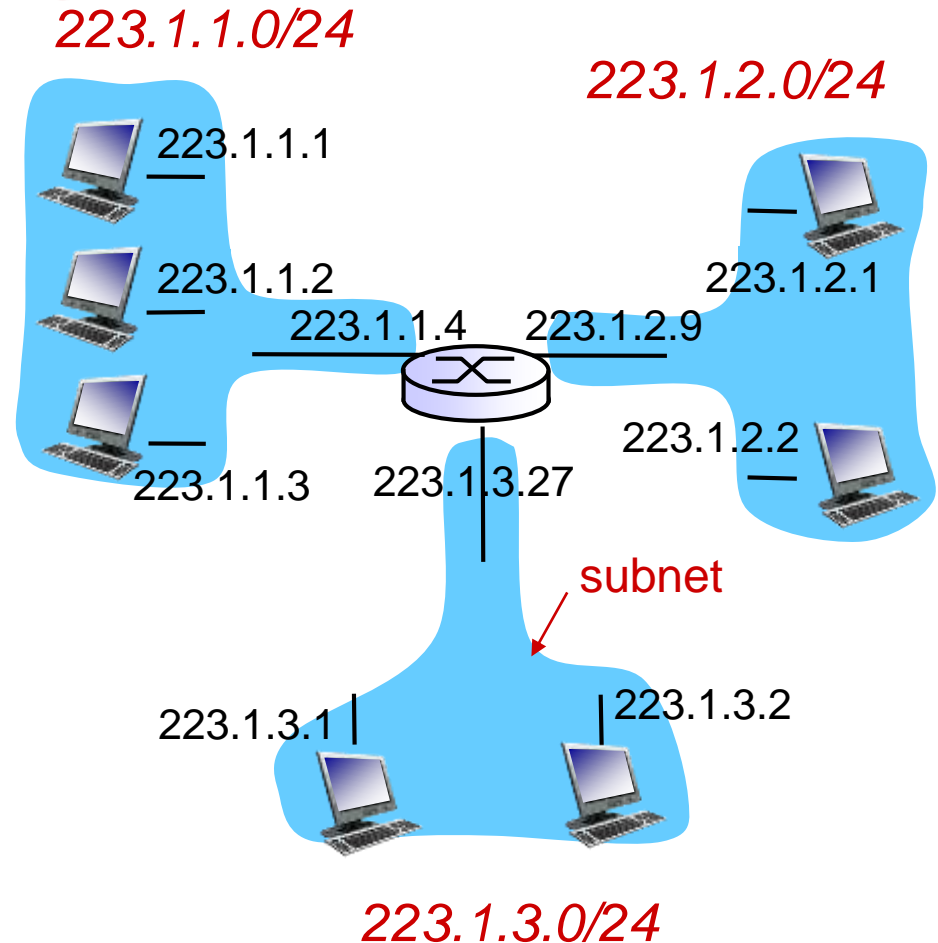


rete composta da 3 sottoreti



# Sottoreti (subnet)

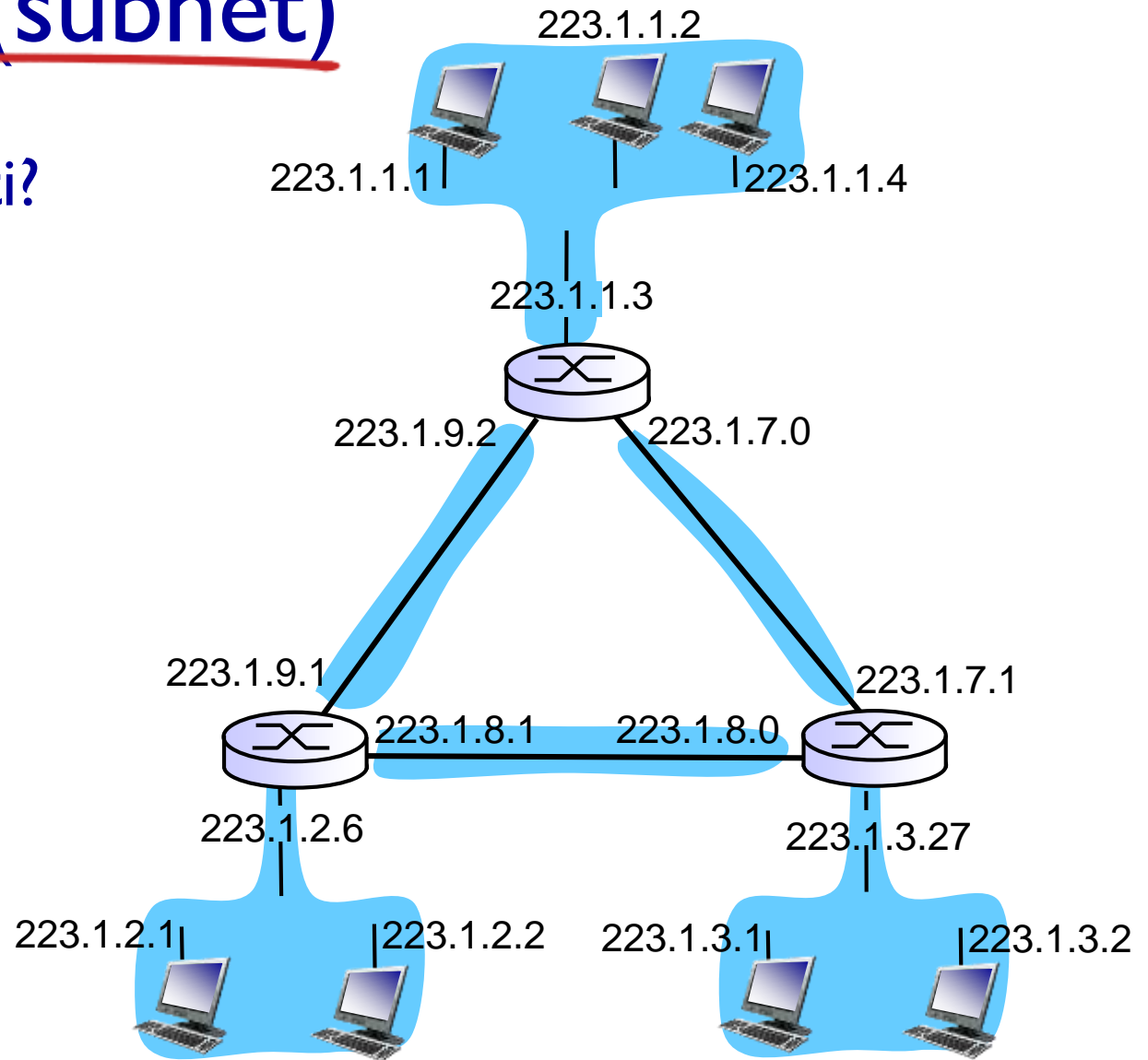
- ❖ per determinare le sottoreti, separate ogni interfaccia dal suo host o router, ottenendo blocchi di reti isolate
- ❖ ogni rete isolata è chiamata **sottorete** (*subnet*)



maschera di sottorete (subnet mask): /24

# Sottoreti (subnet)

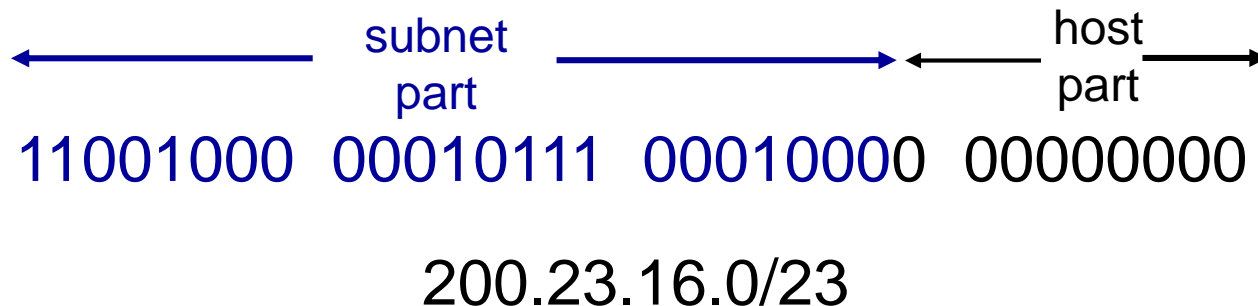
quante sottoreti?



# Indirizzamento IP: CIDR

## CIDR: Classless InterDomain Routing

- porzione di indirizzo di una sottorete di lunghezza arbitraria
- formato dell'indirizzo: **a.b.c.d/x**, dove x è il numero di bit della porzione di indirizzo della sottorete



# Indirizzi IP: come ottenerli?

**D:** Come acquisisce l'indirizzo IP un *host*?

- ❖ impostato dall'amministratore del sistema in un file
  - Windows: control-panel-> network-> configuration-> tcp/ip-> properties
  - UNIX: /etc/rc.config
- ❖ **DHCP: Dynamic Host Configuration Protocol:** ottiene dinamicamente l'indirizzo da un server
  - “plug-and-play”

# DHCP: Dynamic Host Configuration Protocol

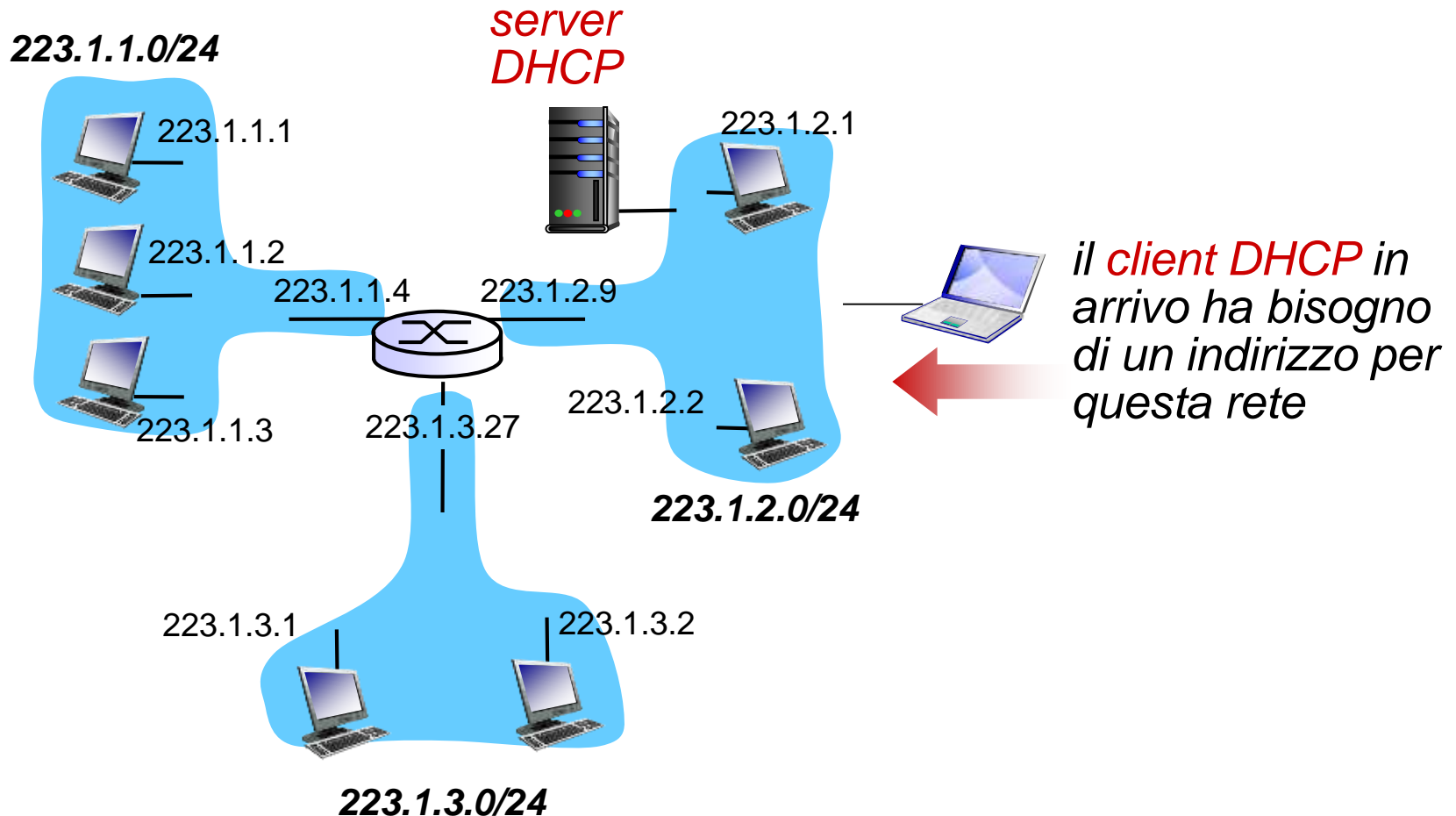
**obiettivo:** consentire a un host di ottenere *dinamicamente* un indirizzo IP da un server quando si collega a una rete

- assegna un indirizzo a ogni host (lease)
- può rinnovare la sua assegnazione dando l'indirizzo precedente
- permette il riuso degli indirizzi (mantiene gli indirizzi solo degli elementi connessi)
- supporto per utenti mobili che vogliono unirsi alla rete (rapidamente)

**principio del DHCP:**

- l'host manda un messaggio “**DHCP discover**” [opzionale]
- il server DHCP risponde con un “**DHCP offer**” [opzionale]
- l'host richiede un indirizzo IP: messaggio di “**DHCP request**”
- il server DHCP invia l'indirizzo: messaggio di “**DHCP ack**”

# Scenario DHCP client-server



# Scenario DHCP client-server

server DHCP : 223.1.2.5

DHCP discover

cliente in  
arrivo



Broadcast: c'è un server  
DHCP ?

offerta DHCP

Broadcast: sono un server  
DHCP! Ecco un indirizzo  
IP che puoi usare  
192.168.1.100

richiesta DHCP

Broadcast: OK. Prenderò  
quell'indirizzo IP!

DHCP ACK

Broadcast: OK.  
Quell'indirizzo IP è tuo!

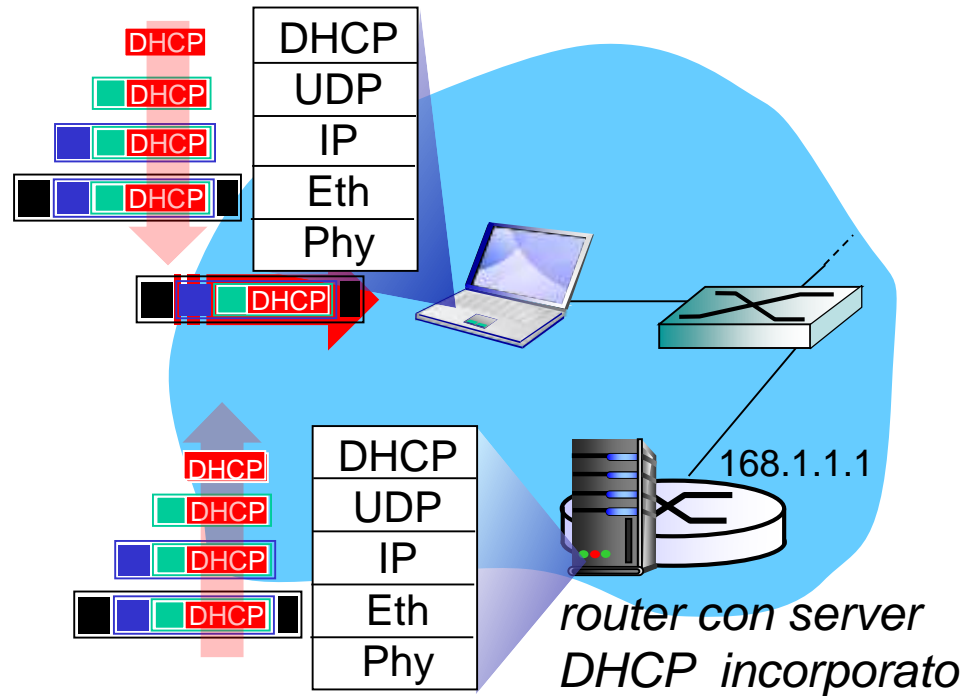
# DHCP: non solo indirizzi IP

il DHCP può restituire molto di più che non il semplice indirizzo IP per la sottorete:

- l'indirizzo per il router di uscita (gateway)
- nome e indirizzo IP del server DNS
- network mask (indicante parte rete e parte host dell'indirizzo)

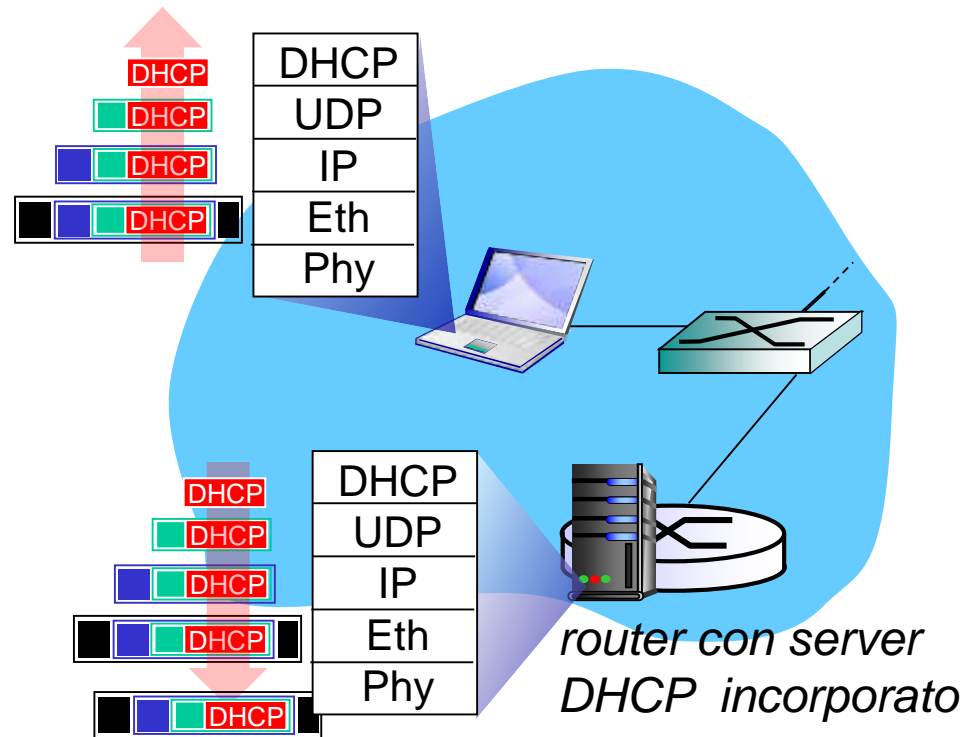


# DHCP: esempio



- ❖ il portatile ha bisogno di indirizzo IP, indirizzo del router di first-hop (gateway), indirizzo del DNS: usa DHCP
- ❖ a richiesta DHCP viene incapsulata in UDP, incapsulata in IP, incapsulata in 802.1 Ethernet
- ❖ il frame Ethernet va in broadcast (dest: FFFFFFFFFFFFFFFF) sulla LAN, e viene ricevuto dal router con il server DHCP
- ❖ processo inverso all'incapsulamento (demuxing) fino al DHCP

# DHCP: esempio



- ❖ il server DHCP formula il DHCP ACK contenente indirizzo, gateway e DNS
- ❖ incapsulamento del server DHCP, frame inoltrato al client, demuxing fino al client
- ❖ il client ora ha indirizzo IP, gateway e DNS

# DHCP: output di Wireshark

Message type: **Boot Request (1)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

**Transaction ID: 0x6b3a11b7**

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

**Client MAC address: Wistron\_23:68:8a (00:16:d3:23:68:8a)**

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) **DHCP Message Type = DHCP Request**

Option: (61) Client identifier

Length: 7; Value: 010016D323688A;

Hardware type: Ethernet

Client MAC address: Wistron\_23:68:8a (00:16:d3:23:68:8a)

Option: (t=50,l=4) Requested IP Address = 192.168.1.101

Option: (t=12,l=5) Host Name = "nomad"

**Option: (55) Parameter Request List**

Length: 11; Value: 010F03062C2E2F1F21F92B

**1 = Subnet Mask; 15 = Domain Name**

**3 = Router; 6 = Domain Name Server**

44 = NetBIOS over TCP/IP Name Server

.....

richiesta

Message type: **Boot Reply (2)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

**Transaction ID: 0x6b3a11b7**

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

**Client IP address: 192.168.1.101 (192.168.1.101)**

Your (client) IP address: 0.0.0.0 (0.0.0.0)

**Next server IP address: 192.168.1.1 (192.168.1.1)**

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron\_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

**Option: (t=53,l=1) DHCP Message Type = DHCP ACK**

**Option: (t=54,l=4) Server Identifier = 192.168.1.1**

**Option: (t=1,l=4) Subnet Mask = 255.255.255.0**

**Option: (t=3,l=4) Router = 192.168.1.1**

**Option: (6) Domain Name Server**

Length: 12; Value: 445747E2445749F244574092;

IP Address: 68.87.71.226;

IP Address: 68.87.73.242;

IP Address: 68.87.64.146

**Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."**

risposta

# Come ottenere un blocco di indirizzi?

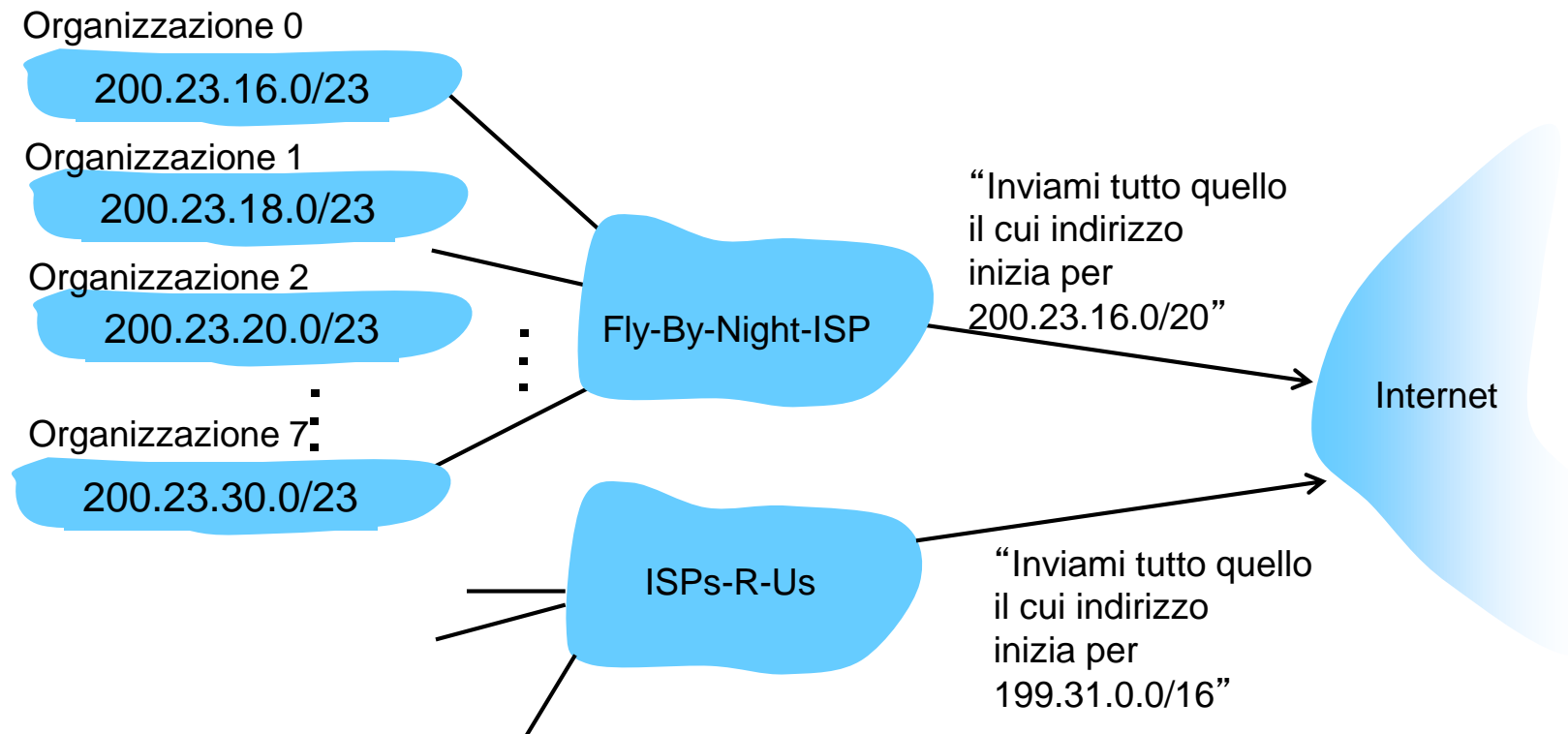
**D:** cosa deve fare un amministratore di rete per ottenere un blocco di indirizzi IP da usare in una sottorete?

**R:** deve contattare il proprio ISP e farsi assegnare una porzioni dello spazio di indirizzi

blocco dell'ISP	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organizzazione 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organizzazione 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organizzazione 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...	....			....	....
Organizzazione 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

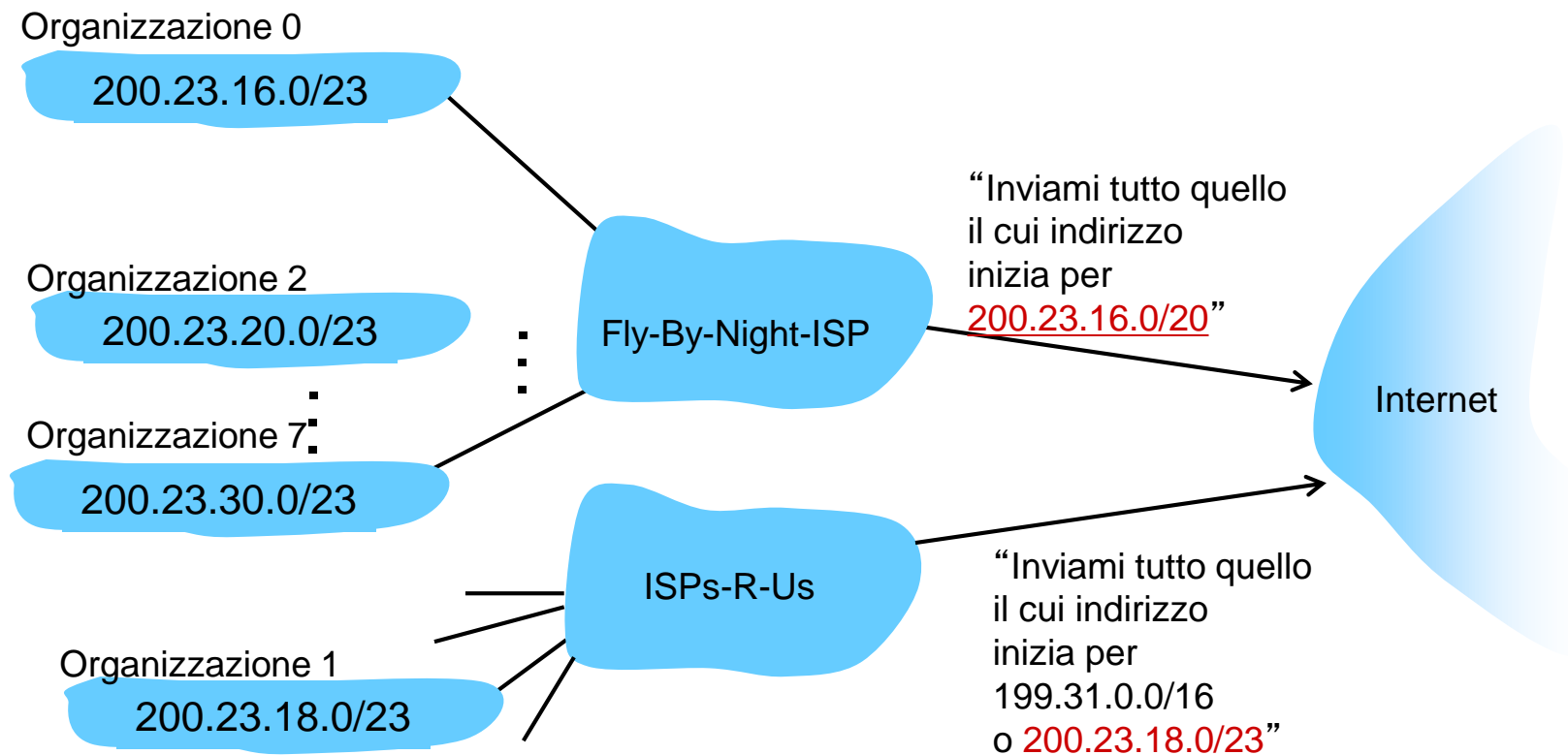
# Indirizzamento gerarchico

l'indirizzamento gerarchico consente una diffusione efficiente delle informazioni di routing:



# Percorsi specifici

ISPs-R-Us presenta un percorso più specifico verso Organizzazione I



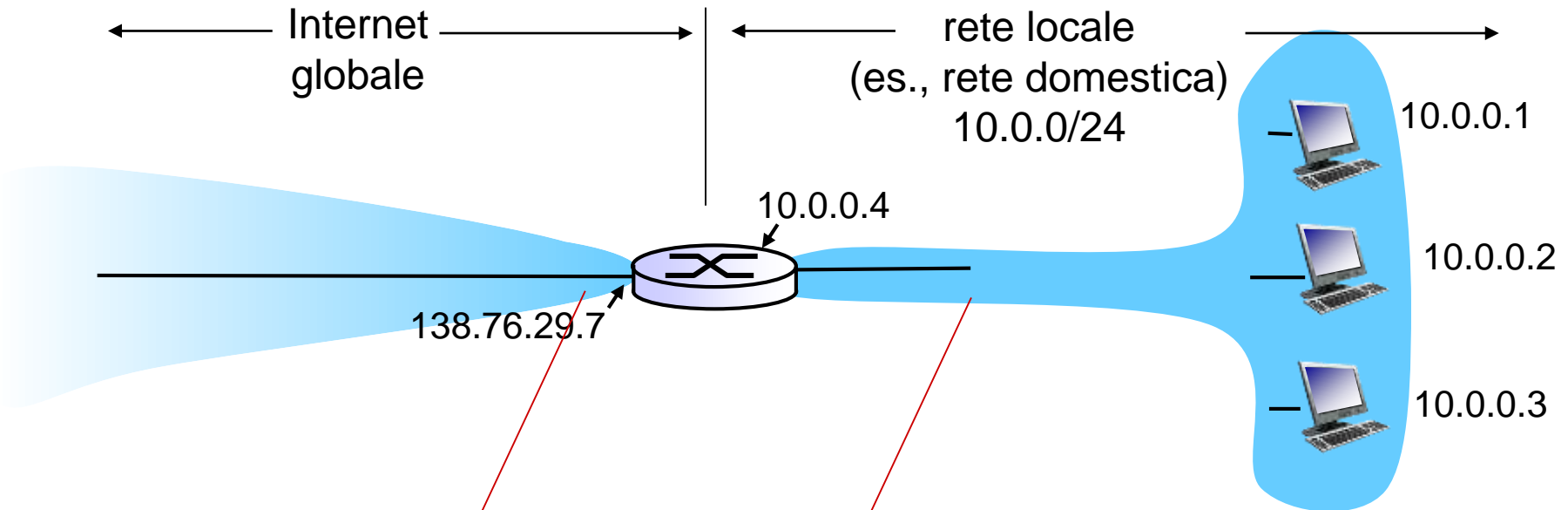
# Indirizzi IP: l'ultima parola...

**D:** ma come fa un ISP, a sua volta, a ottenere un blocco di indirizzi?

**R:** **ICANN**: Internet Corporation for Assigned Names and Numbers <http://www.icann.org/>

- ha la responsabilità di allocare i blocchi di indirizzi
- gestisce i server radice DNS
- assegna e risolve dispute sui nomi di dominio

# NAT: network address translation



*tutti* i datagrammi *uscenti* dalla rete locale hanno un unico indirizzo IP di NAT *uguale per tutti*: 138.76.29.7, con differenti numeri di porta sorgente

i datagrammi con sorgente o destinazione in questa rete hanno indirizzi 10.0.0/24 per sorgente e destinazione (come di norma)



# NAT: network address translation

*motivazione:* la rete locale usa un solo indirizzo IP visibile al mondo esterno:

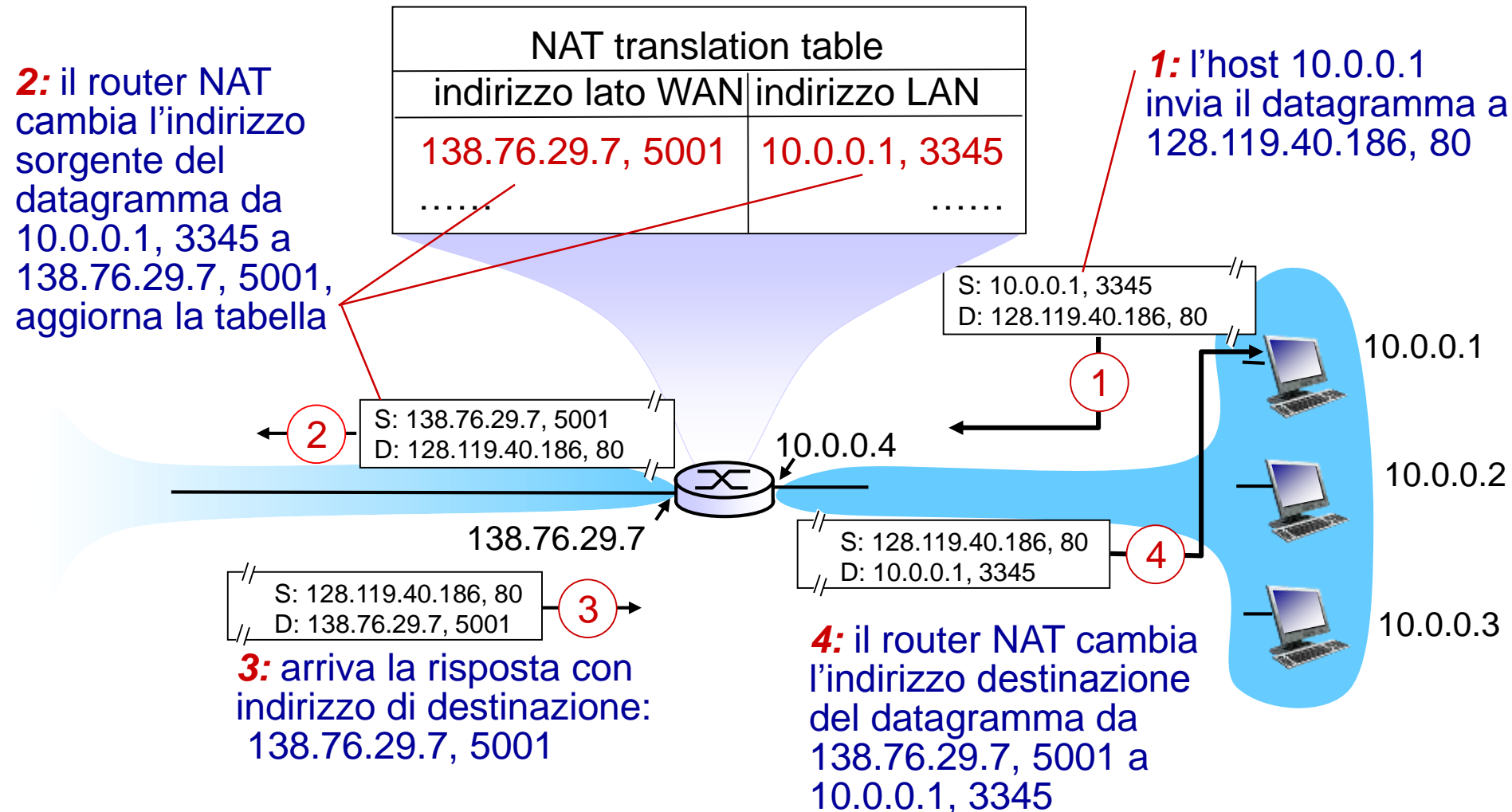
- non è necessario allocare un intervallo di indirizzi da un ISP: un unico indirizzo IP è sufficiente per tutte le macchine di una rete locale
- è possibile cambiare gli indirizzi delle macchine di una rete privata senza doverlo comunicare all'Internet globale
- i dispositivi della rete locale non sono esplicitamente indirizzabili e quindi raggiungibili dall'esterno (una sicurezza in più)
- è possibile cambiare ISP senza modificare gli indirizzi delle macchine della rete privata

# NAT: network address translation

*implementazione:* il router NAT deve:

- *datagrammi uscenti: sostituire* (IP sorgente, # di porta) di ogni datagramma uscente in (indirizzo IP di NAT, nuovo # di porta)  
... client/server remoti risponderanno usando (indirizzo IP di NAT, nuovo # di porta) come destinazione
- *ricordare (nella tabella di traduzione NAT)* ogni coppia di traduzione da (IP sorgente, # di porta) a (indirizzo IP di NAT, nuovo # di porta)
- *datagrammi entranti: sostituire* (indirizzo IP di NAT, nuovo # di porta) nei campi destinazione di ogni datagramma entrante con i corrispondenti (IP sorgente, # di porta) memorizzati nella tabella di NAT

# NAT: network address translation

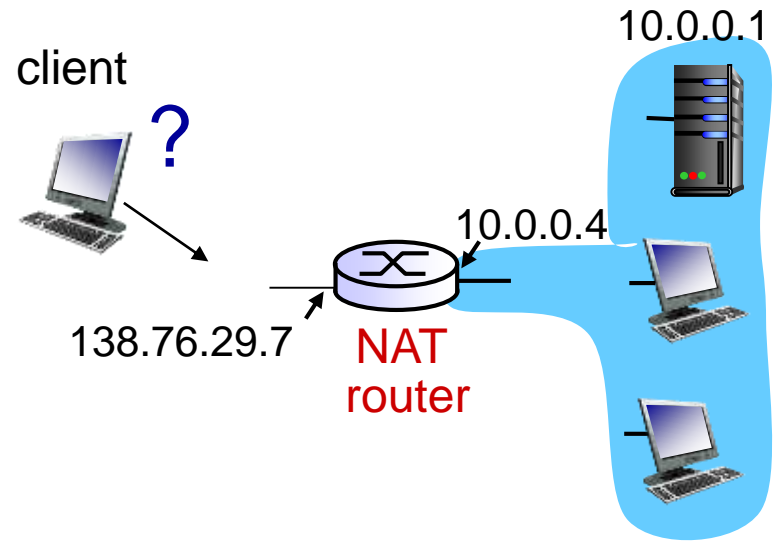


# NAT: network address translation

- ❖ campo numero di porta a 16-bit:
  - 60,000 connessioni simultanee con un singolo indirizzo esterno!
- ❖ NAT è controverso:
  - i router dovrebbero occuparsi solo del livello 3
  - viola il cosiddetto *argomento punto-punto*
    - interferenza con le applicazioni (es. P2P), che devono tenerne conto
  - per risolvere la scarsità di indirizzi IP si dovrebbe usare IPv6

# Problema del NAT traversal

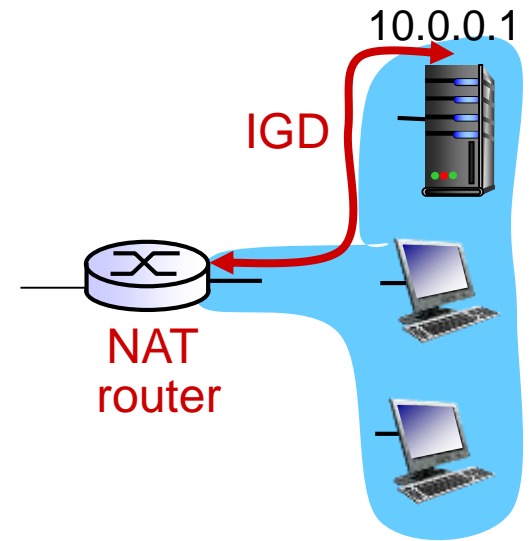
- ❖ il client si vuole connettere con il server all'indirizzo 10.0.0.1
  - l'indirizzo del server 10.0.0.1 è locale alla LAN (il client non può usarlo come indirizzo di destinazione)
  - un solo indirizzo NAT visibile esternamente: 138.76.29.7
- ❖ **soluzione I**: configurare staticamente il NAT per inoltrare le connessioni in ingresso, con una determinata porta, verso il server
  - es., (123.76.29.7, porta 2500) sempre inoltrate a 10.0.0.1 porta 25000



# Problema del NAT traversal

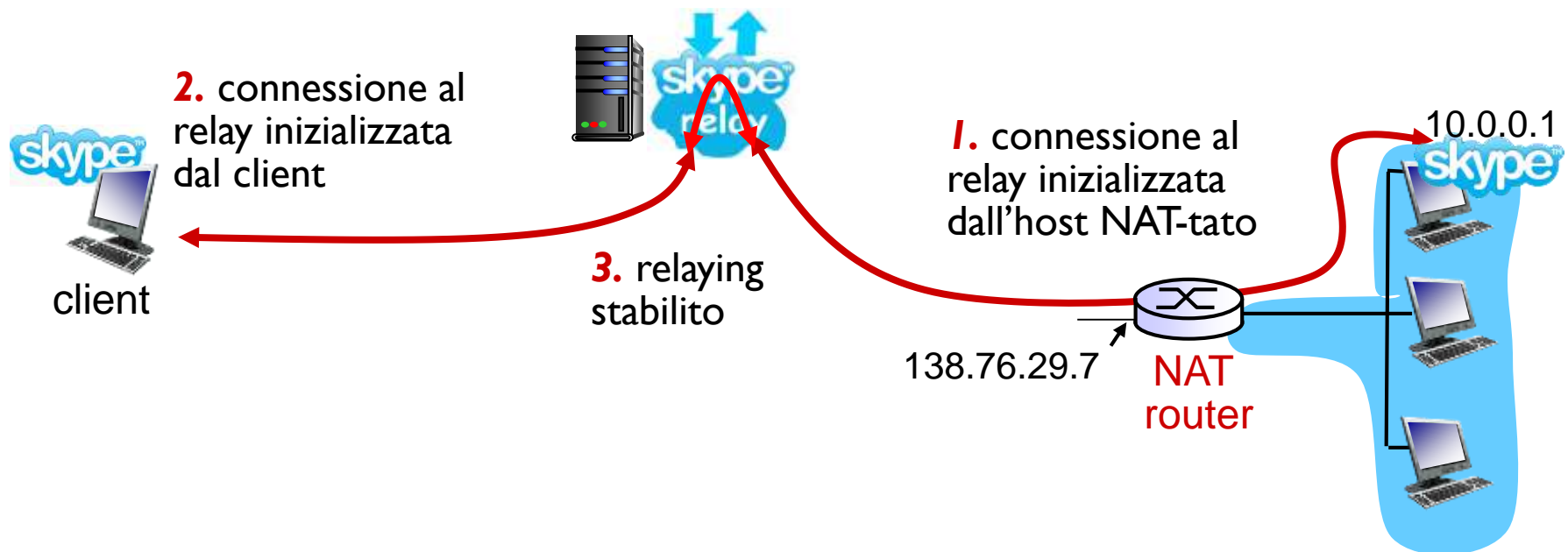
- ❖ **soluzione 2:** Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Consente agli host NAT-tati di:
  - ❖ imparare l'indirizzo IP pubblico (138.76.29.7)
  - ❖ aggiungere/rimuovere il port mapping (con un tempo di lease)

i.e., automatizza la configurazione statica del NAT port map



# Problema del NAT traversal

- ❖ **soluzione 3:** relaying (usato in Skype)
  - il client NAT-tato stabilisce una connessione con il relay
  - il client esterno si connette al relay
  - il relay smista i pacchetti tra le connessioni



# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast



# ICMP: internet control message protocol

- ❖ usato da host e router per scambiarsi informazioni a livello di rete

- report degli errori: host irraggiungibili , rete, porta, protocollo
- echo request/reply (usato da ping)

- ❖ livello di rete “sopra” IP:
  - i messaggi ICMP sono trasportati da datagrammi IP

- ❖ **messaggi ICMP:** hanno un campo tipo e un campo codice, più i primi 8 byte del datagramma IP che ha causato l'errore

<u>Tipo</u>	<u>Codice</u>	<u>descrizione</u>
0	0	risposta eco (a ping)
3	0	rete destin. irraggiungibile
3	1	host destin. irraggiungibile
3	2	protocollo dest. irraggiungibile
3	3	porta destin. irraggiungibile
3	6	rete destin. sconosciuta
3	7	host destin. sconosciuto
4	0	riduzione (controllo di congestione)
8	0	richiesta eco (ping)
9	0	annuncio del router
10	0	scoperta del router
11	0	TTL scaduto
12	0	errata intestazione IP

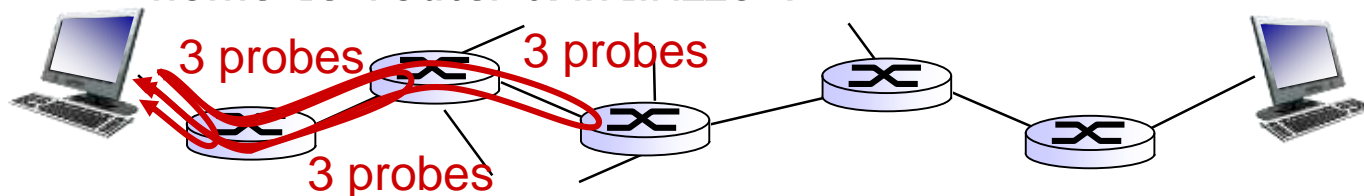
# Traceroute e ICMP

- ❖ la sorgente invia alcune serie di segmenti UDP verso la destinazione
  - la prima serie ha TTL = 1
  - la seconda ha TTL=2, etc.
  - numero di porta improbabile
- ❖ quando l' $n$ -esima serie di datagrammi arriva all' $n$ -esimo router:
  - il router scarta il datagramma
  - invia al sorgente un messaggio ICMP (tipo 11, codice 0)
  - il messaggio ICMP include nome del router & indirizzo IP

- ❖ quando arrivano i messaggi ICMP, l'origine calcola i RTT

## *criteri di arresto:*

- ❖ quando un segmento UDP arriva all'host di destinazione
- ❖ l'host di destinazione restituisce un messaggio ICMP di porta non raggiungibile (tipo 3, codice 3)
- ❖ quando l'origine riceve questo messaggio ICMP, si ferma



# IPv6: motivazioni

- ❖ *motivazione iniziale*: lo spazio di indirizzamento IP a 32 bit vicino all'esaurimento.
- ❖ ulteriori motivazioni:
  - il formato dell'intestazione aiuta a rendere più veloci i processi di elaborazione e inoltre
  - agevolare la QoS

## *Formato dei datagrammi IPv6:*

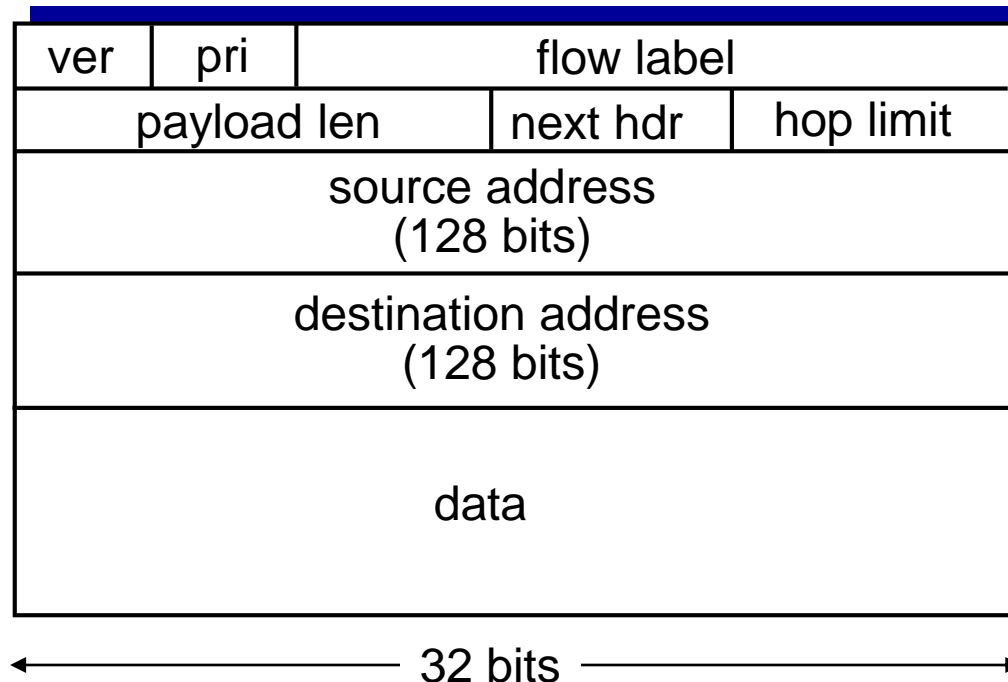
- intestazione a 40 byte e a lunghezza fissa
- non è consentita la frammentazione

# Formato dei datagrammi IPv6

*priorità:* attribuisce priorità a determinati datagrammi di un flusso.

*etichetta di flusso:* identifica i pacchetti che appartengono a flussi particolari (anche se non è ben definito il concetto di “flusso”).

*intestazione successiva:* identifica il protocollo del livello superiore

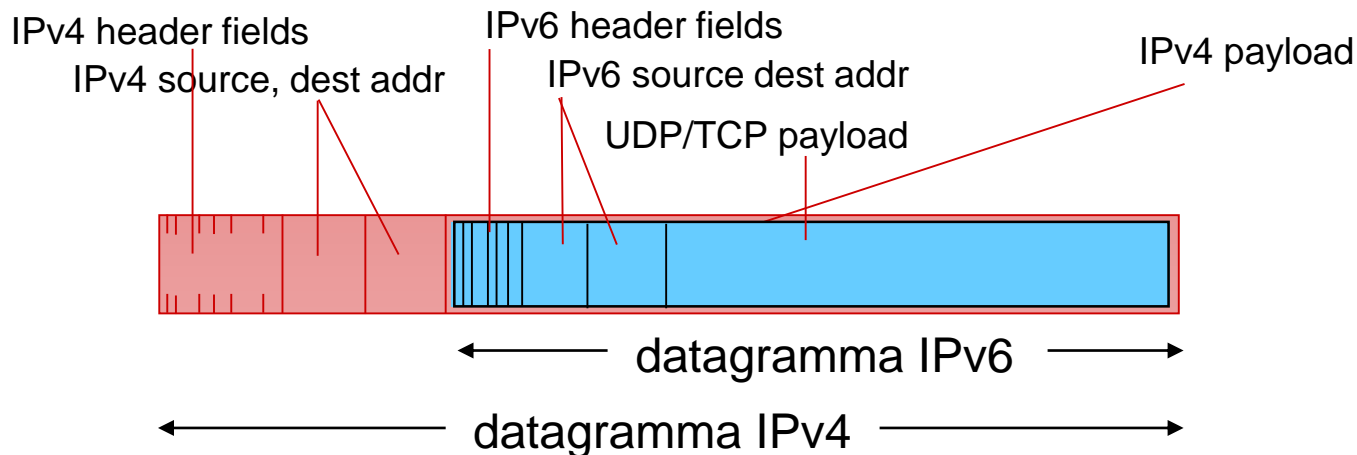


# Altri cambiamenti da IPv4

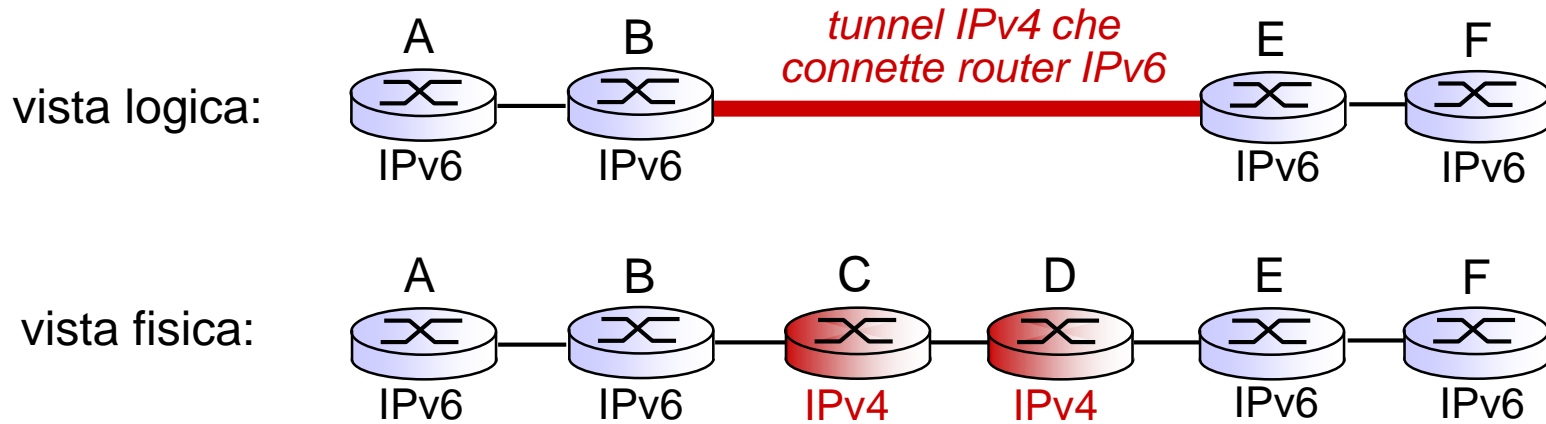
- ❖ *checksum*: rimossa per ridurre il tempo di calcolo a ogni hop
- ❖ *opzioni*: consentite, ma fuori dall'header, indicate dal campo "Next Header"
- ❖ *ICMPv6*: nuova versione of ICMP
  - nuovi tipi di messaggio, es. "Packet Too Big"
  - gestisce l'ingresso e l'uscita di host nei gruppi multicast

# Passaggio da IPv4 a IPv6

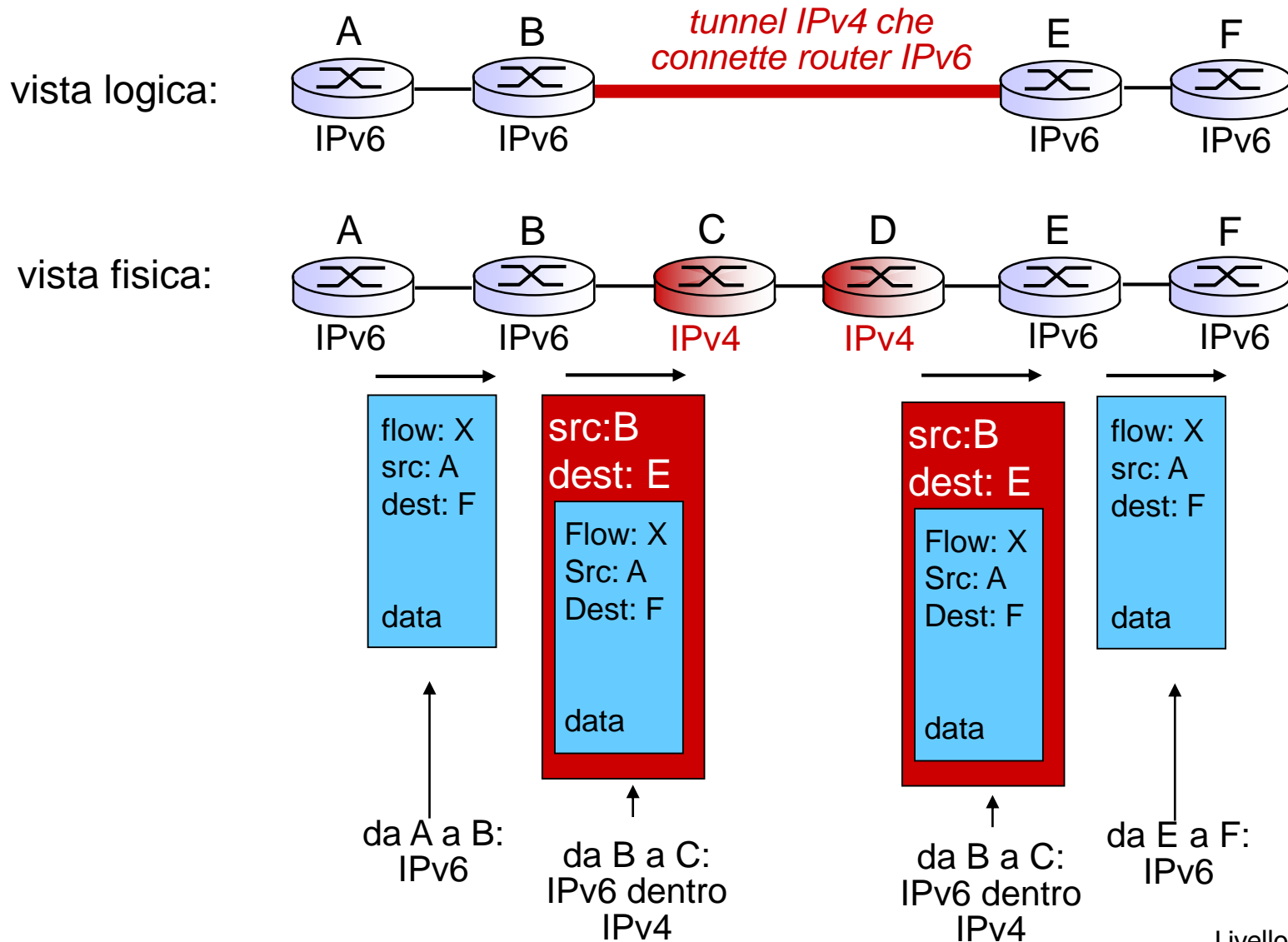
- ❖ non è possibile aggiornare simultaneamente tutti i router
  - impossibile dichiarare una “giornata campale”
  - come riuscirà la rete a funzionare in presenza di router IPv4 e IPv6?
- ❖ **tunneling**: IPv6 viene trasportato come *payload* in datagrammi IPv4 quando attraversa router IPv4



# Tunneling



# Tunneling





# IPv6: adozione

- ❖ US National Institutes of Standards stimava nel [2013]:
  - ~3% dei router IP nell'industria
  - ~11% nei router del governo degli USA
- ❖ *Lungo (lungo!) periodo per la diffusione e l'uso*
  - 20 anni e oltre!
  - tante applicazioni diffusissime si basano su IPv4: WWW, Facebook, ...

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

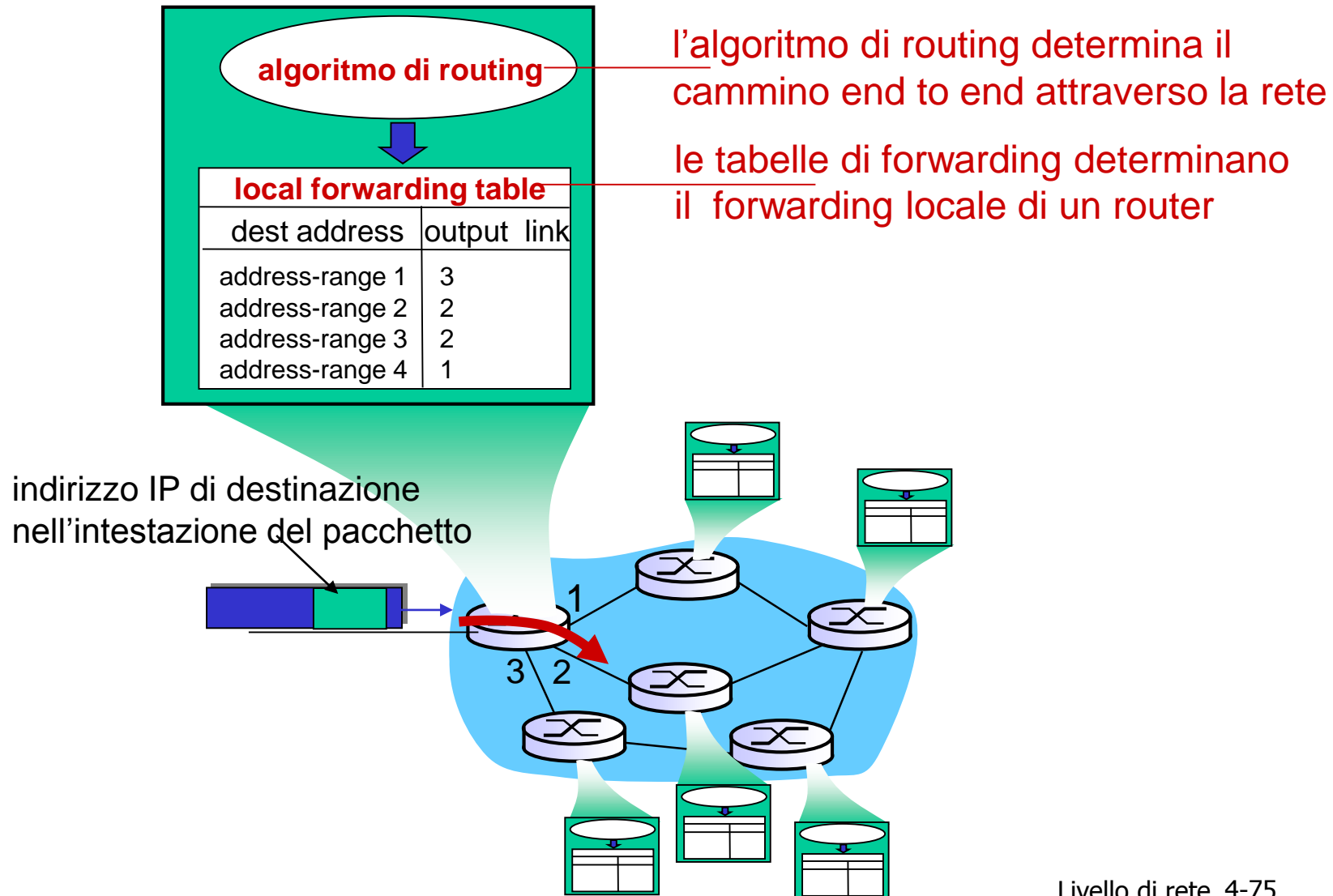
- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

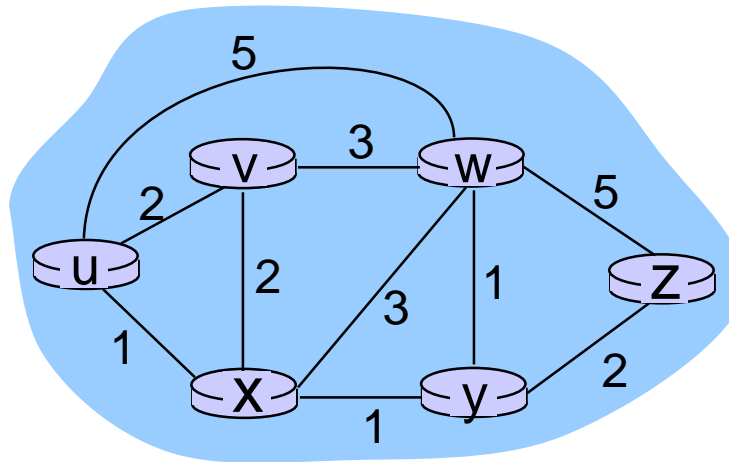
- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Relazione tra routing e forwarding



# Grafo di una rete di calcolatori



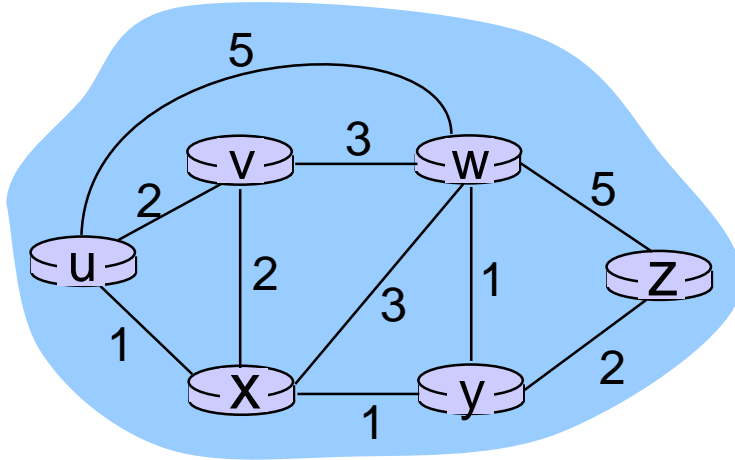
grafo:  $G = (N, E)$

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

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

*nota:* il grafo è un'astrazione utile anche in altri contesti di rete, es., P2P,  
dove  $N$  è un insieme di peer ed  $E$  è un insieme di connessioni TCP

# Grafo di una rete : costi



$c(x, x') = \text{costo del link } (x, x')$   
es.,  $c(w, z) = 5$

il costo può essere costante, o  
inversamente proporzionale alla  
bandwidth, o direttamente  
proporzionale alla congestione, e  
così via

costo di un cammino  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

**domanda chiave:** qual è il cammino di costo minimo tra u e z ?  
**algoritmo di routing :** algoritmo che calcola il cammino di costo minimo

# Classificazione degli algoritmi di routing

*D: informazione globale o distribuita?*

*globale:*

- ❖ tutti i routers hanno la topologia completa e i costi dei link
- ❖ algoritmi “link state”

*distribuiti:*

- ❖ i router conoscono i vicini fisicamente connessi e i costi verso tali vicini
- ❖ processo iterativo di calcolo e scambio di informazioni con i vicini
- ❖ algoritmi “distance vector”

*D: statico o dinamico?*

*statici:*

- ❖ i cammini cambiano molto lentamente nel tempo

*dinamici:*

- ❖ i cammini cambiano molto velocemente
  - aggiornamenti periodici
  - in risposta a cambiamenti dei costi di un link

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Un algoritmo di routing Link-State

## *algoritmo di Dijkstra*

- ❖ la topologia di rete e i costi dei link sono noti a tutti i nodi
  - tramite il “link state broadcast”
  - tutti i nodi hanno le stesse informazioni
- ❖ calcola il cammino a costo minimo da un nodo (origine) a tutti gli altri nodi della rete
  - crea una *forwarding table* per quell nodo
- ❖ iterativo: dopo k iterazioni si hanno i cammini a costo minimo verso k destinazioni

## *notazione:*

- ❖  $C(x,y)$ : costo del link dal nodo x al nodo y;  $= \infty$  ie non sono adiacenti
- ❖  $D(v)$ : costo corrente del cammino dal nodo origine alla destinazione v
- ❖  $p(v)$ : immediato predecessore di v lungo il cammino dal nodo origine alla destinazione v
- ❖  $N'$ : sottoinsieme di nodi per cui il cammino a costo minimo dall'origine è calcolato definitivamente



# Algoritmo di Dijkstra

1 **Inizializzazione:**

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  $D(v) = \infty$

7

8 **Loop**

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

14 il costo del cammino minimo noto verso  $w$  più il costo da  $w$  a  $v$  \*/

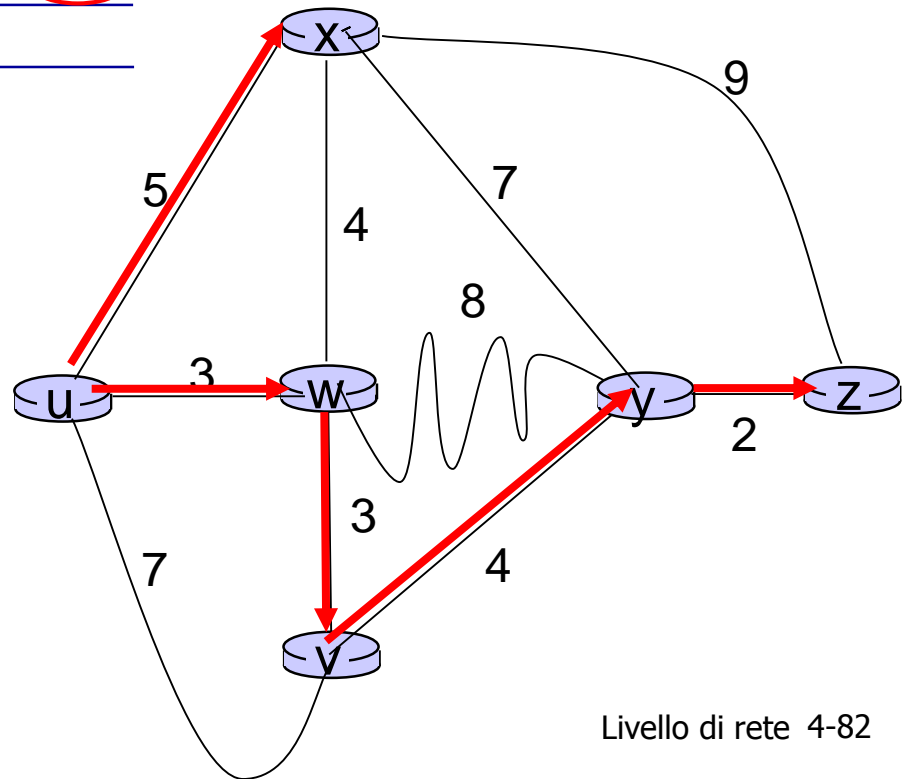
15 **finché tutti i nodi sono in  $N'$**

# 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	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy				12,y	
5	uwxvyz					

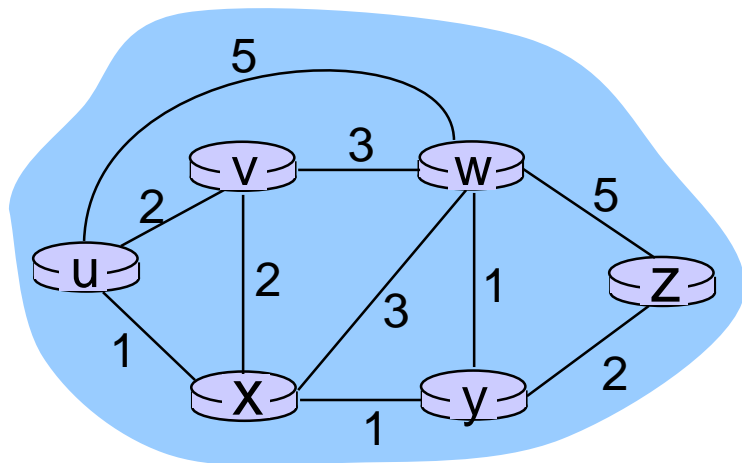
## note:

- ❖ costruisce l'albero dei cammini minimi tenendo traccia dei nodi predecessori
- ❖ possono esserci cammini equivalenti (da scegliere arbitrariamente)



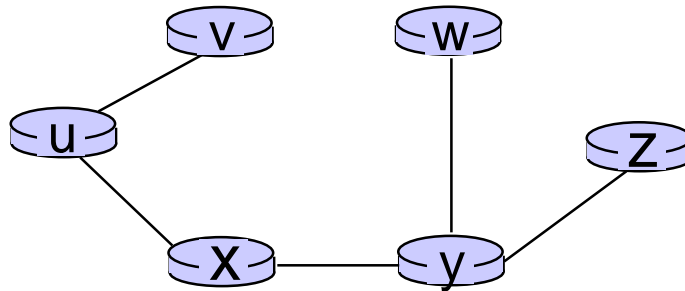
# Algoritmo di Dijkstra: altro esempio

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



# Algoritmo di Dijkstra: esempio (2)

albero dei cammini minimi da u ottenuto:



forwarding table in u:

destinazione	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

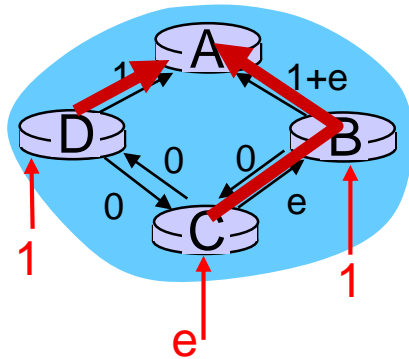
# Algoritmo di Dijkstra, discussione

*complessità dell'algoritmo:* per  $n$  nodi

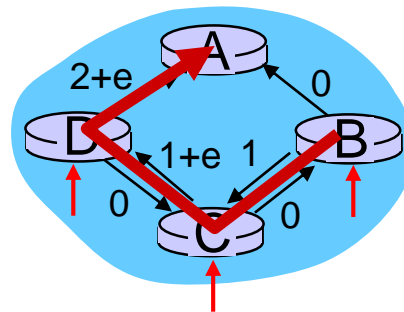
- ❖ ogni iterazione: deve controllare tutti i nodi,  $w$ , non in  $N$
- ❖  $n(n+1)/2$  confronti:  $O(n^2)$
- ❖ con implementazioni più efficienti si può ottenere:  $O(n \log n)$

*possibili oscillazioni:*

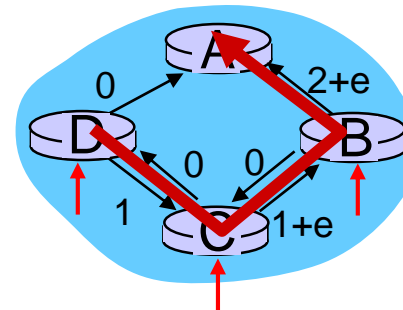
- ❖ es., costo del link calcolato in base al traffico trasportato:



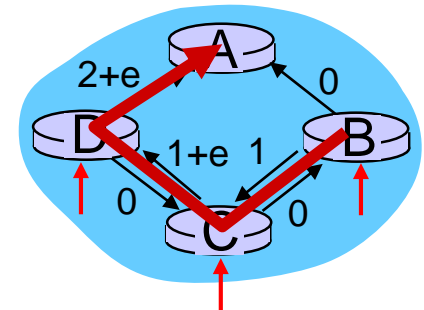
situazione  
iniziale



con questi costi,  
calcola un nuovo  
routing....  
che porta a nuovi costi



con questi costi,  
calcola un nuovo  
routing....  
che porta a nuovi costi



con questi costi,  
calcola un nuovo  
routing....  
che porta a nuovi costi

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Algoritmo distance vector

*formula di Bellman-Ford (programmazione dinamica)*

sia

$d_x(y)$  := il costo del cammino a costo minimo da  $x$  a  $y$

allora

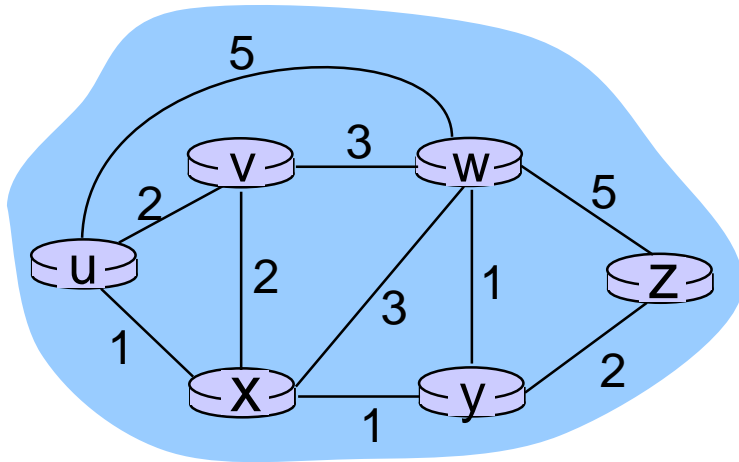
$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

costo dal vicino  $v$  alla destinazione  $y$

costo verso il vicino  $v$

$\min$  tra tutti i vicini  $v$  di  $x$

# Bellman-Ford: esempio



$$d_v(z) = 5, d_x(z) = 3, d_w(z) = 3$$

l'equazione B-F 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 dà il cammino minimo è il next hop , usato nella forwarding table



# Algoritmo distance vector

- ❖  $D_x(y)$  = stima del costo minimo da  $x$  a  $y$ 
  - $x$  mantiene il vettore distanza  $\mathbf{D}_x = [D_x(y): y \in N]$
- ❖ nodo  $x$ :
  - conosce il costo verso ogni vicino  $v$ :  $c(x,v)$
  - mantiene il vettore distanza dei suoi vicini. Per ogni vicino  $v$ ,  $x$  mantiene  $\mathbf{D}_v = [D_v(y): y \in N]$

# Algoritmo distance vector

## *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, lo salva e usa la formula B-F per aggiornare il proprio vettore distanza:

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

- ❖ in condizioni normali, la stima dei costi  $D_x(y)$  converge verso gli effettivi costi  $d_x(y)$

# Algoritmo distance vector

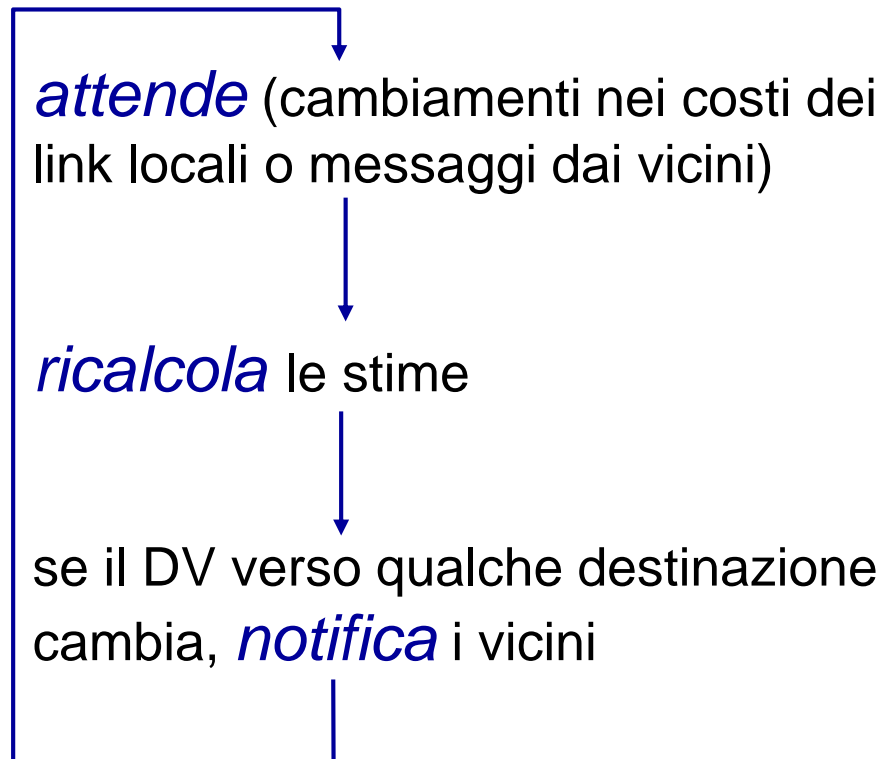
*iterativo, asincrono:* ogni iterazione locale è causata da:

- ❖ cambio del costo di uno dei collegamenti
- ❖ messaggio di aggiornamento del DV da un vicino

*distribuito:*

- ❖ ogni nodo aggiorna i suoi vicini *solo* quando il suo DV cambia
  - i vicini allora avvisano i vicini se necessario

*ogni nodo:*



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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

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

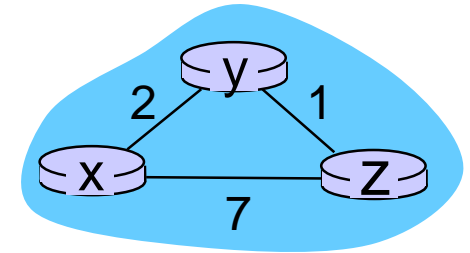
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y  
table**

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

**node z  
table**

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



tempo

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

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

**node x  
table**

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

**node y  
table**

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

**node z  
table**

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

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

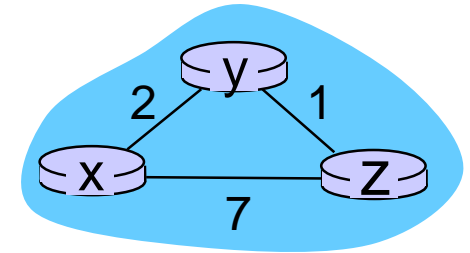
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

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

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

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

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

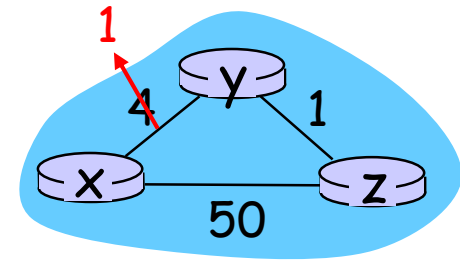


time →

# Distance vector: cambiamenti nei costi

## *i costi dei link cambiano:*

- ❖ un nodo rileva un cambiamento nel costo dei collegamenti
- ❖ aggiorna il proprio vettore distanza DV
- ❖ se il DV cambia, notifica i vicini



“le  
buone  
notizie  
viaggiano  
veloce”

$t_0$ : y rileva il cambiamento nel costo del collegamento, aggiorna il proprio DV e informa i vicini del cambiamento.

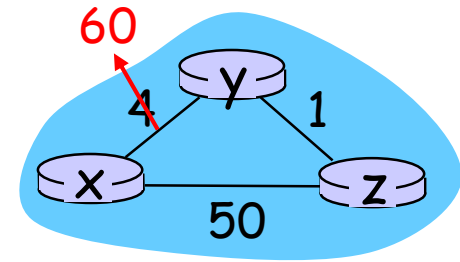
$t_1$ : z riceve l'aggiornamento da y e aggiorna la propria tabella, calcola un nuovo costo minimo verso x e invia il nuovo DV ai vicini.

$t_2$ : y riceve l'aggiornamento di z e aggiorna la propria tabella di distanza. I costi minimi di y non cambiano e y non manda alcun messaggio a z.

# Distance vector: cambiamenti nei costi

## *i costi dei link cambiano:*

- ❖ un nodo rileva un cambiamento nel costo dei collegamenti
- ❖ *le cattive notizie viaggiano lentamente* – problema del “conteggio all’infinito”!
- ❖ 44 iterazioni prima che l’algoritmo di stabilizzi



## *inversione avvelenata (poisoned reverse):*

- ❖ se Z fa passare per Y per giungere alla destinazione X:
  - Z dirà a Y che la sua distanza verso X è infinita (così Y non tenterà mai d'instradare verso X passando per Z)
- ❖ questo resolve completamente il problema del conteggio all’infinito?

# Confronto tra LS e DV

## *complessità dei messaggi*

- ❖ **LS:** con  $n$  nodi,  $E$  collegamenti, si ha  $O(nE)$  messaggi
- ❖ **DV:** richiede scambi solo tra nodi adiacenti
  - il tempo di convergenza varia

## *velocità di convergenza*

- ❖ **LS:** l'algoritmo  $O(n^2)$  richiede  $O(nE)$  messaggi
  - ci possono essere oscillazioni di velocità
- ❖ **DV:** il tempo di convergenza varia
  - possono esserci routing loop
  - problema del conteggio all'infinito

**robustezza:** cosa avviene se un router funziona male?

### **LS:**

- un nodo può comunicare un costo sbagliato di un *link*
- un nodo calcola soltanto la *propria* tabella

### **DV:**

- un nodo può comunicare un costo sbagliato di un *cammino*
- la tabella di ciascun nodo è usata dagli altri
  - un calcolo errato si propaga per l'intera rete



# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Routing gerarchico

finora abbiamo considerato la rete come se

- ❖ ogni router fosse identico agli altri
- ❖ rete “piatta”

... *non* è così nella realtà

**scala:** con milioni di destinazioni:

- ❖ non si possono tenere tutte le destinazioni nelle tabelle di routing!
- ❖ il traffico generato dagli aggiornamenti LS non lascerebbero banda per i pacchetti di dati!

**autonomia amministrativa**

- ❖ internet = la rete delle reti
- ❖ ogni amministratore di rete vorrebbe poter controllare il routing della propria rete

# Routing gerarchico

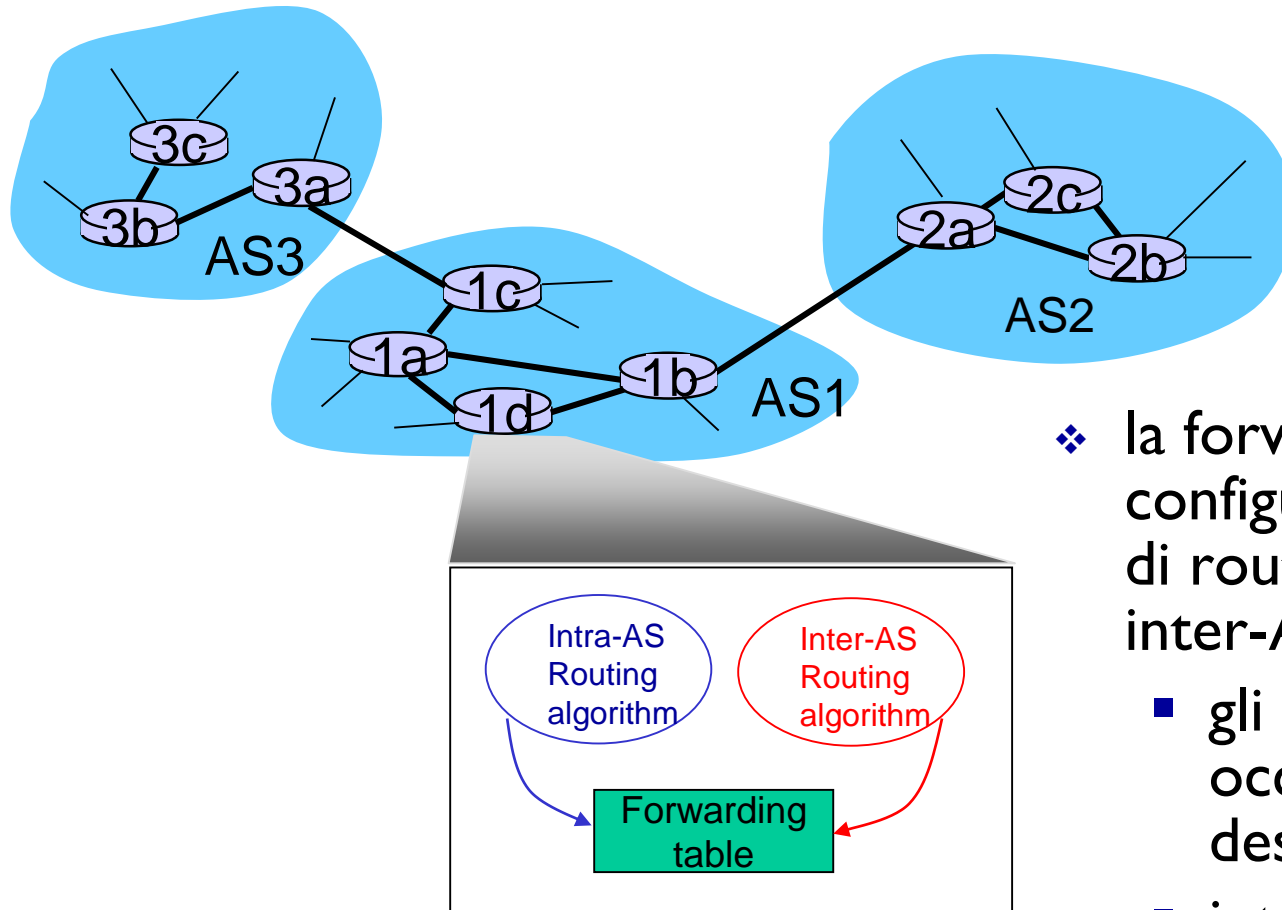
- ❖ i router sono aggregati in regioni,  
“autonomous systems” (AS)

- ❖ i routers nello stesso AS eseguono lo stesso protocollo di routing
  - protocollo di routing “intra-AS”
  - i router appartenenti a differenti AS possono eseguire protocolli di routing intra-AS diversi

## *gateway router:*

- ❖ al “confine” del proprio AS
- ❖ hanno dei link verso router in altri AS

# AS interconnessi



- ❖ la forwarding table è configurata dagli algoritmi di routing sia intra-AS che inter-AS
  - gli intra-AS si occupano delle destinazioni interne
  - inter-AS & intra-AS combinati si occupano delle destinazioni esterne

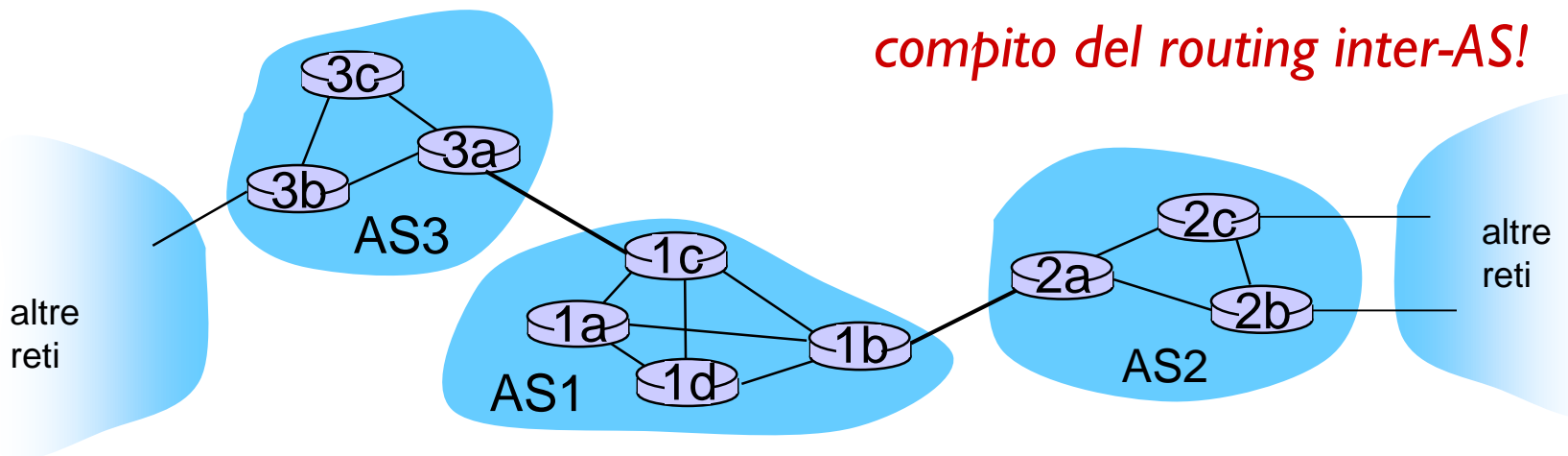
# Compiti dei router inter-AS

- ❖ supponiamo che un router in AS1 riceva un datagramma la cui destinazione ricade al di fuori di AS1
  - il router dovrebbe inoltrare il pacchetto verso uno dei due gateway, ma quale?

*AS1 deve:*

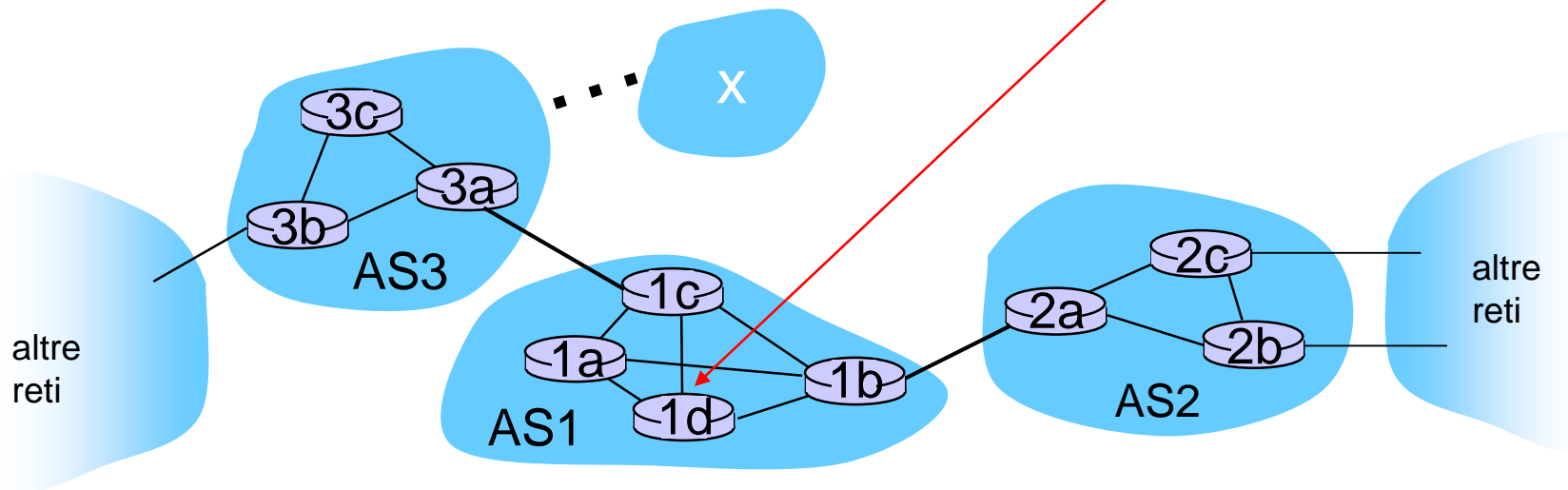
1. sapere quali destinazioni sono raggiungibili attraverso AS2 e quali attraverso AS3
2. far propagare questa informazione di raggiungibilità a tutti i router dentro AS1

*compito del routing inter-AS!*



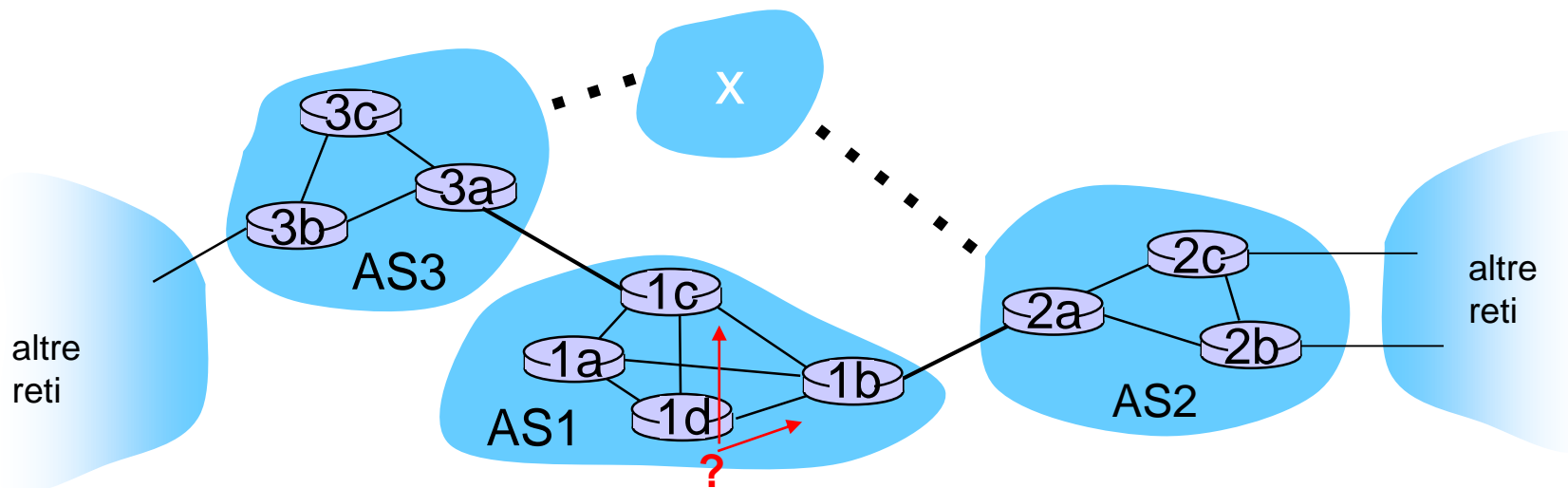
# Esempio: impostare la forwarding table nel router 1d

- ❖ supponiamo che AS1 apprenda (dal protocollo inter-AS) che la sottorete **x** è raggiungibile da AS3 (gateway 1c), ma non da AS2
  - il protocollo inter-AS propaga questa informazione a tutti i propri router interni
- ❖ il router 1d determina dalle informazioni del protocollo intra-AS che la sua interfaccia 1 è sul percorso a costo minimo verso 1c
  - può inserire nella forwarding table la riga **(x,1)**



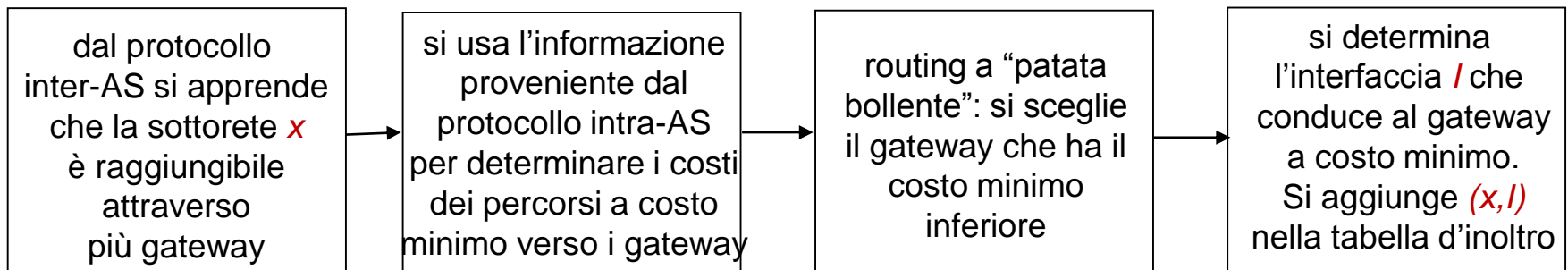
# Esempio: scegliere tra AS multipli

- ❖ ora supponiamo che AS1 apprenda (dal protocollo inter-AS) che la sottorete x è raggiungibile da AS3 (gateway 1c), e da AS2
- ❖ per impostare la forwarding table, il router 1d deve determinare verso quale gateway deve inoltrare i pacchetti verso la destinazione **x**
  - anche questo è un compito del protocollo inter-AS!



# Esempio: scegliere tra AS multipli

- ❖ ora supponiamo che AS1 apprenda (dal protocollo inter-AS) che la sottorete  $x$  è raggiungibile da AS3 (gateway  $I_c$ ), e da AS2
- ❖ per impostare la forwarding table, il router  $I_d$  deve determinare verso quale gateway deve inoltrare i pacchetti verso la destinazione  $x$ 
  - anche questo è un compito del protocollo inter-AS!
- ❖ *routing a “patata bollente”*: il sistema autonomo si sbarazza del pacchetto (patata bollente) non appena possibile





# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

## 4.6 routing in Internet

- RIP
- OSPF
- BGP

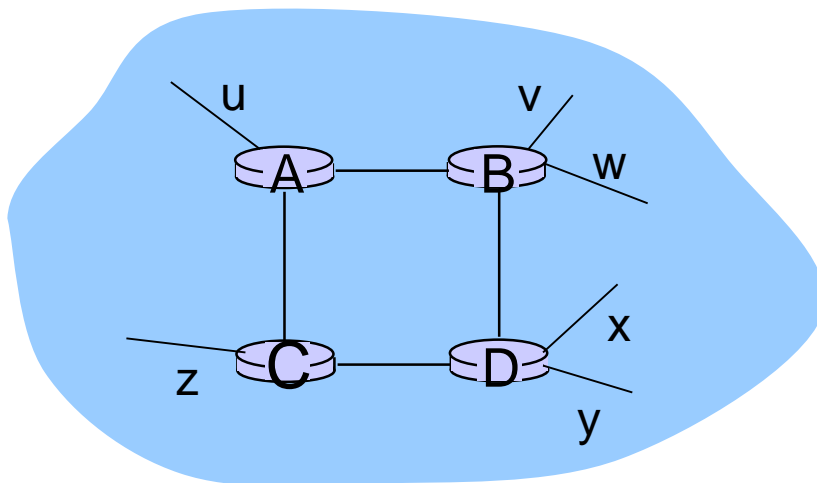
## 4.7 routing broadcast e multicast

# Routing intra-AS

- ❖ noti anche come *interior gateway protocols (IGP)*
- ❖ i protocolli di routing intra-AS più comuni sono:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (proprietario Cisco)

# RIP ( Routing Information Protocol)

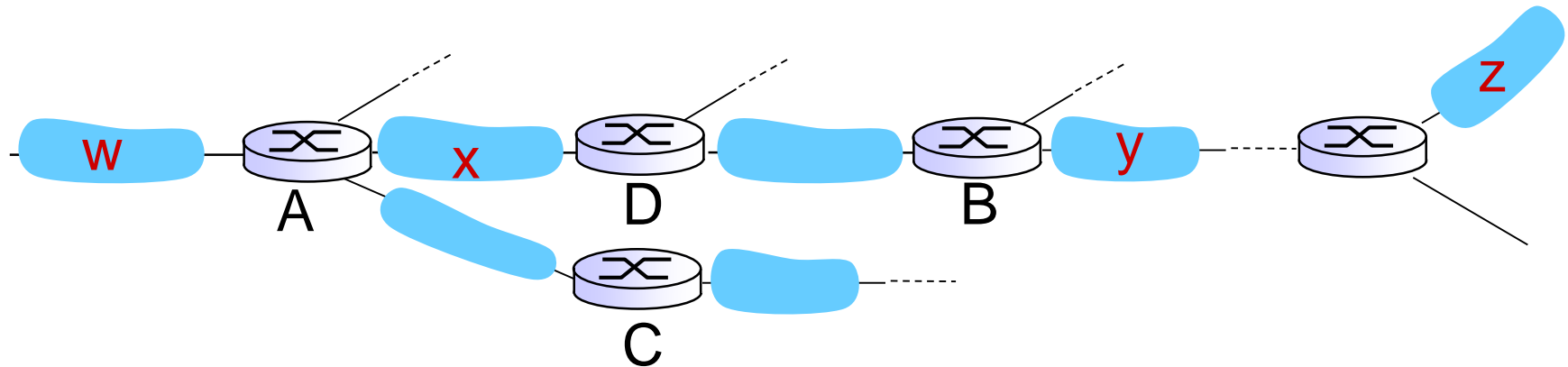
- ❖ incluso nelle distribuzioni BSD-UNIX nel 1982
- ❖ algoritmo distance vector
  - metrica di costo: numero di hop (max = 15 hop), ogni link ha costo 1
  - i DV vengono scambiati con i vicini ogni 30 sec in messaggi di risposta (**RIP advertisement**)
  - advertisement: lista con fino a 25 **subnet** di destinazione



dal router A alle **subnet** di destinazione:

<u>subnet</u>	<u>hop</u>
u	1
v	2
w	2
x	3
y	3
z	2

# RIP: esempio



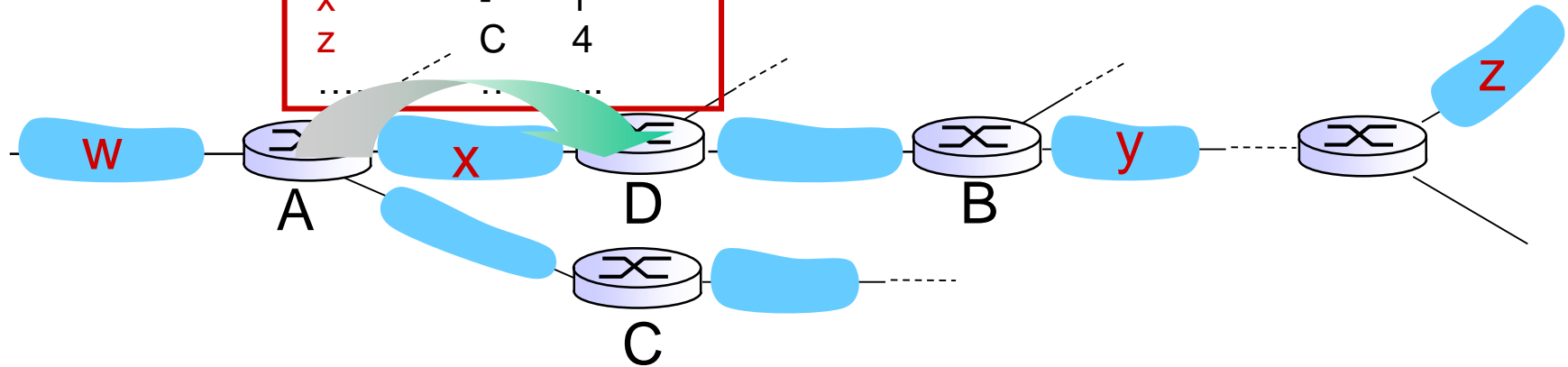
routing table del router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

# RIP: esempio

A-to-D advertisement

dest	next	hops
W	-	1
X	-	1
Z	C	4
...	...	...



routing table del router D

destination subnet	next router	# hops to dest
W	A	2
y	B	2
Z	<del>B</del> → A	<del>7</del> → 5
X	--	1
....	....	....

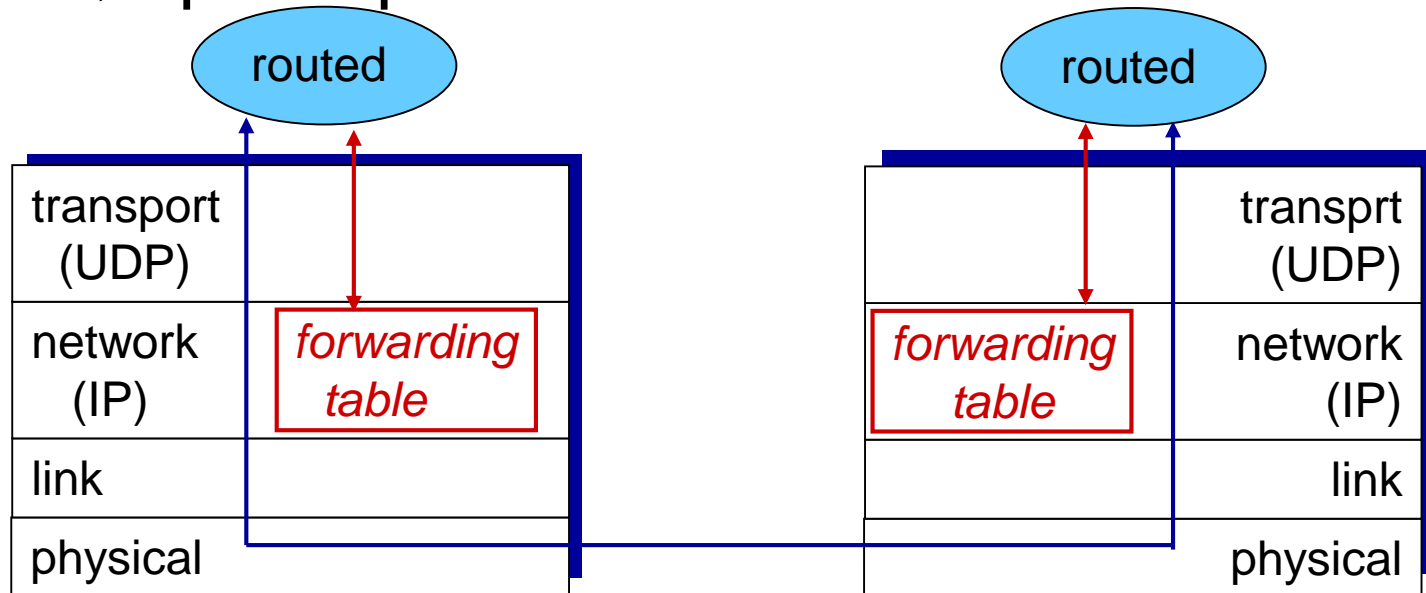
# RIP: guasto sul collegamento e recupero

se un router non riceve nulla dal suo vicino per 180 sec  
--> il vicino/il link viene considerato guasto (dead)

- le rotte attraverso quel vicino vengono invalidate
- nuovi advertisement vengono inviati ai vicini
- i vicini rispondono con nuovi advertisement (se le loro tabelle cambiano)
- l'informazione che il collegamento è fallito si propaga rapidamente su tutta la rete
- l'uso del *poison reverse* evita i loop (distanza infinita = 16 hop)

# Elaborazione delle tabelle RIP

- ❖ le tabelle di routing RIP vengono gestite da un processo a *livello di applicazione* chiamato route-d (route daemon)
- ❖ gli advertisement vengono inviati in pacchetti UDP, ripetuti periodicamente



# OSPF (Open Shortest Path First)

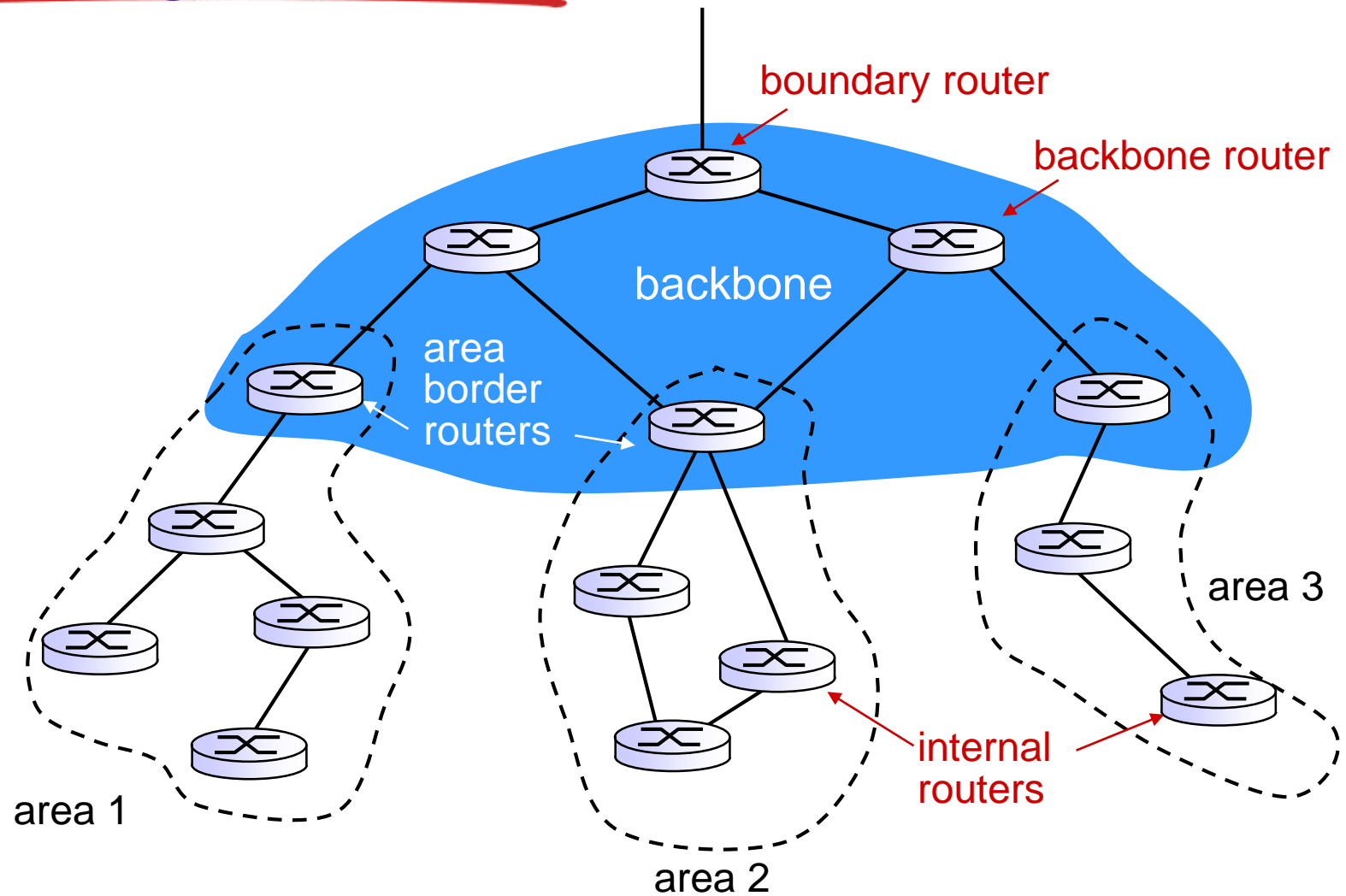
- ❖ “open”: protocollo pubblico
- ❖ usa un algoritmo link state
  - diffusione di pacchetti LS
  - topologia conosciuta da ogni nodo
  - usa l'algoritmo di Dijkstra per il calcolo delle rotte
- ❖ gli advertisement OSPF hanno una riga per vicino
- ❖ vengono diffusi sull'*intero* AS (flooding)
  - trasportati direttamente da IP (non da TCP o UDP)
- ❖ protocollo *IS-IS*: molto simile a OSPF



# Funzioni “avanzate” di OSPF (non in RIP)

- ❖ **sicurezza**: i messaggi OSPF sono autenticati
- ❖ sono consentiti cammini multipli con lo stesso costo (**multipath**) (solo un cammino in RIP)
- ❖ per ogni link, vi possono essere più metriche di costo per differenti **TOS** (es. il costo di un link satellitare sarà “basso” per un best effort; elevato per un real time)
- ❖ supporto integrato per uni- e **multicast** :
  - Multicast OSPF (MOSPF) usa lo stesso database sulla tipologia di OSPF
- ❖ OSPF **gerarchico** in grandi domini.

# OSPF gerarchico



# OSPF gerarchico

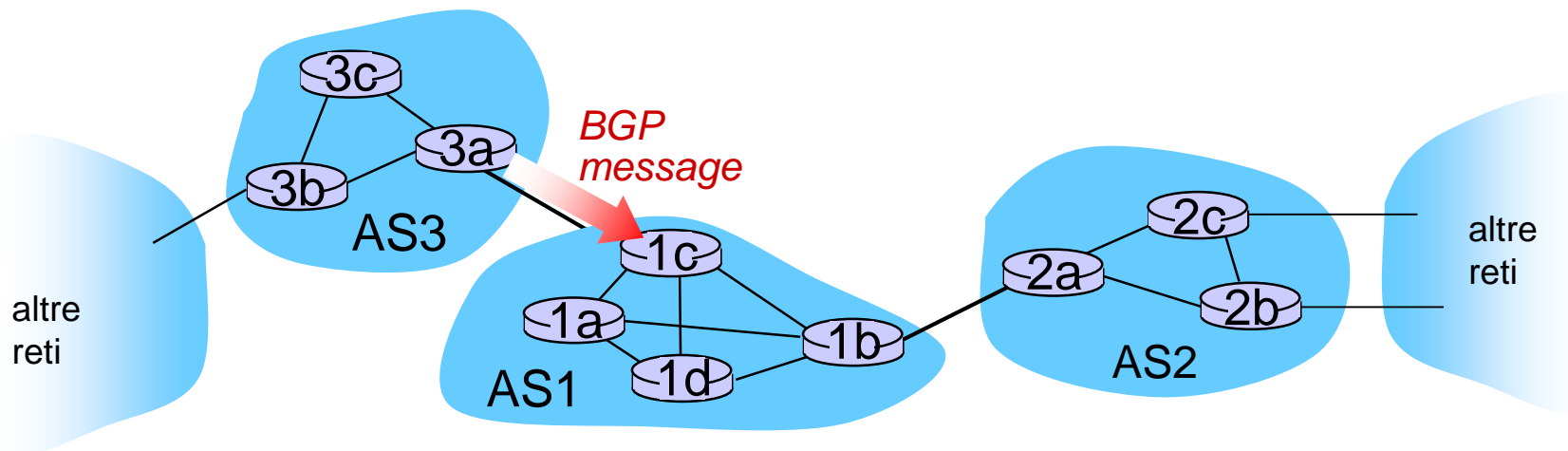
- ❖ *gerarchia a due livelli*: area locale, backbone.
  - messaggi di link-state solo all'interno dell'area
  - ogni nodo ha la topologia dettagliata dell'area; verso le reti nelle altre aree conosce solo la direzione (shortest path)
- ❖ *router di border area* : “riassumono” le distanze verso le reti della propria area, e le inviano agli altri router di Border Area.
- ❖ *router di dorsale (backbone)*: elaborano il routing OSPF limitatamente al backbone.
- ❖ *router di confine (boundary)*: scambiano informazioni con i router degli altri AS.

# Routing inter-AS in Internet: BGP

- ❖ **BGP (Border Gateway Protocol)**: il protocollo di routing inter-domain *de facto*
  - “la colla che tiene insieme Internet”
- ❖ BGP mette a disposizione di ciascun AS un modo per:
  - **eBGP**: ottenere informazioni sulla raggiungibilità delle sottoreti da parte di AS confinanti
  - **iBGP**: propagare le informazioni di raggiungibilità a tutti i router interni di un AS
  - determinare percorsi “buoni” verso le sottoreti sulla base delle informazioni di raggiungibilità e delle politiche dell’AS
- ❖ consente a ciascuna sottorete di comunicare la propria esistenza al resto di Internet: “*Sono qui*”

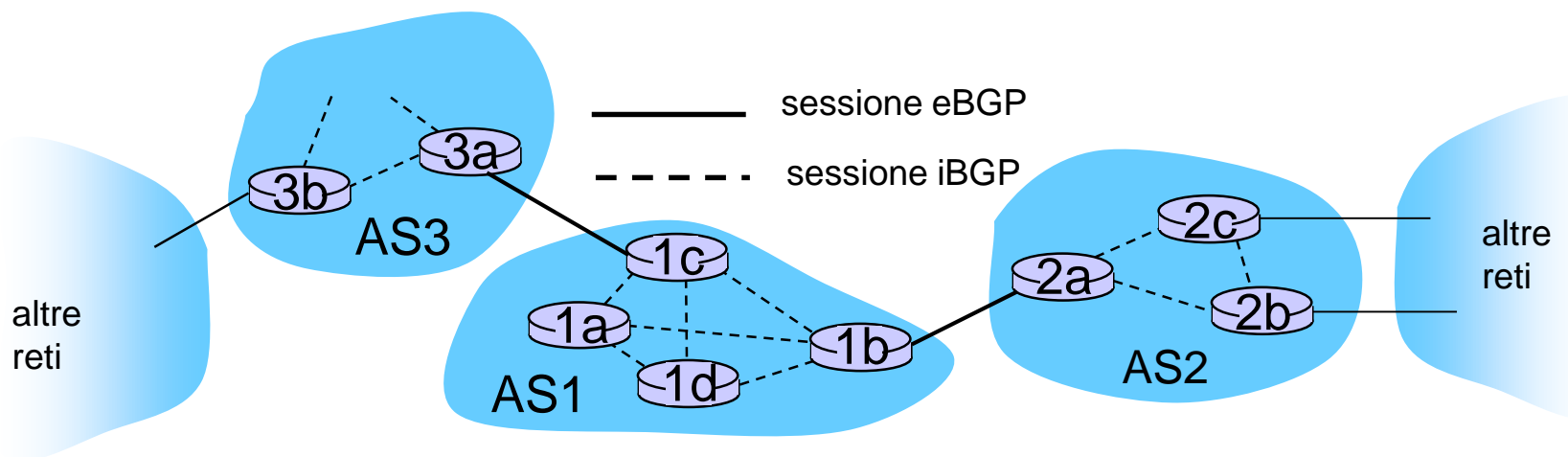
# Fondamenti di BGP

- ❖ **sessione BGP** : due router BGP (“peer”) si scambiano messaggi BGP:
  - annunciano *cammini* verso reti con un certo prefisso di destinazione (protocollo “path vector”)
  - gli scambi avvengono tramite connessioni TCP semi-permanenti
- ❖ quando AS3 invia un prefisso a AS1:
  - AS3 *promette* che inoltrerà i datagrammi verso quel prefisso
  - AS3 può aggregare più prefissi nel suo annuncio



# Distribuzione delle informazioni sui cammini

- ❖ in una sessione eBGP session tra 3a e 1c, AS3 invia a AS1 informazioni sui prefissi raggiungibili.
  - 1c utilizza le proprie sessioni iBGP per distribuire i prefissi agli altri router in AS1
  - 1b può allora ri-annunciare le nuove informazioni di raggiungibilità a AS2 tramite la sessione eBGP da 1b a 2a
- ❖ quando un router viene a conoscenza di un nuovo prefisso, lo memorizza in una nuova riga della propria forwarding table.



# Attributi dei percorsi e rotte BGP

- ❖ i prefissi annunciati includono attributi BGP
  - prefissi + attributi = “route”
- ❖ due attributi importanti:
  - **AS-PATH**: elenca i sistemi autonomi attraverso i quali è passato l'annuncio del prefisso: es., AS 67, AS 17
  - **NEXT-HOP**: indicano il router specifico interno all'AS collegato al next-hop AS. (may be multiple links from current AS to next-hop-AS)
- ❖ quando un router gateway riceve un annuncio di rotta, utilizza le proprie **politiche d'importazione** per accettare/rifiutare
  - es., mai passare attraverso l'AS x
  - routing *policy-based*

# Selezione dei percorsi BGP

- ❖ un router può ricavare più di una rotta verso un determinato prefisso, e deve quindi sceglierne una in base a:
  1. valore dell'attributo local preference (preferenza locale)
  2. shortest AS-PATH
  3. router NEXT-HOP più vicino : routing a patata bollente
  4. criteri aggiuntionali



# Messaggi BGP

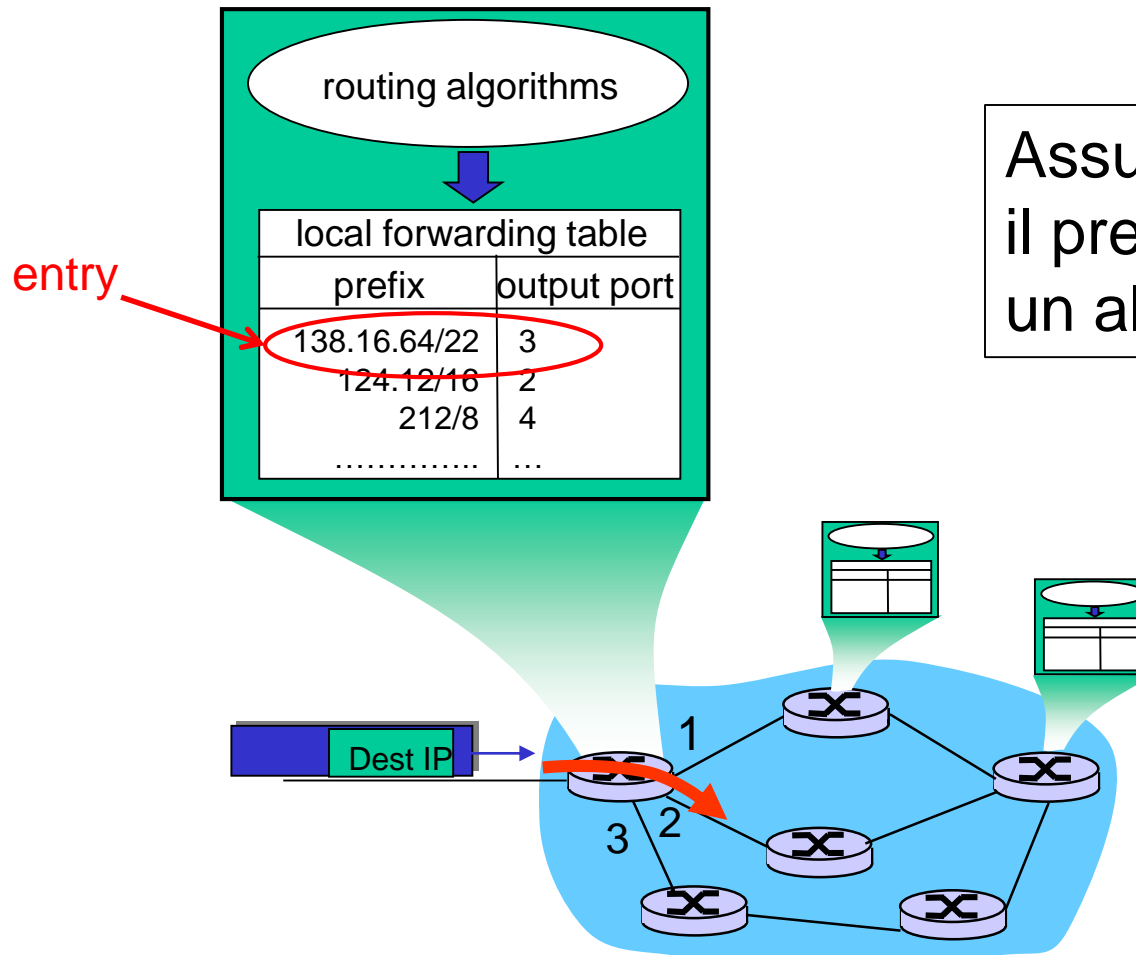
- ❖ i messaggi BGP vengono scambiati tra i peer attraverso connessioni TCP
- ❖ messaggi BGP:
  - **OPEN**: apre la connessione TCP e autentica il mittente
  - **UPDATE**: annuncia un nuovo percorso (o ne cancella uno vecchio)
  - **KEEPALIVE**: mantiene la connessione attiva in mancanza di UPDATE; dà anche l'ACK per le richieste OPEN
  - **NOTIFICATION**: riporta gli errori nei precedenti messaggi; usato anche per chiudere il collegamento

# Mettendo tutto insieme:

## *Come arriva una entry nella Forwarding Table di un router?*

- ❖ La risposta non è semplice!
- ❖ Mette insieme routing gerarchico (Sezione 4.5.3) con BGP (4.6.3) e OSPF (4.6.2).
- ❖ Fornisce una panoramica di BGP!

# Come arriva una entry nella Forwarding Table?

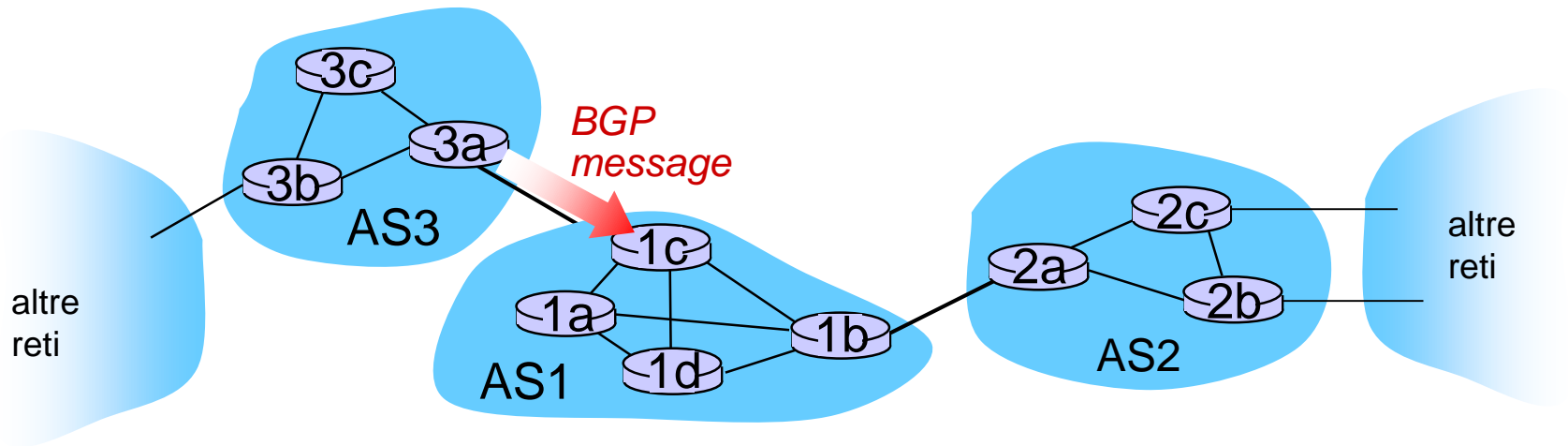


# Come arriva una entry nella Forwarding Table?

## Panoramica ad alto livello

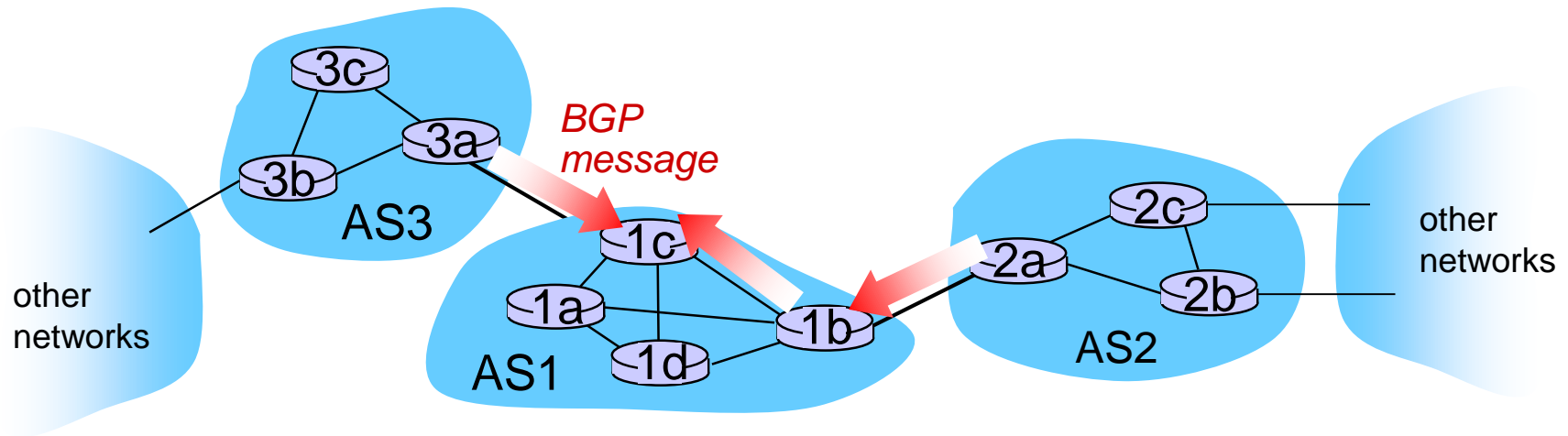
1. Il router acquisisce la conoscenza del prefisso
2. Il router determina l'interfaccia di output per quel prefisso
3. Il router inserisce la prefix-port nella forwarding table

# Il router acquisisce il prefisso



- ❖ il messaggio BGP contiene “rotte”
- ❖ la “rotta” è un prefisso più attributi: AS-PATH, NEXT-HOP,...
- ❖ Esempio: rotta:
  - ❖ Prefix: 138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

# Il router può ricevere rotte multiple



- ❖ Il router può ricevere rotte multiple per lo stesso prefisso
- ❖ Deve selezionare una rotta

# Migliore rotta BGP verso il prefisso

- ❖ Il router seleziona la rotta in base al percorso AS-PATH più breve

- ❖ Esempio:

- ❖ AS2 AS17 to 138.16.64/22

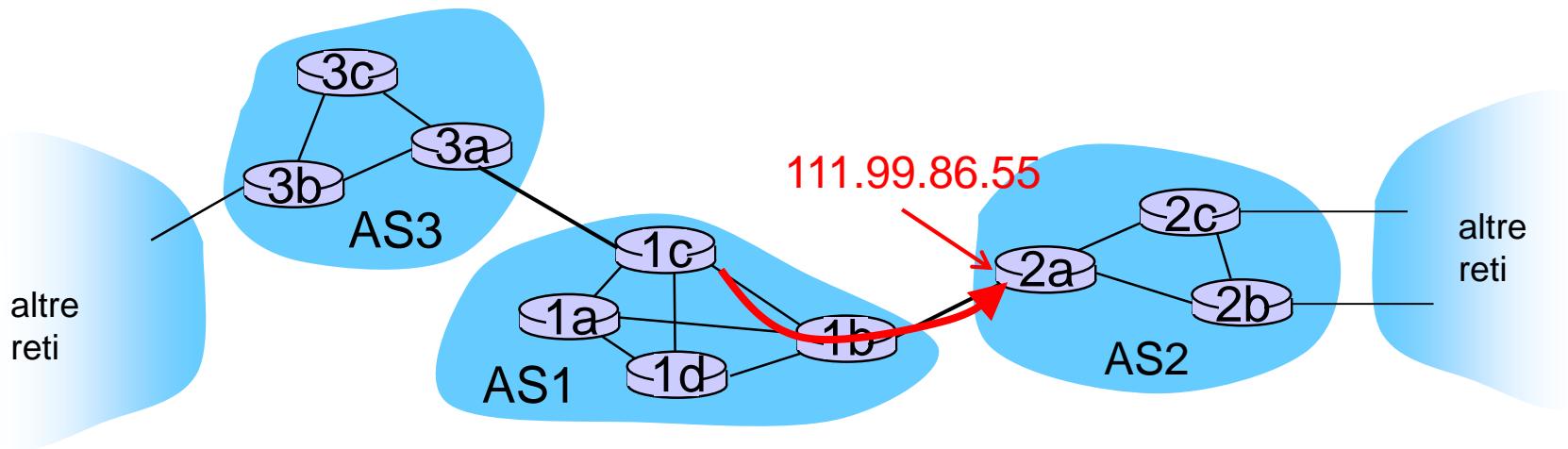
selezione

- ❖ AS3 AS131 AS201 to 138.16.64/22

- ❖ Se sono pari? Più avanti!

# Migliore intra-route BGP

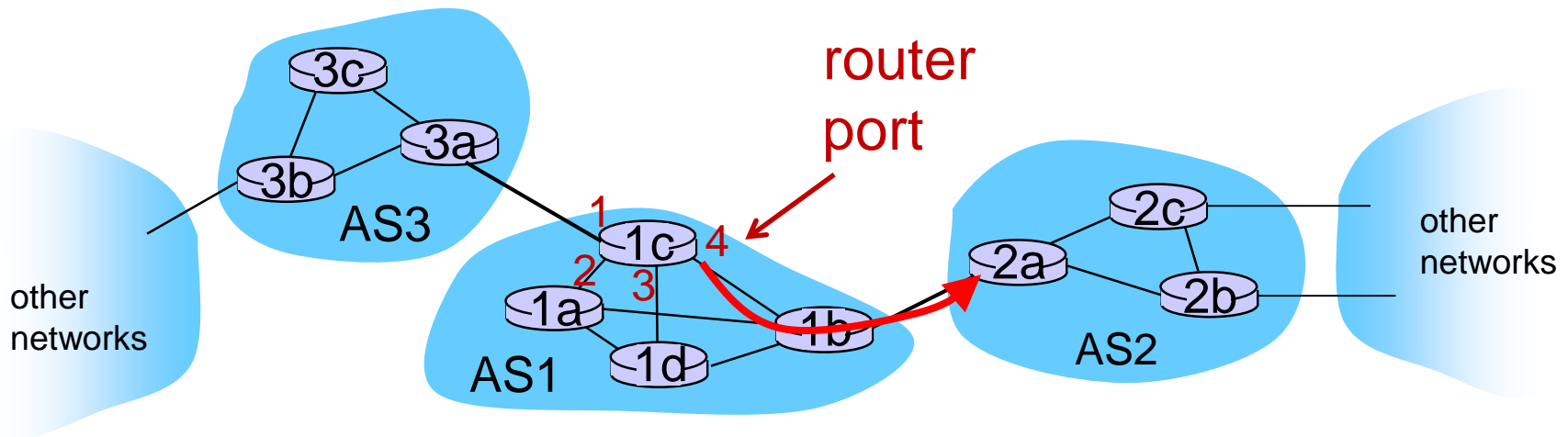
- ❖ Usa l'attributo NEXT-HOP della rotta selezionata
  - l'attributo NEXT-HOP della rotta è l'indirizzo IP dell'interfaccia del router con cui inizia l'AS PATH.
- ❖ Esempio:
  - ❖ AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- ❖ il router usa OSPF per trovare il cammino minimo da 1c a 111.99.86.55





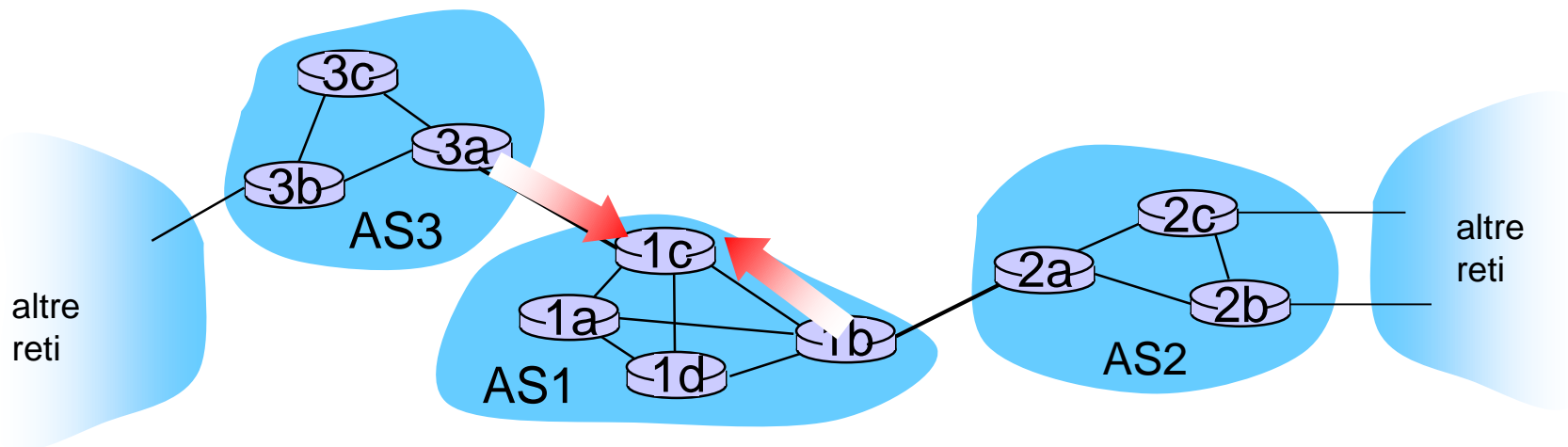
# Il router identifica l'uscita per la rotta

- ❖ identifica l'uscita lungo il cammino minimo OSPF
- ❖ aggiunge la riga prefix-port alla sua forwarding table:
  - (138.16.64/22 , port 4)



# Routing a 'Patata Bollente'

- ❖ supponiamo ci siano due o più migliori inter-route.
- ❖ allora sceglie la rotta con il NEXT-HOP più vicino
  - usa OSPF per determinare quale gateway è il più vicino
  - D: da 1c, si sceglie AS3 AS131 o AS2 AS17?
  - R: rotta AS3 AS201 perché è più vicina

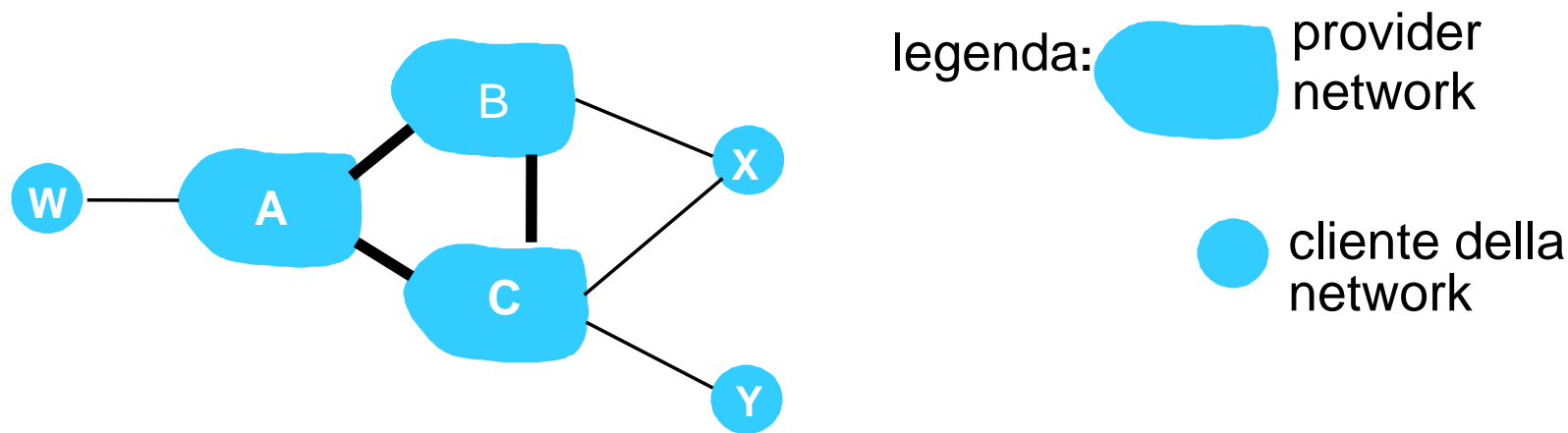


# Come arriva una entry nella Forwarding Table?

## riassunto

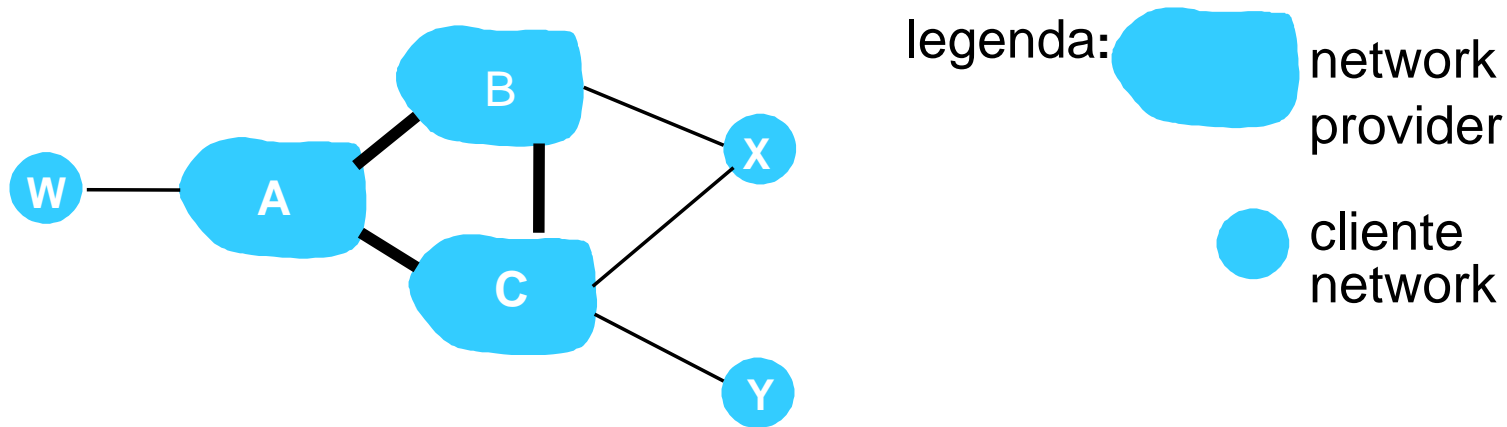
1. Il router acquisisce il prefisso
  - tramite un route advertisement BGP dagli altri router
2. Il router determina l'uscita per quel prefisso
  - in base ai criteri BGP trova la migliore rotta inter-AS
  - usa OSPF per trovare la migliore rotta intra-AS
  - il router identifica l'uscita del router verso quella rotta
3. Aggiunge la riga prefix-port nella forwarding table

# Politiche di routing BGP



- ❖ A,B,C sono *provider network (reti di provider)*
- ❖ X,W,Y sono client (dei provider network)
- ❖ X è *dual-homed*: collegato a due reti
  - X non vuole che si passi per esso per andare da B a C
  - ..allora X non annuncia a B una rotta verso C

# Politiche di routing BGP (2)



- ❖ A annuncia a B il percorso AW
- ❖ B annuncia a X il percorso BAW
- ❖ B deve annunciare a C del percorso BAW?
  - Certo che no! B non ha nessun “interesse” nella rotta CBAW poiché né W né C sono clienti di B
  - B vuole costringere C a passare attraverso A per andare verso W
  - B vuole traffico *solo* da/verso i suoi clienti!

# Perché routing Intra-AS e Inter-AS diversi?

## *politiche:*

- ❖ inter-AS: l'amministratore vuole il controllo su come il suo traffico viene instradato e su chi instrada attraverso le sue reti.
- ❖ intra-AS: amministratore unico, quindi non sono necessarie particolari politiche

## *scala:*

- ❖ il routing gerarchico riduce le dimensioni delle tabelle e il traffico sugli aggiornamenti

## *prestazioni:*

- ❖ intra-AS: orientato alle prestazioni
- ❖ inter-AS: le politiche possono prevalere sulle prestazioni

# Capitolo 4: livello di rete

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi
- indirizzamento IPv4
- ICMP
- IPv6

## 4.5 algoritmi di routing

- link state
- distance vector
- routing gerarchico

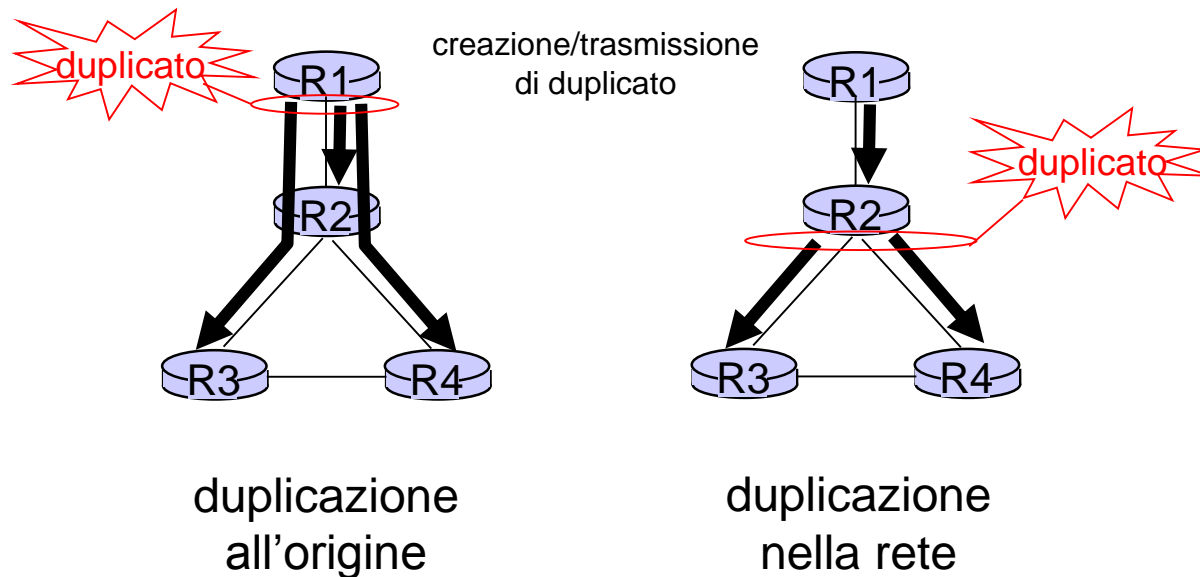
## 4.6 routing in Internet

- RIP
- OSPF
- BGP

## 4.7 routing broadcast e multicast

# Routing broadcast

- ❖ consegna pacchetti spediti da un'origine a tutti gli altri nodi della rete
- ❖ la duplicazione all'origine è inefficiente:



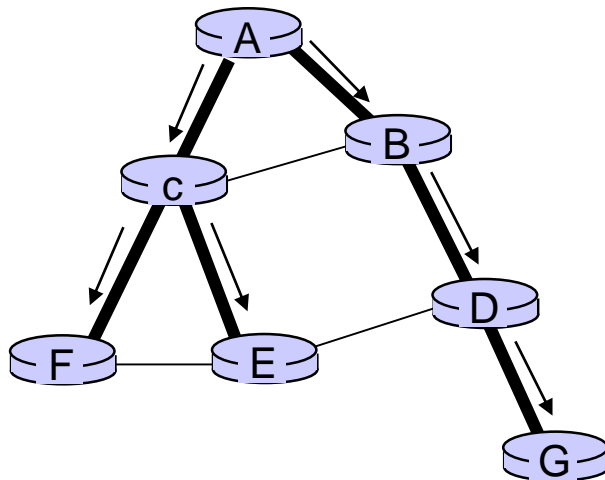


# Duplicazione nella rete

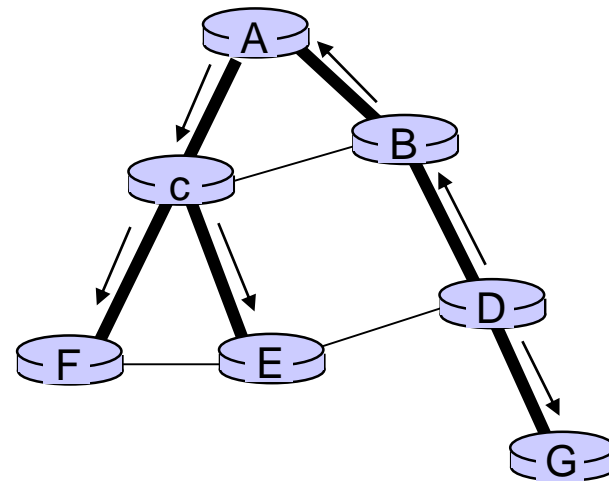
- ❖ *flooding*: quando un nodo riceve un pacchetto broadcast, ne inoltra una copia a tutti i propri vicini
  - problemi: cicli & broadcast storm (tempesta di broadcast)
- ❖ *flooding controllato*: il nodo manda un pacchetto in broadcasts solo se non lo ha già mandato
  - il node tiene traccia, tramite identificativi, dei pacchetti già broadcast-ati
  - oppure reverse path forwarding (RPF): inoltra il pacchetto solo se è arrivato dal cammino minimo tra nodo e sorgente
- ❖ *spanning tree (albero di copertura)*:
  - i nodi non ricevono pacchetti ridondanti

# Spanning tree

- ❖ si costruisce uno spanning tree
- ❖ i nodi inoltrano/mandano copie solo lungo l'albero



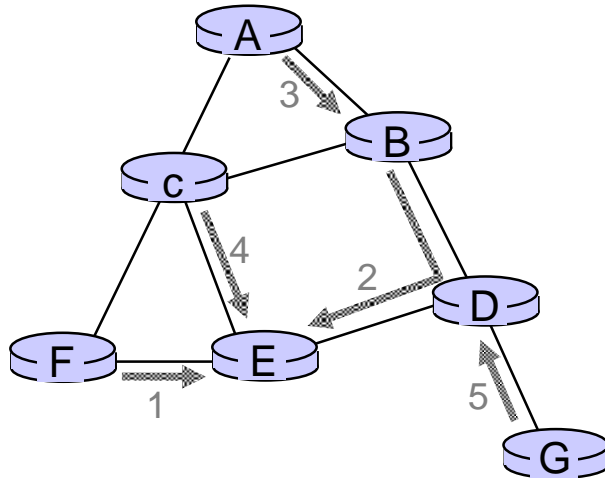
(a) broadcast iniziato da A



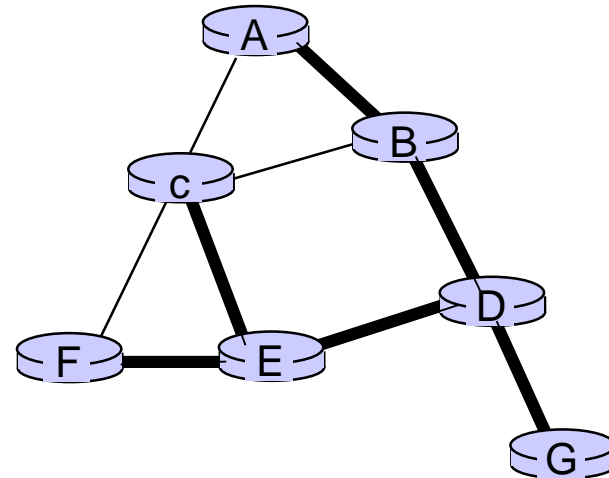
(b) broadcast iniziato da D

# Spanning tree: creazione

- ❖ si stabilisce un nodo centrale
- ❖ ogni nodo invia al nodo centrale un messaggio (unicast) di adesione
  - il messaggio viene inoltrato fino a quando raggiunge un router che già appartiene all'albero



(a) costruzione dello  
spanning tree (centro: E)



(b) spanning tree risultante

# Routing multicast

**obiettivo:** trovare un albero che colleghi tutti i router connessi ad host che appartengono al gruppo multicast

- ❖ **albero:** non tutti i cammini tra i router sono usati
- ❖ **shared-tree:** stesso albero per tutti i gruppi
- ❖ **source-based:** un albero diverso per ogni origine

*legenda*



*membro del gruppo*



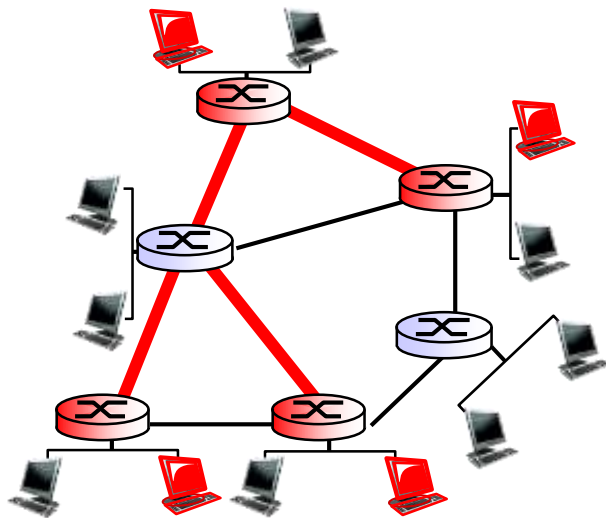
*non membro del gruppo*



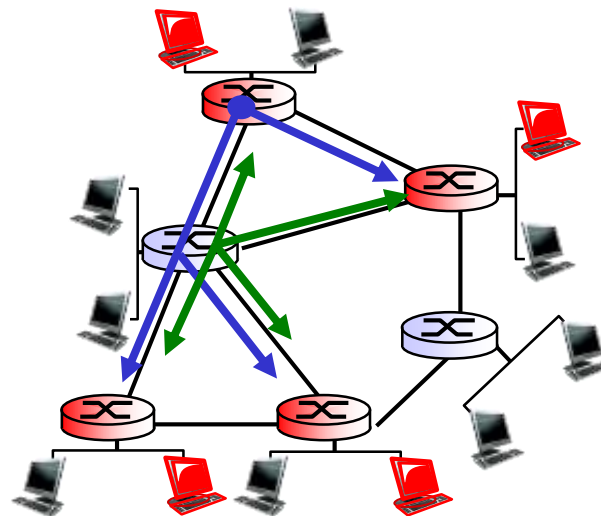
*router con un membro*



*router senza membri*



shared tree



alberi source-based

# Costruzione degli alberi multicast

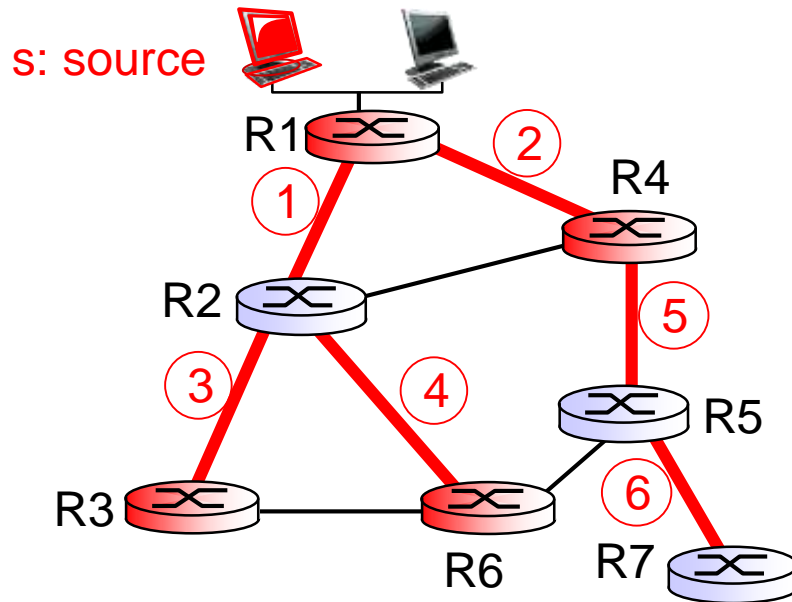
approcci:

- ❖ *source-based tree*: un albero per ciascuna origine
  - alberi dei cammini minimi
  - reverse path forwarding
- ❖ *group-shared tree*: il gruppo usa un albero
  - minimal spanning (Steiner)
  - basato su un nodo centrale

...diamo prima un'occhiata agli approcci, e poi agli specifici protocolli che adottano questi approcci.

# Shortest path tree

- ❖ albero di inoltra multicasting: albero con il percorso più breve dall'origine a tutti i destinatari
  - algoritmo di Dijkstra



## LEGENDA



router con membro di gruppo collegato



router senza membri di gruppo collegati



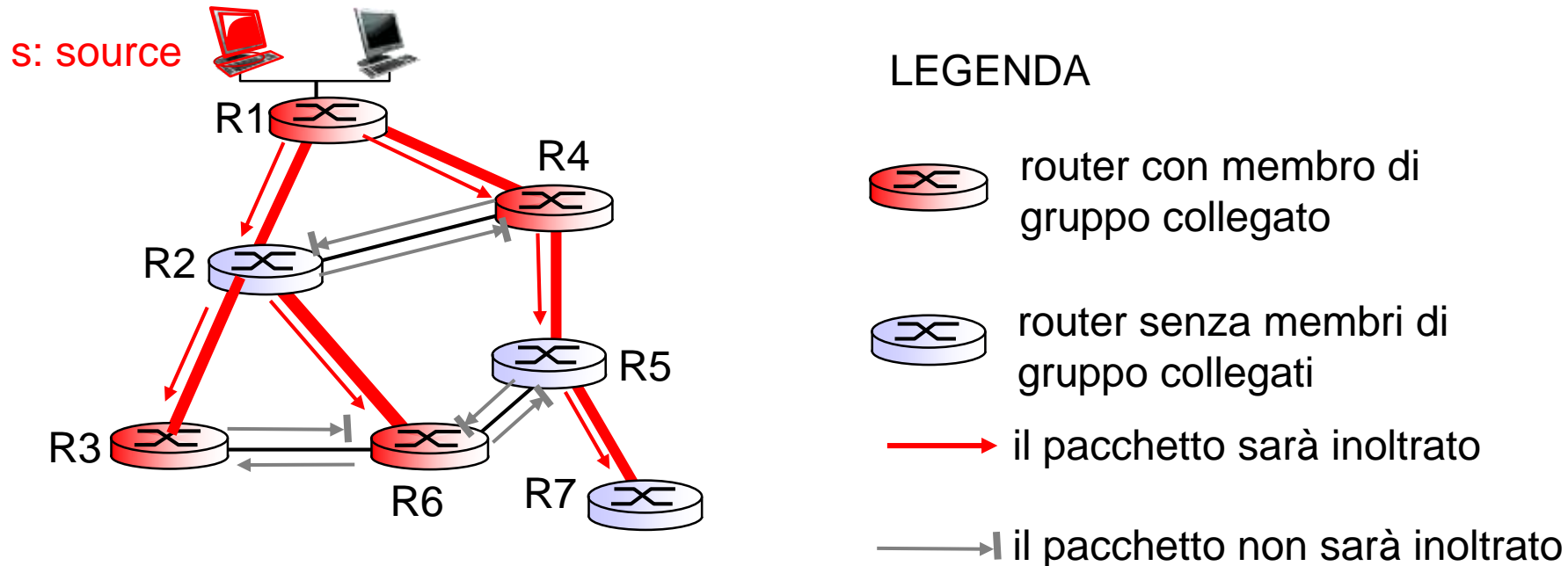
link usato per l'inoltro, i indica l'ordine del link indicato dall'algoritmo

# Reverse path forwarding

- ❖ si basa sul presupposto che il router conosca il percorso unicast più breve da esso verso il mittente
- ❖ ogni router si comporta secondo questo semplice schema:

***if*** (il pacchetto multicast è pervenuto attraverso il percorso unicast più breve tra il router e l'origine)  
***then*** lo trasmette su tutti i propri collegamenti in uscita  
***else*** ignora il pacchetto

# Reverse path forwarding: esempio

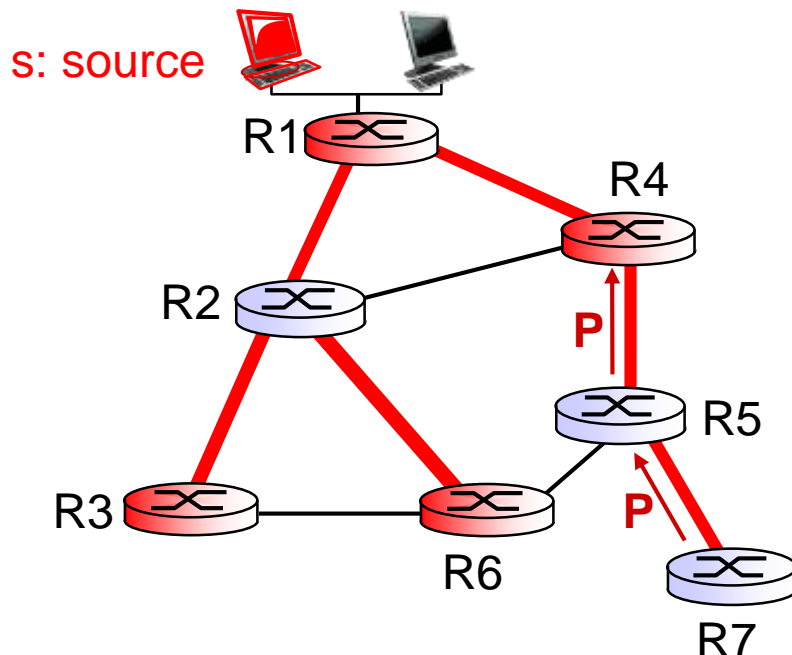


- ❖ Il risultato è un *reverse SPT* specifico per un'origine
  - può essere una cattiva scelta con link asimmetrici







# Reverse path forwarding: pruning

- ❖ l'albero può contenere sottoalberi senza membri di gruppi multicast
  - non è necessario l'inoltro verso il sottoalbero
  - un messaggio di "prune" (potatura) viene inviato dal router senza membri



## LEGENDA

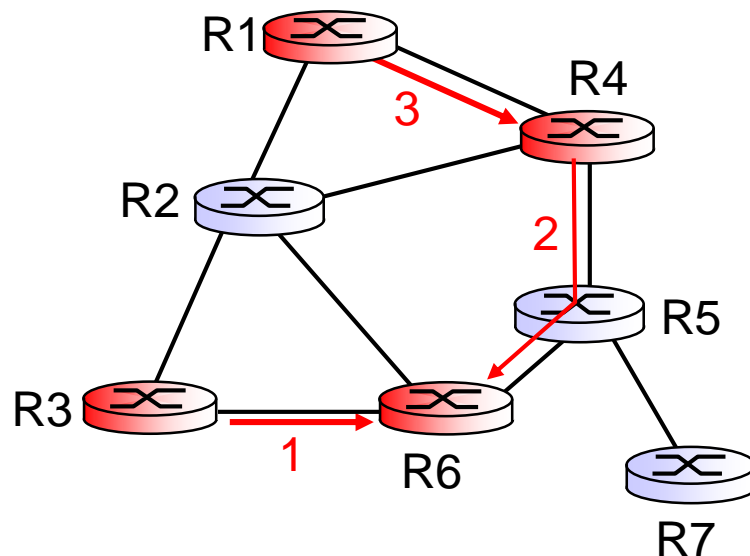
-  router con membro di gruppo collegato
-  router senza membri di gruppo collegati
-  messaggio di prune
-  link con inoltro multicast

# Alberi center-based




- ❖ singolo albero di inoltra condiviso da tutti
- ❖ un router viene identificato come “*centro*” dell'albero
- ❖ per unirsi:
  - un edge router invia un *messaggio join* indirizzato al router centrale
  - il *join-msg* viene “processato” dai router intermedi e inoltrato verso il centro
  - il *join-msg* o arriva a una parte esistente dell'albero di quel centro o arriva al centro
  - il percorso fatto dal *join-msg* diventa un nuovo ramo dell'albero per il router

# Alberi center-based: esempio

supponiamo che R6 sia scelto come centro:



## LEGENDA

-  router con membro di gruppo collegato
-  router senza membri di gruppo collegati
-  path ordine nel cammino preso dal join messages

# Routing multicast in Internet : DVMRP

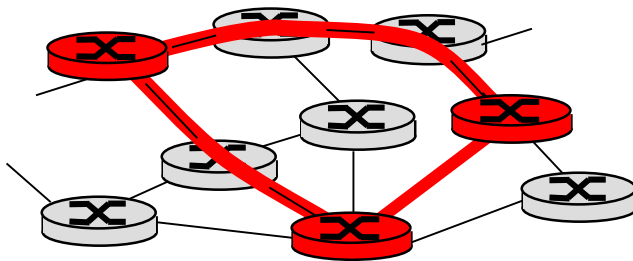
- ❖ **DVMRP**: distance vector multicast routing protocol, RFC1075
- ❖ ***flood and prune***: alberi source-based con reverse path forwarding
  - gli alberi RPF sono basati sulle tabelle di routing DVMRP costruite dai router DVMRP
  - nessuna assunzione sugli unicast
  - i datagrammi iniziali verso il gruppo multicast diffuso (flooded) ovunque tramite RPF
  - i router che non vogliono gruppi: inviano un messaggio di prune

# DVMRP: continua...

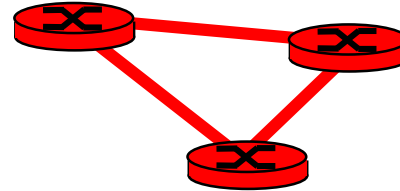
- ❖ *soft state*: i router DVMRP periodicamente (1 min.) “dimenticano” i rami pruned:
  - i dati multicast rifluiscono verso i rami unpruned
  - downstream router: reprune oppure continuano a ricevere dati
- ❖ i router possono rapidamente riunirsi all'albero
  - a seguito di join IGMP da parte di qualche foglia

# Tunneling

**D:** come connettere “isole” di router multicast in un “mare” di router unicast?



topologia fisica



topologia logica

- ❖ i datagrammi multicast sono incapsulati dentro datagrammi “normali” (indirizzi non-multicast)
- ❖ i datagrammi IP normali vengono inviati tramite “tunnel” attraverso regolari IP unicast verso i router multicast riceventi (come tunneling di IPv6 dentro IPv4)
- ❖ i router multicast disincapsula per ottenere i datagrammi multicast

# PIM: Protocol Independent Multicast

- ❖ non dipende da nessun particolare algoritmo di routing unicast sottostante (funziona con tutti)
- ❖ prende in considerazione due scenari di distribuzione multicast:

## *dense:*

- ❖ i membri del gruppo multicast sono concentrati in una area.
- ❖ bandwidth abbondante

## *sparse:*

- ❖ numero di reti con membri di gruppi piccolo rispetto al numero di reti
- ❖ membri di gruppi “sparsi qua e là”
- ❖ bandwidth non abbondante

# Conseguenze della situazione sparse-dense:

## *dense*

- ❖ appartenenza dei router ai gruppi *assodata* fino a che i router mandano esplicitamente un segnale di prune
- ❖ costruzione dell'albero multicast *guidata-dai-dati* (es., RPF)
- ❖ bandwidth e capacità di calcolo dei router non di gruppo *sprecate*

## *sparse:*

- ❖ nessuna membership fino a che i routers chiedono esplicitamente il join
- ❖ costruzione dell'albero multicast *guidata-dai-riceventi* (es., center-based)
- ❖ bandwidth e capacità di calcolo dei router non di gruppo *conservate*



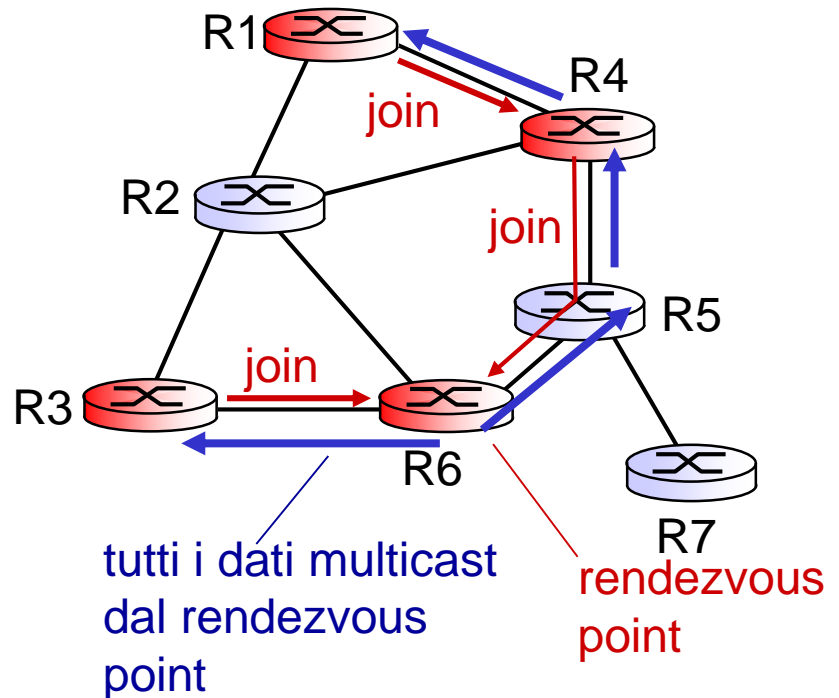
# PIM- dense mode

**flood-and-prune RPF:** simile a DVMRP ma...

- ❖ il protocollo unicast sottostante fornisce informazioni RPF per i datagrammi in arrivo
- ❖ il flood meno complicato (meno efficiente) rispetto a DVMRP riduce la dipendenza protocollo di routing sottostante
- ❖ il protocollo ha dei meccanismi per far capire a un router se è un nodo foglia

# PIM - sparse mode

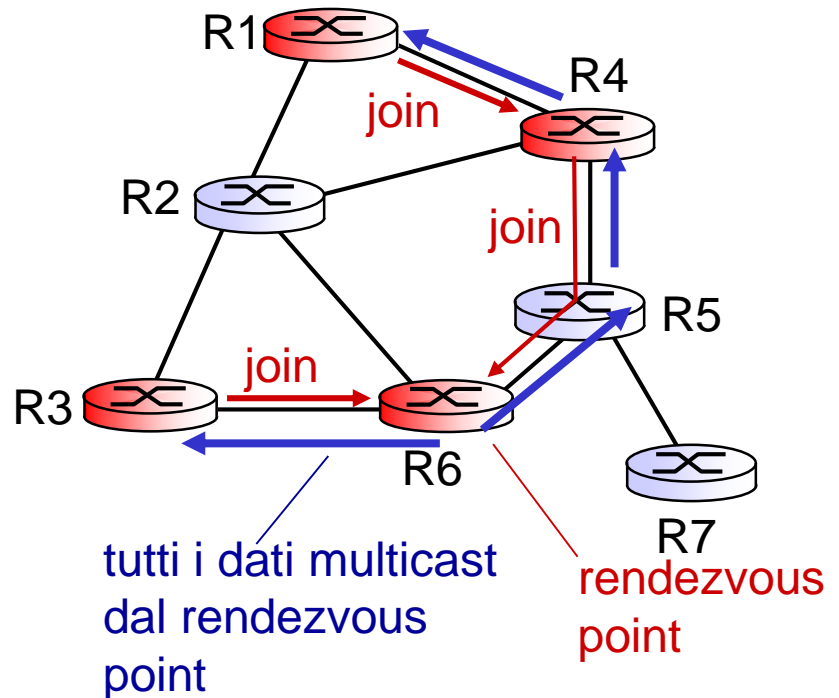
- ❖ approccio center-based
- ❖ il router invia il messaggio *join* al rendezvous point (RP)
  - i router intermedi aggiornano lo stato e inoltrano i *join*
- ❖ dopo il joining via RP, il router può unirsi a un albero con una specifica sorgente
  - performance migliorata: meno concentrazione, cammini minimi



# PIM - sparse mode

## *sorgente(i):*

- ❖ invia dati unicast al RP, che ridistribuisce lungo alberi con il RP come radice
- ❖ RP può estendere l'albero multicast su fino alla sorgente
- ❖ RP può inviare messaggi di *stop* se non ci sono ricevitori
  - “nessuno all’ascolto!”



# Capitolo 4: *fatto!*

## 4.1 introduzione

## 4.2 reti a circuito virtuale e a datagramma

## 4.3 cosa si trova all'interno di un router

## 4.4 IP: Internet Protocol

- formato dei datagrammi, indirizzamento IPv4, ICMP, IPv6

- ❖ capire i principi che stanno dietro i servizi del livello di rete:
  - modelli di servizio del livello di rete, forwarding e routing, come lavora un router, routing (scelta del percorso), broadcast, multicast
- ❖ implementazione in Internet

## 4.5 algoritmi di routing

- link state, distance vector, routing gerarchico

## 4.6 routing in Internet

- RIP, OSPF, BGP

## 4.7 routing broadcast e multicast