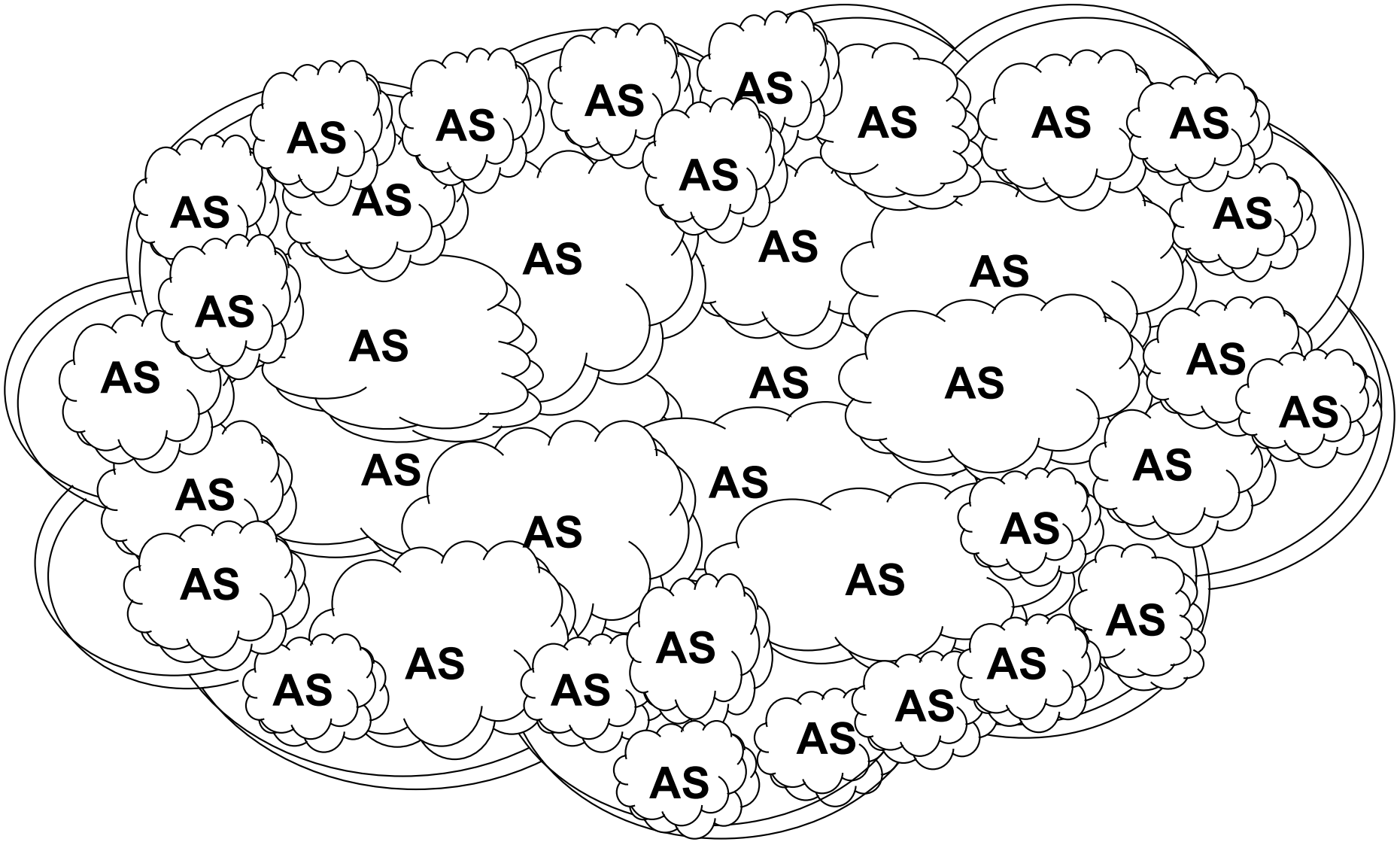
Cos'è INTERNET (4)?

DAL PUNTO DI VISTA ORGANIZZATIVO:

Un insieme di oltre 50000 Autonomous Systems alcuni su scala nazionale altri su scala continentale e intercontinentale



Autonomous Systems

- Internet non è un insieme di router "sparsi" nel mondo che sono interconnessi tra di loro in modo casuale

→ I router sono aggregati in ***regioni***, chiamate **Autonomous Systems** (**AS**):

"Un insieme di reti, di indirizzi IP (*network prefix*) e di router sotto il controllo di una organizzazione (o consorzio di) nell'ambito del quale si utilizza una politica di *interior routing*. Gli AS sono le unità delle politiche di *exterior routing*, come nel caso del BGP"
[RFC 1930]

Situazione degli Autonomous Systems (AS)

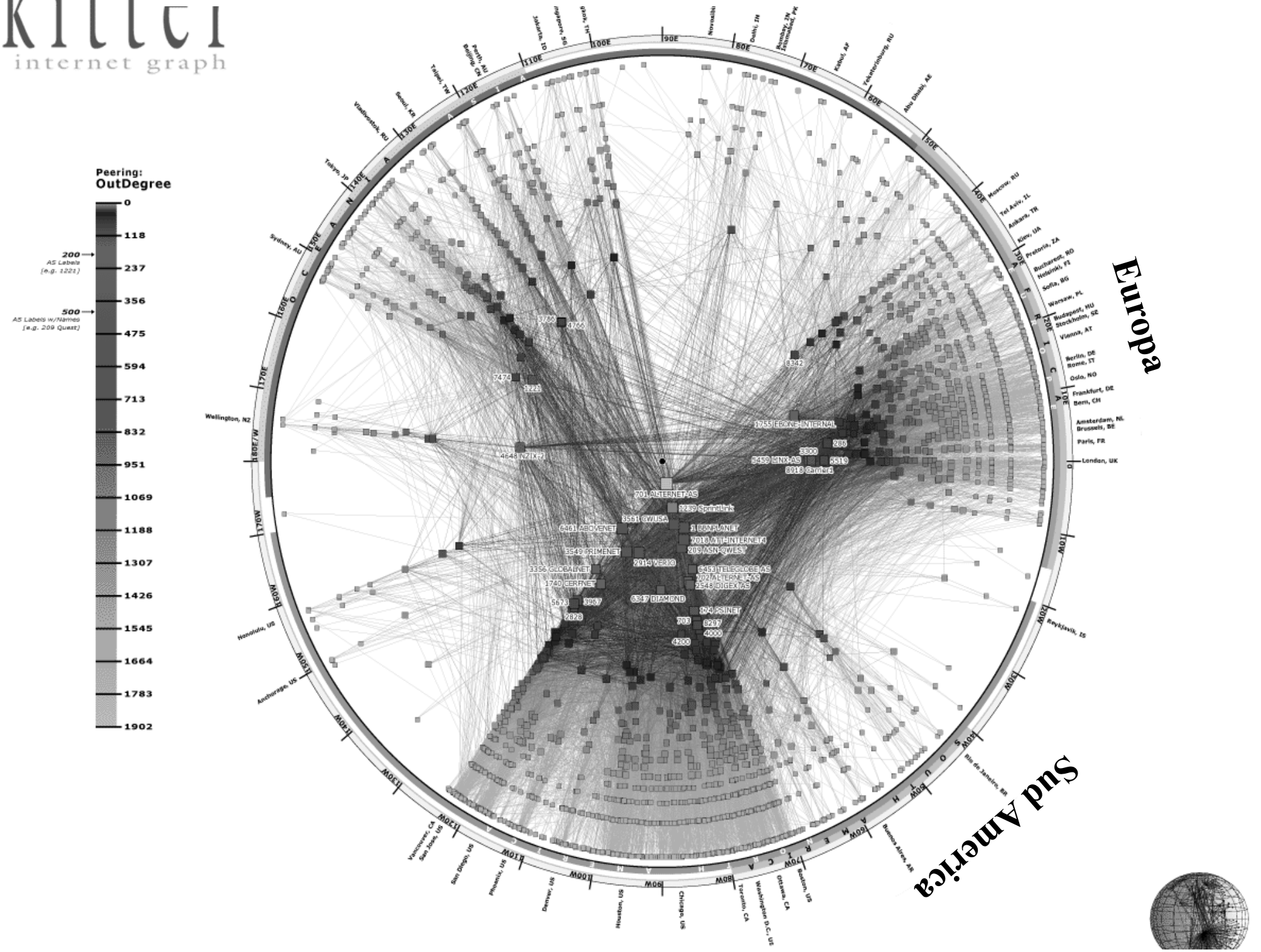
- Il traffico Internet si distribuisce tra ***Autonomous Systems***
- Ciascun AS è caratterizzato da un **numero identificativo** chiamato **Autonomous System Number (ASN)**
  - assegnato da **IANA-ICANN**
  - In origine (ancora in uso, in via di deprecazione) **2 byte**
  - Transizione verso **4 byte**
- Ciasun AS controlla in modo esclusivo uno o più network prefix (es., 120.240.0.0/16)

NOTA IMPORTANTE:

- **Nessun AS gestisce più del 5% del traffico**
- **La stragrande maggioranza degli AS gestisce molto meno dell'1% del traffico**

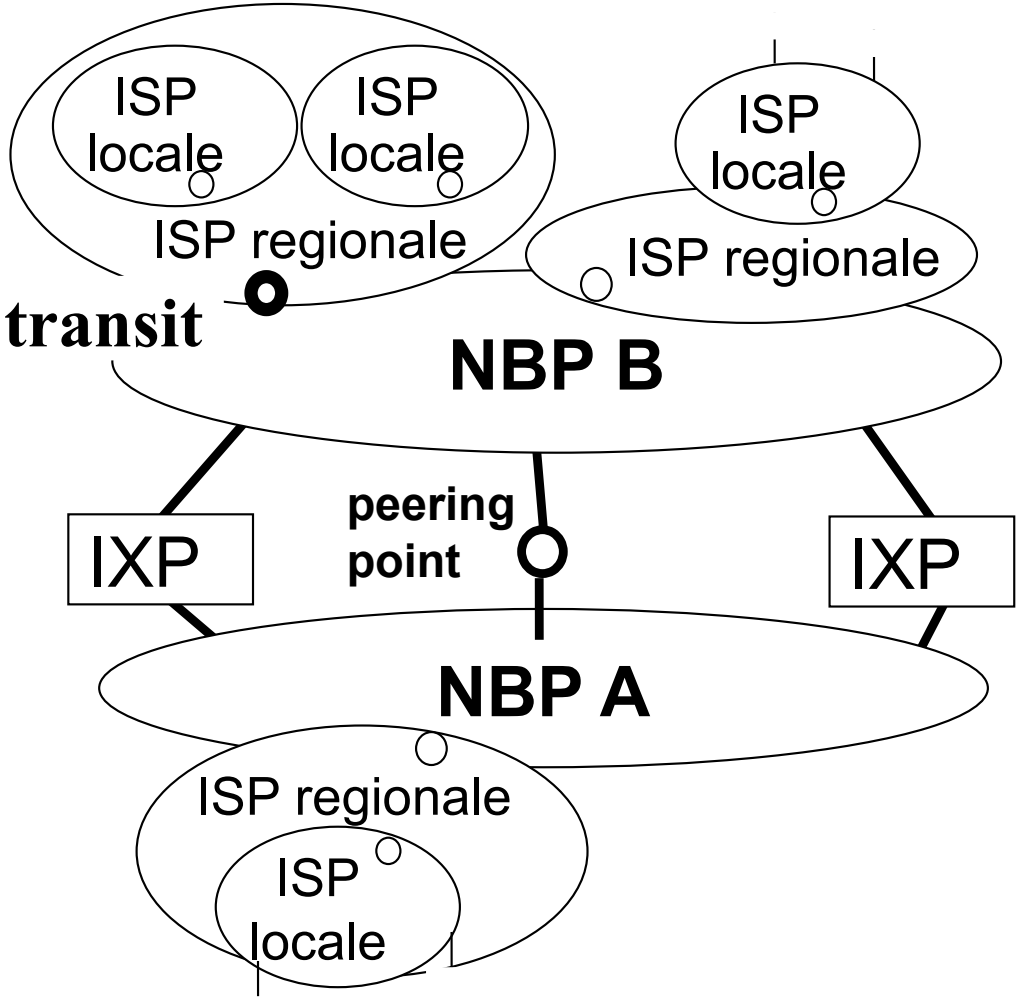
INTERCONNESSIONI TRA AUTONOMOUS SYSTEMS

skitter
AS internet graph



Interconnessioni tra AS [1]

Gli ISP “regionali” (nazionali) e internazionali sono collegati tra di loro al più alto livello della gerarchia, mediante **peering point** (privati) oppure mediante **Internet Exchange Point** (**IXP** o IX), una volta chiamati **Network Access Point** (**NAP**)

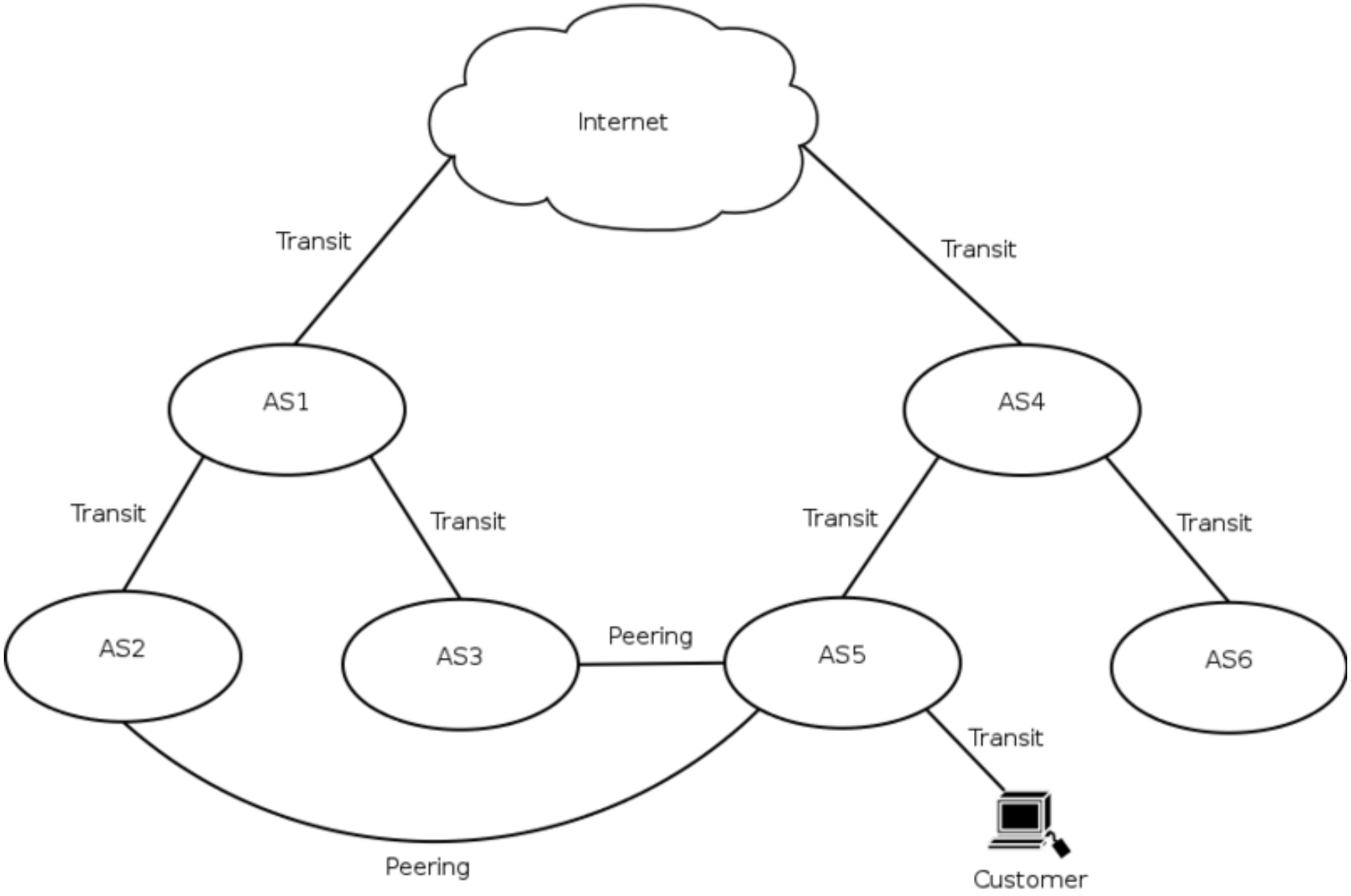


Interconnessioni tra AS [2]

- **Transit**: un ***AS pay*** paga un altro ***AS sell*** per avere accesso o transito su Internet; si accetta traffico interno ma anche traffico esterno in transito

- **Peering** (o *swap*): Due AS si scambiano il traffico dei rispettivi utenti senza costi, per reciproco interesse:
  - non solo per evitare costi, ma anche per aumentare affidabilità creando strade alternative, e per diminuire la lunghezza dei percorsi creandone uno diretto

Esempio


Internet
Transit
Transit
AS1
AS4
Transit
Transit
Transit
Transit
AS2
AS3
Peering
AS5
AS6
Transit
Peering
Customer

Internet Exchange Point (IXP o IX)

- Tipicamente consorzi indipendenti senza scopo di lucro
- Creati fra AS, talvolta supportati da finanziamenti pubblici
  - Offrono servizio tra gli associati, ma anche ad altri
- Spesso sono ***Metropolitan Area Exchange*** (MAE)
  - Predisposti in luoghi con altissima interconnettività disponibilità, ovvero spesso aree metropolitan
- Vedere ad esempio:
  - IXP GARR
  - NAMEX: Rome Internet Exchange Point
  - MIX: Milan Internet Exchange Point

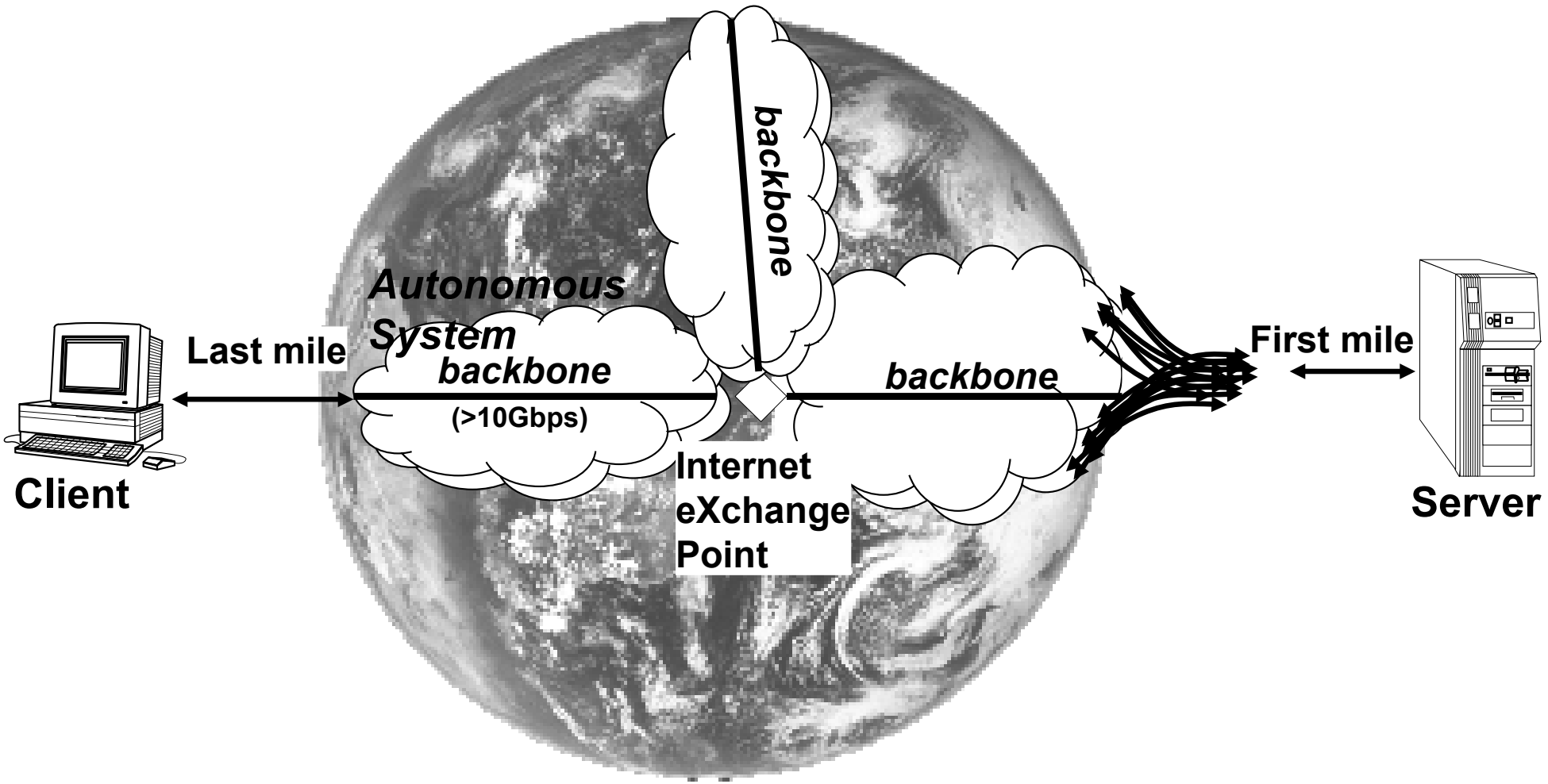
Alcuni IX

LINX (Londra)

AMS-IX
(Amsterdam)

NYIIX (New York)

Sintesi: *first mile*, *peering point*, *last mile*



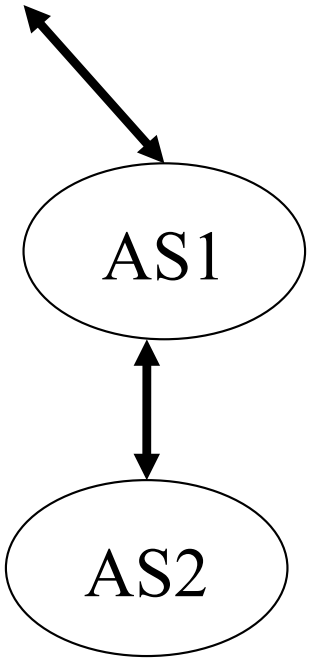
Protocolli di routing su scala geografica
(e in reti aziendali di grandi dimensioni)

Routing gerarchico (e autonomo)

- Nella realtà, i router di Internet:
  - **non rappresentano un insieme omogeneo di risorse**
  - **non eseguono lo stesso algoritmo di routing**
- Diversi motivi:
  - **Scalabilità**: all'aumentare del numero di router, se tutti dovessero essere considerati per il routing, il costo degli algoritmi di routing diventerebbe proibitivo

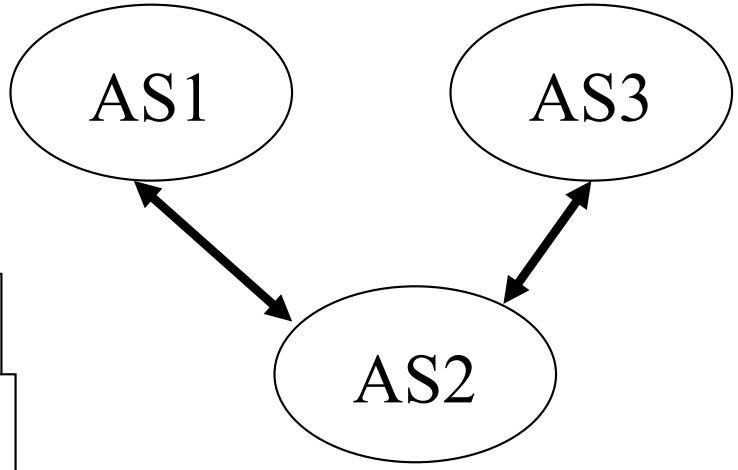
    ➔ occorre ridurre la complessità del calcolo del cammino
  - **Autonomia amministrativa**: un'organizzazione definita "AS" dovrebbe/vorrebbe scegliere autonomamente come amministrare il traffico tra i propri router e le proprie reti

Tipi di AS



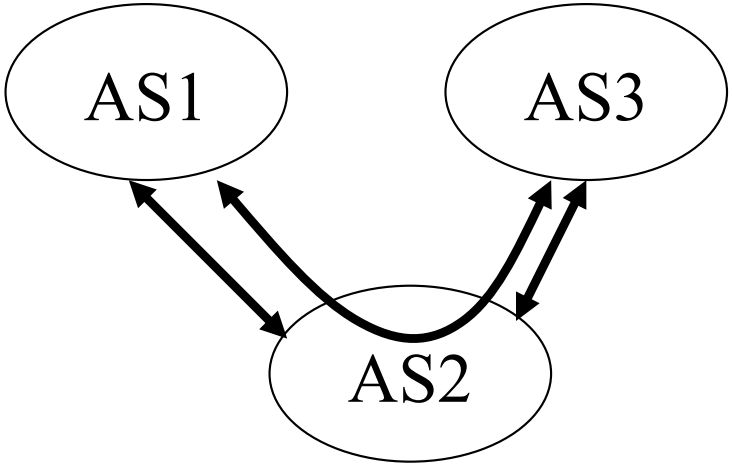
AS2: **Stub**

AS con una sola connessione ad un altro AS



AS2: **Multi-homed**

AS connesso con diversi AS, ma non permette traffico che non è generato o diretto verso l'AS (accetta solo traffico locale)



AS2: **Transit**

AS connesso a diversi AS che consente di fare da tramite per gli AS a cui è collegato (accetta sia traffico locale sia traffico in transito)

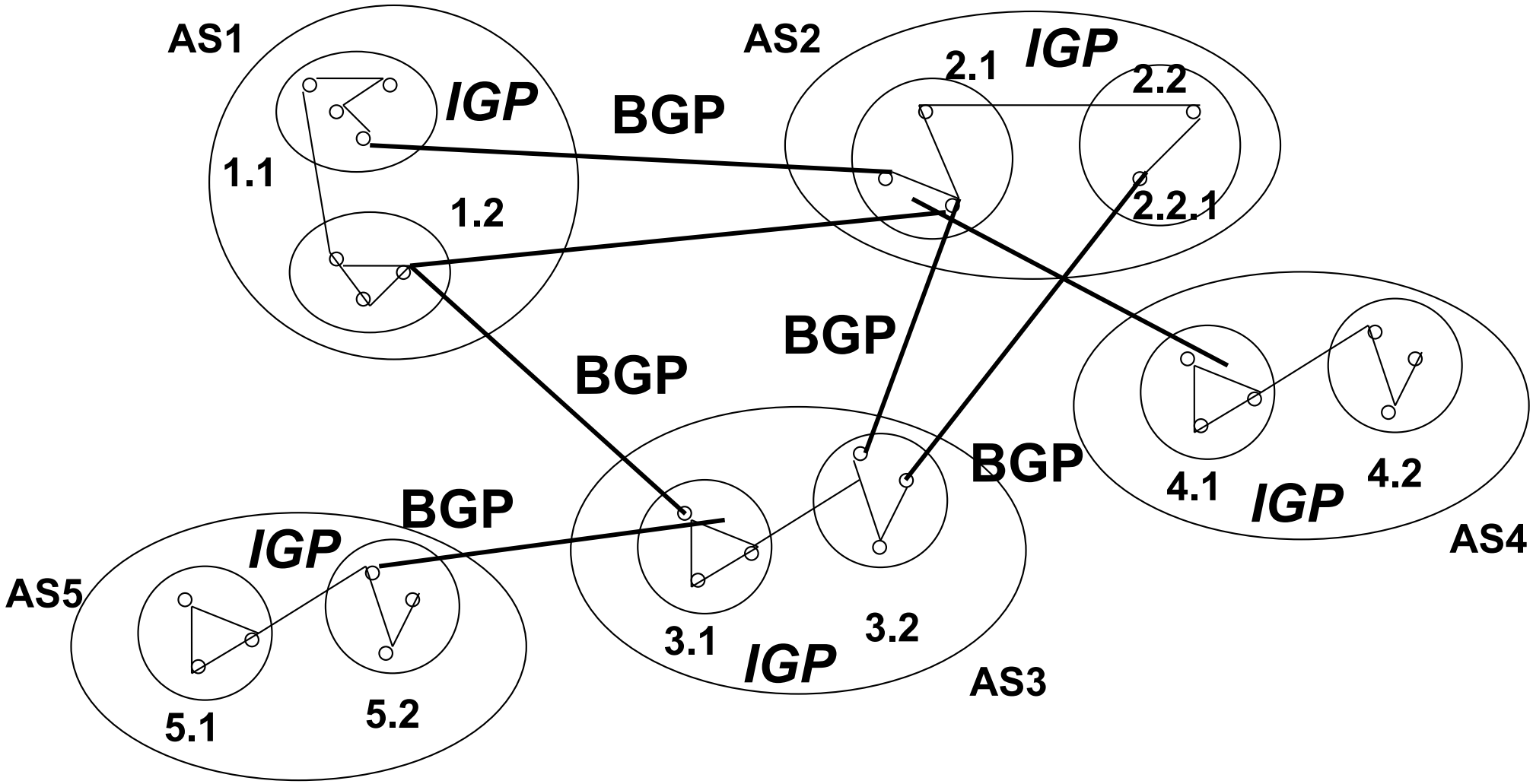
Autonomous Systems per il routing

- I router vengono aggregati in “regioni” o **sistemi autonomi** (**AS**)
- In pratica, un AS è un insieme di nodi e router gestiti da **un’unica entità di controllo centrale** (es., stesso ISP)
- Ciascun AS ha un **numero identificativo** assegnato da una **authority** di registrazione Internet:
  - Il numero è compreso fra 1 e 64511
  - I numeri di AS compresi nell’intervallo 64512-65535 sono riservati

Autonomous Systems per il routing (2)

- Per il routing all'interno di un AS (***routing Intra-AS***) i router utilizzano qualche ***Interior Gateway Protocol*** (**IGP**) dove i router di un AS possono possedere un'informazione completa su tutti gli altri router dell'AS
- Per il routing verso altri AS (***routing Inter-AS***) viene utilizzato qualche ***Exterior Gateway Protocol*** (prima EGP, oggi **BGP**)
- Ciascun AS può usare metriche multiple per il routing interno, **ma appare come un unico AS ad altri AS**

Esempio con 5 AS



Politiche di routing negli AS

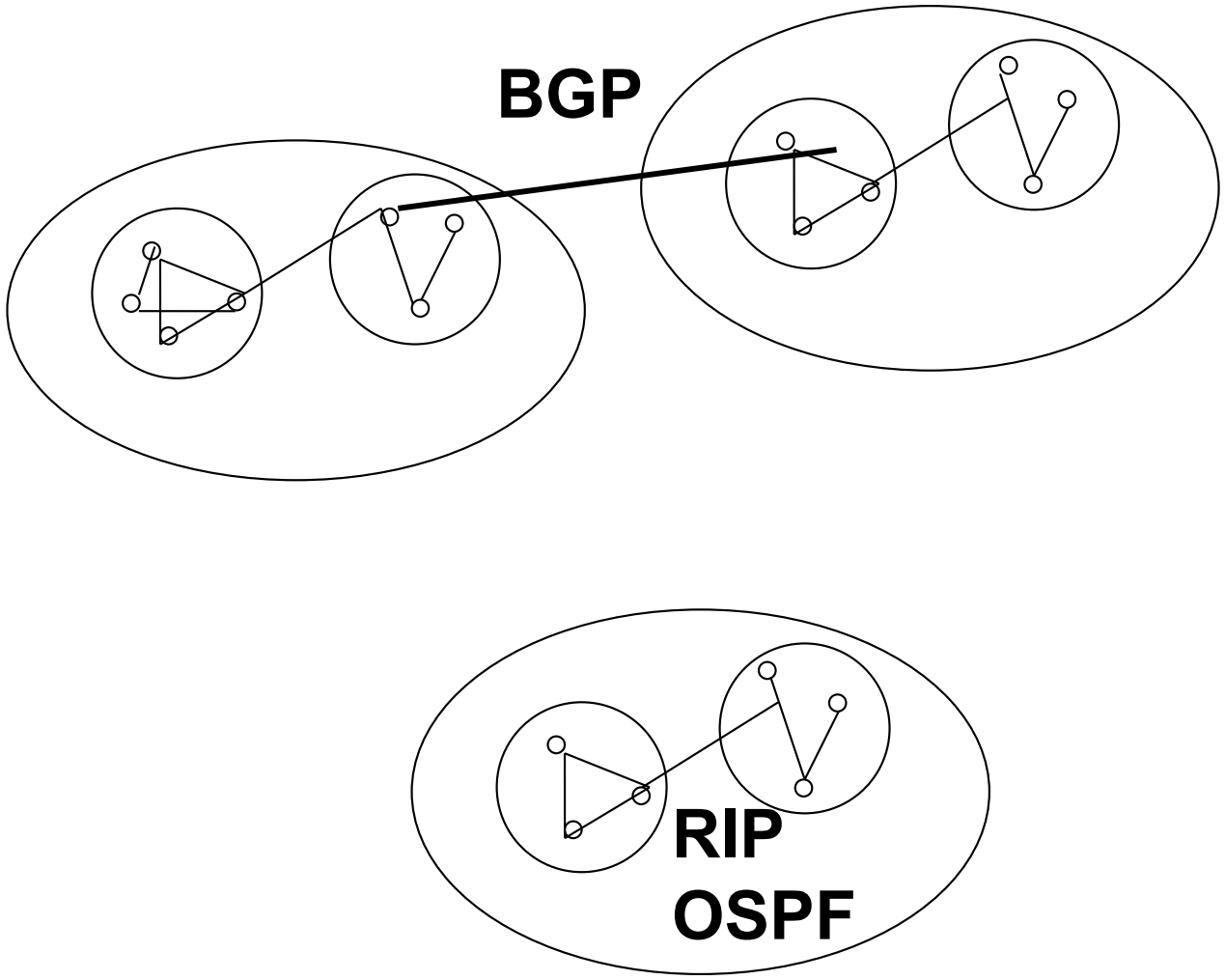
- Le **politiche di routing** sono le regole per decidere come instradare il traffico
  - Se sono un AS, quale traffico accetto di far passare attraverso la mia rete?
  - Ci sono spesso accordi commerciali alla base di queste decisioni (**RICORDARE: *peering point* e IXP**)

- Ogni AS vuole poter decidere le proprie politiche e potrebbe anche non volere farle conoscere agli altri AS

Protocolli di routing: intra-AS, inter-AS

1. **Principali protocolli di routing intra-AS**
   - **Routing Information Protocol (RIP)**
     - Routing distribuito (*Distance vector*)
   - **Open Shortest Path First (OSPF)**
     - Routing centralizzato (*Link state*)
   - **Enhanced Interior Gateway Routing Protocol (EIGRP)**
     - Routing distribuito (algoritmo proprietario CISCO, recente tentativo di trasformarlo in standard open)
2. **Principale protocollo di routing inter-AS**
   - **Border Gateway Protocol (BGP)**
     - Algoritmo di routing distribuito
     - E' oggi lo *standard de facto* per il routing tra diversi AS

Protocolli di routing: inter-AS, intra-AS



Autonomous Systems e Indirizzi IP

- Nominato nella slide prima e nelle slide precedenti, ma fondamentale: **Ogni AS gestisce un insieme di blocchi di indirizzi IP in modo esclusivo, oggigiorno definiti in notazione classless e chiamati "network prefix"**

Autonomous Systems e routing [1]

- **Tutti i router all'interno dello stesso AS usano lo stesso algoritmo di *instradamento dei messaggi* (*routing*) e si scambiano continue informazioni con gli altri router**
- **Gli Autonoums Systems dall'esterno vengono visti come un'unica entità**

Autonomous Systems e routing [2]

➔ **Gerarchia architettura Internet: 2 livelli**

1. Se l'IP destinazione è all'interno del suo stesso AS, routing sulla base di
2. Altrimenti, cercherà di inviarlo verso sono dei router speciali [all'interno dell'AS] che mantengono mappe fra Network Prefix e AS, e tabelle di routing in cui i percorsi sono espressi in base a ASN
   - Il routing in questo caso è definito sulla base di ASN (autonomous system number): consideriamo una rete astratta in cui i nodi gli AS, non i router
     - Vedere BGP in fondo a questo blocco di slide

Basi teoriche per il routing

Algoritmi di routing

OBIETTIVO:

Dato un insieme di router interconnessi, determinare il cammino *ottimale* dal *source router* al *destination router*

Cammino ottimale → "Costo" minimo

Cosa si intende per “costo”

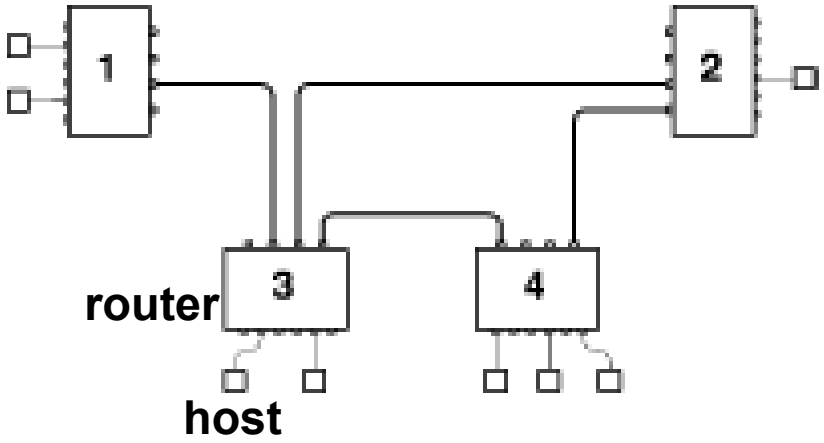
- ***Fattori statici***: topologia della rete (per esempio, numero di hop tra mittente e destinatario), banda del link (senza tener conto del traffico)
- ***Fattori dinamici***: traffico della rete, guasti (per es., in termini di *bit error rate*), carico dei router (per es., utilizzazione della CPU o numero di pacchetti gestiti al secondo)
- ***Costi economici***: accordi tra Autonomous Systems
- ***Tipi di traffico***: ***Violazione della Net neutrality!***
- ***Combinazioni (di alcuni) dei costi precedenti***

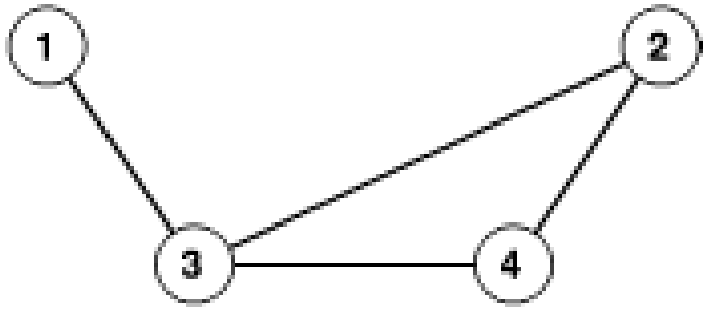
Modello della rete

Per formulare un algoritmo di routing, si modella la rete tramite un **grafo pesato G(N,E)** dove:

- i nodi ***N*** rappresentano i router (oppure gli AS)
- gli archi rappresentano le connessioni tra i router
- le etichette ***E*** degli archi rappresentano il "costo" delle connessioni tra i router

Rappresentazione mediante grafo (interessante esempio di modellazione)

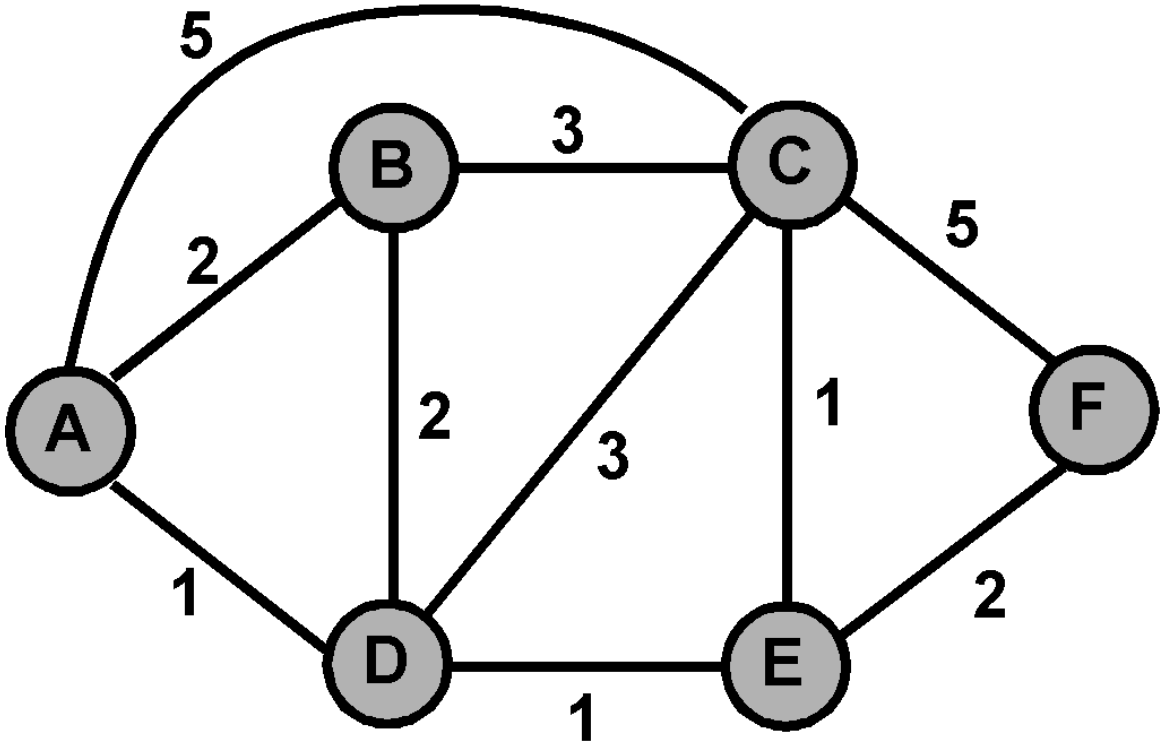




+Etichetta sugli archi:

"costi" per l'invio di un pacchetto

Cammini minimi



Qual è il cammino minimo (e il suo costo) tra A e C?

Qual è il cammino minimo (e il suo costo) tra A e F?

Quanti percorsi ci sono tra A e F?

Classi di algoritmi di routing

- **Algoritmi di routing globale**
  - Ogni nodo offre a tutti gli altri nodi la sua visione sui link della rete
  - Es., ***Link state protocol***
- **Algoritmi di routing locale (*distribuito*)**
  - Ogni nodo comunica ai suoi vicini la sua visione del costo di trasmissione dei link
  - Es., ***Distance vector protocol***

Link state protocol

Algoritmi di tipo Link State

Gli algoritmi **Link State** (**LS**) sono globali cioé prevedono che la **topologia di rete** e i **costi** di ogni link siano noti a tutti (disponibili in input all'algoritmo):

1. Ogni nodo calcola lo stato dei link ad esso connessi
2. Ciascun nodo **periodicamente** trasmette identità e costi dei suoi link (***link state broadcast***)

   (Quindi tutti i nodi hanno una **visione completa e identica –*salvo i ritardi di trasmissione*–** della rete)
3. Ciascun nodo calcola i **cammini di costo minimo** verso tutti gli altri nodi della rete mediante l'**Algoritmo di Dijkstra**

Pacchetti con informazioni sullo stato dei link (Link State Protocol - LSP)

Periodicamente vengono inviati in broadcast, su tutti i link del nodo, dei **pacchetti LSP** con le seguenti informazioni:

- **Node ID**
- **Lista dei vicini** e **costo** dei rispettivi **link**
- **Informazioni aggiuntive:**
  - **Numero di sequenza** per accorgesi di errori in caso di delivery out-of-order delle informazioni
  - **Time-To-Live** (TTL) per evitare di usare informazioni vecchie e quindi non affidabili

Propagazione dei pacchetti LSP

Inoltro con un algoritmo di **flooding** (***inondazione***)

Quando il nodo ***i*** riceve un pacchetto LSP dal nodo ***j***:

- Se il pacchetto LSP proveniente da ***j*** è valido (TTL non scaduto e numero di sequenza successivo all'ultimo ricevuto):
  - salva il valore nella tabella di routing
  - inoltra una copia del pacchetto LSP su tutti i link connessi al nodo ***i*** (ad eccezione del link da cui il pacchetto LSP è stato ricevuto)
- Altrimenti scarta il pacchetto LSP

“Forward search algorithm” di Dijkstra

- **Già visto in materie precedenti del Corso di Studio, ricordiamo funzionamento intuitivamente**
- Algoritmo **iterativo**: alla *k*-esima iterazione, il nodo *i* conosce il cammino di costo minore verso *k* nodi destinazione
- Si definiscono:
  - **c(i,j)** costo del link tra nodo *i* e nodo *j*
  - **D(v)** costo minimo del cammino verso il nodo *v* (minimo relativamente alla iterazione corrente). **D(v)=∞** se il costo del link non è noto
  - **p(v)** immediato predecessore di *v* lungo il cammino a costo minimo verso *v*
  - **N** gruppo nodi il cui cammino di costo minore è noto definitivamente

Algoritmo di Dijkstra - *inizializzazione*

- Passo di inizializzazione seguito da un ciclo eseguito una volta per ogni nodo del grafo
- Al termine saranno stati calcolati i cammini minimi dal nodo *u* verso tutti gli altri nodi

Inizializzazione

N = {***u***}

Per tutti i nodi ***v***

se ***v*** è adiacente a ***u*** */* conosco il costo */*

D(v) = c(u,v)

altrimenti D(v) = ∞

Algoritmo di Dijkstra - *ciclo*

Ciclo

1. Calcola il costo **D(i)** per tutti i nodi adiacenti ***i*** non in **N**
2. Aggiungi a **N** il nodo ***w*** con il minimo costo **D(w)**
3. Aggiorna **D(v)** per ciascun nodo ***v*** adiacente a ***w***, non in **N**:

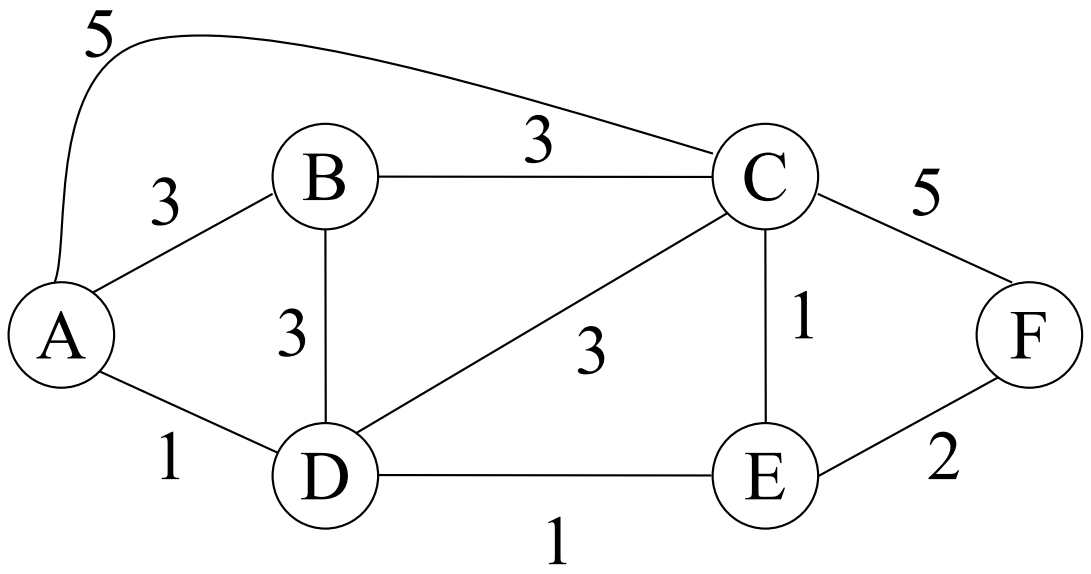
   D(v) = min{ D(v), D(w) + c(w,v) }

until tutti i nodi del grafo sono nell'insieme **N**

Il nuovo costo verso ***v*** è il vecchio costo verso ***v*** oppure è il costo del cammino minimo verso ***w*** più il costo da ***w*** a ***v***

Esempio (step 1)

Calcola per A il costo D(i) per tutti i nodi *i* adiacenti ad A che non sono in N

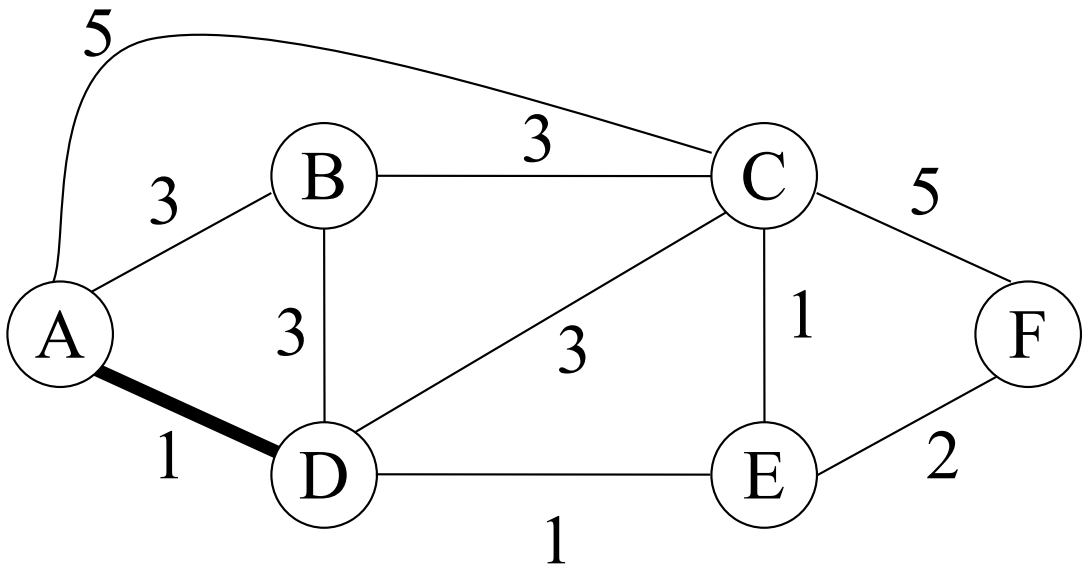


Ciclo	N	D(B), p(B)	D(C), p(C)	D(D), p(D)	D(E),p(E)	D(F), p(F)
	A	3, A	5, A	1, A	∞	∞

Conoscendo già i costi per B, C, D, posso passare subito a calcolare il cammino minimo per E e poi per F?

Esempio (step 2)

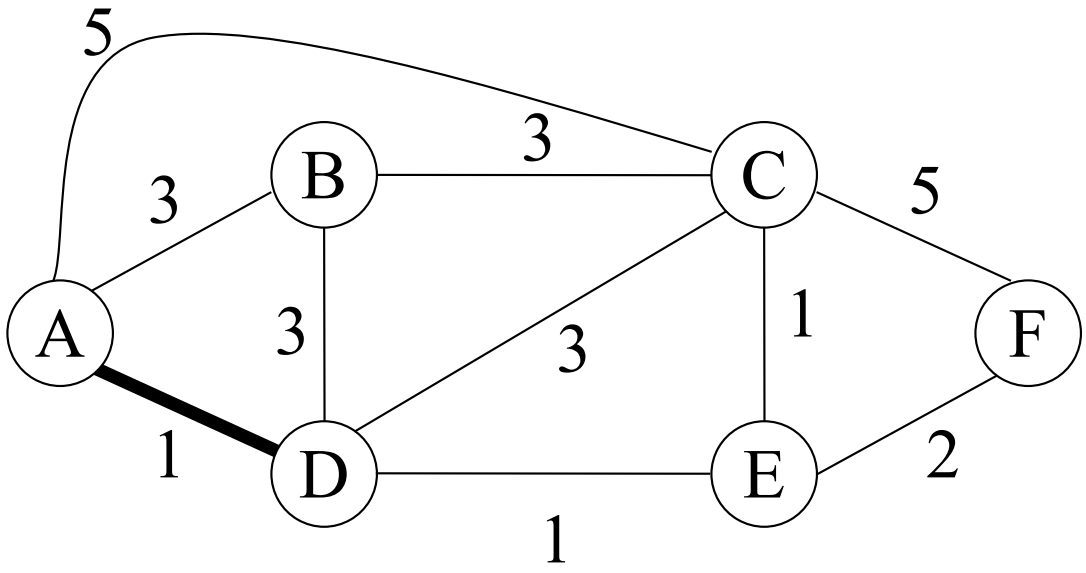
Aggiungi a N il nodo w con il minimo costo D(w)



Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
	A	3,A	5,A	**1**,A	∞	∞
	AD					

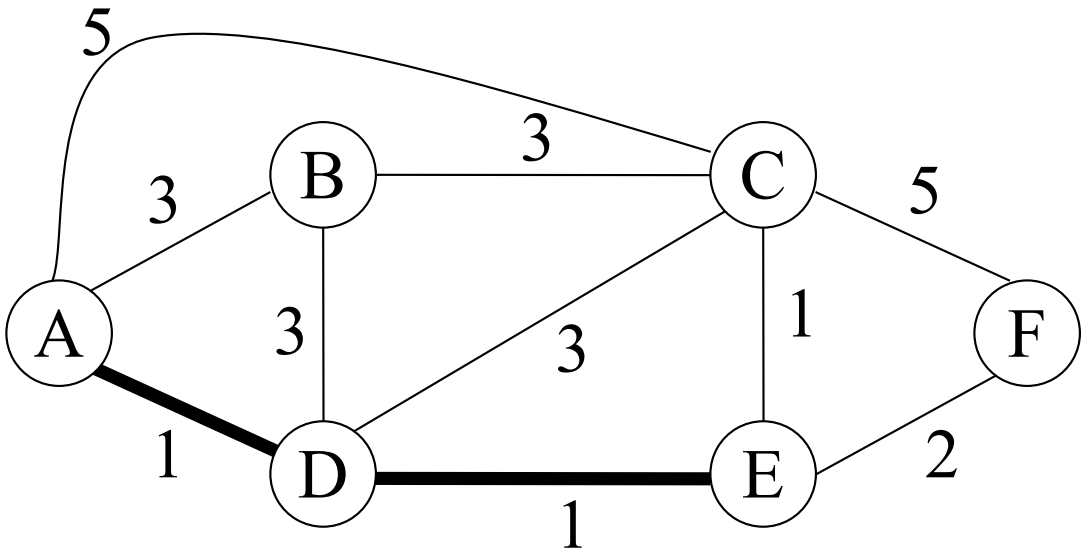
Esempio (step 3)

Aggiorna D(v) per ciascun nodo v adiacente a w, non in N:
D(v) = min{ D(v), D(w) + c(w,v) }



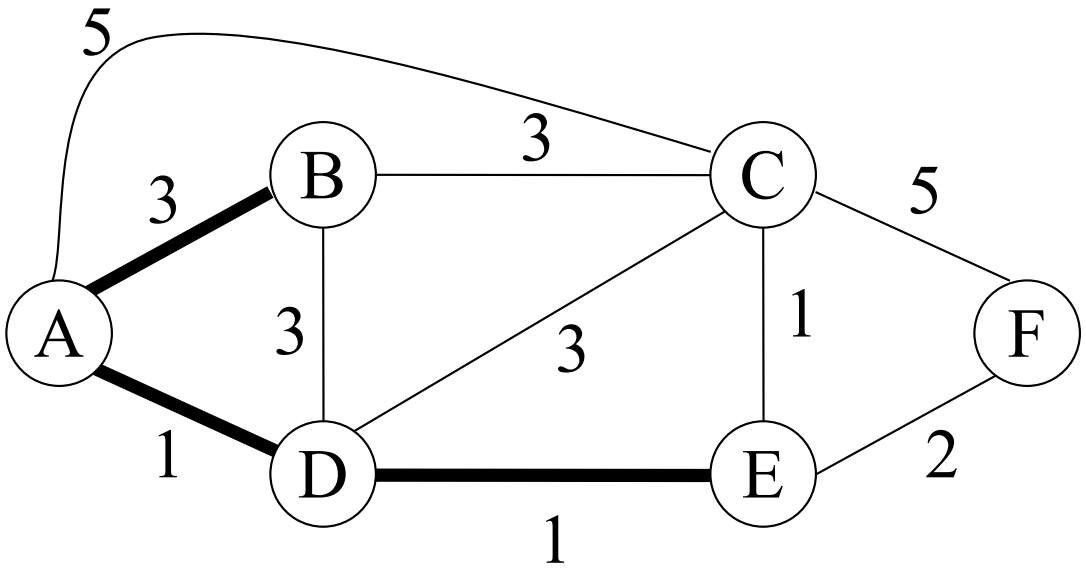
Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
	A	3,A	5,A	**1**,A	∞	∞
	AD	3,A	4,D		2,D	∞

Esempio (ciclo 2)



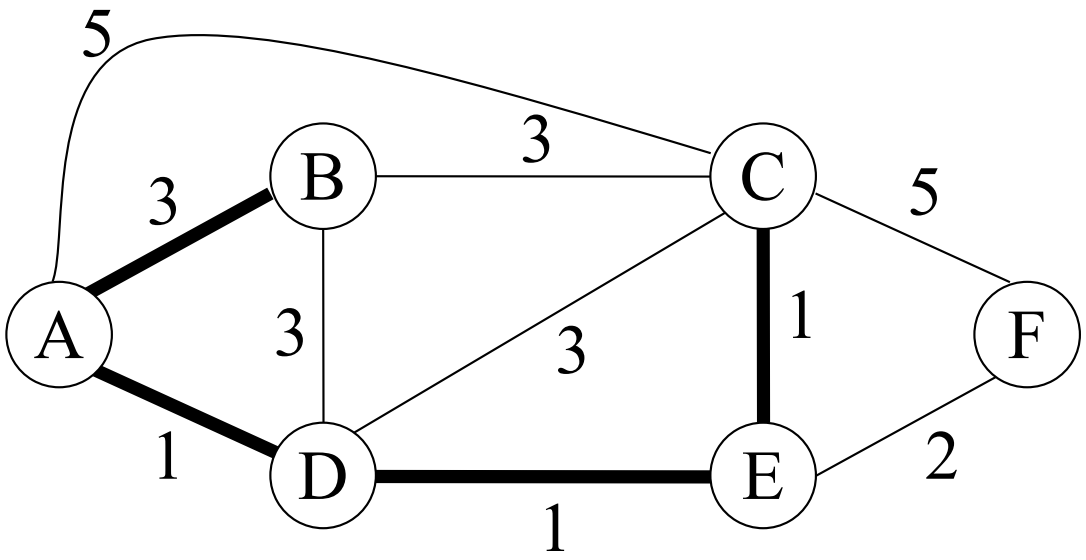
Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	**A**	3,A	5,A	1,A	∞	∞
1	**AD**	3,A	4,D		**2**,D	
2	**ADE**	3,A	3,E			4,E

Esempio (ciclo 3)



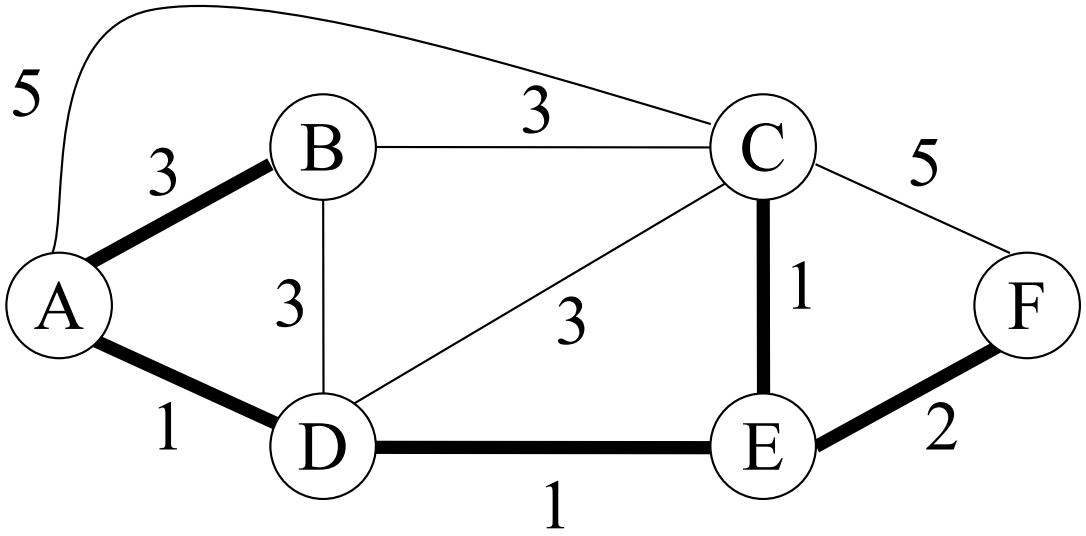
Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	**A**	3,A	5,A	1,A	∞	∞
1	**AD**	3,A	4,D		2,D	
2	**ADE**	**3**,A	3,E			4,E
3	**ADEB**		3,E			

Esempio (ciclo 4)



Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	**A**	3,A	5,A	1,A	∞	∞
1	**AD**	3,A	4,D		2,D	
2	**ADE**	3,A	3,E			4,E
3	**ADEB**		**3**,E			
4	**ADEBC**					4,E

Esempio (ciclo 5)



Ciclo	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	**A**	3,A	5,A	1,A	∞	∞
1	**AD**	3,A	4,D		2,D	
2	**ADE**	3,A	3,E			4,E
3	**ADEB**		3,E			
4	**ADEBC**					**4**,E
5	**ADEBCF**					

Esempio (*finale*)

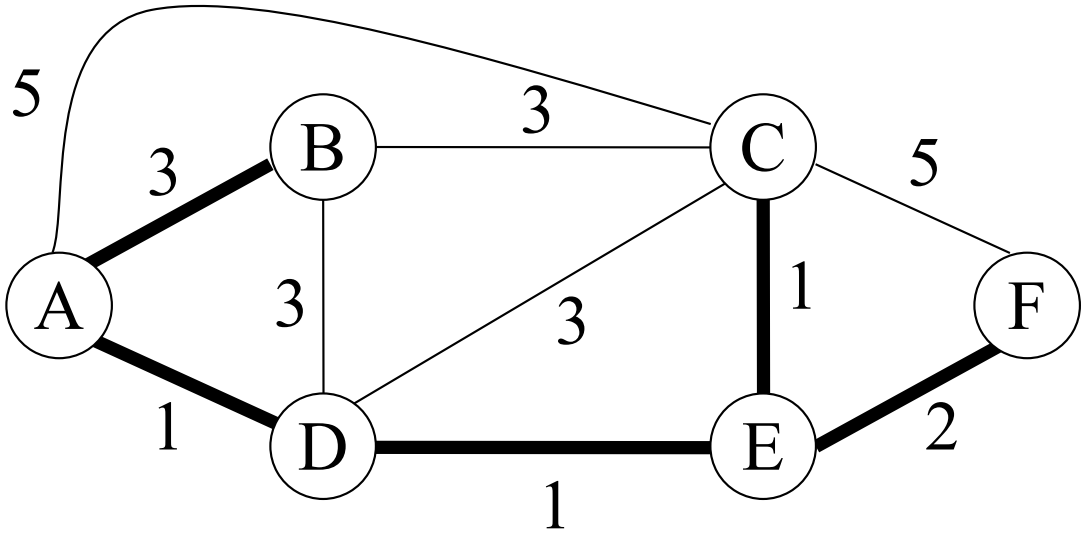


Tabella		B	C	D	E	F
per A		3, A	3, E (D)	1, A	2, D	4, E (D)

Quando i pacchetti arrivano ad A, vengono inoltrati a B se diretti a B, e inoltrati a D in tutti gli altri casi

Modulo *8b*:
Distance vector protocol

Algoritmi Distance Vector

- Usati nel primo periodo di Internet (ARPANET)
- Calcolo distribuito del next hop. E' un algoritmo:
  - adattativo rispetto ai cambiamenti di stato della rete
  - iterativo
  - asincrono
- Unità di scambio dell'informazione:
  - **distance**: "costo" delle varie destinazioni
  - **vector**: c'è una direzione per ogni destinazione

Algoritmo Bellman-Ford: *premessa*

- Si usa la formula di Bellman-Ford per il calcolo del **costo minimo** tra *x* e *y*:

$$D(y) = \min \{ c(x,v) + D(y) \}$$

dove **min** è calcolato tra tutti i nodi vicini ***v*** del nodo ***x***

Intuitivamente, la formula è chiara:

- tra tutti i nodi *v* adiacenti al nodo *x*, il percorso da scegliere per andare a *y* è quello che mi porta con il minor costo da *v* a *y*,
- a meno che (da cui la considerazione del primo addendo) il costo tra *x* e *v* sia talmente alto che mi conviene percorrere altre strade

Algoritmo Bellman-Ford: *premessa*

- Ogni nodo ***x***:
  - aggiorna il proprio vettore (di distanze e direzioni) in risposta a variazioni di costi sui link adiacenti
  - invia un aggiornamento ai nodi adiacenti se il proprio vettore cambia
- Ogni nodo ***x*** mantiene una tabella di routing con i seguenti dati:
  - **c(x,v)** costo del link tra nodo ***x*** e nodo ***v*** in ***N***
  - **c(x,v)=∞** se non c'è link tra nodo ***x*** e nodo ***v*** in ***N***
  - **D =[D (y): y in N]** distanze e direzioni del nodo ***x*** verso tutti i nodi ***y*** nella rete N
  - **D =[D (y): y in N]** distanze e direzioni dei vicini ***v*** di ***x***

Algoritmo Bellman-Ford

Start

Per tutte le destinazioni $y \in N$:

D (y) = c(x,y) se *y* è adiacente

D (y) = ∞ se *y* non è adiacente

Invia il vettore **D = [D (y) | y ∈ N]** a ogni vicino *v*

Loop

Attendi (finchè il costo di un collegamento verso qualche vicino *v* cambia o ricevi un nuovo vettore da un vicino *v*)

Per ogni destinazione *y* in N:

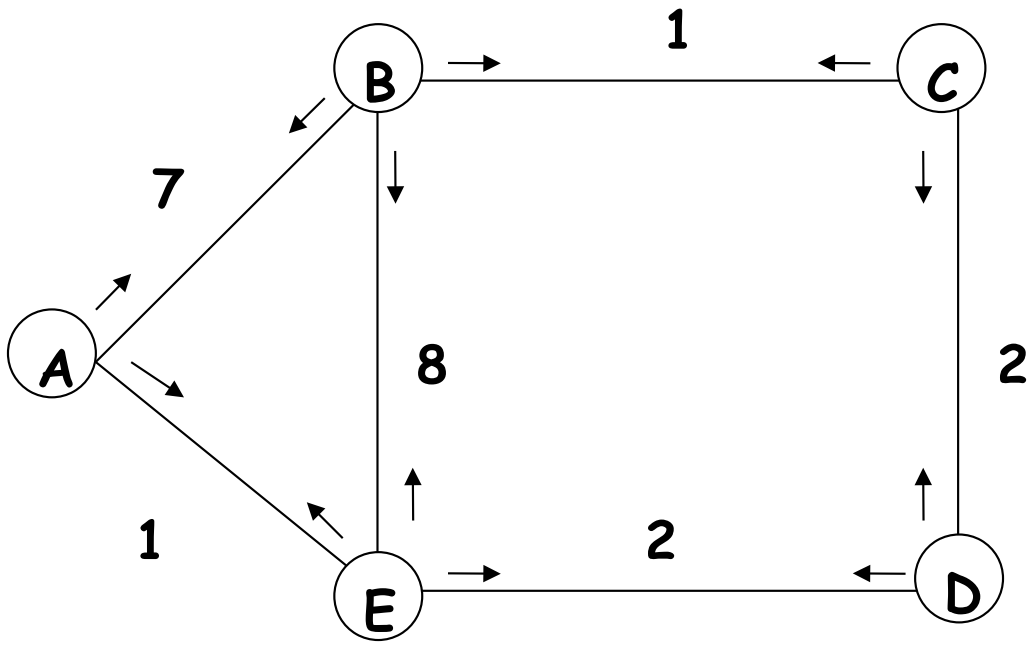
D (y) = min { c(x,v) + D (y) }

Se D (y) è cambiato per qualche destinazione *y*

invia il vettore D = [D (y) | y ∈ N] a tutti i vicini

Esempio: *come si aggiorna lo stato*

Distanze iniziali (*start*)

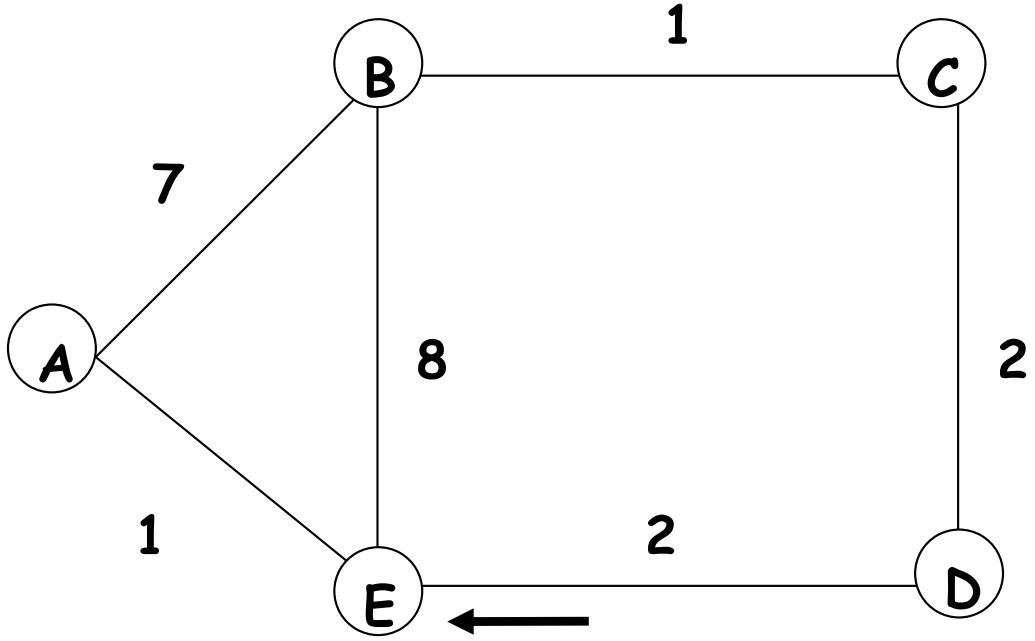


	Distanza dal nodo				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

Ipotesi: si inizia da D

E riceve il vettore di distanze di *D*

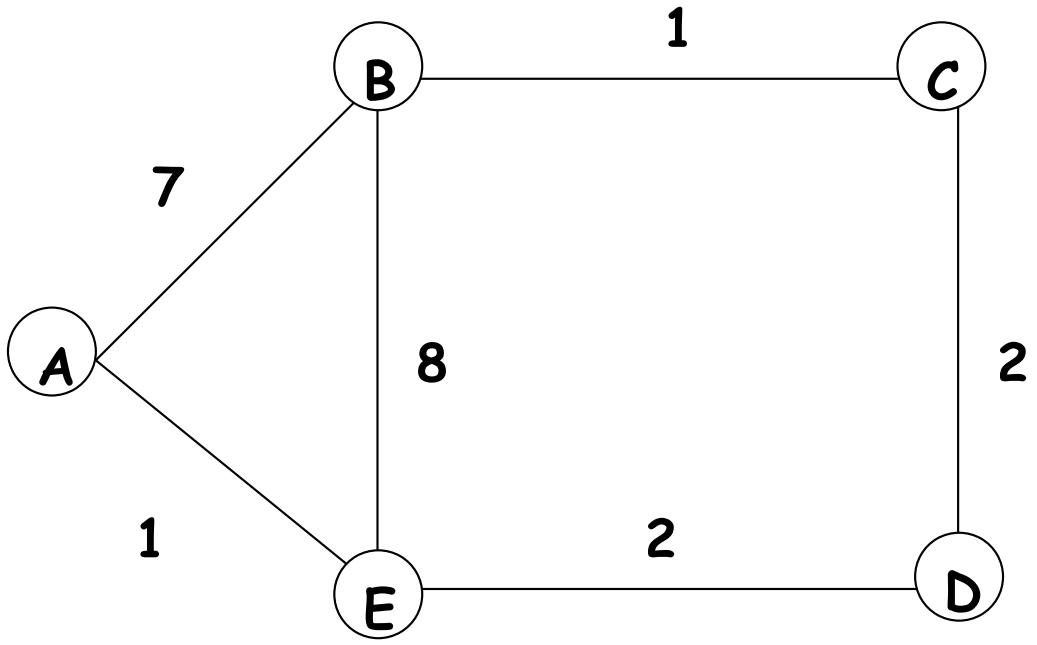
D dice: "Arrivo a C in 2, e a E in 2 (non sono collegato a A e B)"



	Distanza dal nodo				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

E aggiorna i costi per *C*

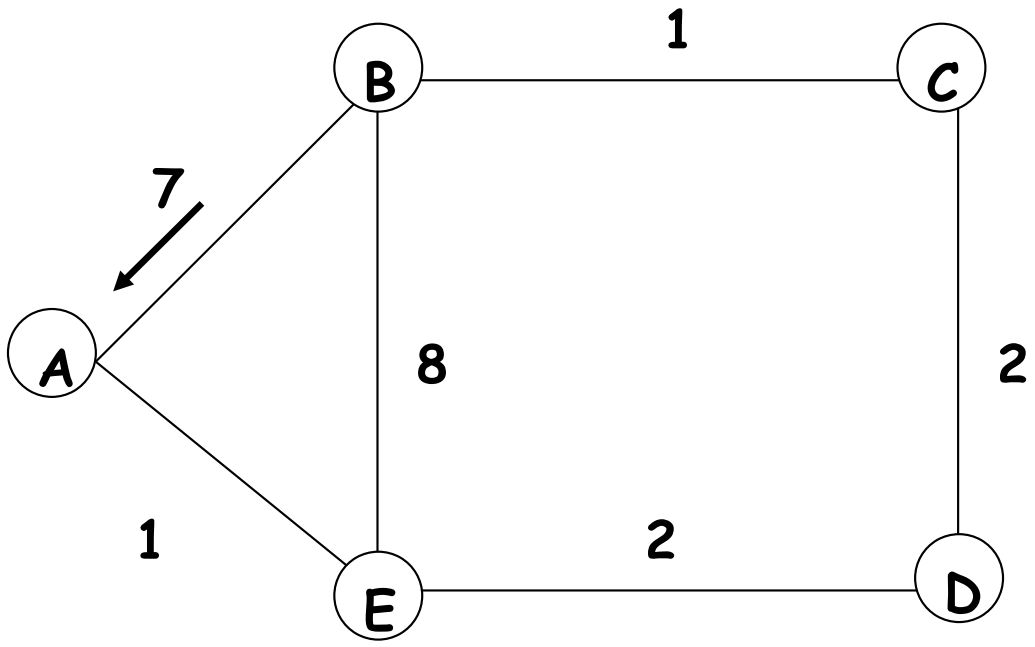
E scopre che può arrivare a C in 2+2 passando attraverso D (prima il costo verso C era ∞)



	Distanza dal nod				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

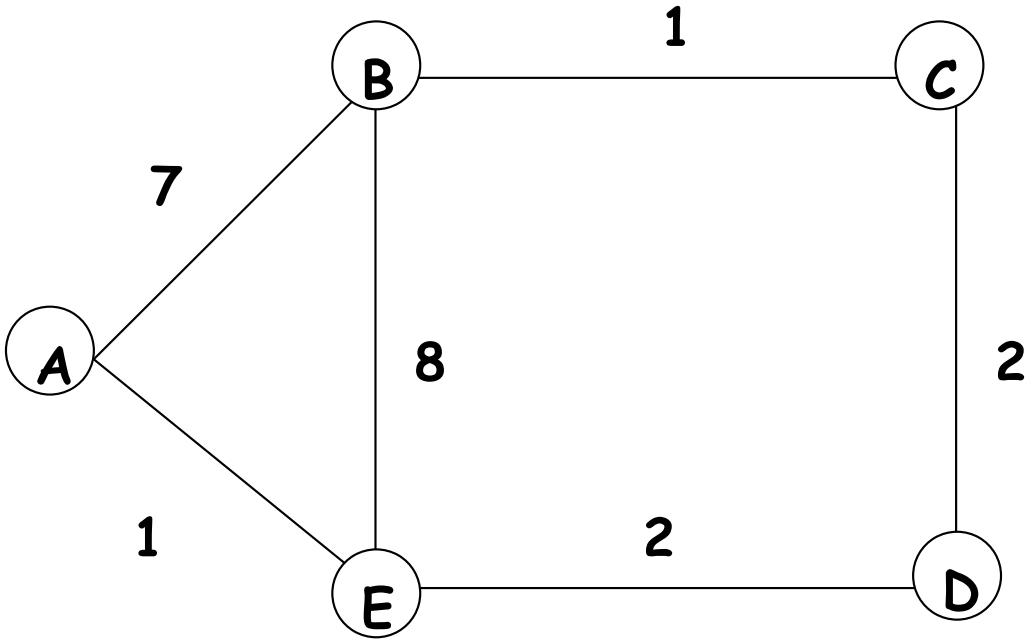
A riceve il vettore di distanze da *B*

B dice: "Arrivo a C in 1, e a E in 8 (non sono collegato a D)"



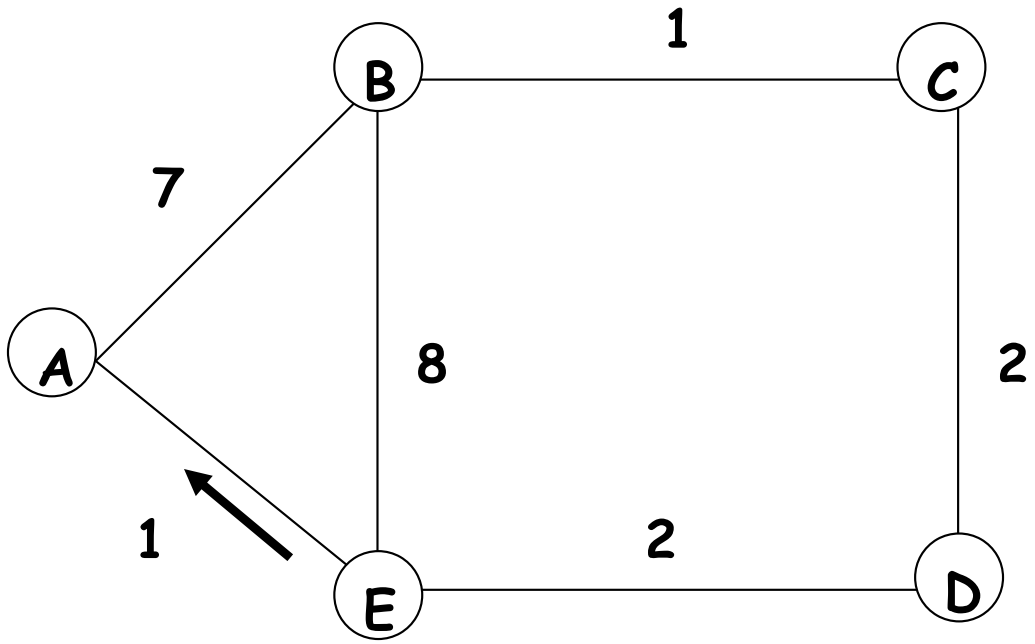
	Distanza dal nodo				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

A aggiorna i costi solo per *C*



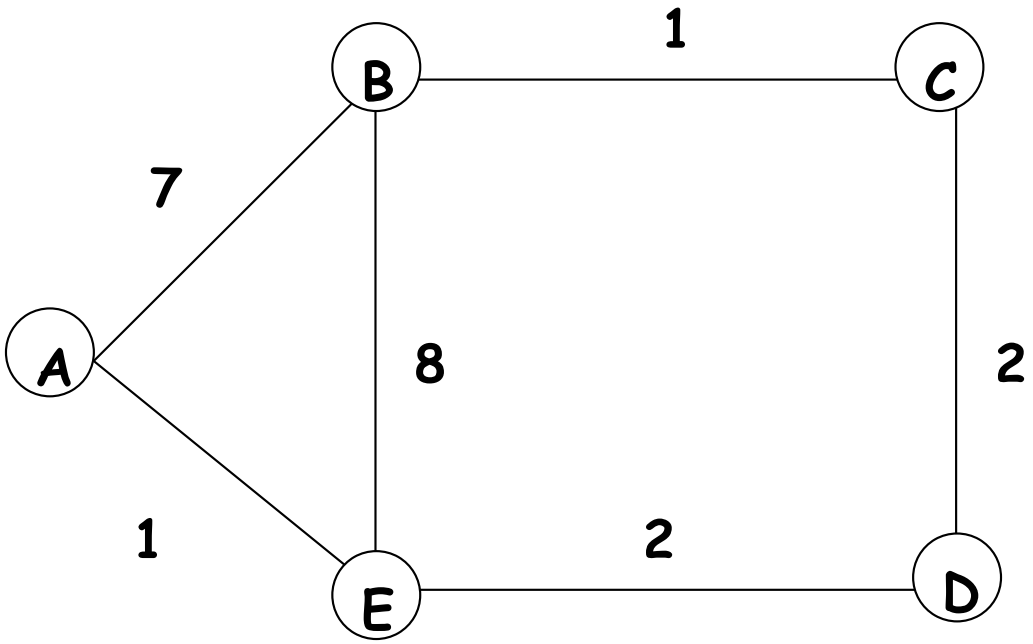
	Distanza dal nodo				
	A	B	C	D	E
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

A riceve il vettore di distanze da *E*



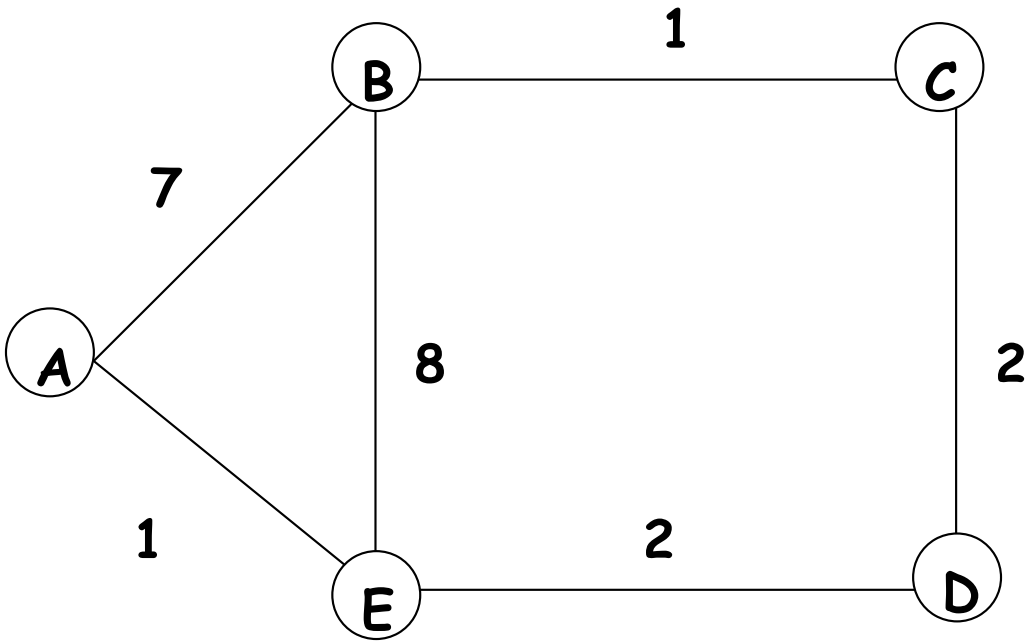
	Distanza dal nodo				
	A	B	C	D	E
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

A aggiorna i costi per *C* e *D*



	Distanza dal nodo				
	A	B	C	D	E
A	0	7	5	3	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Distanze finali (*aggiornate*)

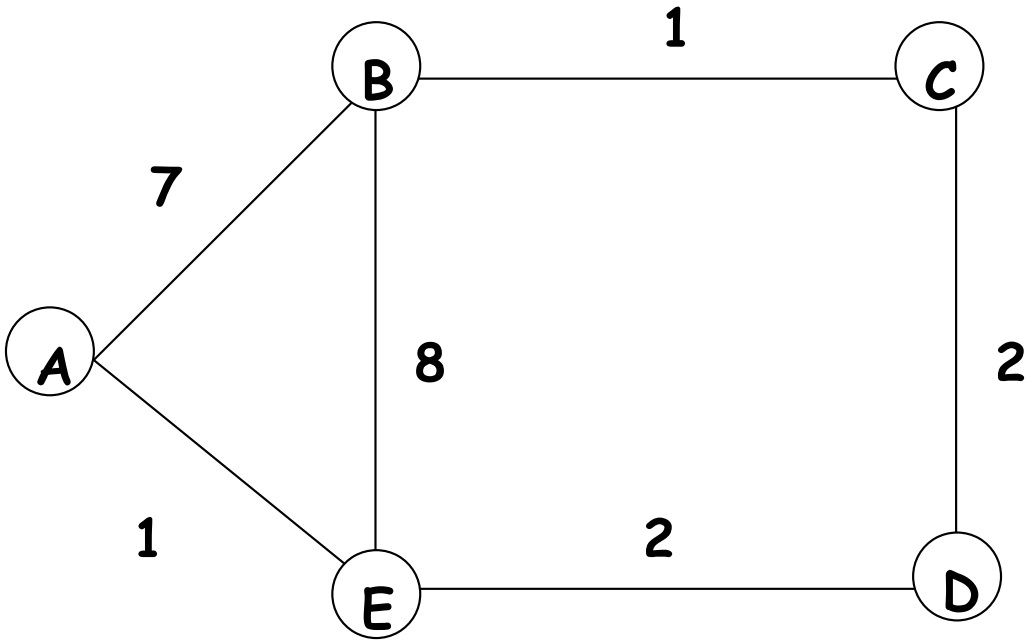


	Distanza dal nodo				
	A	B	C	D	E
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

Tabella di routing

- L'algoritmo di Bellman-Ford ha un'immediata ricaduta pratica. Serve, infatti per calcolare i valori della ***Tabella di routing*** di ciascun router
- La *Tabella di routing* del nodo *x* ha:
  - una riga per ogni *nodo destinazione* della rete (router o AS)
  - tante colonne quanti sono i *nodi adiacenti* al nodo *x*
  - i costi di cammino come elementi della tabella e la conseguente direzione (next hop minimo)
- In questo modo, **nel momento in cui arriva un pacchetto con un indirizzo destinazione, il router può subito decidere verso quale link inoltrarlo**

Instradamento visto dal nodo *E*



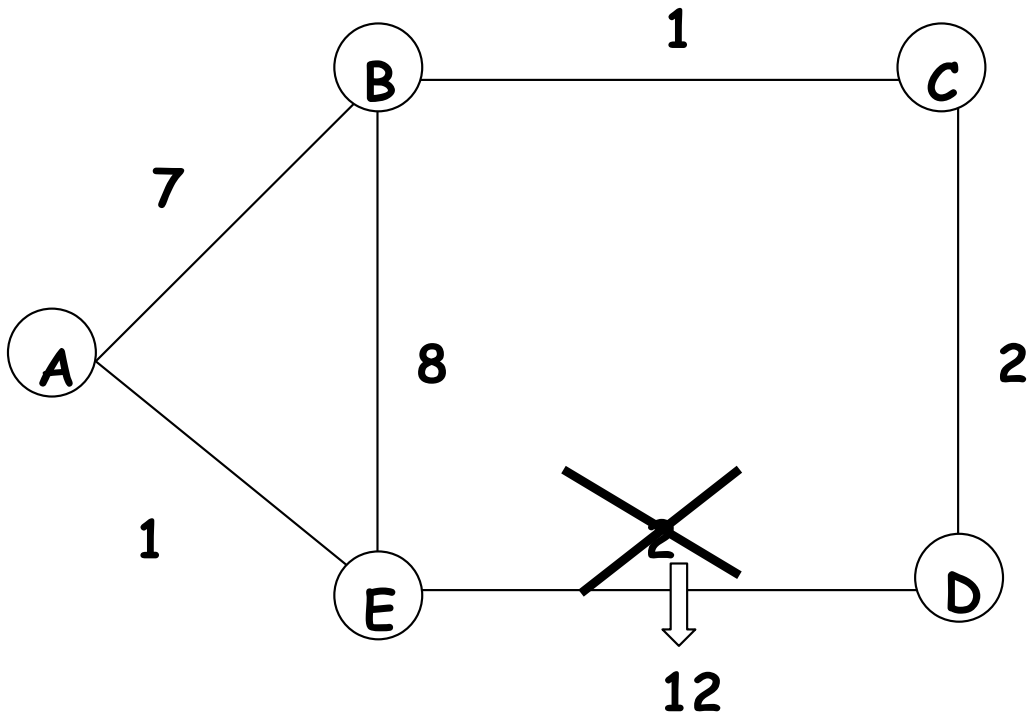
Potenziali tabelle di routing di E

	Next hop		
Dest	**A**	**B**	**D**
A	**1**	**15**	**5***
B	**7***	**8**	**5**
C	**6***	**9**	**4**
D	**4***	**11**	**2**

La tabella di routing di E ha una riga per ogni destinazione nella rete e tante colonne quanti sono i nodi adiacenti a E (attenzione ai percorsi con cicli – discussione dopo*)**

I percorsi di minor costo per la corrispondente destinazione sono indicati in rosso nella *tabella di routing*

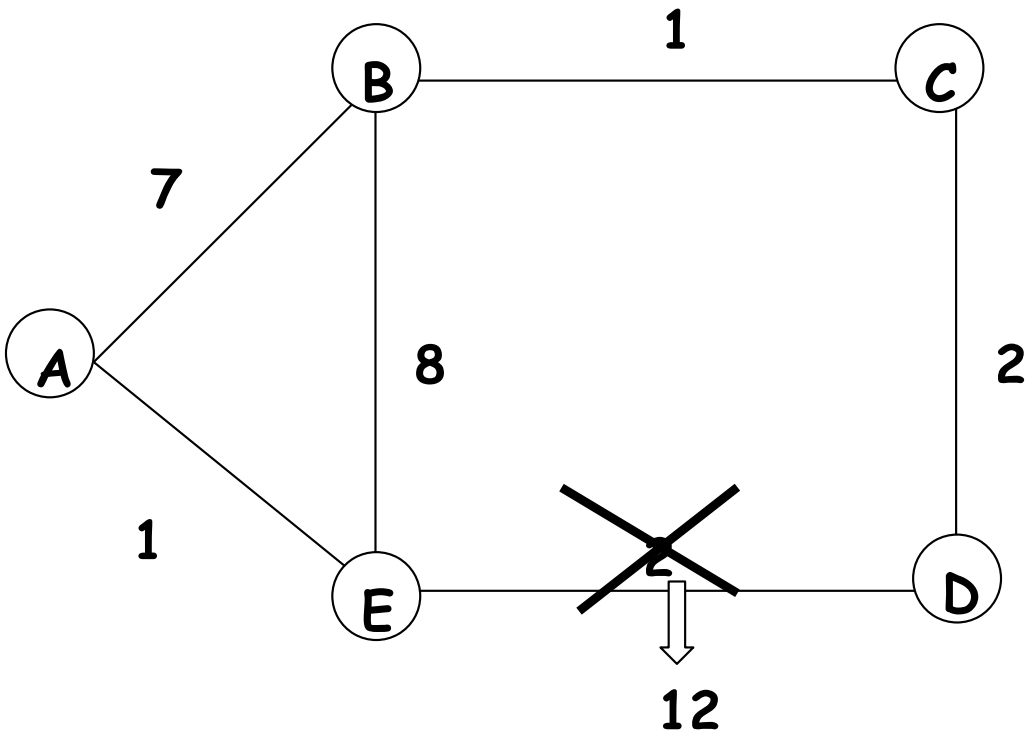
Problema 1: un link “degrada”



	Distanza dal nodo				
	A	B	C	D	E
A	0	6	8	10	1
B	6	0	1	3	8
C	8	1	0	2	9
D	10	3	2	0	12
E	1	8	9	12	0

- I nodi che vertono sul link E-D, ricalcolano il vettore
- Aggiornano la propria routing table e trasmettono il nuovo vettore ai vicini
- Ciascun nodo ricalcolerà il proprio vettore e, iterativamente, lo invierà ai nodi vicini

Nuovo instradamento visto dal nodo *E*



Potenziali tabelle di routing di E

	Next hop		
Dest	A	B	D
A	1	15	5*
B	7*	8	5
C	6*	9	4
D	4*	11	2

	Next hop		
Dest	A	B	D
A	1	15	22*
B	8	8	15
C	9	9	14
D	11	11	12

I percorsi di minor costo per la corrispondente destinazione sono indicati in rosso nella *tabella di routing*

Problema 2

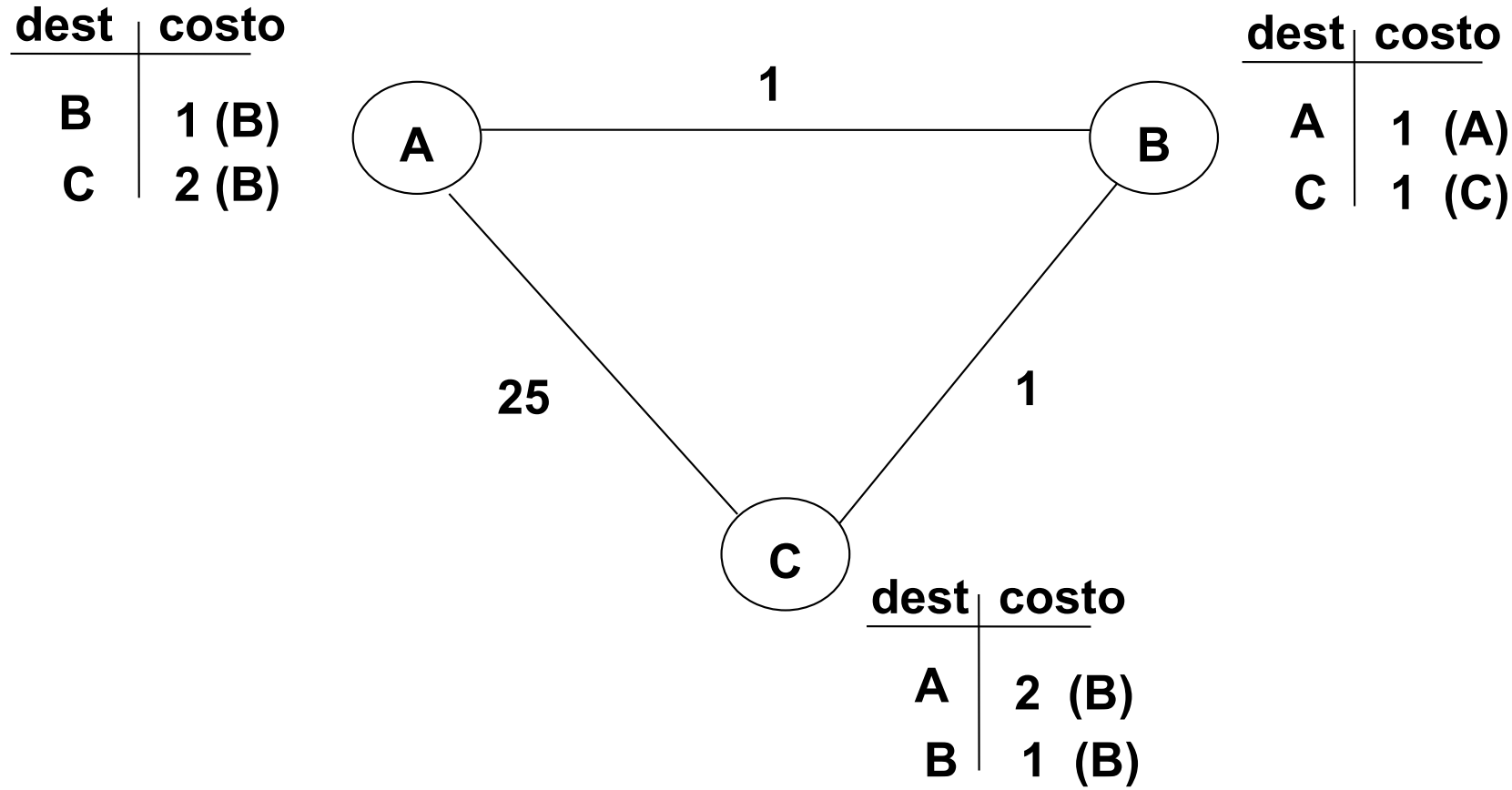
Ci sono più di 1 miliardo di host e milioni di router

- E' credibile una tabella che contenga tutti i router di Internet come destinazione?
- Come si gestisce nella realtà il problema?

→ ***Tecniche di aggregazione indirizzi***

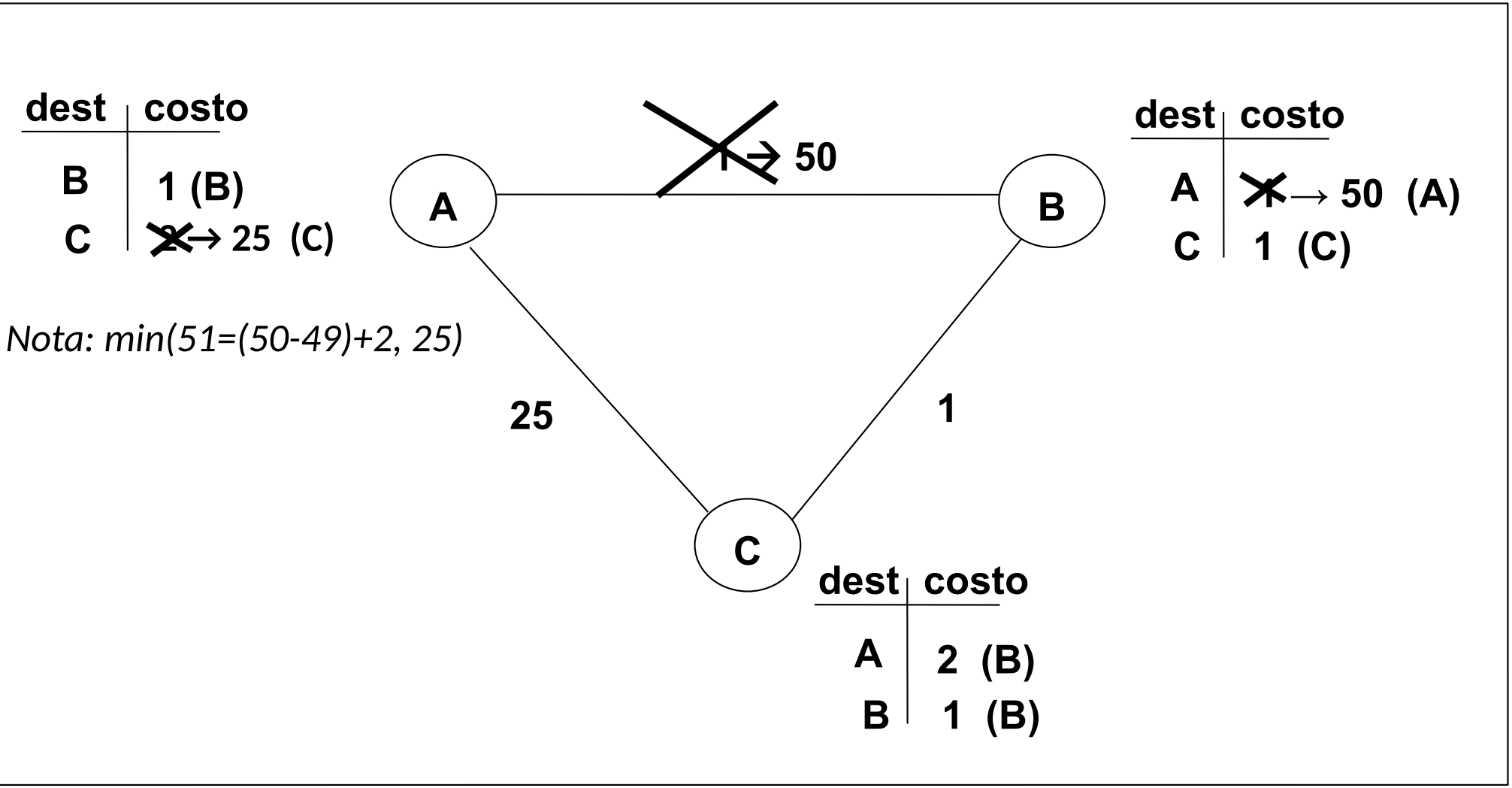
Problema 3: effetto rimbalzo [1]

Consideriamo una rete con cammini minimi già calcolati



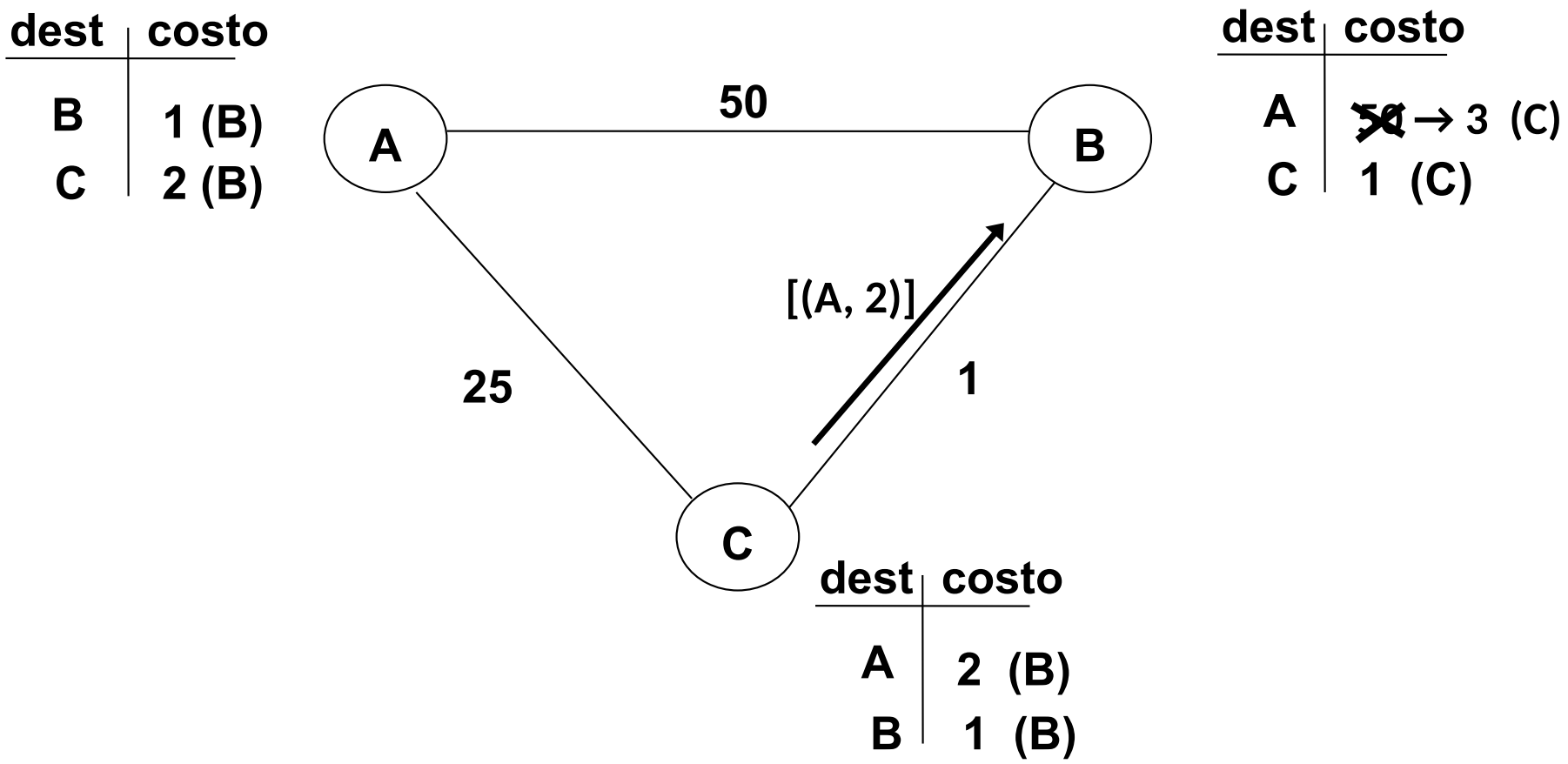
Problema 3: effetto rimbalzo [2]

Il costo del collegamento fra A e B aumenta da 1 a 50



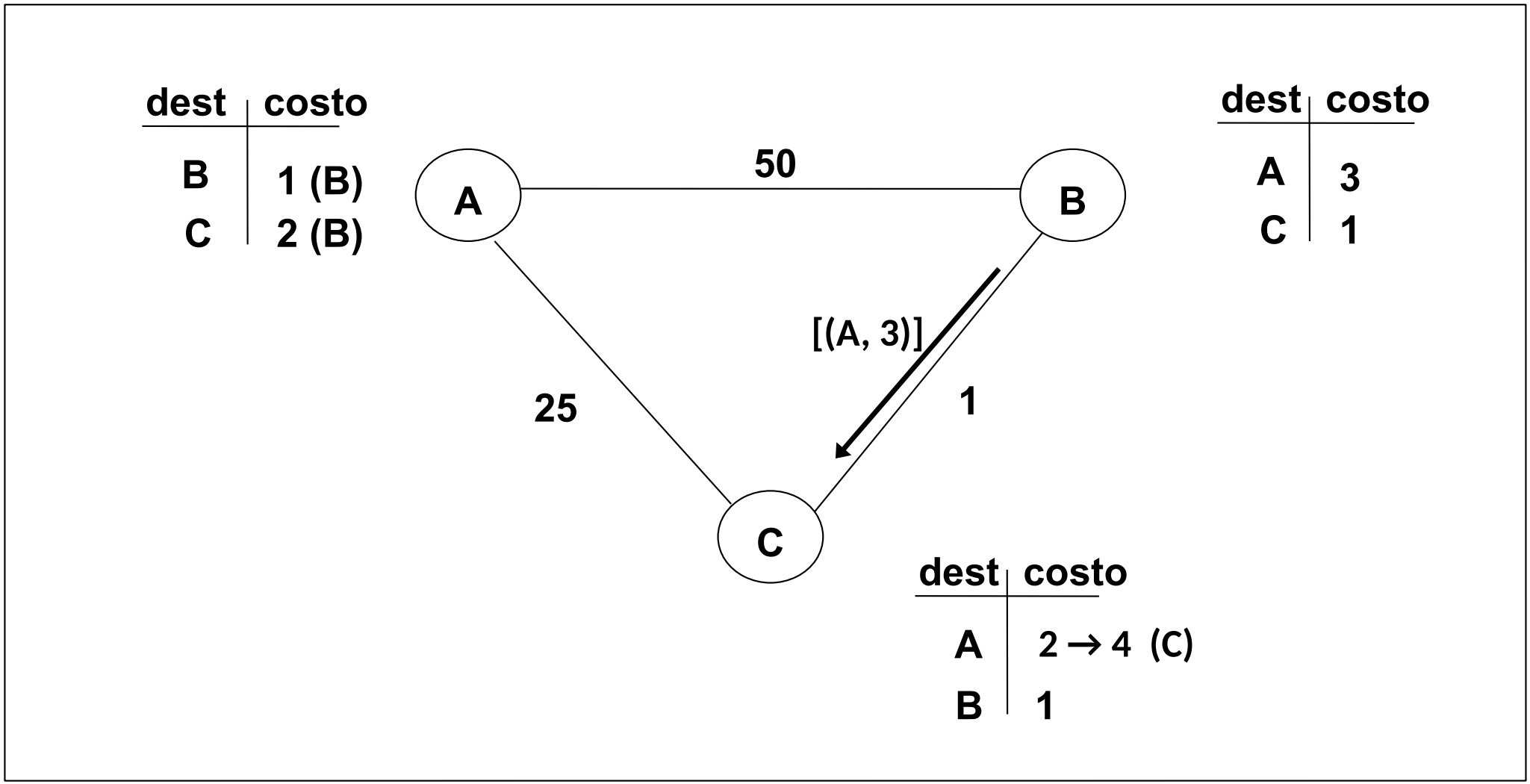
Problema 3: effetto rimbalzo [3]

C pubblicizza la sua conoscenza, e B aggiorna il costo



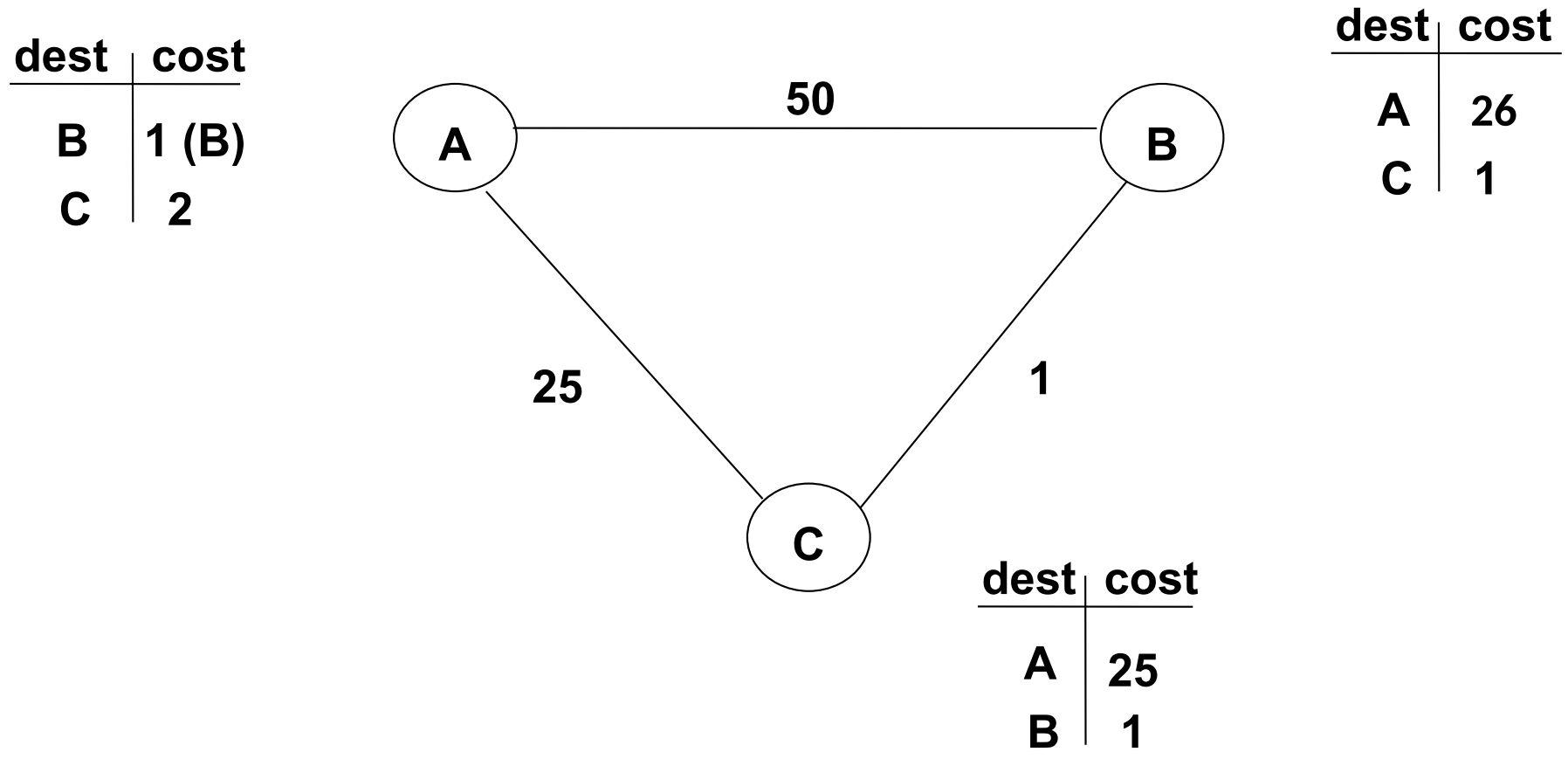
Problema 3: effetto rimbalzo [4]

B pubblicizza la sua conoscenza, e C aggiorna il costo



Problema 3: effetto rimbalzo [5]

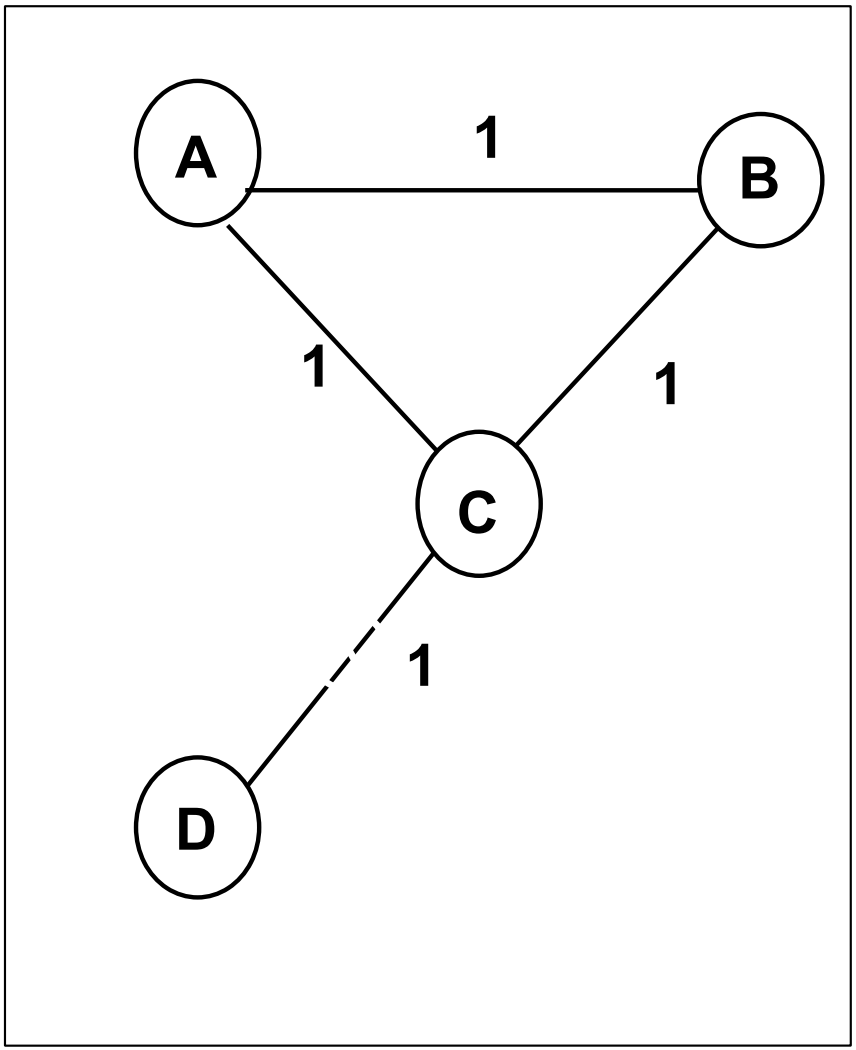
I costo si aggiornano fino a quando non supereranno un costo alternativo (25)



Come si crea l'effetto rimbalzo

- La distanza diretta da *B* verso *A* **cresce molto**
- Quindi, *B* sceglie *C* come prossimo hop per *A*
- Ma…, il **percorso implicito** da *C* verso *A* include B!
- Le tabelle di B e C si aggiornano gradualmente, ma si crea un loop che proseguirà fino a quando *C* considererà il proprio percorso verso *A* attraverso *B* minore di 25
- Un pacchetto che arrivi a *B* o a *C* durante l'esistenza del loop rimbalzerà tra questi due nodi (*almeno se non si azzera prima il TTL*)

Caso peggiore: non c'è stabilizzazione



- Nel caso in cui il link C-D diventa inutilizzabile, C marca D come irraggiungibile e lo elimina dagli aggiornamenti inviati ad A e B
- Si supponga che A riceva per primo l'aggiornamento. Adesso A considera che il cammino minimo verso D sia attraverso B
- A dichiara D irraggiungibile a B e a C notifica un costo pari a 3
- C vede D raggiungibile attraverso A a costo 4 e lo notifica a B
- B notifica un costo di 5 ad A che notificherà un costo aggiornato di 6 a C
- Rischio: "count-to-infinity"

Possibili soluzioni (1)

- **Evitare il "count-to-infinity"**
  - Scegliere una soglia (abbastanza bassa) per "rappresentare" l'infinito. Es., massimo numero di hop necessari = 16 (vecchia scelta)
- **Split Horizon**
  - Bisogna differenziare i vettori di distanze inviati ai nodi adiacenti: il vettore di B inviato a C non dovrà contenere alcuna delle destinazioni raggiungibili tramite C
  - Obiettivo: "Se B raggiunge A attraverso C, non ha senso per C cercare di raggiungere A attraverso B"

Possibili soluzioni (2)

- **Split Horizon with poisoned reverse**
  - Se B raggiunge A attraverso C, B avvertirà C che la sua distanza verso A è infinita. In questo modo, anche se in realtà sa di poter instradare i pacchetti tramite C, il costo risulta troppo alto

- **Queste soluzioni non funzionano per cicli che coinvolgono 3 o più nodi**
- **In questo caso, si ricorre all'azzeramento del TTL del datagram**

Link State *vs*. Distance Vector

- Si devono confrontare due parametri:
  - *overhead* del routing (numero e dimensione dei messaggi)
  - *robustezza*
- Dimensione dei messaggi
  - LS: piccola
  - DV: tendenzialmente molto grande (=sua tabella di routing)
- Numero di messaggi
  - LS: molto grande, di tipo O(*n*), dove *n* sono i nodi del grafo
  - DV: piccolo in quanto le comunicazioni sono solo ai vicini

Link State *vs*. Distance Vector

Robustezza

- **LS**: calcolo dei percorsi effettuato in maniera indipendente da ogni nodo

  → protezione contro guasti ai router

- **DV**: calcolo dei percorsi basato sui calcoli degli altri router

  → il calcolo sbagliato di un router può essere propagato a gran parte della rete (es., “count-to-infinity”)

Conclusione

- Non c'è un chiaro vincitore tra i due algoritmi:
  - Link state (*globale*) ha dei vantaggi
  - Distance vector (*distribuito*) ha altri vantaggi

- Gli algoritmi di tipo *Link state* (*globali*) oggi tendono ad essere utilizzati all'interno degli AS (anche se il primo intra-AS era Distance vector)
- Gli algoritmi di tipo *Distance vector* (*distribuiti*) sono utilizzati per il routing tra diversi AS

Protocollo RIP
(Intra-AS)

Origini del RIP

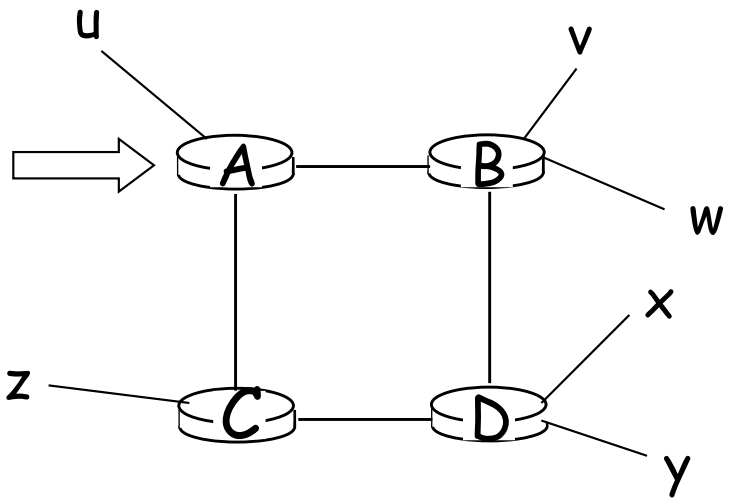
- E' il primo protocollo storico per il routing intra-AS, utilizzato per molti anni in Internet
- Definito in [RFC 1058]
- La sua diffusione fu determinata soprattutto dal fatto che era implementato in modo nativo nel sistema operativo Unix BSD (*daemon routed*): primo sistema operativo ad avere tutto il software per supportare l'intero stack TCP/IP
- E' un protocollo distribuito basato sull'algoritmo ***Distance Vector*** (**"propago quello che so su tutta la rete solo ai vicini"**)

Routing Information Protocol

- Basato sul protocollo ***Distance Vector***
- Metrica di costo (*semplificata*) = **numero di hop**
  ➔ ***ipotesi: tutti i link hanno costo unitario***
- **Costo massimo di un percorso = 16 hop**
  ➔ limite massimo del diametro di un AS

Numero di hop dal router A alle varie sottoreti destinazione

Destinazione	Hop
u	1
v	2
w	2
x	3
y	3
z	2



RIP advertisement e RIP request

- I router adiacenti si scambiano i vettori di distanze ogni 30 secondi utilizzando un **messaggio di *RIP advertisement***
- Ogni messaggio contiene fino a 25 sottoreti di destinazione all'interno dell'AS con le relative distanze in hop

- Un router può anche chiedere informazioni sul vettore di distanze dei router adiacenti, tramite **messaggi di *RIP request***

RIP: esempio

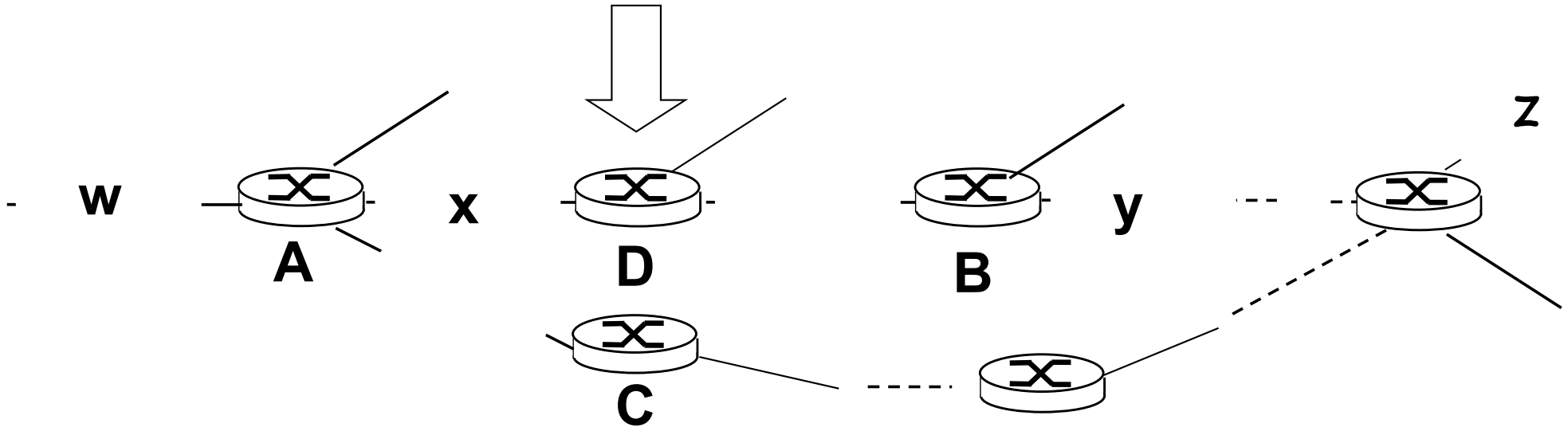


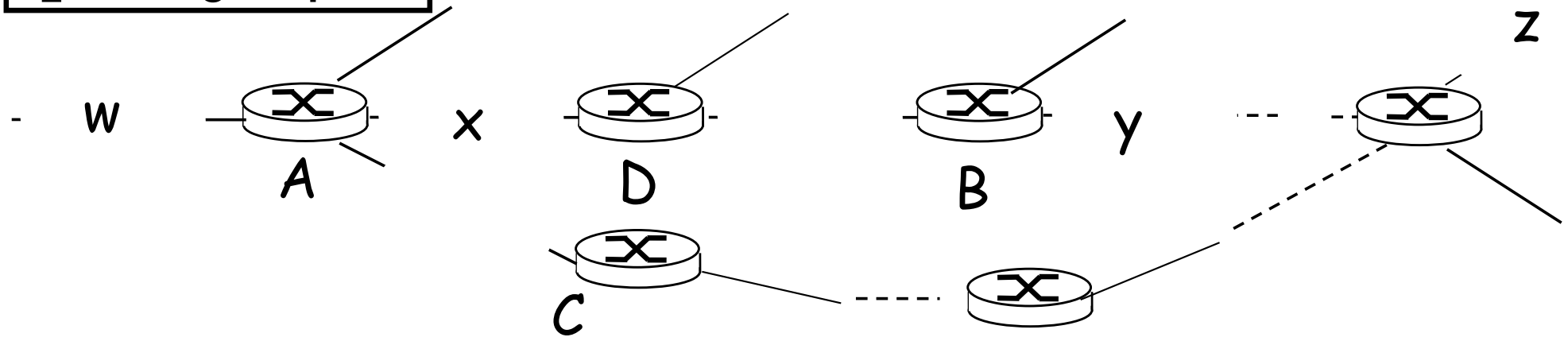
Tabella di routing in D

Sottorete destinazione	Router successivo	Numero di hop verso destinazione
w	A	2
y	B	2
z	B	7
x	--	1

RIP: esempio

Dest	Next	hop
w	-	1
x	-	1
z	C	4

RIP advertisement
da A a D (“io arrivo a z con 4 hop”)



Sottorete destinazione	Router successivo	Numero di hop verso destinazione
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1

Tabella di routing in D

Link failure

- Se un router non riceve messaggi dal suo vicino dopo **180 secondi** → lo considera **irraggiungibile**
  - I percorsi passanti per il router vicino vengono invalidati: si setta il flag U(nreacheable)
  - Nuovi RIP advertisement vengono inviati agli altri router vicini
  - A loro volta, i vicini inviano RIP advertisement se le loro tabelle subiscono cambiamenti

→ ***Rapida propagazione delle informazioni sui link failure (permanenti o temporanei) della rete***

Pro e contro del RIP

- **Come ogni algoritmo distribuito di tipo *Distance vector*, RIP funziona bene per reti non grandi, stabili e veloci**
- Ciascun router comunica ai vicini il percorso "migliore" misurato in numero di hop
- Nel momento in cui c'è instabilità, il RIP "soffre":
  - Poiché ciascun router comincia a inviare la propria nuova tabella, prima di arrivare a convergenza, non c'è una visione unitaria sullo stato della rete e sui percorsi migliori
  - Non c'è intrinseca protezione dai loop e c'è il rischio del "count-to-infinity"
  - Proprio per questa mancata protezione, si usa un "infinito" piccolo (16) che tuttavia impedisce al RIP di essere utilizzato per reti di grandi dimensioni

Protocollo OSPF
(Intra-AS)

Open Short Path First

- **Definito nell'RFC 1131**
- **"open"** = disponibile pubblicamente
- E' un algoritmo globale di tipo ***link state protocol***
- Ha varie funzioni migliorative rispetto a RIP e quindi è adatto a:
  - reti più grandi
  - reti il cui stato tende a cambiare dinamicamente
- Attualmente, si tende a utilizzare:
  - RIP nell'ambito di AS piccoli di secondo livello e di reti aziendali molto grandi
  - OSPF all'interno di AS medio-grandi di primo livello

Open Short Path First

- Il routing si basa sull'**algoritmo centralizzato Link State** ("tell the world about the neighbors")
  - Topologia della rete e costi noti a ogni nodo
  - Calcola l'albero dei cammini di costo minimo mediante l'***algoritmo di Dijkstra***
  - Memorizza tale albero nel cosiddetto "link state database" che viene distribuito a tutti i router
  - Invia in broadcast eventuali aggiornamenti di costo con i vicini ai router dell'**intero AS** (***flooding***)
  - I messaggi OSPF viaggiano direttamente su IP
  - Il "link state database" viene inviato periodicamente (almeno ogni 30 minuti) anche se non è cambiato

Caratteristiche di OSPF (non in RIP)

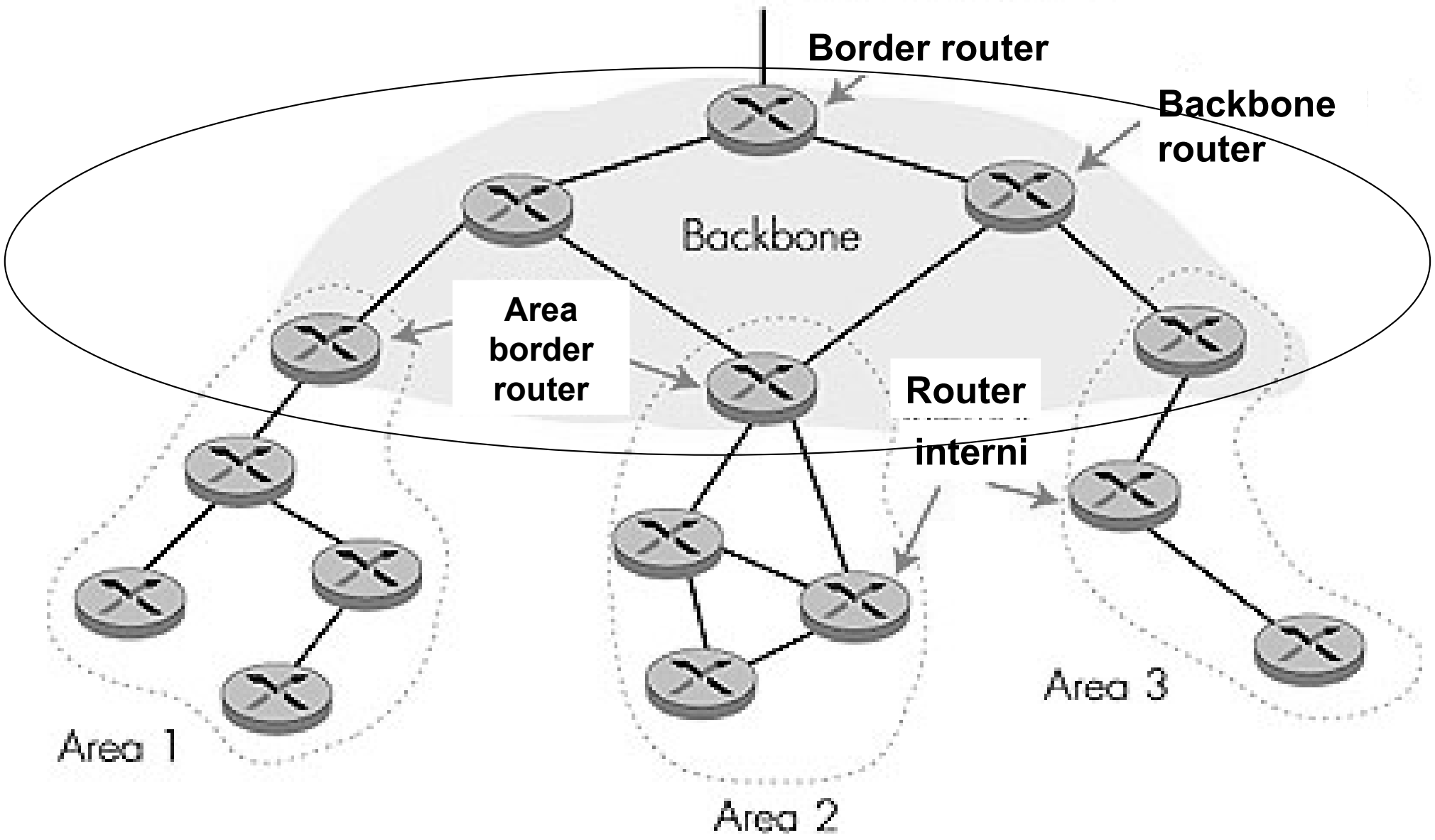
- **Sicurezza**: possibilità di autenticare i messaggi OSPF con algoritmi di crittografia
- **Percorsi multipli con costo uguale**: possibilità di usare più percorsi per instradare il traffico (mentre è solo uno in RIP)
- **Supporto integrato per instradamento *unicast* e *multicast***: multicast OSPF (**MOSPF**) usa lo stesso database di collegamenti usato da OSPF
- **Struttura gerarchica degli AS**: possibilità di strutturare grandi domini di instradamento in gerarchie di AS

Però anche OSPF non è perfetto perché il broadcast costa …

Gerarchia OSPF in grandi aziende

- Consente una suddivisione dei router di grandi aziende in **aree**
- Esempio: azienda con 500 router. Possibilità di creare 10 aree ciascuna con 50 router → ogni router memorizza e tiene aggiornate informazioni solo su 50 router, invece di 500
- Quindi, si avranno:
  - **Internal router** che applicano OSPF all'interno della propria area
  - **Area border router** che comunicano i percorsi verso altre aree ai router di quell'area. Questi router non usano i dettagli, ma solo i ***prefissi***

Sistema autonomo OSPF gerarchico



Protocollo BGP
(Inter-AS)

Un po' di storia…

- Fino agli anni '80: **EGP**
  - Storicamente il primo protocollo ad essere usato per il routing inter-AS (analogamente al RIP per intra-AS)
  - Presuppone una rete con topologia ad albero senza cicli (come la vecchia ARPANET)
  - Limiti nella massima dimensione delle reti gestibili
  - Entra in crisi con l'introduzione delle dorsali Internet e dei cammini multipli tra nodi

- ***BGP viene introdotto per sostituire EGP***

BGP

Border Gateway Protocol versione 4 (**BGP4**): **RFC 1771** del marzo 1995

- E' un protocollo complesso, ma fondamentale per il funzionamento di Internet, in quanto è il protocollo delle dorsali Internet per muoversi da un AS a un altro AS in modo completamente decentralizzato
- Utilizzato dagli ISP
- Può essere utilizzato anche come protocollo intra-AS nel caso di AS molto grandi (in quanto il protocollo intra-AS OSPF non scala molto bene)

BGP: quale algoritmo?

- **Problemi con il *distance-vector*:**
  - L'algoritmo di Bellman-Ford converge lentamente e ha problemi di *loop* e *counting to infinity*

- **Problemi con il *link state*:**
  - Le metriche usate dai router di diversi AS possono essere diverse
  - Il database di LS è troppo grande per tutta Internet
  - Espone le politiche adottate da un AS ad altri AS ←

SOLUZIONE → *Path Vector* (di tipo *distance-vector*)

Algoritmo *Path Vector*

- **Con il *Path Vector*** (una variante dell'algoritmo distribuito *Distance Vector Protocol*), ogni *routing update* contiene informazioni sull'**intero cammino** verso la destinazione attraverso gli AS

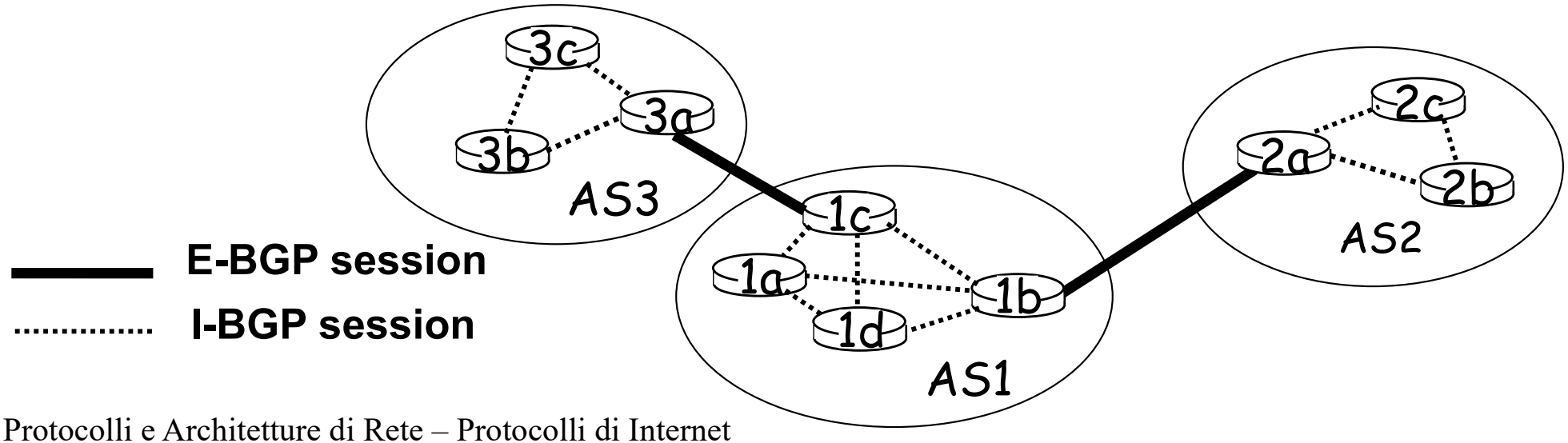
- **Individuazione dei loop**:
  - Quando un AS riceve un update riguardo un percorso, controlla se il percorso contiene se stesso:
    - Se sì, scarta l'update
    - Se no, aggiunge se stesso e, se necessario, propaga il percorso ulteriormente

Interconnessioni tra BGP router

- Il BGP usa il **protocollo TCP** per connettere i *router peer* (porta 179) ➔ Robustezza della comunicazione (anche se sembra anomalo avere un protocollo con controllo di congestione per gestire un protocollo di routing best-effort)
- Vantaggi del BGP:
  - Un AS determina il percorso, e il protocolllo garantisce che non vi siano loop
  - Non ci sono refresh periodici frequenti: i percorsi sono considerati validi fino a che non vengono sovrascritti o la connessione con un peer è persa
  - Gli aggiornamenti sono incrementali
  - **Le metriche di un AS sono locali e non esposte**

Sessioni BGP

- Poiché il BGP fa uso di **connessioni TCP** [si vedrà in seguito] semi-permanenti per far comunicare i router confinanti (**BGP peers**), i due peer BGP formano una **sessione BGP**
  - Sessione **esterna** (**E-BGP**) tra router di AS diversi
  - Sessione **interna** (**I-BGP**) tra router dello stesso AS



BGP implementa l'hop-by-hop di Internet

BGP è consistente con il modello *hop-by-hop* previsto dal paradigma progettuale di Internet

- Un router BGP informa i router vicini solo riguardo ai percorsi che utilizza
- In altre parole, AS1 non può chiedere a AS2 di instradare il traffico in modo diverso da quello che AS2 ha scelto, ovvero di chiedere di inviare i suoi pacchetti a un AS diverso da quello che AS2 sceglie di utilizzare

Funzioni BGP e prefissi

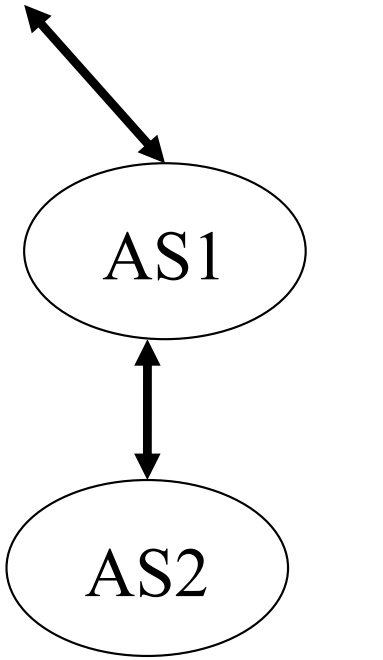
Funzioni principali

1. Scambiare informazioni di raggiungibilità tra **AS confinanti**, detti ***peer*** (configurando manualmente i router)
2. Propagare le informazioni di raggiungibilità a tutti i router all'interno di un AS → ***meccanismo distribuito*** basato sull'algoritmo **Path Vector** (della classe ***Distance Vector Protocol***)
3. Determinare i percorsi migliori in base a informazioni di raggiungibilità e policy di routing (non solo metriche!)

- Le destinazioni sono indicate con **prefissi** che rappresentano una o più sottoreti (ampio utilizzo di **aggregazione CIDR** di indirizzi per ridurre le entry della routing table)

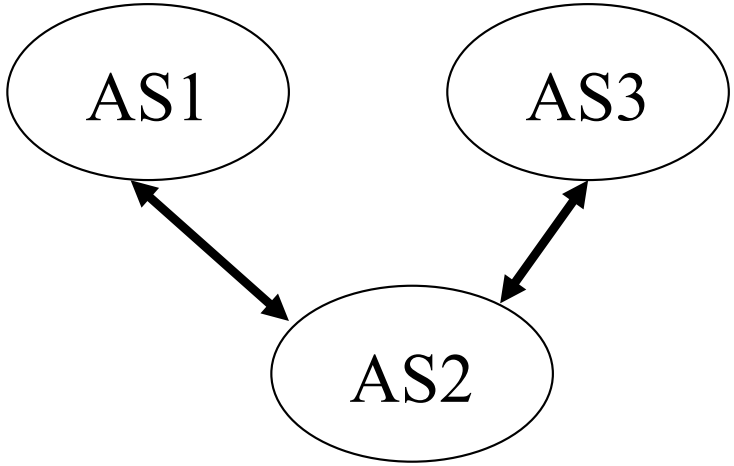
Categorie di AS

BGP NON NECESSARIO

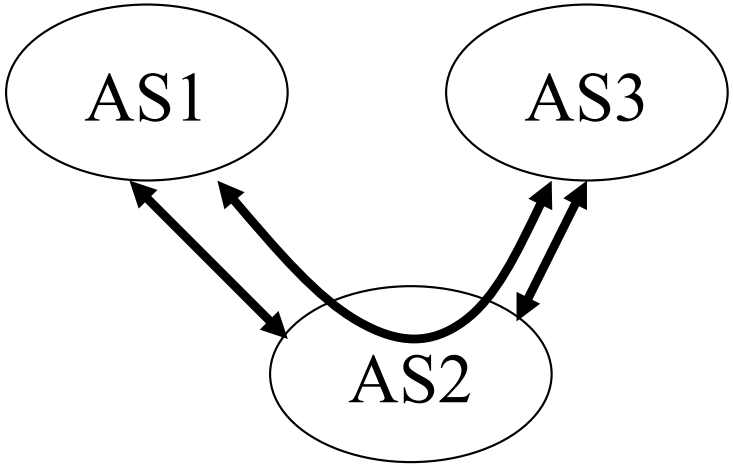


AS2: **Stub**

BGP NECESSARIO



AS2: **Multi-homed**



AS2: **Transit**

- In generale, BGP non serve nei casi di: *Single homed network* (*stub*), AS non fornisce *downstream routing*, AS usa un *default route*

Router e BGP

- **Transit router**: router che gestiscono traffico **I-BGP** all'interno dell'AS
- I transit router devono essere configurati a maglia (*mesh*), cioè tutti devono essere peer di tutti gli altri. Questo pone dei problemi di scalabilità, risolti mediante *confederazioni*

- **Border router** (o **edge router**) router che gestiscono traffico **E-BGP** tra diversi AS
- La scelta dei peer di un border router dipende dalle politiche del gestore dell'AS

Selezione del percorso

Informazioni basate sui *path attributes* + Informazioni esterne (*policy*)

Esempi di attributi

- hop count
- presenza o assenza di certi AS
- AS_path origin (prefisso appreso da protocollo IGP, prefisso appreso da protocollo EGP, non definito)
- AS_path attribute (elenco di AS attraversati; “non modificare path” se inoltrato a router interno, “inserire se stesso nel percorso” se inoltrato a router BGP)
- dinamica dei link (stabili, instabili)
- Next_hop: specifico router da cui giunge l’annuncio (possibilità di più collegamenti tra gli AS)

Politiche del BGP

- BGP offre la possibilità di implementare diverse politiche
- Le politiche non sono parte del BGP, ma sono fornite al BGP come informazioni di configurazione

- BGP implementa le politiche:
  1. **Scegliendo percorsi tra diverse alternative**
  2. **Controllando l'invio di advertisement ad altri AS**

Esempi di politiche disponibili

- A un certo punto, un AS multi-homed rifiuta di agire come transit per altri AS
  → Limita il path advertisement
- Un AS multi-homed decide di agire come transit per alcuni AS
  → Effettua il path advertisement solo per quegli AS
- Un AS può favorire o penalizzare certi AS per il traffic transit che viene originato da lui

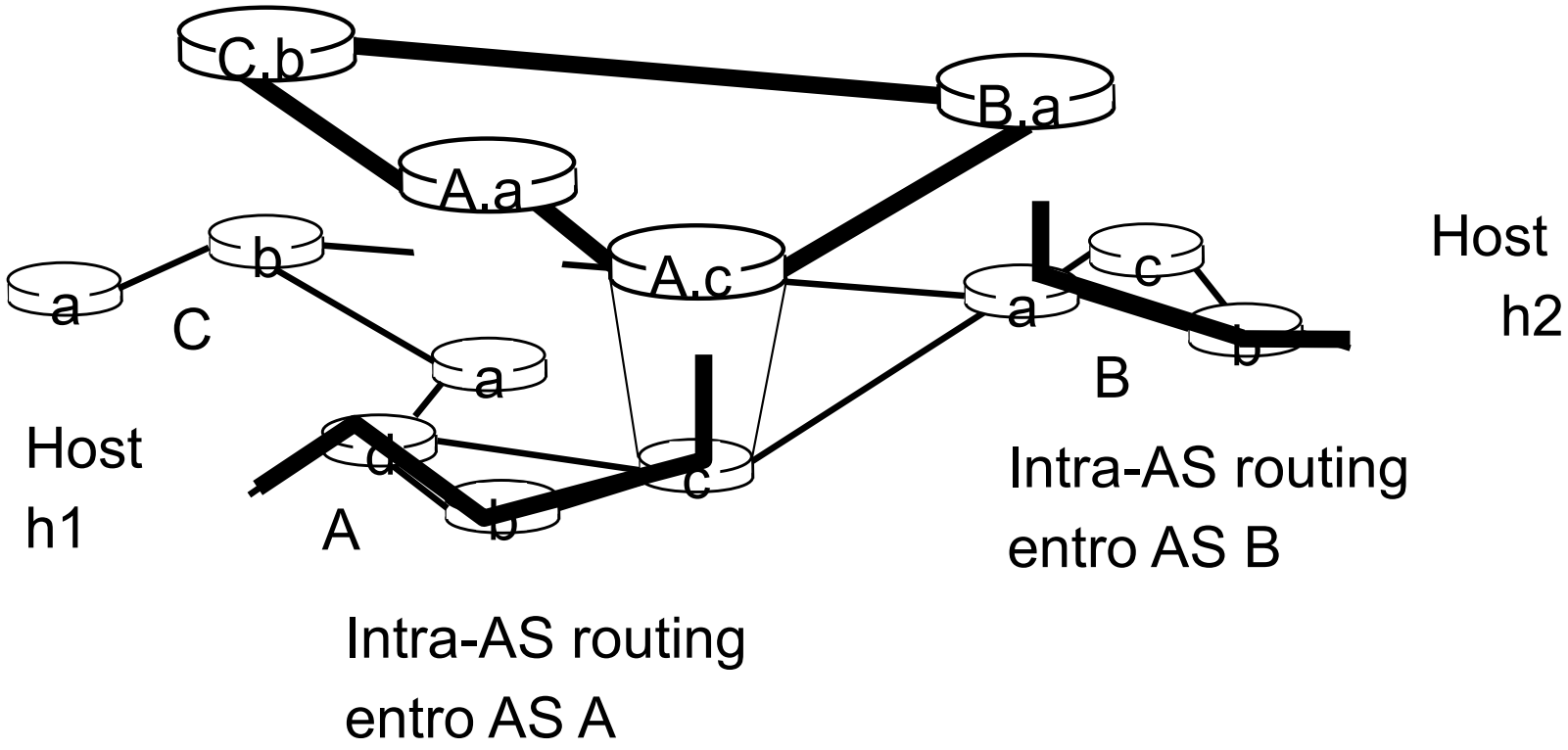
Annuncio di un prefisso

- **"Annunciare un prefisso" (*prefix advertisement*) da parte di un AS equivale alla "promessa" di questo AS di inoltrare i pacchetti su un percorso verso il prefisso di destinazione**

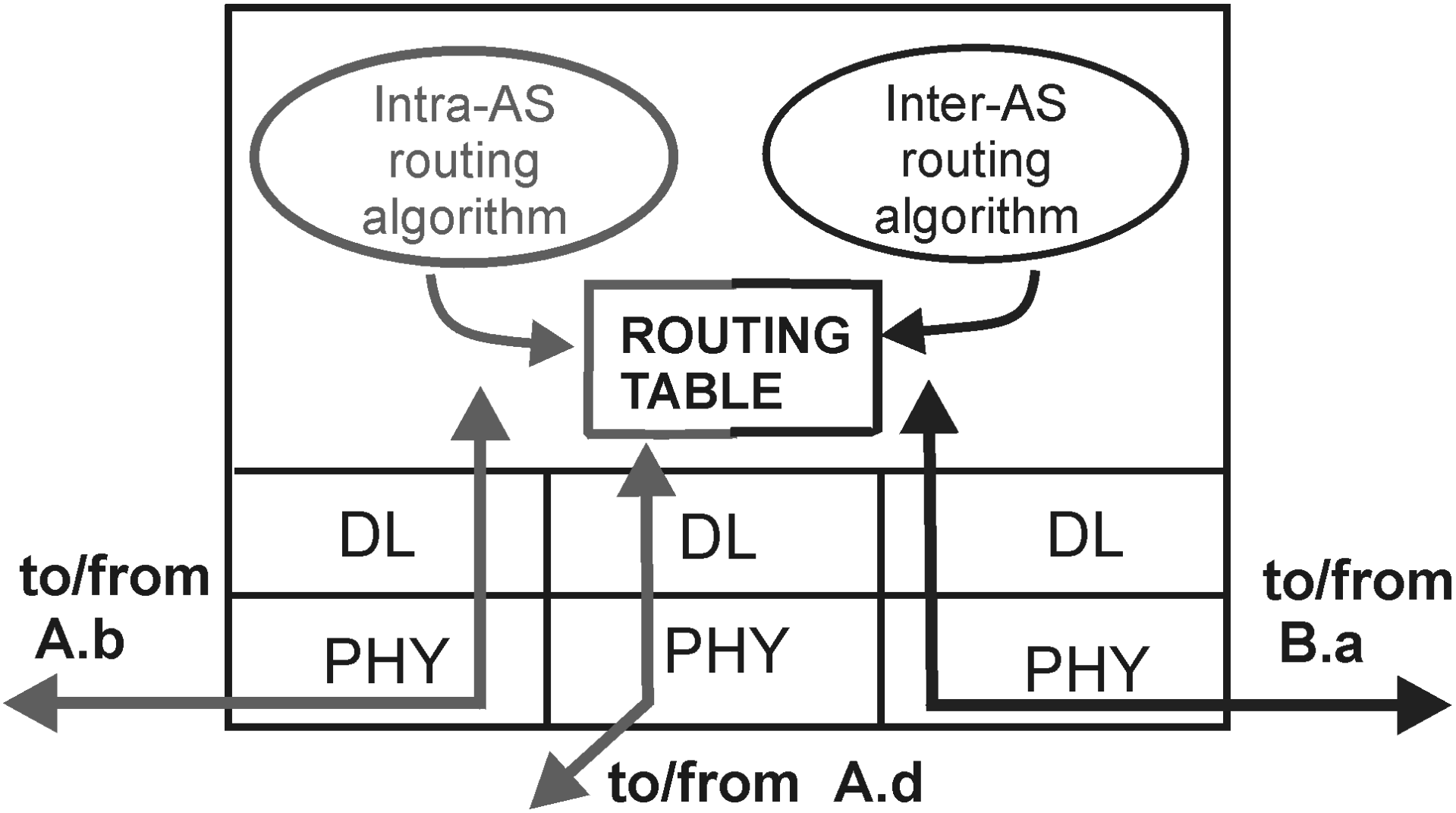
- L'annuncio di un prefisso può comprendere anche **attributi BGP**

Connettere diversi AS

I router di confine (***border router***) hanno la responsabilità di inoltrare pacchetti a destinazioni esterne all'AS

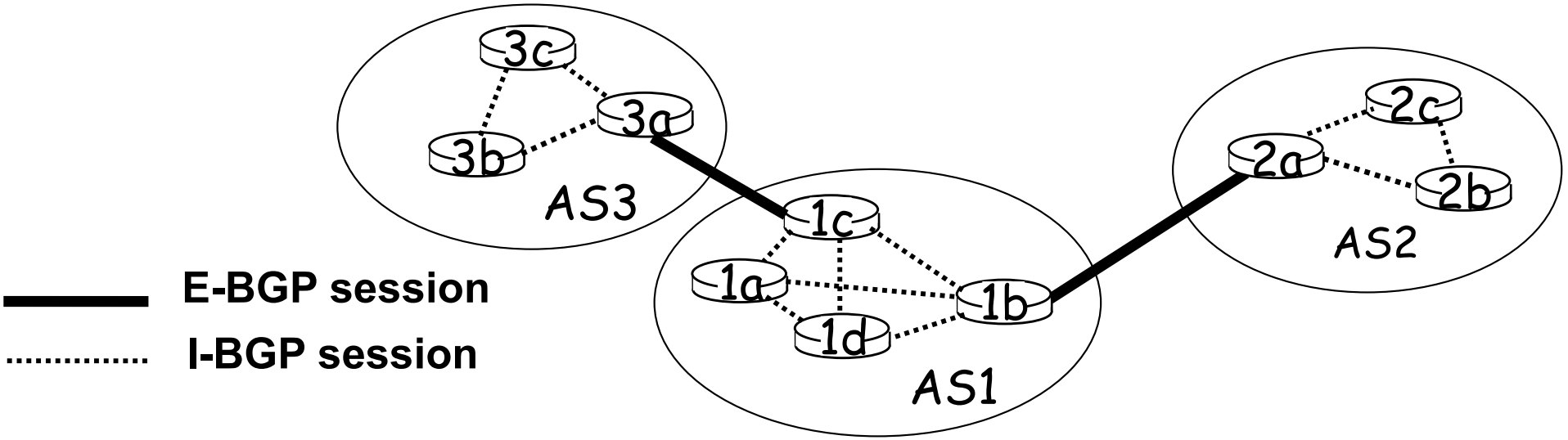


Architettura di un edge router



Distribuzione di informazioni per la raggiungibilità

- 3a annuncia a 1c i prefissi di rete raggiungibili da AS3 attraverso una sessione E-BGP
- 1c usa I-BGP per distribuire le informazioni di raggiungibilità a tutti i router in AS1 (1a – 1d – 1b)
- 1b annuncia a 2a i prefissi raggiungibili da AS3 e AS1 attraverso una sessione E-BGP
- Quando un router viene a conoscenza di un nuovo prefisso, crea una **nuova riga nella propria tabella di routing**



Selezione del percorso

- Un router può venire a conoscenza di più di un percorso verso un prefisso

  ➔ **deve selezionarne uno**

- Quando un border router riceve un prefix advertisement utilizza le **policy locali** per decidere se **accettare** o **scartare** l'annuncio

Varie regole possibili:

1. Valore di preferenza locale: ***policy***
2. AS-PATH più breve
3. Router di NEXT-HOP più vicino
4. Altri criteri, inclusi quelli economici

Alcune risorse utili Online

- https://www.nro.net/
- https://bgp.he.net/
- https://stat.ripe.net/about/
- https://stat.ripe.net/widget/bgplay