Università di Bergamo

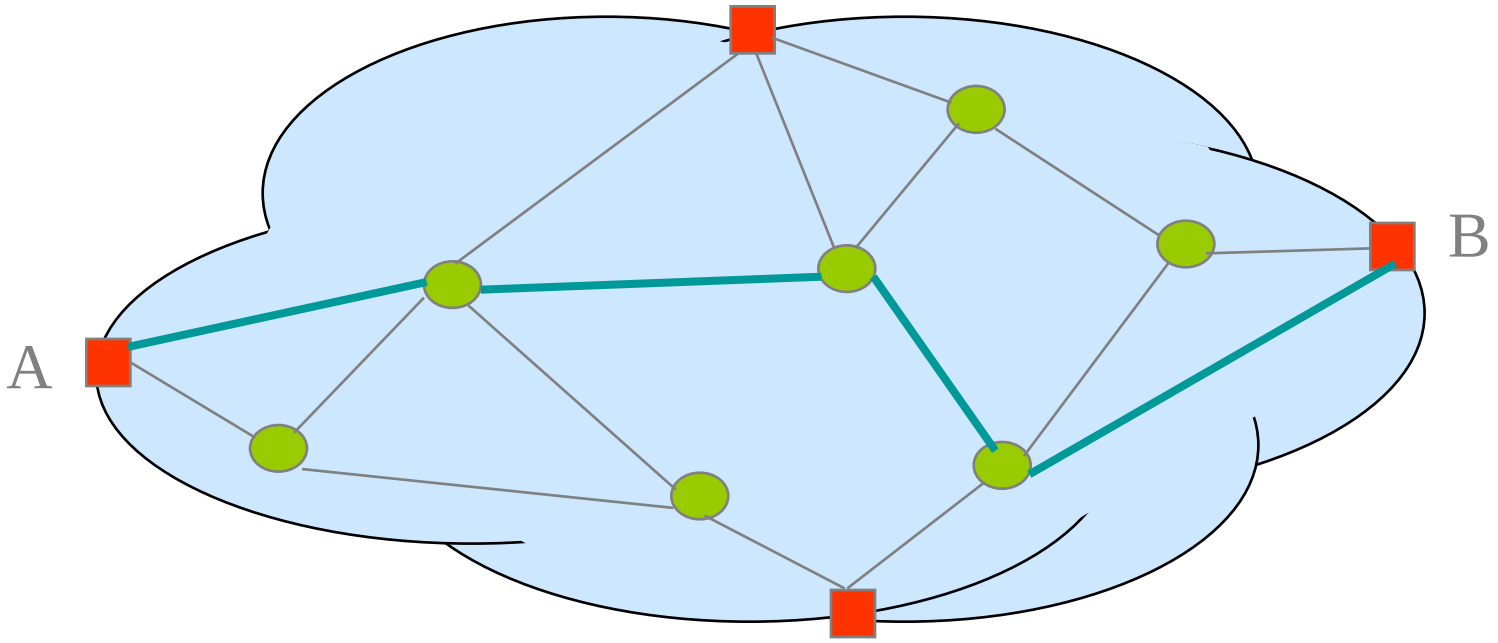
Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

6 - Routing

Architetture e Protocolli per Internet

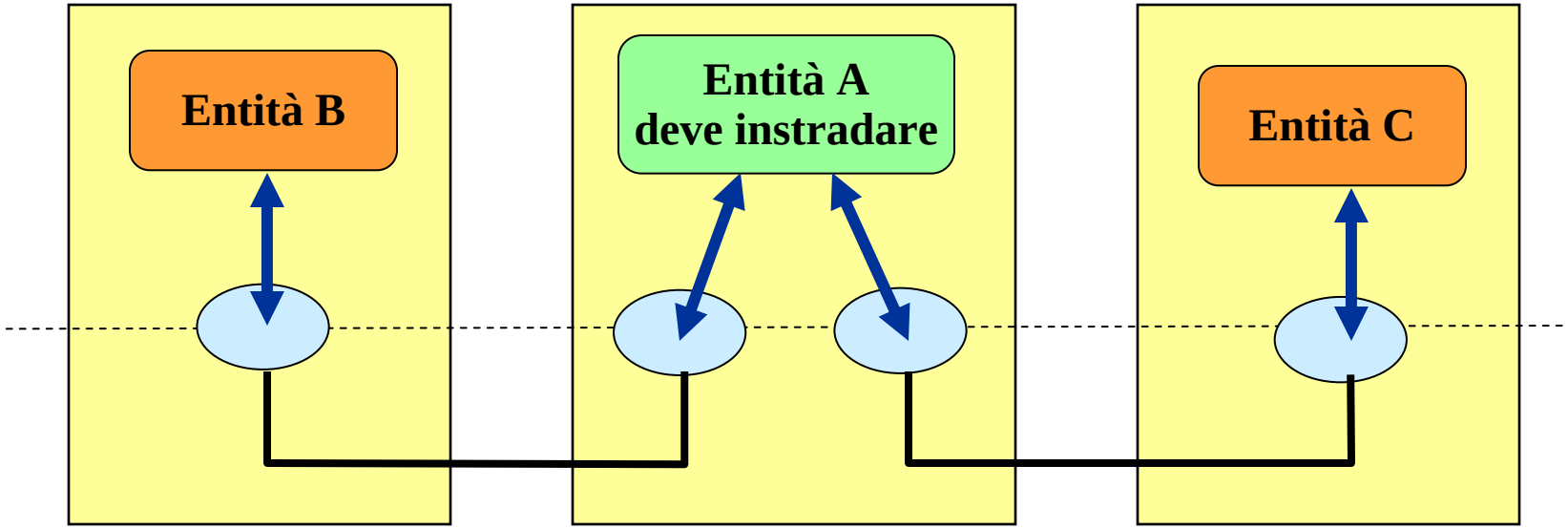
Routing

- L’instradamento è alla base della funzionalità di rete implementata dalle entità di livello 3 (OSI) dei nodi
- consente a due nodi A e B, non collegati direttamente, di comunicare tra loro mediante la collaborazione di altri nodi posti su un cammino nella rete che connette A e B



Routing

- Le entità di livello 3 sul cammino basano la commutazione (forwarding) verso il SAP d'uscita sulla base di un indirizzo (o di una etichetta) posta sul pacchetto
- La corrispondenza tra indirizzo e SAP d'uscita è mantenuta dal nodo in una tabella di routing



Routing

- La **politica di routing** è quella che definisce i criteri di scelta del cammino nella rete per i pacchetti che viaggiano tra un nodo di ingresso ed uno di uscita …
- …e dunque è quella che costruisce le tabelle di routing che vengono usate dai nodi per effettuare il forwarding
- Il tipo di rete (datagram, circuito virtuale) determina il tipo di tabelle da utilizzare e i gradi di libertà della politica di routing nella scelta dei cammini

Routing e Capacità di Rete

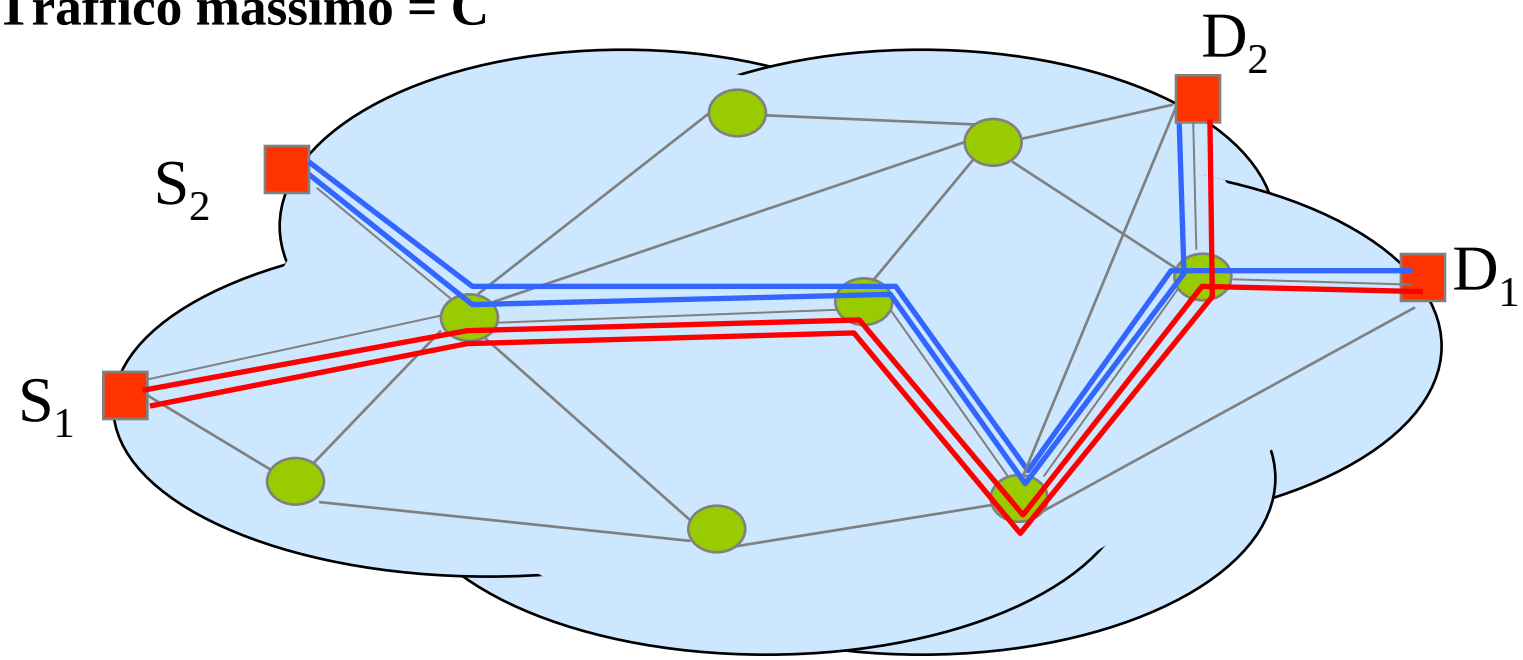
- **Nelle reti broadcast (come le LAN non switched) non vi sono nodi che effettuano instradamento ed il mezzo condiviso può essere usato a turno**
- **Il risultato è che il traffico massimo che può essere smaltito dalla rete (capacità) è al più pari alla capacità del canale**
- **Nelle reti magliate la trasmissione di un pacchetto non occupa tutte le risorse di rete e più canali e cammini possono essere usati in parallelo**
- **E' facile comprendere come in questo caso la politica di instradamento abbia un forte impatto sul traffico smaltibile dalla rete (detto *Capacità di Rete*)**

Routing e Capacità di Rete

- **Ad esempio:**
  - **se il traffico viene fatto passare da pochi cammini nella rete il traffico massimo sarà basso**

Capacità di tutti i link = C

Traffico massimo = C

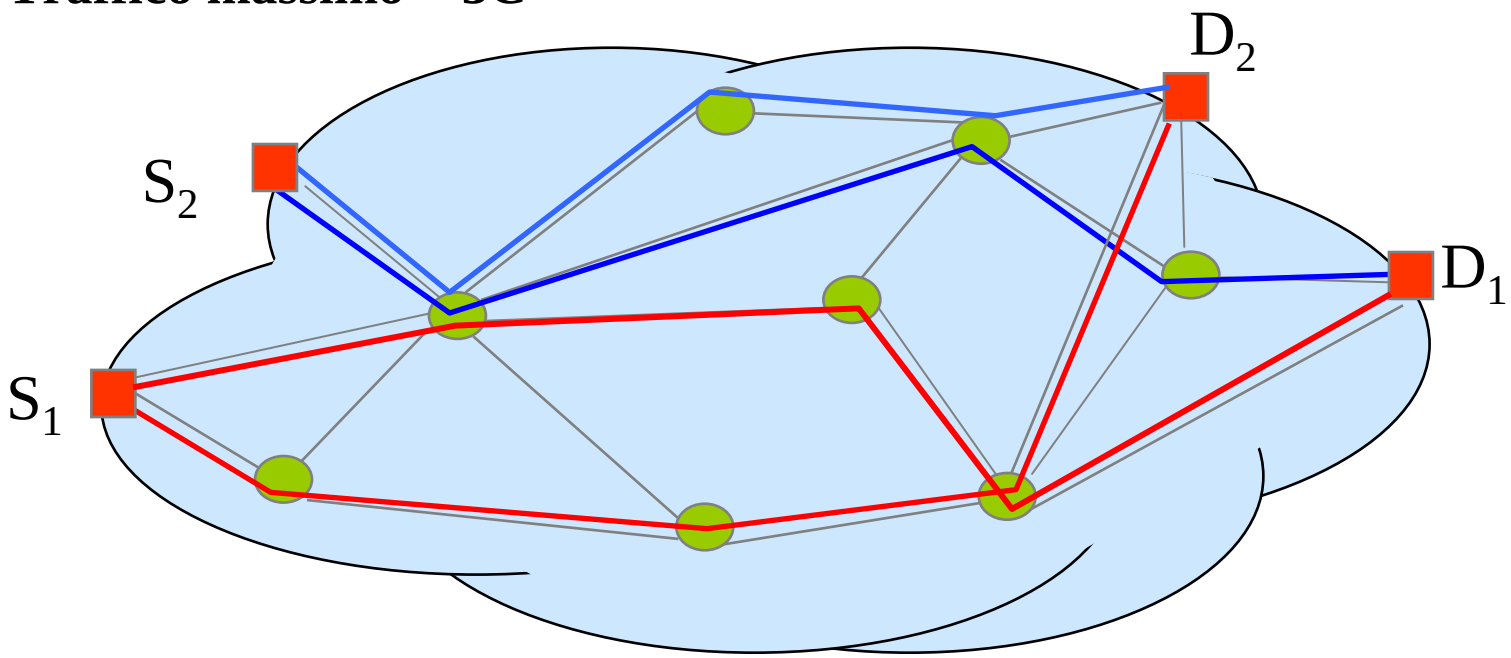


Routing e capacità

- **se invece si usano molti cammini ripartendo il carico il traffico massimo sarà elevato**

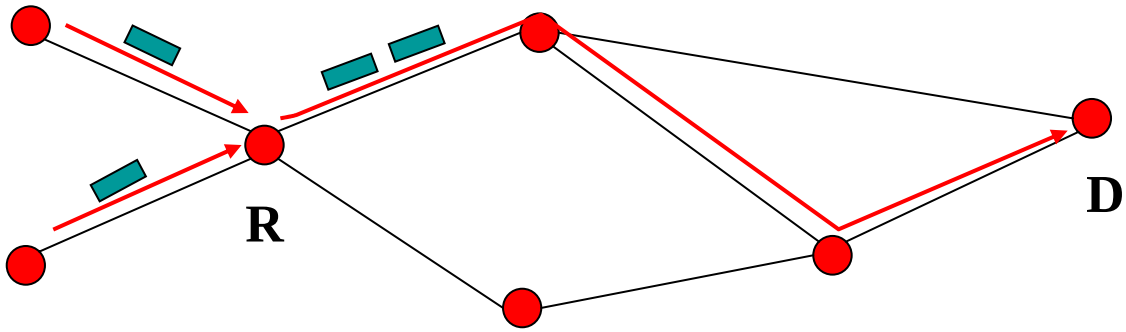
Capacità di tutti i link = C

Traffico massimo = 3C



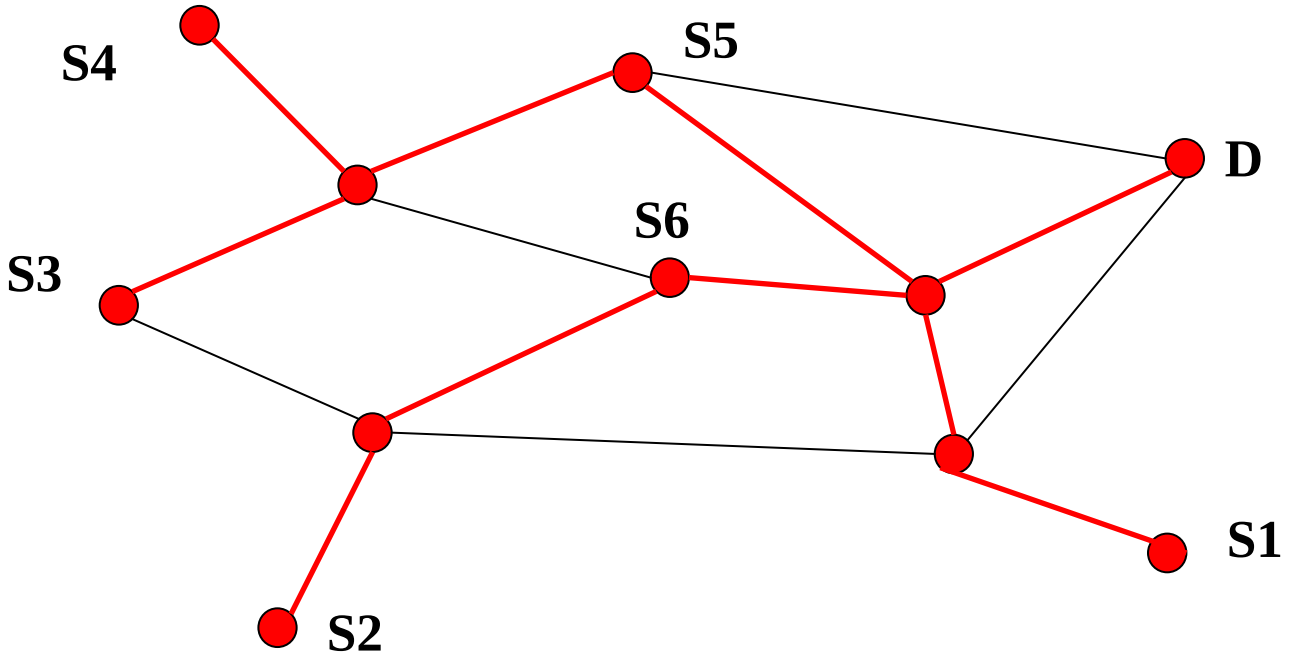
Politiche di routing per Internet

- **Il tipo di inoltro (forwarding) utilizzato dalle reti IP condiziona la scelta delle politiche di routing**
- **Ricordiamo che il forwarding IP è**
  - ***Basato sull'indirizzo di destinazione (destination-based …)***
  - ***Con inoltro al nodo successivo (…next-hop routing)***
- **Come conseguenza:**
  - ***I pacchetti diretti ad una stessa destinazione D che giungono in un router R seguono lo stesso percorso da R verso D indipendentemente dal link di ingresso in R***



Politiche di routing per Internet

- **Quindi il vincolo che ogni politica di routing deve soddisfare è che**
  - ***l'insieme dei cammini da ogni sorgente verso una destinazione D sia un albero, per ogni possibile destinazione D***



  - ***Non è dunque possibile instradare in modo indipendente ogni relazione di traffico (coppia sorgente-destinazione)***

Cammini minimi

- La politica di routing utilizzata sin dall'introduzione delle reti TCP/IP è basata sul calcolo dei cammini minimi
- Il calcolo è effettuato sul grafo che rappresenta la rete nel quale ad ogni arco è associato un peso opportunamente scelto (metrica)
- I motivi di questa scelta sono fondamentalmente:
  - i cammini minimi verso una destinazione formano un albero (l'albero dei cammini minimi);
  - esistono degli algoritmi di calcolo dei cammini minimi che possono essere eseguiti in modo distribuito nei nodi della rete.
- Dato un grafo e dei pesi associati agli archi il calcolo del cammino minimo si può ottenere con algoritmi di complessità polinomiale nel numero di nodi.

Richiami sui Grafi

- digrafo $G(N,A)$
  - N nodi
  - $A=\{(i,j),\ i \in N,\ j \in N\}$ archi (coppie ordinate di nodi)
- percorso: $(n_1, n_2, \ldots, n_l)$ insieme di nodi con $(n_i, n_{i+1}) \in A$
- cammino: percorso senza nodi ripetuti
- ciclo: percorso con $n_1 = n_l$
- digrafo connesso: per ogni coppia di nodi *i* e *j* esiste almeno un cammino da *i* a *j*
- digrafo pesato: d_{ij} peso associato all'arco $(i,j) \in A$
- lunghezza di un cammino $(n_1, n_2, \ldots, n_l)$:

  $d_{n1,\,n2} + d_{n2,n3} + \ldots + d_{n(l-1),\,nl}$

Problema del cammino minimo

Dato un digrafo *G(N,A)* e due nodi *i* e *j*, trovare il cammino di lunghezza minima tra tutti quelli che consentono di andare da *i* a *j*

- **il problema è di complessità polinomiale**

Proprietà:

se il nodo k è attraversato dal cammino minimo da i a j, il sotto-cammino fino a k è anch'esso minimo

Algoritmo di Bellman-Ford

- Ipotesi:
  - pesi degli archi sia positivi che negativi
  - non esiste alcun ciclo di lunghezza negativa
- Scopo:
  - trovare i cammini minimi tra un nodo (sorgente) e tutti gli altri nodi, oppure
  - trovare i cammini minimi da tutti i nodi ad un nodo (destinazione)

Algoritmo di Bellman-Ford

- **Variabili aggiornate nelle iterazioni:**
  - $D_i^{(h)}$ **lunghezza del cammino minimo tra il nodo 1 (sorgente) e il nodo *i* composto da un numero di archi** $\leq h$
- **Valori iniziali:**

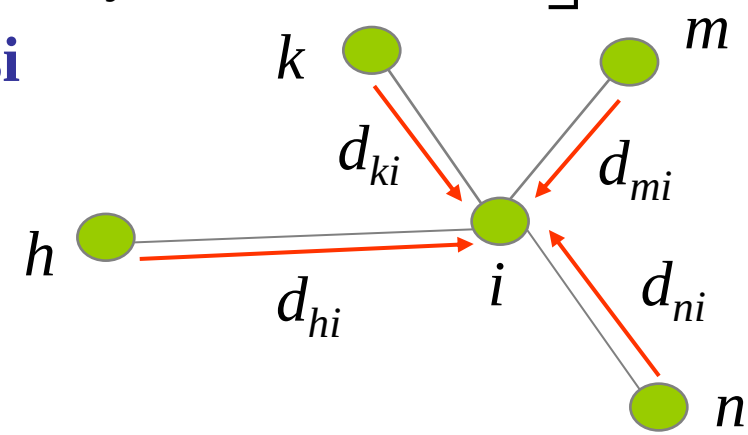
$$D_1^{(h)} = 0 \qquad \forall h$$

$$D_i^{(0)} = \infty \qquad \forall i \neq 1$$

- **Iterazioni:**

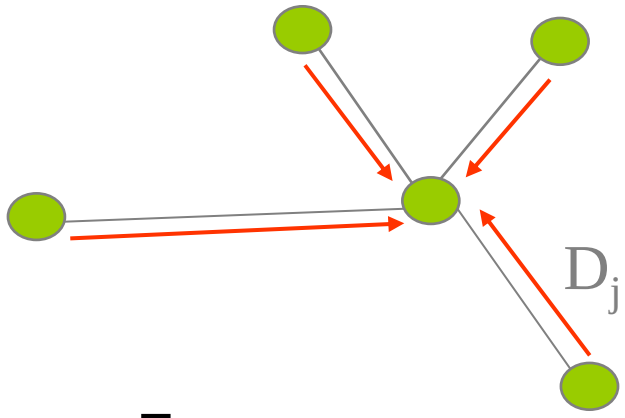
$$D_i^{(h+1)} = \min\left[D_i^{(h)}, \min_j \left(D_j^{(h)} + d_{ji} \right) \right]$$

- **l'algoritmo termina in N-1 passi**



Algoritmo di Bellman-Ford in forma distribuita

- Si dimostra che l'algoritmo converge in un numero finito di passi anche nel caso in cui viene implementato in modo distribuito
- Periodicamente i nodi inviano l'ultima stima del cammino minimo ai vicini e aggiornano la propria stima secondo il criterio delle iterazioni

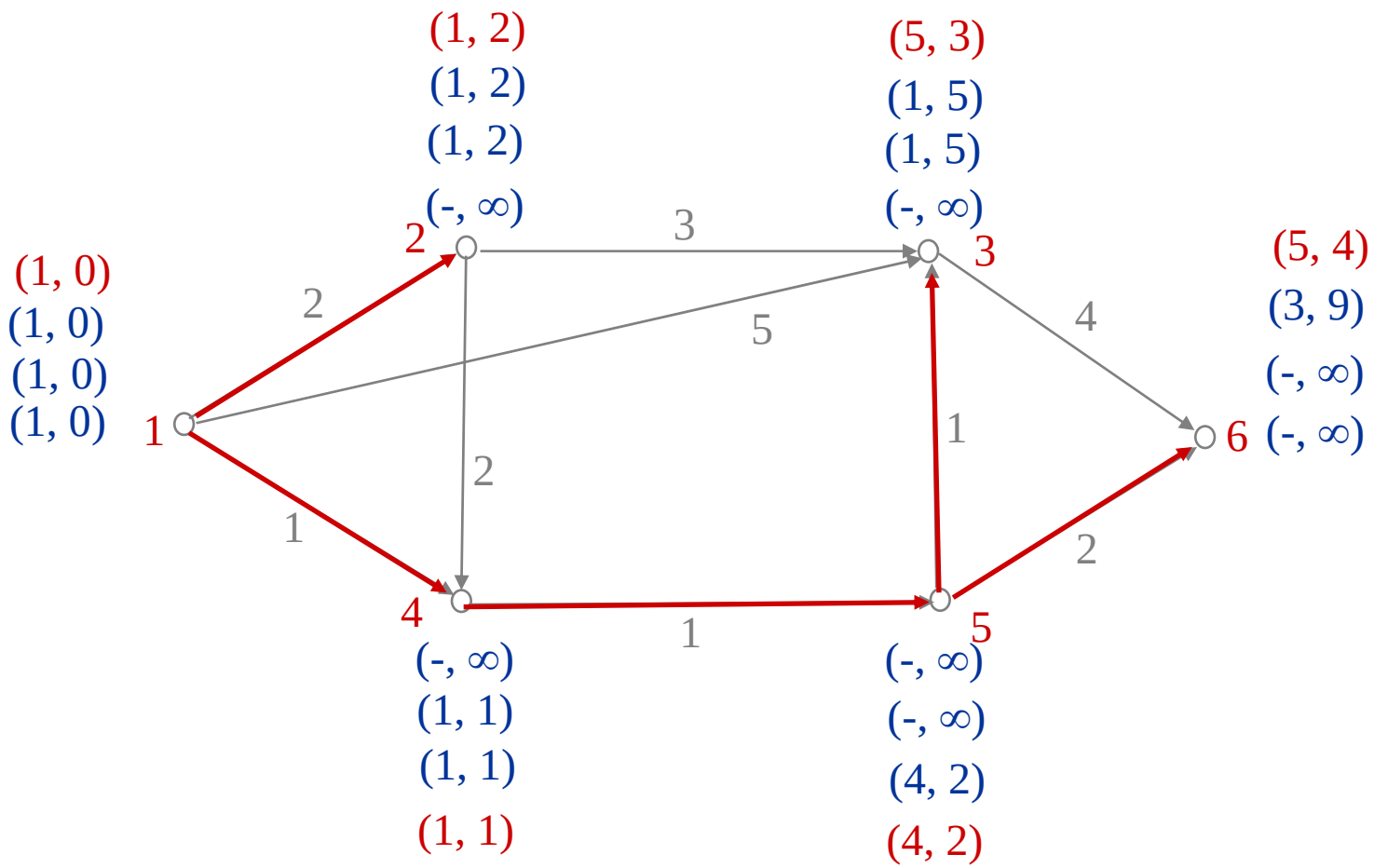


$$D_i := \min\left[D_i , \min_j \left(D_j + d_{ji} \right) \right]$$

Algoritmo di Bellman-Ford in pratica

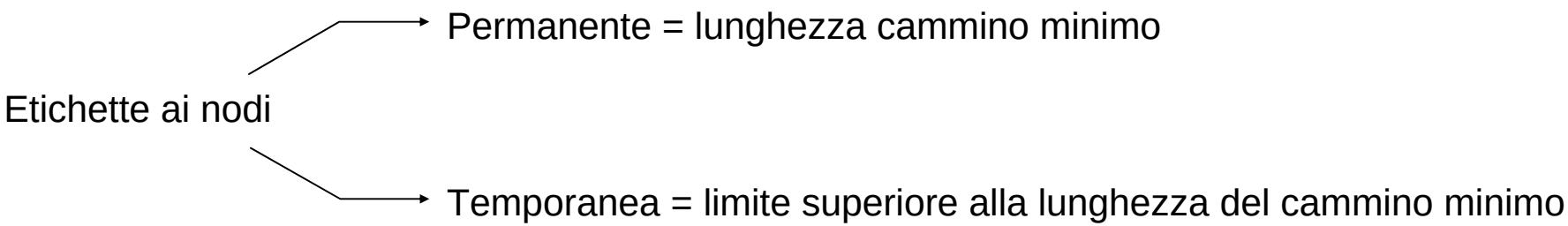
- Per poter applicare praticamente l'algoritmo in modo centralizzato si può procedere in questo modo:
- Si usano delle etichette per i nodi ***(n, L)*** dove ***n*** indica il nodo precedente lungo il cammino minimo ed ***L*** la sua lunghezza
- Le etichette vengono aggiornate guardando le etichette dei vicini (l'ordine non conta grazie alla proprietà dell'algoritmo distribuito)
- Quando le etichette non cambiano più si ricostruisce l'albero dei cammini minimi ripercorrendo le etichette

Esempio: Bellman-Ford



Algoritmo di Dijkstra

- **Ipotesi:**
  - **archi con pesi positivi**
- **Scopo:**
  - **trovare i cammini minimi tra un nodo 1 (sorgente) e tutti gli altri nodi**
- **Metodo di Dijkstra:**



Algoritmo di Dijkstra

1. **Inizializzazione:**
   1. **Etichetta permanente al nodo 1 = 0 : $u_1 = 0$**
   2. **Etichetta temporanea al nodo $i = l_{1i}$ $\forall i$ $u_i = l_{1i}$**
   3. **Si assume $l_{1i} = \infty$ se l'arco tra 1 e i non esiste**
2. **Trovare nodo *k* con etichetta temporanea minima**
   Dichiarare etichetta di *k* permanente
   Aggiornare etichette temporanee di tutti i nodi raggiungibili da *k*
   $u_j = \min (u_j , u_k + d_{kj})$ $\forall j$ con etichetta temporanea
3. **Se tutte le etichette sono permanenti STOP, altrimenti torna a 2.**

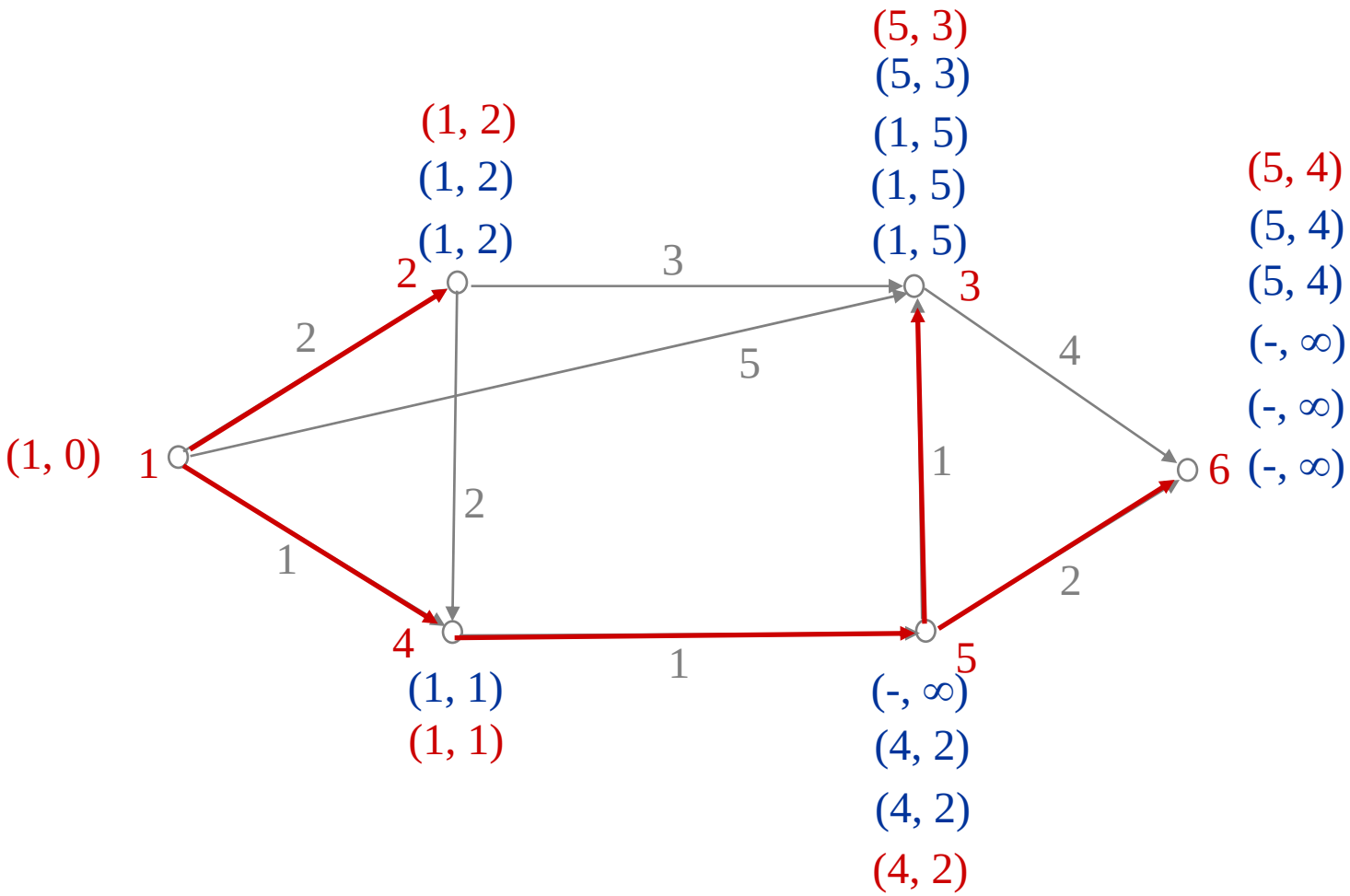
Algoritmi: complessità

- L'algoritmo di Bellman-Ford ha una complessità: $O(N^3)$
- L'algoritmo di Dijkstra ha una complessità: $O(N^2)$
- L'algoritmo di Dijkstra è in generale più conveniente

Algoritmi di Dijkstra in pratica

- Si applica lo stesso criterio di Bellman-Ford
- L'unica differenza consiste nella distinzione tra etichette temporanee e permanenti
- all'inizio l'unica etichetta permanente è quella della sorgente
- ad ogni iterazione l'etichetta temporanea con la lunghezza più corta diventa permanente

Esempio: Dijkstra



Protocolli di routing IP

Protocolli di Routing

- Con questo nome si indicano in genere due diverse funzionalità, anche se legate fra loro
  - lo scambio fra i router di informazioni di raggiungibilità
  - la costruzione delle tabelle di routing
- Formalmente il protocollo è solo la parte che descrive lo scambio di messaggi tra i router
- in realtà questo scambio è poi strettamente legato al modo con cui sono calcolate le tabelle di routing

Tabelle di Routing IP

- Le Tabelle di Routing IP sono costituite da un elenco di route
- Ogni route comprende:
  - rete di destinazione
  - netmask
  - next hop (first hop)
- Quindi il forwarding è fatto in generale
  - sulla base del solo indirizzo di destinazione
  - su un solo cammino
  - indicando solo il primo router sul cammino

Routing IP

- Il principio su cui si basa il routing IP è molto semplice
  - inviare i pacchetti sul cammino minimo verso la destinazione
  - la metrica su cui si calcolano i cammini minimi è generale
  - il calcolo viene eseguito in modo distribuito dai router mediante uno scambio di informazioni con gli altri router
  - nella tabella viene indicato solo il primo router sul cammino grazie alla proprietà secondo la quale anche i sotto-cammini di un cammino minimo sono minimi

Protocolli di Routing

- Per gestire lo scambio di informazioni tra i router ed eseguire il calcolo del cammino minimo esistono due grandi famiglie di protocolli

  - ➔ Distance Vector
  - ➔ Link State

Distance Vector

- l'informazione sulla raggiungibilità/connettività scambiata dai nodi è costituita dal Distance Vector (DV):

  [indirizzo destinatario, distanza]

  ➔ la distanza è una stima che il nodo possiede

- Il DV è inviato ai soli nodi adiacenti

- la stima delle distanze avviene tramite Bellman-Ford distribuito

Distance Vector

- Ogni nodo invia il DV

  ➔ periodicamente

  ➔ se il risultato di un ricalcolo differisce dal precedente

- Ogni nodo esegue il ricalcolo delle distanze se

  ➔ riceve un DV diverso da quello memorizzato in precedenza

  ➔ cade/nasce una linea attiva a cui è connesso

Ricalcolo: $D_j' = \min_k [D_k + d_{kj}]$

Processing dei messaggi Distance Vector

- Quando un nodo riceve un DV: ogni distanza che non ha valore infinito viene incrementata di 1. Quindi:

1. Se l'entry **non esiste** (nodo sconosciuto) e se la metrica relativa è **diversa da infinito**, setta come next-hop router per quella destinazione il router da cui è ricevuto il DV con la rispettiva distanza. Viene fatto partire un timer associato alla entry.

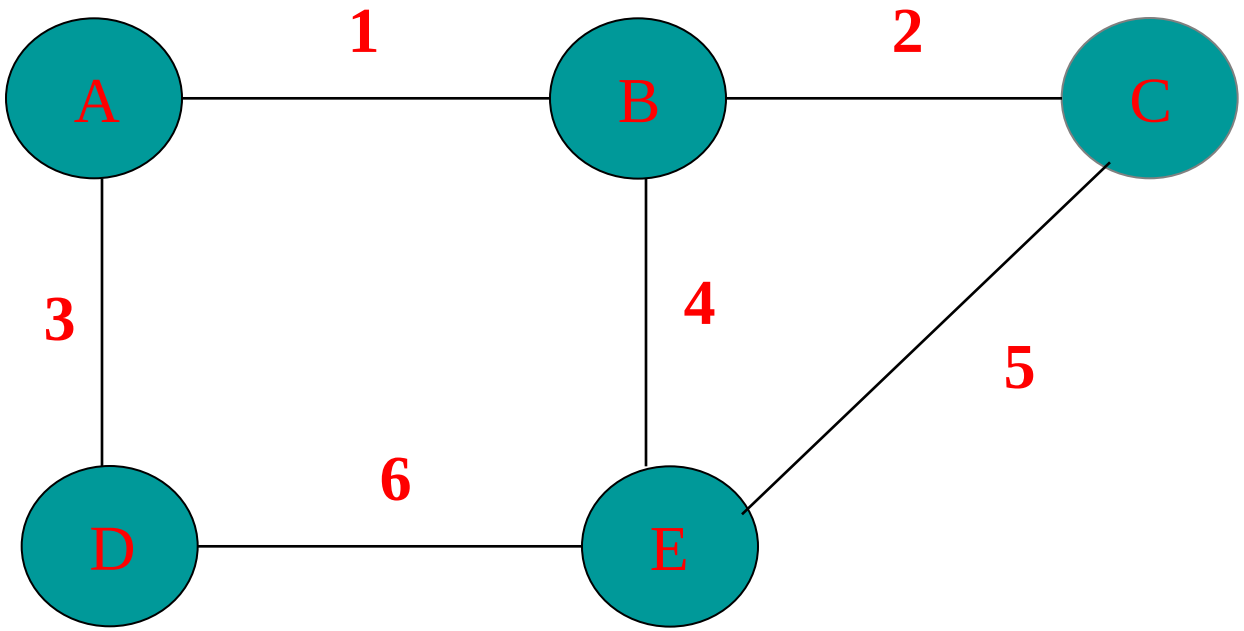
2. Se la entry **esiste già**, ma con una distanza **maggiore** a quella ricevuta, modifica next-hop router e distanza e fai partire timer.

3. Se la entry **esiste già** ed il next-router corrisponde al router che ha inviato il messaggio DV, modifica la entry se la distanza **differisce** da quella attuale. In ogni caso fai ripartire il timer.

4. In tutti gli altri casi: ignora il messaggio

Esempio: Distance Vector

- **Consideriamo la seguente rete:**



- ogni nodo (non differenziamo tra host e router) è identificato da un suo indirizzo (A,B,C,D o E)
- supponiamo che ogni link abbia costo 1

Esempio: Distance Vector

- **inizializziamo la rete attivando tutti i nodi contemporaneamente**
  - ☞ **procedura cold start**
- **ogni nodo ha delle informazioni iniziali permanenti (*conoscenze locali*), in particolare conosce il suo indirizzo e a quali link è direttamente connesso, non conosce gli altri nodi nella rete**
- **inizialmente le tabelle di routing contengono solo la entry del nodo, per esempio il nodo A**

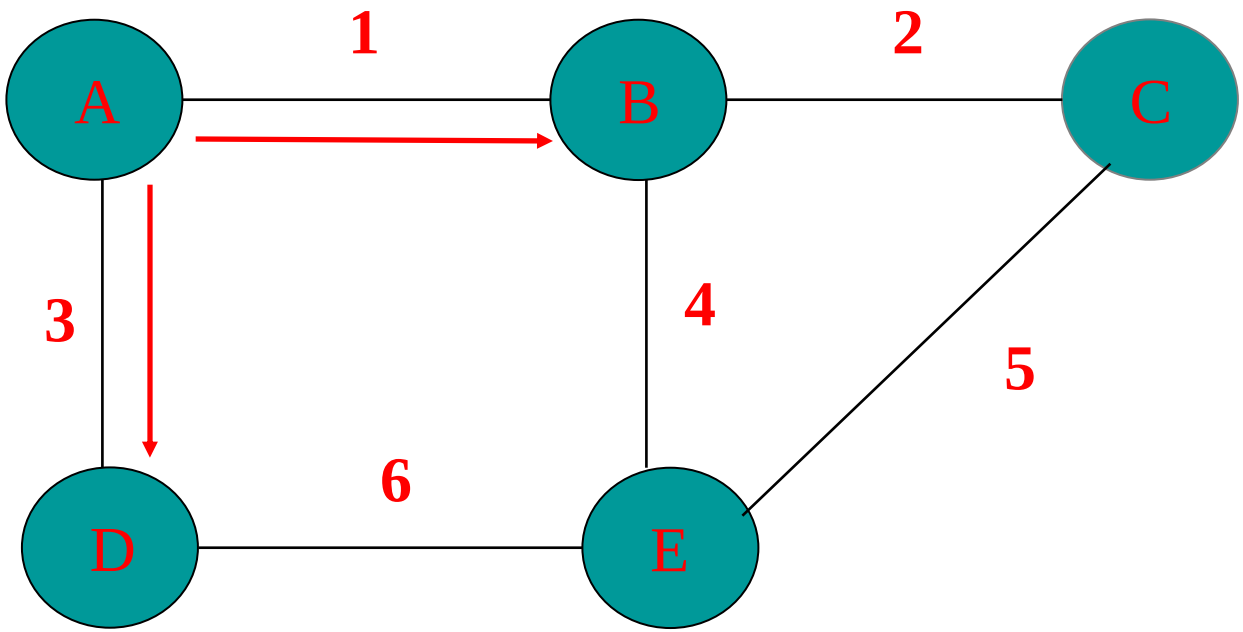
Da A verso	Link	Costo
A	locale	0

Esempio: Distance Vector

- da questa tabella il nodo A estrae il Distance Vector

  A=0

  e lo trasmette a tutti i vicini, cioè su tutti i link locali
- B e D ricevono l'informazione e allargano le loro conoscenze locali,



Esempio: Distance Vector

- il nodo B, dopo aver ricevuto il Distance Vector, aggiorna la distanza aggiungendo il costo del link locale trasformando il messaggio in A=1, lo confronta con le informazioni nella sua tabella di routing e vede che il nodo A non è conosciuto

Da B verso	Link	Costo
B	locale	0
A	1	1

- Anche il nodo D aggiorna in modo analogo la sua tabella dopo aver ricevuto il DV da A sul link 3

Da D verso	Link	Costo
D	locale	0
A	3	1

Esempio: Distance Vector

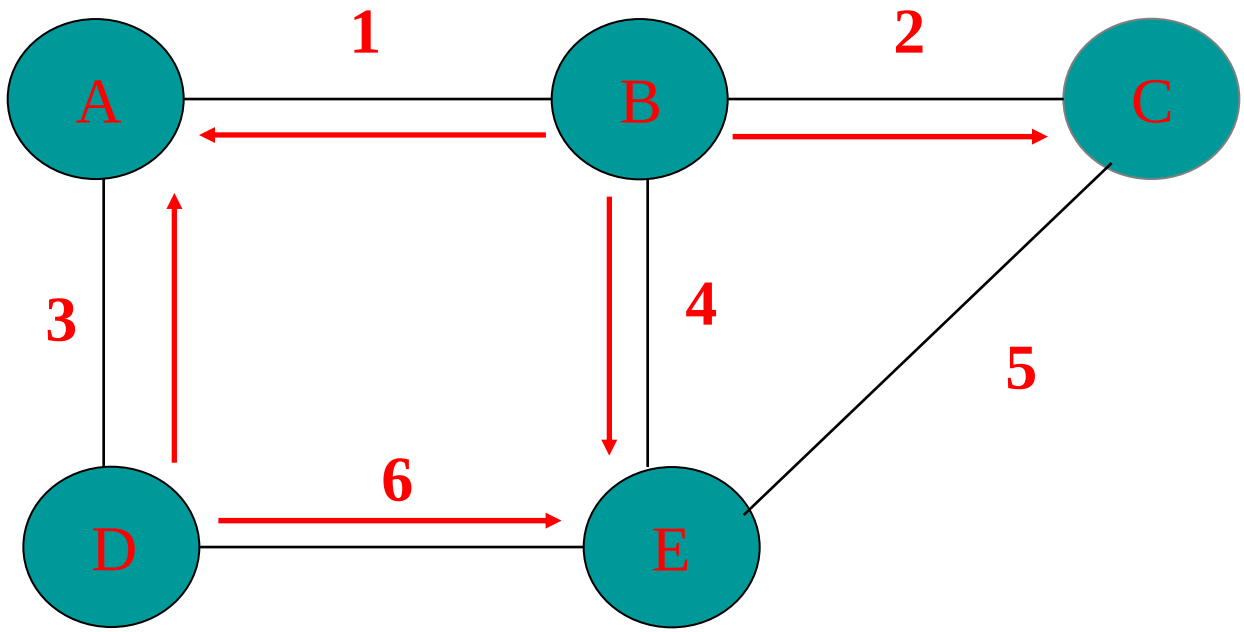
- il nodo B prepara il proprio DV

  B=0, A=1

  e lo trasmette su tutti i link locali

- Anche il nodo D prepara il DV e lo invia:

  D=0, A=1



Esempio: Distance Vector

- il messaggio di B viene ricevuto da A, C ed E mentre quello di D è ricevuto da A ed E,
- A riceve i due DV

  Da B: B=0, A=1

  Da D: D=0, A=1

  … e aggiorna la sua tabella

Da A verso	Link	Costo
A	local	0
B	1	1
D	3	1

Esempio: Distance Vector

- il nodo C riceve da B sul link 2 il DV

  B=0, A=1

  … e aggiorna la sua tabella:

Da C verso	Link	Costo
C	local	0
B	2	1
A	2	2

Esempio: Distance Vector

- il nodo E riceve da B sul link 4 il DV

  B=0, A=1

  e da D sul link 6 il DV

  D=0, A=1

  … e aggiorna la sua tabella di routing

  ✓ la distanza verso il nodo A utilizzando i link 4 e 6 è la stessa

Da E verso	Link	Costo
E	local	0
B	4	1
A	4	2
D	6	1

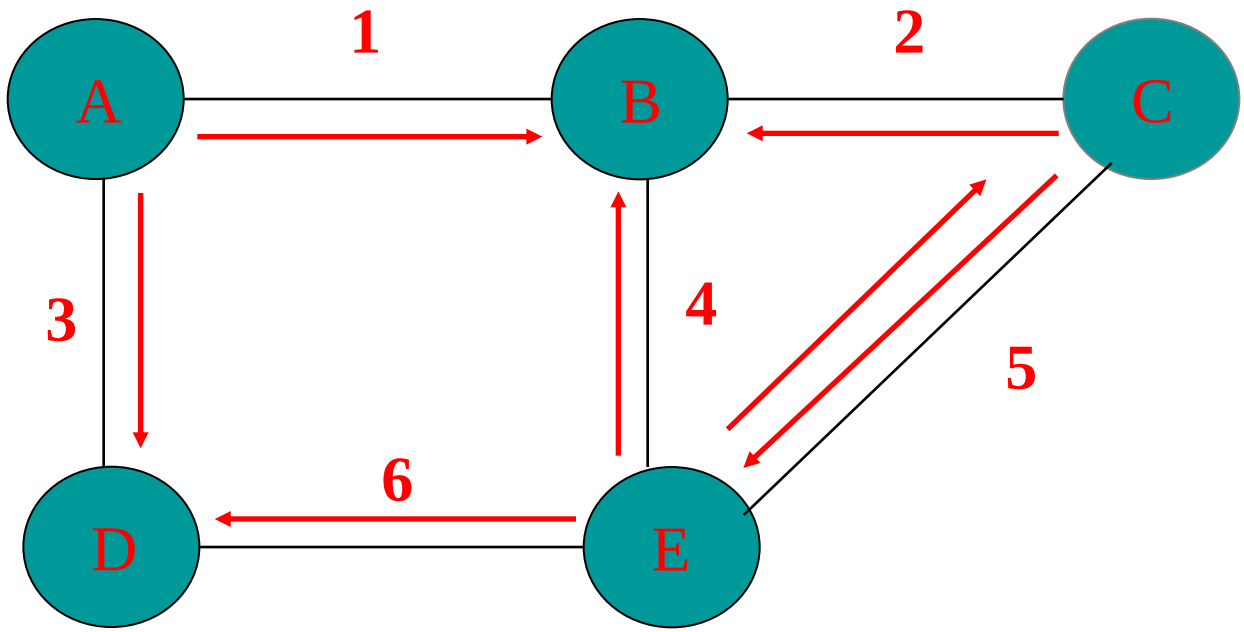
Esempio: Distance Vector

- i nodi A,C ed E hanno aggiornato le proprie tabelle di routing e trasmettono sui link locali i DV aggiornati

  nodo A: A=0, B=1, D=1

  nodo C: C=0, B=1, A=2

  nodo E: E=0, B=1, A=2, D=1



Esempio: Distance Vector

- i nodi B,D ed E ricevono i DV dai vicini e aggiornano le proprie tabelle

Da B verso	Link	Costo
B	local	0
A	1	1
D	1	2
C	2	1
E	4	1

Da D verso	Link	Costo
D	local	0
A	3	1
B	3	2
E	6	1

Da E verso	Link	Costo
E	local	0
B	4	1
A	4	2
D	6	1
C	5	1

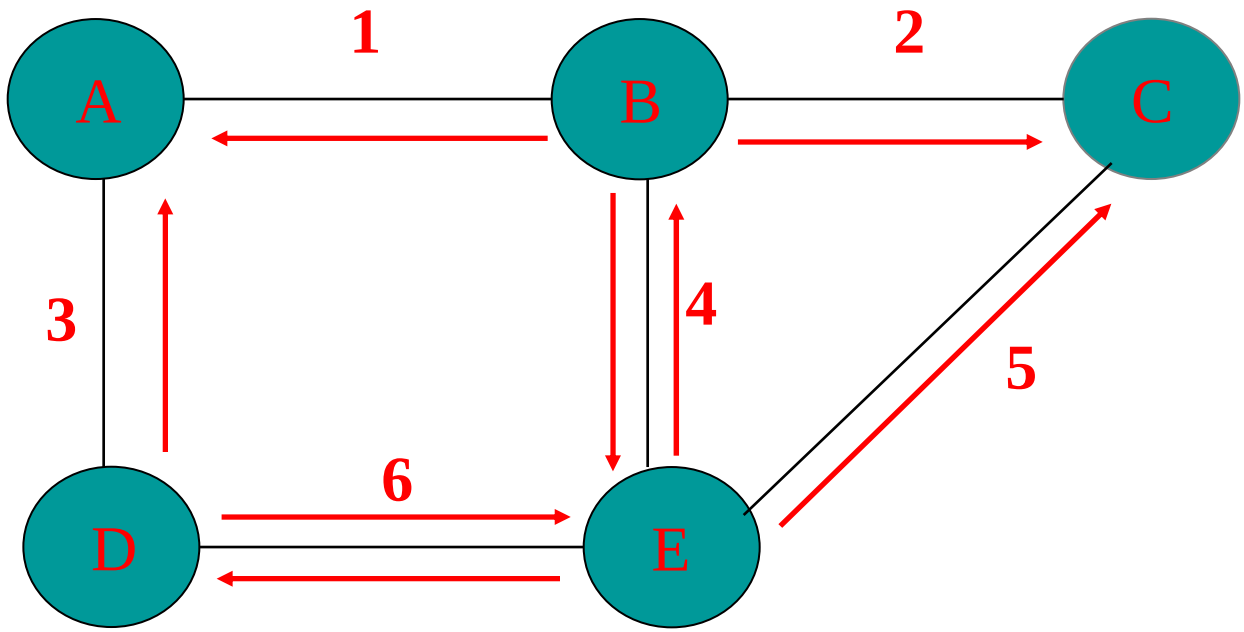
Esempio: Distance Vector

- i nodi B, D ed E trasmettono i nuovi DV sui link locali

  nodo B: B=0, A=1, D=2, C=1, E=1

  nodo D: D=0, A=1, B=2, E=1

  nodo E: E=0, B=1, A=2, D=1, C=1



Esempio: Distance Vector

- Tutti i nodi ricevono nuovi DV ma solo A, C e D aggiornano le tabelle

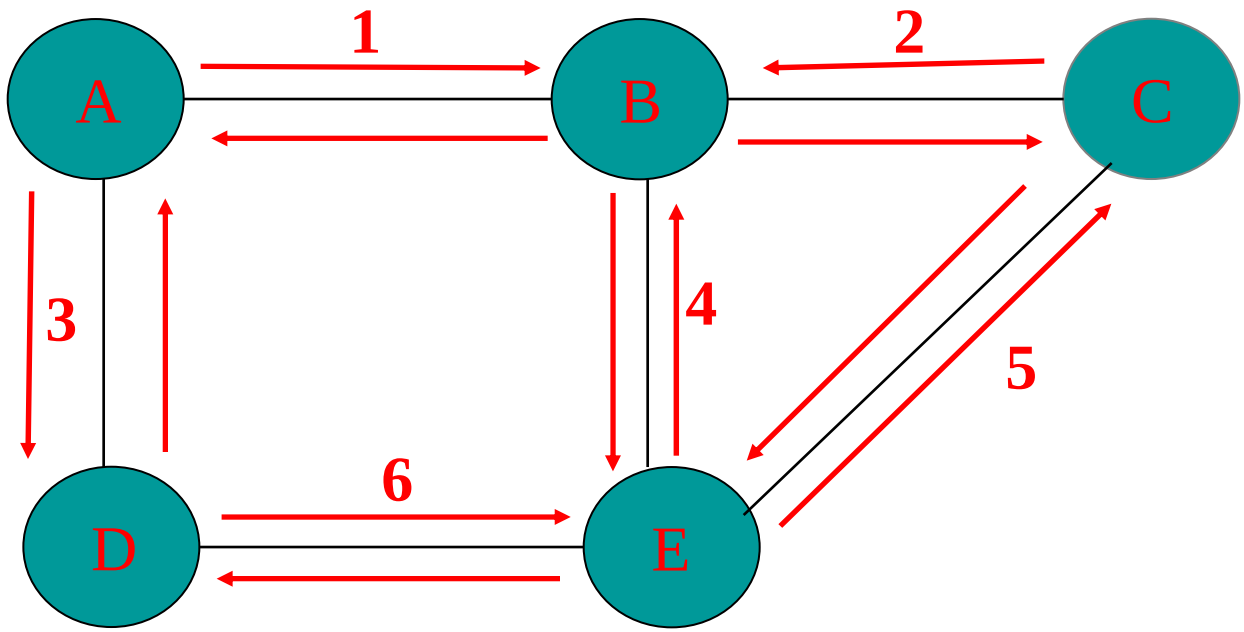
Da A verso	Link	Costo
A	local	0
B	1	1
D	3	1
C	1	2
E	1	2

Da C verso	Link	Costo
C	local	0
B	2	1
A	2	2
E	5	1
D	5	2

Da D verso	Link	Costo
D	local	0
A	3	1
B	3	2
E	6	1
C	6	2

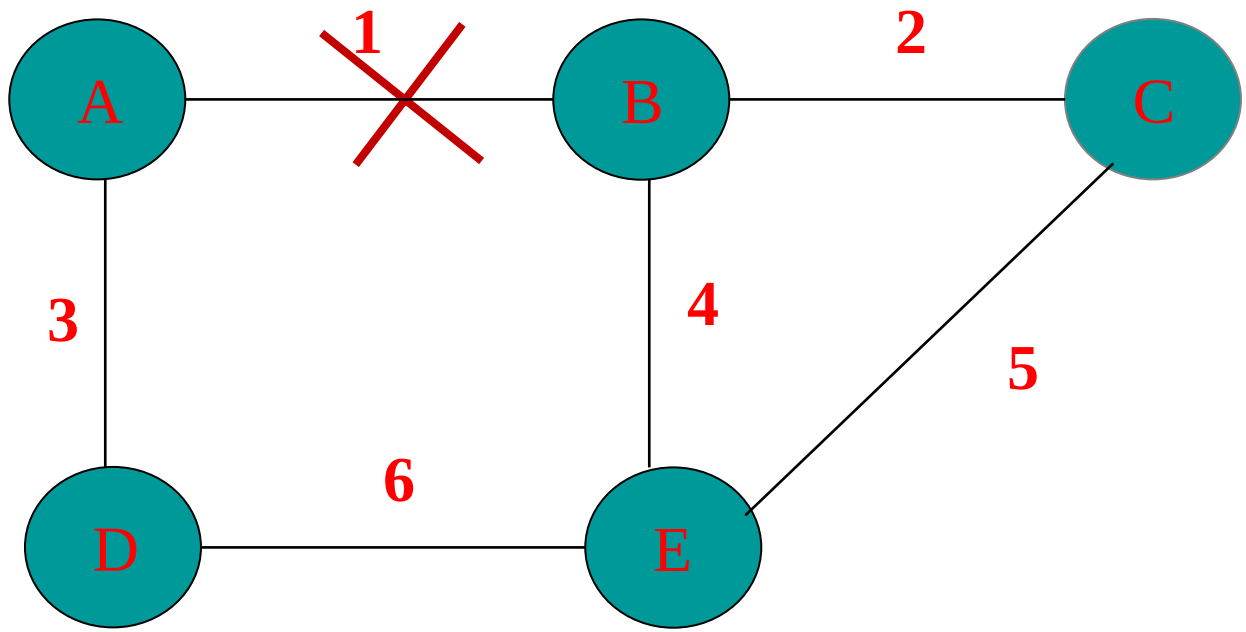
Esempio: Distance Vector

- l'algoritmo è arrivato a convergenza, i nodi trasmettono i nuovi DV sui link che però non provocano aggiornamenti nelle tabelle di routing degli altri nodi



Distance Vector: rottura del link 1

- **Vediamo come le tabelle di routing si aggiornano quando si rompe il link 1**



- i nodi A e B agli estremi del link 1 monitorano e riscontrano la rottura del link
- i nodi A e B aggiornano le proprie tabelle di routing assegnando costo infinito al link 1

Distance Vector: rottura del link 1

Da A verso	Link	Costo
A	local	0
B	1	1⇒inf
D	3	1
C	1	2⇒inf
E	1	2⇒inf

Da B verso	Link	Costo
B	local	0
A	1	1⇒inf
D	1	2⇒inf
C	2	1
E	4	1

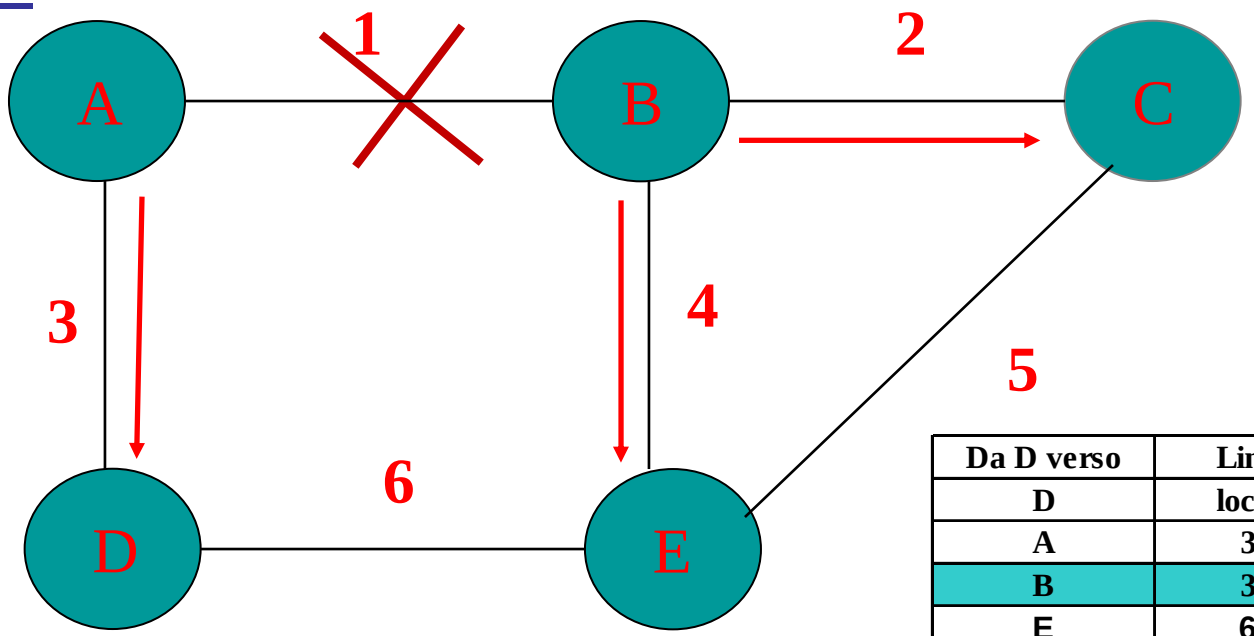
- trasmettono i nuovi DV

  nodo A: A=0, B=inf, D=1, C=inf, E=inf

  nodo B: B=0, A=inf, D=inf, C=1, E=1

Distance Vector: rottura del link 1

- il messaggio trasmesso da A viene ricevuto da D che confronta gli elementi con quelli presenti nella sua tabella di routing
- tutti i costi sono maggiori o uguali a quelli presenti nella tabella, ma siccome il link da cui riceve il messaggio (link 3) è quello che utilizza per raggiungere il nodo B, aggiorna la tabella



Da D verso	Link	Costo
D	local	0
A	3	1
B	3	2
E	6	1
C	6	2

Distance Vector: rottura del link 1

Da D verso	Link	Costo
D	local	0
A	3	1
B	3	2⇒inf
E	6	1
C	6	2

- Anche i nodi C ed E aggiornano le tabelle

Da C verso	Link	Costo
C	local	0
B	2	1
A	2	2⇒inf
E	5	1
D	5	2

Da E verso	Link	Costo
E	local	0
B	4	1
A	4	2⇒inf
D	6	1
C	5	1

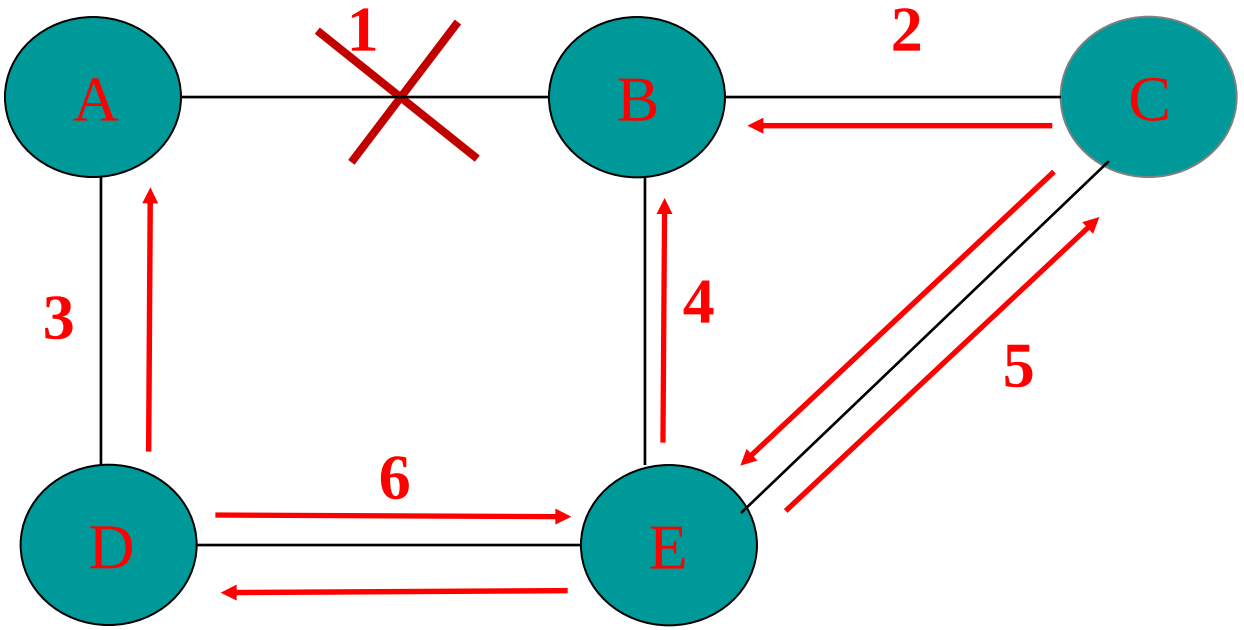
Distance Vector: rottura del link 1

- i nodi D, C ed E trasmettono il loro DV

  nodo D: D=0, A=1, B=inf, E=1, C=2

  nodo C: C=0, B=1, A=inf, E=1, D=2

  nodo E: E=0, B=1, A=inf, D=1, C=1



Distance Vector: rottura del link 1

- questi messaggi aggiornano le tabelle di routing dei nodi A, B, D ed E

Da A verso	Link	Costo
A	local	0
B	1	inf
D	3	1
C	1⇒3	inf⇒3
E	1⇒3	inf⇒2

Da B verso	Link	Costo
B	local	0
A	1	inf
D	1⇒4	inf⇒2
C	2	1
E	4	1

Da D verso	Link	Costo
D	local	0
A	3	1
B	3⇒6	inf⇒2
E	6	1
C	6	2

Da E verso	Link	Costo
E	local	0
B	4	1
A	4⇒6	inf⇒2
D	6	1
C	5	1

Distance Vector: rottura del link 1

- I nodi A,B,D ed E trasmettono i nuovi DV
  - nodo A: A=0, B=inf, D=1, C=3, E=2
  - nodo B: B=0, A=inf, D=2, C=1, D=1
  - nodo D: D=0, A=1, B=2, E=1, C=2
  - nodo E: E=0, B=1, A=2, D=1, C=1
- A, B e C aggiornano le tabelle

Da A verso	Link	Costo
A	local	0
B	1⇒3	inf⇒3
D	3	1
C	3	3
E	3	2

Da B verso	Link	Costo
B	local	0
A	1⇒4	inf⇒3
D	4	2
C	2	1
E	4	1

Da C verso	Link	Costo
C	local	0
B	2	1
A	2⇒5	inf⇒3
E	5	1
D	5	2

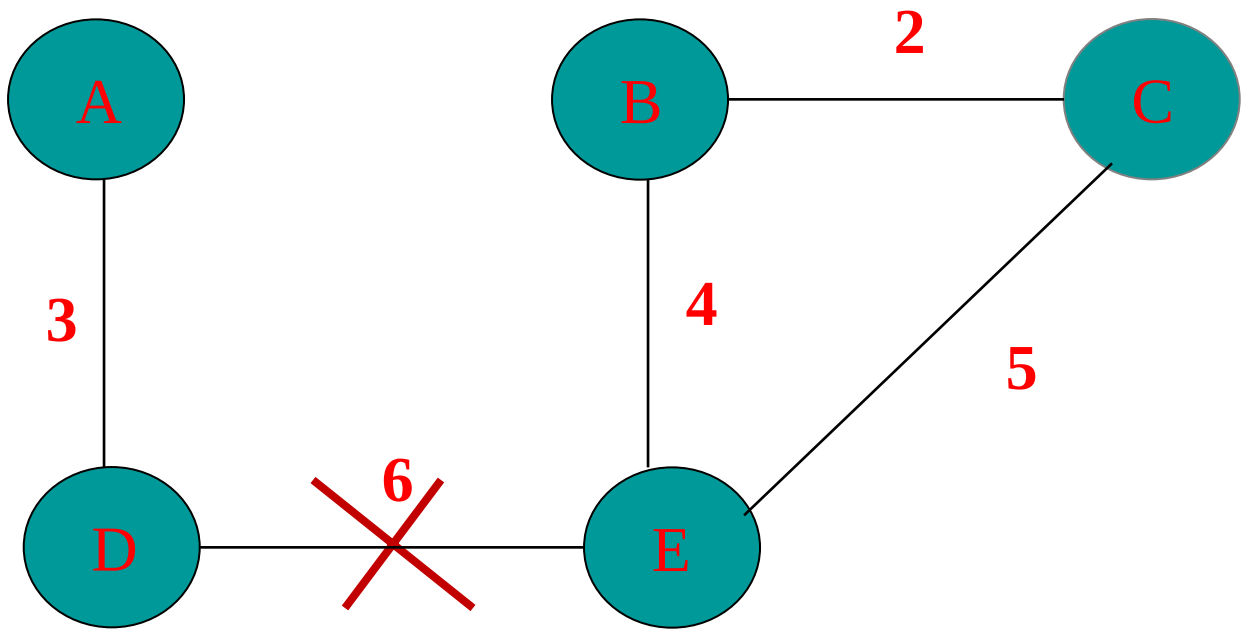
- l'algoritmo è arrivato a convergenza !!!

Distance Vector: caratteristiche

- Vantaggi:
  - molto facile da implementare
- Svantaggi:
  - Problema della velocità di convergenza
  - limitato dal nodo più lento
  - dopo un cambiamento possono sussistere dei cicli (loop) per un tempo anche lungo
  - difficile mantenere comportamento stabile su reti grandi
  - Problema del *counting to infinity*

Distance Vector: counting to infinity

- **Supponiamo che si rompa anche il link 6**



- guardiamo cosa succede tra i nodi A e D isolati dal resto della rete

Distance Vector: counting to infinity

- **il nodo D si accorge della rottura del link 6 e aggiorna la sua tabella di routing**

Da D verso	Link	Costo
D	local	0
A	3	1
B	6	2⇒inf
E	6	1⇒inf
C	6	2⇒inf

- se il nodo D ha l'opportunità di trasmettere immediatamente il nuovo Distance Vector sul link 3, il nodo A aggiorna immediatamente la sua tabella di routing e riconosce che l'unico nodo raggiungibile è D.

Da A verso	Link	Costo
A	local	0
B	3	3
D	3	1
C	3	3
E	3	2

Distance Vector: counting to infinity

- se invece il nodo A trasmette il suo DV prima che lo faccia D

  nodo A: A=0, B=3, D=1, C=3, E=2

  il nodo D aggiorna la sua tabella

Da D verso	Link	Costo
D	local	0
A	3	1
B	6⇒3	inf⇒4
E	6⇒3	inf⇒3
C	6⇒3	inf⇒4

- si installa un loop tra i nodo A e D e non c'è modo di convergere naturalmente verso uno stato stabile
- ad ogni step le distanze verso i nodi B, C ed E si incrementano di 2 ☛counting to infinity

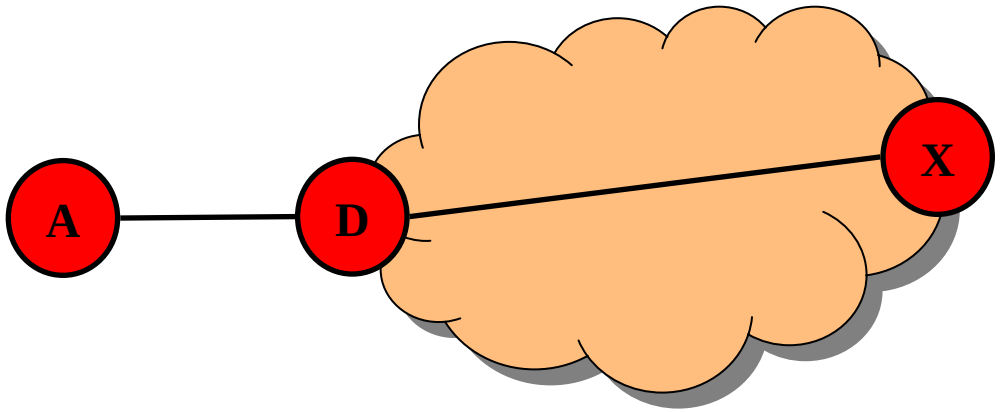
Counting to infinity: rimedi

- **Hop Count Limit:**
- **Il counting to infinity termina se si utilizza la convenzione di rappresentare l'infinito mediante un valore finito**
  - **tale valore deve essere maggiore del percorso più lungo nella rete**
  - **quando la distanza raggiunge tale valore viene posta ad infinito e il nodo non raggiungibile**
- **Durante il periodo di counting to infinity la rete si trova in uno stato intermedio in cui:**
  - **i pacchetti sono in loop**
  - **il link diventa congestionato**
  - **alcuni pacchetti, compresi i messaggi di routing, possono essere persi a causa della congestione**

☛ **la convergenza verso uno stato stabile è lenta**

Counting to infinity: rimedi

- **Split-Horizon:**
  - **se il nodo A manda a D i pacchetti destinati al nodo X, non ha senso che A annunci a D la raggiungibilità di X nel suo Distance Vector**

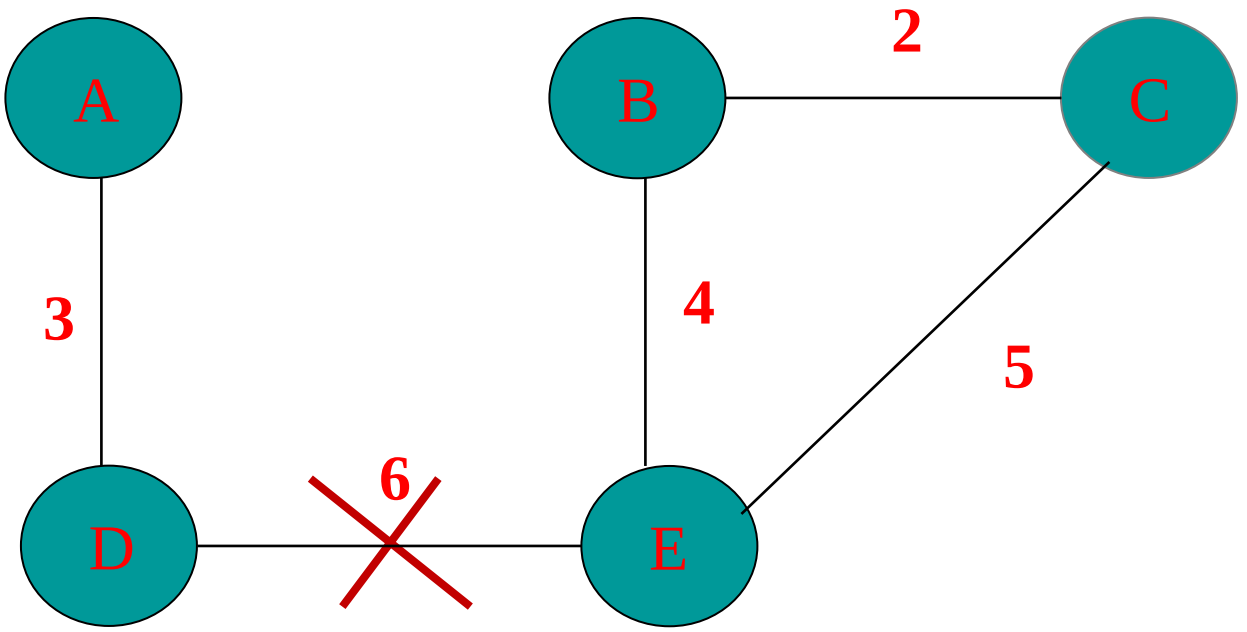


  - **il nodo A <u>non annuncia</u> a D con quale costo raggiunge X**

Distance Vector: Split Horizon

- Quindi il nodo A manda messaggi di routing *diversi* sui vari link locali
- Split Horizon esiste in due versioni:
  - forma semplice: il nodo omette nel messaggio ogni informazione sulle destinazioni che raggiunge tramite quel link
  - con Poisonous Reverse: il nodo include nel messaggio tutte le destinazioni ma pone a distanza infinita quelle raggiungibili tramite quel link
    - ✓ questo meccanismo elimina il counting to infinity dell'esempio precedente
- non funziona con certe topologie

Distance Vector: Split Horizon



- quando il link 6 si rompe le tabelle di routing dei nodi B,C e E contengono, tra le altre, le seguenti righe:

Da	Link	Costo
B verso D	4	2
C verso D	5	2
E verso D	6	1⇒inf

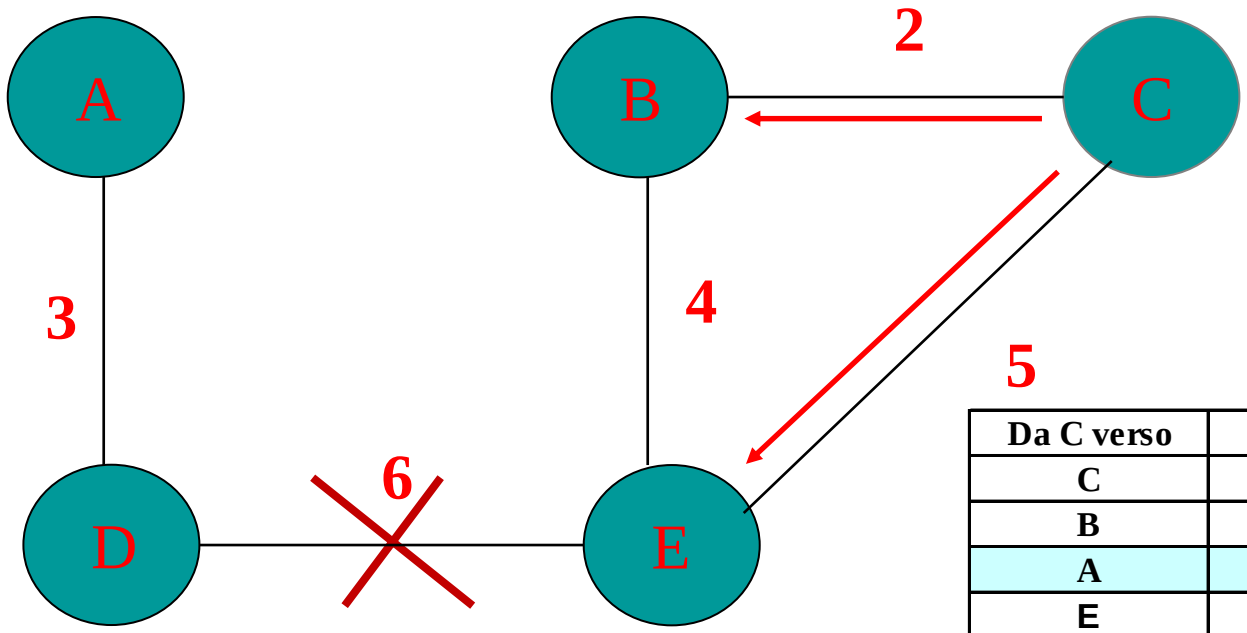
Distance Vector: Split Horizon

- il nodo E comunica sui link 4 e 5 che la distanza da D è ora infinita (D=inf)
- supponiamo che il messaggio sia ricevuto da B mentre a causa di un errore non sia ricevuto da C

Da	Link	Costo
B verso D	4	2⇒inf
C verso D	5	2
E verso D	6	inf

Distance Vector: Split Horizon

- il nodo C trasmette il DV, utilizzando lo Split Horizon con Poisonous Reverse. Trametterà dunque:
  - al nodo E: C=0, B=1, A=inf, E=inf, D=inf
    - ✓ sul link 5 che vede il nodo D con costo infinito
  - al nodo B: C=0, B=inf, A=3, E=1, D=2
    - ✓ sul link 2 che vede il nodo D con costo 2



Da C verso	Link	Costo
C	local	0
B	2	1
A	5	3
E	5	1
D	5	2

Distance Vector: Split Horizon

- il nodo B aggiorna la sua tabella di routing e utilizzando lo Split Horizon Poisonous Reverse trasmette:
  - D=inf sul link 2
  - D=3 sul link 4
- nei nodi B,C ed E ora avremo

Da	Link	Costo
B verso D	4⇒2	inf⇒3

Da	Link	Costo
B verso D	4⇒2	inf⇒3
C verso D	5	2
E verso D	6⇒4	inf⇒4

- si forma un loop tra i nodi B,C ed E fino a quando i valori di costo superano la soglia e sono posti ad infinito
  - ✓si ripresenta il fenomeno del counting to infinity

Counting to infinity: rimedi

- Utilizzo dei contatori (Hold down)
  - Supponiamo che ogni nodo G invii un messaggio DV a tutti i suoi vicini ogni 30 secondi.
  - Supponiamo inoltre che, in un certo nodo N, il next-hop attualmente usato per raggiungere la destinazione D sia G.
  - Se N non riceve nessun messaggio da G per 180 secondi ($T_{invalid}$), può assumere che o G si è guastato o che è divenuto irraggiungibile.
  - Dopo un tempo T_{flush} (240 s), la route è cancellata
  - Il tempo tra $T_{invalid}$ e T_{flush} deve essere tarato in modo che l'informazione relativa ad un cambiamento (guasto) si propaghi nella rete

Counting to infinity: rimedi

- …
  - le route non più valide sono annunciate con distanza infinita (valore della soglia, 16 in RIPv1)
  - i nodi che ricevono un annuncio con distanza infinita mettono la route in hold-down (non valida)
- Triggered Updates
  - I cambiamenti di topologia sono annunciati immediatamente e distinti dagli altri
  - aumenta la velocità di convergenza e fa scoprire prima i guasti

Link State

- Ogni nodo impara a conoscere i nodi e le destinazioni sue adiacenti, e le relative distanze per raggiungerle
- Ogni nodo invia a tutti gli altri nodi (flooding) queste informazioni mediante dei Link State Packet (LSP)
- Tutti i nodi si costruiscono un database di LSP e una mappa completa della topologia della rete
- Sulla base di questa informazione vengono calcolati i cammini minimi verso tutte le destinazioni (ad esempio con Dijkstra)

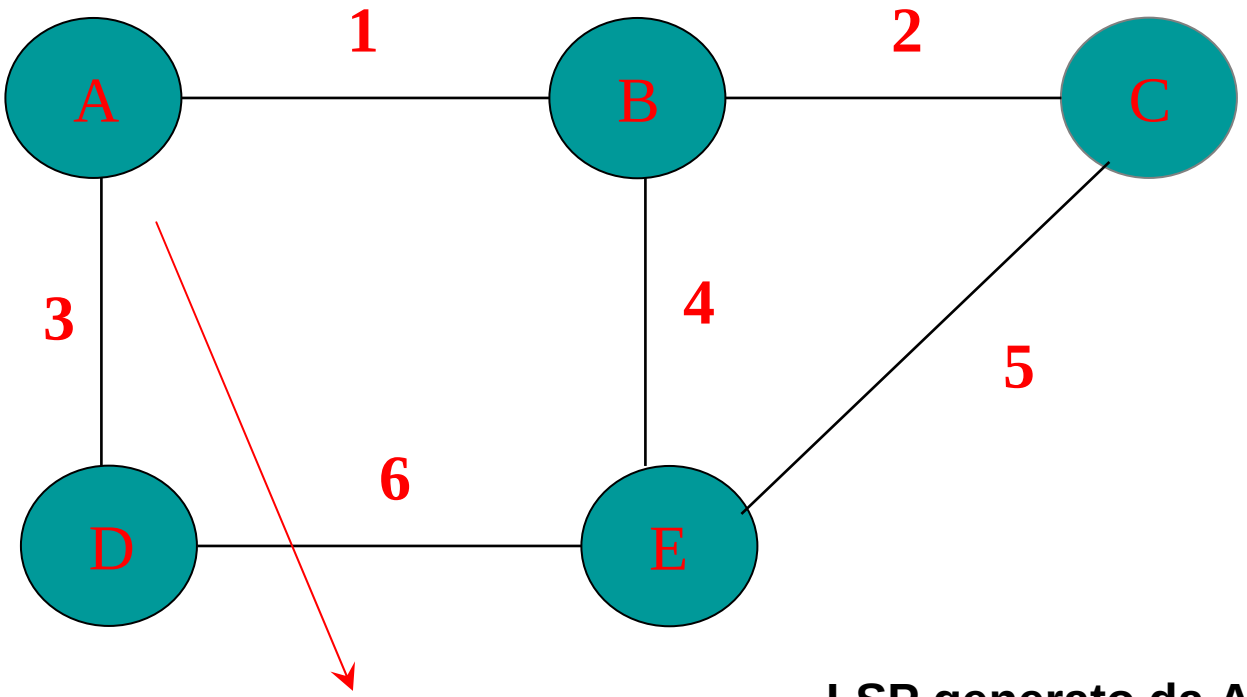
Link State

- **Vantaggi:**
  - **più flessibile in quanto ogni nodo ha una mappa completa della rete (routing ottimale, source routing, multipath routing)**
  - **non è necessario inviare l'informazione (LSP) periodicamente ma solo dopo un cambiamento. Inoltre tale informazione è inviata solo in modo incrementale (solo le entry che sono cambiate)**
  - **tutti i nodi vengono subito informati dei cambiamenti (in particolare topologici)**

Link State

- **Svantaggi:**
  - **è necessario un protocollo dedicato a mantenere l'informazione sui vicini (Hello)**
  - **è necessario l'utilizzo del flooding**
  - **è necessario un riscontro dei pacchetti di routing inviati**
  - **Più difficile da implementare**

Link State: esempio



LSP generato da A

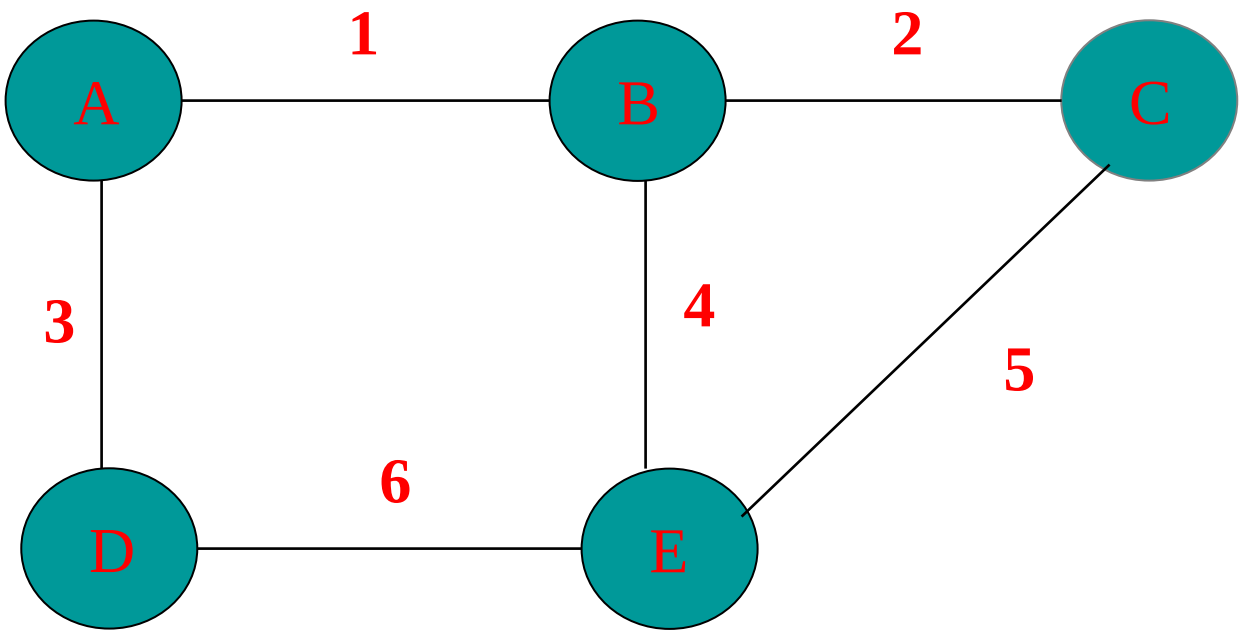
Da	Verso	Link	Costo	Seq.Number
A	B	1	1	1
A	D	3	1	1

Flooding

- Ogni pacchetto in arrivo viene ritrasmesso su tutte le uscite eccetto quella da cui è stato ricevuto
- occorre prevenire i loop e la conseguente generazione incontrollata di traffico
- numero di sequenza (SN) + database degli SN ricevuti in ogni nodo: i pacchetti non vengono ritrasmessi una seconda volta
- contatore di hop (come TTL di IP)

Esempio: Link State

- Ogni nodo ha un database (archivio degli LSP) in cui è descritta una mappa della rete



Esempio: Link State

- la rete è rappresentata dal database

Da	Verso	Link	Costo	Sequence Number
A	B	1	1	1
A	D	3	1	1
B	A	1	1	1
B	C	2	1	1
B	E	4	1	1
C	B	2	1	1
C	E	5	1	1
D	A	3	1	1
D	E	6	1	1
E	B	4	1	1
E	C	5	1	1
E	D	6	1	1

- ogni nodo può calcolare il percorso più breve verso tutti gli altri nodi

All'arrivo di un LSP

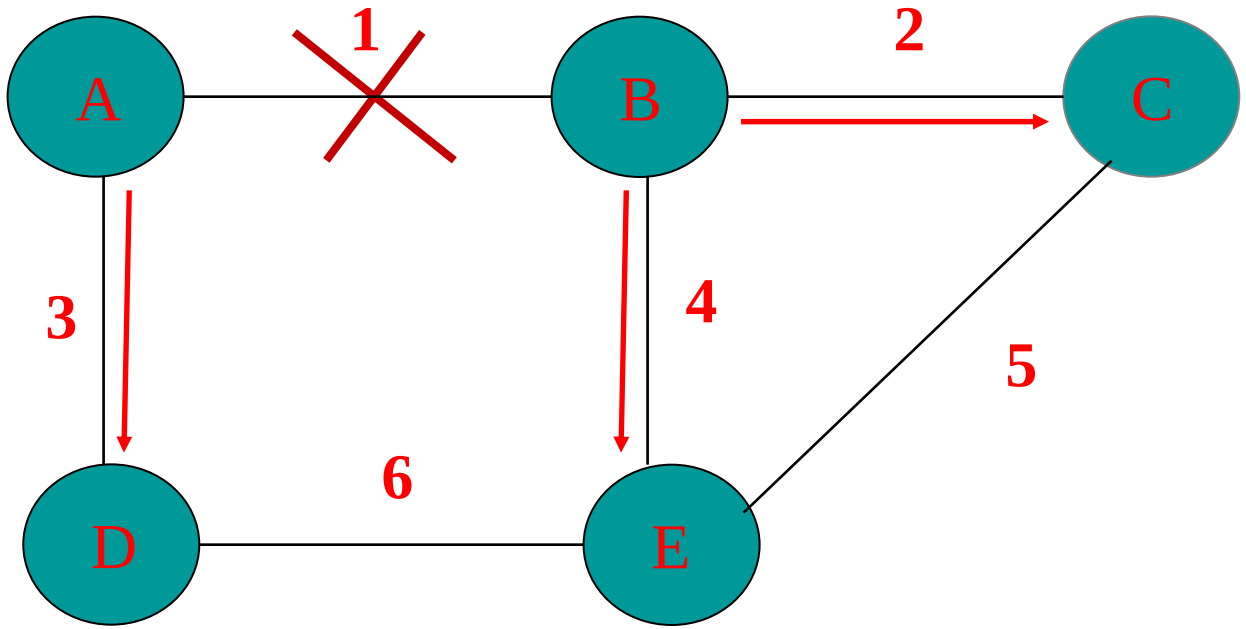
- Se il LSP non è mai stato ricevuto, o il SN è superiore a quello memorizzato precedentemente:
  - memorizza il LSP
  - lo ritrasmette in flooding sulle uscite
- Se il LSP ha lo stesso SN di quello memorizzato
  - non fa nulla
- Se il LSP è più vecchio di quello memorizzato
  - trasmette quello più recente al mittente

Calcolo tabelle di routing

- Ogni volta che l'archivio dei LSP varia si verifica se varia il grafo pesato
- in questo caso si ri-esegue l'algoritmo di calcolo dell'albero dei cammini minimi
- si inserisce nella tabella di routing per ogni possibile destinazione il primo hop [destinazione, nodo_vicino]

Esempio: Link State

- il protocollo di routing deve aggiornare il database quando la rete cambia



- la rottura del link 1 viene riscontrata dai nodi A e B che aggiornano il proprio database e mandano un messaggio di update sui link 3, 2 e 4

  nodo A: Da A, Verso B, link 1, Costo=inf, S.Number=2

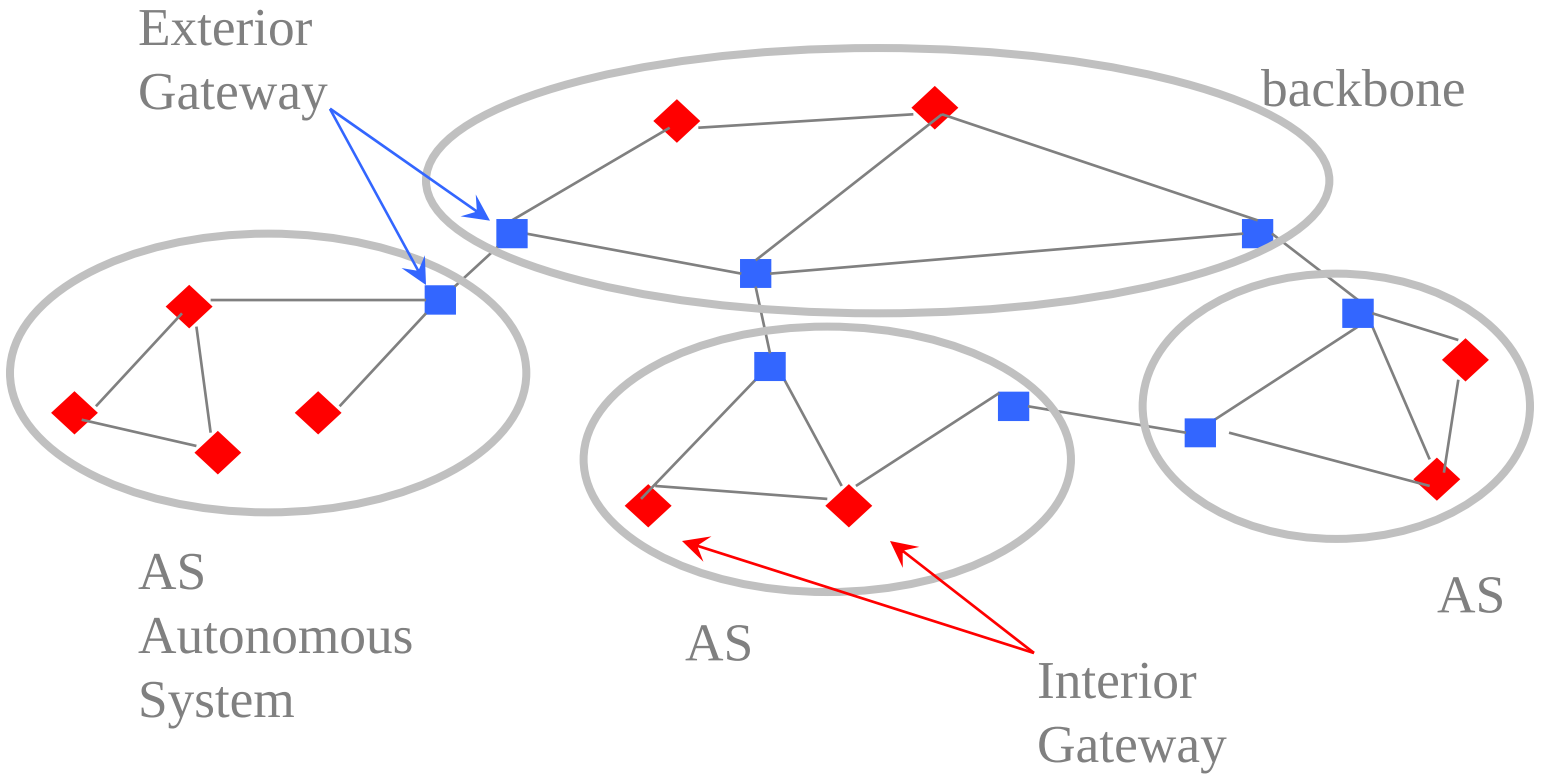
  nodo B: Da B, Verso A, link 1, Costo= inf, S.Number=2

Esempio: Link State

- i messaggi sono ricevuti dai nodi D,E ed C che aggiornano il proprio database e li trasmettono sui link locali, i nuovi messaggi non modificano i database perché hanno lo stesso SN del record nel database
- il nuovo database dopo il flooding

Da	Verso	Link	Costo	Sequence Number
A	B	1	1⇒inf	1⇒2
A	D	3	1	1
B	A	1	1⇒inf	1⇒2
B	C	2	1	1
B	E	4	1	1
C	B	2	1	1
C	E	5	1	1
D	A	3	1	1
D	E	6	1	1
E	B	4	1	1
E	C	5	1	1
E	D	6	1	1

Routing in Internet

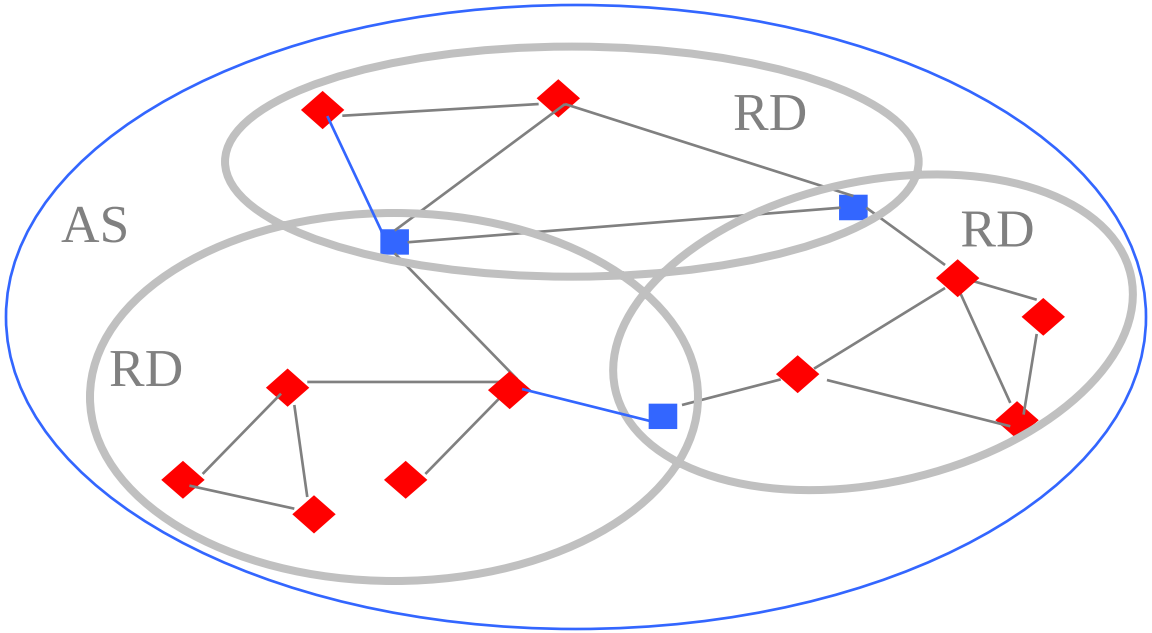


- **Autonomous System: porzione di rete gestita da una stessa autorità**
- **Protocollo di Routing: protocollo per lo scambio di informazioni di routing tra gateway**

- **EGP - Exterior Gateway Protocol**
- **IGP - Interior Gateway Protocol**

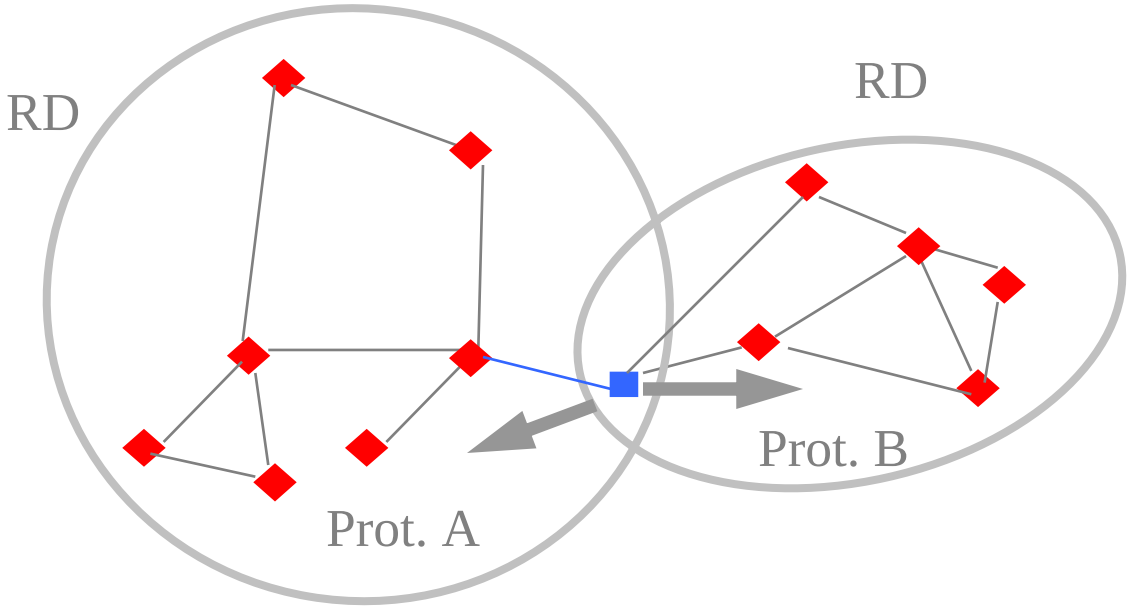
J. Elias: Architetture e Protocolli per Internet

Domini di Routing



- **Dominio di Routing (RD): porzione di AS in cui è implementato un unico protocollo di routing**
- **se i RD comunicano, alcuni router appartengono a più RD e implementano più protocolli di routing**

Ridistribuzione



- I router su più domini possono “ridistribuire” le informazioni di un dominio nell’altro e viceversa
- La traduzione delle informazioni dal Prot. A al Prot. B dipende dall’implementazione e dalle caratteristiche di A e B
- I due protocolli possono anche essere un IGP e un EGP (per alcuni sono definiti dei criteri di redistribuzione)

Protocolli di Routing usati

- IGP
  - RIP (Routing Information Protocol), versione 1 e 2
  - IGRP (Interior Gateway Routing Protocol) proprietario CISCO
  - IS-IS (Intermediate System to Intermediate System)
  - OSPF (Open Shortest Path First)
- EGP
  - BGP (Border Gateway Protocol)

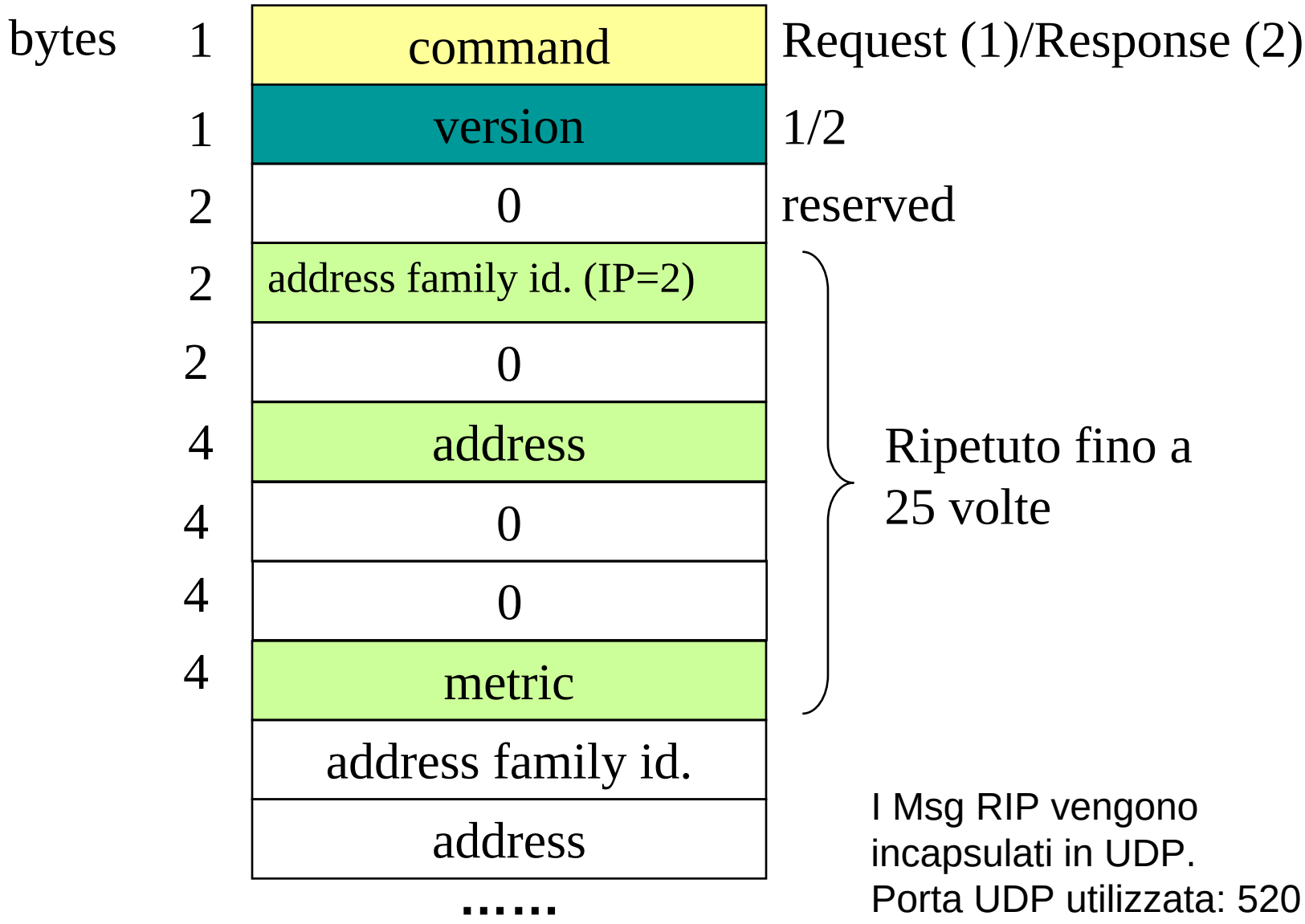
Distance Vector

Link State

RIP v1

- RFC 1058
- Protocollo IGP
- Della famiglia Distance Vector
- Metrica utilizzata: numero di hop (anche se in teoria si possono usare altre metriche)
- Il valore 16 denota infinito
- è tra i più diffusi e semplici protocolli di routing dinamico

RIP v1



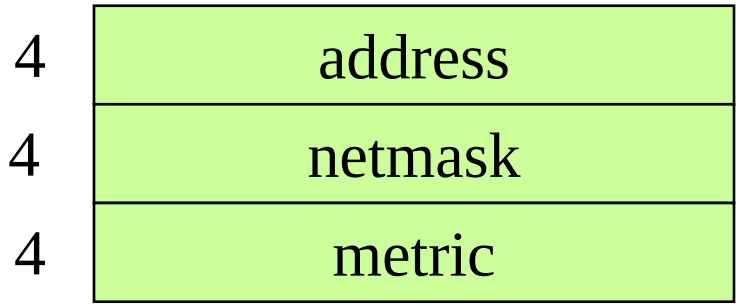
I Msg RIP vengono incapsulati in UDP.
Porta UDP utilizzata: 520

RIP v1: Timer

- route update timer (default 30 s)
  - Ogni router invia DV periodicamente ogni 30 secondi
- route invalid timer (default 180 s)
  - intervallo dopo il quale, se non si ricevono annunci dalla stessa interfaccia, una route è dichiarata non valida. Questo evento viene notificato ai vicini.
- route flush timer (default 240 s)
  - intervallo di tempo dopo cui una route è cancellata (se arrivano nuovi DV da altre interfacce sono accettati)

RIP v2

- **RFC 1723**
- **permette di inserire anche le netmask**
- **Permette una blanda autenticazione**



OSPF

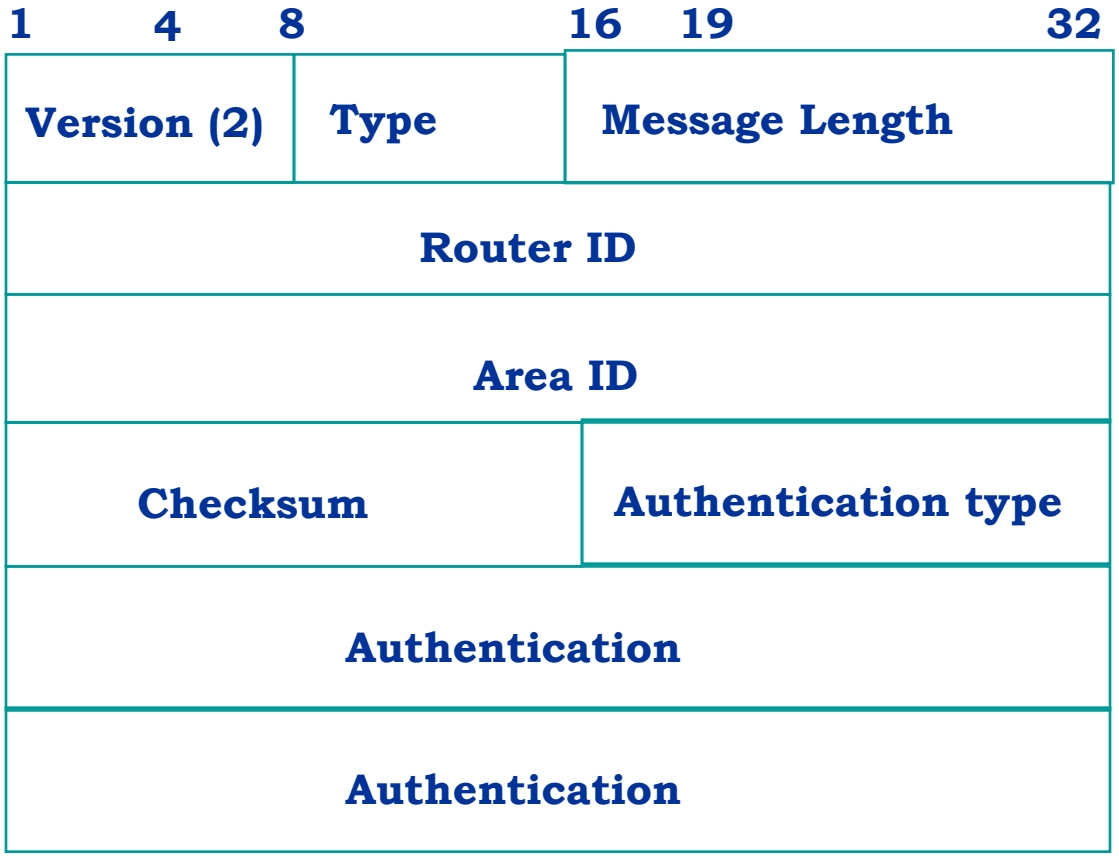
- RFC 1247, 1583
- Link state routing protocol
- Supporta routing gerarchico
- I messaggi di OSPF viaggiano incapsulati direttamente in pacchetti IP (a differenza di RIP, che utilizza UDP). Il campo Protocol nel pacchetto IP viene settato a 89 (OSPF, appunto)
- OSPF è costituito da 3 “sotto-protocolli”:
  - Hello
  - Exchange
  - Flooding

OSPF

- **Protocollo Hello: svolge principalmente 2 funzioni**
  - **Verificare che i link siano attivi (i pacchetti sono inviati ogni *hello-interval* secondi ai router vicini)**
  - **Eleggere Designated Router e Backup Router nelle reti Broadcast**
- **Protocollo Exchange**
  - **Viene utilizzato principalmente quando un router si connette ad un altro (ad es: Designated e/o Backup Router) ed inizializza il proprio Link State Database**
- **Protocollo Flooding**
  - **Le variazioni incrementali e successive a tale inizializzazione vengono gestite tramite il protocollo di Flooding**

OSFP: Open Shortest Path First

Common Header



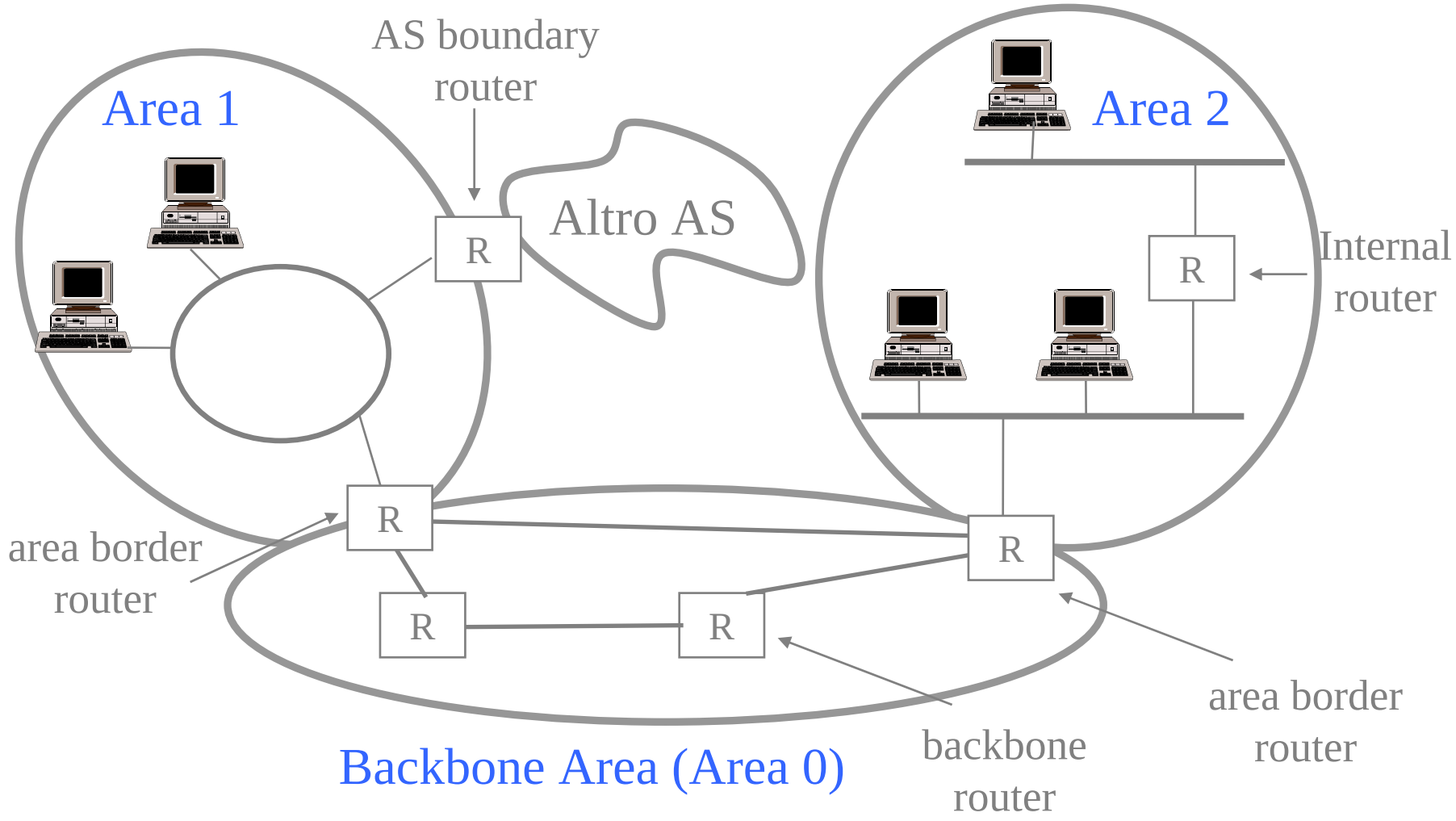
OSFP: Open Shortest Path First

- Il campo *type* specifica il tipo di messaggio OSPF che può essere
  1) HELLO
  2) DATABASE DESCRIPTION
  3) LINK STATUS REQUEST
  4) LINK STATUS UPDATE
  5) LINK STATUS ACKNOWLEDGEMENT
- Il campo Router ID contiene l'indirizzo IP scelto per identificare il router mittente.
- Area ID codifica l'area di appartenenza. Il Valore 0 è riservato per la *backbone area.* In genere, per le altre aree è pratica comune utilizzare un IP network number per identificare un'area.
- Il campo Authentication type può essere 0 (No authentication) o 1 (Simple authentication)

OSFP: Open Shortest Path First

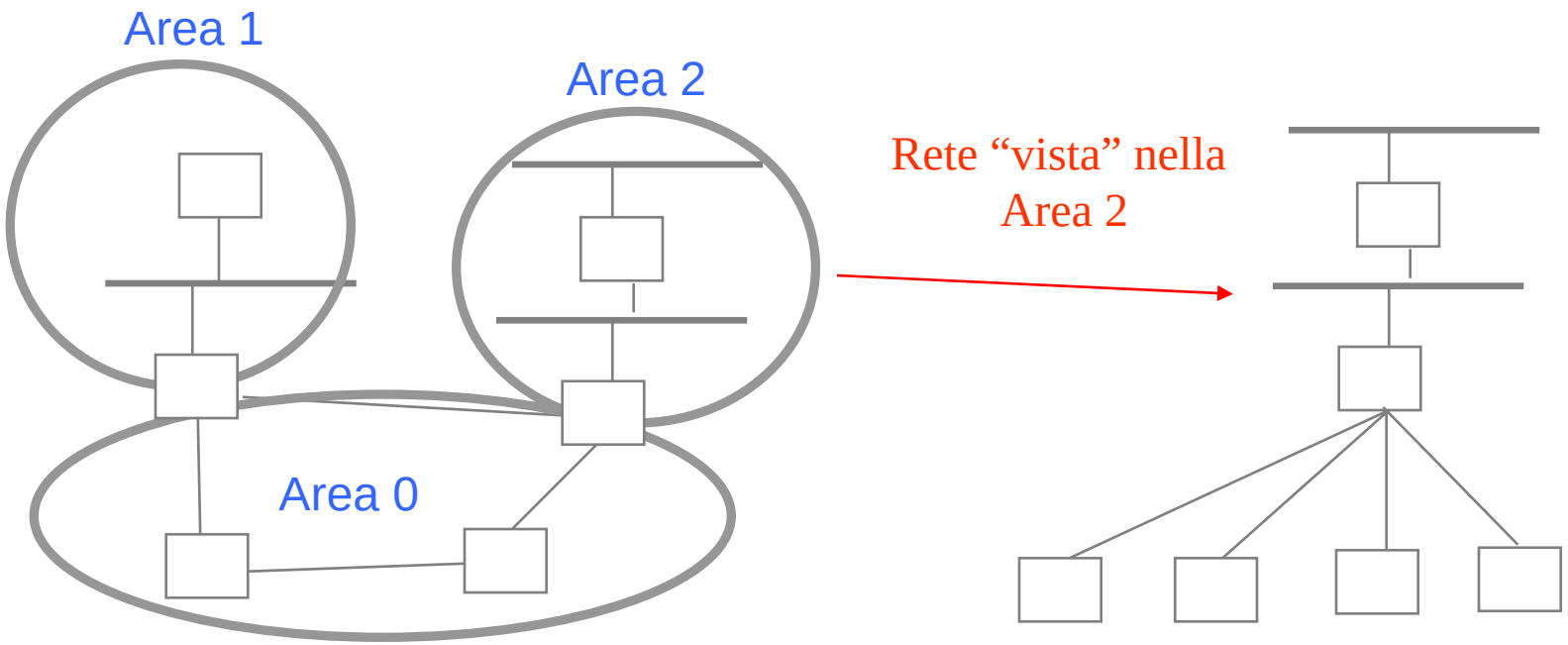
- OSPF invia periodicamente messaggi di HELLO per verificare la raggiungibilità dei vicini
- I messaggi di tipo database description servono ad inizializzare il database topologico dei gateway
- I dati sulle metriche dei link vengono passati tramite i messaggi di link status

OSPF: gerarchia e classificazione dei router



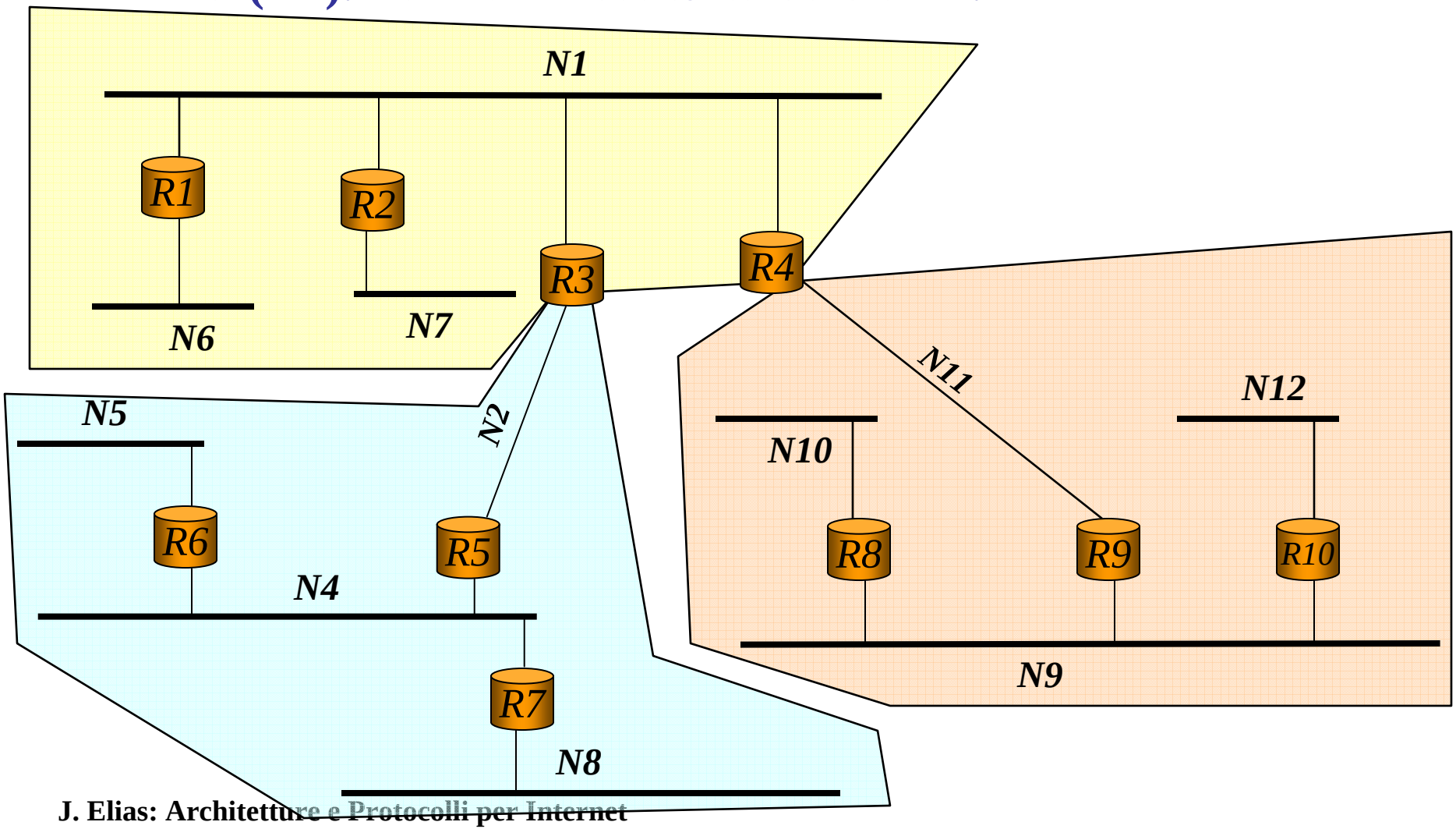
OSPF

- Gli area border router diffondono in ciascuna area un riassunto delle informazioni raccolte nell'altra
  - Si parla di "contaminazione" distance vector



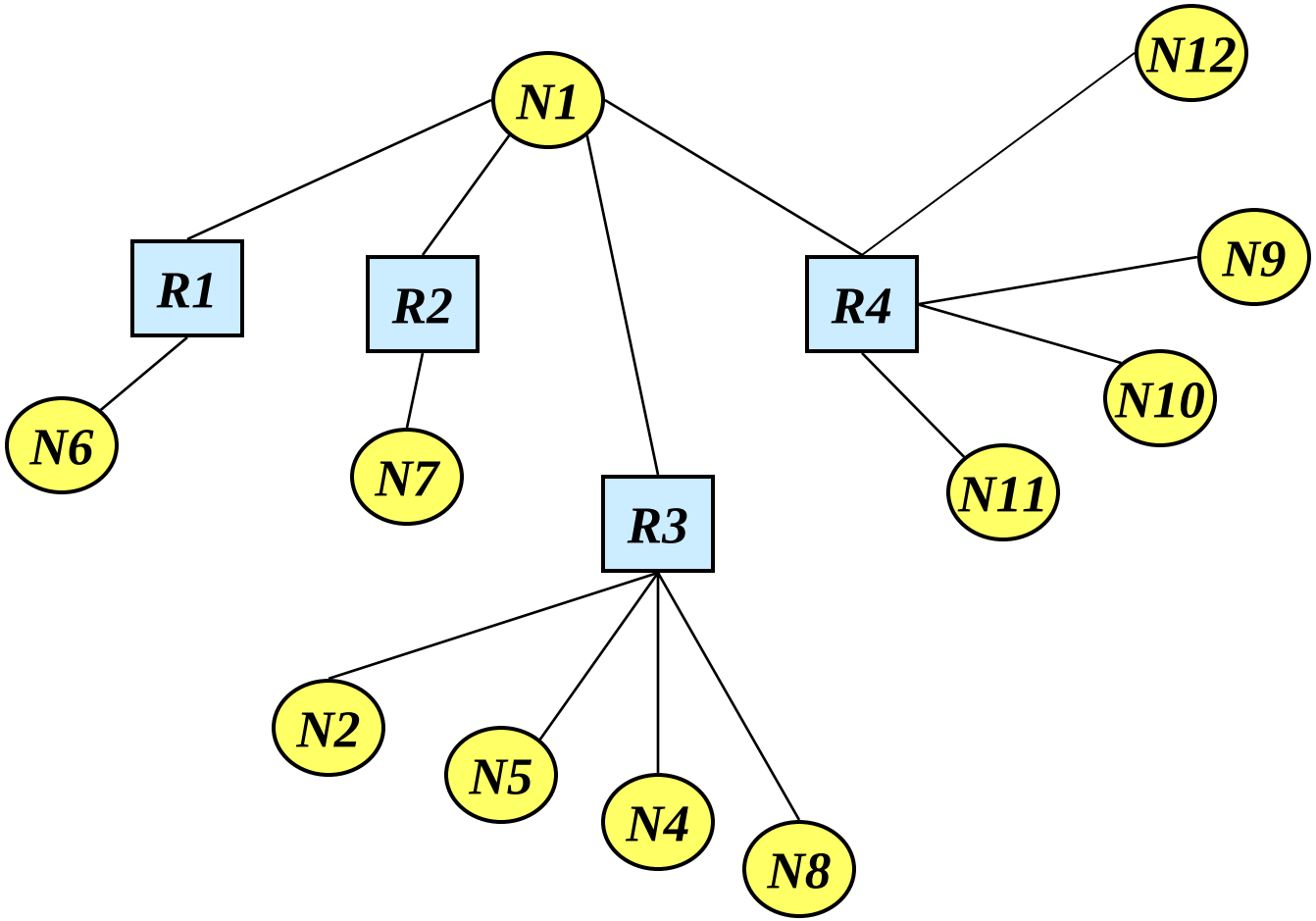
Esempio

- Si consideri la rete in figura dove sono indicati router (Rx) e reti (Nx). La rete è costituita da 3 aree.



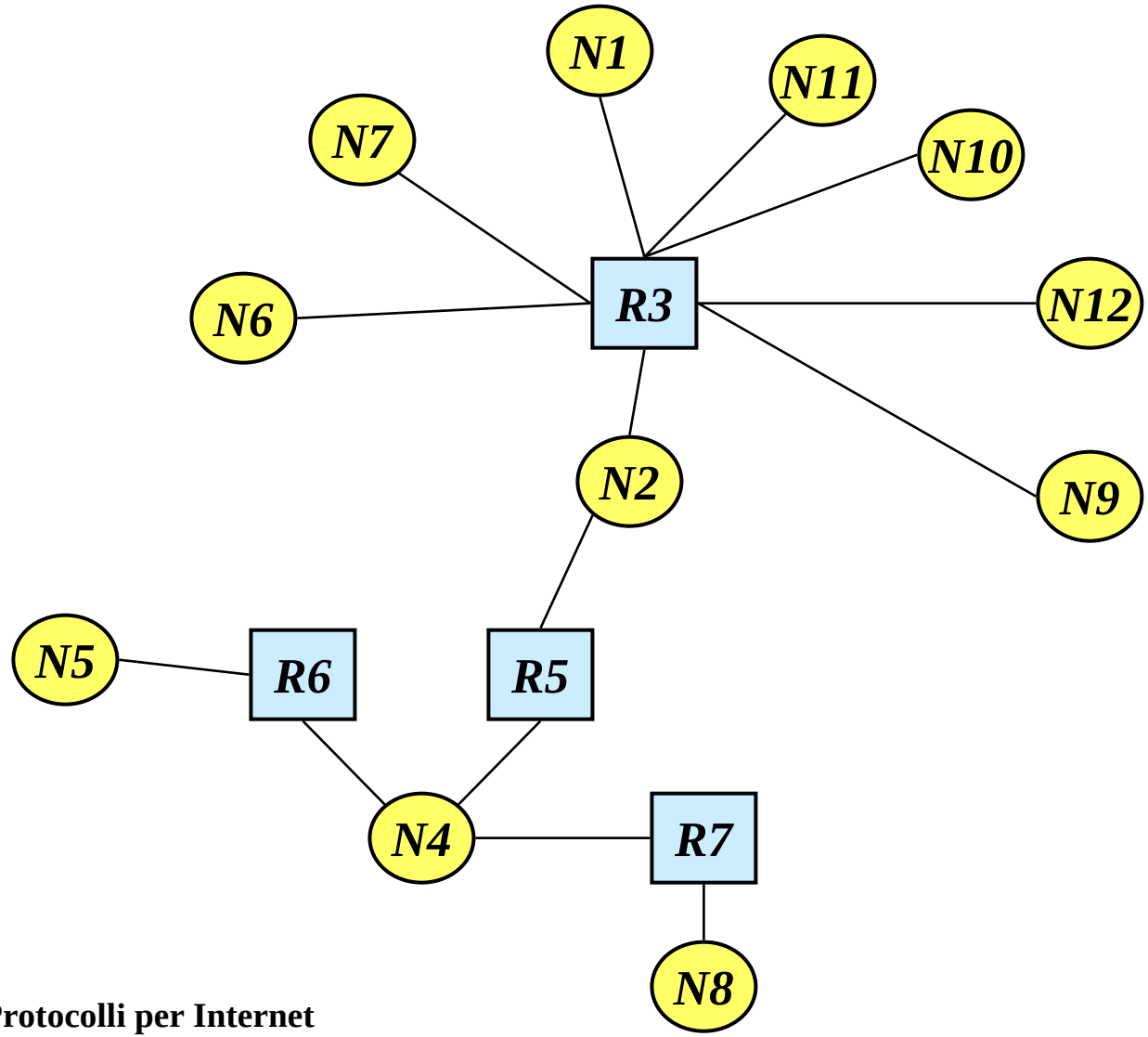
Esempio

La rete “vista” dal router R1 è la seguente



Esempio

La rete “vista” dal router R7 è la seguente



BGP

- E’ il più diffuso protocollo EGP
- Il problema del routing tra AS è diverso da quello di routing interno
  - I criteri di scelta del percorso sono difficilmente traducibili in metriche per il calcolo dei cammini
  - I gestori di un AS hanno bisogno di scegliere il percorso in base ad una propria politica
  - La scelta può essere fatta sulla base della conoscenza dell’intero percorso verso la destinazione
- Quindi:
  - DV non va bene perché non consente la conoscenza dell’intero percorso
  - LS non va bene perché occorrerebbe costruire informazioni topologiche sull’intera rete mondiale

BGP: Path vector

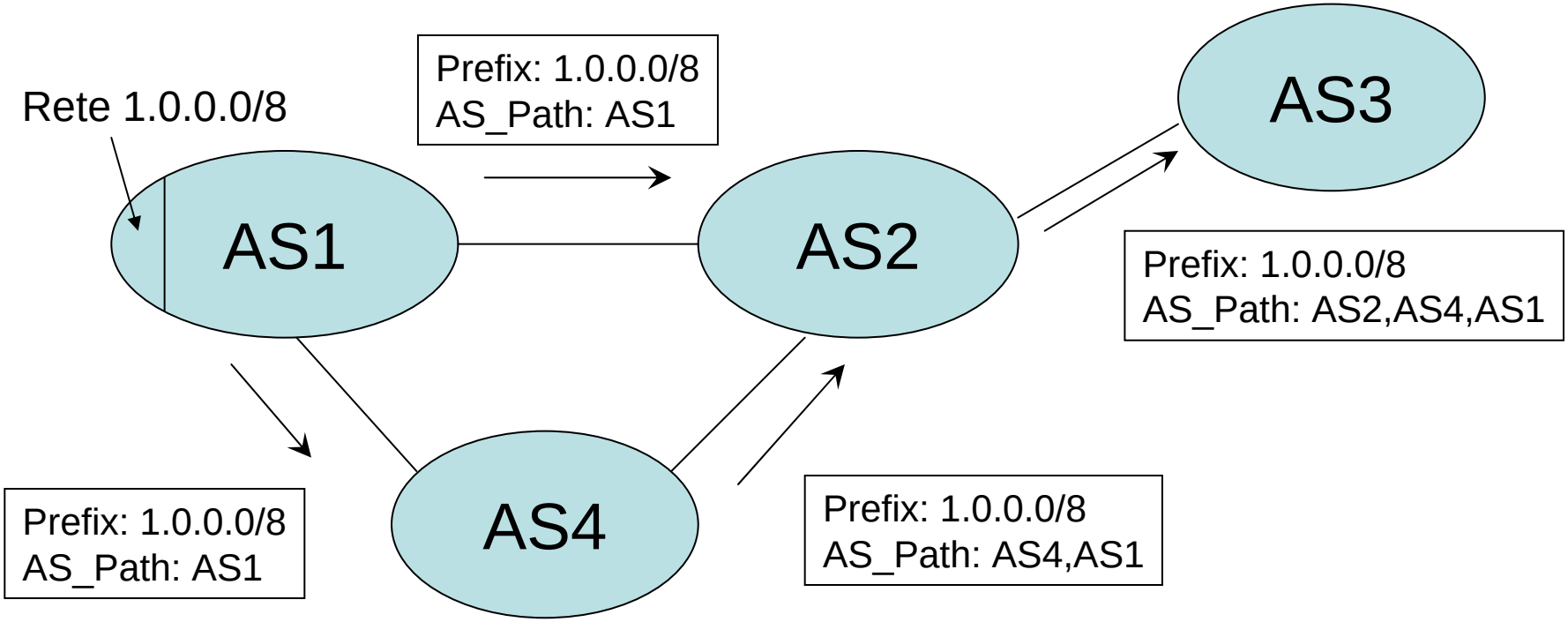
- Il BGP è un protocollo simile al distance vector, ma nei DV inviati dai nodi non è indicata una "distanza dalla destinazione", ma l'intero percorso verso la destinazione

Rete	Router successivo	Percorso
N01	R01	AS2,AS5,AS7,AS12
N02	R07	AS4,AS13,AS6,AS9
N03	R09	AS11,AS12,AS8,AS6
…	…	…

BGP: Path vector

- In realtà un messaggio di path vector che si scambiano due EG vicini non contiene un percorso ma una sequenza di "attributi"
- Si distinguono attributi obbligatori, che devono essere interpretati da tutte le implementazioni di BGP, e facoltativi
- Tra gli attributi obbligatori:
  - ORIGIN: protocollo IGP da cui proviene l'informazione (ad es. OSPF, RIP, IGRP)
  - AS_PATH: sequenza di AS attraversati
  - NEXT_HOP: prossimo router

BGP: Esempio di Funzionamento



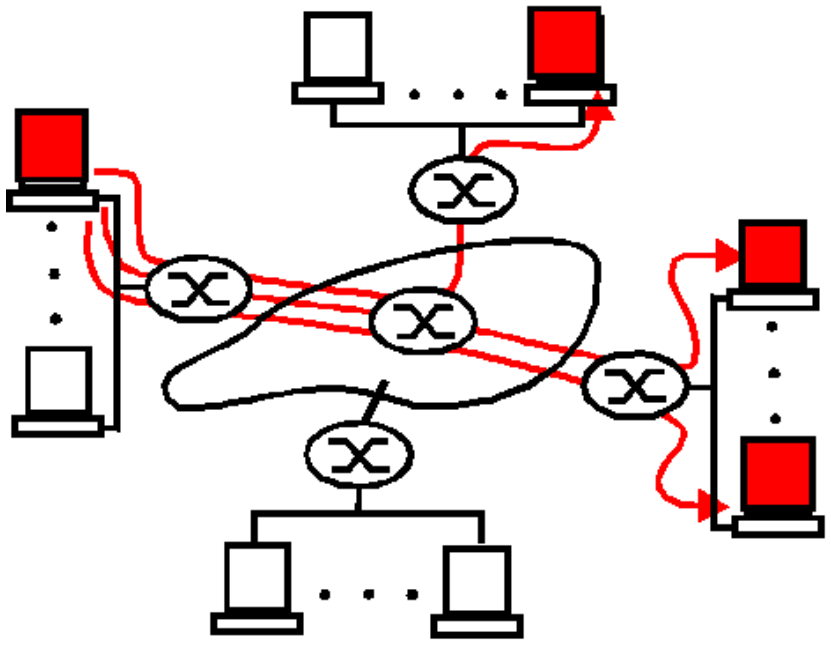
- **AS2 può scegliere se raggiungere la rete 1.0.0.0/8 attraverso AS4 o AS1. Nell'esempio, sceglie AS4 per meri motivi economici**
- **Quindi AS2 inoltra ad AS3 un path vector che riporta il percorso da lui scelto per raggiungere tale rete: AS2-AS4-AS1**

Multicasting

Multicasting

- Nuove applicazioni possono richiedere l'uso di collegamenti di tipo punto-multipunto
  - audio e video streaming in broadcast
  - giochi in rete (Quake), etc.

- Il multicasting può essere implementato direttamente dalla sorgente anche su una rete che supporta solo unicast

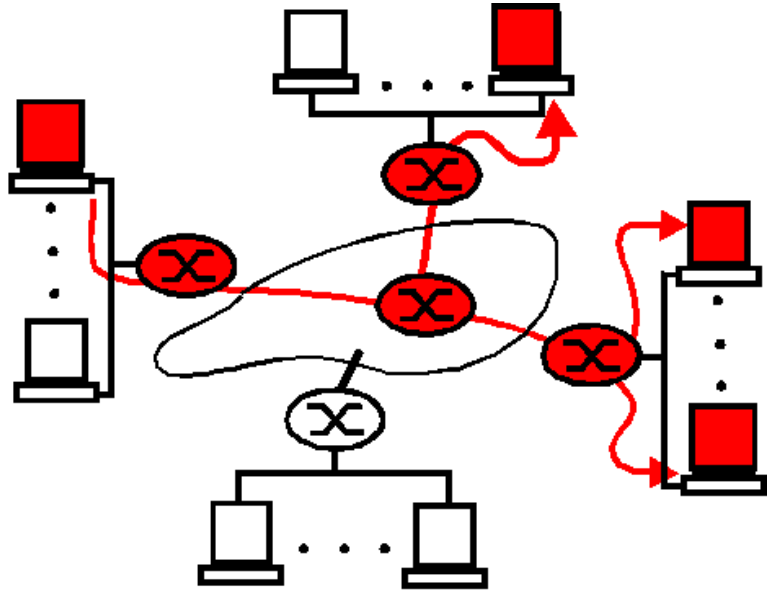


multicast via unicast

Multicasting

- Nel caso in cui la rete supporti il multicast è possibile inviare un solo pacchetto.
- Ad es: Ethernet supporta il Multicast.
- In Internet, poiché un gruppo Multicast può estendersi su molte reti fisiche, è necessario che alcuni nodi della rete svolgano un ruolo attivo (router in rosso nella figura)

- Funzionalità richieste
  - definizione del gruppo di destinatari
  - indirizzamento
  - definizione dell'albero di routing



network multicast

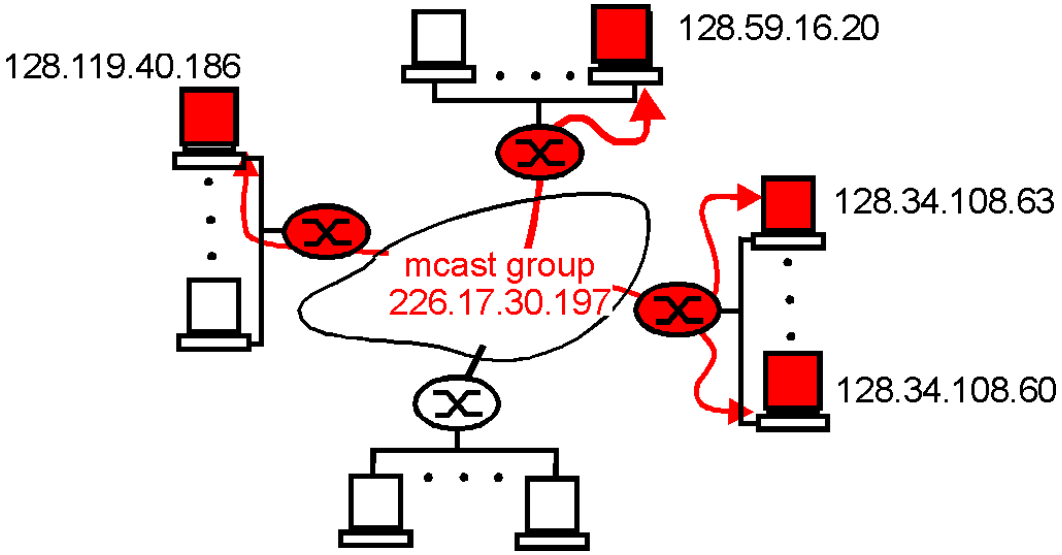
Gruppi ed indirizzi

- **Per identificare un gruppo di ricevitori non è efficiente usare l'insieme dei loro indirizzi IP**
- **IP definisce una classe di indirizzi per applicazioni multicast (utilizzabili solo come indirizzi destinazione, non sender). Alcuni gruppi sono riservati.**

1110	Group Identifier (28 bit)

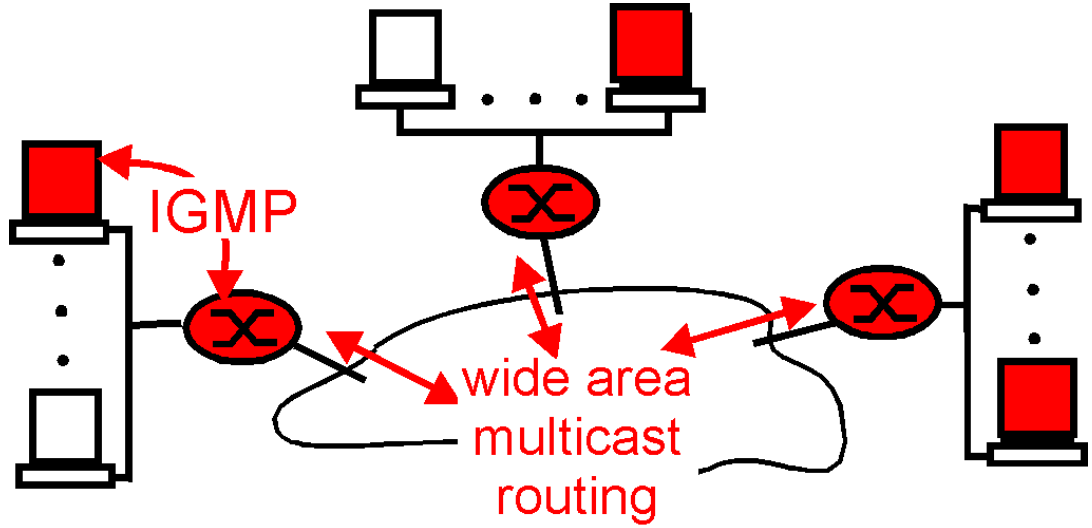
da 224.0.0.0 a 239.255.255.255

- **L'uso degli identificativi di gruppo riduce l'overhead, ma pone problemi:**
  - **come si istituisce un gruppo**
  - **come si aggiungono membri**
  - **come si controlla l'ingresso nel gruppo**
  - **chi conosce l'elenco dei membri**



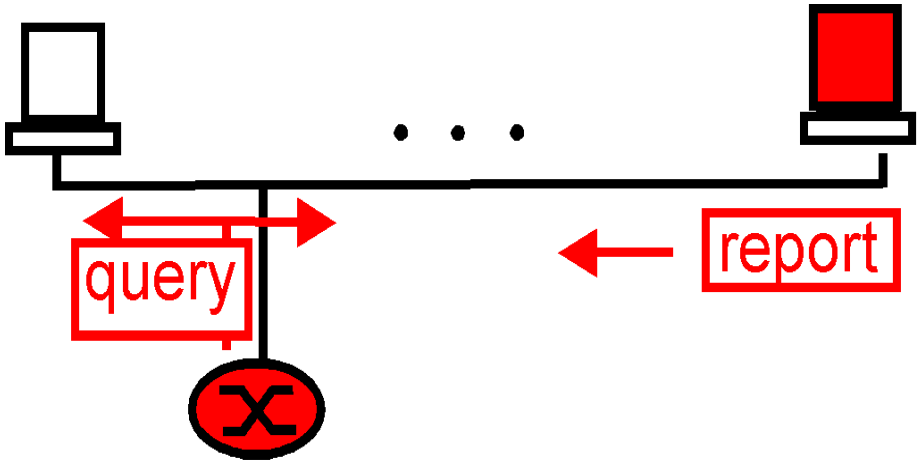
Internet Group Management Protocol (IGMP)

- Specifici router nella rete si preoccupano di gestire il servizio di multicast
- Il protocollo IGMP è usato per il colloquio tra gli host e i multicast router
- ogni host colloquia con il multicast router sulla propria sottorete per la gestione dell'iscrizione ai gruppi



Gestione dei gruppi

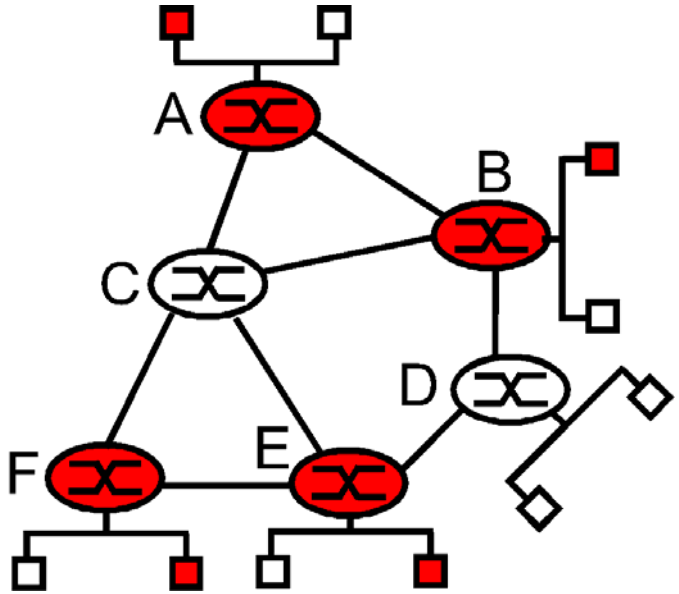
- Periodicamente il router IGMP manda dei messaggi broadcast (224.0.0.1, “all systems on this subnet, hosts and routers”, per es. una LAN)
- Gli host rispondono con l’elenco dei gruppo in uso da qualche processo applicativo



IGMP Message types	Sent by	Purpose
membership query: general	router	query multicast groups joined by attached hosts
membership query: specific	router	group joined by attached hosts
membership report	host	is joined to given multicast group
leave group	host	report leaving given multicast group

Multicast routing

- Come inoltrare i pacchetti di un gruppo?
- Scopo dei protocolli di multicast routing è quello di costruire un albero (Steiner tree) degli instradamenti che consenta di inoltrare i pacchetti senza effettuare dei cicli
- I router che non hanno utenti del gruppo associati possono essere esclusi dall'albero
- Il problema è simile a quello che si incontra nei transparent bridge

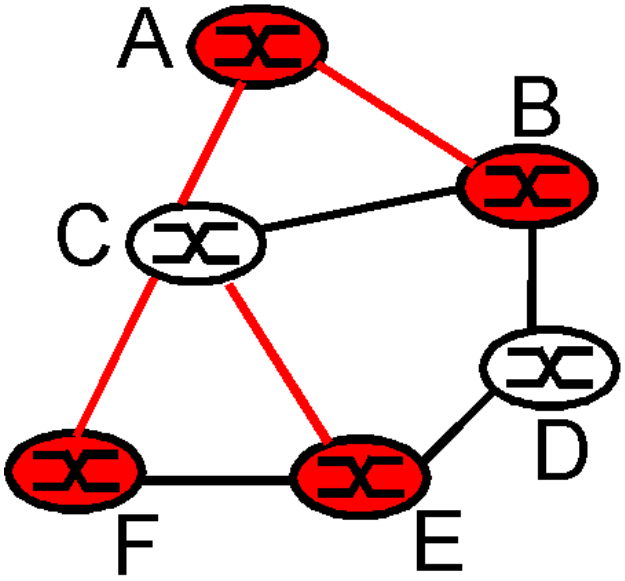


- Steiner Tree problem: trovare l'albero che copre tutte le destinazioni avente il minimo costo possibile. (NP-Complete)
- Constrained Steiner Tree problem: trovare l'albero di costo minimo con vincoli sul massimo ritardo

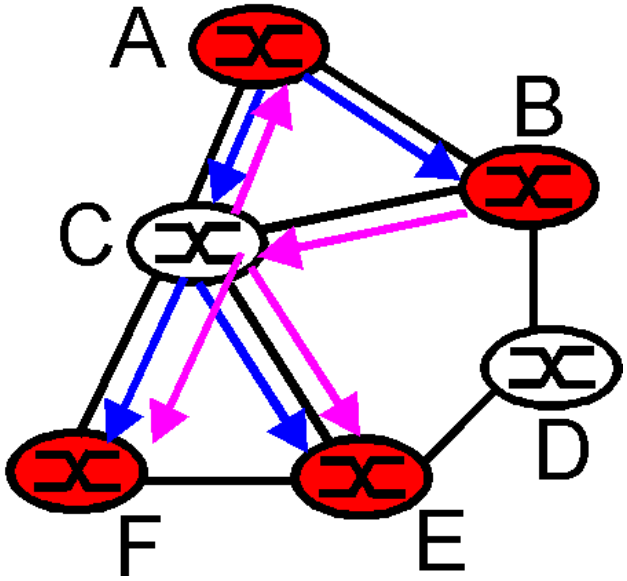
Quali alberi

- E’ possibile costruire un albero che serve a distribuire i pacchetti di tutte le sorgenti
- oppure un albero differente per ognuna delle sorgenti attive

Group-shared Tree

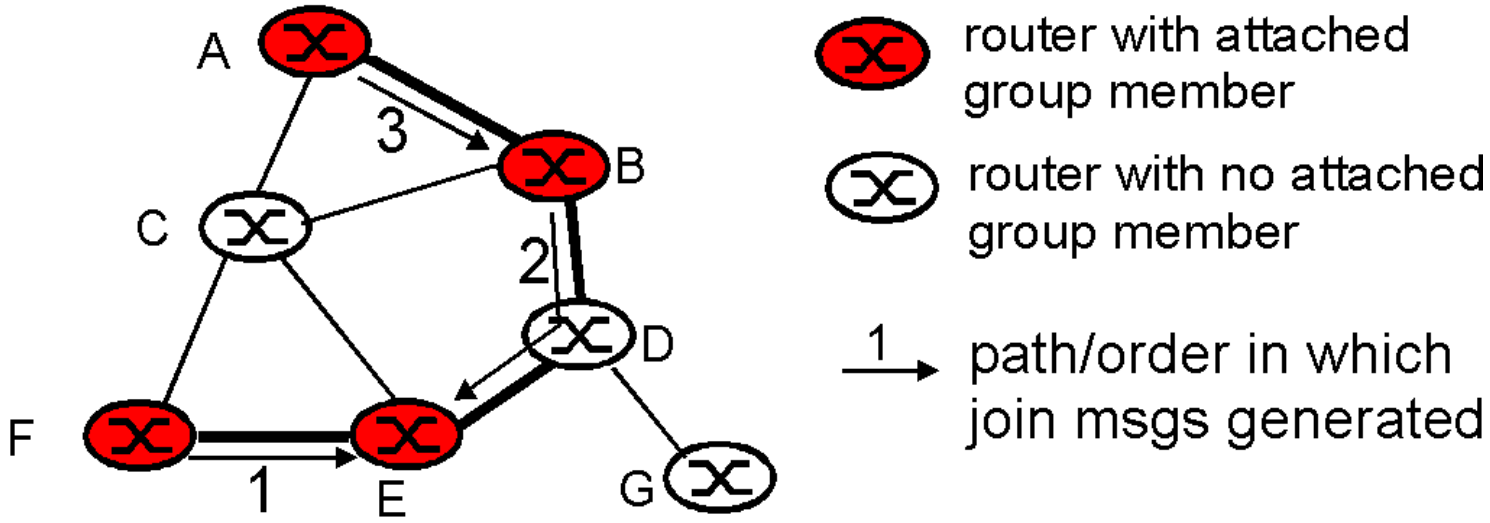


Source-Specific Tree



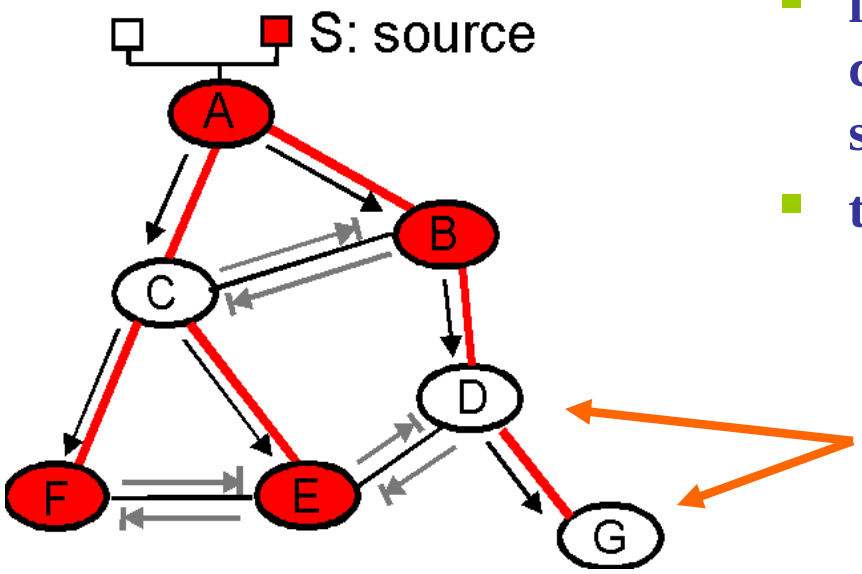
Group-Shared Tree

- In linea teorica è possibile cercare l'albero di costo minimo (Problema NP-completo)
- In realtà gli algoritmi (euristiche) cercano soluzioni sub-ottime
- *center-based approach:*
  - viene eletto un router "centrale" (nell'esempio sotto, il router E)
  - vengono inviati dei messaggi di "join" in unicast verso il router centrale
  - i messaggi tracciano dei rami dell'albero fino a che non raggiungono il centro o un altro router già associato all'albero



Source-based Trees

- La costruzione dei source-based trees è di solito basata sull'albero dei cammini (unicast) minimi, costruibile dai router in modo analogo a quanto fatto con il routing unicast
- Reverse Path Forwarding (RPF): un router accetta un pacchetto multicast solo sull'interfaccia sulla quale il router invierebbe un messaggio unicast verso la sorgente del messaggio multicast ricevuto



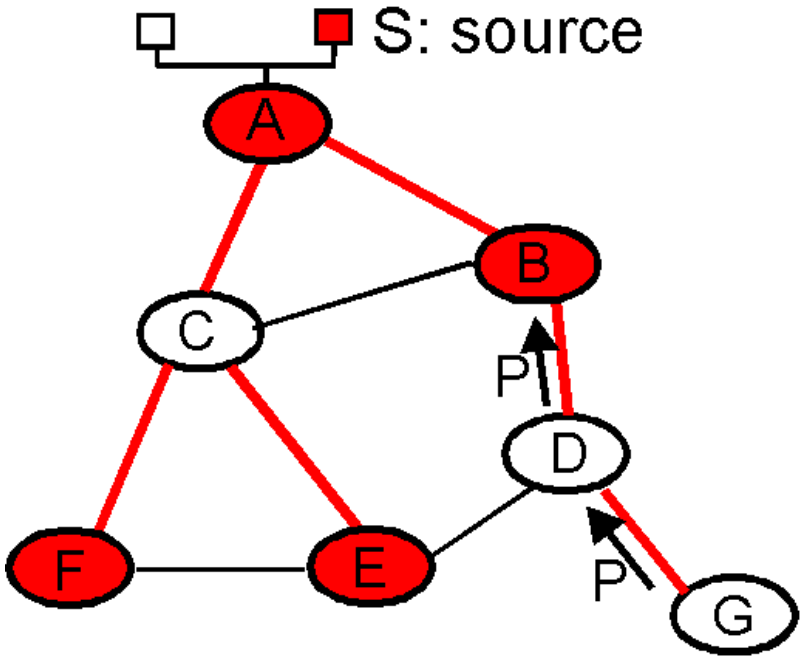
- i pacchetti in arrivo sul cammino minimo verso la sorgente sono inoltrati
- tutti gli altri sono scartati

i pacchetti giungono anche a nodi senza host associati

Source-based Trees: pruning

- Il problema dei pacchetti che arrivano a nodi non coinvolti nel gruppo può essere risolto con la tecnica detta di *pruning*
- I router che non hanno host associati al gruppo possono inviare dei messaggi di *prune* risalendo al contrario lungo l'albero

- Problemi:
  - avere informazioni sui router posti a valle (segnalazione opportuna)
  - consentire l'ingresso di nuovi utenti (messaggi di *unprune* o timer sull'informazione di prune)

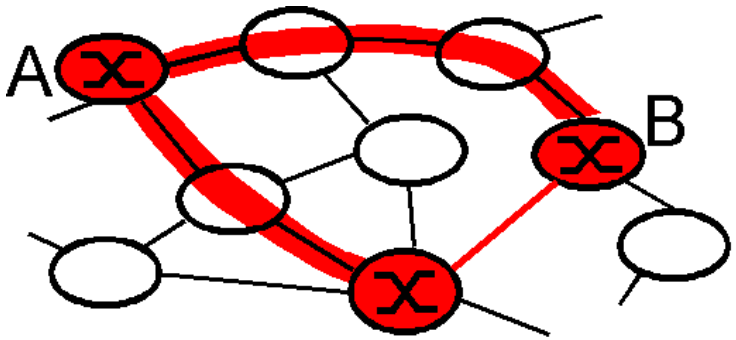


Distance Vector Multicast Routing Protocol (DVMRP, RFC 1075)

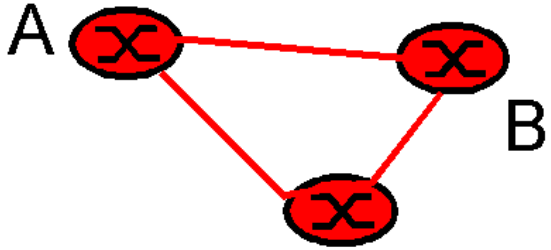
- è il più usato protocollo RPF
- usa il distance vector per costruire l'albero dei cammini minimi
- ogni router mantiene una lista di router dipendenti
- un messaggio di *pruning* viene inviato solo se tutti i router in tale lista lo hanno già fatto
- è previsto un esplicito messaggio di *unprune*
- le informazioni di *pruning* hanno un time-out

Multicast in Internet

- solo una piccolissima frazione dei ruoter di Internet sono multicast routers
- cosa succede se nessuno dei router vicini ad un multicast router ha capacità di multicast?
- La tecnica in uso in MBone (Multicast Backbone) è il tunneling:



physical topology



logical mcast topology