

Algoritmo di Routing Link State

Fulvio Riso

fulvio.riso[at]polito.it

Mario Baldi

<http://staff.polito.it/mario.baldi>



Nota di Copyright

Questo insieme di trasparenze (detto nel seguito slide) è protetto dalle leggi sul copyright e dalle disposizioni dei trattati internazionali. Il titolo ed i copyright relativi alle slide (ivi inclusi, ma non limitatamente, ogni immagine, fotografia, animazione, video, audio, musica e testo) sono di proprietà degli autori indicati a pag. 1.

Le slide possono essere riprodotte ed utilizzate liberamente dagli istituti di ricerca, scolastici ed universitari afferenti al Ministero dell'Istruzione, dell'Università e della Ricerca, per scopi istituzionali, non a fine di lucro. In tal caso non è richiesta alcuna autorizzazione.

Ogni altra utilizzazione o riproduzione (ivi incluse, ma non limitatamente, le riproduzioni su supporti magnetici, su reti di calcolatori e stampate) in toto o in parte è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte degli autori.

L'informazione contenuta in queste slide è ritenuta essere accurata alla data dell'edizione. Essa è fornita per scopi meramente didattici e non per essere utilizzata in progetti di impianti, prodotti, reti, ecc. In ogni caso essa è soggetta a cambiamenti senza preavviso. Gli autori non assumono alcuna responsabilità per il contenuto di queste slide (ivi incluse, ma non limitatamente, la correttezza, completezza, applicabilità, aggiornamento dell'informazione).

In ogni caso non può essere dichiarata conformità all'informazione contenuta in queste slide.

In ogni caso questa nota di copyright non deve mai essere rimossa e deve essere riportata anche in utilizzi parziali.




Routing Distribuito

■ Modello Peer

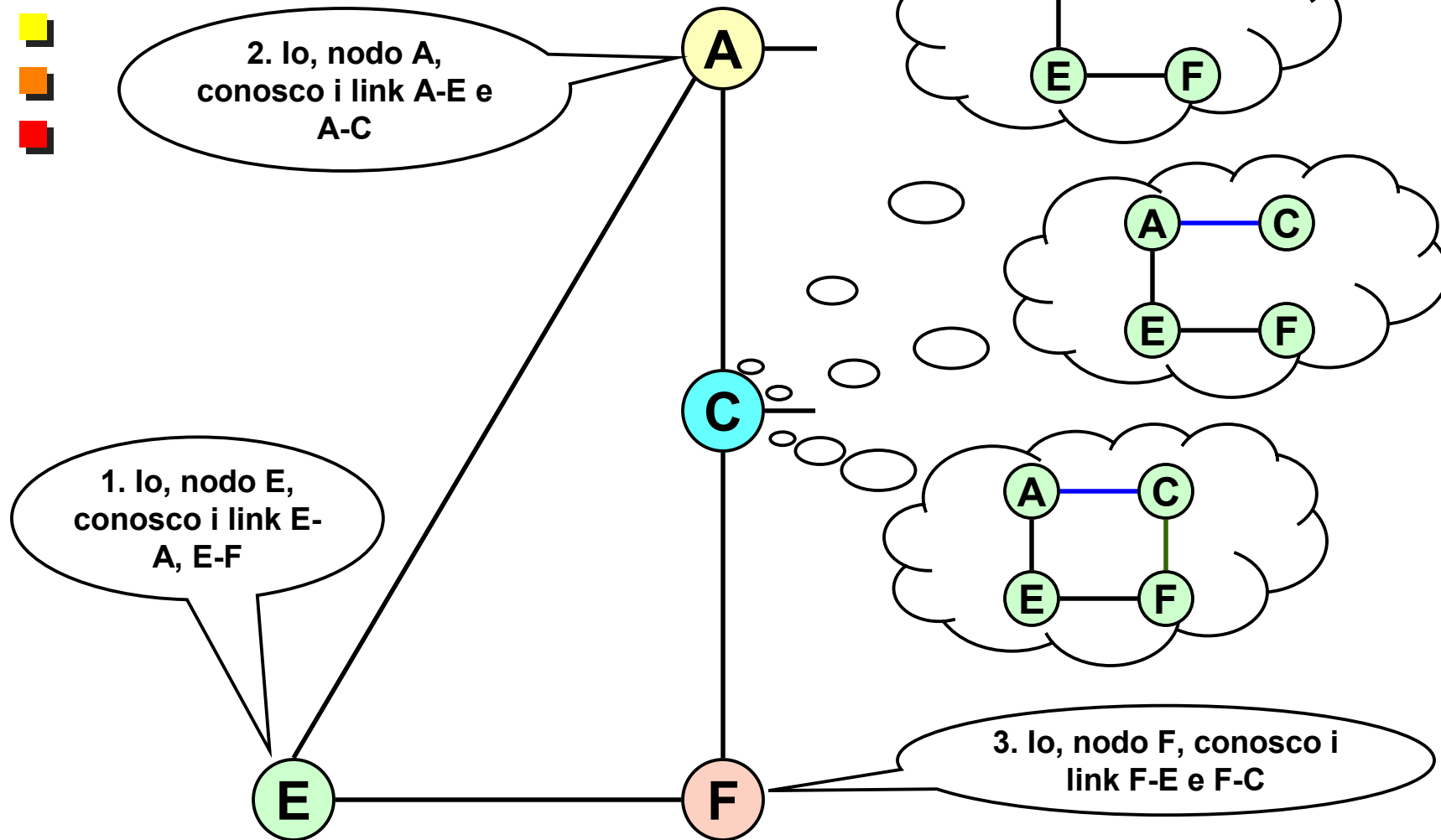
- Unione dei vantaggi di routing Isolato e Centralizzato
- Centralizzato: i router cooperano allo scambio di informazioni di connettività
- Isolato: i router sono paritetici e non esiste un router “migliore”

■ Due algoritmi principali

- Distance Vector
 - Si distribuiscono ai vicini le informazioni su tutta la rete
 - Variante: Path Vector
 - Link State
 - Si distribuiscono a tutti i router le informazioni sui vicini
 - Utilizzati da gran parte dei protocolli di routing moderno
- 



Principio del Link State

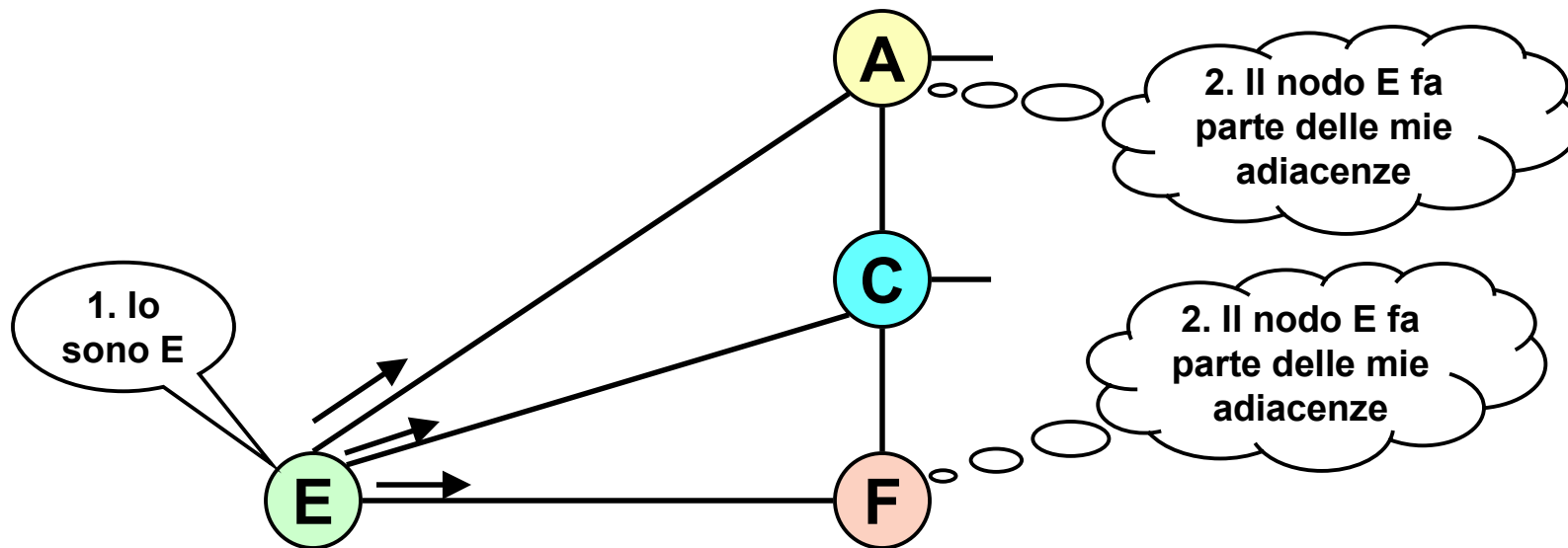


Componenti dell'algoritmo Link State

- Link State: invio delle informazioni sullo stato dei link adiacenti a tutti i nodi della rete
- Ogni nodo dispone di una mappa della rete
 - La costruzione è fatta in maniera cooperativa
- Necessari più componenti per il funzionamento del routing Link State
 - Neighbor Greetings (Hello)
 - Link State (Packet)
 - Flooding
 - Può essere un algoritmo Selective Flooding classico
 - Dijkstra (o Shortest Path First)
 - Bringing up Adjacencies

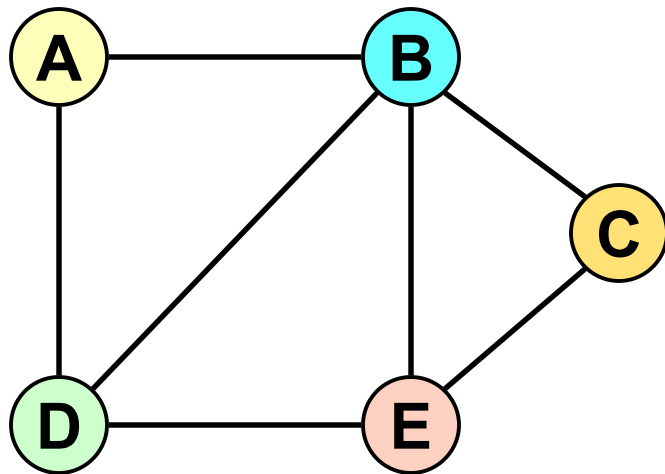
Neighbor Greetings (Hello)

- **Necessario per riconoscere l'esistenza dei nodi adiacenti**
 - **Funzionamento periodico, molto simile all'invio del Distance Vector**
 - **Periodicità elevata per il riconoscimento delle variazioni sulle adiacenze in tempo ragionevoli**
 - **Evita il ricorso ai segnali link-up (non sempre affidabili)**



Link State (Packet)

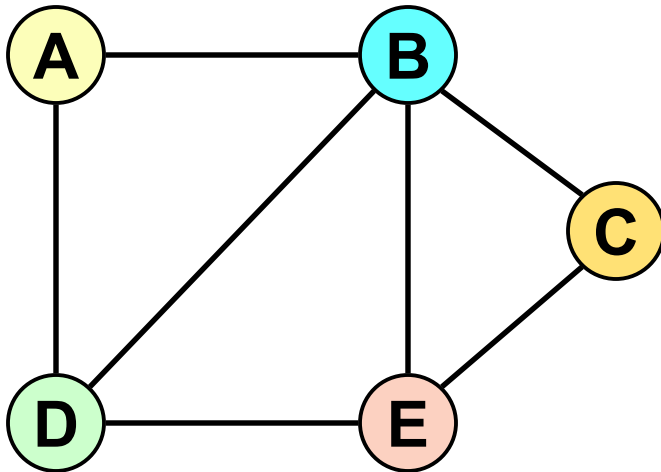
- Insieme delle adiacenze (*adiacenza – costo*)
 - Link: caratterizzato da nodo sorgente – nodo destinazione
 - Nodo sorgente: implicito nel mittente del LS
 - Generato indipendentemente da ogni nodo
 - Ogni nodo inserisce l'elenco delle adiacenze e il loro costo
 - Ogni nodo memorizza i LS di tutti gli altri nodi della rete
 - Propagazione veloce (non c'è elaborazione locale del LS)



Memoria di A

<u>LS (A)</u>	<u>LS (B)</u>	<u>LS (C)</u>	<u>LS (D)</u>	<u>LS (E)</u>
B, 1	A, 1	B, 1	A, 1	B, 1
D, 1	C, 1	E, 1	B, 1	C, 1
	D, 1		E, 1	D, 1
	E, 1			

Link State Database



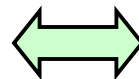
Memoria di A

LS (A)	LS (B)	LS (C)	LS (D)	LS (E)
B, 1	A, 1	B, 1	A, 1	B, 1
D, 1	C, 1	E, 1	B, 1	C, 1
	D, 1		E, 1	D, 1
	E, 1			

Ogni nodo ha a disposizione il grafo della rete:

- i nodi rappresentano i router
- gli archi (con relativo costo) rappresentano i link

Link State Database					
A	B/1	D/1			
B	A/1	C/1	D/1	E/1	
C	B/1	E/1			
D	A/1	B/1	E/1		
E	B/1	C/1	D/1		



(da)

(a)

	A	B	C	D	E
A		1		1	
B	1		1	1	1
C		1			1
D	1	1			1
E		1	1	1	

Database: identico su tutti i nodi della rete



Flooding

■ I Link State Packets

- Devono essere inviati in “broadcast” a tutta la rete
- Devono essere ricevuti invariati da qualunque nodo della rete

■ Protocolli reali: implementano una forma di Selective flooding

- La routing table non può ancora essere utilizzata per raggiungere (in unicast) tutti i nodi della rete



Algoritmo di Dijkstra

■ Algoritmo di Dijkstra

- Usato per il calcolo dello Spanning Tree (albero dei cammini di costo minimo avente il nodo come radice) del grafo

■ Funzionamento

- Si individua il nodo “più vicino” a quello radice
- Se il nodo esiste:
 - Si inserisce questo percorso nella routing table
 - Si ripete da capo, selezionando il nodo a costo immediatamente superiore
- Altrimenti: termina quando è stato trovato un percorso per tutti i nodi

Algoritmo di Dijkstra

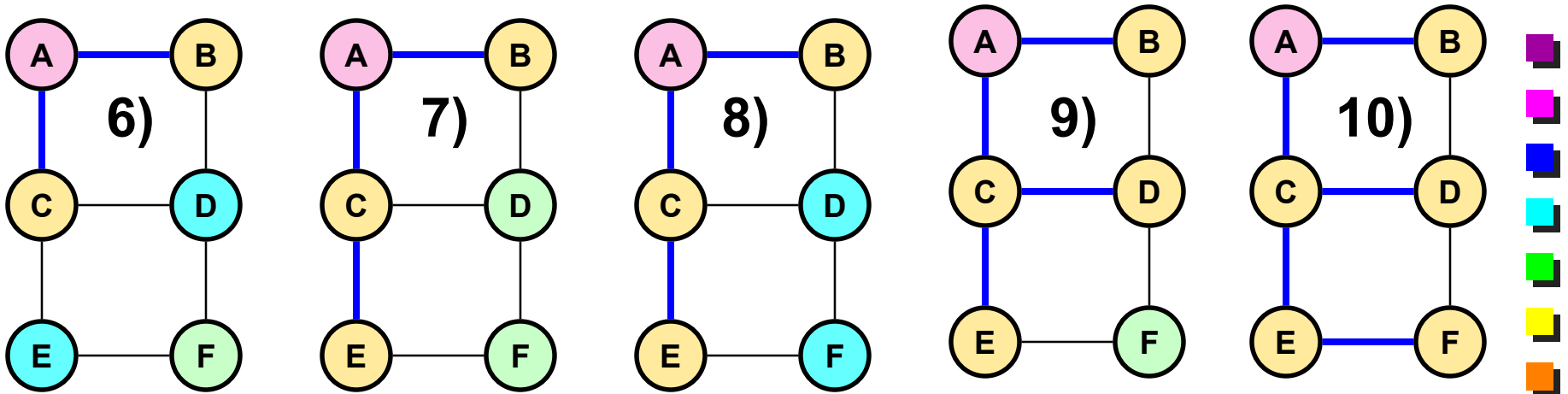
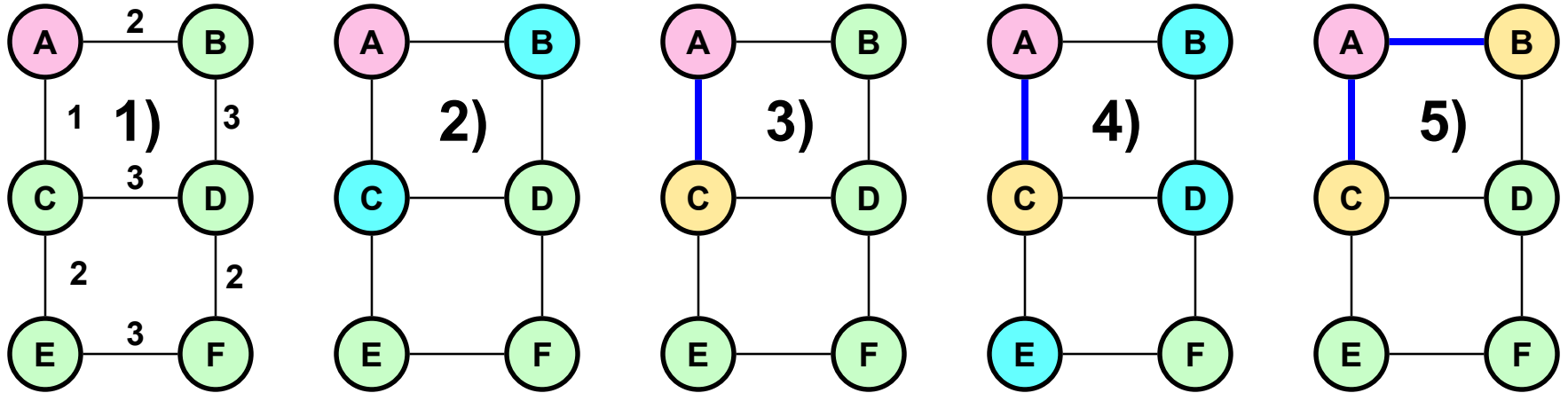
■ Si definiscono

- 1 nodo radice (root), il nodo che sta calcolando l'algoritmo
- 1 insieme PATH di nodi per i quali si è già trovato il percorso migliore
- 1 insieme TEMP di nodi per i quali si sta cercando un percorso

■ Algoritmo

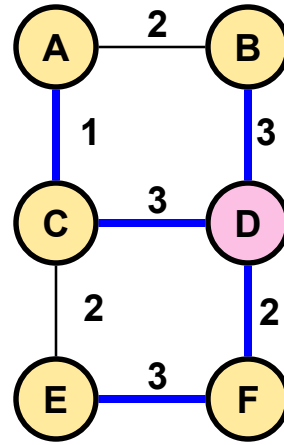
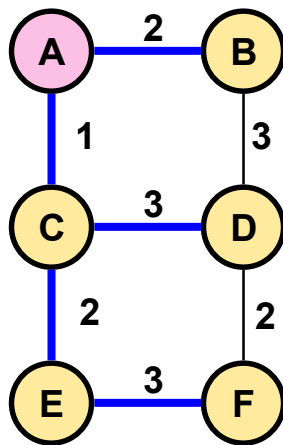
- Si inserisce il nodo root in PATH
- Si inseriscono tutti i nodi vicini del precedente in TEMP
- Si prende il nodo N con il percorso a costo minore in TEMP e lo si promuove in PATH
- Per ogni vicino V del nodo N promosso
 - Se V non esiste ancora in TEMP lo si inserisce ora
 - Se V già esiste se ne analizza il costo verso la root ($D(\text{root}, N) + D(N, V)$) e se questo è minore del precedente riportato in TEMP si aggiorna cost e link di quel nodo in TEMP

Esempio

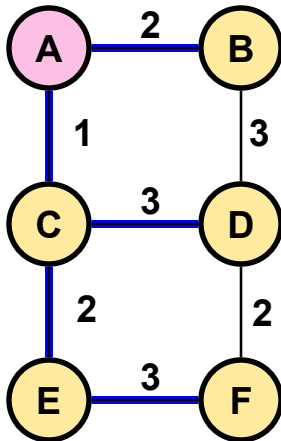


Alberi di instradamento

- Ogni nodo ha lo stesso DB
 - Ogni nodo ha però un diverso albero di instradamento verso le destinazioni
 - Spanning Tree: l'albero di instradamento è invece condiviso tra i nodi
 - Non esistono link inutilizzati
- Gli albero di instradamento sono coerenti tra i vari nodi



Dijkstra su matrice



1. Cancellare la colonna della radice
2. Selezionare il nodo a costo minore verso la radice
3. Inserirlo in PATH
4. Cancellarne la relativa colonna
5. Aggiornare i costi del nodo verso le altre destinazioni in modo da tenere conto del costo tra la radice e il nodo stesso
6. Se ci sono ancora colonne, vai al punto 2.

	A	B	C	D	E	F
A		2	1			
B	2			3		
C	1			3	2	
D		3	3			2
E			2			3
F				2	3	

	B	C	D	E	F
A	2	1			
B			3		
C			3	2	
D	3				2
E		2			3
F			2	3	

C+1

	B	D	E	F
A	2			
B		3		
C		4	3	
D				2
E				3
F		2	3	

B+2

	D	E	F
A			
B	5		
C	4	3	
D			2
E			3
F	2	3	

E+3

	D	F
A		
B	5	
C	4	
D		2
E		6
F	2	

D+4

	F
A	
B	
C	
D	6
E	6
F	



Complessità dell'algoritmo di Dijkstra

■ Complessità algoritmica

- Dijkstra : $L * \log N$
- Bellman-Ford: $N * L$
- L è il numero di link, N è il numero di nodi
 - Normalmente N e L sono dello stesso ordine di grandezza
- Dijkstra è molto più scalabile di Bellman-Ford

■ Complessità algoritmica: è solo una componente di quella dell'algoritmo Link State nella sua interezza



Generazione dei Link State

- **Teoricamente: quando il router rileva una variazione nella topologia locale (adiacenze)**
 - riconosce di avere un nuovo vicino
 - il costo verso un vicino è cambiato
 - ha perso la connettività verso un vicino prima raggiungibile
- **In pratica: generazione periodica**
 - Aumento dell'affidabilità



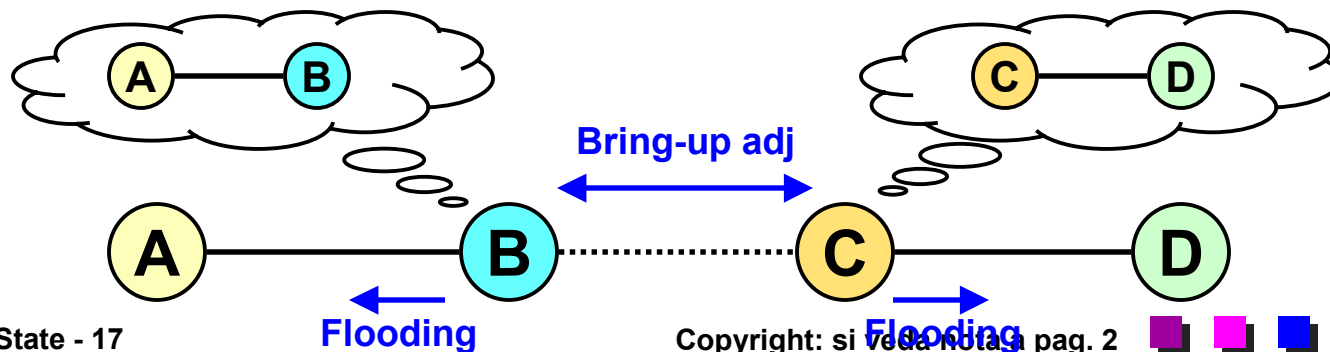
Bringing up adjacencies

■ Utilizzato per:

- Popolare immediatamente il DB di un nodo appena acceso
- Allineare i DB dei nodi in caso di partizionamento della rete
 - Ambedue i router impareranno qualcosa l'uno dall'altro

■ Procedura

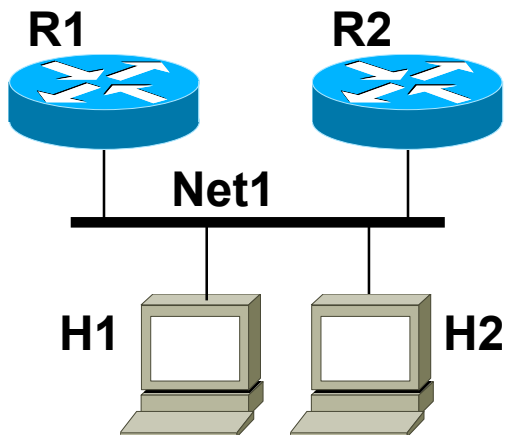
- Attivazione della procedura di HELLO
- Se viene rilevato una nuova adiacenza, inizia una fase di sincronizzazione del database con essa
 - La sincronizzazione viene ripetuta per ogni adiacenza
 - I LS non conosciuti verranno anche inviati in flooding agli altri nodi di rete



Link State e reti reali: stub network

■ Reti periferiche

- Ospitano End System
 - Normalmente non hanno capacità di routing
 - Non generano Link State
- Possono essere sostituite con una entry unica valida per tutta la rete



Link State Database (ideale)

R1	R2/1	H1/1	H2/1
R2	R1/1	H1/1	H2/1
H1	R1/1	R2/1	H2/1
H2	R1/1	R2/1	H1/1

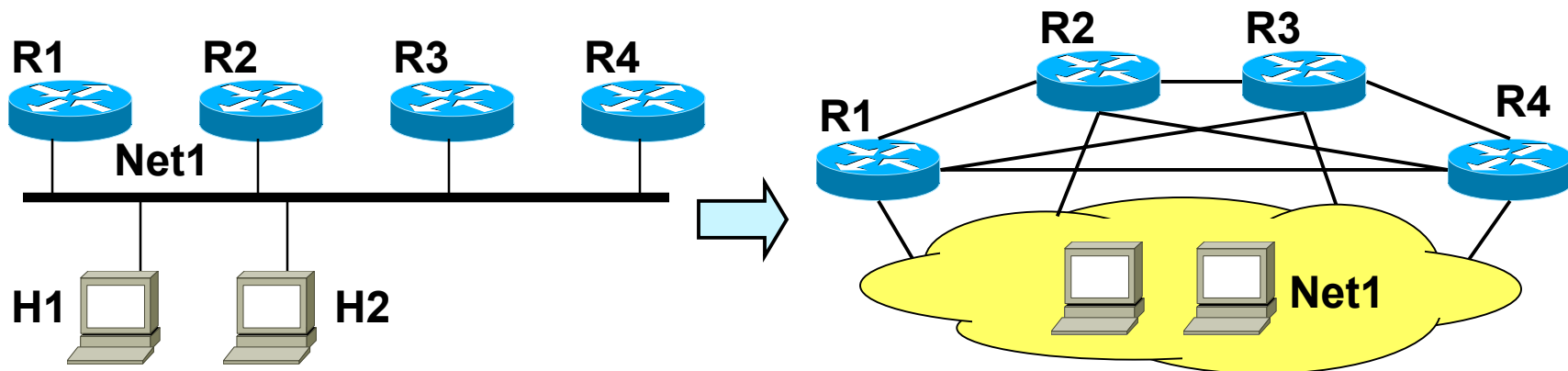


Link State Database (reale)

R1	R2/1	Net1/1
R2	R1/1	Net1/1

Link State e reti reali: reti broadcast

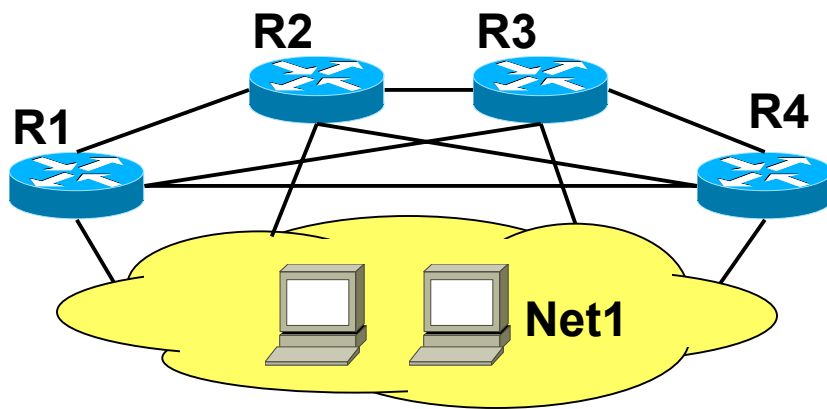
- **Link State: contiene le adiacenze**
 - N IS su rete broadcast = N^2 adiacenze (N^2 link)
 - ➔ Link State Database diventa ingestibile
 - ➔ Complessità di Dijkstra esplose (proporzionale al numero di link)
 - ➔ Fase di *Bringing up adjacencies* esplose



Link State e reti reali: reti broadcast

■ Pseudo-nodo

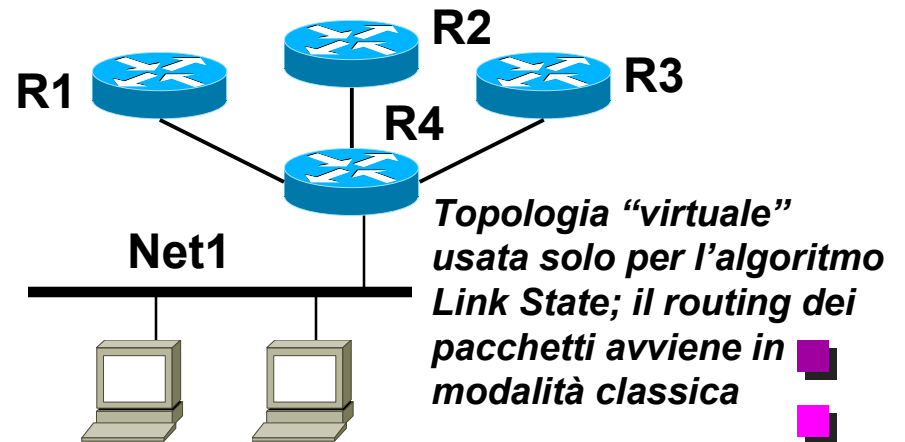
- Nodo fittizio che trasforma la topologia equivalente da maglia completa a stella
- In realtà: uno dei nodi precedenti viene "promosso" centro stella



Link State Database

```
R1  R2/1 R3/1 R4/1 Net1/1
R2  R1/1 R3/1 R4/1 Net1/1
R3  R1/1 R2/1 R4/1 Net1/1
R4  R1/1 R2/1 R3/1 Net1/1
```

linkState - 20



Link State Database

```
R1  R4/1
R2  R4/1
R3  R4/1
R4  R1/1 R2/1 R3/1 Net1/1
```

Copyright: si veda nota a pag. 2

L'algoritmo Link State

■ L'algoritmo LS

- Ha convergenza rapida
 - I LS si propagano velocemente senza alcuna elaborazione intermedia
- Difficilmente genera loop
- E' comunque in grado di identificarli e interromperli facilmente
 - Dispone della mappa della rete
- E' facile da capire e da farne il debug
 - Tutti i nodi hanno basi di dati identiche
- Presenta una scalabilità maggiore
 - Si consiglia comunque di non avere domini troppo grossi (OSPF consiglia di non avere oltre 200 router in un'area)

Distance Vector vs Link State

■ Neighbors

- LS necessita di protocolli di Neighbor Greetings
- DV conosce i vicini tramite i distance vector

■ Mappa della rete

- Gli IS LS cooperano per mantenere aggiornata la mappa della rete, poi ognuno di essi calcola il proprio spanning tree autonomamente
 - Ogni IS conosce tutta la topologia della rete e conosce esattamente il percorso per giungere a destinazione
- Gli IS DV cooperano per calcolare le tabelle di routing
 - Ogni IS conosce solo il suo intorno e ogni router si fida del vicino per inviare i dati verso la destinazione (conosce solo il next hop)

Distance Vector vs Link State

■ Semplicità

- Distance Vector: unico algoritmo
- Link State: ingloba molti componenti distinti

■ Memoria occupata (in ogni nodo)

- Dijkstra: $N * A$ [ogni LS contiene A adiacenze]
- Bellman-Ford: $A * N$ [ogni DV contiene N destinazioni]
- Valori equivalenti

■ Traffico

- Favorevole al Link State
- Pacchetti di Hello molto più piccoli rispetto a DV