

# Algoritmi di Routing Distance Vector

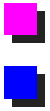
**Fulvio Riso**

**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

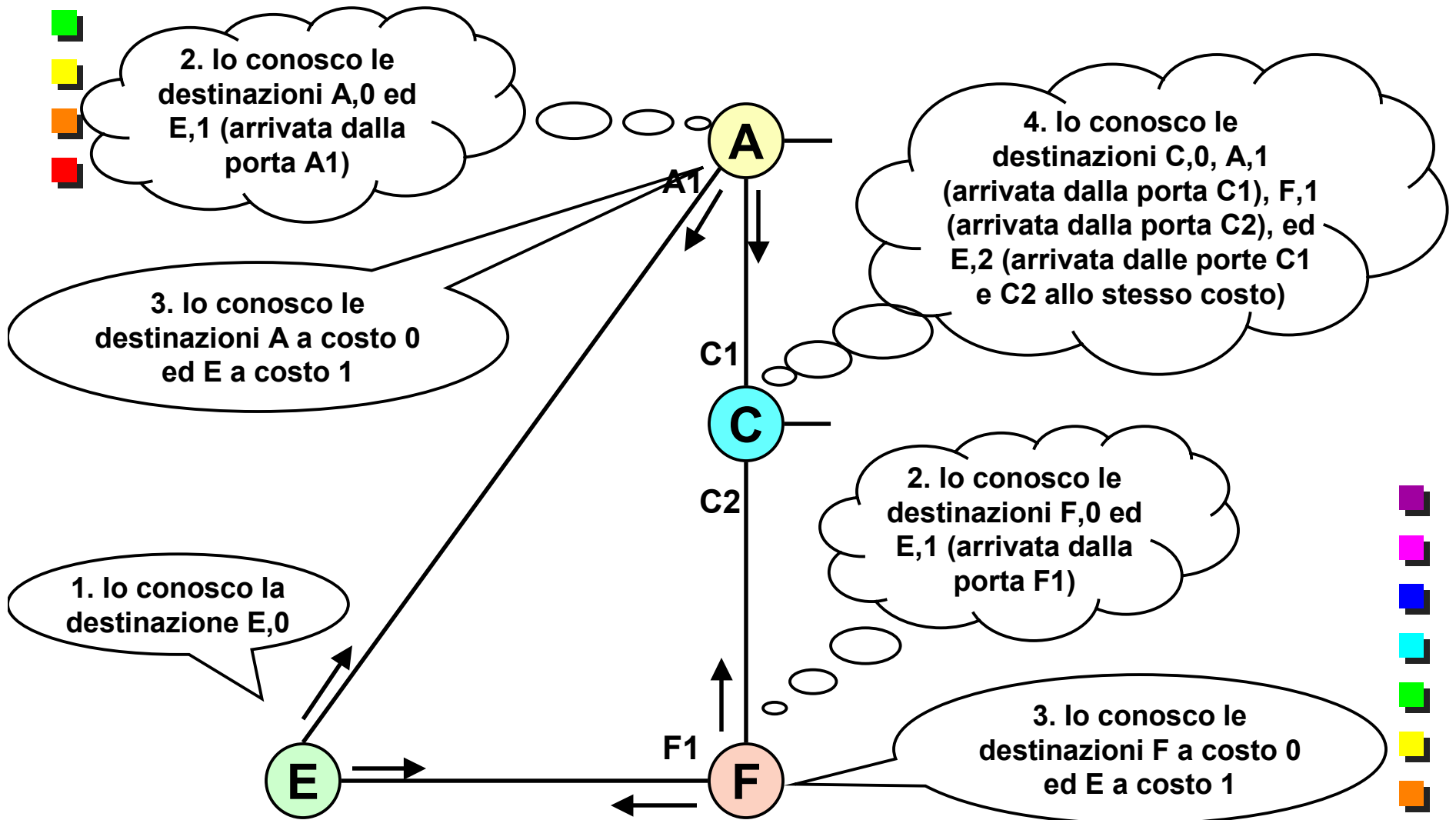
- Unione dei vantaggi di routing Isolato e Centralizzato
- Centralizzato: i router cooperano allo scambio di informazioni di connettività
- Isolato: i router sono paritetici e non esiste un router “migliore”

## ■ 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 Distance Vector





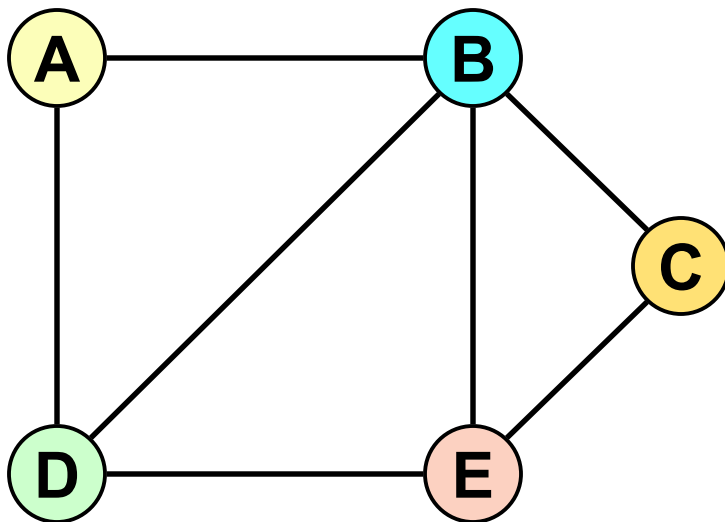
## Significato di un Distance Vector

- Ogni Distance Vector inserisce nel nodo ricevente una certa serie di informazioni:
  - Esiste una certa serie di destinazioni raggiungibili
  - Queste destinazioni si possono raggiungere in una certa direzione attraverso un certo nodo X (quello da cui è arrivato l'annuncio)
  - Il costo di raggiungimento in questa direzione è ricavabile sommando, al costo riportato nell'annuncio, il costo di attraversamento del link tra il nodo in esame e il nodo adiacente X



# Distance Vector

- Insieme di coppie destinazione – costo
  - Generato indipendentemente da ogni nodo
  - Fondamentalmente è un estratto della tabella di routing
  - Ogni nodo inserisce l'elenco delle destinazioni conosciute e il costo del miglior percorso da questo nodo alla destinazione
  - Ogni nodo memorizza i DV dei vicini più le informazioni locali al nodo stesso



distanceVector - 6

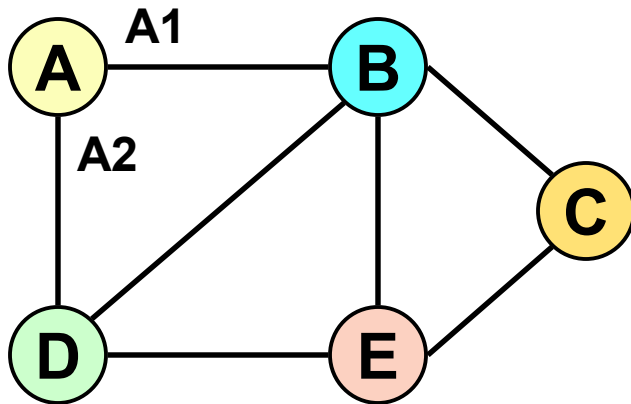
## Memoria di A

| <u>Loc (A)</u> | <u>DV (B)</u> | <u>DV (D)</u> |
|----------------|---------------|---------------|
| A, 0           | A, 1          | A, 1          |
|                | B, 0          | B, 1          |
|                | C, 1          | C, 2          |
|                | D, 1          | D, 0          |
|                | E, 1          | E, 1          |

↑  
*Informazioni  
locali al nodo*

Copyright: si veda nota a pag. 2

# Fusione e generazione di DV



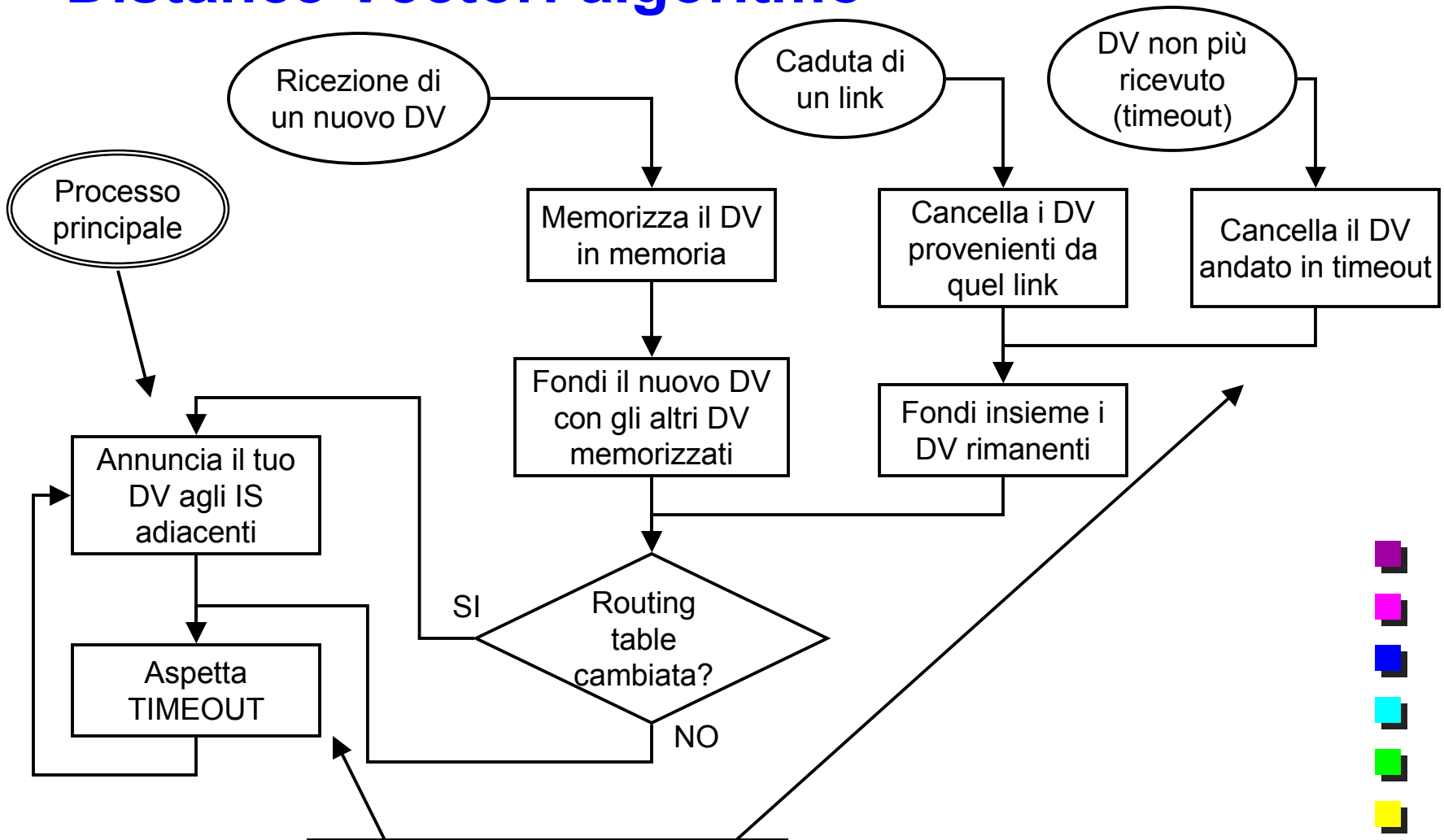
*Ricevuto dalla linea A1*

*Ricevuto dalla linea A2*

**IS A:**

| <u>Loc (A)</u> | <u>DV (B)</u>   | <u>DV (D)</u>   | <u>ROUT. TABLE (A)</u> | <u>DV (A)</u> |
|----------------|-----------------|-----------------|------------------------|---------------|
| A, 0           | <del>A, 1</del> | <del>A, 1</del> | A, local, 0            | A, 0          |
|                | B, 0            | <del>B, 1</del> | B, A1, 1               | B, 1          |
|                | C, 1            | <del>C, 2</del> | C, A1, 2               | C, 2          |
|                | <del>D, 1</del> | D, 0            | D, A2, 1               | D, 1          |
|                | E, 1            | E, 1            | E, A2, 2               | E, 2          |

# Distance Vector: algoritmo



**Affidabilità:** evita il ricorso al segnale link-up che potrebbe non essere disponibile.

# Esempio: Cold Start

| RT (A)  | RT (B)  | RT (C)  | RT (D)  | RT (E)  |
|---------|---------|---------|---------|---------|
| A,loc,0 | B,loc,0 | C,loc,0 | D,loc,0 | E,loc,0 |

A emette il DV

| RT (A)  | RT (B)  | RT (C)  | RT (D)  | RT (E)  |
|---------|---------|---------|---------|---------|
| A,loc,0 | A,B1, 1 | C,loc,0 | A,D1, 1 | E,loc,0 |
|         | B,loc,0 |         | D,loc,0 |         |

B e C emettono il DV

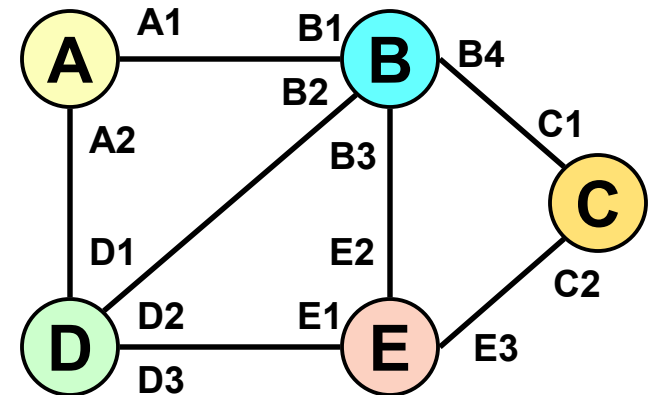
| RT (A)  | RT (B)  | RT (C)  | RT (D)  | RT (E)  |
|---------|---------|---------|---------|---------|
| A,loc,0 | A,B1, 1 | A,C1, 2 | A,D1, 1 | A,E2, 2 |
| B,A1, 1 | B,loc,0 | B,C1, 1 | B,D2, 1 | B,E2, 1 |
| D,A2, 1 | D,B2, 1 | C,loc,0 | D,loc,0 | D,E1, 1 |
|         |         |         |         | E,loc,0 |

Tutti emettono il DV

...

| RT (A)  | RT (B)  | RT (C)  | RT (D)  | RT (E)  |
|---------|---------|---------|---------|---------|
| A,loc,0 | A,B1, 1 | A,C1, 2 | A,D1, 1 | A,E2, 2 |
| B,A1, 1 | B,loc,0 | B,C1, 1 | B,D2, 1 | B,E2, 1 |
| C,A1, 2 | C,B4, 1 | C,loc,0 | C,D2, 2 | C,E3, 1 |
| D,A2, 1 | D,B2, 1 | D,C2, 2 | D,loc,0 | D,E1, 1 |
| E,A2, 2 | E,B3, 1 | E,C2, 1 | E,D3, 1 | E,loc,0 |

distanceVector - 9



Copyright: si veda nota a pag. 2

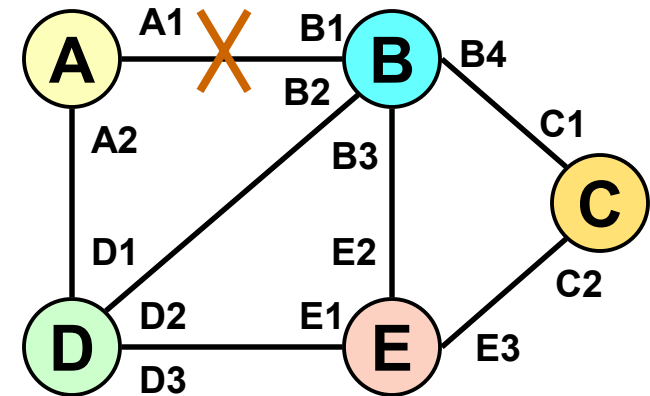
# Esempio: caduta di un link

## ■ Procedura:

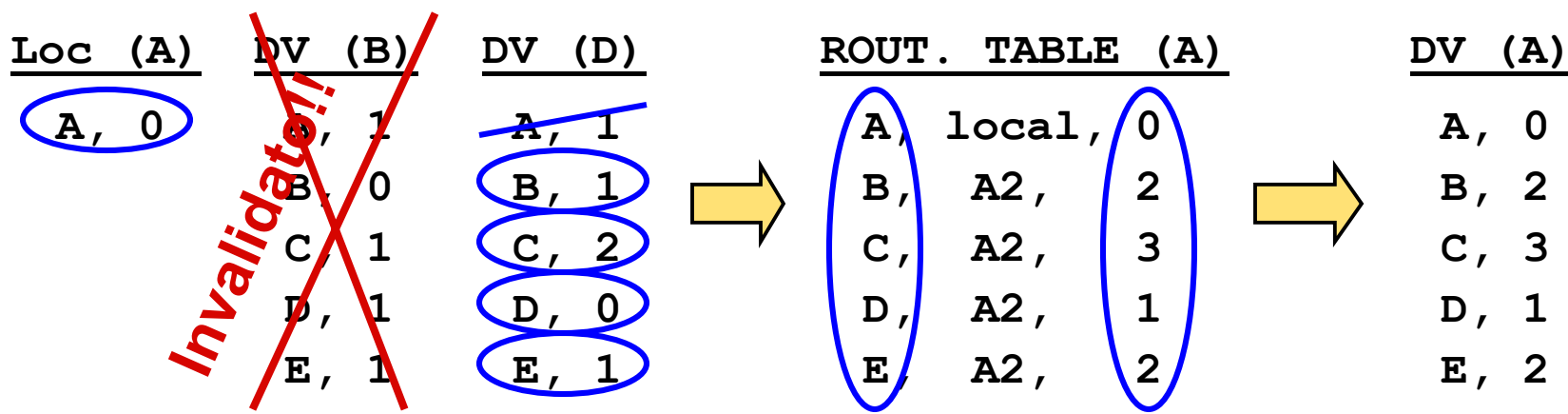
- Invalidati i DV provenienti dal link A1
- Mantenuto valido gli altri DV
- Attivato il processo di fusione

## ■ Efficienza

- Il nodo A ricava la nuova RT senza scambi di DV con i nodi adiacenti



IS A:



# Problemi


## ■ Black Hole (*dati*)

- I pacchetti per una destinazione sono inviati ad un router il quale, non disponendo di una route per la destinazione, li scarta

## ■ Bouncing Effect (*dati*)

- Un pacchetto è inoltrato su un percorso circolare (*routing loop*)
- Normalmente il pacchetto contiene un contatore (*time to live*) che ne limita la vita ad un massimo di N hops

## ■ Count to Infinity (*route*)

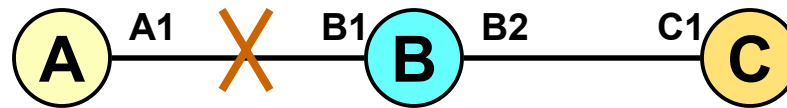
- Il costo per il raggiungimento di una destinazione (normalmente non più raggiungibile, viene progressivamente incrementato (all'infinito)
- Allunga la durata dal transitorio

## ■ Black Hole e Bouncing Effect

- Comuni anche all'algoritmo Link State; più evidenti nel Distance Vector
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 



# Count to Infinity



IS B:

| <del>Loc (B)</del> | <del>DV (A)</del> | DV (C) | RT (B)   |
|--------------------|-------------------|--------|----------|
| <del>B, 0</del>    | <del>A, 0</del>   | A, 2   | A,B2, 3  |
| <del>B, 1</del>    | <del>B, 1</del>   | B, 1   | B,loc, 0 |
| <del>C, 2</del>    | <del>C, 2</del>   | C, 0   | C,B2, 1  |

IS C:

| Loc (C) | DV (B) | RT (C)   |
|---------|--------|----------|
| C, 0    | A, 1   | A,C1, 2  |
|         | B, 0   | B,C1, 1  |
|         | C, 1   | C,loc, 0 |

B emette il DV

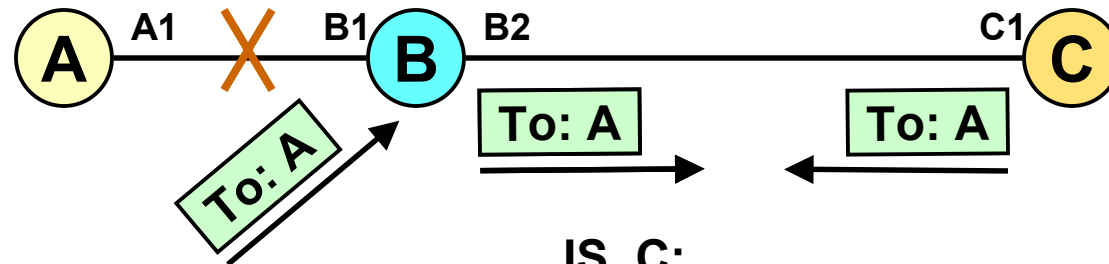
| Loc (C) | DV (B) | RT (C)   |
|---------|--------|----------|
| C, 0    | A, 3   | A,C1, 4  |
|         | B, 0   | B,C1, 1  |
|         | C, 1   | C,loc, 0 |

C emette il DV

| Loc (B) | DV (C) | RT (B)   |
|---------|--------|----------|
| B, 0    | A, 4   | A,B2, 5  |
|         | B, 1   | B,loc, 0 |
|         | C, 0   | C,B2, 1  |

Count to Infinity!

# Bouncing Effect



IS B:

| Loc (B) | <del>DV (A)</del> | DV (C) | RT (B)          |
|---------|-------------------|--------|-----------------|
| B, 0    | <del>A, 0</del>   | A, 2   | <u>A, B2, 3</u> |
|         | <del>B, 1</del>   | B, 1   | B, loc, 0       |
|         | <del>C, 2</del>   | C, 0   | C, B2, 1        |

IS C:

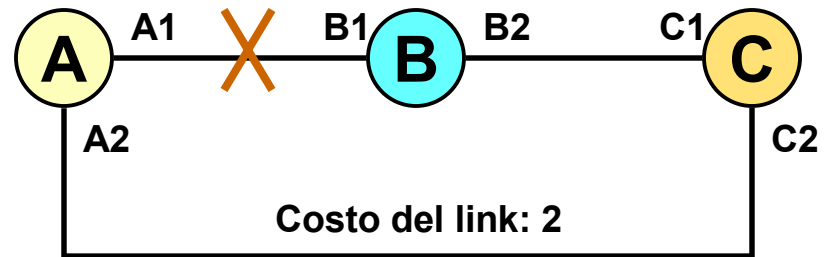
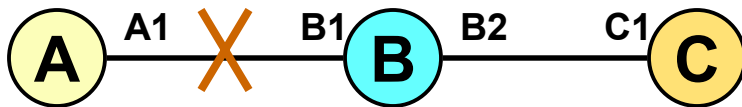
| Loc (C) | DV (B) | RT (C)    |
|---------|--------|-----------|
| C, 0    | A, 1   | A, C1, 2  |
|         | B, 0   | B, C1, 1  |
|         | C, 1   | C, loc, 0 |

B emette il DV

| Loc (C) | DV (B) | RT (C)          |
|---------|--------|-----------------|
| C, 0    | A, 3   | <u>A, C1, 4</u> |
|         | B, 0   | B, C1, 1        |
|         | C, 1   | C, loc, 0       |

# Conoscenza topologica

- Il Distance Vector non riconosce la topologia
  - Il DV di C è identico (tranne per la destinazione D) nei due casi
    - Nel primo darà origine ad un Count to infinity
    - Nel secondo caso, invece no



IS B:


|                |                   |        |          |
|----------------|-------------------|--------|----------|
| <u>Loc (B)</u> | <del>DV (A)</del> | DV (C) | RT (B)   |
| B, 0           | <del>A, 0</del>   | A, 2   | A,B2, 3  |
|                | <del>B, 1</del>   | B, 1   | B,loc, 0 |
|                | <del>C, 2</del>   | C, 0   | C,B2, 1  |

IS B:

|                |                   |        |          |
|----------------|-------------------|--------|----------|
| <u>Loc (B)</u> | <del>DV (A)</del> | DV (C) | RT (B)   |
| B, 0           | <del>A, 0</del>   | A, 2   | A,B2, 3  |
|                | <del>B, 1</del>   | B, 1   | B,loc, 0 |
|                | <del>C, 2</del>   | C, 0   | C,B2, 1  |



## Soluzioni

- Aggiunte / modifiche all'algoritmo Distance Vector originale
  - Algoritmi più noti
    - Split Horizon
    - Path Hold Down
    - Route Poisoning
  - Soluzioni sempre parziali
    - Distance Vector: presenta il problema di fondo legato alla mancata conoscenza delle topologia
    - Ulteriori tecniche appesantiscono il protocollo e tendono comunque a non renderlo affidabile al 100%
- 

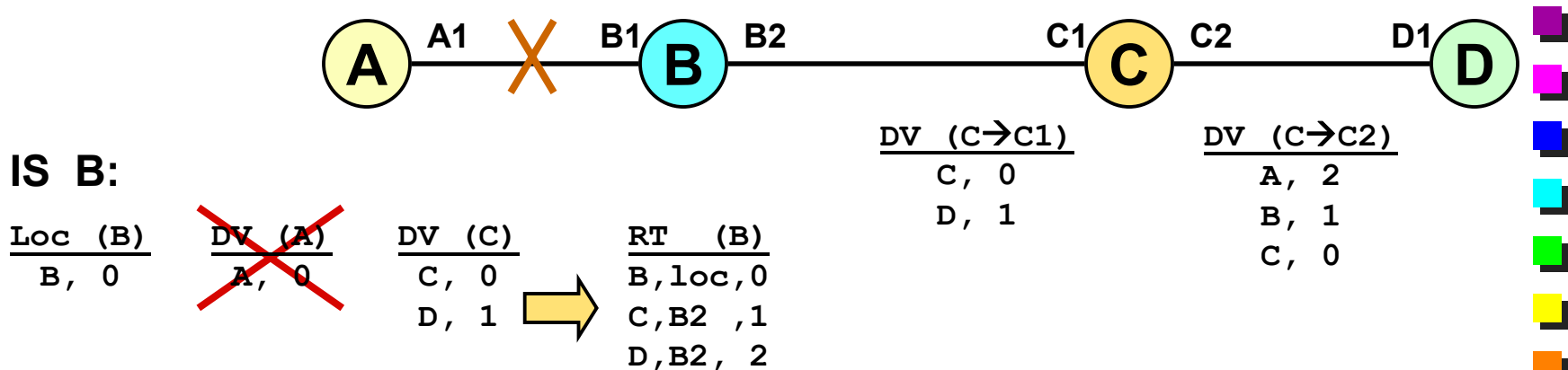


# Split Horizon

*“Se C raggiunge la destinazione A attraverso B, non ha senso per B cercare di raggiungere A attraverso C”*

## ■ Caratteristiche

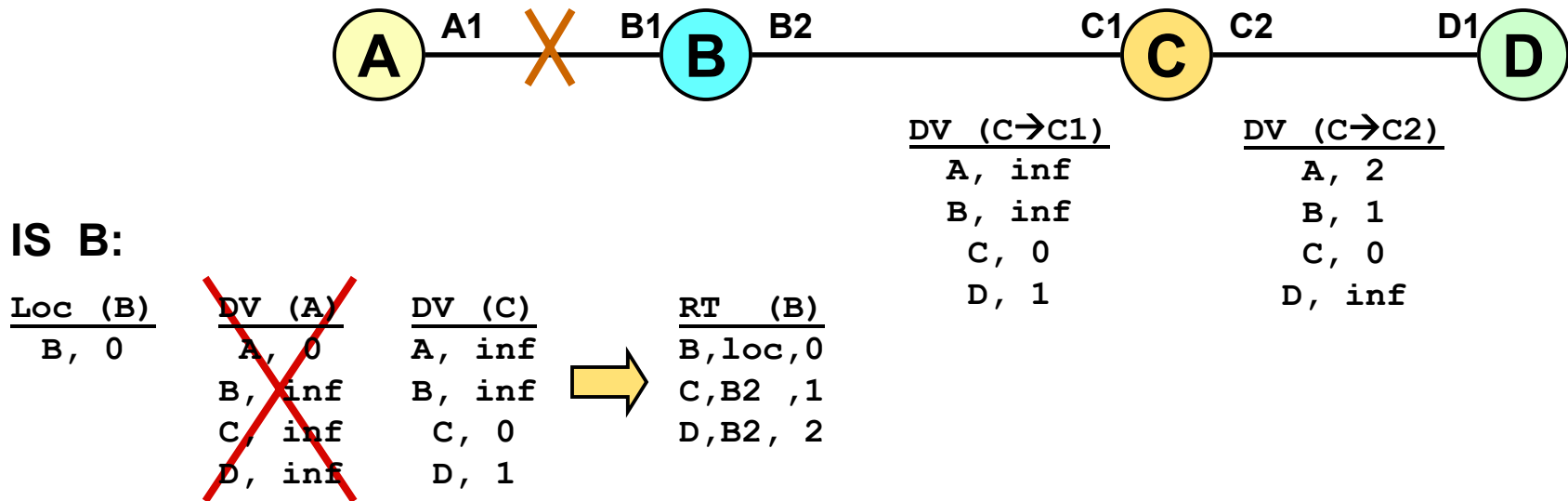
- Previene loop tra due nodi
- Rende la convergenza più veloce
- Consiste nel differenziare i distance vector inviati sulle varie linee del router: il DV di C verso B non conterrà le destinazioni raggiungibili attraverso a B
  - Gli IS dovranno calcolare un diverso DV per ogni linea



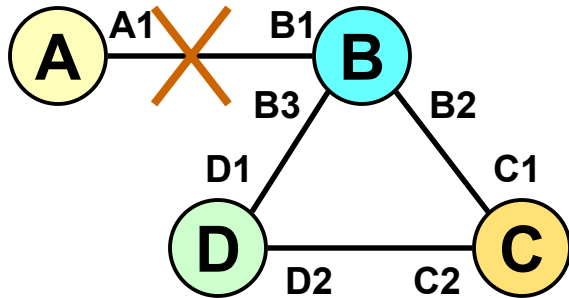
# Split Horizon con Poisonous Reverse

## ■ Più aggressivo

- Non presenta vantaggi dal punto di vista teorico
- In pratica aumenta la velocità di convergenza in quanto non aspetta lo scadere del timeout per le route che vengono omesse



# Split Horizon e maglie



IS C:

| Loc (C) | DV (B) | DV (D) | RT (C)  |
|---------|--------|--------|---------|
| C, 0    | A, 1   | A, 2   | A,C1, 2 |
|         | B, 0   | B, 1   | B,C1, 1 |
|         | D, 1   | D, 0   | C,loc,0 |
|         |        |        | D,C2, 1 |

IS D:

| Loc (D) | DV (B) | DV (C) | RT (D)  |
|---------|--------|--------|---------|
| D, 0    | A, 1   | A, 2   | A,D1, 2 |
|         | B, 0   | B, 1   | B,D1, 1 |
|         | C, 1   | C, 0   | C,D2, 1 |
|         |        |        | D,loc,0 |

IS B:

| Loc (B) | <del>DV (A)</del> | DV (C) | DV (D) | RT (B)  |
|---------|-------------------|--------|--------|---------|
| B, 0    | <del>A, 0</del>   | C, 0   | C, 1   | B,loc,0 |
|         |                   | D, 1   | D, 0   | C,B2, 1 |
|         |                   |        |        | D,B2, 1 |

B emette il DV

IS C:

| Loc (C) | DV (B) | DV (D) | RT (C)  |
|---------|--------|--------|---------|
| C, 0    | B, 0   | A, 2   | A,C2, 3 |
|         | D, 1   | B, 1   | B,C1, 1 |
|         |        | D, 0   | C,loc,0 |
|         |        |        | D,C2, 1 |

IS D:

| Loc (D) | DV (B) | DV (C) | RT (D)  |
|---------|--------|--------|---------|
| D, 0    | B, 0   | A, 2   | A,D2, 3 |
|         | C, 1   | B, 1   | B,D1, 1 |
|         |        | C, 0   | C,D2, 1 |
|         |        |        | D,loc,0 |

# Split Horizon e maglie

**IS C:** (dalla slide precedente)

| Loc (C) | DV (B) | DV (D) | RT (C)    |
|---------|--------|--------|-----------|
| C, 0    | B, 0   | A, 2   | A, C2, 3  |
|         | D, 1   | B, 1   | B, C1, 1  |
|         |        | D, 0   | C, loc, 0 |
|         |        |        | D, C2, 1  |

**IS D:**

| Loc (D) | DV (B) | DV (C) | RT (D)    |
|---------|--------|--------|-----------|
| D, 0    | B, 0   | A, 2   | A, D2, 3  |
|         | C, 1   | B, 1   | B, D1, 1  |
|         |        | C, 0   | C, D2, 1  |
|         |        |        | D, loc, 0 |

C e D emettono il DV

**IS B:**

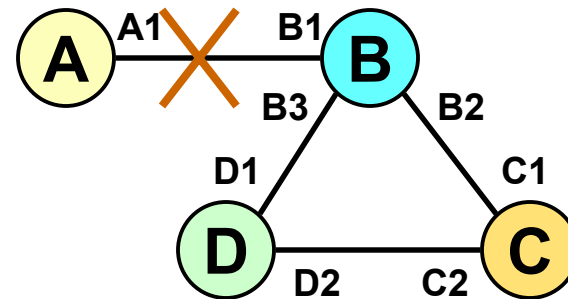
| Loc (B) | DV (C) | DV (D) | RT (B)    |
|---------|--------|--------|-----------|
| B, 0    | A, 3   | A, 3   | A, B3, 4  |
|         | C, 0   | C, 1   | B, loc, 0 |
|         | D, 1   | D, 0   | C, B2, 1  |
|         |        |        | D, B2, 1  |

**IS C:**

| Loc (C) | DV (B) | DV (D) | RT (C)    |
|---------|--------|--------|-----------|
| C, 0    | B, 0   | B, 1   | B, C1, 1  |
|         | D, 1   | D, 0   | C, loc, 0 |
|         |        |        | D, C2, 1  |

**IS D:**

| Loc (D) | DV (B) | DV (C) | RT (D)    |
|---------|--------|--------|-----------|
| D, 0    | B, 0   | B, 1   | B, D1, 1  |
|         | C, 1   | C, 0   | C, D2, 1  |
|         |        |        | D, loc, 0 |

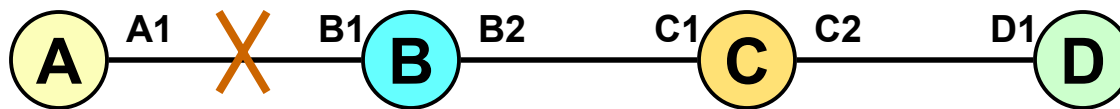


# Path Hold Down

*“Se il link L cade, tutte le route che utilizzano il link L non potranno essere modificate per un certo periodo e le destinazioni verranno considerate irraggiungibili”*

## ■ Caratteristiche

- Alto tempo di convergenza per il nodo in esame (anche se esiste in percorso alternativo)
- Il router che ha rilevato il guasto non può partecipare ad alcun loop fino almeno alla scadenza dell’Hold Down timer



IS B:

| Loc (B) | DV (A) | DV (C)          | RT (B)    |
|---------|--------|-----------------|-----------|
| B, 0    | A, 0   | <del>A, 2</del> | B, loc, 0 |
|         | B, 1   | B, 1            | C, B2, 1  |
|         | C, 2   | C, 0            | D, B2, 2  |
|         | D, 3   | D, 1            |           |

**Quarantena!**

IS C:

| Loc (C) | DV (B) | DV (D) | RT (C)          |
|---------|--------|--------|-----------------|
| C, 0    | B, 0   | A, 3   | <u>A, C2, 4</u> |
|         | C, 1   | B, 2   | B, C1, 1        |
|         | D, 2   | C, 1   | C, loc, 0       |
|         |        | D, 0   | D, C2, 1        |

**Count to Infinity!**



# Route Poisoning

- **Idea: un routing loop è individuato dalla ricezione di una route a costo crescente**
  - Si blocca l'utilizzo di tutte le route che aumentano di costo
  - La route viene sbloccata quando due annunci successivi confermano lo stesso costo per quella route
- **Caratteristiche**
  - Risolve più casi rispetto al path Hold Down
  - Ha tempo di convergenza normalmente più rapido (non è necessario aspettare l'Hold Down timer ma è sufficiente che due DV successivi confermino la stessa informazione)
  - Blocca anche eventuali route il cui incremento di costo potrebbe essere legittimo
  - Implementato in IGRP come sostituto del Path Hold Down






# Distance Vector: pregi e difetti

## ■ Pregi:

- Facilità di implementazione

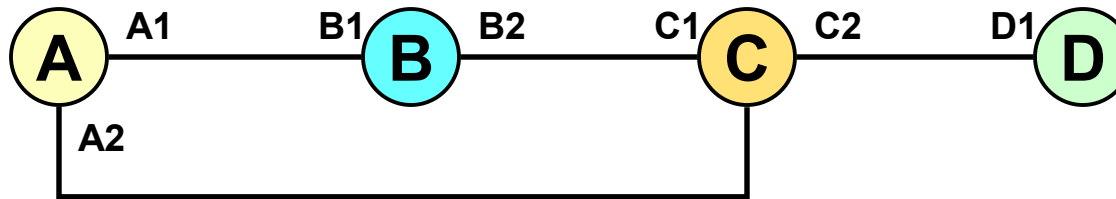
## ■ Difetti:

- L'algoritmo ha un worst case esponenziale e un comportamento normale tra  $O(n^2)$  e  $O(n^3)$ ;
  - Lenta convergenza proporzionale a link e router più lenti;
  - Difficile capirne e prevederne il comportamento su reti grandi e complesse
    - Nessun nodo ha una mappa della rete
    - L'uso è quindi limitato a reti non troppo magliate
  - Possono innescarsi routing loop a causa di particolari variazioni della topologia
  - La realizzazione di meccanismi migliorativi appesantisce notevolmente il protocollo
  - La soglia "infinito" limita l'impiego di questo algoritmo a reti piccole (ad es. con pochi hop)
    - La soglia può essere personalizzata da management
- 



# Path Vector

- Inserisce l'informazione del percorso necessario a raggiungere la destinazione
  - Evita l'insorgenza di routing loop
  - BGP: non contempla il costo



IS A:

|                |               |               |   |               |   |               |
|----------------|---------------|---------------|---|---------------|---|---------------|
| <u>Loc (A)</u> | <u>PV (B)</u> | <u>PV (C)</u> |   | <u>RT (A)</u> |   | <u>PV (A)</u> |
| A, 0           | A, 1, [B]     | A, 1 [C]      |   | A, loc, 0     | → | A, 0, [-]     |
|                | B, 0, [-]     | B, 1, [B]     | → | B, A1, 1      |   | B, 1, [A]     |
|                | C, 1, [B]     | C, 0, [-]     |   | C, A2, 1      |   | C, 1, [A]     |
|                | D, 2, [B,C]   | D, 1, [D]     |   | D, A2, 2      |   | D, 2, [A,C]   |




# 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
- 

