UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

CDS IN INFORMATICA E COMUNICAZIONE DIGITALE

**Anno Accademico 2015-2016**

***Corso di***

***"Reti di Calcolatori e Comunicazione Digitale"***

# *Modulo 4 : TCP/IP : livello 3, algoritmi di routing*

*Prof. Sebastiano Pizzutilo*
*Dipartimento di Informatica*

# *Il Routing*

***1) CALCOLO DEL PERCORSO / INSTRADAMENTO / ROUTING = Determinazione del percorso ottimale dei messaggi in base alle informazioni della ROUTING TABLE.***

***2) INOLTRO / FORWARDING = Invio del pacchetto verso l'interfaccia di output del router collegata in rete al router cui è destinato il pacchetto o al router di default.***

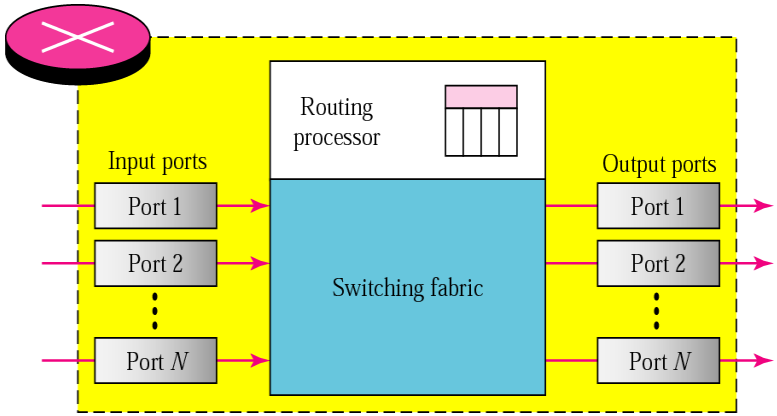
***INOLTRO dei pacchetti verso l'host destinazione :***

- *Acquisito l'indirizzo IP di un pacchetto da inoltrare, il router controlla,* ***attraverso la propria netmask, se è relativo ad un host della propria rete****;*
- ***Se l'indirizzo appartiene alla stessa rete del router****, l'IP del router utilizzerà i servizi dello strato inferiore (data-link) per spedire il pacchetto* ***direttamente*** *all'host destinatario.*
- ***Se l'indirizzo appartiene ad un'altra rete****, il router consulterà la propria* ***Routing Table****, che* ***associa ad ogni rete l'indirizzo del router di frontiera delle reti connesse****.*

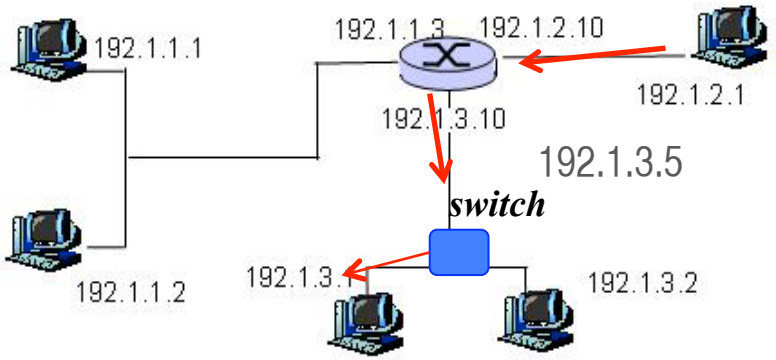
  *Se il router (che è collegato a più reti) ha una connessione alla rete dove* ***è collegato l'host destinatario*** *gli* ***inoltra direttamente*** *il pacchetto, altrimenti lo passa al router più vicino,* ***fino a raggiungere un router in grado di consegnare il pacchetto all'host destinatario****.*

# Il ROUTER

*Componenti base di un router*




- La CPU calcola la tabella di routing, esegue i protocolli e gestisce l'intero sistema.
- Le interfacce memorizzano i pacchetti ed eseguono i lookup su copie locali della tabella.
- La matrice di commutazione *(switching fabric)* gestisce l'inoltro dei pacchetti *e* permette il trasferimento diretto e contemporaneo di più pacchetti.



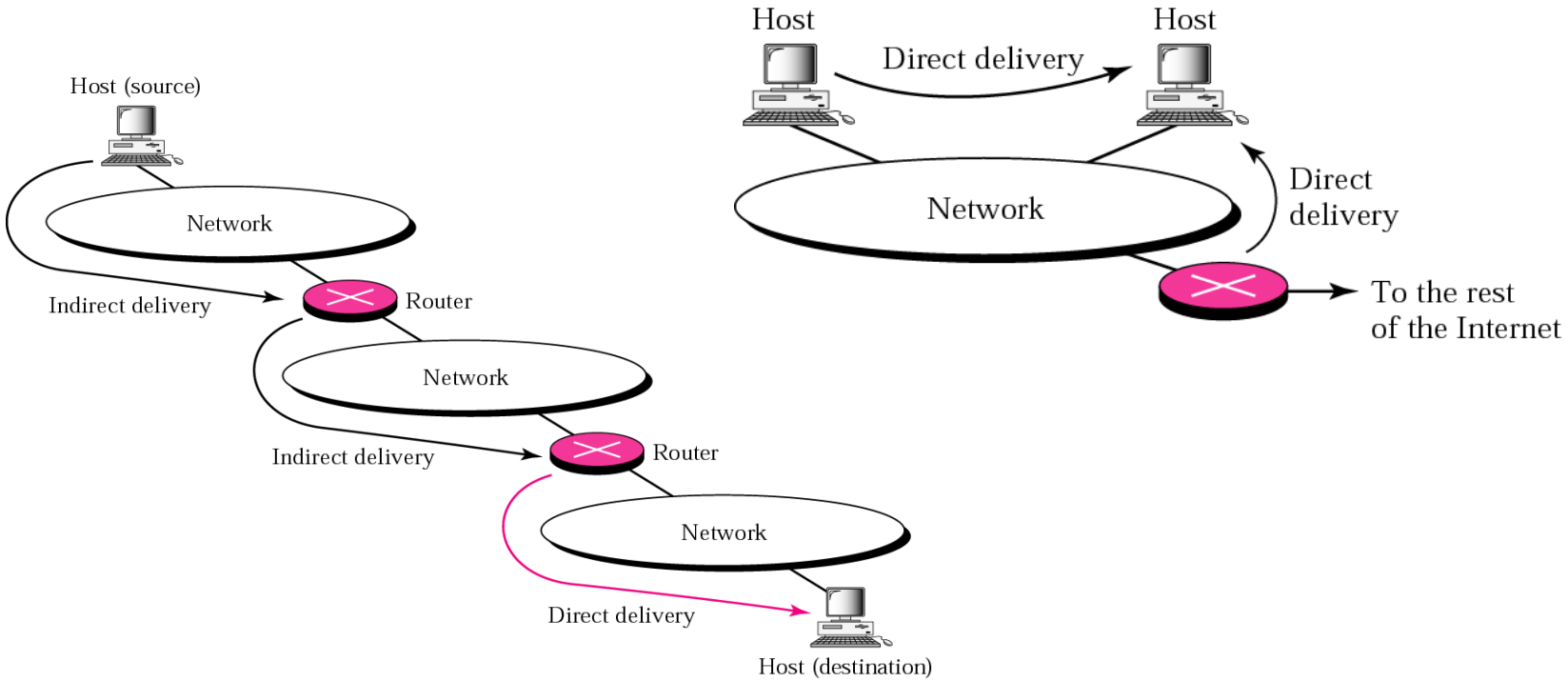
# *La tabella di routing*

***La tabella di routing contiene:***

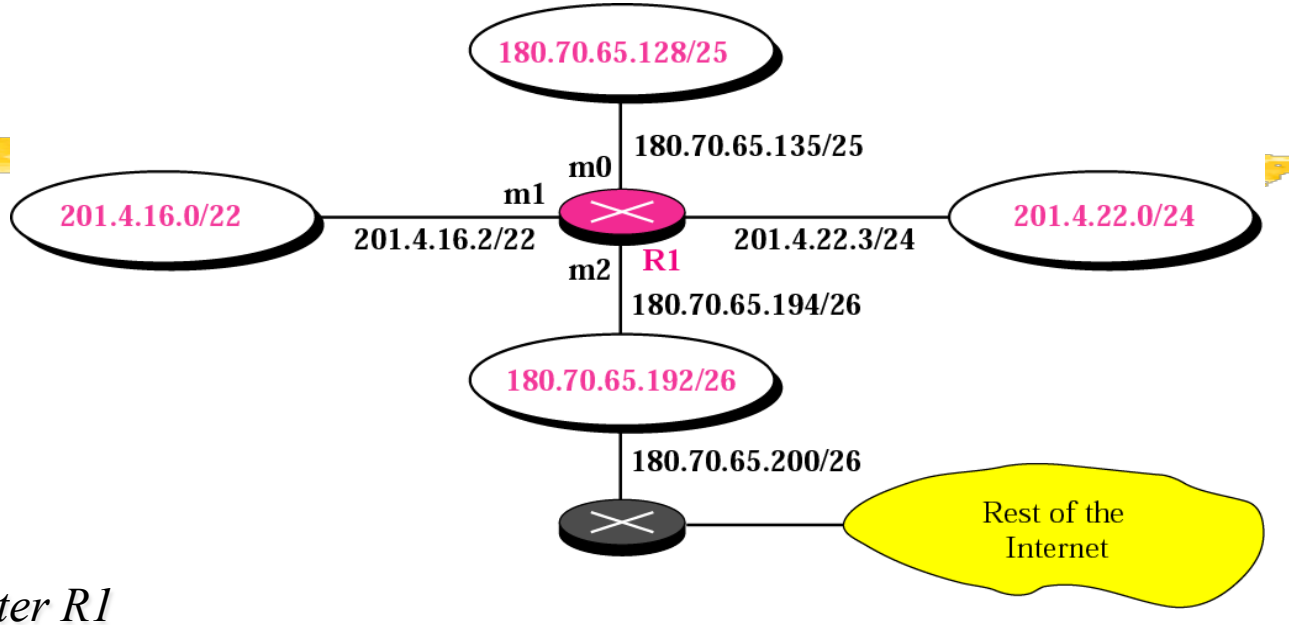
- un record **per ciascuna rete collegata direttamente** al router, insieme con l'indicazione della relativa **interfaccia di rete**;
- un numero variabile di record **per reti non collegate direttamente** al router, insieme con l'indicazione di un **router adiacente** a cui i pacchetti devono essere inviati;
- un record per un **router adiacente di default**, a cui inviare i pacchetti destinati a reti sconosciute.

- *L'ampiezza dei campi "Network Prefix" e "Host-ID" viene definita tramite il parametro **netmask**.*
- *Nel caso in cui due indirizzi (mittente e destinatario) appartengano alla stessa subnet, l'host mittente pacchetto invierà il pacchetto direttamente verso il destinatario (**routing diretto**),*
- *nel caso in cui due indirizzi (mittente e destinatario) appartengano a diverse subnet il mittente invierà il pacchetto ad un router di default o al router del destinatario, se conosciuto **(routing indiretto)** .*

# ROUTING DIRETTO E INDIRETTO



# *Esempio tabella di routing*




*Routing table for router R1*

| *Mask* | *Network Address* | *Next Hop* | *Interface* |
|---|---|---|---|
| /26 | 180.70.65.192 | - | m2 |
| /25 | 180.70.65.128 | - | m0 |
| /24 | 201.4.22.0 | - | m3 |
| /22 | 201.4.16.0 | .... | m1 |
| Default | Default | 180.70.65.200 | m2 |

# *Tabelle di routing*

| Nome | Descrizione |
|---|---|
| Destination | La rete o il nodo di destinazione. |
| Gateway | Il router. Se appare un asterisco (‘*’) o l’indirizzo 0.0.0.0 significa che non si tratta di un instradamento attraverso un router. |
| Genmask | In linea di massima corrisponde alla maschera di rete; in particolare, se è un instradamento verso un nodo appare 255.255.255.255, se invece è l’instradamento predefinito appare 0.0.0.0 (‘**default**’). |
| Flags | Indica diversi tipi di informazioni utilizzando lettere o simboli. |
| Metric | La distanza o il costo della strada. Rappresenta la distanza (espressa solitamente in *hop* o salti) per raggiungere la destinazione. |
| Ref | Il numero di riferimenti all’instradamento. Questa informazione non viene utilizzata dal kernel Linux e, di conseguenza, l’informazione appare sempre azzerata. |
| Use | Conteggio del numero di volte in cui la voce è stata visionata. |
| Iface | Il nome dell’interfaccia da cui partono i pacchetti IP. |

$ **route -n**[ *Invio* ]

| Destination | Gateway | Genmask | Flags | Metric | Ref | Use | Iface |
|---|---|---|---|---|---|---|---|
| 192.168.21.0 | 0.0.0.0 | 255.255.255.128 | U | 0 | 0 | 0 | eth1 |
| 193.204.187.0 | 0.0.0.0 | 255.255.255.0 | U | 0 | 0 | 0 | eth0 |
| 0.0.0.0 | 193.204.187.1 | 0.0.0.0 | UG | 0 | 0 | 0 | eth0 |

# *Esempio di routing table su host LINUX*

***route*** è un comando che permette di vedere e modificare la tabella di routing.

*ad es.:* ***route –n***

Kernel IP routing table

| Destination | Gateway | Genmask | Flags | Metric | Ref | Use | Iface |
|---|---|---|---|---|---|---|---|
| 192.168.101.0 | 192.168.102.102 | 255.255.255.0 | UG | 0 | 0 | 0 | eth0 |
| 192.168.102.0 | 0.0.0.0 | 255.255.255.0 | U | 0 | 0 | 0 | eth0 |
| 192.168.103.0 | 192.168.102.102 | 255.255.255.0 | UG | 0 | 0 | 0 | eth0 |
| 192.168.12.0 | 0.0.0.0 | 255.255.255.0 | U | 0 | 0 | 0 | eth0 |
| 0.0.0.0 | 192.168.12.1 | 0.0.0.0 | UG | 0 | 0 | 0 | eth0 |

*Nell'esempio* ***il computer dove è stato lanciato il comando :***

***-*** *si connette con l'interfaccia* ***eth0*** *direttamente alle reti* ***192.168.102.0*** *e* ***192.168.12.0*** *(il campo* ***gateway*** *indicato con* ***0.0.0.0*** *indica che* ***non*** *passa dal router) (righe 2 e 4);*

*- si connette con l'interfaccia* ***eth0*** *alle reti* ***192.168.101.0*** *e* ***192.168.103.0*** *tramite il* ***gateway 192.168.102.102*** *(righe 1 e 3);*

*- per tutte le altre destinazioni (****0.0.0.0****) si connette con l'interfaccia* ***eth0*** *mediante il gateway* ***192.168.12.1*** *(riga 5).*

# Metriche di routing

## *Parametri per selezionare il miglior percorso*

***Due parametri (metriche) universalmente accettati sono:***

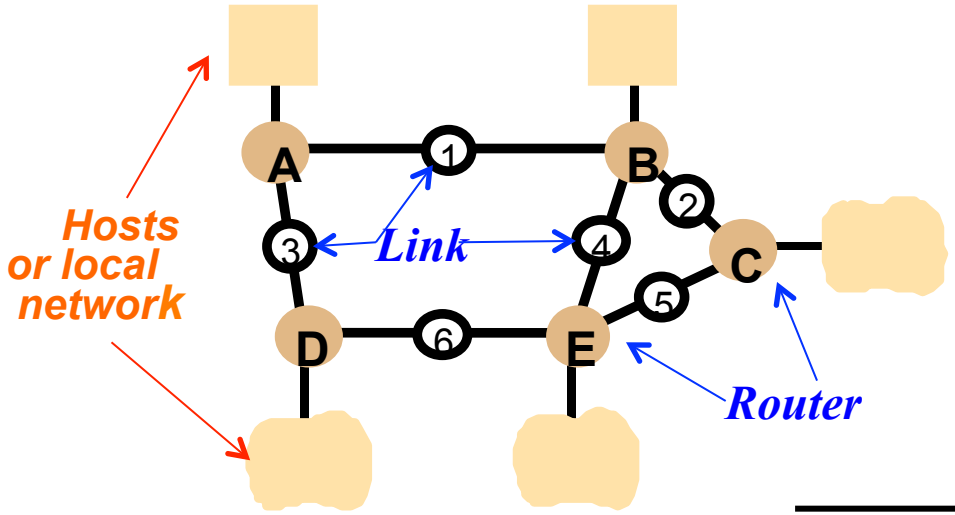
- ***Hops:*** numero di salti effettuati, cioè il numero di IS attraversati lungo il cammino,

- ***Costo:*** somma dei costi di tutte le linee attraversate; il costo di una linea è inversamente proporzionale alla sua velocità (banda trasmissiva, tipo e affidabilità del mezzo trasmissivo, lunghezza del percorso, traffico di rete,….)

# Distance Vector Routing

**Gli algoritmi del tipo *distance vector* effettuano l'invio da parte di ciascun router della propria tabella ai soli router vicini.**

**La tabella inviata ha l'aspetto di un vettore (ad es. il router A invia sul link 1 a B il vettore "A=0,0, B=1,1, C=1,2, D=3,1, E=1,2" )**




*Routing from A*

| *To* | *Link* | *Cost* |
|---|---|---|
| A | local | 0 |
| B | 1 | 1 |
| C | 1 | 2 |
| D | 3 | 1 |
| E | 1 | 2 |

*Routing from B*

| *To* | *Link* | *Cost* |
|---|---|---|
| A | 1 | 1 |
| B | local | 0 |
| C | 2 | 1 |
| D | 1 | 2 |
| E | 4 | 1 |

*Routings from C*

| *To* | *Link* | *Cost* |
|---|---|---|
| A | 2 | 2 |
| B | 2 | 1 |
| C | local | 0 |
| D | 5 | 2 |
| E | 5 | 1 |

*Routings from D*

| *To* | *Link* | *Cost* |
|---|---|---|
| A | 3 | 1 |
| B | 3 | 2 |
| C | 6 | 2 |
| D | local | 0 |
| E | 6 | 1 |

*Routings from E*

| *To* | *Link* | *Cost* |
|---|---|---|
| A | 4 | 2 |
| B | 4 | 1 |
| C | 5 | 1 |
| D | 6 | 1 |
| E | local | 0 |

# *Algoritmo di routing: Distance Vector* *(2)*

•E' ***distribuito***, nel senso che ciascun nodo riceve parte dell'informazione da uno o più dei suoi vicini direttamente connessi.
•E' ***iterativo***, nel senso che questo processo si ripete fino a quando non avviene ulteriore scambio informativo tra vicini.
•E' ***asincrono***, nel senso che non richiede che tutti i nodi operino al passo con gli altri.

Il calcolo del percorso viene realizzato tramite l'algoritmo di ***Bellman-Ford***

# *ALGORITMO DI BELLMAN-FORD* (3)

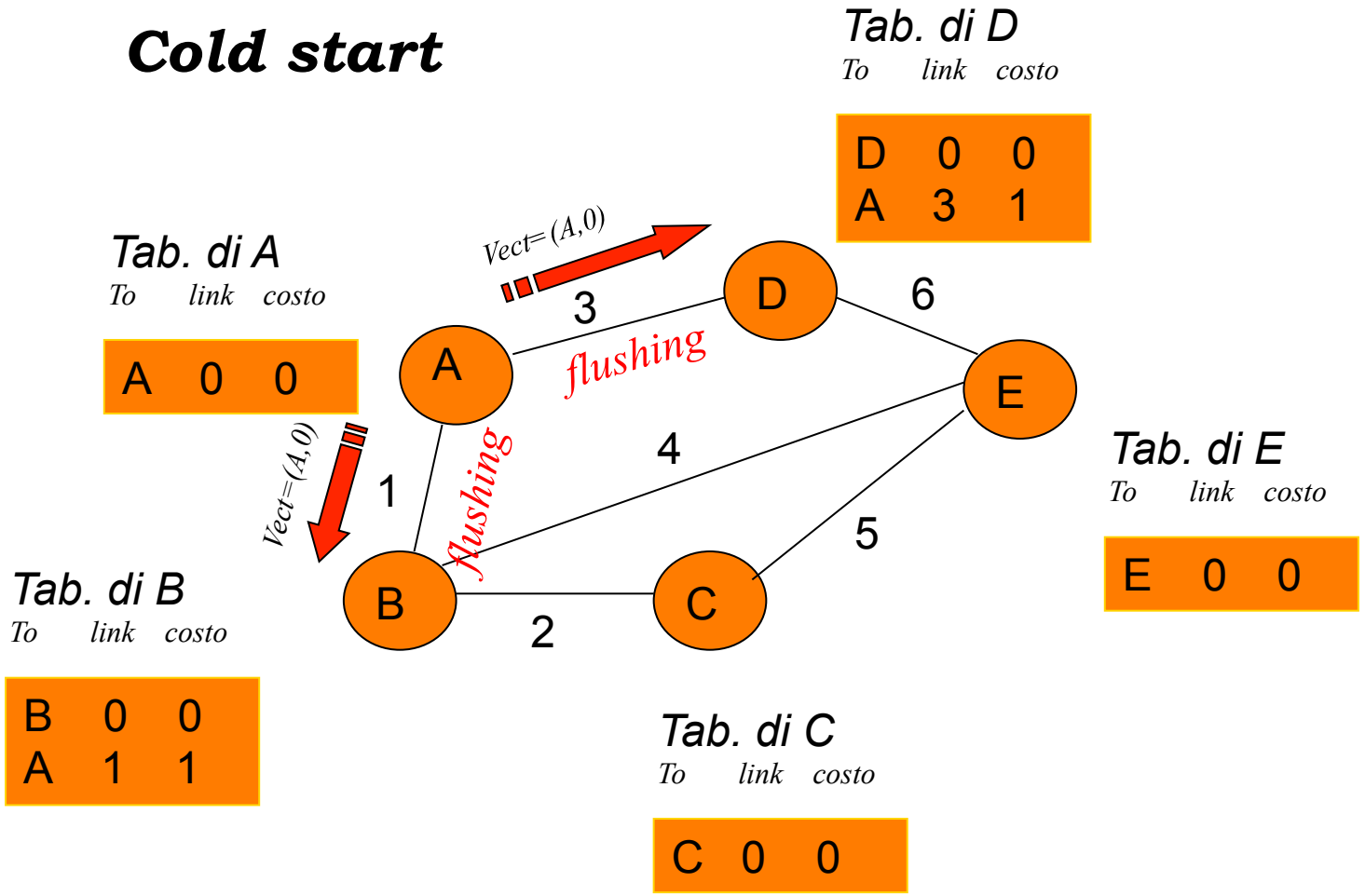
**I protocolli distance vector sono basati sull'algoritmo di *Bellman-Ford = versione distribuita* di un algoritmo molto semplice per la ricerca del percorso più breve in un grafo rappresentato in forma tabellare.**

- ***N*** = *numero dei nodi,* ***M*** = *numero dei links.*
- ***L*** = *tabella dei link di dimensione M, dove :* ***L[l].m = metrica del link,***
  ***L[l].s = sorgente del link,*** ***L[l].d = destinazione del link .***
- ***D*** = *matrice delle distanze di dimensione [N,N], dove* ***D[i,j] e' la distanza da i a j***
- ***H*** = *matrice dei link di dimensione [N,N], dove* ***H[i,j] e' il link*** *sul quale* ***i*** *instrada i pacchetti destinati a* ***j***

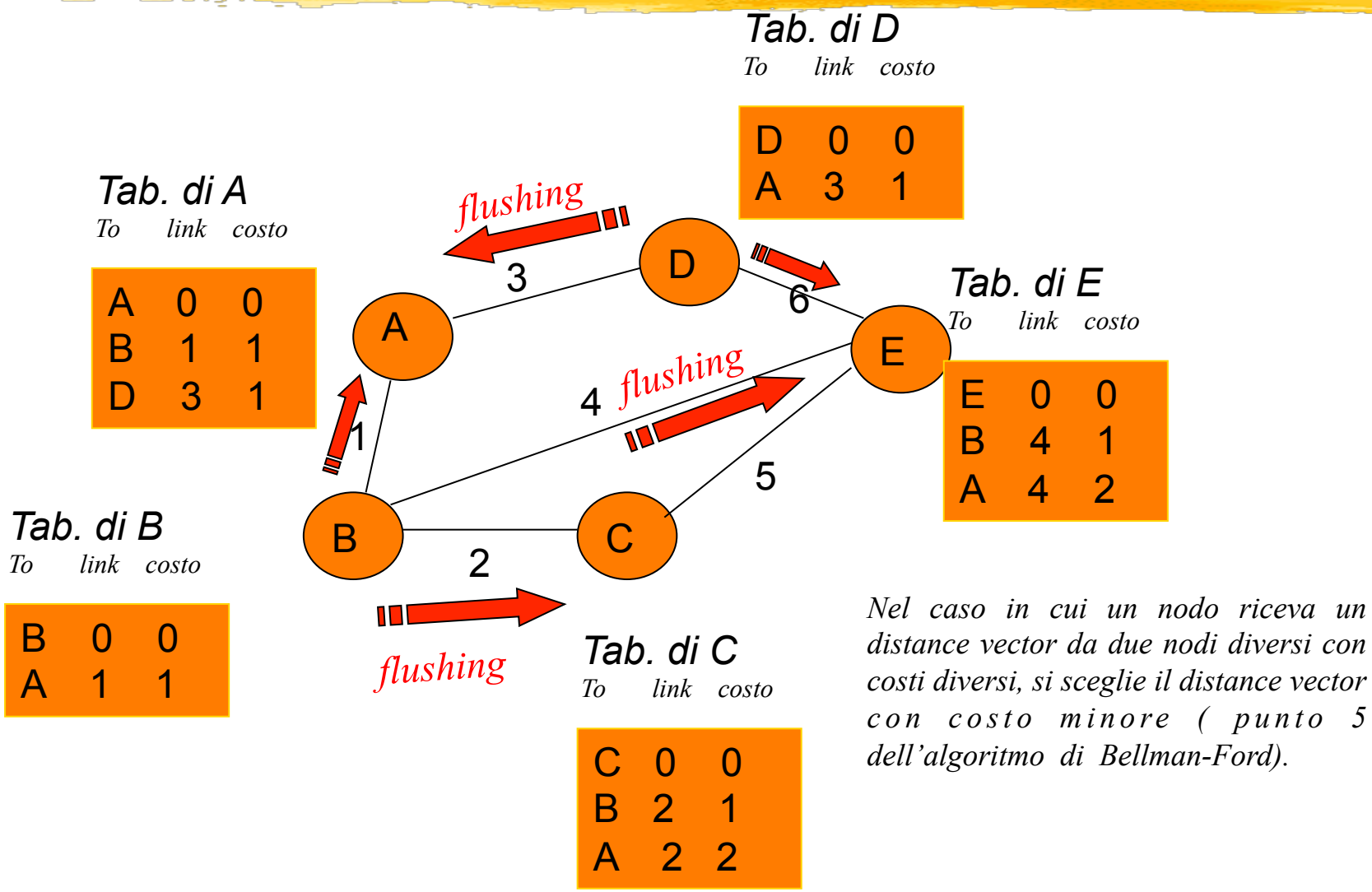
**Algoritmo :**

1) Se **i = j** allora tutti ***D[i,j] = 0***, altrimenti ***D[i,j]*** = ∞.
2) Inizializza tutti gli ***H[i,j] = -1***.
3) Per tutti i link ***l*** e per tutte le destinazioni ***k*** : ***i = L[l].s***; ***j = L[l].d;***
4) Calcola per tutti i link ***l*** ***Dist= L[l].m + D[j,k].***
5) Se **Dist < D[i,k]**, aggiorna ***D[i,k] = Dist*** ; ***H[i,k] = 1***
6) Se almeno un **D[i,k]** e' stato aggiornato, ripeti lo ***step 3***, altrimenti **stop**.

# Distance vector start up (4)

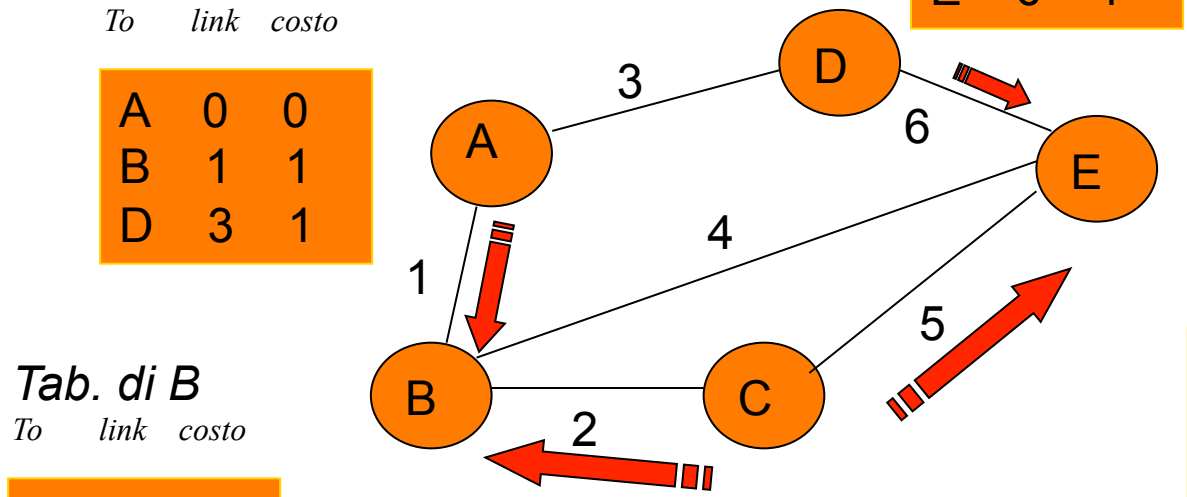


# D.V. Start up : Flushing 1




*Nel caso in cui un nodo riceva un distance vector da due nodi diversi con costi diversi, si sceglie il distance vector con costo minore ( punto 5 dell'algoritmo di Bellman-Ford).*

# D.V. Start up: Flushing 2




*Tab. di D*

| *To* | *link* | *costo* |
|---|---|---|
| D | 0 | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |

*Tab. di A*

| *To* | *link* | *costo* |
|---|---|---|
| A | 0 | 0 |
| B | 1 | 1 |
| D | 3 | 1 |

*Tab. di E*

| *To* | *link* | *costo* |
|---|---|---|
| E | 0 | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

*Tab. di B*

| *To* | *link* | *costo* |
|---|---|---|
| B | 0 | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

*Tab. di C*

| *To* | *link* | *costo* |
|---|---|---|
| C | 0 | 0 |
| B | 2 | 1 |
| A | 2 | 2 |

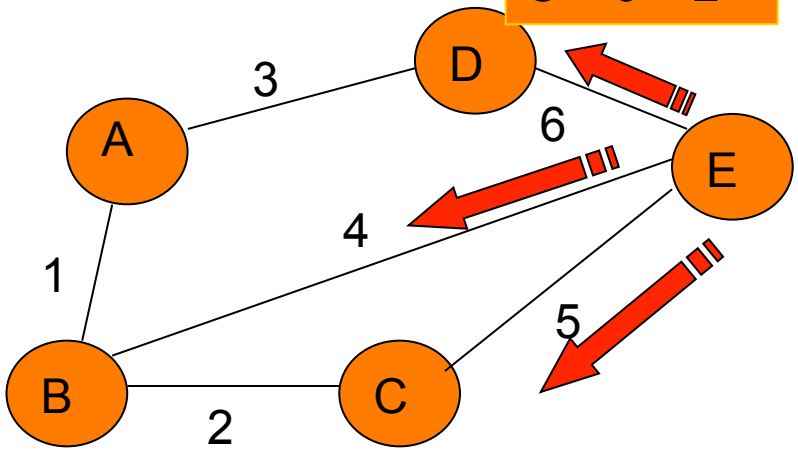
# D.V. Start up: Flushing 3

*Tab. di D*

| *To* | *link* | *costo* |
|---|---|---|
| D | 0 | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

*Tab. di A*

| *To* | *link* | *costo* |
|---|---|---|
| A | 0 | 0 |
| B | 1 | 1 |
| D | 3 | 1 |
| C | 1 | 2 |
| E | 1 | 2 |




*Tab. di E*

| *To* | *link* | *costo* |
|---|---|---|
| E | 0 | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

*Tab. di B*

| *To* | *link* | *costo* |
|---|---|---|
| B | 0 | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

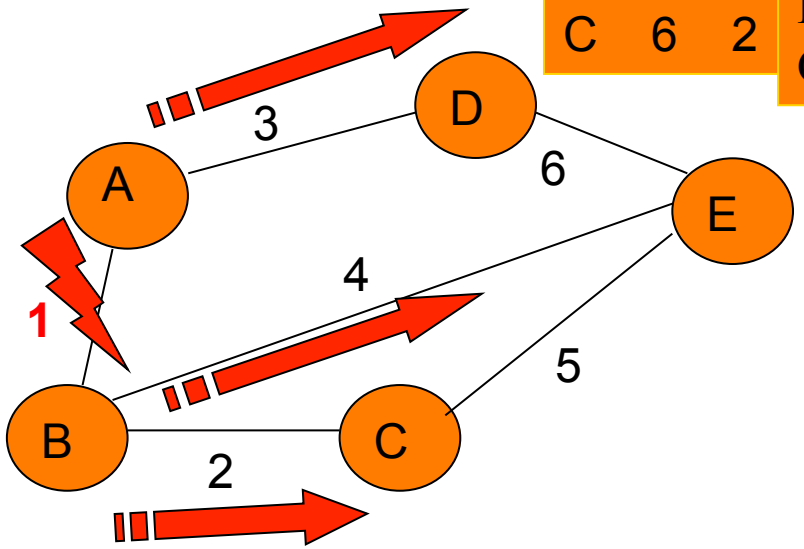
*Tab. di C*

| *To* | *link* | *costo* |
|---|---|---|
| C | 0 | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |

… a questo punto
l'algoritmo converge !!!

# ***Problema della instabilità :***

*ad es. crash di un ramo*
*(link 1 tra A e B)*




**Tab. di A**

| To | link | costo |
|---|---|---|
| A | 0 | 0 |
| B | 1 | **1** |
| D | 3 | 1 |
| C | 1 | **2** |
| E | 1 | **2** |

| | | |
|---|---|---|
| A | 0 | 0 |
| **B** | **1** | ∞ |
| D | 3 | 1 |
| **C** | **1** | ∞ |
| **E** | **1** | ∞ |

**Tab. di D**

| To | link | costo |
|---|---|---|
| D | 0 | 0 |
| A | 3 | 1 |
| B | 3 | 2 |
| E | 6 | 1 |
| C | 6 | 2 |

| | | |
|---|---|---|
| D | 0 | 0 |
| A | 3 | 1 |
| **B** | **3** | ∞ |
| E | 6 | 1 |
| C | 6 | 2 |

**Tab. di E**

| To | link | costo |
|---|---|---|
| E | 0 | 0 |
| B | 4 | 1 |
| A | 4 | 2 |
| D | 6 | 1 |
| C | 5 | 1 |

| | | |
|---|---|---|
| E | 0 | 0 |
| B | 4 | 1 |
| **A** | **4** | ∞ |
| D | 6 | 1 |
| C | 5 | 1 |

**Tab. di B**

| To | link | costo |
|---|---|---|
| B | 0 | 0 |
| A | 1 | 1 |
| D | 1 | 2 |
| C | 2 | 1 |
| E | 4 | 1 |

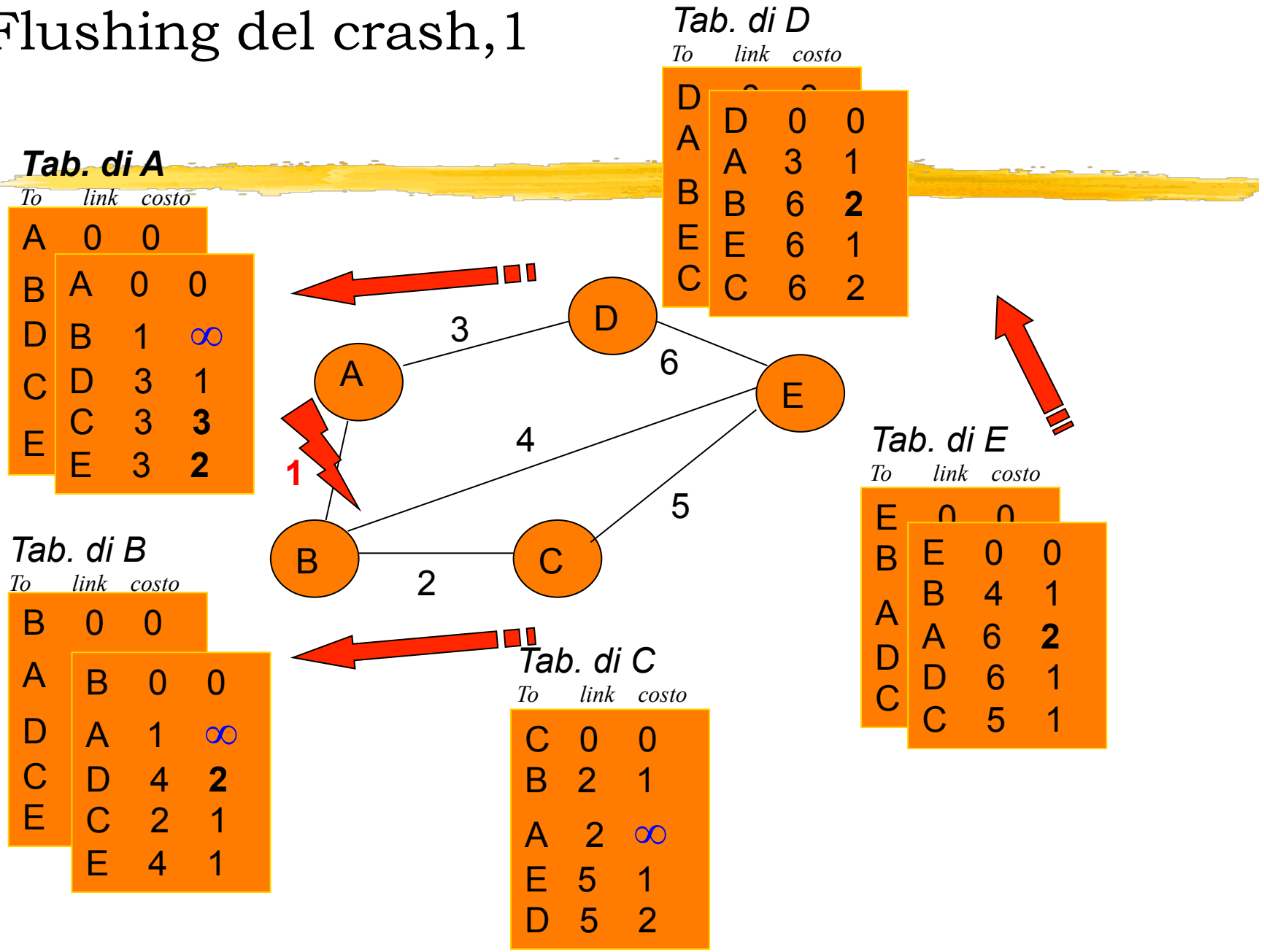
| | | |
|---|---|---|
| B | 0 | 0 |
| **A** | **1** | ∞ |
| **D** | **1** | ∞ |
| C | 2 | 1 |
| E | 4 | 1 |

**Tab. di C**

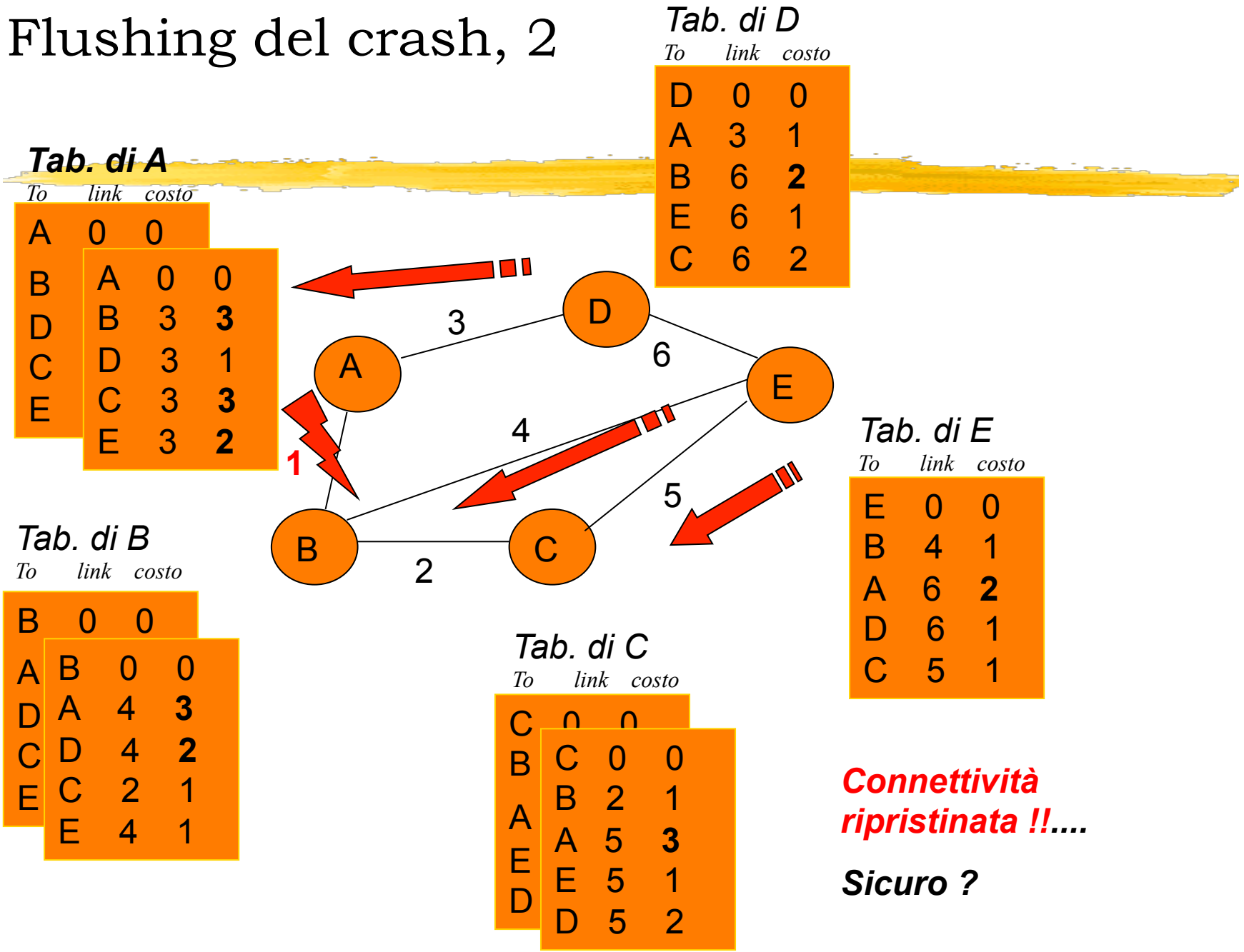
| To | link | costo |
|---|---|---|
| C | 0 | 0 |
| B | 2 | 1 |
| A | 2 | 2 |
| E | 5 | 1 |
| D | 5 | 2 |

| | | |
|---|---|---|
| C | 0 | 0 |
| B | 2 | 1 |
| **A** | **2** | ∞ |
| E | 5 | 1 |
| D | 5 | 2 |

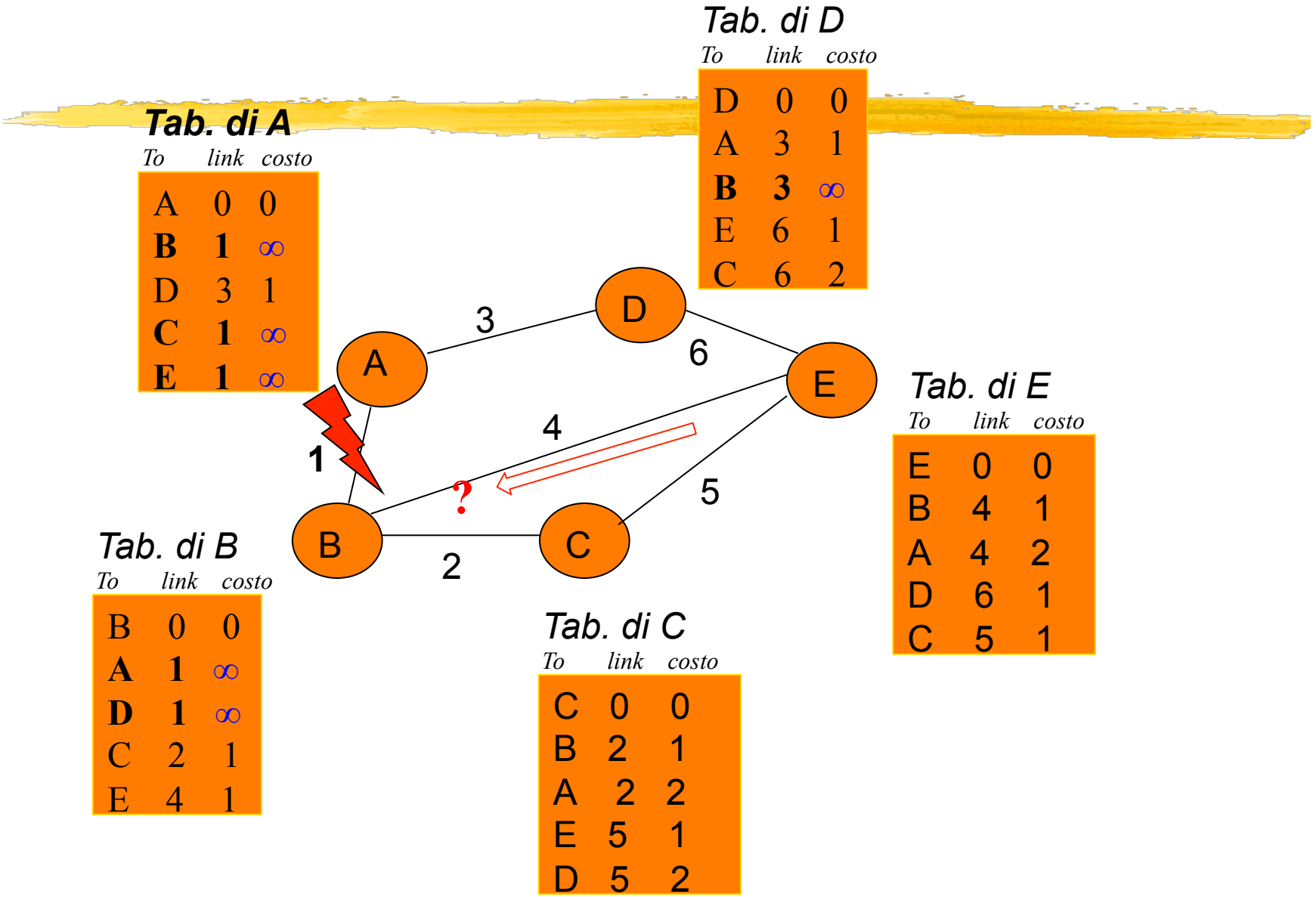
# Flushing del crash, 1



# Flushing del crash, 2



# COUNT-TO-INFINITY PROBLEM



# *Count-to-infinity problem*

*L'algoritmo di* ***Bellman-Ford*** ***NON risolve il problema della possibilità di routing loop*** *(**"count-to-infinity problem**")*

***Supponiamo di avere la rete A-B-C-D-E-F del lucido precedente, con la metrica degli hop:***

**- Se A (o il link 1) è fuori servizio, B non riceve il vettore delle distanze da A, ma riceve da E** (*che non sa ancora che A è fuori servizio*) **l'informazione che A è raggiungibile con due hop (E-B, B-A) , ma questo è FALSO.**

***Questo tipo di propagazione del vettore rallenta il processo fino al raggiungimento del valore massimo consentito del numero di hop (infinito).***
***Il tempo di convergenza aumenta così fino alla possibilità di formare un bouncing effect o un loop.***

# *Split horizon* e *poison reverse* (6)

***Per cercare di risolvere i problemi di loop sono stati introdotti nell'algoritmo base di distance vector alcune soluzioni:***

-***route poisoning***: In presenza di un **count to infinity**, si è notato che il costo verso una certa destinazione cresce progressivamente. L'idea è quella di bloccare (**ponendo a $\infty$ il costo del link**) l'utilizzo di tutte le route che aumentano in modo progressivo di costo. La route verrà rimessa in servizio solo quando due annunci consecutivi confermeranno la presenza della route.

-***split horizon*: principio secondo il quale un nodo non comunica ad un nodo vicino percorsi che ha appreso proprio da quel nodo.**

**- *split horizon with poison reverse*, principio secondo il quale le destinazioni raggiungibili passando per il nodo stesso da cui si è ricevuto il distance vector vengono messe a costo infinito.**

# RIP (Routing Information Protocol) *(1)*

E’ un protocollo relativamente semplice appartenente alla famiglia di protocolli di tipo **"distance vector".**

E’ il **protocollo di routing IGP** più vecchio ancora in uso : esistono due versioni, la seconda versione aggiunge nuove funzionalità a questo protocollo. **RIPv1** è di tipo **classfull** (RFC 1058) mentre ***RIPv2*** **è** ***classless*** *(*RFC 2453) .

- *I processi RIP utilizzano la porta* ***520*** *sia per la trasmissione che per la ricezione.*
- *Gli indirizzi presenti nelle tabelle RIP sono* ***indirizzi Internet a 32 bit****.*
- *Una voce (entry) nella tabella di routing puo' rappresentare un host, una rete o una sottorete. Non sono presenti specifiche sul tipo di indirizzo nei pacchetti RIP; e' compito dei router analizzare l'indirizzo per capire di cosa si tratta.*
- Di default, RIP utilizza la **metrica «hop count»** : la distanza e' il numero di link che vengono attraversati per raggiungere la destinazione. Questa distanza e' espressa come **un numero intero variabile tra 1 e 15**.
  **Il valore *16 rappresenta una distanza infinita.***

# RIP (Routing Information Protocol) *(2)*

- **Normalmente i pacchetti vengono inviati in** modalita' **broadcast ogni 30 secondi**, o meno nel caso di aggiornamenti alle tabelle.
- **Se una route non viene aggiornata entro 3 minuti, la distanza viene fissata ad infinito (16) e l'entry verra' successivamente rimossa dalle tabelle.**

- Il processo RIP, in seguito alla ricezione di un messaggio di risposta, aggiorna la propria tabella. Ogni voce della tabella sara' al limite composta da :
  a) Indirizzo di destinazione  b) Metrica associata con la destinazione
  c) Indirizzo del "next router"  d) Un "recently updated" flag
  e) Numerosi timers :
  - ✓ *Routing update timer (default 30 s): intervallo di tempo per l'invio degli annunci*
  - ✓ *Route invalid timer (default 90 s): intervallo di tempo dopo il quale una route è dichiarata irraggiungibile (distanza posta ad infinito)*
  - ✓ *Route flush timer (default 270 s): intervallo di tempo dopo il quale la route è cancellata dalla routing table*
  - ✓ *Triggered updates: sono inviate con un ritardo casuale compreso tra 1 e 5 secondi.*

# RIP (Routing Information Protocol) *(3)*

- Processando le risposte in arrivo, il router **esaminerà le voci una ad una** ed eseguirà una serie di ***check***, ad esempio verificherà che l'indirizzo sia valido ed appartenente ad una delle classi A, B o C, e che la metrica *(TTL)* non sia maggiore di 15 *(infinito)*.

- **Se la metrica in arrivo risulta minore di infinito (16)**, viene incrementata di **1** per il successivo hop, quindi la tabella di routing viene scandita per una voce corrispondente alla destinazione e viene eseguito il generico *processo* "***distance vector***":

*a)* ***Se la voce non e' presente*** *e la sua metrica nel messaggio ricevuto* ***non e' infinito****, la aggiunge alla tabella, inizializzando la metrica al valore ricevuto ed il next router al mittente del messaggio..*

*b)* ***Se la voce è presente*** *con una metrica più grande, aggiorna i campi della metrica e del next router.*

*c)* ***Se la voce è presente*** *ed il next router è il mittente del messaggio di risposta, aggiorna la metrica se questa differisce dal valore memorizzato.*

*d)* ***In tutti gli altri casi****, il messaggio ricevuto è ignorato.*

- **Se la metrica o il next router cambiano**, l'entry viene marcata come "aggiornata". Un messaggio di risposta viene inviato ad intervalli regolari di 30 secondi o puo' essere attivato in seguito ad un aggiornamento alle tabelle di routing.

# IGRP (Interior Gateway Router Protocol) *(1)*

**Protocollo di routing DV sviluppato dalla CISCO a metà '80.**
**Agli inizi degli anni '90, e' stata sviluppata la versione "Enhanced " (EIGRP).**

**IGRP** è caratterizzato da:

- ***Metriche multiple piu' sofisticate***
- ***Supporto del multipath routing***
- ***Migliore stabilita'***_

- ✓ IGRP utilizza una frequenza di update (**90 s**) delle tabelle di routing.
- ✓ Il protocollo IGRP permette il routing **all'interno dell'Autonomous System** (l'AS è identificato da un intero su 16 bit).
- ✓ IGRP permette la **gestione di più entry nella routing table per la stessa destinazione.** Il carico può essere ripartito tra le diverse route in funzione della metrica associata.
- ✓ **Hop count** massimo pari a **255**

# Metriche di IGRP *(2)*

**Nel** IGRP le **metriche** si basano su 4 parametri:

**B - Bandwidth** (1 - 224) (1 = 1.2 kbit/s) Il valore effettivo corrisponde al "numero di slot temporali da 10ms necessari per trasmettere 10000 bit"

**D - Delay** (1 - 224) (1 = 10 ms)

**R - Reliability** (1 - 255) (255 = 100%)

**L - Load** (1 - 255) (255 = 100%)

La variazione dei coefficienti permette di privilegiare determinati parametri a scapito di altri (es. **banda** piuttosto che **ritardo**)

Bandwidth e Delay sono per default associati al tipo di portante fisica.

*Ad esempio:*

*Ethernet -> B=10000, D=100* *CDN 64 kbit/s -> B=64, D=2000*

*Per ciascun link, B e D possono comunque essere impostati dal gestore.*

I parametri Reliability e Load possono essere difficili da quantificare e possono variare con una frequenza piuttosto elevata.

# *Link State Routing (1)*

**Algoritmo basato sullo stato dei collegamenti tra i nodi.**

- Problema della costruzione della mappa della rete
- Problema del calcolo del percorso migliore

Per permettere la **costruzione della mappa della rete**, a ciascun router vengono inviati dei pacchetti, detti **LSP** *(Link State Packet)* , che contengono lo stato di ogni link, l'identità di ogni vicino ed i costi dei link connessi al nodo che lo invia.

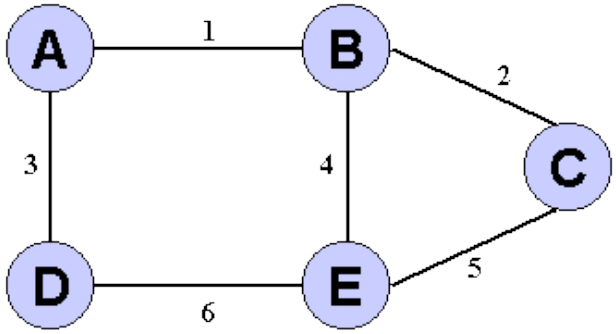
Per il **calcolo del percorso migliore**, ogni nodo della rete esegue il calcolo del percorso più breve tramite l'algoritmo di ***Dijkstra***

# *Link State Routing* (2)

***Instradamento di tipo Link State:***

*invece di calcolare i percorsi migliori in modo distribuito,* ***tutti i nodi mantengono una copia intera della mappa della rete ed eseguono un calcolo completo*** *(algoritmo di Dijkstra)* ***dei migliori percorsi da questa mappa locale.***

*La mappa e' contenuta in un database del router, dove ciascun record rappresenta un link nella rete.*




| *Da* | *A* | *Link* | *Distanza* |
|---|---|---|---|
| A | B | 1 | 1 |
| A | D | 3 | 1 |
| B | A | 1 | 1 |
| B | C | 2 | 1 |
| B | E | 4 | 1 |
| C | B | 2 | 1 |
| C | E | 5 | 1 |
| D | A | 3 | 1 |
| D | E | 6 | 1 |
| E | B | 4 | 1 |
| E | C | 5 | 1 |
| E | D | 6 | 1 |

# Link State Packet *(3)*

*I pacchetti inviati in **flooding** **su tutti i link di un router** per la costruzione della mappa della rete, sono detti **Link State Packet (LSP)** e contengono:*

1. Stato di ogni link connesso al router,
2. Identità di ogni vicino connesso all'altro estremo del link (sulle LAN possono esserci migliaia di vicini),
3. Costo del link,
4. Numero di sequenza per il LSP (a seguito di frequenti variazioni di topologia un router può ricevere un LSP vecchio dopo uno nuovo, quindi deve essere in grado di valutare il più recente),
5. Checksum,
6. Lifetime (la validità di ogni LSP è limitata nel tempo, diversamente un errore sul numero di sequenza potrebbe rendere un LSP valido per anni).

*Un **LSP** viene **generato periodicamente**, oppure **quando viene rilevata una variazione** nella topologia locale (adiacenze), ossia quando:*

*a) viene riconosciuto un nuovo vicino,*

*b) il costo verso un vicino e' cambiato,*

*c) si e' persa la connettivita' verso un vicino precedentemente raggiungibile.*

# Link State Packet *(4)*

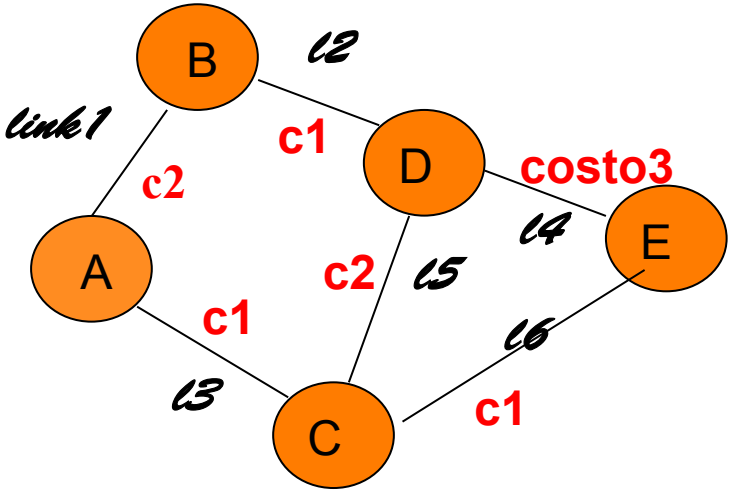
All'atto del ricevimento di un ***LSP***, il router compie le seguenti azioni:

***a) Se non ha mai ricevuto LSP da quel router o se il LSP è più recente di quello precedentemente memorizzato*** *(campo Sequence Number)****, memorizza il pacchetto e lo ritrasmette in flooding su tutte le linee eccetto quella da cui l'ha ricevuto;***

***b) Se il LSP ha lo stesso numero di sequenza di quello posseduto non fa nulla;***

***c) Se il LSP è più vecchio di quello posseduto trasmette al mittente il pacchetto più recente.***

# Tabella del link state *(5)*




| *from* | *to* | *Link* | *costo* |
|---|---|---|---|
| A | B | 1 | 2 |
| A | C | 3 | 1 |
| B | A | 1 | 2 |
| B | D | 2 | 1 |
| C | A | 3 | 1 |
| C | E | 6 | 1 |
| C | D | 5 | 2 |
| D | B | 2 | 1 |
| D | E | 4 | 3 |
| D | C | 5 | 2 |
| E | C | 6 | 1 |
| E | D | 4 | 3 |

# *L'algoritmo di DIJKSTRA* (6)

*L'algoritmo "**Shortest Path First**" **(SPF)** di **E.W. Dijkstra** calcola sulla mappa di rete **il percorso piu' breve tra un nodo sorgente ed un altro nodo della rete.***

*Si definiscono:*

**a) 1 nodo radice (root), ossia il nodo che sta calcolando l'algoritmo,**

**b) 1 insieme PATH di nodi (ID, cost, link) per i quali si è già trovato il percorso migliore,**

**c) 1 insieme TEMP di nodi (ID, cost, link) per i quali si sta cercando un percorso.**

# *L'Algoritmo di Dijkstra (7)*

1. *Si inserisce il nodo root (sorgente) in PATH ;*
2. *Si inseriscono tutti i nodi vicini del root in TEMP ;*
3. *Si prende il nodo N con il percorso più piccolo in TEMP e lo si promuove in PATH ;*
4. *Per ogni vicino V del nodo N promosso in PATH :*

   *Se V non esiste ancora in TEMP lo si inserisce ora con il costo Dist verso la root ( Dist( root, N ) + Dist( N, V))*

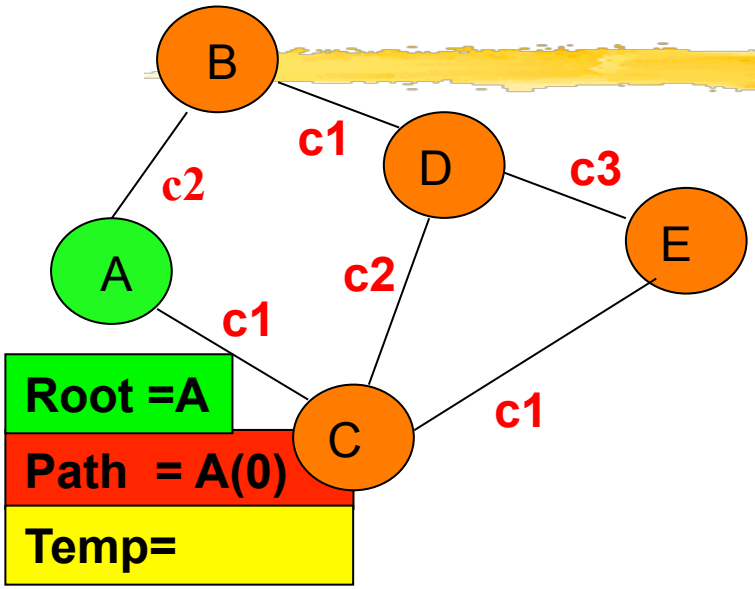
   *Se V già esiste, se ne analizza il costo Dist e se questo è minore del precedente riportato in TEMP si aggiorna cost e link di quel nodo in TEMP.*
5. *Salta al passo 3 se sono ancora presenti nodi in TEMP, altrimenti STOP*

**Principi di funzionamento dell'algoritmo**

- **Ad ogni nodo è associata una etichetta che rappresenta il costo del cammino migliore trovato per raggiungerlo.**
- L'algoritmo etichetta progressivamente i nodi partendo da ciascun nodo ;
- Il prossimo nodo etichettato è quello raggiungibile a costo più basso a partire da un nodo già etichettato.
- **L'algoritmo termina quando a tutti i nodi è stata associata un'etichetta.**

## *L'Algoritmo di Dijkstra* (8)

*1) Si inserisce il nodo root in PATH*

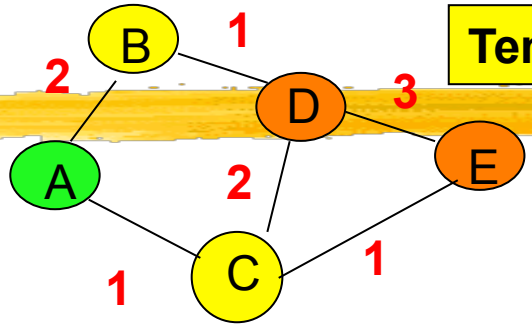



**Root =A**

**Path = A(0)**

**Temp=**

*2) Si inseriscono tutti i nodi vicini del precedente in TEMP*



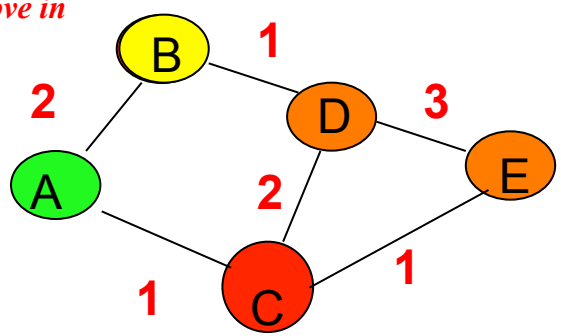

**Path = A(0)**

**Temp= B(2), C(1)**

*3) Si prende il nodo C con il percorso più piccolo in TEMP e lo si promuove in PATH*

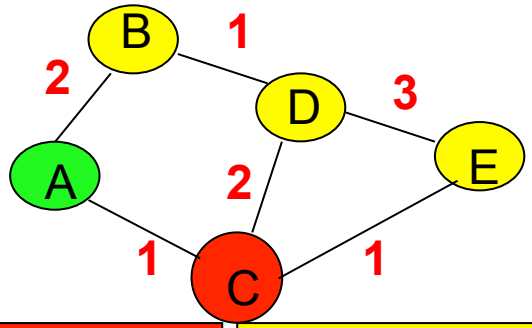
**Path =A(0), C(1)**

**Temp= B(2)**




*4) Per ogni vicino V del nodo C promosso:….*

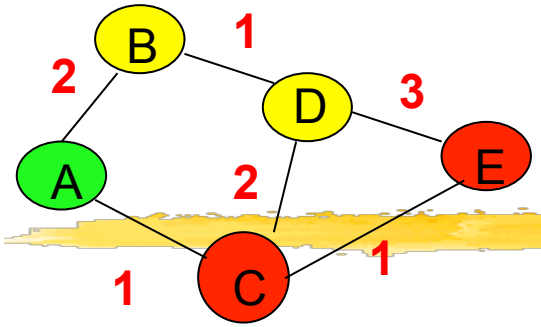
*4.1) Se V non esiste ancora in TEMP lo si inserisce ora con il costo verso la root D=(dist (root, C ) + dist (C, D)) = 1+2 =3 e E= (dist (root, C) + dist (C, E)) = 1+1 =2*




**Path = A(0), C(1)**

**Temp= B(2), D(1+2), E(1+1)**

*5) Si ripete il passo 3*



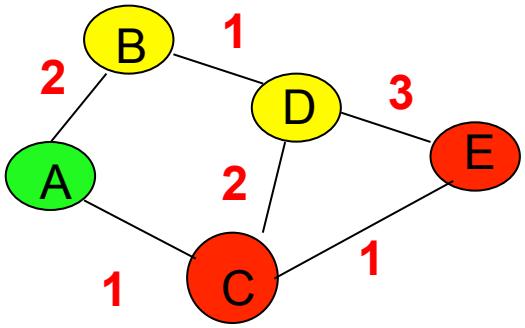

*3) Si prende il nodo E con il percorso più piccolo in TEMP e lo si promuove in PATH*

**Path = A(0), C(1), E(2)**

**Temp= B(2), D(3)**

*4) Per ogni vicino V del nodo E promosso:….*

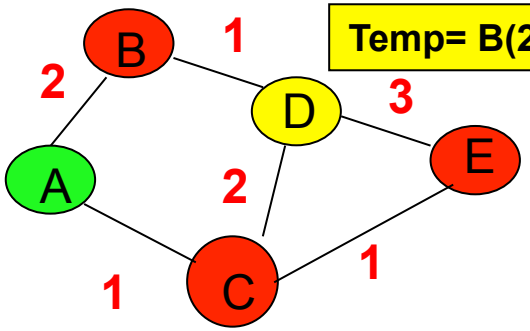
*4.1) Se D non esiste ancora in TEMP lo si inserisce ora*




*4.2) Se D già esiste se ne analizza il costo verso la root A (dist (root, E) + dist (E, D)) = 2+3 =5 e se questo è minore del precedente riportato in TEMP si aggiorna il costo del nodo D in TEMP (… e non è il caso di D).*

**Path =A(0), C(1), E(2), D(3)**

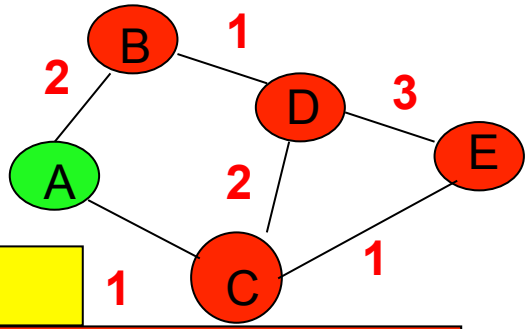
**Temp= B(2)**




**Path =A(0), C(1), E(2)**

**Temp= B(2), D(3)**

*Si procede fino ad inserire in PATH tutti i nodi*




**Temp=**

**Path =A(0), C(1), E(2), D(3), B(2)**

# OSPF (Open Shortest Path First) *(10)*

**Protocollo Interior Gateway Protocol sviluppato dalla IETF** (Internet Engineering Task Force) **in sostituzione di RIP (**RFC 2328**).**

- **Protocollo basato sullo stato dei LINK in Autonomous System di grosse dimensioni.**
- **LSP comunicato a tutti i router interni in *flooding.*** Ciascun router si costruisce una mappa topologica completa dell'intero AS **.**
- **Ciascun router esegue localmente *l'algoritmo di Dijkstra* "Shortest Path first" per determinare il percorso più breve verso le altre reti.**

✓ *Un AS di grosse dimensioni viene suddiviso in **aree**, le quali contengono un gruppo di reti contigue.* OSPF prevede di utilizzare il "**routing gerarchico**" ossia la suddivisione della rete in ***aree*** connesse attraverso un "**backbone**".

# OSPF *(11)*

- ***OSPF** e' stato progettato per **separare host e router***

*Gli host IP sono connessi a reti locali e, applicando strettamente il modello Link State, si dovrebbe descrivere la relazione tra ciascun host ed il router attraverso un **record di link state** .*

**OSPF** invece **permette una semplificazione basata sul concetto di "subnet":** poiche' tutti gli host della rete appartengono ad una singola **sottorete** IP, e' sufficiente **inviare un messaggio** sul **link tra il router e la sottorete**.
*Nella terminologia OSPF, questa e' una variante di "router link", chiamata **"link to a stub network".***
*Il link verso un vicino e' identificato dall'indirizzo IP del vicino stesso, mentre quello verso una "stub network" e' identificato dal suo numero di rete o sottorete.*

# *Border Gateway Protocol = BGP*

**E' un protocollo di routing EGP a indicazione di percorso (*path vector*), che prende le decisioni di instradamento basandosi su politiche (regole) di instradamento determinate da ciascuna rete (**RFC 4271) piuttosto che su determinate metriche.

***Rappresenta l'attuale standard de facto dei protocolli di routing tra AS (peer)* .**

***BGP*** *prevede che ciascun **S**istema **A**utonomo possa configurare manualmente i propri router, stabilendo una sessione TCP con i router adiacenti al fine di:*
***a.*** *ottenere informazioni sulla raggiungibilità delle sottoreti da parte dei sistemi confinanti;*
***b.*** *propagare le informazioni di raggiungibilità a tutti i router interni ad un sistema autonomo;*
***c.*** *determinare percorsi "buoni" verso le sottoreti sulla base delle informazioni di raggiungibilità e delle politiche del Sistema Autonomo.*