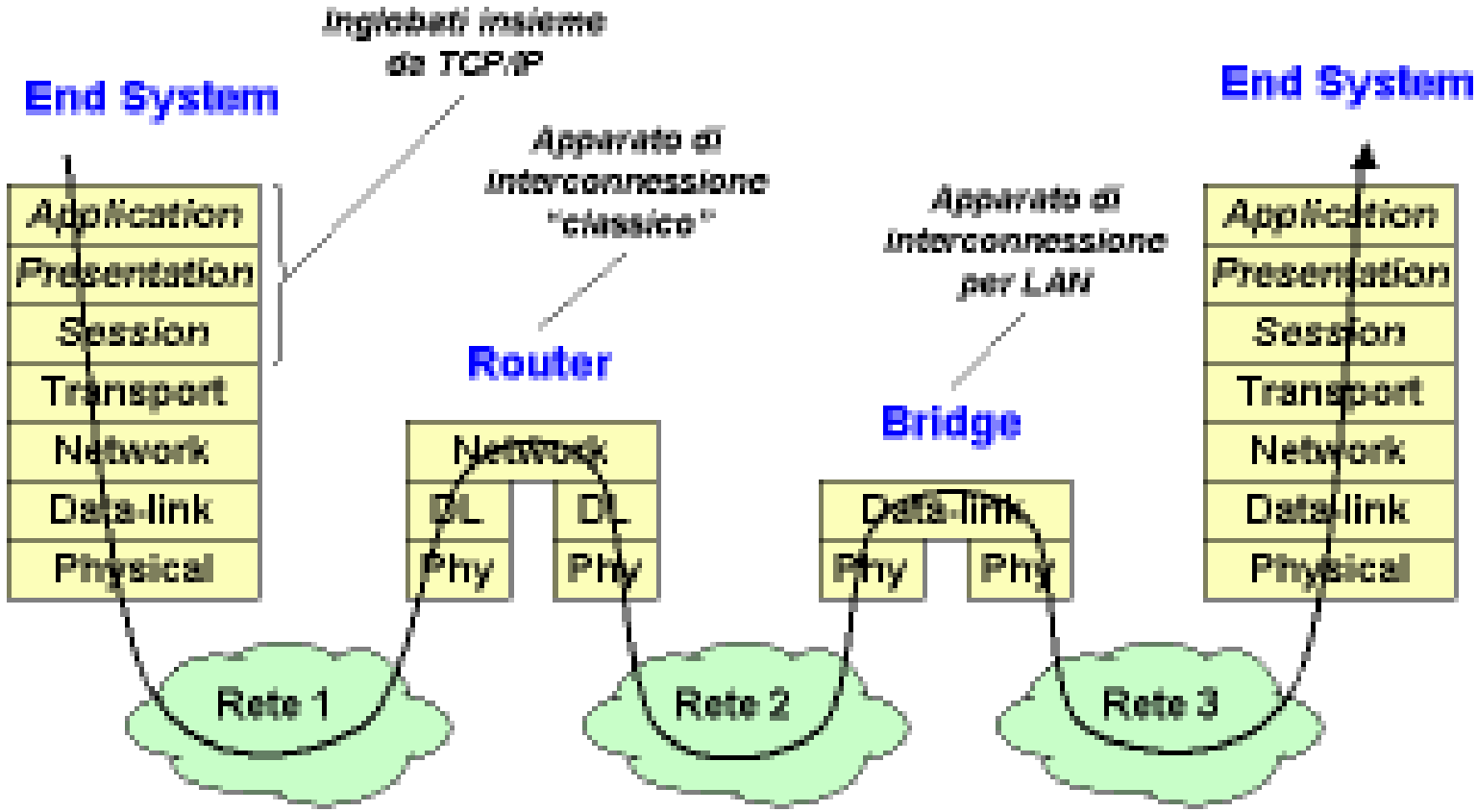
Reti: modello LAN- WAN

Approfondimento: algoritmi di forwarding e routing

Terminologia

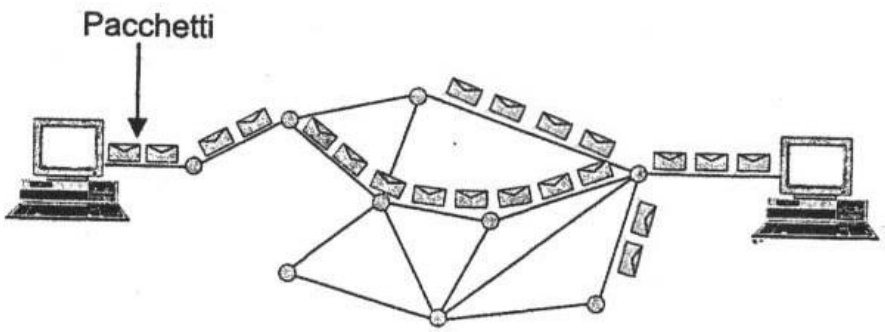
- System (node, host)
  - Dispositivo contenente al suo interno almeno i livelli fisico, data link e network
- End System (ES, end node, DTE)
  - Nodo "edge", che agisce come mittente e destinatario finale dei dati
- Intermediate System (IS, router)
  - Nodo "core", che fornisce il transito ai pacchetti tra la sorgente e la destinazione
  - Solitamente non genera dati e non è il destinatario finale dei dati (tranne nel caso di traffico di management)
- Solitamente ES implementano tutti e sette i livelli OSI, mentre gli IS sono limitati ai primi tre
  - L'implementazione dei livelli superiori negli IS è fatta solamente per esigenze di gestione

Terminologia e reti reali



Il livello Network

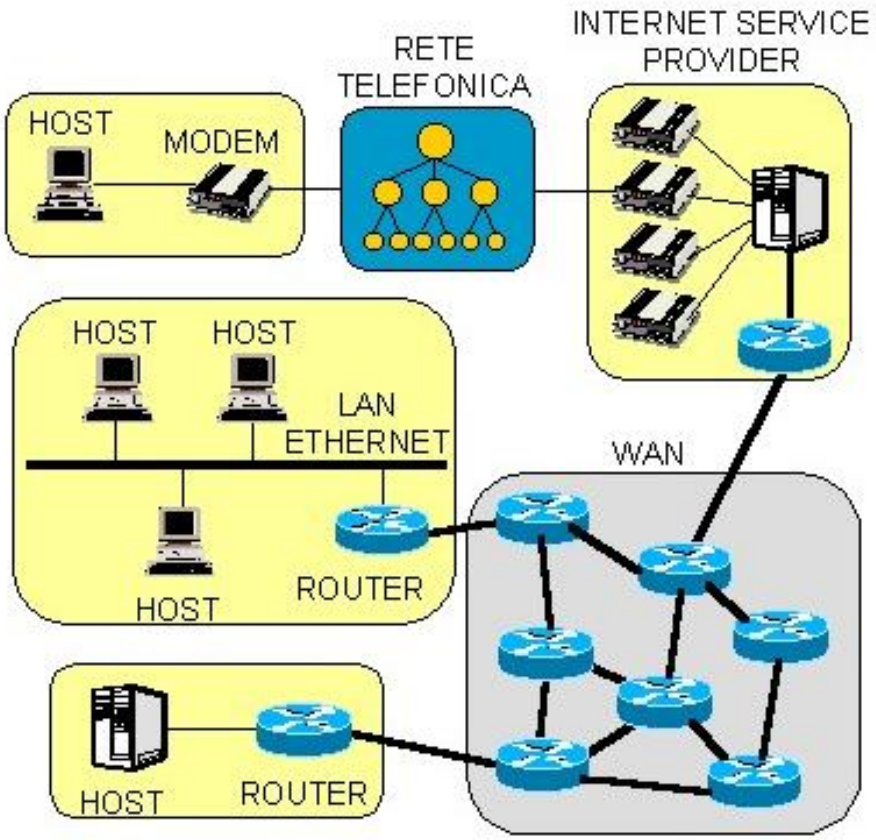
- Tratta le problematiche di Internetworking
  - Instrada i messaggi sulla rete
  - Provvedere a trovare instradamenti alternativi in caso di guasti
  - Può fornire sia servizi non connessi sia servizi connessi



Per **instradamento** intendiamo la determinazione del percorso dei messaggi nella rete, scelto tra i possibili percorsi che collegano la sorgente con la destinazione.
Con una metafora: il percorso ottimale dell'auto dallo start allo stop

Livello 3

La funzione di **instradamento** è basata sugli indirizzi di livello 3 del modello ISO/ OSI (rete)

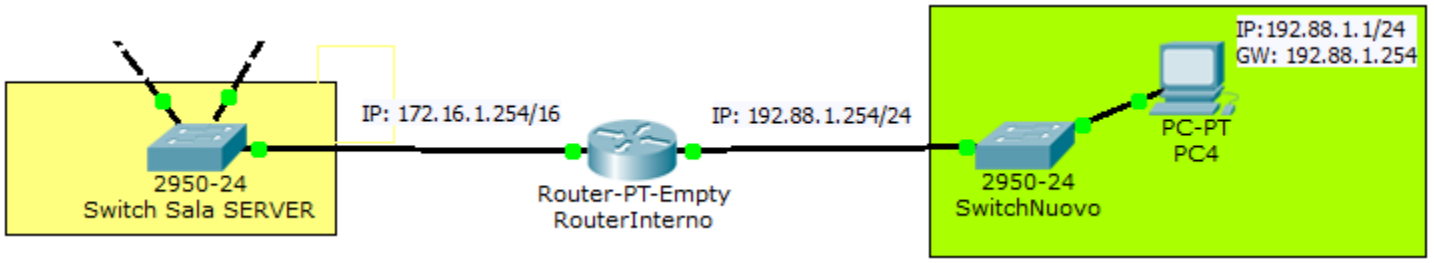


Gli elementi della tabella di instradamento non sono singoli calcolatori ma *reti locali*. Questo permette di interconnettere grandi reti senza crescite incontrollabili della **tabella di instradamento**.

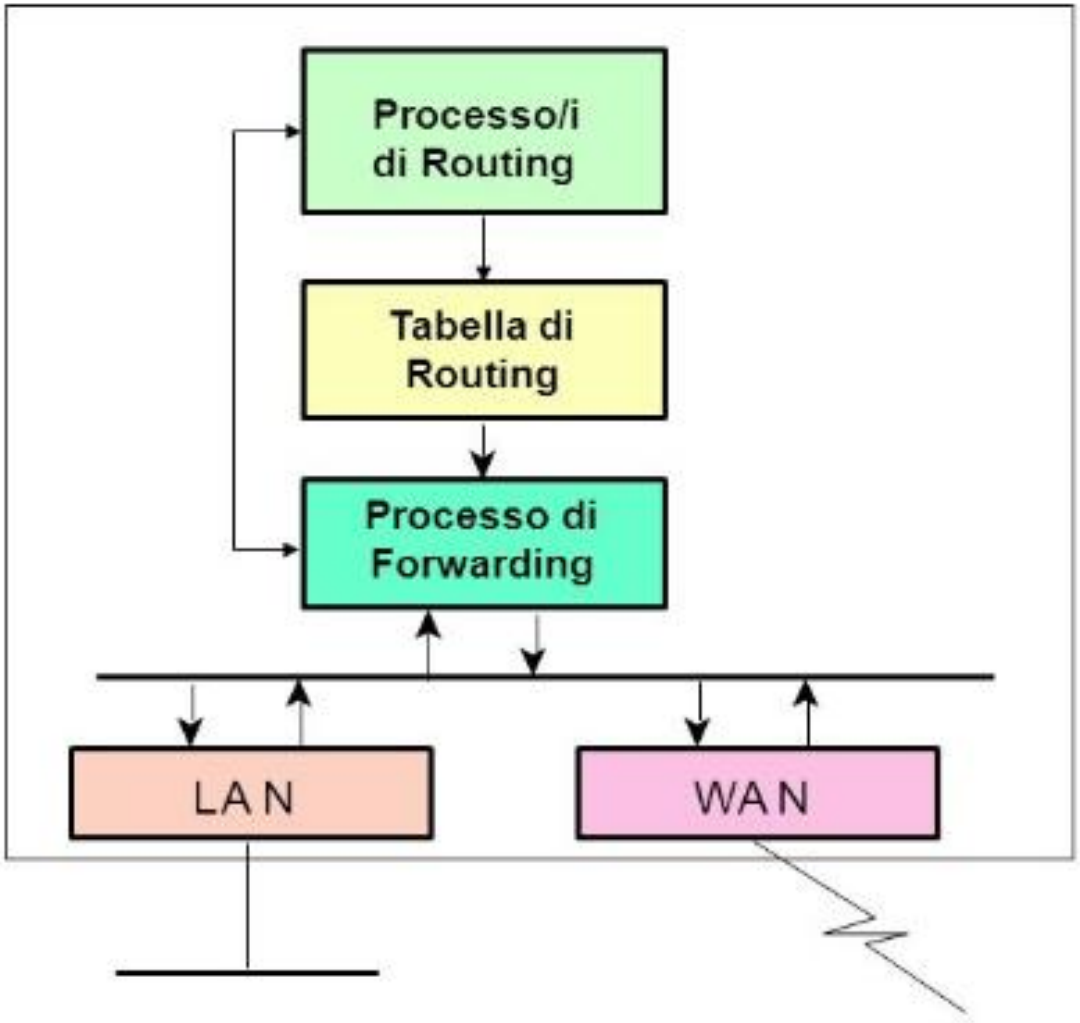
Livello 3: Routers

Il compito del router è costituito da 2 attività principali :

1. Determinazione del percorso ottimale
2. Trasporto delle informazioni tra 2 reti diverse



Architettura di un router



Processi di Routing e Forwarding

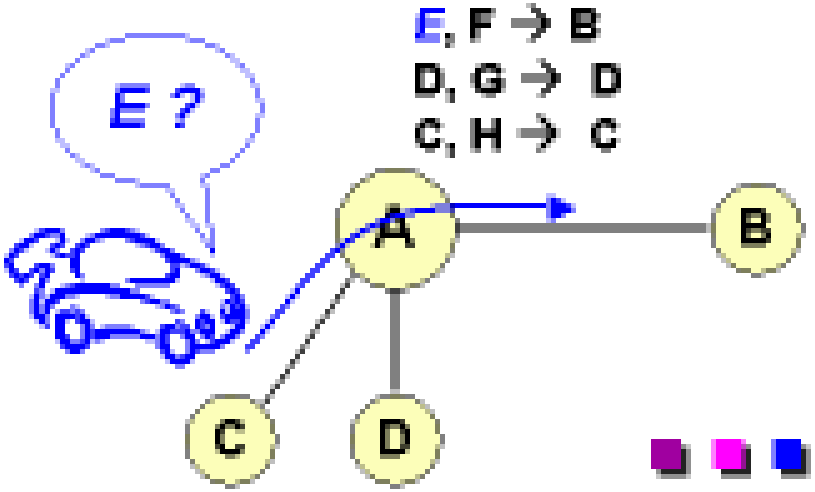
- **Routing**
  - Determinazione delle destinazioni raggiungibili da ogni nodo
  - Calcolo del percorso migliore
  - Inserimento di informazioni locali in ogni nodo
- **Forwarding**
  - Inoltro dei pacchetti nodo per nodo utilizzando le informazioni locali
  - Avviene indipendentemente su ogni nodo
  - Non ha conoscenza globale della rete
- **Forwarding e routing**
  - Ambedue necessari per l'operatività di una rete
  - Routing: può essere "manuale" (demandato all'amministratore)

Algoritmi di Forwarding

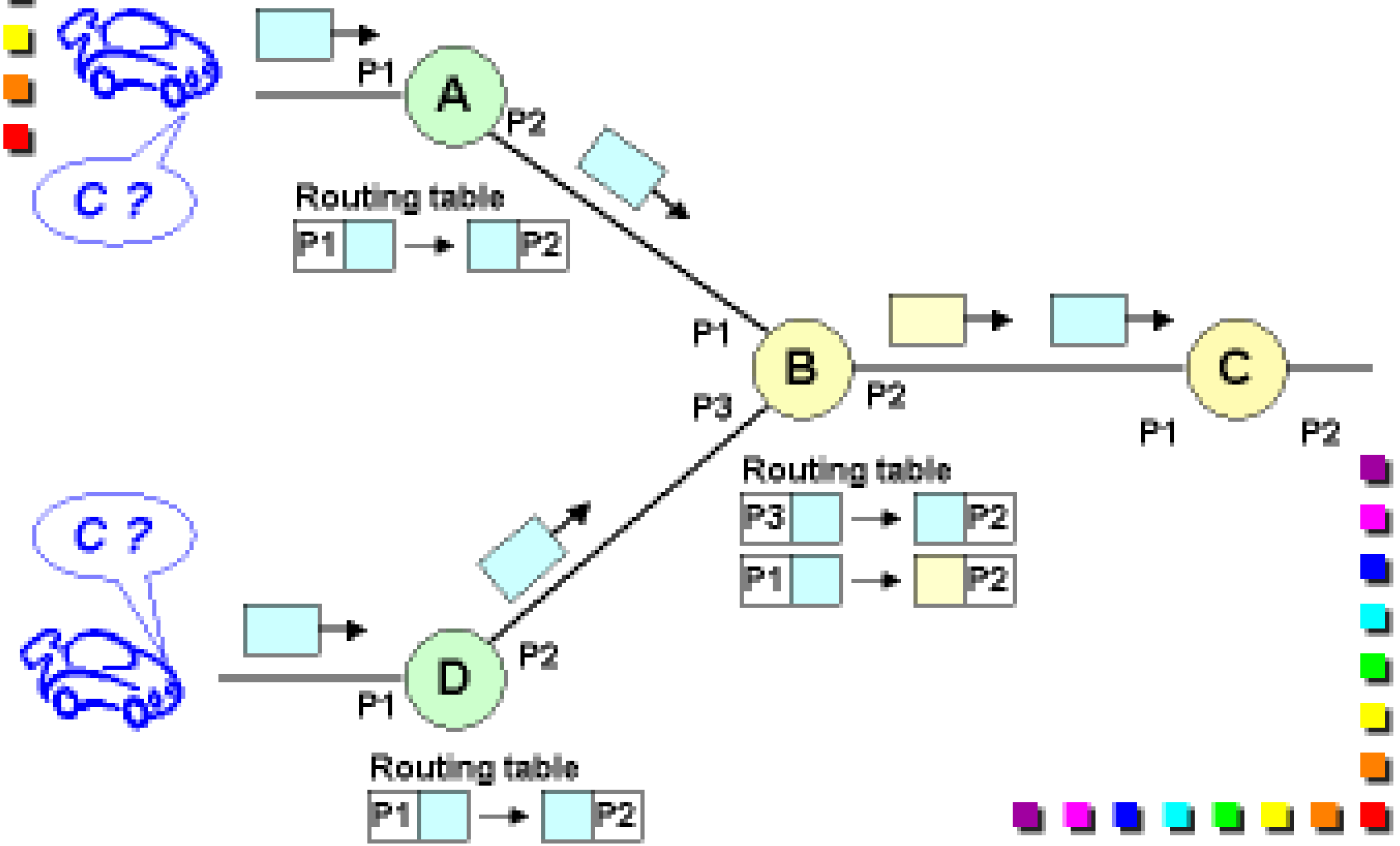
- Tre algoritmi principali
  - **Routing by Network Address**
  - **Label Swapping**
  - **Source Routing**
- Routing e forwarding
  - **Storicamente la differenza tra i due concetti era abbastanza labile, da cui la confusione nei nome di questi algoritmi**
- Reti reali
  - **Ogni tecnologia sceglie un algoritmo e lo adotta come preferenziale**

Routing by network address

- Usato tipicamente nei protocolli non connessi
  - **Bridge trasparenti, Decnet, IPX, IPv4, IPv6, CLNP**
- Ogni pacchetto contiene l'indirizzo del nodo destinatario
- Indirizzo di destinazione
  - **Viene usato come chiave di accesso alle tabelle di routing**
  - **Destinatari sulla stessa rete**
  - **Destinatario su una rete diversa**



Label Swapping (1)



Label Swapping (2)

- Etichetta
  - Identificativo molto piccolo, con valore locale
  - Abbinato alla porta di ingresso (oppure destinazione)
- Label swapping
  - Ogni nodo commuta in base alla porta da cui arriva il pacchetto e in base all'etichetta posseduta
  - Dopo la commutazione viene assegnato al pacchetto una nuova etichetta da utilizzarsi nel nodo successivo
- Percorso
  - Univocamente determinato dall'insieme (*port_in, label_in* → *port_out, label_out*) sui nodi dalla sorgente alla destinazione
  - Mittente: non indicato nei pacchetti dati
    - È univocamente determinato dal percorso

Assegnazione globale e locale

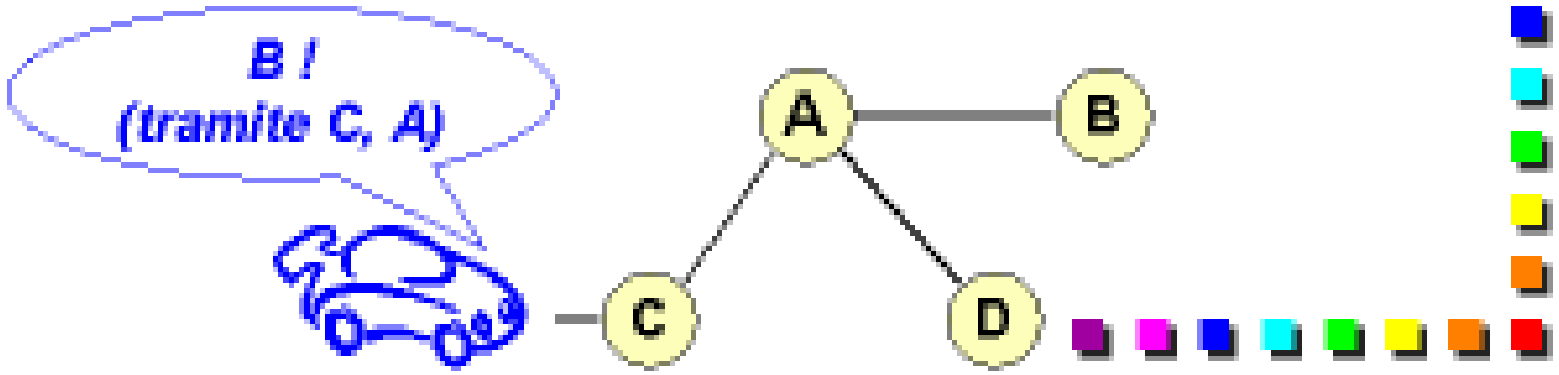
- Locale: management semplificato
  - **Assegnazione possibile da parte del nodo stesso, senza interazioni con altri**
    - Richiede la memorizzazione della porta di arrivo / inoltro
  - **Problematica su mezzi ad accesso condiviso**
    - Impossibile procedere ad assegnazione puramente locale al nodo
- Globale
  - **On demand: necessita di una procedura per determinare una etichetta libera su tutto il percorso**
  - **Offline: deve essere configurata a priori**
  - **Necessita di un numero di etichette molto ampio**
    - Globale: N*(N-1)
    - Locale: pari al max numero di connessioni che insistono su un IS

Path Setup

- Richiede solitamente una fase di connessione
  - Preparazione delle etichette in ogni nodo
  - Spesso la segnalazione adotta una diversa tecnica di forwarding (ad esempio *routing by network address*) su un circuito di segnalazione dedicato
    - Complica la rete, la quale deve gestire due tecniche di instradamento distinte
- Utilizzata in tecnologie di derivazione TLC
  - X.25, Frame Relay, ATM
  - Eccezioni
    - APPN: mondo IBM, dove la garanzia del servizio è cruciale
    - MPLS: mondo IP, ma utilizzato sul backbone dove sono necessarie altissime velocità di commutazione

Source Routing

- Scrittura dell'intero percorso da parte del mittente
  - Il mittente deve avere la conoscenza della topologia della rete
  - I sistemi devono mantenere a bordo una tabella di instradamento contenente le destinazioni con cui sono interessati a comunicare
  - Le entry di tali tabelle vengono solitamente calcolate in automatico tramite un processo di route discovery
- Usato:
  - Bridge source-routing
  - APPN+/HPR



Confronto

Forwarding by Network Address

+ semplicità: no setup, no stato
+ scalabilità (FW): nessuno stato "per-session"
- efficienza: dimensione dell'header dei pacchetti dati
- scalabilità (RT): tabelle di instradamento grosse (a meno di organizzazione gerarchica)
- garanzia: difficoltà a fornire garanzie di servizio
- multipath: non supporta percorsi multipli tra due destinazioni

Label Swapping

+ scalabilità (RT): tabelle di instradamento ridotte
+ efficienza: ridotta dimensione nell'header dei pacchetti dati
+ garanzia: possibilità di garanzia di servizio
+ multipath: percorsi multipli tra due destinazioni
- scalabilità (SETUP): processamento dei pacchetti path setup (critico con sessioni "corte")
- scalabilità (FW): stato "per-session" (necessario per la QoS)
- complessità: preparazione del percorso (processo di path setup, forwarding ad-hoc per pacchetti di path setup)

Source Routing

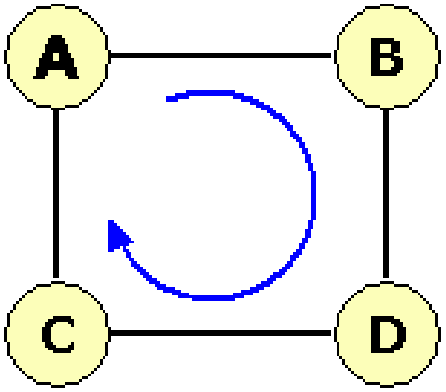
+ efficienza (IS): gli IS sono estremamente semplici
- efficienza (IS): gli ES devono preoccuparsi del calcolo del percorso

Livelli per l'Instradamento

- Livello 3 Network
  - OSI / Decnet
  - X.25
  - IP
- Livello 2
  - Bridged-LAN
  - Frame Relay
  - ATM
  - SMDS
- Livello 2-3 non distinguibili
  - APPN+/HPR

Algoritmi di routing: criteri di scelta

- **Orientati prevalentemente a realizzare il forwarding by network address**
- **Obiettivi**
  - Coerenza della rete: evitare loop, buchi neri, etc
  - Automaticità dell'algoritmo
- **Ottimalità**
  - Forma "preferibile" di coerenza
  - Metrica: criterio con cui l'ottimalità è misurata



Esempio di Coerenza: "inviare sempre i dati in senso orario"

Metriche

- **Esprimono la bontà del percorso tramite il costo**
  - Confronto di due percorsi: in base al costo inferiore
  - Criteri possibili
    - Minimizzazione delle distanze (es. numero di hops)
    - Minimizzazione dei ritardi
    - Più criteri combinati insieme con determinati pesi
    - Possono esserci criteri contrastanti (massimizzazione dell'utilizzo della rete e minimizzazione dei ritardi)
  - Complessità
    - Ottimalità assoluta → algoritmo magari ingestibile
    - Più criteri → algoritmo più complesso

Algoritmi di routing

- Possono non richiedere la conoscenza della topologia della rete

Problematiche di transitorio

- **Propagazione delle informazioni**
  - Una variazione di topologia / costo si propaga in tempo finito
  - Alcuni IS possono operare secondo la situazione aggiornata, mentre altri hanno ancora la situazione vecchia
- **Problemi**
  - Black Hole: un IS perde la conoscenza di una certa destinazione
  - Routing Loop: due IS hanno informazioni contrastanti e i pacchetti per alcune destinazioni vengono rimpallati tra due o più IS
    - Pericoloso perchè disturba anche traffico altrimenti regolare

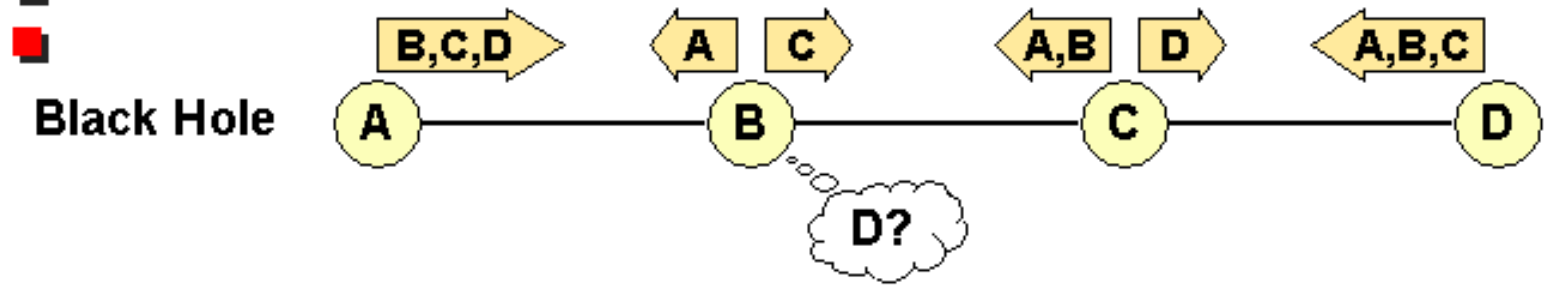
Caratteristiche di un algoritmo

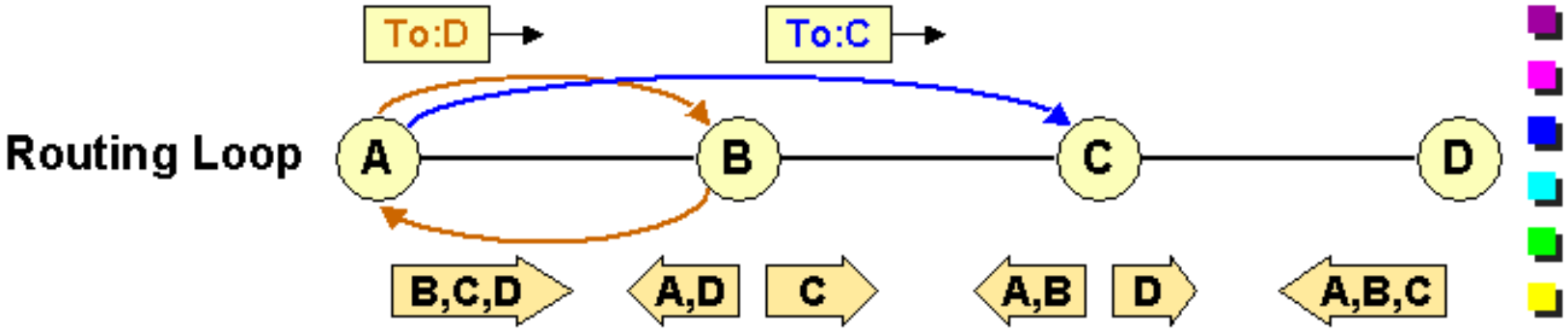
- **Semplicità: implementazione; inoltre IS hanno CPU e memoria finite**
- **Robustezza ed Adattabilità: alle variazioni di topologia; comprende**
  - Fault Detection
  - Autostabilizzazione
  - Robustezza Bizantina
- **Ottimalità: rispetto alla metrica scelta**
- **Stabilità: l'algoritmo deve attivarsi solo in caso di variazioni topologiche e di costi**
- **Equità: evitare di favorire / sfavorire particolari nodi**

Problematiche di transitorio

- **Propagazione delle informazioni**
  - Una variazione di topologia / costo si propaga in tempo finito
  - Alcuni IS possono operare secondo la situazione aggiornata, mentre altri hanno ancora la situazione vecchia
- **Problemi**
  - Black Hole: un IS perde la conoscenza di una certa destinazione
  - Routing Loop: due IS hanno informazioni contrastanti e i pacchetti per alcune destinazioni vengono rimpallati tra due o più IS
    - Pericoloso perchè disturba anche traffico altrimenti regolare

Black Hole e Routing Loop

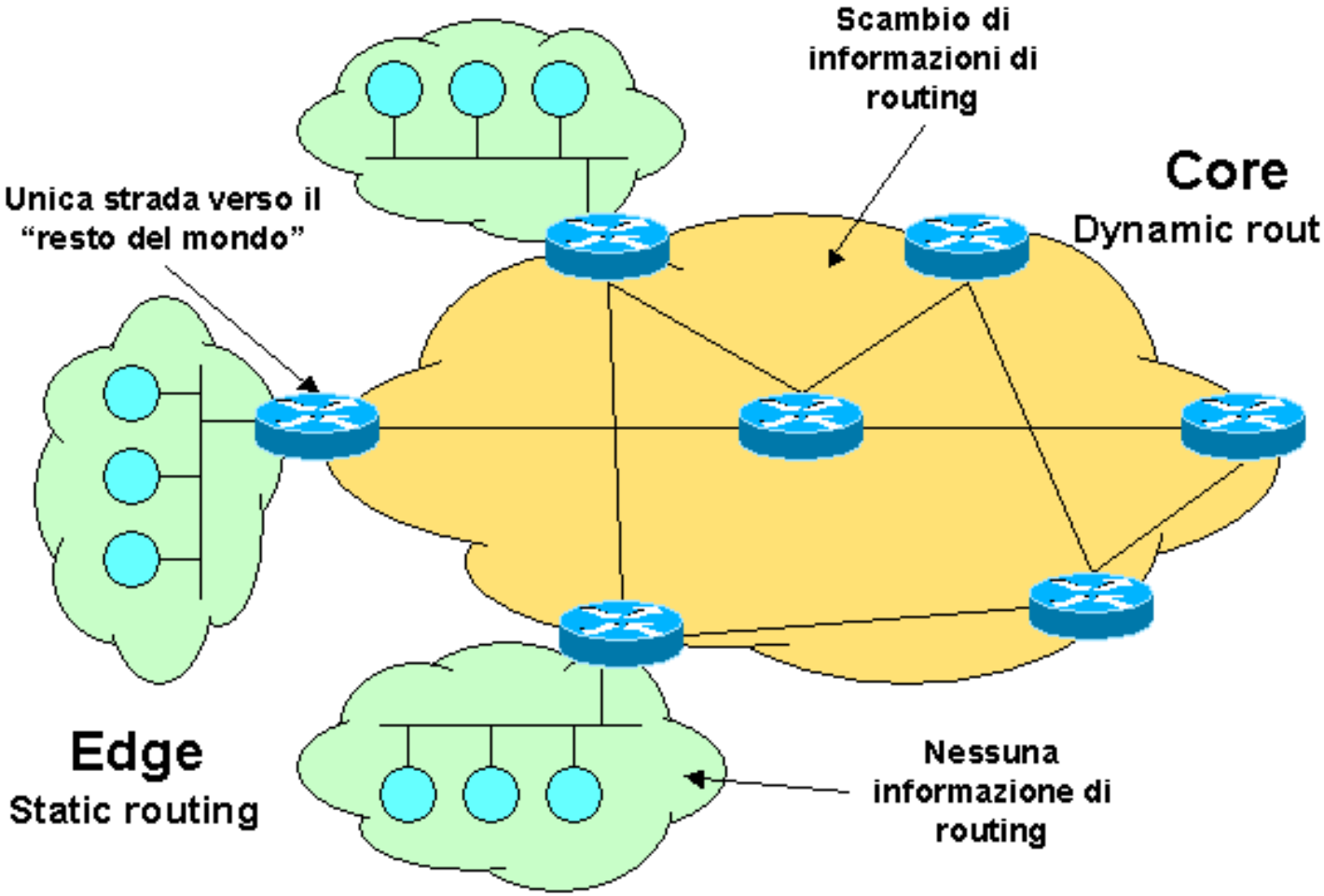




Classificazione

- **Algoritmi non adattativi (statici)**
  - Fixed Directory routing (routing statico)
  - Flooding, Selective Flooding e derivati
- **Algoritmi adattativi (dinamici)**
  - Routing Centralizzato
  - Routing Isolato
  - Routing Distribuito
    - Distance Vector
    - Link State
- **Algoritmi gerarchici**

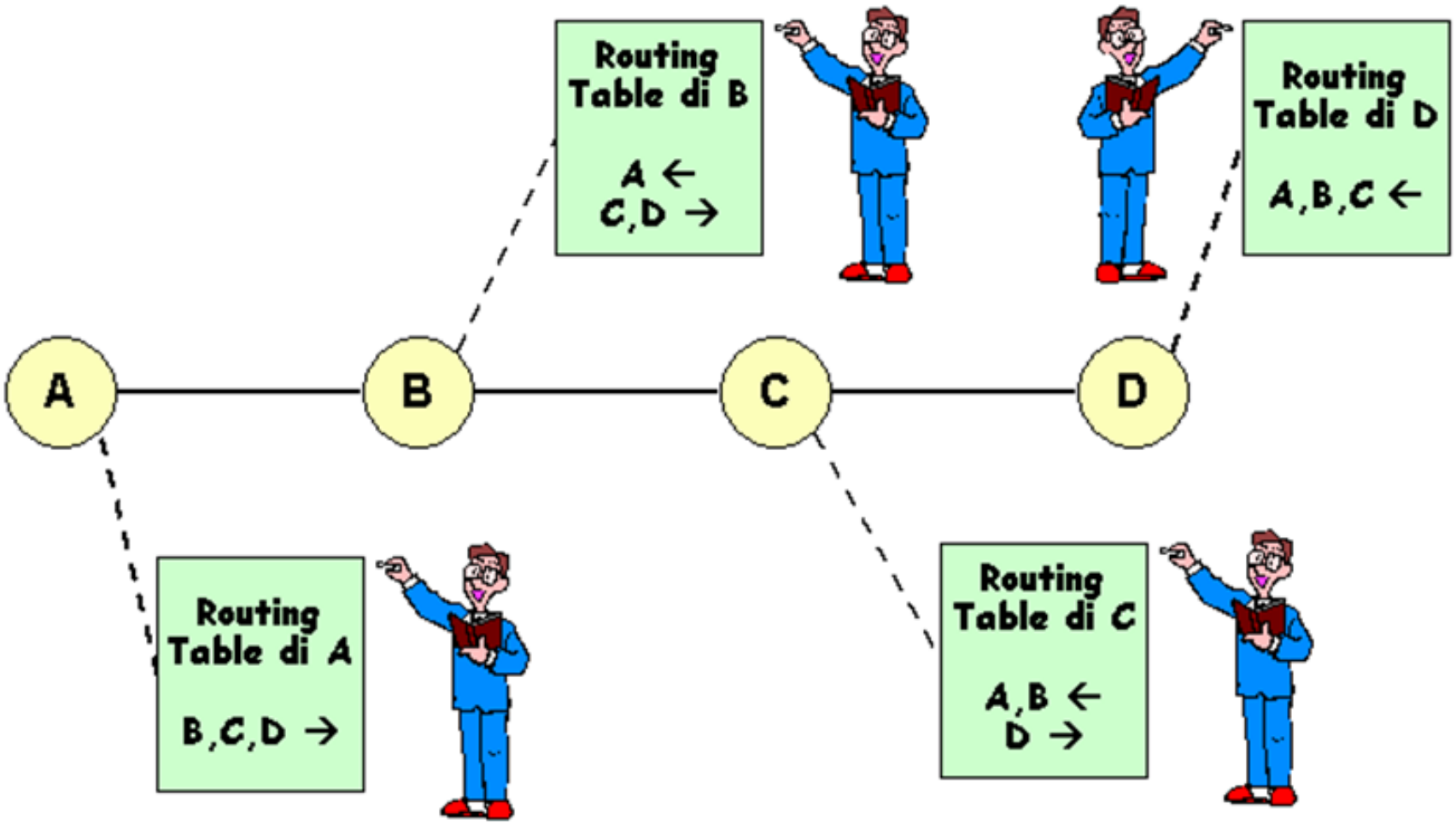
Routing statico e dinamico



Route di backup

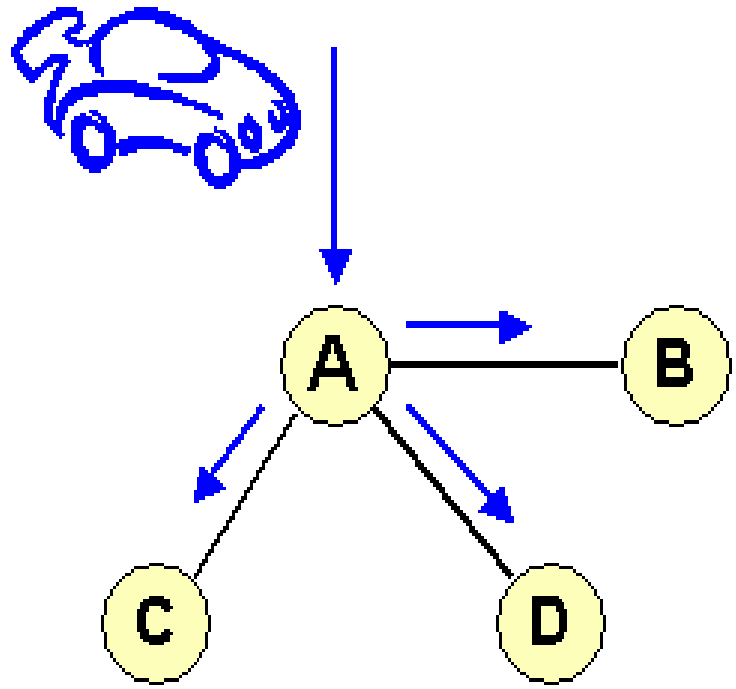
- **Concetto in uso in tecnologie statiche**
  - Rete telefonica, etc
- **Tecnologie distribuite**
  - In ogni rete sono ricavabili i percorsi di instradamento migliori
  - Se cambia la rete, vengono ricalcolati i percorsi di instradamento secondo la nuova configurazione
  - Non è necessario prevedere tutti i guasti (quindi il percorso di backup)
- **Variazione topologica della rete**
  - Non si ha solamente in caso di guasto
  - Anche l'inserimento in rete di una nuova destinazione è una variazione topologica

Fixed Directory Routing



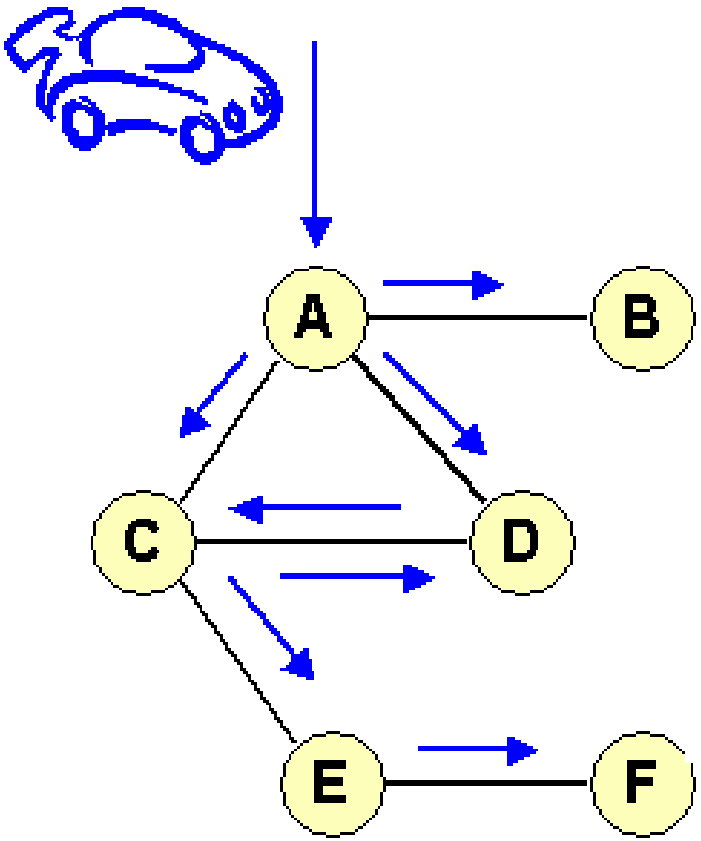
Flooding

- **Altissima probabilità di recapito del dato**
  - Altissima probabilità di ricevere il dato duplicato
  - Deve esistere ovviamente almeno un percorso disponibile
- **Notevole traffico di rete**
  - Utilizzato in reti dove il costo non è importante, ma l'affidabilità sì
  - Necessita di un campo "time to live" nel pacchetto dati

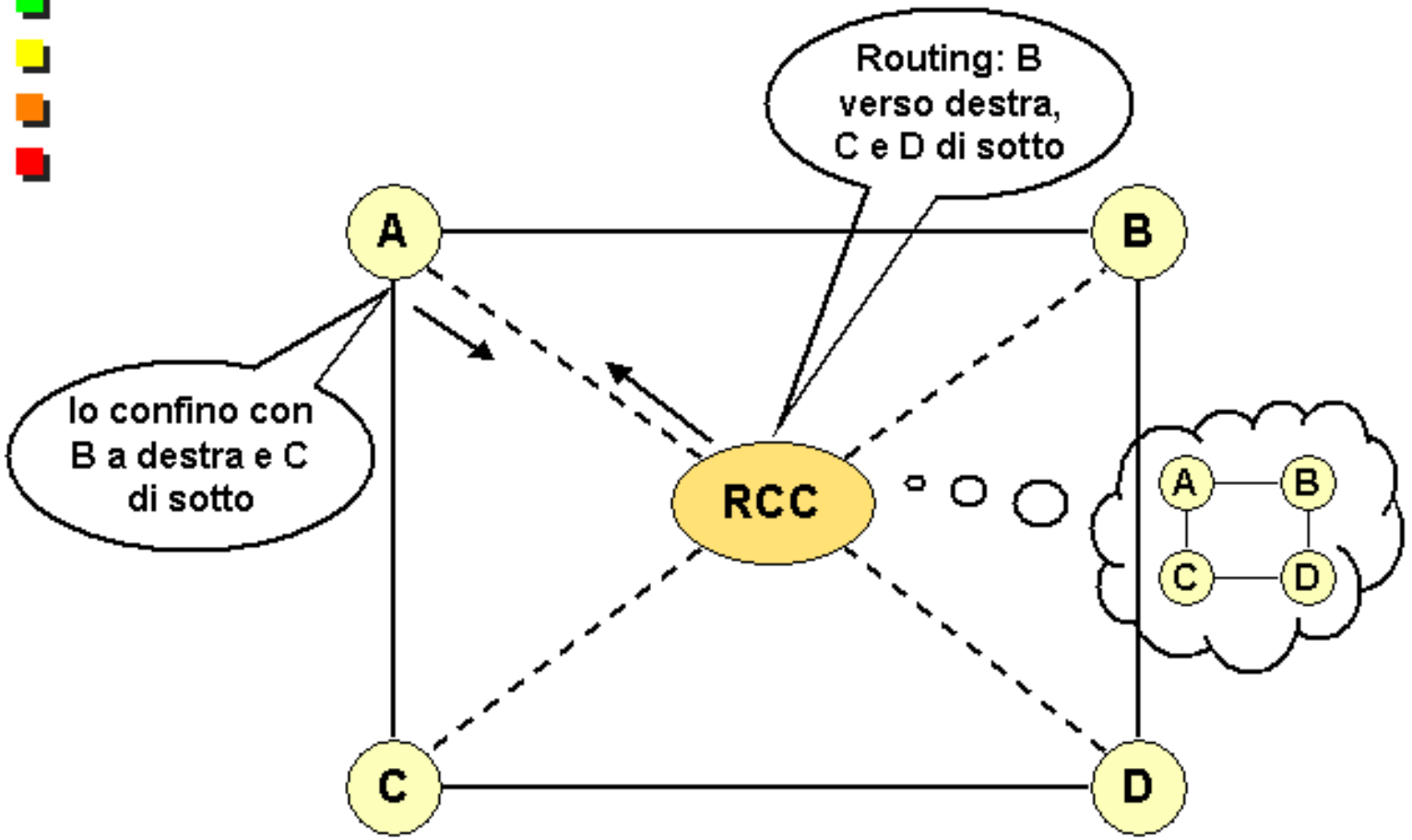


Selective Flooding

- **Obiettivi del flooding**
  - Robustezza nel consegnare i dati a destinazione
  - Nessuna configurazione degli IS
- **Nuovo obiettivo**
  - Traffico di rete ridotto
- **Random Walk**
  - Accademico
- **Selective Flooding (propriamente detto)**
  - Maggiore overhead di gestione sugli IS
  - Ogni nodo deve memorizzare se il pacchetto è già passato o meno
  - Varianti usate da IS-IS e OSPF



Routing Centralizzato (1)



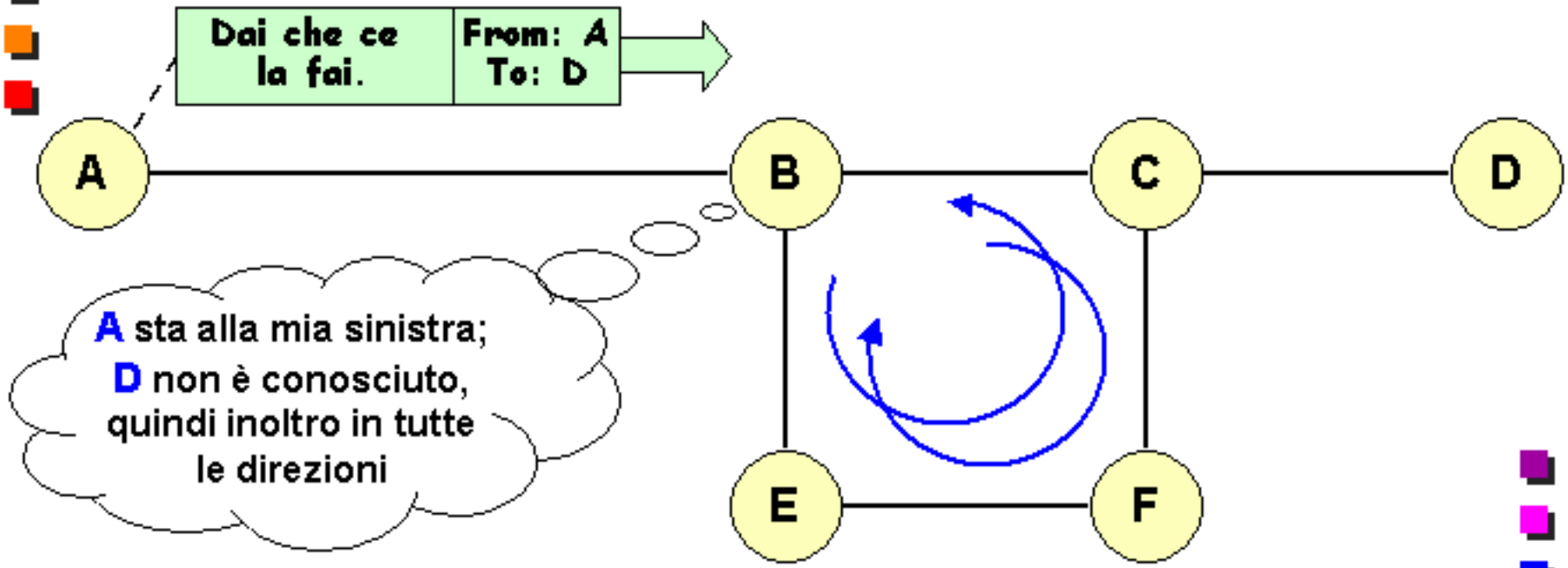
Routing Centralizzato (2)

- Esiste un Routing Control Center (RCC) che calcola e distribuisce le tabelle
- Il RCC riceve informazioni sullo stato della rete da tutti i nodi e le usa per calcolare le nuove tabelle
- Ottimizza le prestazioni e facilita il debugging
- Induce un notevole carico sulla rete specialmente in prossimità del RCC; single point of failure; poco adatto a reti con alta dinamicità
- Principi simili sono usati nella reti telefoniche

Routing Isolato

- **Duale rispetto al Routing Centralizzato**
  - Non esiste un RCC e ogni nodo decide l'instradamento in modo autonomo senza scambiare informazioni con gli altri IS
- **Backward Learning**
  - Più famoso algoritmo appartenente a questa classe
  - Usato dai bridge 802.1D per scoprire la locazione delle stazioni sulle varie LAN

Backward Learning



Variante: memorizzazione dei costi all'interno del pacchetto per la determinazione di percorsi secondari

Backward Learning

- **Caratteristiche**
  - Apprendimento basato sull'indirizzo sorgente del pacchetto
  - Riconosce solo i nodi "loquaci"
  - Non tiene conto di eventuali percorsi peggiorativi (che diventano corretti in caso di guasto)
  - Non contempla l'irraggiungibilità di un nodo (validità delle entry)
    - Necessario un apposito timer per poter eliminare una destinazione divenuta irraggiungibile
- **Variante: memorizzazione dei costi all'interno del pacchetto per la determinazione di percorsi secondari**
  - C: conoscerà che A è raggiungibile a costo 2 attraverso B e a costo 3 attraverso F

Routing Distribuito

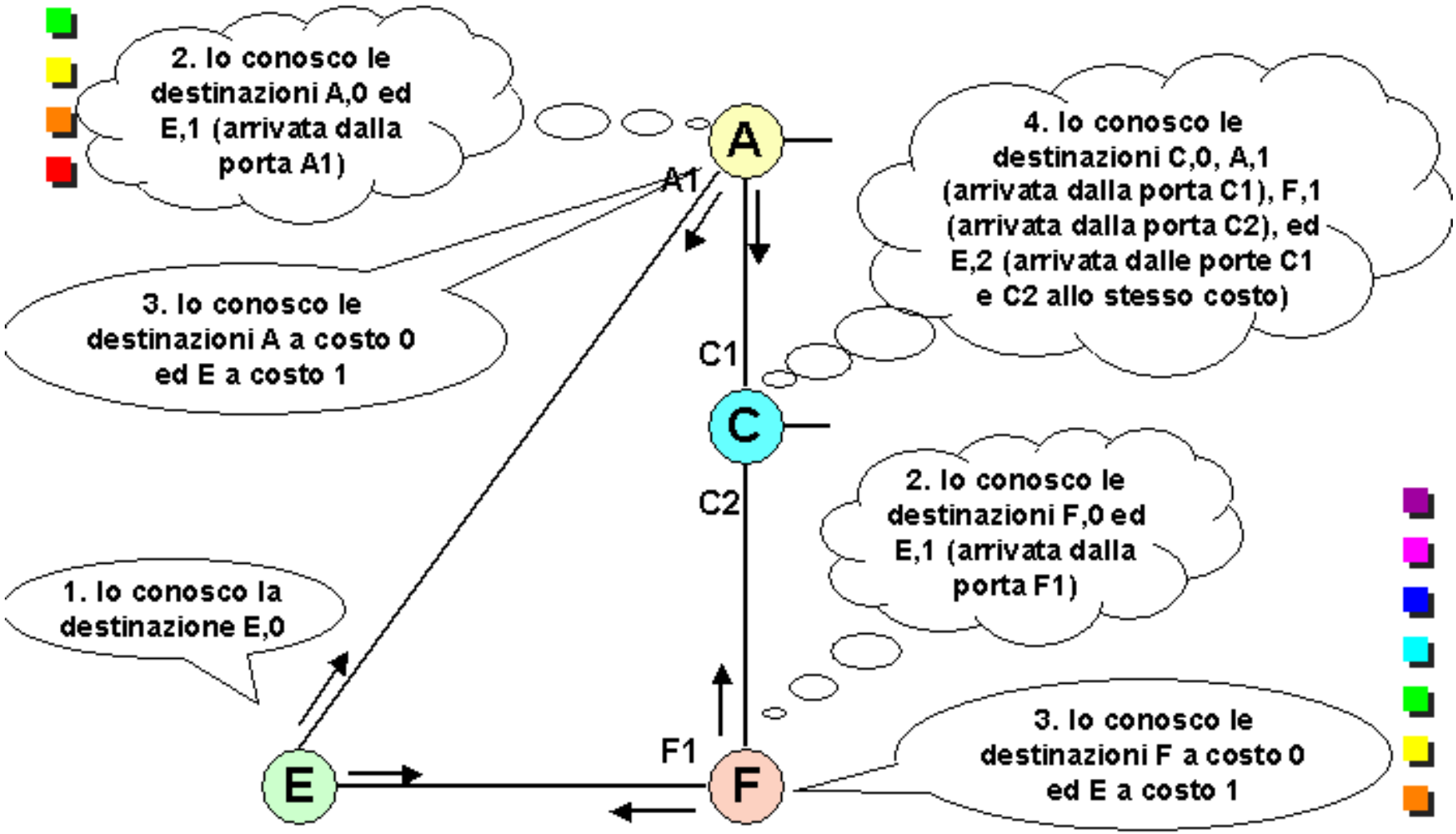
- **Modello Peer**
  - Unione dei vantaggi di routing Isolato e Centralizzato
  - Centralizzato: i router cooperano allo scambio di informazioni di connettività
  - Isolato: i router sono paritetici e non esiste un router "migliore"
- **Due algoritmi principali**
  - Distance Vector
    - Si distribuiscono 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

Algoritmo di routing: Distance Vector

Si supponga che i segnastradisti comunichino l'un l'altro a vista, per mezzo di opportuni segnali visibili solamente a distanza limitata, tutte le informazioni in loro possesso. Così, il segnastradista del primo incrocio comunicherà tutte le informazioni di routing (coppia località - distanza) in suo possesso a tutti i suoi colleghi che stanno in incroci adiacenti ("Io, nodo di Torino, raggiungo me stesso a costo zero"). Costoro ***apprenderanno*** *di non essere soli sulla rete e, a loro volta, comunicheranno ai loro vicini le informazioni in loro possesso ("Io, nodo di Genova, raggiungo me stesso a costo zero e Torino a costo 100"). In breve tempo, il metodo del passaparola farà sì che tutti i segnastradisti presenti nella rete autostradale conosceranno esattamente tutte le destinazioni raggiungibili e anche la direzione migliore per esse ("Io, nodo di Pisa, raggiungo me stesso a costo zero, Genova a costo 200 e Torino a costo 300").*

L'esempio introduce l'essenza dell'algoritmo Distance Vector (noto anche come algoritmo di Bellman-Ford) che si basa sul ***passaparola*** e sulla comunicazione tra soli **nodi adiacenti**. In altre parole, ogni nodo comunica tutte le informazioni di connettività in suo possesso a tutti gli IS adiacenti.

Principio del Distance Vector

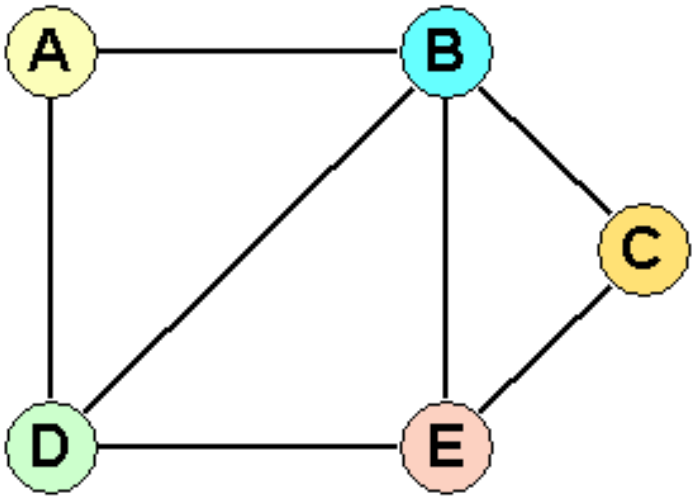


Significato di un Distance Vector

- **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 attravero 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

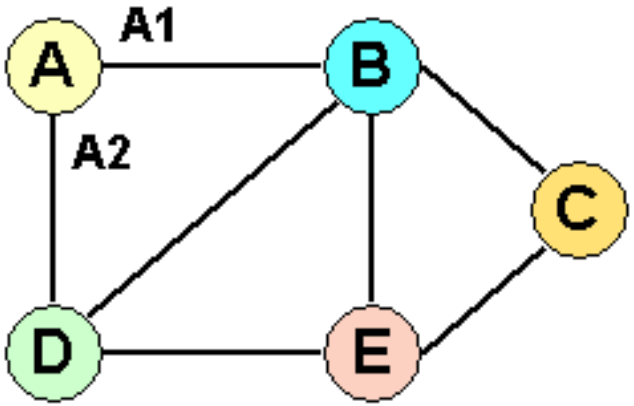


Memoria di A

Loc (A)	DV (B)	DV (D)
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

Fusione e generazione di DV



Ricevuto dalla linea A1

Ricevuto dalla linea A2

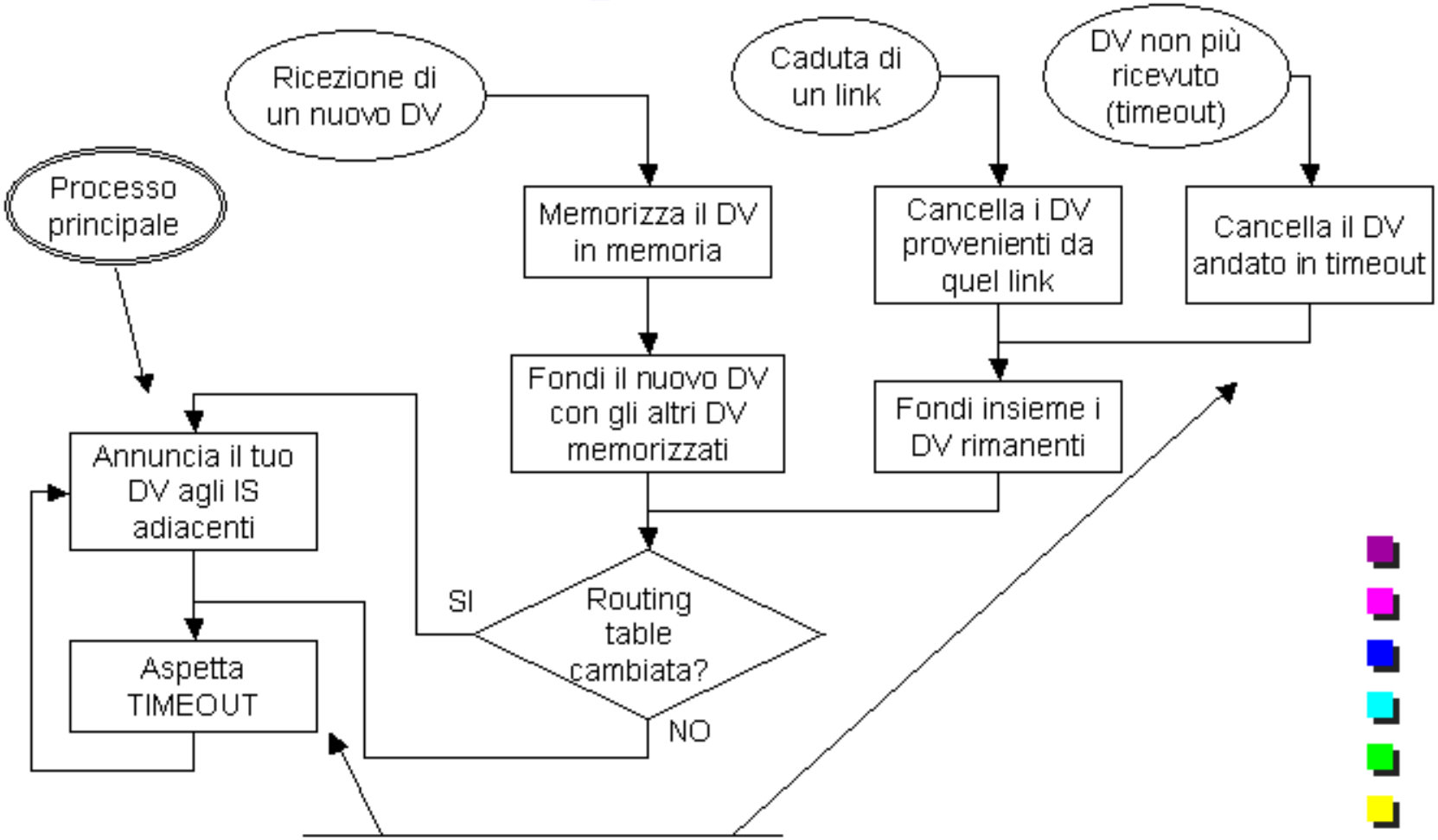
IS A:

Loc (A)	DV (B)	DV (D)
A, 0	~~A, 1~~	~~A, 1~~
	B, 0	~~B, 1~~
	C, 1	~~C, 2~~
	~~D, 1~~	D, 0
	E, 1	E, 1

ROUT. TABLE (A)		
A,	local,	0
B,	A1,	1
C,	A1,	2
D,	A2,	1
E	A2,	2

DV (A)
A, 0
B, 1
C, 2
D, 1
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 B,loc,0	C,loc,0	A,D1, 1 D,loc,0	E,loc,0

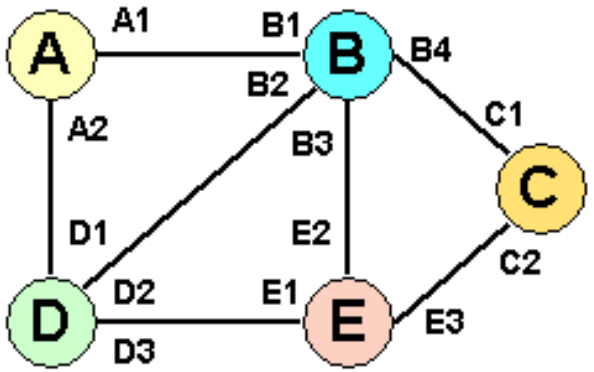
B e C emettono il DV

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

Tutti emettono il DV

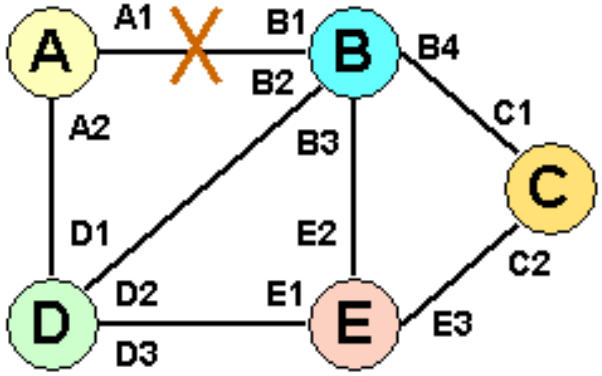
. . .

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

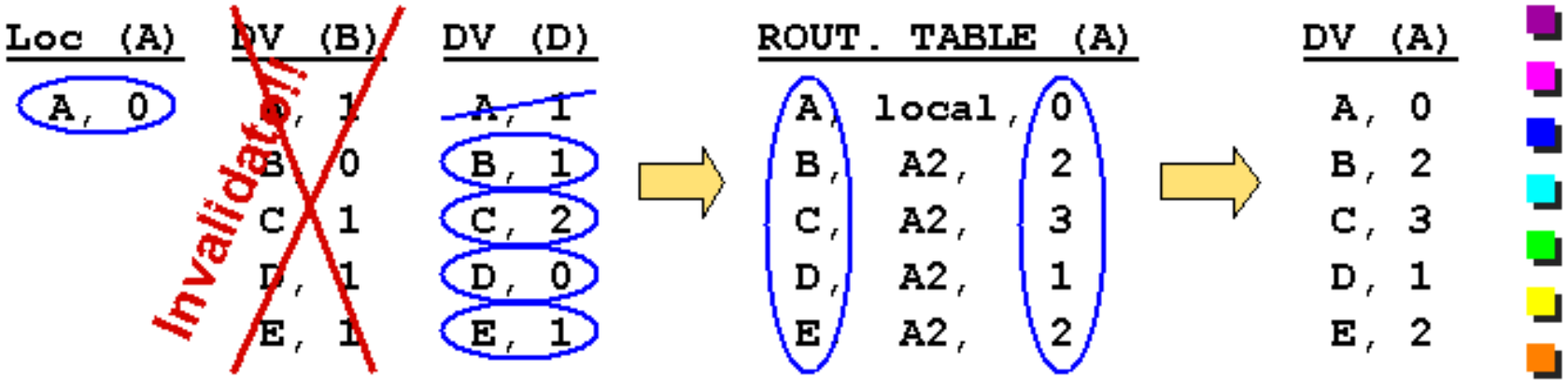


Esempio: caduta di un link

- 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:

Loc (B)	DV (A)	DV (C)	RT (B)
B, 0	A, 0	A, 2	A,B2 ,3
	B, 1	B, 1	B,loc,0
	C, 2	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

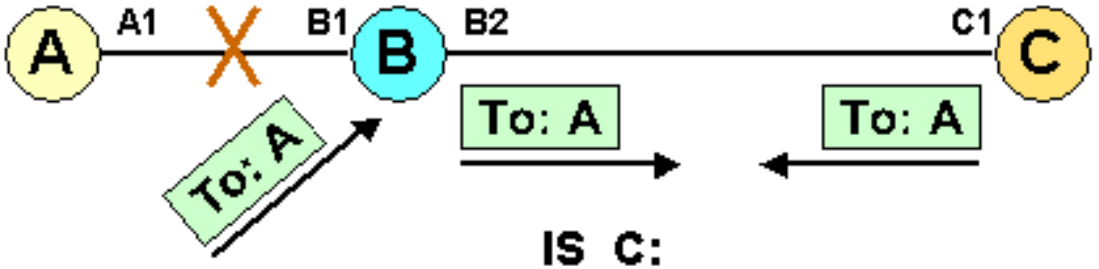
Loc (C)	DV (B)	RT (C)
C, 0	A, 3	A,C1 ,4
	B, 0	B,C1 ,1
	C, 1	C,loc,0

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)
B, 0

~~DV (A)~~
~~A, 0~~
~~B, 1~~
~~C, 2~~

DV (C)
A, 2
B, 1
C, 0

RT (B)
A,B2 ,3
B,loc,0
C,B2 ,1

IS C:

Loc (C)
C, 0

DV (B)
A, 1
B, 0
C, 1

RT (C)
A,C1 ,2
B,C1 ,1
C,loc,0

B emette il DV

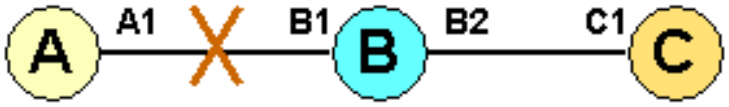
Loc (C)
C, 0

DV (B)
A, 3
B, 0
C, 1

RT (C)
A,C1 ,4
B,C1 ,1
C,loc,0

Conoscenza topologica

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



IS B:

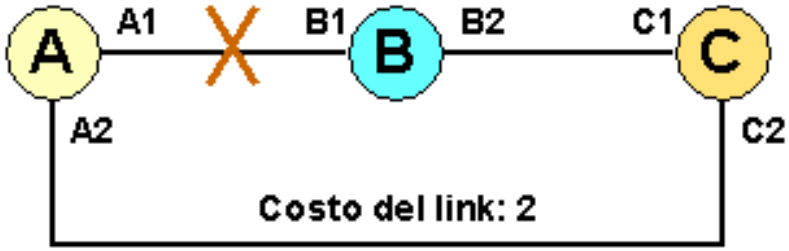
Loc (B)
B, 0

~~DV (A)~~

~~DV (A)~~
~~A, 0~~
~~B, 1~~
~~C, 2~~

DV (C)
A, 2
B, 1
C, 0

RT (B)
A,B2, 3
B,loc,0
C,B2, 1



IS B:

Loc (B)
B, 0

~~DV (A)~~

~~DV (A)~~
~~A, 0~~
~~B, 1~~
~~C, 2~~

DV (C)
A, 2
B, 1
C, 0

RT (B)
A,B2, 3
B,loc,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%

Distance Vector: pregi e difetti

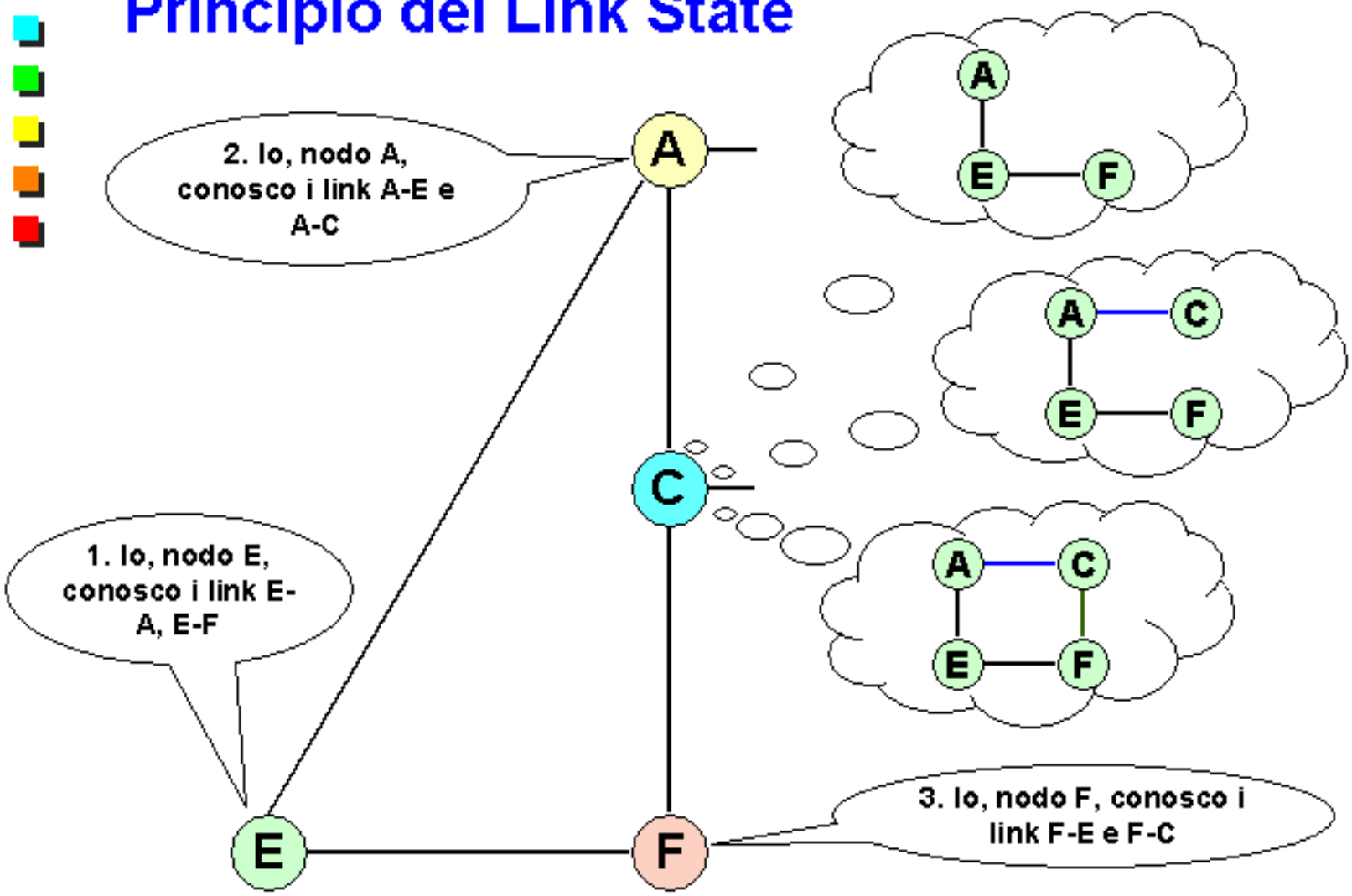
- Pregi:
  - Facilità di implementazione
- Difetti:
  - L'algoritmo ha un worst case esponenziale e un comportamento normale tra $O(n^2)$ e $O(n^3)$;
  - Lenta convergenza proporzionale a link e router più lenti;
  - Difficile capirne e prevederne il comportamento su reti grandi e complesse
    - Nessun 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 manzgement

Algoritmo di routing: Link State

Si supponga che i segnastradisti siano dotati di radio ricetrasmittenti e che comunichino tutti sulla stessa frequenza. Ognuno di essi controlla lo stato del proprio incrocio e verifica lo stato della strada fino all'incrocio successivo. Ogni segnastradista, quindi, conoscerà perfettamente lo stato dell'incrocio di sua pertinenza e delle strade che portano agli incroci adiacenti. Una volta che ognuno di essi ha chiara la situazione del suo incrocio, comunicherà, via radio, queste informazioni a tutti gli altri colleghi.
Questi riceveranno le informazioni dello stato di tutte le strade tra i vari incroci della rete autostradale e saranno pertanto in grado di determinare esattamente la topologia della stessa.

L'esempio introduce l'essenza dell'algoritmo Link State, che si basa su un **broadcast delle informazioni locali** a tutti i nodi della rete. Le informazioni locali, in questo caso, sono composte dallo stato di ogni collegamento tra un nodo e quelli adiacenti, informazione supposta semplice da ottenere. In altre parole, ogni nodo comunica le sole informazioni locali in suo possesso a tutti gli altri nodi della rete

Principio del Link State

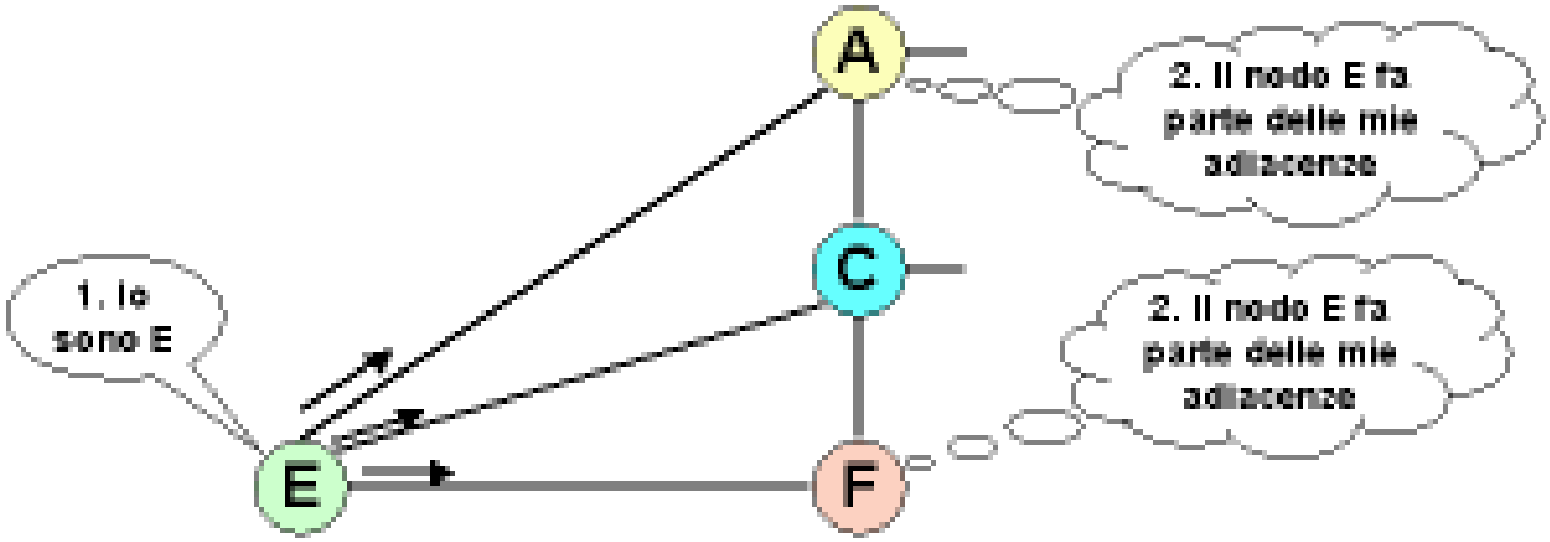


Componenti dell'algoritmo Link State

- 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 Adjiacencies

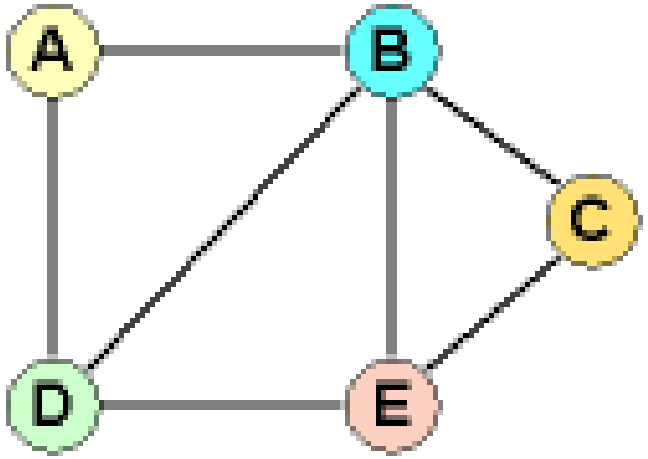
Neighbor Greetings (Hello)

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



Link State (Packet)

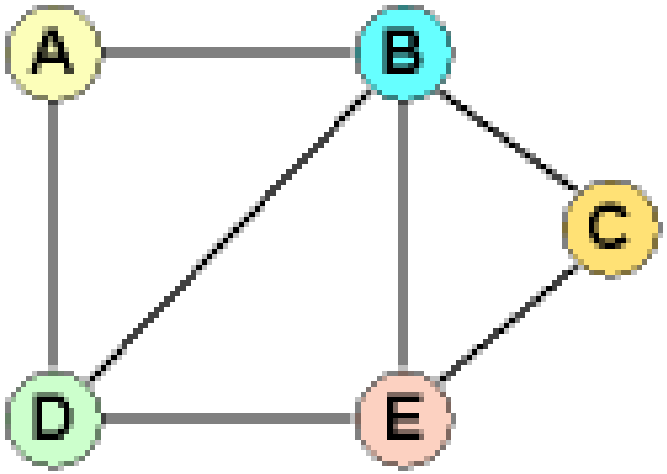
- Insieme delle adiacenze (*adiacenza – costo*)
  - **Link: caratterizzato da nodo sorgente – nodo destinazione**
    - Nodo sorgente: implicito nel mittente del LS
  - **Generato indipendentemente da ogni nodo**
  - **Ogni nodo inserisce l'elenco delle adiacenze e il loro costo**
  - **Ogni nodo memorizza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



Memoria di A

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			

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)

(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

L'algoritmo Link State prevede che il LS, generato attraverso il componente di Neighbor Greeting, debba essere recapitato a tutte le macchine Link State. Questa fase può essere vista come una trasmissione "**broadcas**t", con il problema che questa **funzione non è supportata dal livello network**, mentre è spesso supportata dal livello data-link. A regime, in effetti, questo tipo di trasmissione potrebbe essere simulata mandando lo stesso messaggio (in *unicast*) a tutte le macchine elencate nelle routing table, ma questo sistema non potrebbe funzionare nella fase di transitorio dove alcune destinazioni possono essere ancora sconosciute. La soluzione consiste nell'utilizzare il *Flooding* o meglio *di Selective Flooding*, preferito per la sua maggiore accortezza nell'utilizzo della banda trasmissiva, **per recapitare lo stesso dato a tutte le destinazioni**

Calcolare la routing table: algoritmo di Dijkstra

L'**algoritmo di Dijkstra** (o **Shortest Path First**) è il meccanismo che permette, a fronte dell'insieme delle adiacenze, di ricavare la routing table.

A rigore, non è necessario utilizzare tale algoritmo: sarebbe sufficiente che tutti i nodi utilizzassero lo stesso algoritmo al fine di ottenere percorsi coerenti.

In pratica, però, questo discorso risulta puramente accademico e il Link State ingloba l'algoritmo SPF (che è in grado di calcolare il cammino migliore tra due nodi) per il computo della routing table.

L'algoritmo di Dijkstra è molto semplice ed è basato sull'idea che, dato un nodo radice, esisterà sicuramente un altro nodo la cui distanza dalla radice sarà minima.

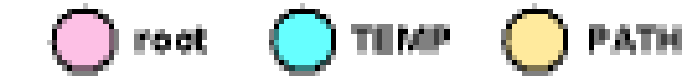
Algoritmo di Dijkstra (1)

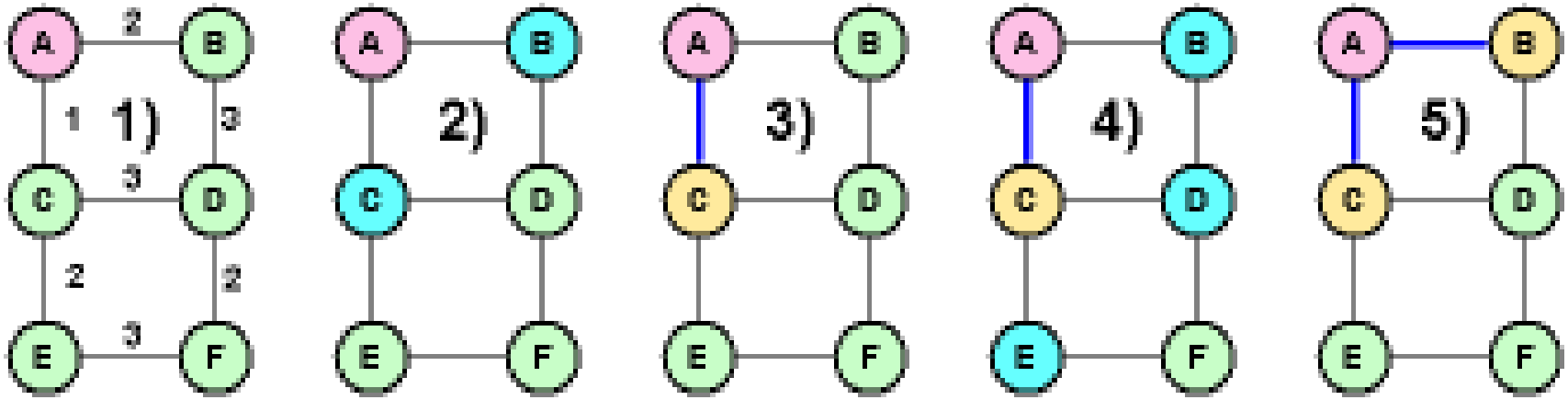
- Algoritmo di Dijkstra
  - **Usato per il calcolo dello Spanning Tree (albero dei cammini di costo minimo avente il nodo come radice) del grafo**
- Funzionamento
  - **Si individua il nodo "più vicino" a quello radice**
  - **Se il nodo esiste:**
    - Si inserisce questo percorso nella routing table
    - Si ripete da capo, selezionando il nodo a costo immediatamente superiore
  - **Altrimenti: termina quando è stato trovato un percorso per tutti i nodi**

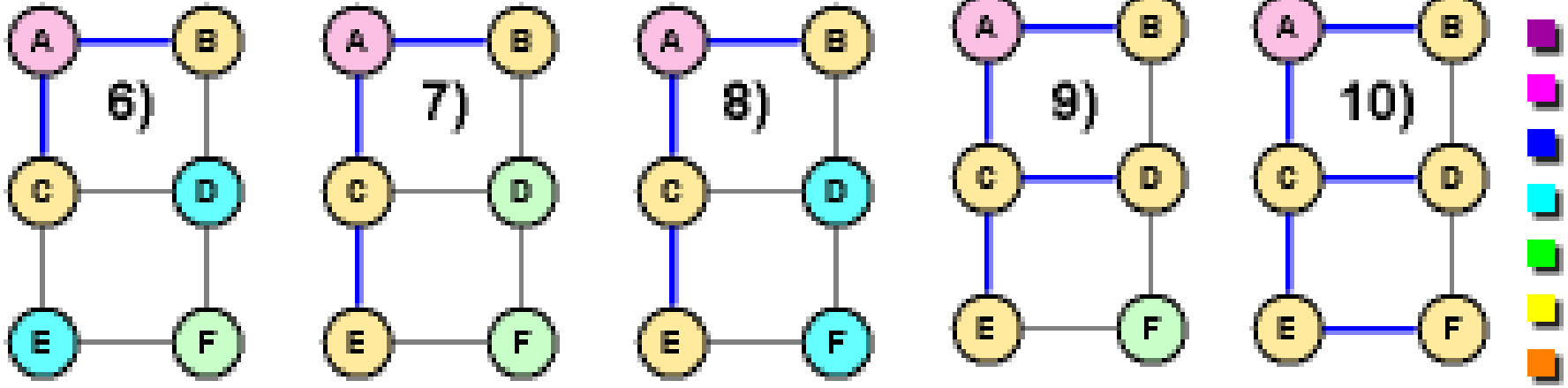
Algoritmo di Dijkstra (2)

- 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(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

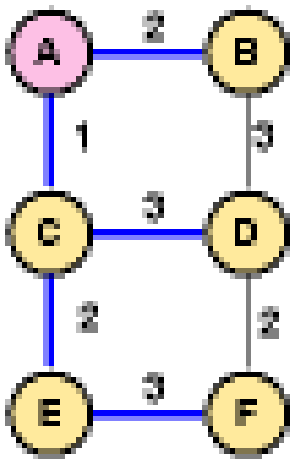


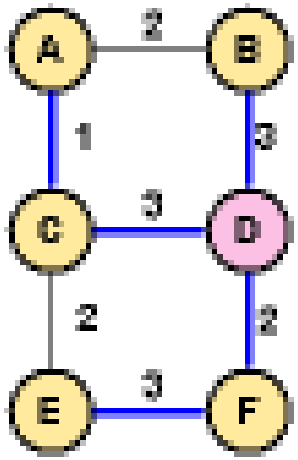




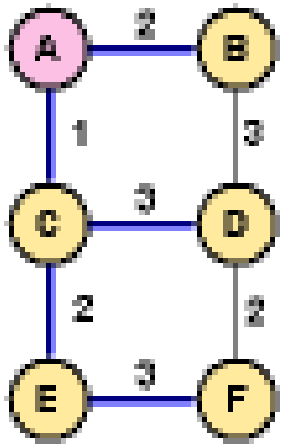
Alberi di instradamento

- 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				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			2
E		2			3
F			2	3	

C+1

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

B+2

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

B+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 L
  - 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

La complessità algoritmica dell'algoritmo di Dijkstra dipende fortemente dal **numero di link presenti nella rete**: il numero totale dei percorsi che verranno considerati è proporzionale al numero dei link nella rete. Viceversa, la complessità dell'algoritmo di Bellman-Ford dipende in equal misura dal numero di nodi presenti sulla rete (N) e dal numero di link (L).

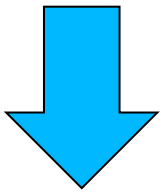
Il primo algoritmo è quindi molto più scalabile del secondo dal momento che un termine compare come logaritmo.

Nb: la complessità sopra citata è solo una delle componenti della complessità di un eventuale protocollo Link State: infatti l'algoritmo di Dijkstra è solo uno dei tanti pezzi necessari a permettere il funzionamento dell'algoritmo Link State.

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

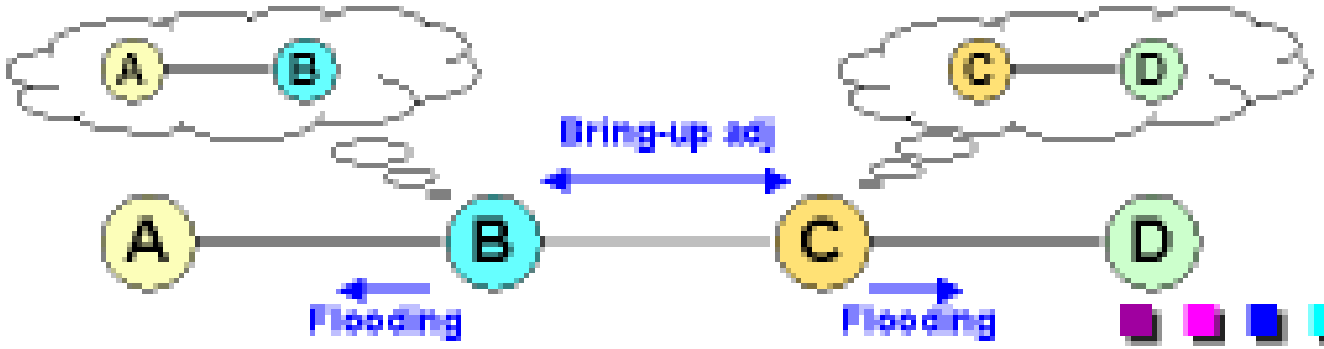
Esiste ancora un problema che è la **costruzione del database** per un nodo che **si attiva dopo che tutta la rete ha già scambiato i propri LS**: in questo caso il nodo dovrebbe comunque aspettare parecchio tempo prima che il nuovo scambio di LS venga attivato, e quindi possa capire correttamente la topologia della rete.



A questo problema risponde l'algoritmo che ha il compito di allineare il contenuto del database di due nodi vicini, ossia il ***bringing up adjiacencies***

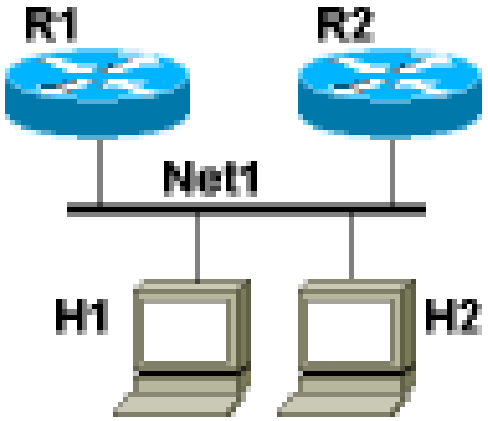
Bringing up adjiacencies

- 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

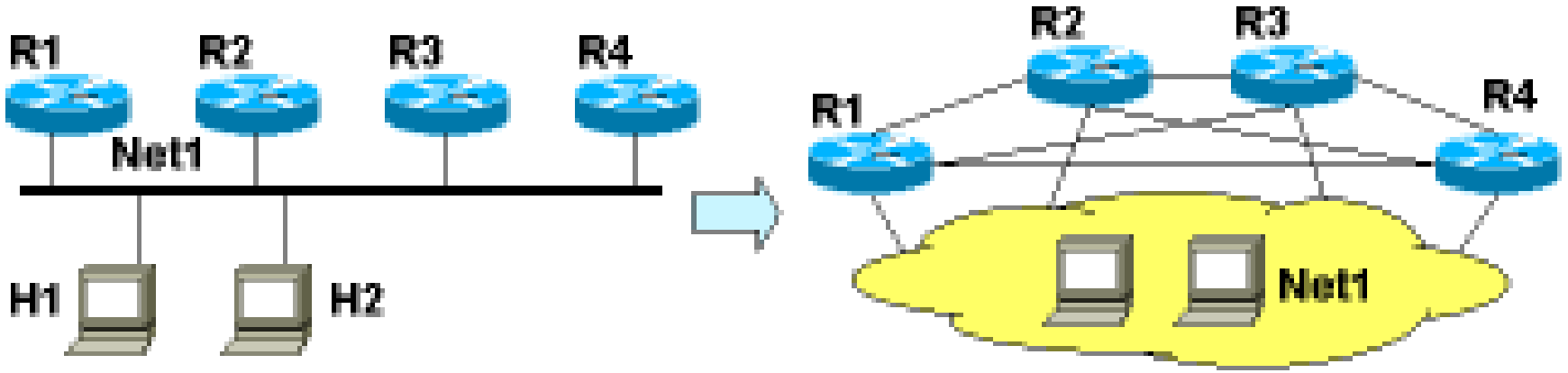



```
Link State Database (ideale)
R1   R2/1 H1/1 H2/1
R2   R1/1 H1/1 H2/1
H1   R1/1 R2/1 H2/1
H2   R1/1 R2/1 H1/1
```

```
Link State Database (reale)
R1   R2/1 Net1/1
R2   R1/1 Net1/1
```

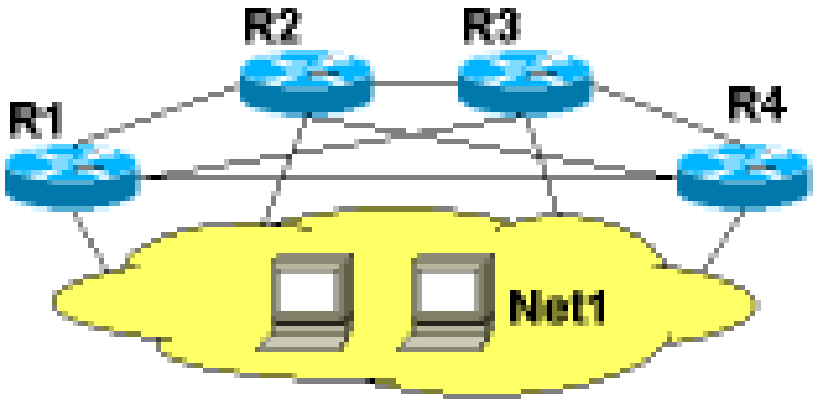
Link State e reti reali: reti broadcast (1)

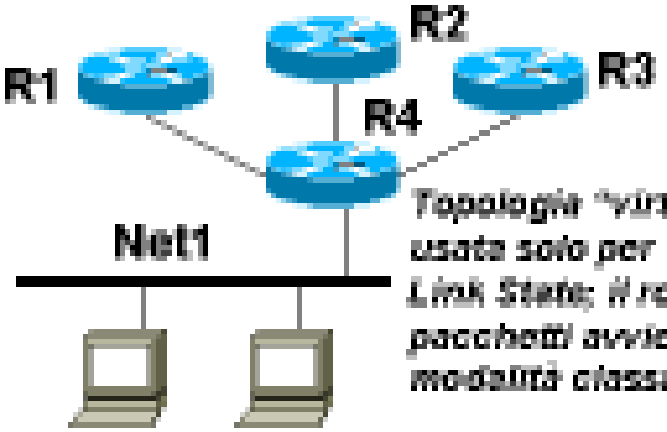
- Link State: contiene le adiacenze
  - N IS su rete broadcast = N² adiacenze (N² link)
  - ➔ Link State Database diventa ingestibile
  - ➔ Complessità di Dijkstra esplode (proporzionale al numero di link)
  - ➔ Fase di *Bringing up adjiacencies* esplode



Link State e reti reali: reti broadcast (2)

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





Topologia "virtuale" usata solo per l'algoritmo Link State; il routing dei pacchetti avviene in modalità classica

```
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
```

```
Link State Database
R1   R4/1
R2   R4/1
R3   R4/1
R4   R1/1 R2/1 R3/1 Net1/1
```

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 (1)

- 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 (2)

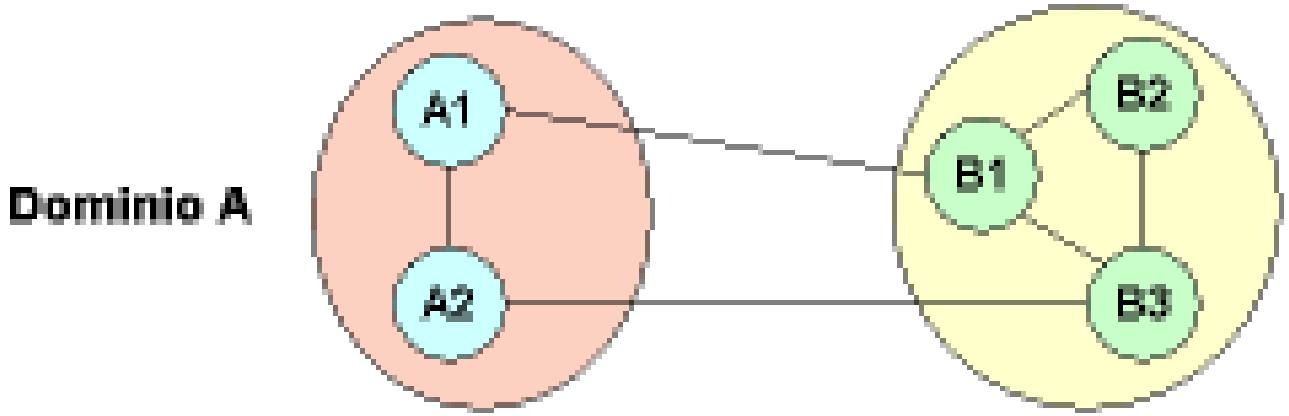
- Semplicità
  - **Distance Vector: unico algoritmo**
  - **Link State: ingloba molti componenti distinti**
- Memoria occupata (in ogni nodo)
  - **Dijkstra: N * A [ogni LS contiene A adiacenze]**
  - **Bellman-Ford: A * N [ogni DV contiene N destinazioni]**
  - **Valori equivalenti**
- Traffico
  - **Favorevole al Link State**
  - **Pacchetti di Hello molto più piccoli rispetto a DV**

Routing Gerarchico

- Problemi
  - **Scelte architetturali diverse da parte di gestori di "domini" differenti**
    - Routing Distance Vector in un dominio, Link State in un altro
  - **Scalabilità del routing**
    - Un modello peer non scala
- Soluzione
  - **Partizionamento della rete in domini di routing autonomi**
  - **Dominio di routing: porzione di rete nel quale è implementata una tecnologia di routing omogenea**
  - **Introduzione di una nuova regola di routing per il collegamento dei nuovi domini**
  - **Soluzione adottata da tutte le architetture di rete**

Regole del routing gerarchico

- Assunto
  - Ogni dominio di routing ha il proprio protocollo di routing attivo su tutte le stazioni interessate
- Regole
  - Se sorgente e destinazione risiedono all'interno dello stesso dominio di routing, utilizza le informazioni di routing generate dall'algoritmo di routing implementato
  - Se sorgente e destinazione risiedono in due distinti domini di routing, seleziona il più vicino punto di uscita dal tuo dominio verso il dominio di destinazione e inoltra il dato a quell'egress router

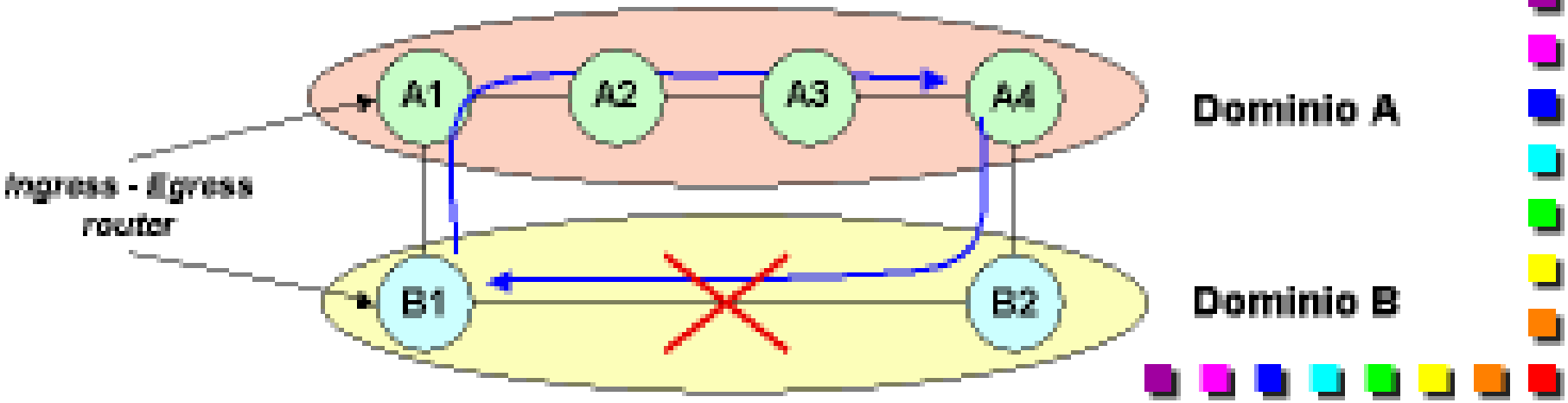


Osservazioni sul routing gerarchico

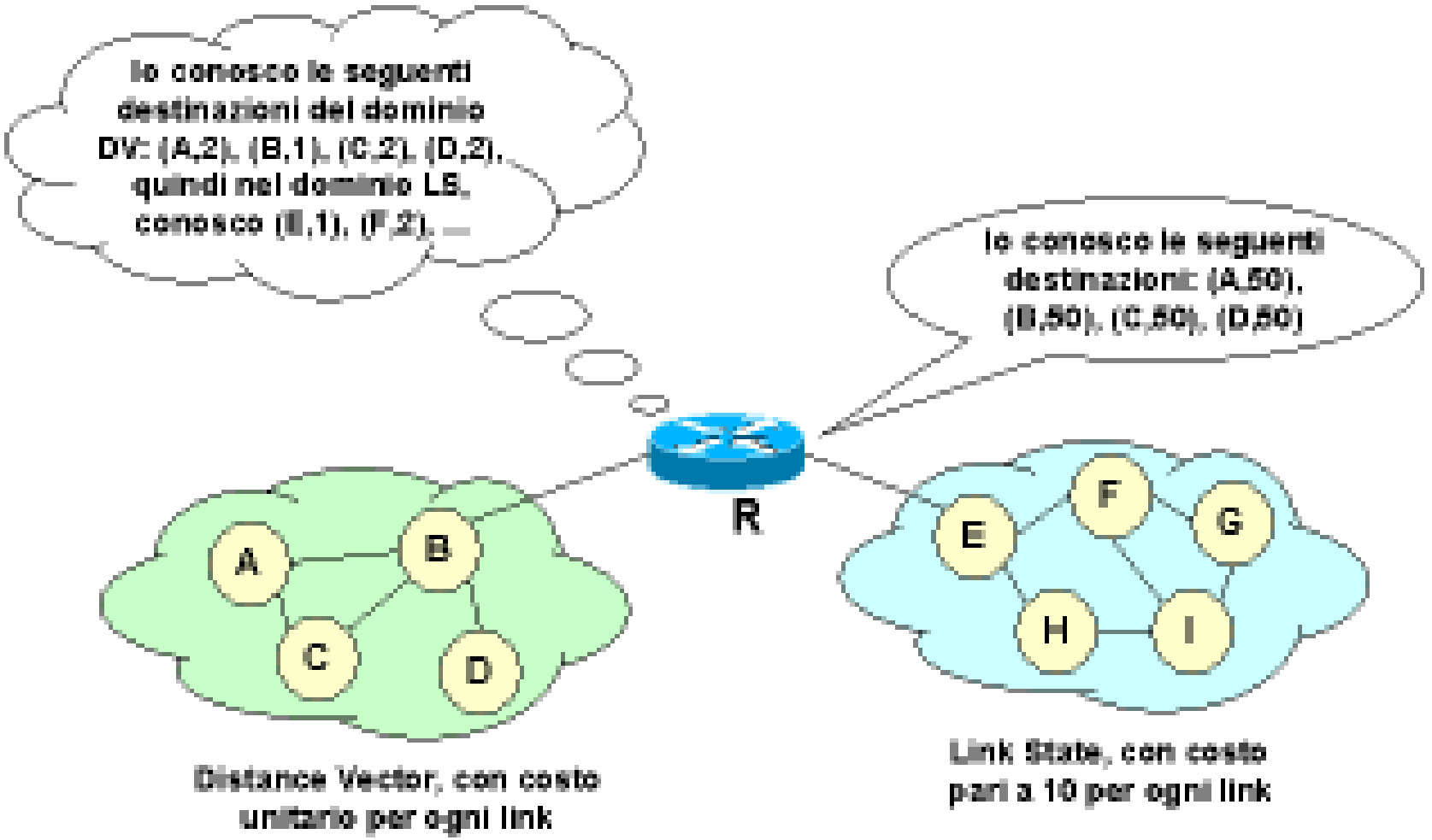
- Conoscenza topologica
  - Limitata al proprio dominio di routing; la conoscenza esterna comprende solo l'elenco delle destinazioni ma non il loro costo
- Percorso dei dati
  - È l'unione di due percorsi ottimi: dal mittente al più vicino IS di uscita verso il dominio di destinazione, quindi da questo alla destinazione vera e propria
  - La prima parte del percorso (mittente – egress router) è la stessa per tutte le località del dominio remoto
- Prestazioni
  - Il routing gerarchico migliora la scalabilità della rete, seppure in taluni casi può portare ad instradamenti non ottimi, ma corretti

Partizione di domini

- Percorsi: possono essere subottimali e asimmetrici
  - Esempio: scambio di informazioni tra B1 ed A4
- Partizione di domini
  - Se il link tra B1 ed B2 si interrompe, non è più disponibile alcun percorso di ritorno, nonostante la connettività fisica esista (attraverso A3-A2-A1)
  - Interruzione dei link A1-B1 e A4-B2: nessun problema
- Ogni dominio deve essere fortemente connesso
  - La caduta di un link non ne deve provocare il partizionamento



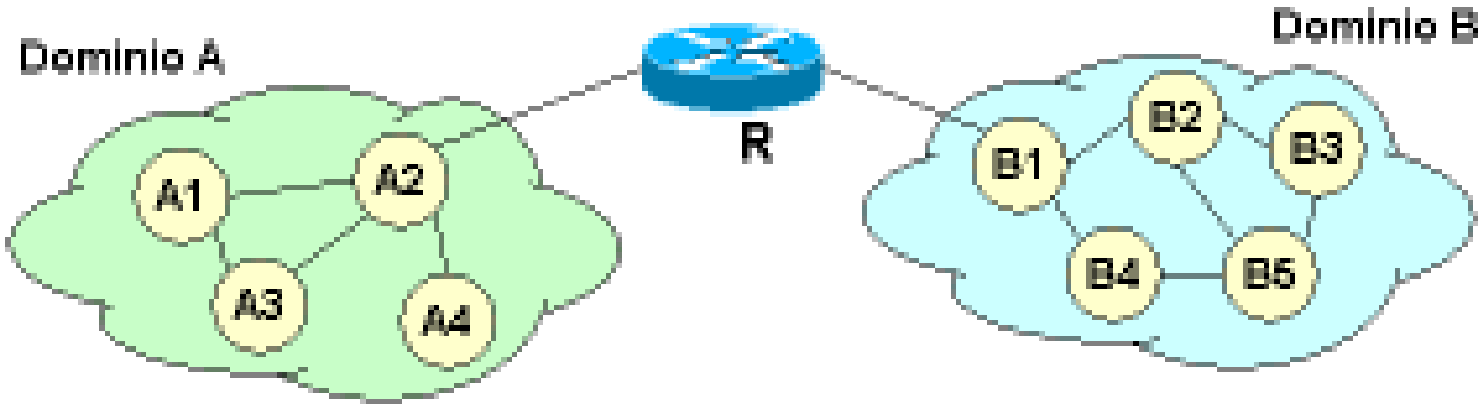
Redistribuzione (1)



Redistribuzione (2)

- Tecnica per collegare due domini di routing distinti
  - **Es. dominio RIP e OSPF**
    - Non necessariamente vi sono algoritmi/protocolli distinti
  - **Può essere usata per realizzare il routing gerarchico**
- Le route apprese in un dominio possono venire distribuite anche nell'altro
  - **Vengono viste come route esterne al dominio**
- L'informazione di costo può venire:
  - **Mantenuta: si ottiene virtualmente un dominio unico**
  - **Posta ad un valore standard: si realizza il routing gerarchico**
- È possibile una redistribuzione selettiva (in una sola direzione)

Perdita delle conoscenza topologica



LS aggiuntivi in B

Unico dominio Link State

R	A1	A2	A3	A4
A2,1	A2,1 A3,1	A1,1 A3,1 A4,1	A1,1 A2,1	A2,1

Due domini Link State

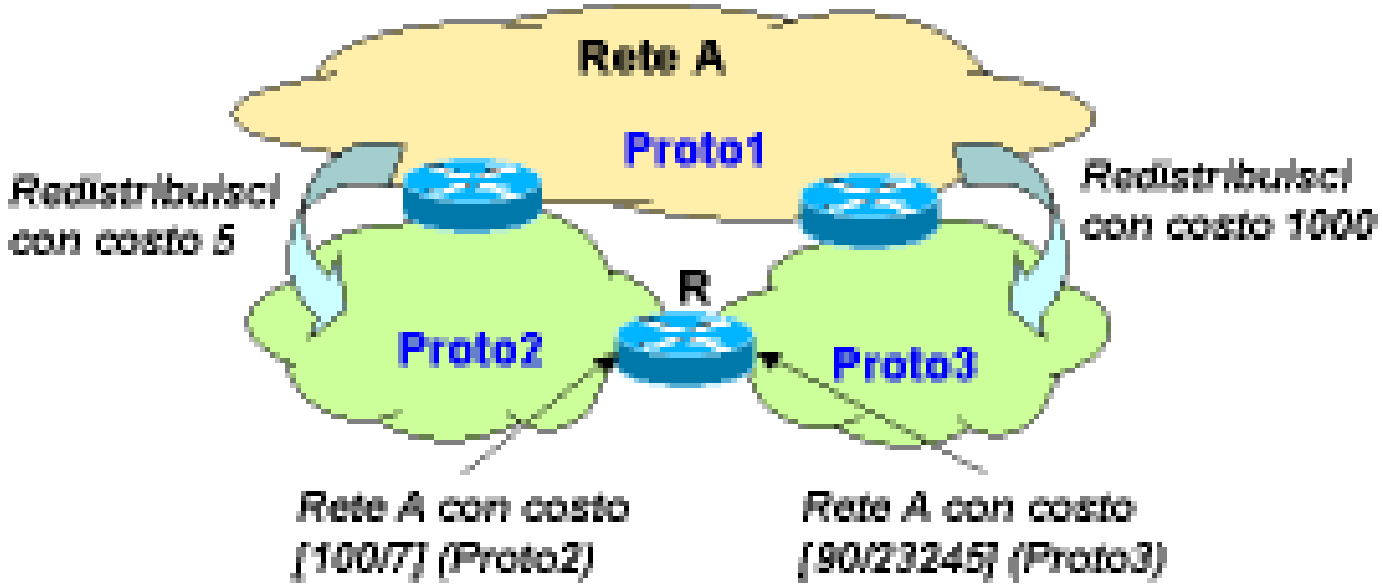
R
A1,2 A2,1 A3,2 A4,2

Due domini Link State con compattamento delle informazioni

R
A*,1

Redistribuzione e costi

- Ogni protocollo di routing ha un costo intrinseco
  - In caso di ambiguità (impossibilità di comparazione dei costi) è questo parametro a decidere la route
  - Nell'esempio viene privilegiato il dominio servito dal protocollo *Proto3* (che ha costo 90 contro 100 di *Proto2*)
  - Non è necessario che il l'IS R abbia la redistribuzione attiva

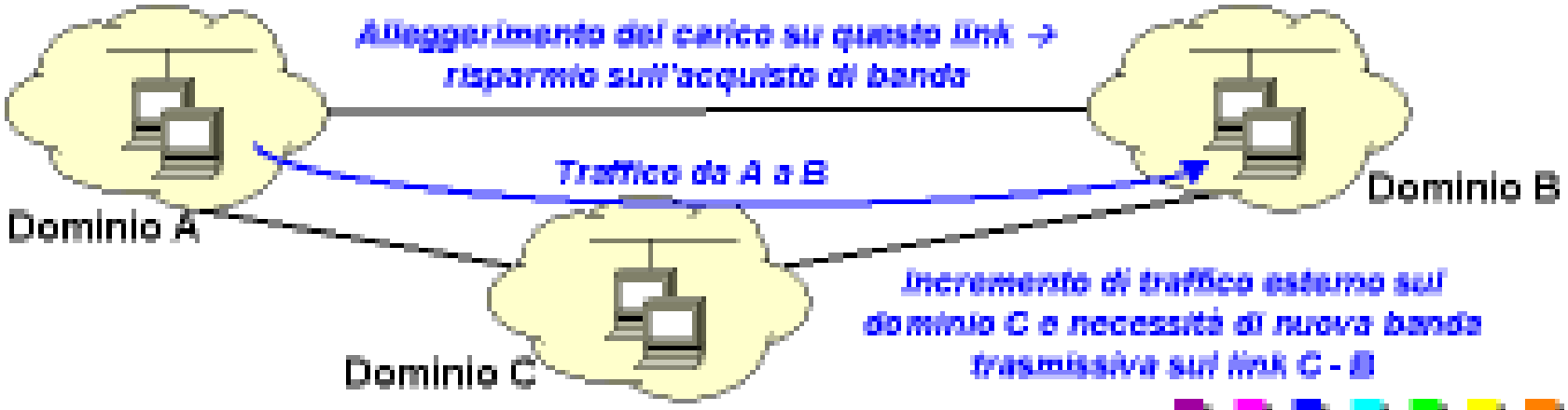


Routing inter-dominio

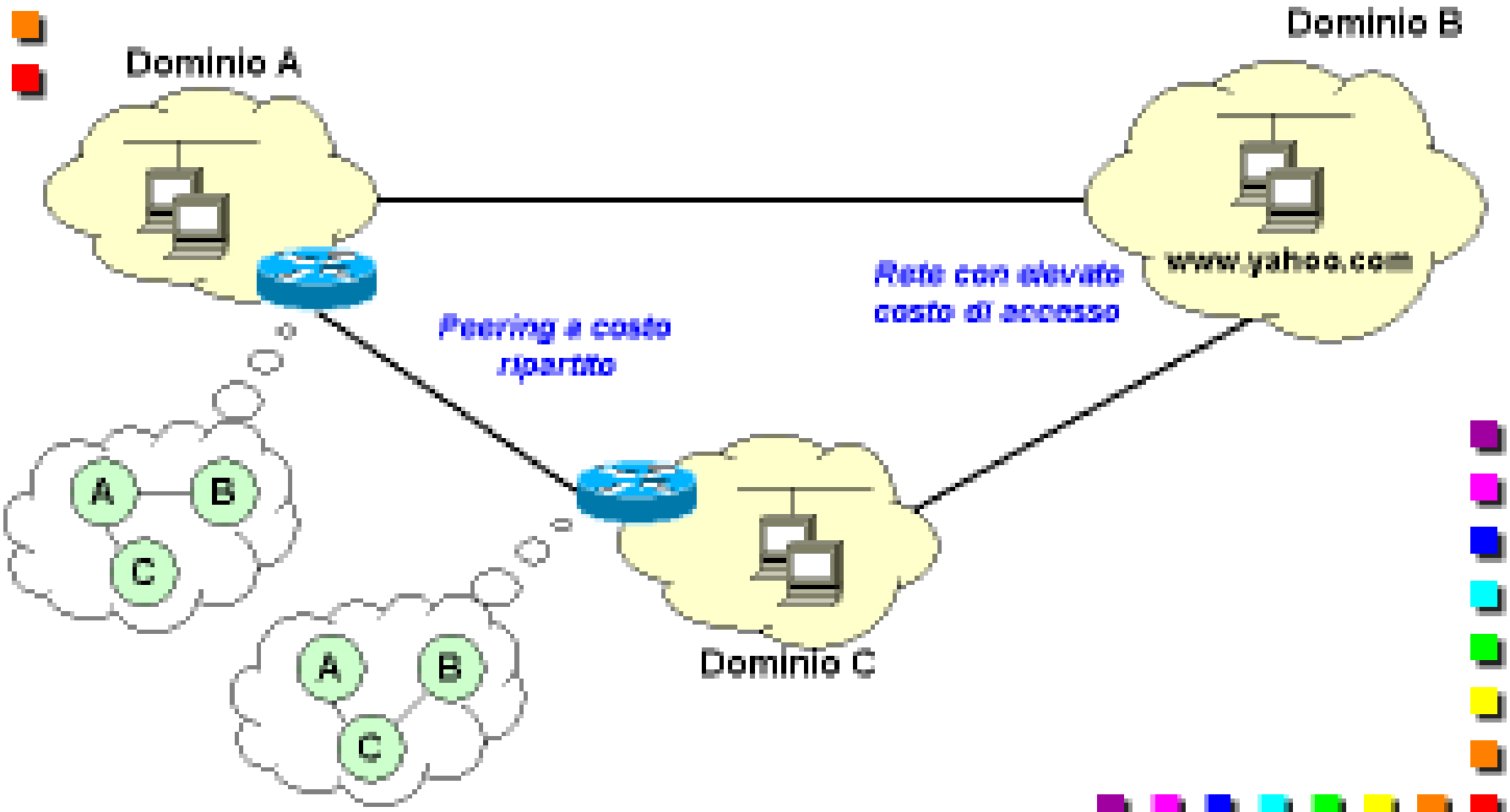
- Spesso sinonimo di routing tra diverse entità amministrative (gestori)
  - Entità amministrative: Autonomous System in IP
  - Può includere più domini di routing
- Esigenze di connettività possono passare in secondo piano rispetto ad altre esigenze
- Nuove esigenze
  - Economiche, amministrative, di sicurezza

Altre esigenze

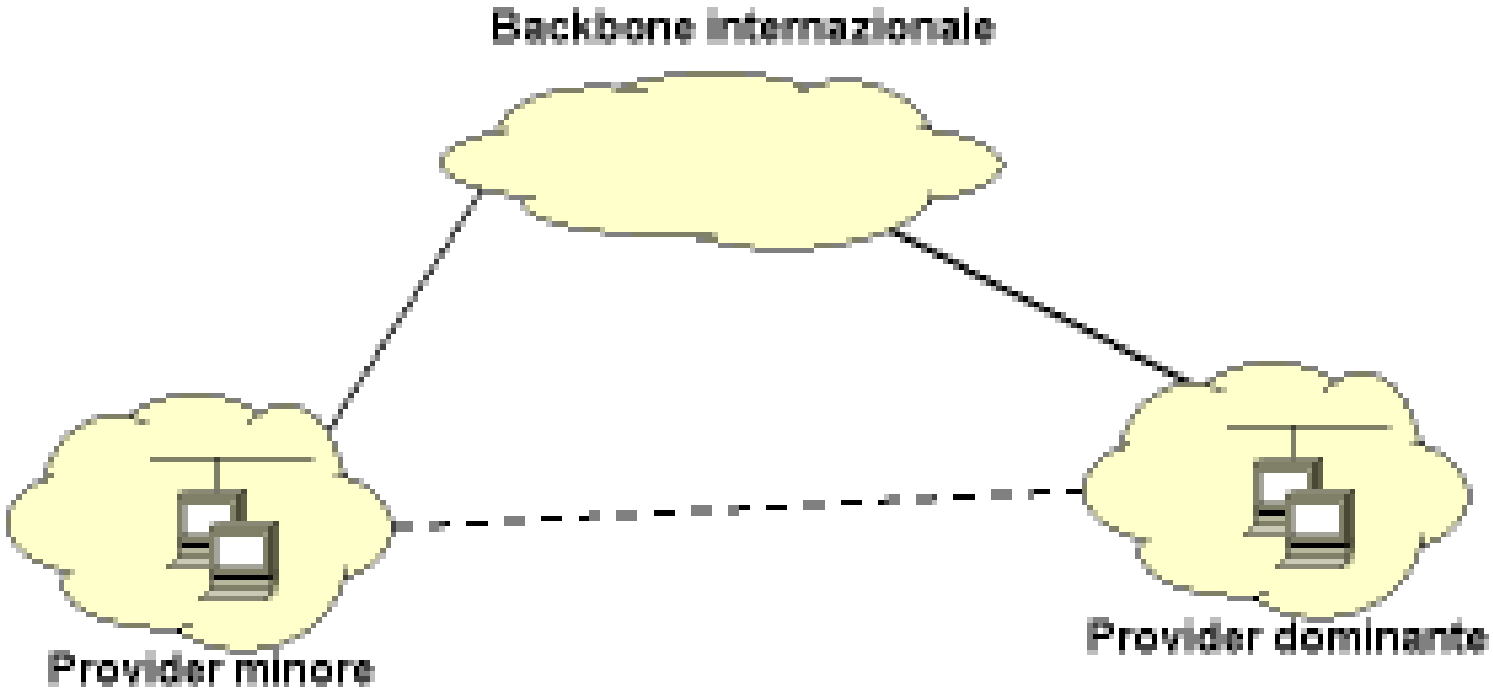
- Esigenze economiche - amministrative
  - Il dominio A può trarre vantaggio dal veicolare traffico destinato a B attraverso C
  - Ogni gestore deve avere degli accordi per poter immettere il traffico in un'altra rete, e questi accordi di peering possono essere costosi
- Esigenze di sicurezza
  - Può essere desiderabile evitare che i dati transitino in domini gestiti da autorità in cui non si ha fiducia



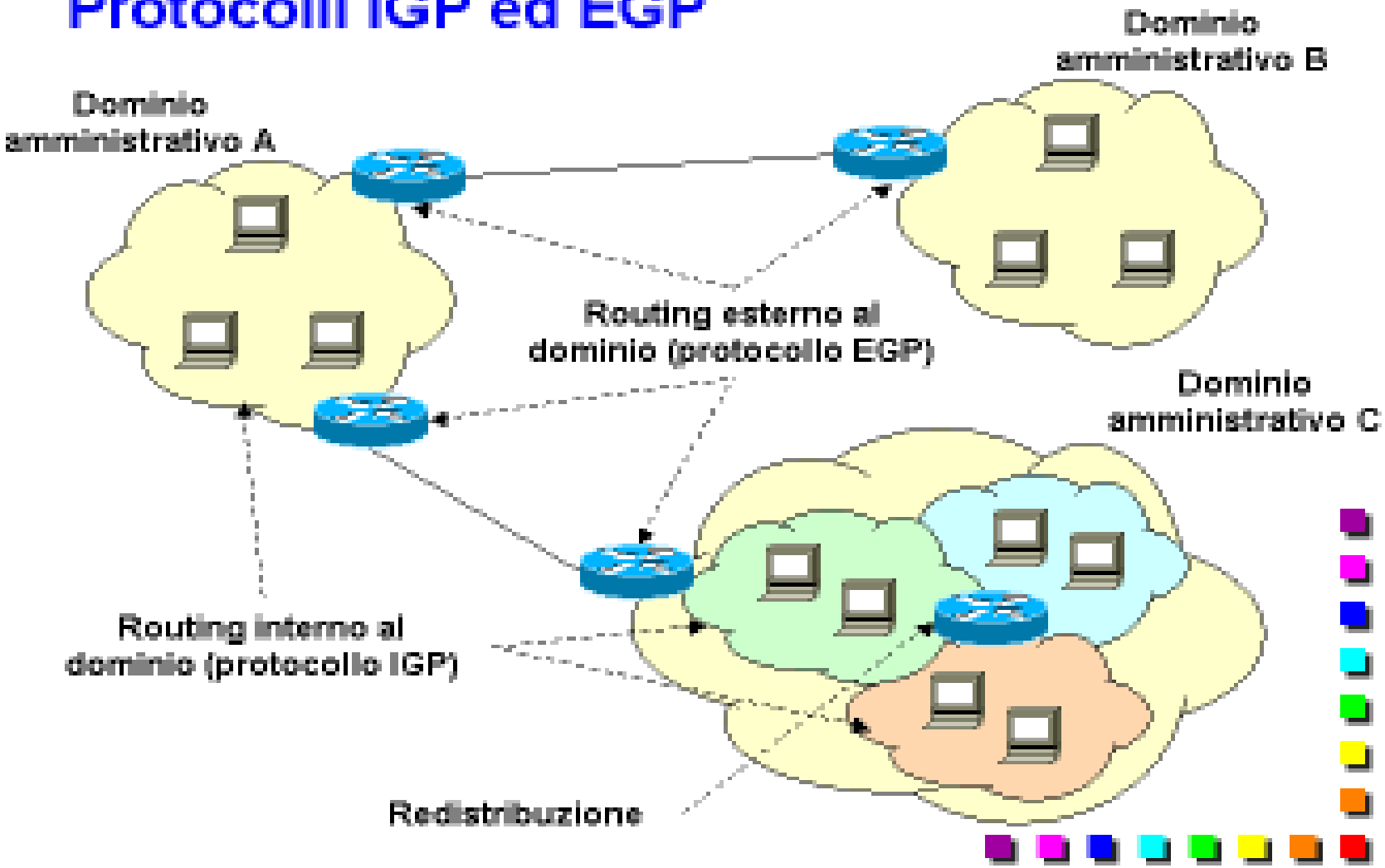
Mascheramento del routing



Altro esempio: provider dominante



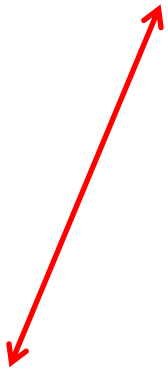
Protocolli IGP ed EGP



IGP (interior Gateway Protocol) e EGP (Exterior Gateway Protocol)

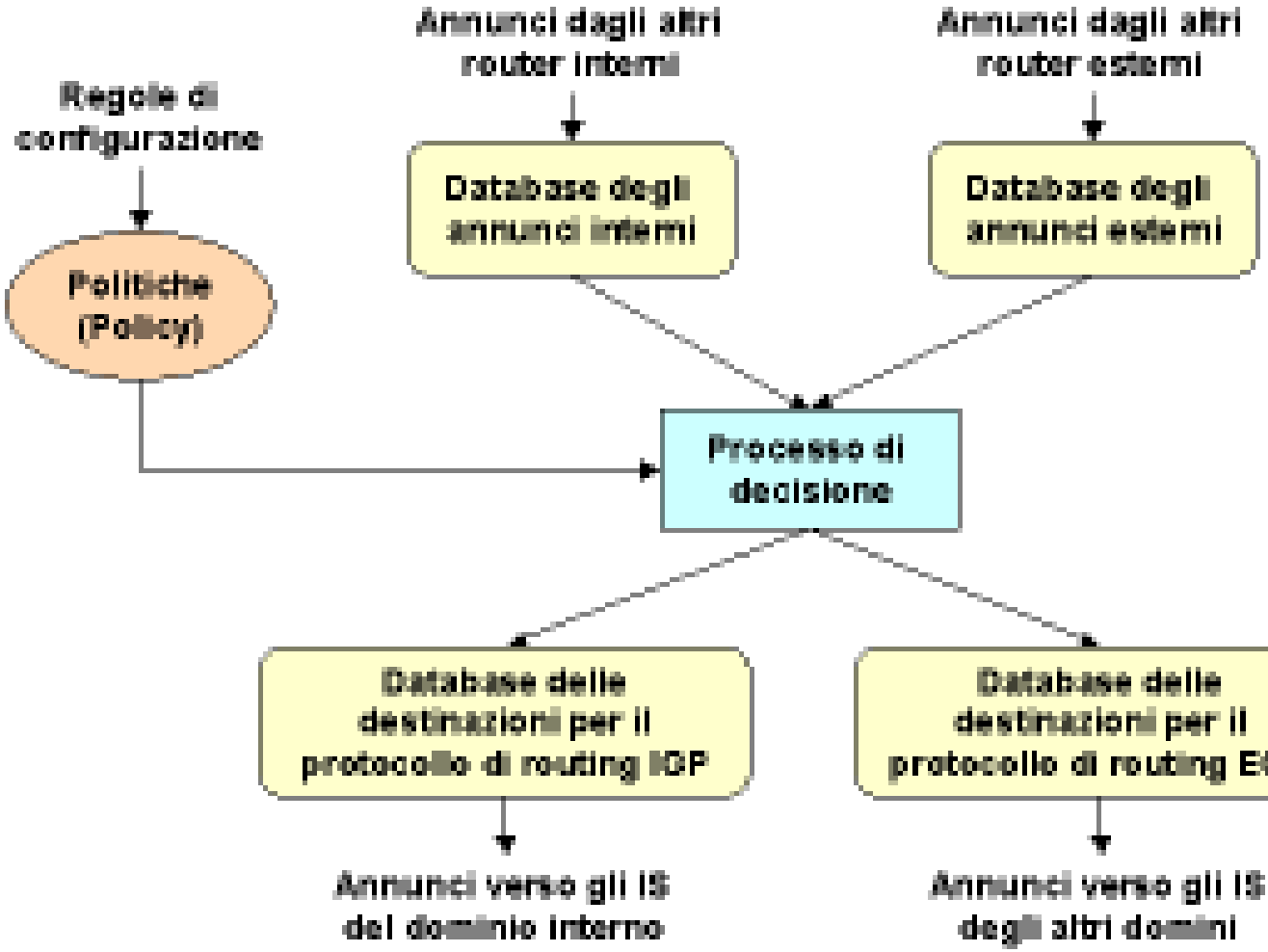
Soluzioni per il routing inter-dominio

- Protocolli ad hoc
  - Algoritmi gerarchici
    - IGP + EGP
  - Connettività + Policy
    - Far si che i miei dati non transitino nel dominio X
    - Dare la precedenza ai dati del dominio Y
    - ...
- Deve essere conosciuto il percorso esatto dei miei dati verso la destinazione
  - È necessario conoscere la lista dei domini (Autonomous System) attraversati
  - È necessario avere una completa conoscenza topologica della rete
    - Il Distance Vector non è appropriato



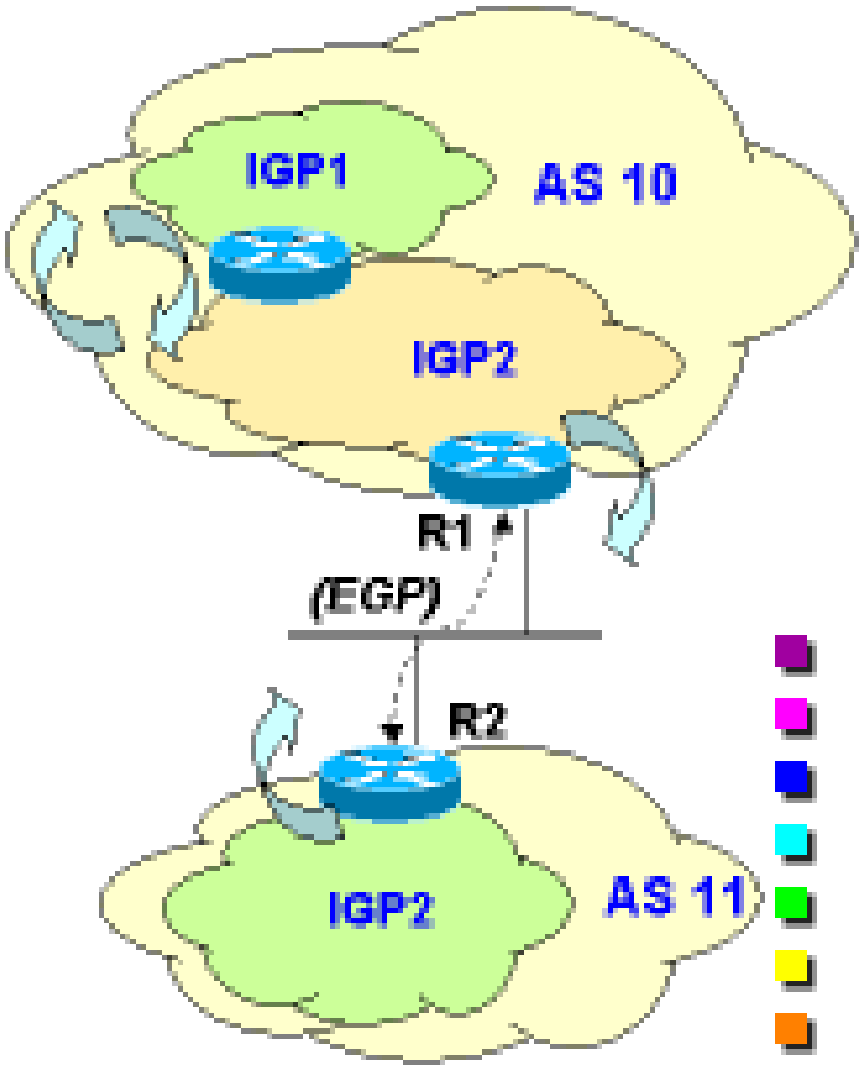
Internet utilizza il concetto di **Autonomous System** per determinare il percorso verso una certa destinazione: ogni annuncio di routing inter-dominio comprenderà quindi la coppia *destinazione - AS path* (elenco degli AS attraversati per giungere alla destinazione)

Decision process



Redistribuzione e domini amministrativi

- Redistribuzione
  - Normalmente usato un solo router "di frontiera"
  - Normalmente usata all'interno di un AS
- Routing esterno
  - Normalmente utilizzati 2 router "di frontiera"
  - Il protocollo di routing esterno (su R1 e R2): ha normalmente la redistribuzione attivata in un solo verso



Conclusioni

- Tecniche di forwarding
  - **Routing by Network address: la più diffusa**
  - **Label Swapping: nuovo interesse per la maggior facilità di commutazione**
- Algoritmi di routing
  - **Molto Link State su installazioni grosse, parecchio Distance Vector su quelle piccole**
    - Routing intra-dominio su Internet
  - **Routing statico, routing isolato: usati in casi specifici**
- Algoritmi gerarchici
  - **Necessari per la scalabilità**
- Algoritmi policy-based
  - **In uso per il routing inter-dominio; in alcuni casi utilizzati anche internamente al dominio**

Bibliografia

Leslie Lamport, Robert Shostak and Marshall Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems, 4(3):382-401, July 1982

Questo è un articolo classico nella letteratura degli algoritmi distribuiti. Definisce il problema dei Generali Bizantini (Byzantine General Problem) e dimostra che questo non è risolvibile per un numero di traditori pari o superiore ad 1/3 del totale dei generali.

Questo articolo analizza i modi con cui è possibile costruire dei sistemi affidabili anche a fronte di guasti (o informazioni intenzionalmente sbagliate).