POLITECNICO DI MILANO
Dipartimento di Elettronica e Informazione (DEI)

Routing nelle reti a pacchetto

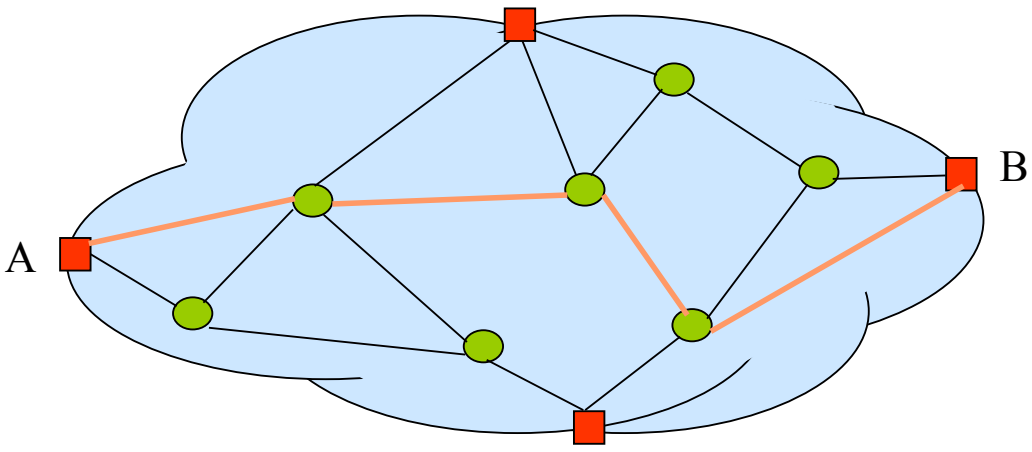
INFRASTRUTTURE E PROTOCOLLI PER INTERNET

Lucidi delle lezioni

Simone Redana
E-mail: redana@elet.polimi.it

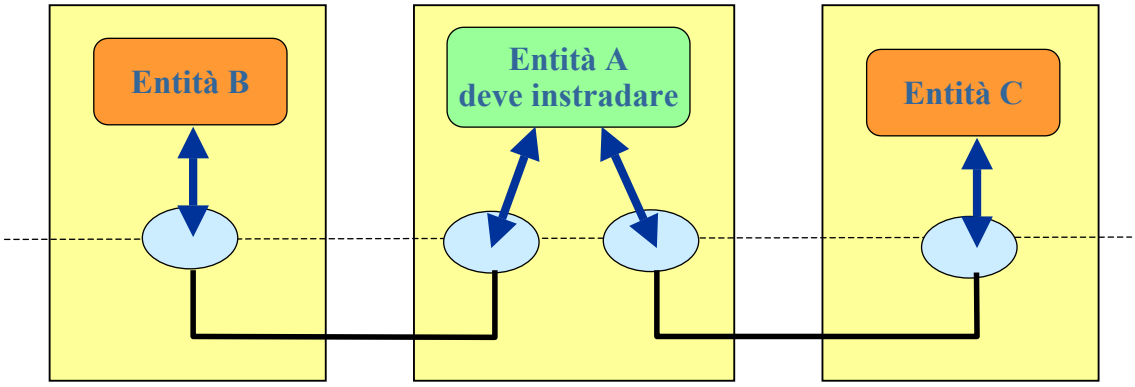
Routing

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



Routing

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



Routing

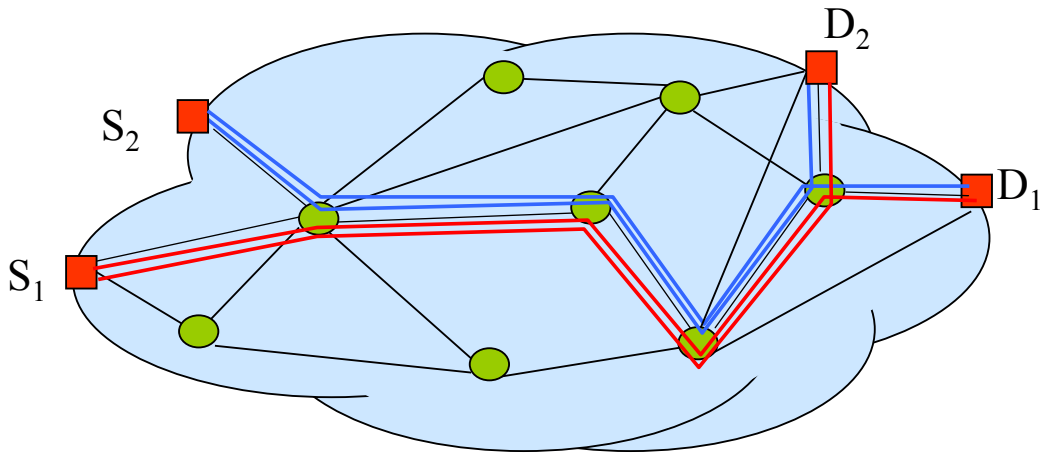
- La politica di routing è quella che definisce i criteri di scelta del cammino nella rete per i pacchetti che viaggiano tra un nodo di ingresso ed uno di uscita
- e dunque quella che costruisce le tabelle di routing che vengono usate dai nodi per effettuare il forwarding
- il tipo di rete (datagram,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

Routing e capacità

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

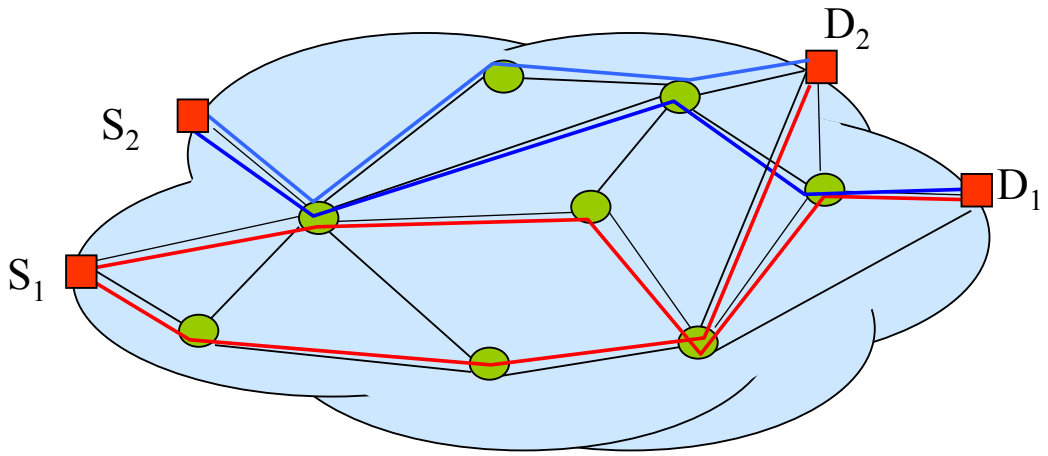
Routing e capacità

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



Routing e capacità

- Ad esempio:
  - se invece si usano molti cammini ripartendo il carico il massimo traffico sarà elevato



Cammini minimi

- Il criterio più comunemente usato nelle reti dati è basato sul calcolo dei cammini minimi sul grafo nel quale ad ogni arco è associato un peso
- infatti, molte politiche di instradamento di Internet, come si vedrà in seguito, instradano tutto il flusso tra una sorgente ed una destinazione sul cammino minimo calcolato utilizzando opportuni pesi degli archi
- dato un grafo e dei pesi associati agli archi il calcolo del cammino minimo si può ottenere con algoritmi di complessità polinomiale nel numero di nodi.

Richiami sui Grafi

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

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

Problema del cammino minimo

- dato un digrafo G(N,A) e due nodi i e j, trovare il cammino di lunghezza minima tra tutti quelli che consentono di andare i a j
- il problema è di complessità polinomiale
- proprietà: se il nodo k è attraversato dal cammino minimo da i a j, il sotto-cammino fino a k è anch'esso minimo

Algoritmi di Bellman-Ford

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

Algoritmi di Bellman-Ford

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

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

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

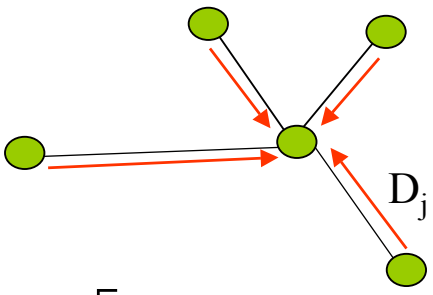
- Iterazioni:

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

- l'algoritmo termina in N-1 passi

Algoritmi di Bellman-Ford in forma distribuita

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

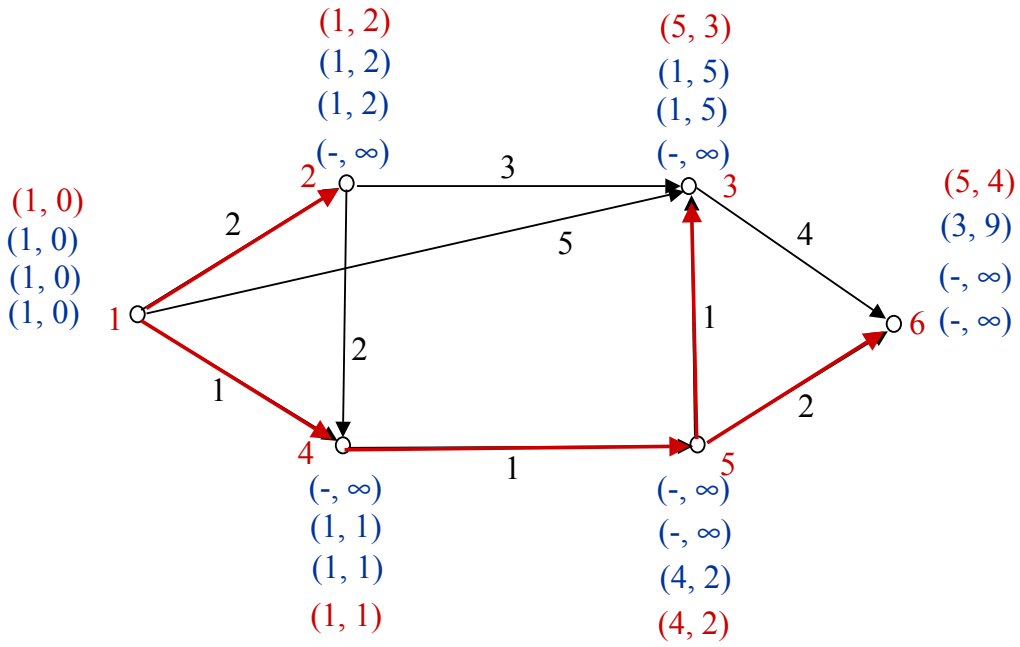


$$D_i := \min\left[D_i , \min_j \left(D_j + d_{ji} \right) \right]$$

Algoritmi di Bellman-Ford in pratica

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

Esempio: Bellman-Ford



Algoritmi di Dijkstra

- Ipotesi:
  - archi con pesi positivi
- Scopo:
  - trovare il cammini tra un nodo 1 (sorgente) e tutti gli altri nodi
- Valori iniziali:

$$P = \{1\},$$

$$D_1 = 0, \quad D_j^{(0)} = d_{1j} \qquad \forall j \neq 1$$

  - si assume $d_{ij}=\infty$ se l'arco tra i e j non esiste

Algoritmi di Dijkstra

- Iterazioni:

1. trova $i \in (N\text{-}P)$ tale che:

$$D_i = \min_{j \in (N-P)} D_j$$

e poni

$$P := P \cup \{i\}. \text{ Se } P = N, \text{ allora STOP.}$$

2. per tutti $j \in (N\text{-}P)$ poni:

$$D_j = \min\left[D_j, \min_k \left(D_k + d_{kj}\right)\right]$$

3. vai allo step 1.

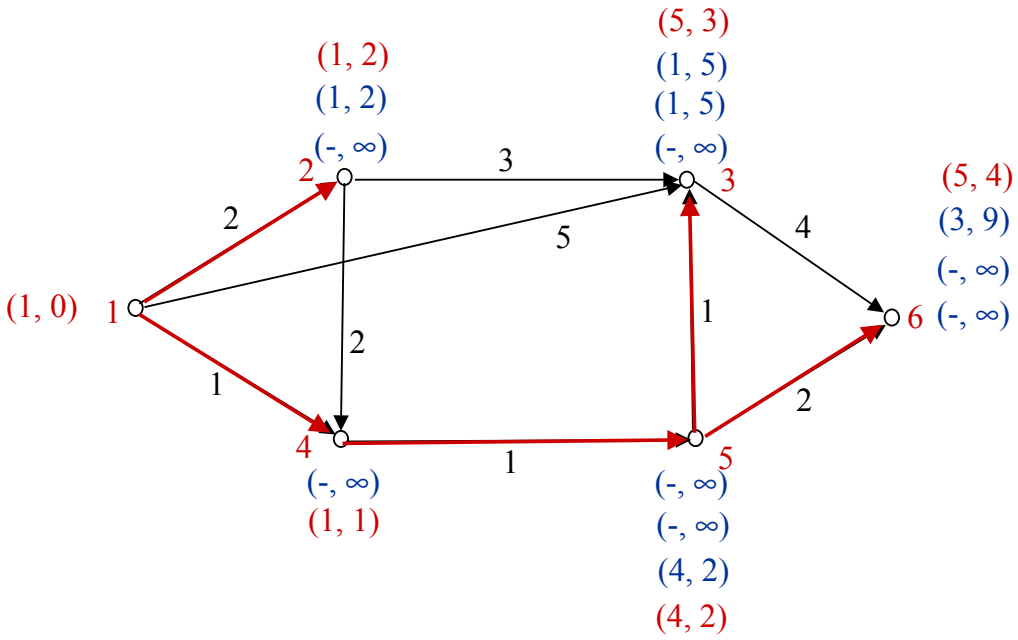
Algoritmi: complessità

- L'algoritmo di Bellman-Ford ha una complessità:
  - N-1 iterazioni
  - N-1 nodi per iterazione
  - N-1 confronti per nodo
    - Complessità: $O(N^3)$
- L'algoritmo di Dijkstra ha una complessità:
  - N-1 iterazioni
  - in media N operazioni per iterazioni
    - Complessità: $O(N^2)$
- L'algoritmo di Dijkstra è in generale più conveniente

Algoritmi di Dijkstra in pratica

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

Esempio: Dijkstra



POLITECNICO DI MILANO
Dipartimento di Elettronica e Informazione (DEI)

Protocolli di routing IP

INFRASTRUTTURE E PROTOCOLLI PER INTERNET

Lucidi delle lezioni

Simone Redana
E-mail: redana@elet.polimi.it

Protocolli di Routing

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

Tabelle di Routing IP

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

Routing IP

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

Protocolli di Routing

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

Distance Vector

- l'informazione sulla raggiungibilità è costituita dal Distance Vector : [indirizzo, distanza]
  - la distanza è una stima che il nodo possiede
- Il DV è inviato ai soli nodi adiacenti
- la stima delle distanze avviene tramite Bellman-Ford distribuito

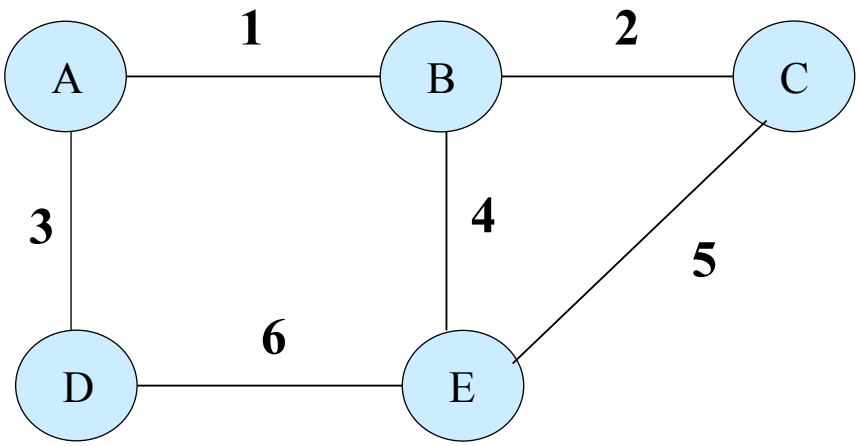
Distance Vector 2

- Ogni nodo invia il DV
  - periodicamente
  - se il risultato di un ricalcolo differisce dal precedente
- Ogni nodo esegue il ricalcolo delle distanze se
  - riceve un DV diverso da quello memorizzato in precedenza
  - cade/nasce una linea attiva a cui è connesso

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

Esempio: Distance Vector

- Consideriamo la seguente rete:



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

Esempio: Distance Vector

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

Da A verso	Link	Costo
A	locale	0

Esempio: Distance Vector

- da questa tabella il nodo A estrae il Distance Vector
  A=0
  e lo trasmette a tutti i vicini, cioè su tutti i link locali
- B e D ricevono l'informazione e allargano le loro conoscenze locali,
- il nodo B, dopo aver ricevuto il Distance Vector, aggiorna la distanza aggiungendo il costo del link locale trasformando il messaggio in A=1, lo confronta con le informazioni nella sua tabella di routing e vede che il nodo A non è conosciuto

Da B verso	Link	Costo
B	locale	0
A	1	1

Esempio: Distance Vector

- il nodo B prepara il proprio DV

  B=0, A=1

  e lo trasmette su tutti i link locali
- nel frattempo anche il nodo D ha ricevuto il messaggio da A, aggiornato la sua tabella di routing e inviato il DV

  D=0, A=1
- il messaggio da B viene ricevuto da A,C ed E mentre quello da D è ricevuto da A ed E, supponiamo che A abbia ricevuto prima il messaggio da B quindi i due DV aggiornati col costo del link locale sono nell'ordine

  B=1, A=2

  D=1, A=2

Esempio: Distance Vector

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

- il nodo C riceve sul link 2 il DV

  B=0, A=1

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

Esempio: Distance Vector

- il nodo E riceve sul link 4 il DV

  B=0, A=1

  e sul link 6 il DV

  D=0, A=1

  aggiorna la sua tabella di routing
  - la distanza verso il nodo A utilizzando i link 4 e 6 è la stessa

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

Esempio: Distance Vector

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

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

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

  nodo E: E=0, B=1, A=2, D=1
- i nodi B,D ed E aggiornano le proprie tabelle

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

Esempio: Distance Vector

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

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

- i nodi B,D ed E trametttono i nuovi DV sui link locali
  - nodo B: B=0, A=1, D=2, C=1, E=1
  - nodo D: D=0, A=1, B=2, E=1
  - nodo E: E=0, B=1, A=2, D=1, C=1
- vengono ricevuti dai nodi A,C ed D che aggiornano le tabelle

Esempio: Distance Vector

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

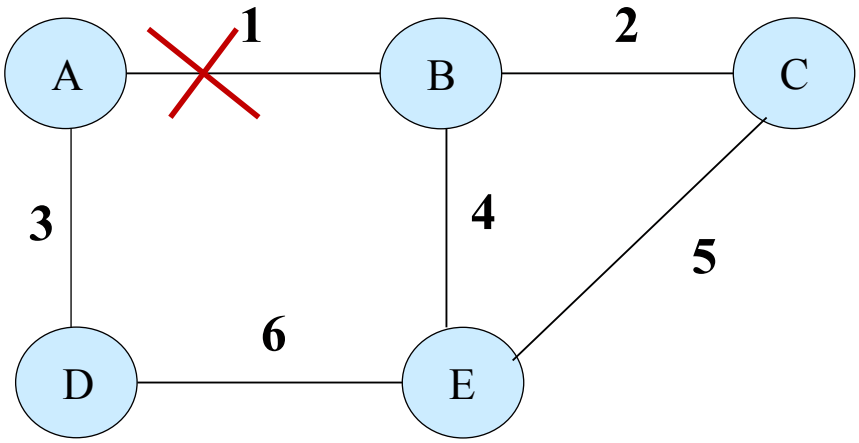
Da C verso	Link	Costo
C	local	0
B	2	1
A	2	2
E	5	1
D	5	2

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

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

Distance Vector: rottura del link 1

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



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

Distance Vector: rottura del link 1

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

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

- trasmettono i nuovi DV

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

  nodo B: B=0, A=inf, D=inf, C=1, E=1
- il messaggio trasmesso da A viene ricevuto da D che confronta gli elementi con quelli presenti nella sua tabella di routing
- tutti i costi sono maggiori o uguali a quelli presenti nella tabella, ma siccome il link da cui riceve il messaggio (link 3) è quello che utilizza per raggiungere il nodo B aggiorna la tabella

Distance Vector: rottura del link 1

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

- i nodi C ed E aggiornano le tabelle

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

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

Distance Vector: rottura del link 1

- i nodi D, C ed E trasmettono il loro DV
  - nodo D: D=0, A=1, B=inf, E=1, C=2
  - nodo C: C=0, B=1, A=inf, E=1, D=2
  - nodo E: E=0, B=1, A=inf, D=1, C=1
- questi messaggi aggiornano le tabelle di routing dei nodi A,B,D ed E

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

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

Distance Vector: rottura del link 1

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

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

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

Distance Vector: rottura del link 1

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

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

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

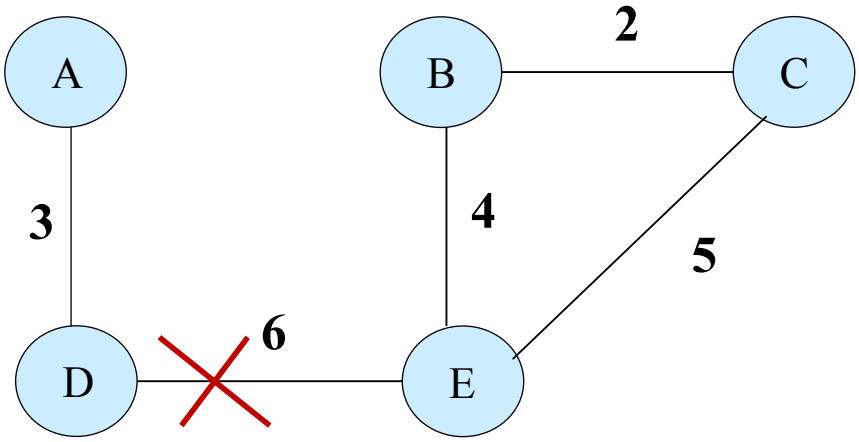
- l'algoritmo è arrivato a convergenza

Distance Vector: caratteristiche

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

Distance Vector: counting to infinity

- Supponiamo che si rompa anche il link 6



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

Distance Vector: counting to infinity

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

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

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

Distance Vector: counting to infinity

- se invece il nodo A trasmette il suo DV
  nodo A: A=0, B=3, D=1, C=3, E=2
  il nodo D aggiorna la sua tabella

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

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

Distance Vector: counting to infinity

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

E la convergenza verso uno stato stabile è lenta

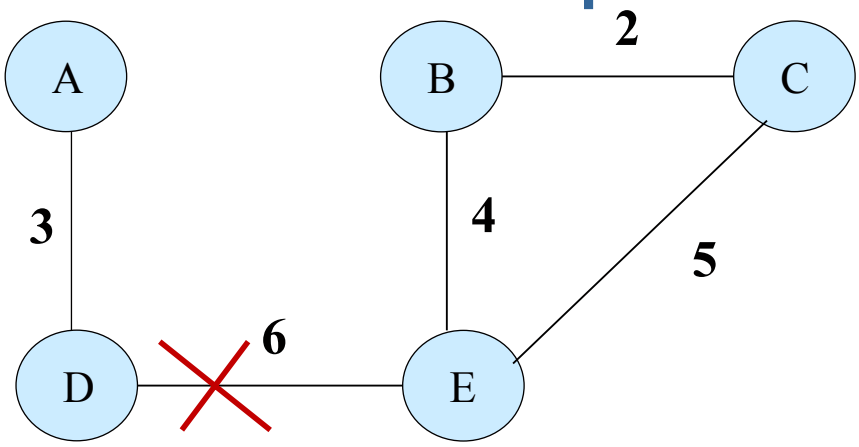
Counting to infinity: rimedi

- Hop Count Limit:
  - I valori di costo superiori ad una soglia sono posti ad infinito
    - svantaggio: tutte le destinazioni ad una distanza maggiore della soglia sono considerate irraggiungibili
- Split-Horizon:
  - se il nodo A manda a D i pacchetti destinati al nodo X, non ha senso che D provi a raggiungere X attraverso A

Distance Vector: Split Horizon

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

Distance Vector: Split Horizon



- quando il link 6 si rompe le tabelle di routing dei nodi B,C ed D contengono

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

Distance Vector: Split Horizon

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

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

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

Distance Vector: Split Horizon

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

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

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

Counting to infinity: rimedi 2

- Hold Down
  - dopo un tempo $T_{invalid}$ che non si riceve il DV di una destinazione (route) dal nodo del primo hop, la si dichiara non piu' valida: non viene annunciata nei DV e non vengono considerati validi per essa i DV ricevuti da altri nodi
  - dopo un tempo T_{flush} la route e' cancellata
  - Il tempo tra $T_{invalid}$ e T_{flush} deve essere tarato in modo che l'informazione relativa ad un cambiamento (guasto) si propaghi nella rete

Counting to infinity: rimedi 3

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

Link State

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

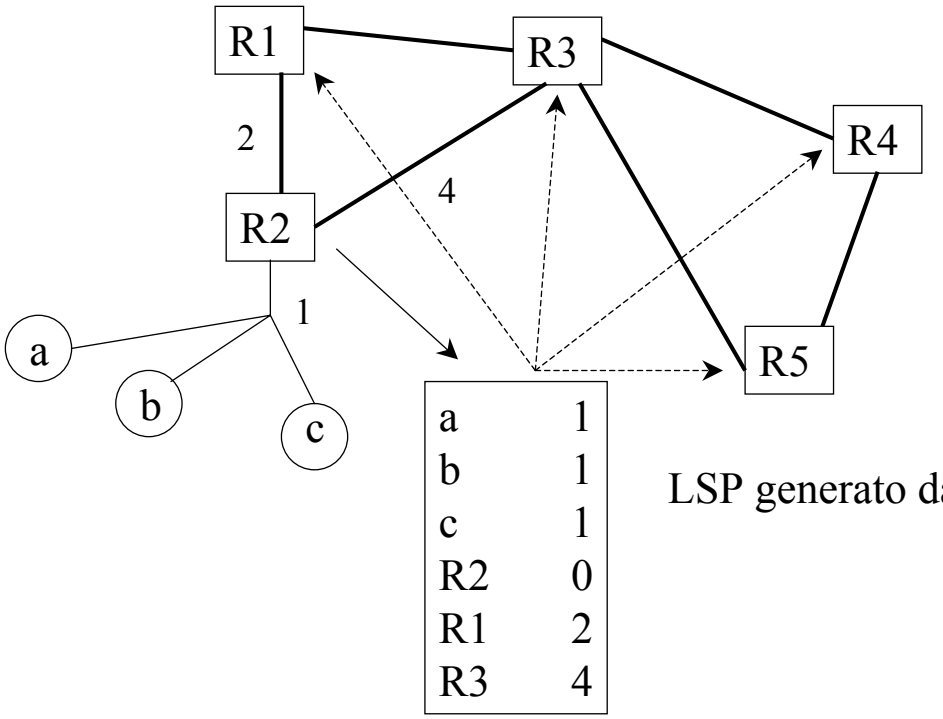
Link State 2

- Vantaggi:
  - più flessibile in quanto ogni nodo ha una mappa completa della rete (routing ottimale)
  - non e' necessario inviare l'informazione (LSP) periodicamente ma solo dopo un cambiamento
  - tutti i nodi vengono subito informati dei cambiamenti (in particolare topologici)

Link State 3

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

Link State: esempio



a	1
b	1
c	1
R2	0
R1	2
R3	4

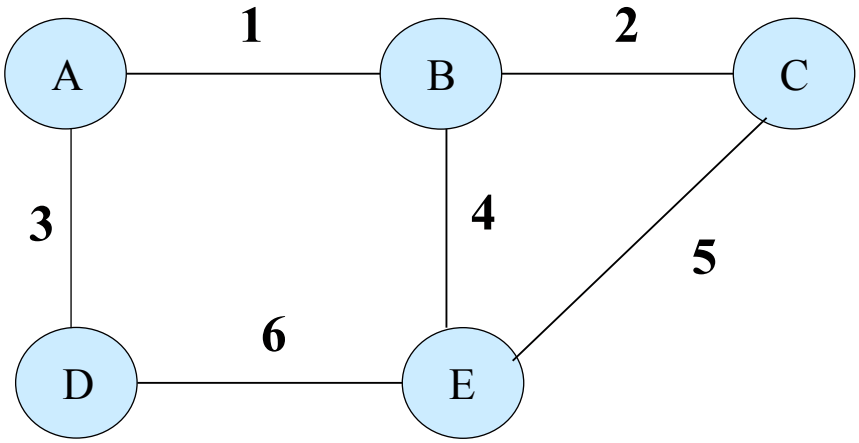
LSP generato da R2

Flooding

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

Esempio: Link State

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



Esempio: Link State

- la rete è rappresentata dal database

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

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

All'arrivo di un LSP

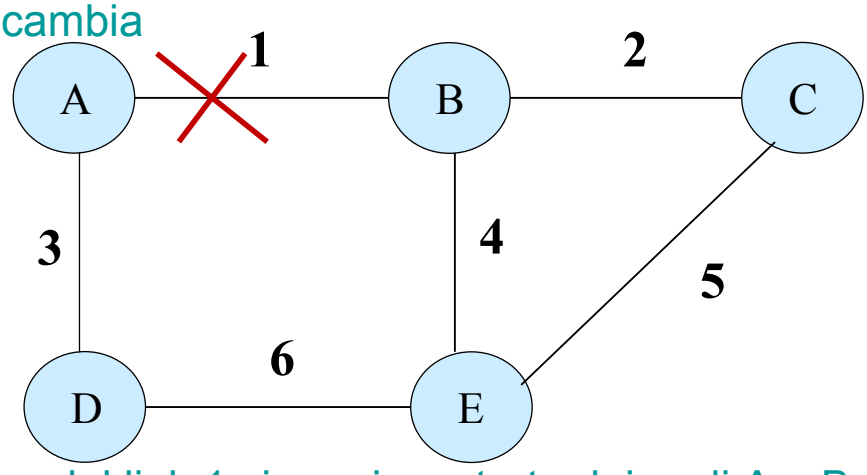
- Se il LSP non e' mai stato ricevuto, o il SN è superiore a quello memorizzato precedentemente:
  - memorizza il LSP
  - lo ritrasmette in flooding sulle uscite
- Se il LSP ha lo stesso SN di quello memorizzato
  - non fa nulla
- Se il LSP e' piu' vecchio di quello memorizzato
  - trasmette quello piu'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

Calcolo tabelle di routing

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

Esempio: Link State

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



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

  nodo A: Da A, Verso B, Link 1, Costo=inf, Number=2

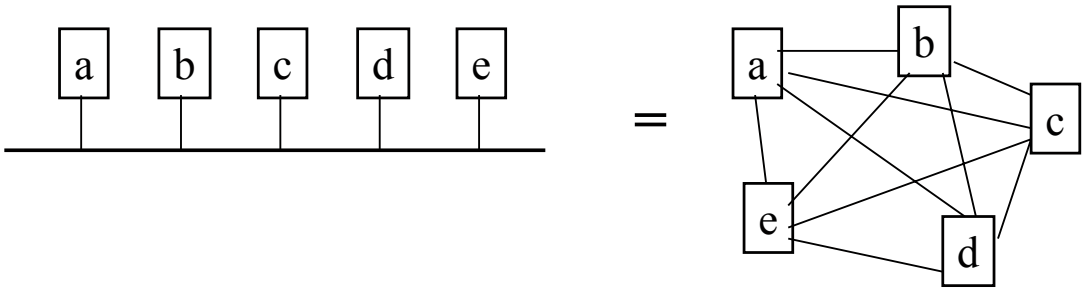
  nodo B: Da B, Verso A, link 1, Costo= inf, Number=2

Esempio: Link State

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

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

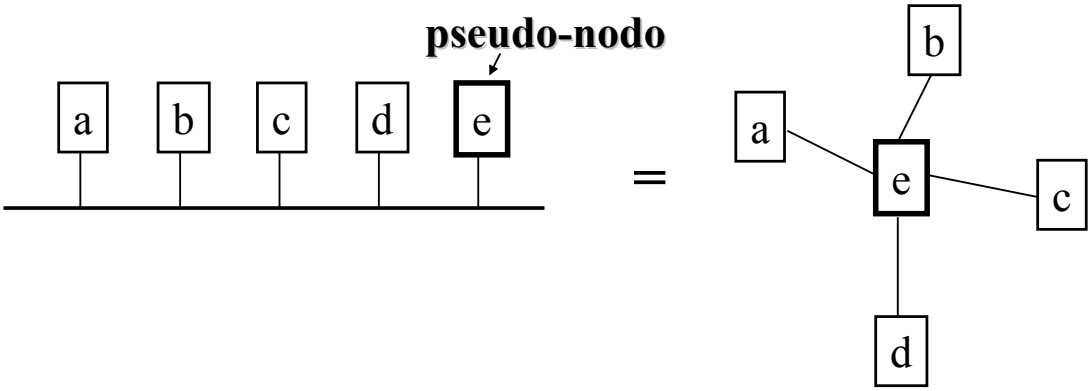
Nelle reti broadcast (LAN)



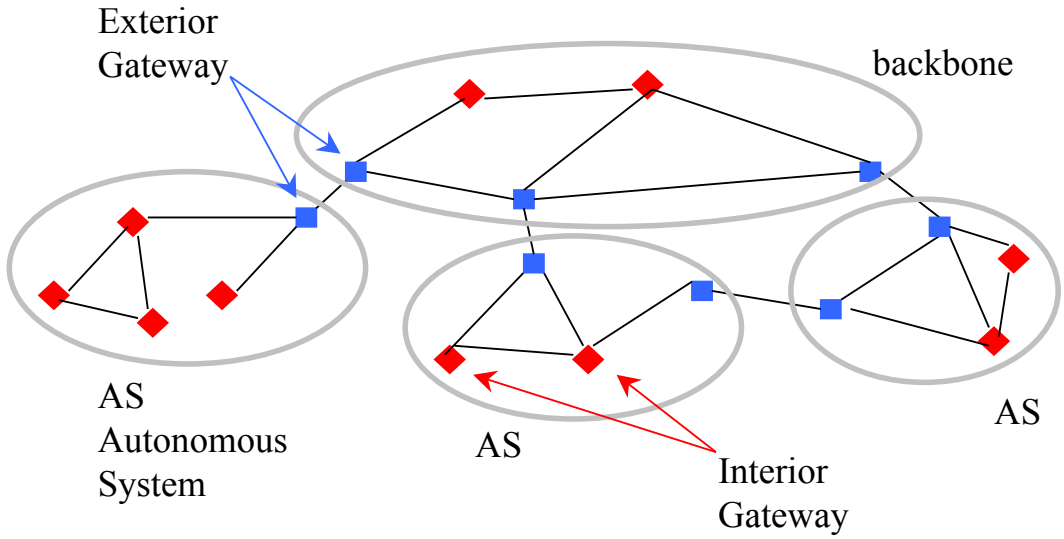
- Ogni nodo sulla LAN risulta logicamente connesso a ciascun altro nodo
- La complessità della rete viene inutilmente resa elevata a causa della natura broadcast della rete utilizzata

Nelle reti broadcast (LAN)

- *pseudo-nodo*: nodo fittizio a cui sono connessi tutti gli altri
- si elegge lo pseudo-nodo tra quelli della LAN
- si passa da topologia *mesh* a topologia *star*

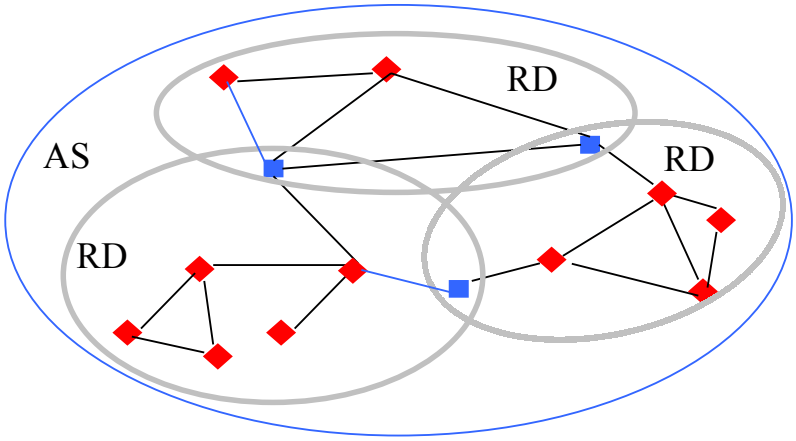


Routing in Internet



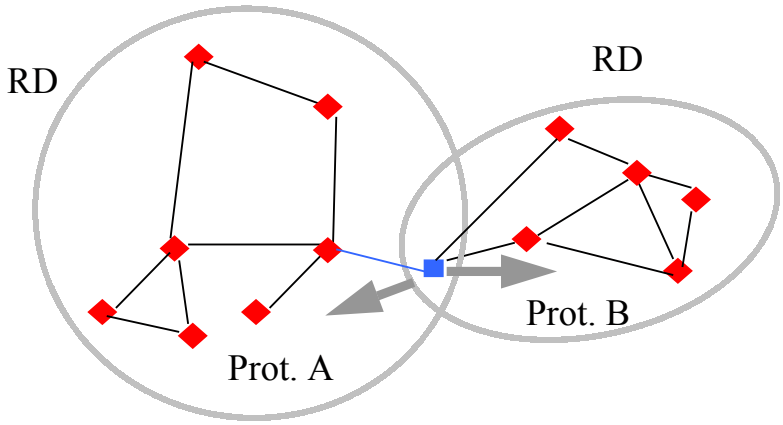
- Autonomous System: porzione di rete gestite da autorità diverse
- Protocollo di Routing: protocollo per lo scambio di informazioni di routing tra gateway
- EGP - Exterior Gateway Protocol
- IGP - Interior Gateway Protocol

Domini di Routing



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

Ridistribuzione



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

Protocolli di Routing usati

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

Tabelle di Routing

All'arrivo di un pacchetto con indirizzo:

172.10.27.105

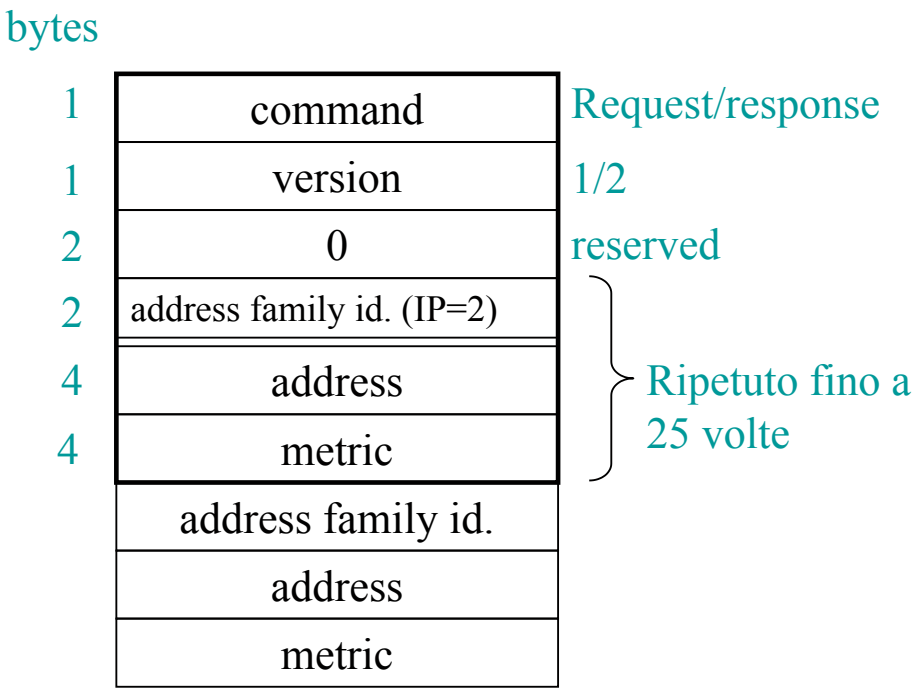
Indirizzo di destinazion	Netmask	Next hop
131.175.21.0	255.255.255.0	129.15.13.254
131.175.15.0	255.255.255.0	129.15.13.254
172.10.0.0	255.255.0.0	129.15.13.254
151.17.0.0	255.255.0.0	129.15.70.254
160.5.21.0	255.255.0.0	129.15.13.254
172.10.27.0	255.255.255.0	129.15.70.254
0.0.0.0	0.0.0.0	129.15.58.254

Per ogni riga della tabella:
1) - AND logico con netmask
2) - confronto con indirizzo destinazione
3) - tra route con esito positivo scelta quella con netmask (prefisso) piu' lunga

RIP v1

- RFC 1058
- Protocollo IGP
- Distance Vector
- Metrica: numero di hop
- è il più diffuso e semplice protocollo di routing dinamico

RIP v1



RIP v1: Timer

- routing update timer (default 30 s)
  - intervallo di tempo per l'invio dei DV
- route invalid timer (default 90 s)
  - intervallo dopo il quale, se non si ricevono annunci dalla stessa interfaccia, una route è dichiarata non valida
- route flush timer (default 270 s)
  - intervallo di tempo dopo cui una route è cancellata (se arrivano nuovi DV da altre interfacce sono accettati)

RIP v2

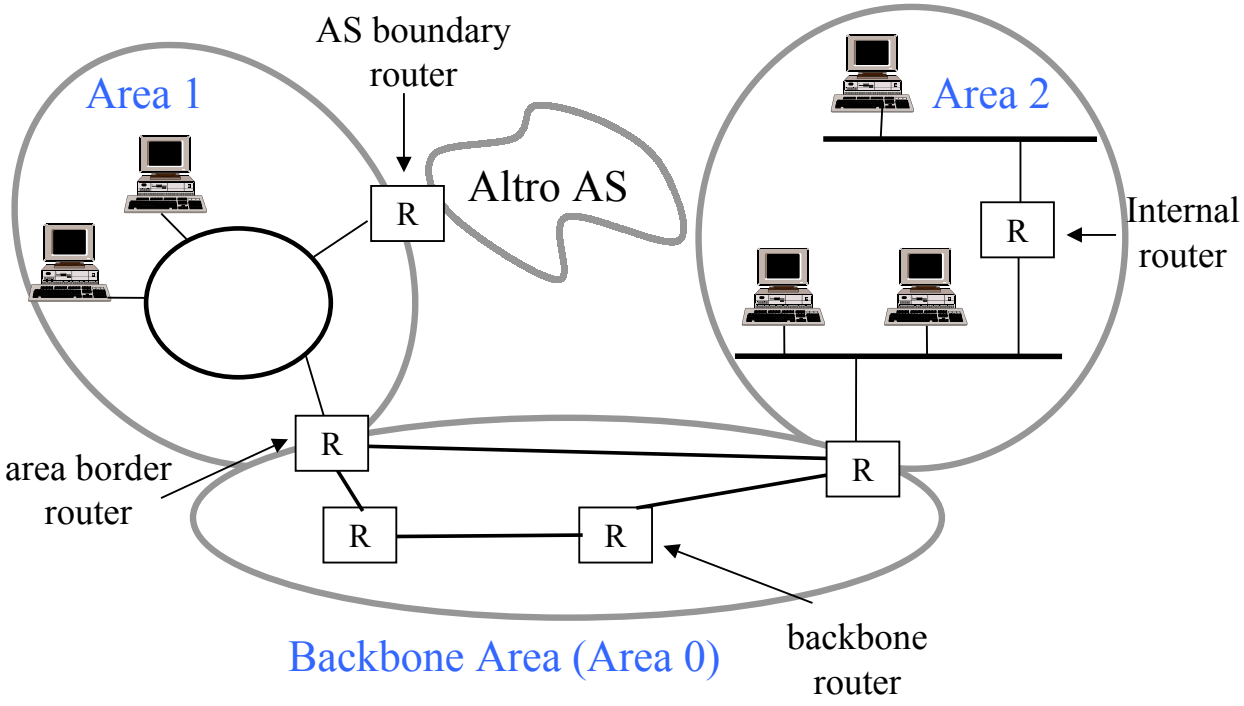
- RFC 1723
- permette di inserire anche le netmask

4	address
4	netmask
4	metric

OSPF

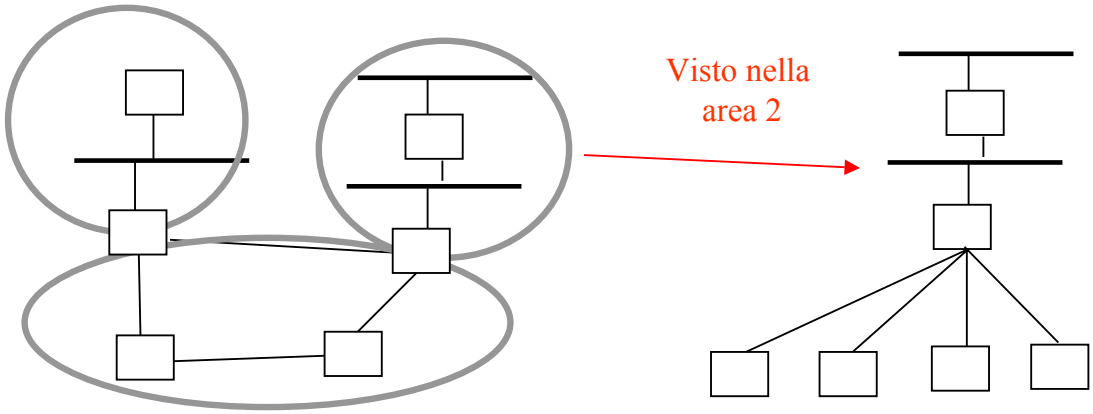
- RFC 1247, 1583
- Link state
- supporta routing gerarchico
- utilizzo di protocollo di Hello
- identificativo univoco router (es. uno dei suoi IP address)
- LSA (link state advertisement)

OSPF: gerarchia e classificazione dei router



OSPF

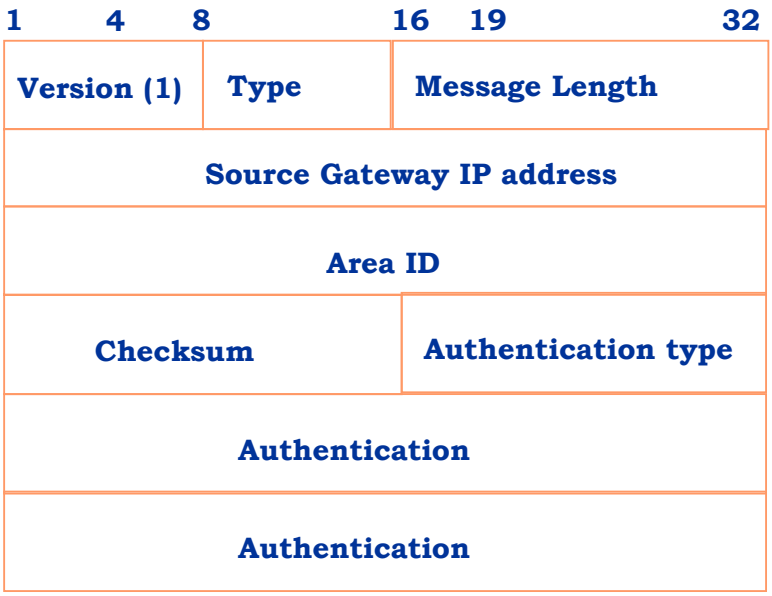
- Gli area border router diffondono in ciascuna area un riassunto delle informazioni raccolte nell'altra
  - contaminazione distance vector



OSPF: tipi di LSA

- Tipo 1: router links advertisement
  - si propaga all'interno della stessa area (classico LSP)
- Tipo 2: network links advertisement
  - generato dallo pseudo-nodo (DR) di una LAN
- Tipo 3: network summary link advertisement
  - generato dagli area border router per riassumere le informazioni di un area in un'altra
- Tipo 4: boundary routers summary link advertisement
  - generato dagli area border router, indica la presenza di un AS boundary router nell'area e il relativo costo
- Tipo 5: AS external link advertisement
  - generato da un AS boundary router e propagato a tutti i router di tutte le aree, contiene destinazioni esterne e relativi costi

OSFP: Open Shortest Path First



OSFP: Open Shortest Path First

- L'header dei messaggi OSPF è mostrato in figura
- Il campo type specifica il tipo di messaggio OSPF che puo' essere
  - HELLO
  - DATABASE DESCRIPTION
  - LINK STATUS REQUEST
  - LINK STATUS UPDATE
  - LINK STATUS ACKNOWLEDGE
- Il campo Source gateway IP address è l'indirizzo IP del mittente e l'Area ID codifica l'area di appartenenza
- Il campo Authentication type puo' essere 0 (NO) o 1 (SI)

OSFP: Open Shortest Path First

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