Università di Genova
Facoltà di Ingegneria

Telematica 3
2. Protocolli di Instradamento

Prof. Raffaele Bolla

dist

Instradamento

- Requisiti
  - Minimizzare lo spazio occupato dalle RT per:
    - » Velocizzare la commutazione
    - » Semplificare i router (meno cari)
    - » Ridurre l'informazione necessaria all'aggiornamento
  - Minimizzare il traffico di controllo
  - Essere robusto, ossia evitare:
    - » Cicli
    - » Buchi neri
    - » Oscillazioni
  - Ottimizzare i percorsi (dal punto di vista della distanza, del ritardo, del costo economico, …)

2.2

Instradamento
Alternative

- Centralizzato o **distribuito** (o isolato)
- Basato sulla sorgente o ***"hop-by-hop"***
- **Deterministico** o stocastico
- **Singolo percorso** o multi-percorso
- Dipendente dallo stato (**dinamico**) o indipendente dallo stato (**statico**)
- ***Distance Vector*** o ***Link State***.

2.3

Algoritmi di instradamento
Shortest path

- Il problema del "**percorso minimo**" o ***shortest path*** è quello di trovare il percorso p fra i e j tale che D sia minimo.
- Esistono diversi metodi per risolvere questo problema, uno di questi prende il nome di
  Algoritmo di Bellman-Ford.
- Tale algoritmo fissata una destinazione, trova il percorso minimo da ogni nodo a tale destinazione nell'ipotesi non ci siano distanze negative ($d_{ij} \geq 0$).

2.4

Algoritmi di instradamento
Shortest path - Bellman Ford

- Definendo
  - 1 come il nodo destinazione
  - $d_{ij} = \infty$ se (i, j) non è un arco (i nodi i e j non sono direttamente connessi),
  - D_i^h come la lunghezza del percorso più corto fra il nodo i ed il nodo 1, contenente al massimo h archi
  - $D_i^h = \infty$ se tale percorso non esiste
  - $D_1^h = 0$, $\forall$ h, per convenzione
  - $D_i^0 = \infty$ per tutti gli $i \neq 1$
- l'iterazione dell'algoritmo di **Bellman-Ford** è
  $D_i^{h+1} = \min_j \{d_{ij} + D_j^h\}$ per ogni $i \neq 1$
- L'algoritmo ha termine quando $D_i^{h+1} = D_i^h$ $\forall i$

2.5

Algoritmi di instradamento
Shortest path Spanning Tree

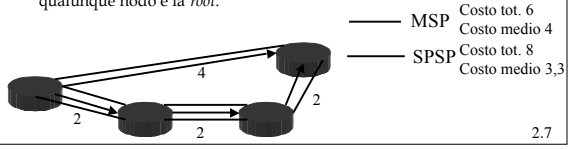
- L'applicazione equazione di Bellman seleziona un solo arco uscente da ogni nodo i (eccetto la dest. 1), cioè quello per cui la somma $d_{ij} + D_j^h$ è minima.
- Quindi in sostanza gli archi scelti dall'algoritmo ed i nodi formano uno *spanning tree* perché:
  - Comprendono tutti i nodi per definizione
  - non possono formare cicli (essendo le lunghezze positive)
- Tale spanning tree viene chiamato ***Shortest Path Spanning Tree*** (SPST) ed il nodo destinazione è chiamato ***root*** (radice).

2.6

Algoritmi di instradamento

Shortest path Spanning Tree

- Un grafo non orientato può essere rappresentato come un grafo orientato a cui ad ogni arco non orientato corrispondono due archi, uno per direzione, con eguale peso.
- In generale il MSP (Minimum weight Spanning Tree) e il SPST sono diversi:
  - Il MSP minimizza il costo di un broadcasting;
  - Il SPST invece minimizza il costo delle comunicazioni fra un qualunque nodo e la *root*.



2.7

Instradamento

Distance vector

- L'algoritmo di Bellman Ford può essere realizzato in modo distribuito ed in questo caso viene chiamato ***Distance Vector Routing***.
- Ogni nodo (router) conosce l'identità di tutti nodi della rete e i nodi a lui direttamente connessi (vicini).
- Ogni nodo mantiene un *Distance Vector*, ossia una lista di coppie (destinazione, costo) per tutte le possibili destinazioni.
- Il costo è la somma stimata dei costi sui singoli link sul percorso "più corto" (*shortest path*) verso quella destinazione.
- Ogni nodo inizializza i costi relativi a destinazioni "lontane" ad una valore alto, convenzionalmente indicato infinito.

2.8

Instradamento

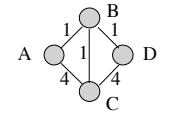
Distance vector

- Periodicamente ogni nodo invia ai propri vicini il proprio DV.
- Quando un *router* A riceve un DV da B (suo vicino), verifica quali sarebbero i costi per le varie destinazioni usando come transito B; per le destinazioni in cui tali costi risultano minori di quelli attuali, sostituisce il costo vecchio con quello calcolato e lo stesso fa con il next-hop nella RT.

2.9

Instradamento

Distance vector



Situazione Iniziale

	A	B	C	D
A	0	1	4	∞
B	1	0	1	1
C	4	1	0	4
D	∞	1	4	0

1	Costo per raggiungere B da A
+	
1 0 1 1	Costi da B agli altri
=	
2 1 2 2	Costi passando per B
0 1 4 ∞	Costi attuali in A
min	
0 1 2 2	Nuovo DV
- - B B	*Next hop*

2.10

Instradamento

Distance vector

- Questo procedimento corrisponde a realizzare in modo distribuito e asincrono l'algoritmo di Bellman-Ford, perché ogni nodo *i* esegue l'iterazione

$$D_i \leftarrow \min_{j \in N(i)} \{d_{ij} + D_j\}$$

(dove $N(i)$ è l'insieme dei nodi adiacenti ad *i*), usando le stime D_j più recenti ricevute dai vicini e trasmettendo D_i ai propri vicini.

- Si dimostra che non è necessaria una inizializzazione con particolari valori di D_j.

2.11

Instradamento

Distance vector

Sia A il numero di archi e N quello dei nodi

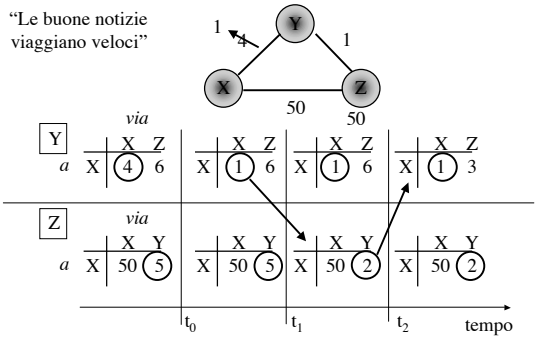
- Nel caso peggiore, l'algoritmo di Bellman-Ford centralizzato compie N-1 iterazioni, ciascuna su N-1 nodi, con al più N-1 alternative per nodo, il che porterebbe a complessità $\mathbf{O(N^3)}$.
- Si può mostrare che la complessità è **O(mA)**, con m numero di iterazioni per la convergenza. Questo porta una complessità generalmente compresa fra $\mathbf{O(N^2)}$ e $\mathbf{O(N^3)}$.
- Nel caso distribuito, se le iterazioni fossero eseguite in modo sincrono (simultaneamente ad ogni nodo), scambiando ad ogni iterazione i risultati con i vicini, partendo dalle condizioni iniziali $D_i^0 = \infty$ per tutti gli $i \neq 1$ e $D_1^0 = 0$, l'algoritmo convergerebbe in al più N-1 passi.

2.12

Instradamento

Distance vector

"Le buone notizie viaggiano veloci"

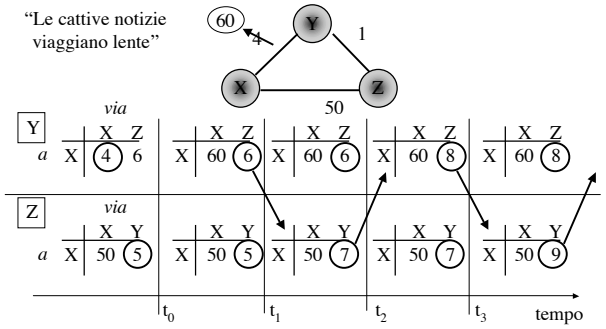


2.13

Instradamento

Distance vector

"Le cattive notizie viaggiano lente"

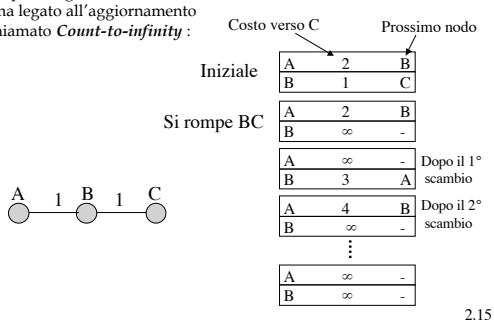


2.14

Instradamento

Distance vector

- Questo tipo di algoritmo ha un problema legato all'aggiornamento che è chiamato ***Count-to-infinity*** :



2.15

Instradamento

Distance vector

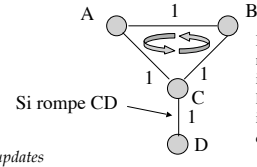
- **Ci sono diverse possibili soluzioni al count to infinity**
  - ***Path vector***
    - » oltre al costo si trasmette il percorso (*path-vector*), in questo modo i nodi possono capire quando non esiste più un percorso valido verso una certa destinazione. (BGP)
  - *Split horizon*
    - » Non viene passato il costo per una certa destinazione ad un vicino se questi è il *next hop* per quella destinazione.
    - » Una versione più complessa detta s*plit horizon with poisonous reverse*, invece di non passare costi passa un costo infinito, questo a volte accelera la convergenza. (RIP)
    - » Nel caso precedente, A non invia a B un costo (o lo invia infinito) verso C. Il ciclo quindi non si crea.

2.16

Instradamento

Distance vector

- » Se ho ,più nodi coinvolti nella rottura direttamente non si riesce a bloccare il conto ad infinito.



B e A hanno sempre mandato a C un costo infinto verso D, ma non l'uno all'altro; questo innesca un ciclo che coinvolge A, B e C.

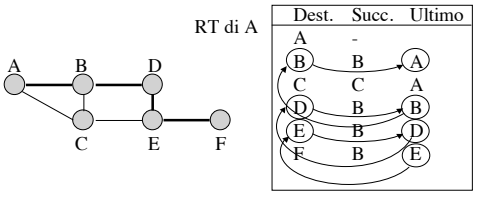
- *Triggered updates*
  - » Normalmente, per evitare un numero eccessivo di aggiornamenti delle tabelle e di traffico di controllo, si limita il ritardo minimo fra due aggiornamenti consecutivi (per es. 30 sec.). Nel caso di collegamento caduto, gli aggiornamenti sono fatti immediatamente (riduce il tempo di convergenza). (RIP)

2.17

Instradamento

Distance vector

- *Source tracing*
  - » Insieme al costo i nodi si scambiano anche il nodo da attraversare immediatamente prima della destinazione. Con questa informazione aggiuntiva è possibile ricavare direttamente dalla tabella il percorso complessivo e quindi, mantenendo la RT più piccola si ottiene lo stesso del *path vector*.



2.18

Instradamento

Link-state

- La filosofia del ***Link State (LS) routing*** è quella di distribuire a tutti i nodi della rete l'intera sua topologia ed i costi di ogni *link* che la compone.
- Con questa informazione ogni *router* è in grado ci calcolarsi i propri percorsi ottimi verso ogni destinazione.
- Se tutti vedono gli stessi costi e tutti usano lo stesso algoritmo, i percorsi saranno liberi da cicli.
- Quindi sono due gli aspetti caratterizzanti questo metodo
  - Il modo in cui la topologia della rete viene diffusa fra i nodi.
  - Il modo in cui ogni nodo calcola i percorsi ottimi.

2.19

Instradamento

Link-state - Dijkstra (Cont.)

- Nel caso LS ogni nodo applica l'algoritmo di **Dijkstra**.
- A differenza dell'algoritmo di Bellman-Ford che itera sul numero di archi attraversati da un percorso, l'algoritmo di Dijkstra itera sulla lunghezza del percorso.
- Nel caso peggiore la sua complessità è **$O(N^2)$**, in media si colloca intorno a **O(A logA)** con A numero degli archi .

2.20

Instradamento

Link-state - Dijkstra

- Sia P un insieme di nodi e D_i la distanza minima "stimata" dal nodo 1.
- Fissando inizialmente
  $P = \{1\}$, $D_1 = 0$, $D_j = d_{j1}$ per tutti i $j \neq 1$
- I passi dell'algoritmo di Dijkstra sono

  1 Trova $i \notin P$ tale che
  $D_i = \min_{j \notin P} \{D_j\}$
  e poni $P \leftarrow P \cup \{i\}$.
  Se P contiene tutti i nodi l'algoritmo è completato.

  2 Per tutti i $j \notin P$, poni $D_j \leftarrow \min\{D_j, d_{ji} + D_i\}$
  e torna al passo 1

2.21

Instradamento

Link-state

- Disseminazione della topologia
  - Ogni nodo crea un insieme di ***Link-State-Packet*** (LSP) che descrivono le sue linee in uscita.
  - Ogni LSP contiene l'indirizzo del nodo, quello dei nodi vicini, ed il costo delle linee verso i nodi vicini.
  - Ogni LSP viene distribuito a tutti i nodi tramite un ***controlled flooding***
    - » Ogni nodo che riceve un LSP lo memorizza in un database e invia una copia su tutte le proprie linee in uscita, tranne quella da cui l'ha ricevuto. Si può dimostrare che nessun LSP passa due volte per lo stesso *link* lungo la stessa direzione, quindi un LSP viene distribuito in al più 2L invii, dove L è il numero dei *link*.

2.22

Instradamento

Link-state

- Numero di sequenza
  - Per poter decidere se un LSP ricevuto è significativo (contiene una informazione più recente di quella attualmente nel nodo) ogni LSP deve contenere un numero di sequenza progressivo.
  - Il numero di sequenza ha valore locale per ogni tipo di LSP (identificato da coppia ordinata di nodi collegati da una linea)
  - Ogni volta che un nodo riceve un LSP più vecchio di quello in memoria, lo elimina senza propagarlo.

2.23

Instradamento

Link-state

- Le sequenze realmente utilizzabili sono di lunghezza finita e quindi soggetta ad "avvolgersi" (*wrapping*) bloccando l'aggiornamento.
- *Wrapped sequence number*
  - Per evitare il problema si può prendere una sequenza molto grande (32 bit => 4.295.967295) e decidere che quando due numeri distano troppo, il più piccolo sia anche il più giovane. Per esempio supponendo che N sia lunghezza della sequenza, allora $\boldsymbol{a}$ è più vecchio di $\boldsymbol{b}$ se
    - » $\boldsymbol{a} < \boldsymbol{b}$ e $|\boldsymbol{b} - \boldsymbol{a}| < N/2$
      oppure se
    - » $\boldsymbol{a} > \boldsymbol{b}$ e $|\boldsymbol{b} - \boldsymbol{a}| \geq N/2$

2.24

Instradamento
Link-state

- La presenza di un numero di sequenza pone il problema dell'inizializzazione della sequenza quando un nodo si (ri)attiva. Due sono i meccanismi adottati
  - Invecchiamento (*Aging*)
  - *Lollipop sequence space*

2.25

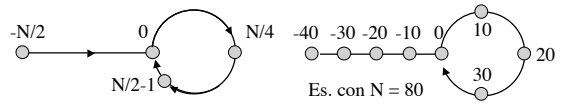
Instradamento
Link-state

- Invecchiamento
  - Prevede l'inserimento di un campo di di anzianità nel LSP, che viene inizializzato ad un valore (MAX_AGE) dal creatore del pacchetto.
  - Ogni nodo copia in un contatore C_AGE il valore MAX_AGE e lo decrementa periodicamente.
  - Quando in un *router* C_AGE raggiunge zero, la corrispettiva informazione viene eliminata dal DB e viene generato un LSP con anzianità zero, per forzare la stessa operazione sugli altri *router*.
  - E' difficile fissare un valore ottimale per MAX_AGE (troppo corto: scade prima di essere stato sostituito; troppo lungo: un nodo che riparte deve attendere a lungo perché i nuovi pacchetti diventino significativi)

2.26

Instradamento
Link-state

- *Lollipop sequence space*
  - Si tratta di una sequenza che si "avvolge" in modo particolare (al *boot* la macchina riparte da -N/2)



  - In questo caso **a** è più vecchio di **b** se:
    - » **a < 0** e **a < b** o
    - » **a > 0, a < b** e | **b - a** | < N/4, o
    - » **a > 0, b > 0, a > b** e | **b - a** | > N/4

2.27

Instradamento
Link-state

- *Lollipop sequence space*
  - Quando un nodo riceve un LSP con un numero di sequenza più vecchio di quello nel DB, lo comunica a chi gli ha inviato il pacchetto fornendo anche l'ultimo valore di sequenza che aveva memorizzato.
  - Un nodo che riparte genera sempre un numero di sequenza più vecchio degli altri e quindi i nodi vicini gli inviano l'ultimo valore da lui usato da cui può ripartire aggiungendogli 1.
  - In pratica i *router* vicini si comportano come una sorta di memoria distribuita.

2.28

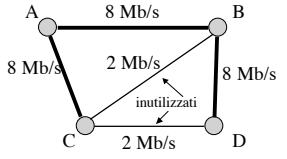
Instradamento
Link-state

- Ci sono alcune altre considerazioni di criticità da fare
  - Se la rete di partiziona per la caduta di una o più linee e le singole parti evolvono indipendentemente, quando si ricollegano possono crearsi problemi (*loop*). (Soluzione: scambio fra nodi vicini di parti di DB)
  - Se invece di una linea, si rompe un nodo, non c'è nessuno che propaga l'informazione. (Soluzione: pacchetti di Hello e anzianità massima degli LSP nel DB)
  - Bisogna proteggere gli LSP da "corruzioni" casuali o volute.

2.29

Instradamento
Metriche e dinamica

- Metriche statiche
  - In genere si tratta di valori inversamente proporzionali alla capacità del *link*. La staticità fa si che le linee a minor velocità tendano ad essere sotto-utilizzate.



2.30

Instradamento
Metriche e dinamica

- Dinamiche
  - Le metriche dipendenti dal traffico sono sicuramente più efficaci, ma comportano alcuni problemi.
  - Consideriamo la sperimentazione avvenuta su ARPAnet, dove in origine si era usata una metrica proporzionale alla lunghezza delle code di uscita dei *router*, per fare alcune osservazioni:
    - » La lunghezza (metrica) derivava da una media su un orizzonte (10 s). La durata dell'orizzonte è critica:
      - Corta: troppi transienti;
      - Lunga: rete converge lentamente;
      - La durata ottima non è omogenea sulla rete: dipende dalle capacità dei *link*

2.31

Instradamento
Metriche e dinamica

- » La dinamica del costo non deve essere alta: altrimenti alcuni percorsi vengono completamente ignorati
- » La lunghezza della coda è usata come "predittore" della situazione futura del *link*: ma linee con code lunghe non verranno scelte nel futuro e quindi si "scaricheranno" (specialmente quelle ad alta capacità) e viceversa.
- » La mancanza di restrizioni fra valori successivi dei costi può generare oscillazioni significative.
- » Il ricalcolo quasi-sincrono delle tabelle tende a raccogliere traffico su alcune linee.

2.32

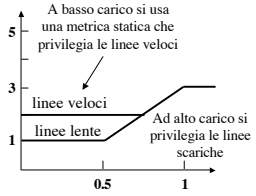
Instradamento
Metriche e dinamica

- La soluzione scelta per ARPAnet è stata:
  - Metrica mista capacità-coda (statica dinamica) dove a carico basso prevale la capacità, carico alto la coda.
  - Costi con una dinamica ridotta: valori da 1 a 3.
  - Massima variazione permessa fra due successivi ricalcoli: 1/2.

2.33

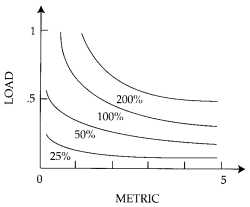
Instradamento
Metriche e dinamica

Mappa della metrica



Esprime il valore del costo in funzione del carico sulla linea

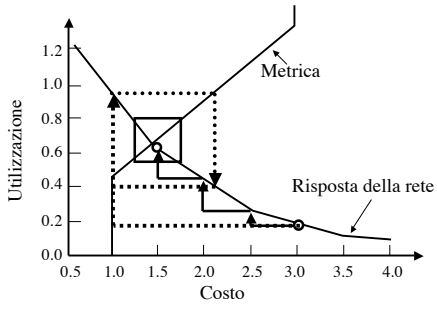
Risposta della rete



Carico medio in funzione del valore del costo su una singola linea (una curva per ogni valore del carico complessivo offerto alla rete)

2.34

Instradamento
Metriche e dinamica



2.35

Instradamento
DVR e LSR

- Il confronto DV *Routing* (DVR) e LS *Routing* (LSR) è complesso, proviamo a distinguere diversi aspetti:
  - Velocità di convergenza
    - » In genere si tende a supporre che gli LSR convergano più rapidamente dei DVR, in pratica la velocità di convergenza dipende molto dalla topologia della rete e dalle caratteristiche del traffico.
  - Volume di messaggi di controllo
    - » LS : con N nodi e A archi richiedono lo scambio di O(NA) messaggi per ciascun nodo.
    - » DV: deve solo scambiare i messaggi con i propri vicini.
  - Robustezza; se un nodo comincia a funzionare male:
    - » LS: il nodo propaga un costo sbagliato, ogni nodo si calcola separatamente la propria tabella
    - » DV: il nodo propaga un percorso sbagliato, ogni tabella viene calcolata facendo uso delle altre

2.36

Instradamento

DVR e LSR

- I DVR sono non escludono la presenza di cicli a priori, ma con le opportune modifiche gli possono evitare efficacemente.
- Gli LSR, per contro, sono più complessi, devono fare uno sforzo significativo per mantenere i DB congruenti (generando anche un traffico di controllo più elevato) ed hanno *Distance Table* più grandi.
- Gli LSR possono usare più metriche diverse contemporaneamente.
- Gli LSR si prestano ad essere estesi per supportare con le stesse tabelle rounting *unicast* e *multicast*

2.37

Instradamento

Gerarchia

Ci sono due ragioni importanti per le quali nelle reti di una certa dimensione si tende ad usare meccanismi di instradamento gerarchici:

- **La scalabilità**
  - Per un numero di nodi elevato (WAN), indipendentemente dal tipo di algoritmo, la complessità dell'instradamento e la dimensione delle RT diventano comunque eccessive (oltre al traffico di segnalazione).
  - Per esempio, nel caso LS, con tanti archi quanti nodi, si ha una complessità di circa O(NlogN) ed una RT con con dimensione O(N), quindi

# nodi	RT	Calcoli
1000	1000	O(3000)
1.000.000	1.000.000	O(6.000.000)

- **L'autonomia amministrativa**

2.38

Instradamento

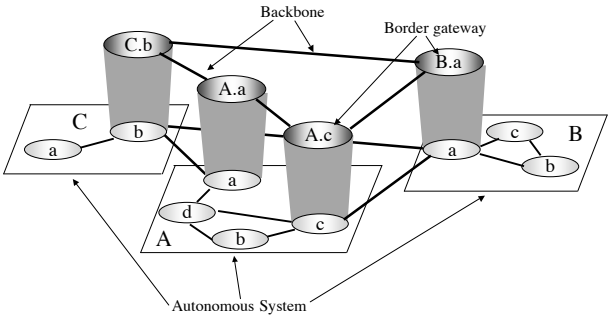
Gerarchia

- Internet distingue tre livelli gerarchici principali:
  - Sottoreti o singoli domini di *broadcast*
    - » All'interno dei quali l'instradamento fa uso dell'ARP.
  - *Autonomous System* (AS)
    In cui i protocolli di instradamento prendono il nome di ***Interior Gatway Protocol*** (IGP) e sono: RIP, IGRP e OSPF.
  - *Backbone*
    In cui i protocolli di instradamento prendono il nome di ***Exterior Gateway Protocol*** (EGP) e sono: EGP e BGP.
- Ulteriori livelli possono essere inseriti tramite alcuni protocolli (OSPF), o sfruttando la "*route aggregation*".

2.39

Instradamento

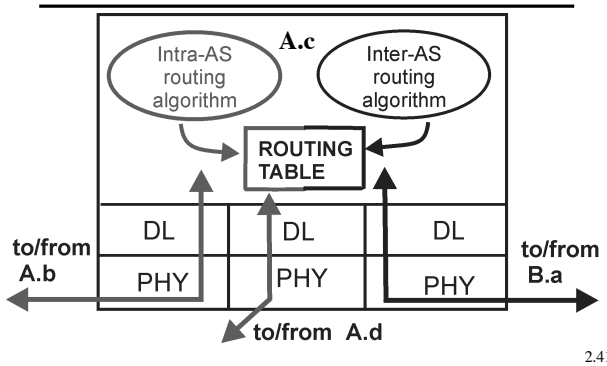
Gerarchia - *Autonomous Systems*



2.40

Instradamento

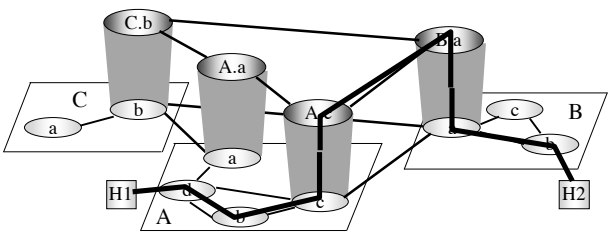
Gerarchia - *Autonomous Systems*



2.41

Instradamento

Gerarchia - *Autonomous Systems*



2.42

Instradamento

Gerarchia

- Il partizionamento è realizzato grazie alla gerarchizzazione degli indirizzi.
- L'obiettivo è avere relativamente "pochi" nodi per ogni livello (e area).
- Potenzialmente ogni livello può usare algoritmi diversi
- La gerarchia non è stretta, ossia il collegamento un area di un livello e il livello superiore può avvenire tramite più nodi (*Border Gatway*, BG)
- Ci sono dei router che partecipano all'instradamento di livelli differenti.
- Alcuni indirizzi possono non essere omogenei con lo spazio di indirizzamento dell'area/livello (questo diminuisce l'efficacia della gerarchia).

2.43

Instradamento

Gerarchia

- I diversi livelli non possono nascondersi reciprocamente tutte le informazioni:
  - Ad es., per poter calcolare l'instradamento più opportuno, un nodo del livello 3 deve conoscere i costi per raggiungere i nodi del livello superiore.
  - Allo stesso modo, un nodo di livello 4 deve conoscere i costi verso i nodi del livello 3.
  - Queste conoscenze sono fornite tramite LSP particolari (detti *external records* e *summary records*) che contengono solo le destinazioni ed i costi per raggiungerle (non la topologia). In pratica le reti dei livelli superiori/inferiori vengono rappresentate come se i loro nodi fossero direttamente collegati ai BG.

2.44

Instradamento

Routing Information Protocol (RIP)

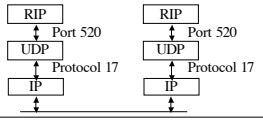
- E' un IGP originariamente progettato dalla Xerox per la propria rete, introdotto dall'Università di Berkley nella propria implementazione di TCP/IP (BSD)
- E' stato standardizzato con RFC 1058 nel 1988, la versione 2 è descritta nel RFC 1723.
- E' un DVR ed usa una metrica statica: il costo di un percorso è il numero di hop (ossia di linee) di cui è composto (ossia ogni linea a costo 1).
- Utilizza lo *split horizon with poisonous reverse*, e *i triggered update*.
- Aggiorna la RT (tramite RIP *response message* o RIP *advertisement*) ogni 30 s. e elimina ogni vettore non aggiornato per 180 s consecutivi (considerando la corrispondente linea non più disponibile).

2.45

Instradamento

Routing Information Protocol (RIP)

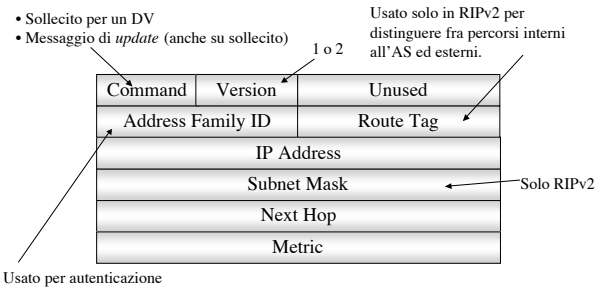
- Ha come valore massimo del costo 15, 16 corrisponde ad infinito. Quindi non permette reti con percorsi con più di 15 *router* attraversati.
- La limitazione di cui sopra è legata al fatto che per reti più grandi è troppo lento a convergere (non alla dimensione del campo costo).
- Ne esistono due versioni, la seconda (RIPv2) consente l'uso del CIDR, ossia la "*route aggregation*", e la "*default route*".
- Lo scambio di informazioni avviene attraverso un protocollo di livello 4 (UDP)



2.46

Instradamento

Routing Information Protocol (RIP)



2.47

Instradamento

Interior Gateway Routing Protocol (IGRP)

- E' nuovamente un DVR, ma di tipo proprietario; infatti è stato sviluppato dalla CISCO verso la metà degli anni '80 ed è disponibile solo sui suoi prodotti.
- Usa una metrica dinamica e sofisticata (considera ritardo, banda, affidabilità, lunghezza del pacchetto ed il carico) in cui il costo della linea viene composto tramite una somma pesato, i cui pesi sono impostabili dal gestore.
- Permette la suddivisione del carico su più linee (multipercorso).
- Usa un meccanismo sofisticato per accelerare la convergenza ed evitare i cicli.

2.48

Instradamento

Open Shortest Path First (OSPF)

- Nasce nel 1990 con l'RFC 1247 per sostituire il RIP
- Caratteristiche fondamentali:
  - protocollo aperto, le specifiche sono di pubblico dominio,
  - basato sull'algoritmo *Shortest Path First* (SPF).
- È un protocollo di tipo *Link State*
  - utilizza i *Link State Advertisements* (LSA) per diffondere informazioni sulla topologia della rete.
- Un *router* OSPF utilizza le informazioni raccolte con gli LSA per calcolare il percorso ottimo (SPF) per ogni nodo

2.49

Instradamento

OSPF – Caratteristiche

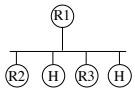
- E' stato realizzato per rispondere a diverse esigenze:
  - ***Open*:** ossia aperto e non proprietario.
  - **Sicurezza**: gli scambi fra *router* vengono autenticati, per proteggere gli aggiornamenti.
  - **Multi-metrica**: permette l'uso di più metriche anche dinamiche e instradamenti differenziati a seconda, ad esempio, del campo TOS.
  - **Multi-percorso**: permette il bilanciamento dei flussi su percorsi a costo uguale.
  - **Multicast**: supporta il multicast (M-OSPF)
  - **Gerarchico**: supporta una gerarchia interna

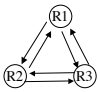
2.50

Instradamento

OSPF – Organizzazione della rete

- Supporta tre tipi di connessione e reti
  - Punto - punto fra *router*,
  - Reti multiaccesso con *broadcast* (LAN)
  - Reti multiaccesso senza *broadcast* (WAN a pacchetto)
- Nel caso di LAN a cui sono connessi più *router*, ne identifica di riferimento (*Designated Router*, DR) per ridurre il traffico di LSP in broadcast sulla LAN.





Senza DR
(n x (n-1) broadcast)

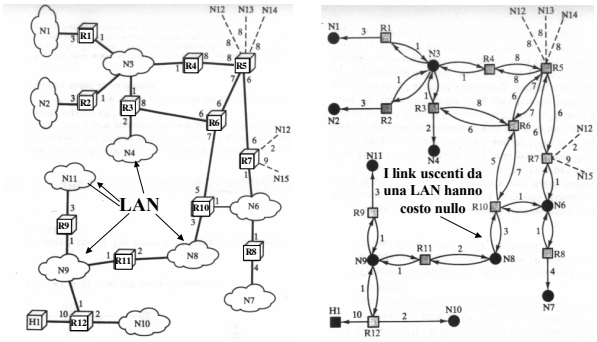
Designated Router
(Backup DR)

Con DR
(minor # di broadcast)

2.51

Instradamento

Open Shortest Path First (OSPF)



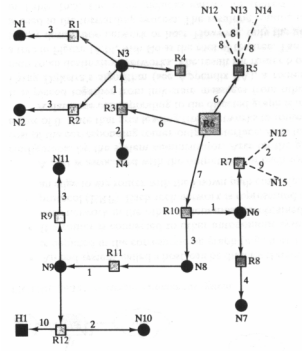
2.52

Instradamento

Open Shortest Path First (OSPF)

TABLE 16.5 Routing Table for RT6.

Destination	Next hop	Distance
N1	RT3	10
N2	RT3	10
N3	RT3	7
N4	RT3	8
N6	RT10	8
N7	RT10	12
N8	RT10	10
N9	RT10	11
N10	RT10	13
N11	RT10	14
H1	RT10	21
RT5	RT5	6
RT7	RT10	8
N12	RT10	10
N13	RT5	14
N14	RT5	14
N15	RT10	17



2.53

Instradamento

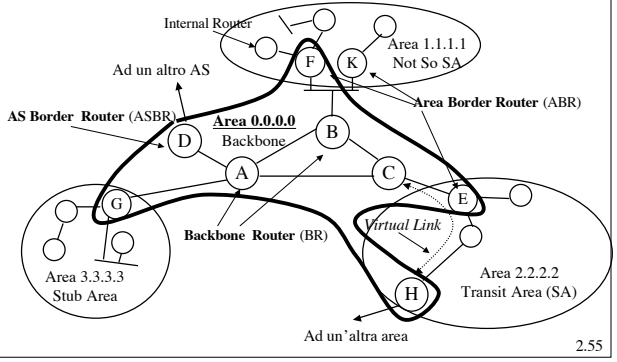
OSPF – Gerarchia

- L'entità più in alto nella gerarchia à l'*Autonomous System* (AS), che è costituito da un insieme di reti con un unico amministratore ed una comune strategia di *routing*.
- Ogni AS può essere suddiviso in *aree,* ovvero gruppi di reti contigue e relativi *host.*
- *Router* con interfacce multiple possono appartenere a più aree (*Area Border Router*).
- La topologia di un'area non è visibile da entità esterne all'area stessa.
- Mantenendo le topologie delle aree "nascoste" si riduce il traffico necessario al protocollo.
- L'area di *backbone* è quella alla tutte le restanti aree dell'AS devono essere collegate (è una sorta ci centro stella).

2.54

Instradamento

Open Shortest Path First (OSPF)



Instradamento

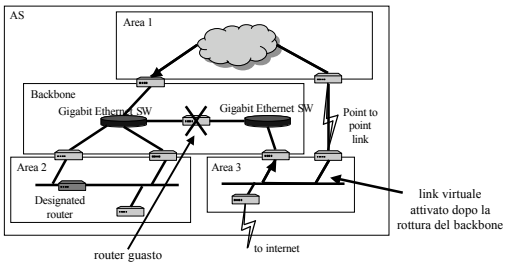
Open Shortest Path First (OSPF)

- Un'area è definita *Stub Area* se ha un solo *Border Router*, che è quindi l'unica via d'accesso alla rete esterna all'area.
- Ciascun *router* nella rete conosce solamente il *Border Router* della *Stub Area.*
- Non vengono propagati gli *external* e i *summary LSA*.
- Restrizioni che caratterizzano le *stub areas*:
  - I *link* virtuali non possono attraversare una *Stub Area*
  - Non possono contenere un *AS Boundary Router*
- L'instradamento fra due aree viene realizzato in tre parti:
  - Il percorso nell'area sorgente fra la sorgente stessa ed un *Area Border Router*.
  - Il percorso fra i due ABR delle due aree tramite il *backbone*
  - Il percorso nell'area destinazione fra l'ABR che riceve il pacchetto dal *backbone* e la destinazione.
- In pratica si forza un instradamento a stella in cui il backbone rappresenta il centro stella.
- Una *Not So Stubby Area (NSSA)* è più flessibile rispetto ad una *Stub Area*:
  - i *router* possono scambiarsi informazioni relative ad altri protocolli di *routing*.

Instradamento

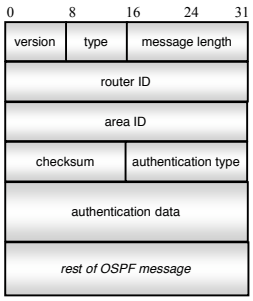
OSPF – Gerarchia

- L'amministratore di rete può creare uno o più *link* virtuali per sopperire ad anomalie



Instradamento

OSPF – *Header*



- *Version* (1 *byte*)*:* versione OSPF. La versione corrente è la 2
- *Type* (1 *byte*)*:* indica il tipo di pacchetto

Value	*Packet type*
1	*Hello Packet*
2	*Database Description Packet*
3	*Link State Request Packet*
4	*Link State Update Packet*
5	*Link State Acknowledgment Packet*

- *Message length* (2 *bytes*)*:* indica la dimensione totale del pacchetto in *bytes*
- *Router ID* (4 *bytes*)*:* identifica in modo univoco il *router* che ha generato il pacchetto (è uno degli indirizzi IP del *router*)

Instradamento

OSPF – *Header*

- *Area ID* (4 *bytes*)*:* indica l'area a cui si riferisce il pacchetto.
- *Checksum* (2 *bytes*)
- *Authentication type* (2 *bytes*)*:* indica il tipo di autenticazione utilizzata dal pacchetto

Value	*Authentication type*
0	*Null authentication*
1	*Password authentication*
2	*Cryptographic authentication*

- *Authentication data* (8 *bytes*)*:* dati di autenticazione

Instradamento

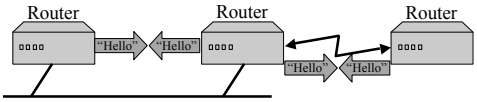
OSPF – Autenticazione

- *Password authentication*:
  - il campo *authentication data* contiene una password comune a tutti i router comunicanti. Non protegge realmente da attacchi perché la password non è criptata e viene trasmessa in ogni pacchetto OSPF. Protegge però da malfunzionamenti dovuti a errate configurazioni.
- *Cryptographic authentication:*
  - Si tratta in sostanza di una verifica della autenticità ed integrità dell'informazione fatta tramite l'algoritmo MD5.

Instradamento

OSPF – Algoritmo I

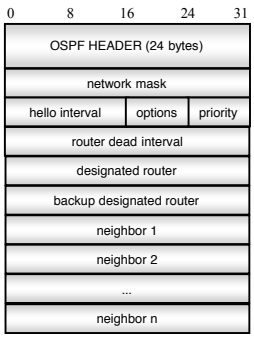
- Prima fase:
  - i *router* adiacenti si scambiano gli *hello packet*.
  - Ciascun *router* genera tali pacchetti su tutti i *link* e le reti ad esso collegati.



2.61

Instradamento

OSPF – *Hello Packet*



- *Network mask (4 bytes):* netmask associata all'interfaccia da cui viene emesso il pacchetto
- *Hello interval (2 bytes):* indica ogni quanti secondi viene emesso un pacchetto di Hello
- *Options (1 byte)*

2.62

Instradamento

OSPF – *Hello Packet*

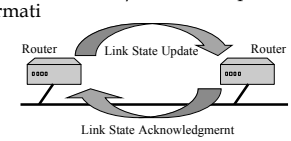
- *Priority (1 byte):* Ciascun router è configurato con una priorità, che può variare tra 0 e 255.
  - Viene eletto *Designated Router* il router che ha la priorità più alta (quindi un router con priorità 0 non potrà mai diventare DR).
- *Router dead interval (4 bytes):* indica il tempo dopo cui il Router va considerato irraggiungibile.
- *Designated router (4 bytes):* indirizzo del DR (0 se non ancora definito).
- *Backup designated router (4 bytes):* indirizzo del *Backup Designated Router* (0 se non ancora definito).
- *Neighbor 1…n:* lista di router ID da cui è stato ricevuto il pacchetto di *Hello* negli ultimi *Dead_Interval* secondi

2.63

Instradamento

OSPF – Algoritmo II

- Seconda fase:
  - Ogni *router* distribuisce le informazioni che ha ricevuto tramite gli *hello packets* al resto della rete. Per fare questo realizza un *flooding* degli *LSA* con dei *Link State Update Packets*.
  - I router devono inviare esplicitamente un *acknowledgment* quando ricevono un *LS Update Packet* specificando tutti gli LSA confermati

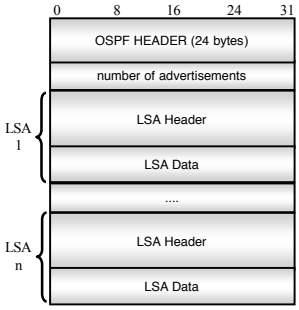


- Terza fase:
  - calcolo dei percorsi ottimi (Dijkstra) e aggiornamento delle tabelle di *routing*.

2.64

Instradamento

OSPF – *Link State Update Packet*

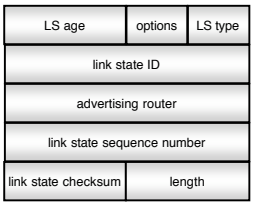


- *Number of advertisements (4 bytes):* indica il numero di LSA contenuti nel pacchetto
- *Link State Advertisement Header* (20 *bytes*)

2.65

Instradamento

OSPF – *Link State Advertisement Header*



- *LS age (16 bytes):* viene incrementato di una unità ogni secondo fino ad un massimo di 3600. Ogni router deve ripetere gli LSA da esso generati ogni 1800 s.
- *Options (1 byte)*
- *LS type (1 byte):* identifica i diversi tipo di LSA:

Value	*LSA type*
1	*Router link*
2	*Network link*
3	*Summary link to network*
4	*Summary link to AS boundary router*
5	*External link*
6	*Group membership advertisement*
7	*NSSA link*
9	*Opaque link confined to a local network*
10	*Opaque link confined to an area*
11	*Opaque link for an entire autonomous system*

2.66

Instradamento

OSPF – *Link State Advertisement Header*

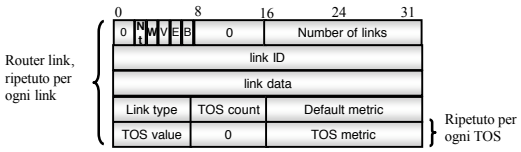
- *Link state ID* (*4 bytes*): identifica univocamente i *link* per ogni *advertising router*.
  - In genere i *routers* scelgono l'indirizzo IP dell'interfaccia sul *link* come identificativo.
- *Advertising router (4 bytes):* l'ID del *router* che genera l'*advertisement*.
- *LS sequence number (4 bytes):* utilizzato per distinguere versioni successive dello stesso LSA.
- *LS checksum (2 bytes):* codice a correzione di errore che copre l'intero LSA eccetto il campo *LS age (Fletcher's checksum).*
- *Length (2 bytes):* dimensione in *bytes* dell'intero LSA inclusi i 20 *bytes* di *header.*

2.67

Instradamento

OSPF – *Link State Advertisement*

- Tipi di LSA
  - Tipo 1: *router links*
    - » Simile ad un LSP tradizionale
    - » Contiene informazioni su *router* adiacenti e LAN collegate
    - » Propagato solo all'interno dell'area
    - » Può essere generato e propagato da un *backbone router* sul *backbone.* In questo caso è simile ad un tradizionale LSP di livello 2. Comunica quali sono i *backbone routers.*

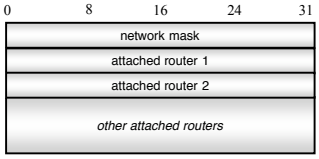


2.68

Instradamento

OSPF – *Link State Advertisement*

- Tipi di LSA
  - Tipo 2: *network links*
    - » Simile ad un tradizionale LSP generato per conto di una LAN
    - » Generato da DR di una LAN
    - » Elenca tutti i *router* presenti sulla LAN
    - » Propagato sul *backbone* dai *backbone router*

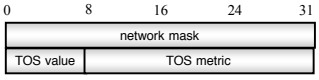


2.69

Instradamento

OSPF – *Link State Advertisement*

- Tipi di LSA
  - Tipo 3: *network summary links*
    - » Informazioni di livello 2 propagate in un'area
    - » Generato da un *area border router*
    - » Identifica altre reti interne all'AS ma esterne all'area
  - Tipo 4: *AS summary links*
    - » Informazioni di livello 2 propagate in un'area
    - » Generato da un *area border router*
    - » Identifica gli *AS border routers*

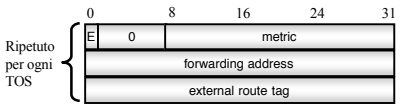


2.70

Instradamento

OSPF – *Link State Advertisement*

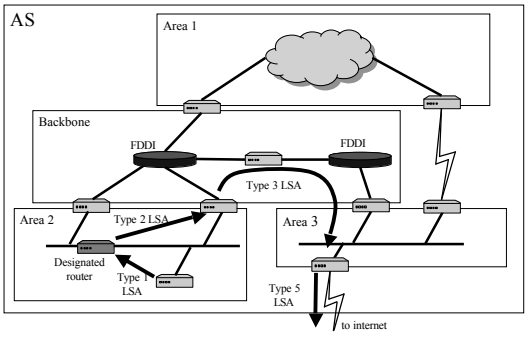
- Tipi di LSA
  - Tipo 5: *external links*
    - » Informazioni di livello 2 propagate a tutti i *routers* dell'AS
    - » Generato da un *AS border router*
    - » Identifica quali destinazioni sono disponibili esternamente all'AS



2.71

Instradamento

OSPF – *Link State Advertisement*

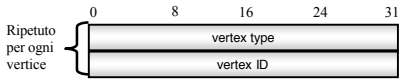


2.72

Instradamento

OSPF – *Link State Advertisement*

- Tipi di LSA
  - Tipo 6: *group membership advertisements* (supporto al multicast)
    - » Generato dal *designated router*
    - » Identifica uno o più "vertici" ovvero percorsi verso destinazioni che fanno parte del gruppo
    - » Il campo *Link state ID* dell'LSA *header* viene utilizzato per indicare l'indirizzo specifico del gruppo



2.73

Instradamento

OSPF – *Link State Advertisement*

- Tipi di LSA
  - Tipo 7: *NSSA advertisements*.
    - » Porta lo stesso tipo di informazioni di un *external LSA*.
    - » Ha lo stesso formato di un *external LSA*.
  - Tipo 9, 10, 11: *Opaque advertisements*.
    - » Confinati rispettivamente alla rete locale, alla singola area e all'intero AS.
    - » Destinati ad usi futuri.

2.74

Instradamento

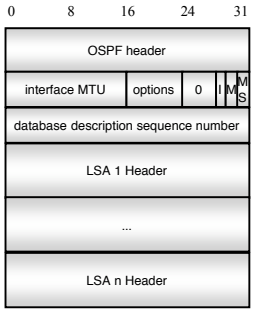
OSPF – Aggiornamento database

- In genere un *router* viene introdotto in reti già funzionanti.
- Non è possibile ripetere tutto il meccanismo di inizializzazione dell'intera rete ogni volta che si si introduce un nuovo nodo.
- Un *router* eventualmente aggiunto deve "agganciarsi" alla rete ed apprenderene la topologia più velocemente possibile; per fare questo interagisce con i *router* vicini a cui chiede di trasferirgli il proprio database.
- Quindi, nel momento in cui due *router* si "incontrano" per la prima volta tramite gli *hello packets,* si scambiano informazioni relative ai propri *link state database (database description packets)*.

2.75

Instradamento

OSPF – *Database description packet*



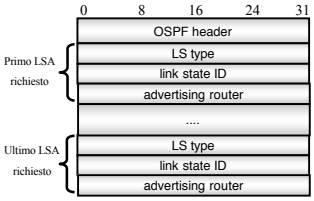
- *Flag I e M*: permettono ai vicini di scambiarsi *database description packets* multipli. Il primo di questi pacchetti ha il *flag I* (*initial*) attivo e i restanti pacchetti tranne l'ultimo hanno attivo il *flag M* (*more*).
- *Flag MS*: identifica *Master* e *Slave*

2.76

Instradamento

OSPF – *Link state request packet*

- Ricevuto il set completo dei *Database description packets*, il *router* lo confronta con il proprio *link state database* ed eventualmente richiede un aggiornamento al proprio vicino (*link state request packet*).



2.77

Instradamento

EGP - "EGP"

- Al più vecchio dei protocolli EGP è stato assegnato lo stesso nome che distingue la categoria: EGP.
- E' un protocollo di stile DV che però non propaga costi ma solo informazioni di raggiungibilità.
- Non è in grado di evitare cicli e quindi non può essere usato in topologie magliate ma solo ad albero.
- La sua struttura di riferimento è composta da "*Core Router*" (CR) collegati fra loro ad albero.
- Ogni AS può essere collegato ad un unico CR e quindi ogni CR fa da centro stella per un gruppo di AS

2.78

Instradamento

EGP - Border Gateway Protocol (BGP)

- E' il protocollo EGP relativamente recente, definito dal RFC 1654.
- La versione in uso attualmente è la 4 (BGP4).
- Permette la cooperazione fra *router* di AS diversi (chiamati *gateway*) per la realizzazione dell'instradamento fra AS.
- Per lo scambio di informazioni fra i nodi usa il TCP (porta 179).
- Non propaga vere e proprie metriche, ma lascia che la scelta dei percorsi venga determinata tramite "politiche" impostate dai singoli gestori.

2.79

Instradamento

EGP - Border Gateway Protocol (BGP)

- Pur appartenendo alla famiglia dei Distance Vector, il BGP non opera secondo un algoritmo di instradamento tradizionale ma usando delle "politiche".
- L'idea è che il gestore abbia il massimo controllo su:
  - Quali altri AS attraversare con il proprio traffico.
  - Quali percorsi annunciare (così che altri possano usarli).
- Il meccanismo adottato è quello del ***Path-Vector***, che consiste nell'annunciare l'elenco degli AS attraversati su ogni percorso, in modo che
  - Si elimini il problema dei "count- to-infinity" (se mi vedo nell'elenco non suo il percorso).
  - Si abbia sempre l'elenco degli AS attraversati e quindi si possano prendere decisioni di tipo "politico"

2.80

Instradamento

EGP - Border Gateway Protocol (BGP)

- Opera in tre passi:
  - Identificazione dei nodi adiacenti (*neighbor*)
  - Raggiungibilità dei nodi adiacenti
  - Raggiungibilità delle reti
- Distingue tre tipi di reti
  - *Stub:* che hanno un'unica connessione con il *backbone* e non possono venir usate come transito
  - *Multi-homed*: che potenzialmente possono essere usate per transito (se lo permettono)
  - *Transit*: costruite per realizzare il transito.

2.81

Instradamento

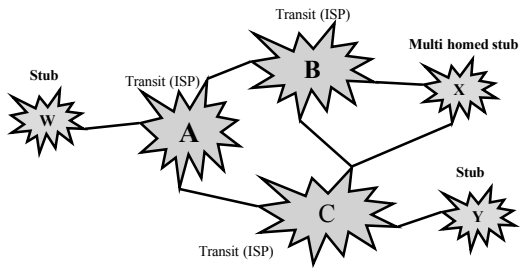
EGP - Border Gateway Protocol (BGP)

- Tre sono le attività principali svolte da un router BGP:
  - Ricezione e filtraggio di annunci di percorsi dai vicini
  - Selezione del percorso; l'instradamento a livello di AS può dipendere:
    - » Dagli accordi "commerciali" fra diversi AS
    - » Dai rapporti politici
    - » Dagli obiettivi delle diverse aziende che possiedono gli AS (ISP, Aziende private,…)
  - Invio di annunci su percorsi
    - » Permette un notevole controllo su cosa verrà instradato sul proprio AS
- A queste tre attività fanno capo tre database logicamente diversi.

2.82

Instradamento

EGP - Border Gateway Protocol (BGP)



2.83

Instradamento

EGP - Border Gateway Protocol (BGP)

- La presenza di più BGP router in un AS richiede un meccanismo interno all'AS per mantenere la consistenza fra i database dei diversi router.
- Viene definito quindi un I(nternal)-BGP.
- Il *path-vector* non evita la presenza di *loop* in questo caso perché tutti i router sono identificati dallo stesso identificativo di AS.
- Allora viene introdotta una regola: i BGP router non possono propagare informazioni ricevute da altri BGP router interni all'AS:
  - Quindi i router BGP di un AS devono essere (almeno logicamente) completamente connessi.
  - In realtà esistono possibili alternative (reflector, confederations)

2.84

Instradamento

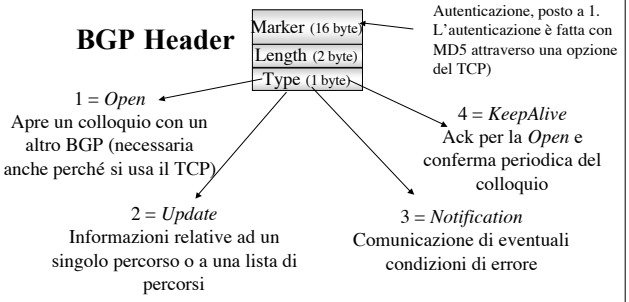
EGP - Border Gateway Protocol (BGP)

- In quanto protocollo di tipo EGP, il BGP differisce dai protocolli IGP in diversi aspetti:
  - Politica: fra AS nella scelta del percorso la "politica" domina, ossia le scelte dipendono da più fattori che coinvolgono considerazioni strategiche, economiche e di sicurezza più che tecniche. Queste considerazioni sono specifiche di ogni AS, quindi la scelta è principalmente sotto il controllo amministrativo.
  - Scala: la scalabilità è molto importante perché le "reti" coinvolte sono generalmente grandi. Invece se un AS cresce eccessivamente lo si può sempre dividere in due.
  - Prestazioni: le prestazioni tecniche contano relativamente poco in un EGP.

2.85

Instradamento

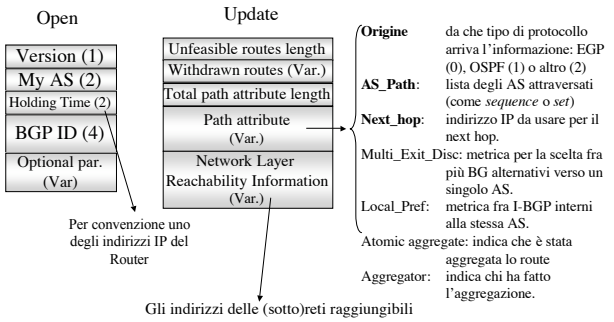
EGP - Border Gateway Protocol (BGP)



2.86

Instradamento

EGP - Border Gateway Protocol (BGP)



2.87