7 LIVELLO RETE (C)

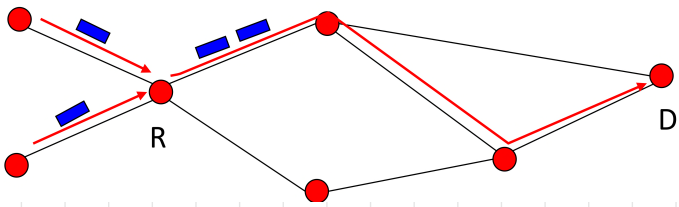
L'instradamento unicast consente a due nodi A e B non collegati direttamente di comunicare. Esistono anche indirizzamenti broadcast e multicast. Le entità di livello 3 eseguono forwarding alla prossima entità basandosi sulle tabelle d'instradamento. Per ora abbiamo supposto che le tabelle fossero scritte e mantenute da comandi. Nella realtà esse sono gestite da algoritmi di routing (politiche di routing). Gli algoritmi di routing determineranno il percorso attraverso la rete, mentre la tabella si occuperà del forwarding locale.

I protocolli di routing racchiudono due funzionalità:
- scambio fra router di informazioni di raggiungibilità
- costruzione di tabelle d'instradamento

Il protocollo, formalmente, è solo la parte che descrive lo scambio di messaggi. L'algoritmo è implementation detail.

Il routing è legato alle capacità delle reti. Il routing può suddividere il carico su un link in modo che tutti vadano più veloce. La capacità totale di una rete, quindi, dipende dal routing.

Il tipo di inoltro IP (next-hop, destination-based) ha delle importanti conseguenze sulle politiche di routing:
- i pacchetti da A a B che arrivano in R seguiranno lo stesso percorso indipendentemente dal link d'ingresso



L'insieme dei cammini da una sorgente verso una destinazione sarà un'albero.

Per il calcolo dei cammini si usa il metodo dei cammini minimi con metrica generale (non lunghezza). Il calcolo avviene in modo distribuito tramite scambio di informazioni. I cammini minimi risultano la restrizione sopra perché tutti i sotto-cammini di un cammino minimo sono anch'essi minimi.
La metrica può essere il numero di salti, lunghezza delle code ecc...

La rete viene rappresentata come un albero completo. Il calcolo di questo albero è distribuito ed è basato su algoritmi efficienti.

7.1 ALGORITMO DI BELLMAN-FORD

Accetta pesi sia positivi che negativi. Non ci deve essere nessun ciclo a lunghezza negativa. Ha come scopo quello di trovare i cammini minimi tra una sorgente e tutti gli altri nodi e viceversa.
La variabile D_i^h indica la lunghezza del cammino minimo tra i e il nodo 1 composto da un numero di archi inferiore ad h.
In ciascuna iterazione aggiorniamo $D_i^{h+1} = \min(D_i^h, \min_j(D_j^h + d_{ji}))$ dove j sono i "vicini" di i. L'algoritmo termina in $N-1$ passi. La complessità è di $O(n^3)$

h=0;
$D_s^h = 0; \quad \forall h$
$D_j^h = \infty \quad \forall j \neq s;$
repeat
$h = h + 1;$
$D_j^h = \min\left\{\min_i\left(D_i^{h-1} + d_{ij}\right), D_j^{h-1}\right\};$
until $D_j^h = D_j^{h-1} \quad \forall j \neq s$

Si può dimostrare che l'algoritmo converge anche in modalità distribuita. Periodicamente i nodi inviano ai nodi vicini le proprie stime e aggiornano la propria secondo i parametri ricevuti. L'ordine di aggiornamento è ininfluente.

Nella pratica l'algoritmo è implementato usando delle etichette per ogni nodo contenenti:

- n: primo nodo sul cammino minimo
- L: lunghezza del cammino.

Le etichette vengono aggiornate guardando le etichette dei vicini. Quando le etichette smettono di cambiare si ricostruisce l'albero seguendo le etichette.