Piano di controllo

giovedì 24 agosto 2023 19:14

Fin ora abbiamo visto il funzionamento di tabelle d'inoltro e di flusso, queste sono la congiunzione tra i due piani a livello di rete.
Nel piano di controllo comprenderemo come queste tabelle vengono calcolate; abbiamo accennato precedentemente due approcci:

- Controllo locale (approccio tradizionale) protocolli come OSPF e BGP sono basati su questo approccio;
- Controllo logicamente centralizzato o controllo remoto che implementa un controller remoto.

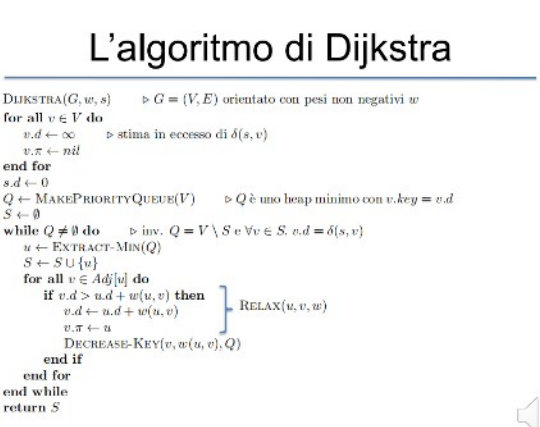
Algoritmi di instradamento

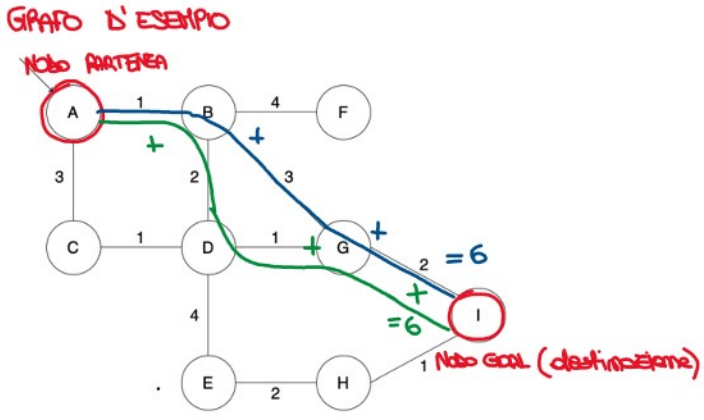
Lo scopo di questi algoritmi è quello di determinare i percorsi sorgenti e destinatari che sono separati da una rete di router.
In precedenti corsi abbiamo studiato i grafi, la loro struttura e le loro proprietà. Facciamo un piccolo ripasso…
Sia dato il grafo *G = (N, E)* dove *N* è l'insieme di nodi ed *E* è l'insieme di archi che congiungono due nodi, ad ogni arco viene associato un costo di percorrenza secondo a delle esigenze arbitrarie, quindi il costo può assumere significati diversi, ad esempio nel nostro caso la lunghezza del collegamento, velocità del collegamento ecc. basta che siano omogenei fra loro, ovvero abbiano tutti lo stesso significato.
Il grafo considerato per il nostro studio è un **grafo non orientato** e dobbiamo trovare il costo del percorso *(-> somma dei costi degli archi che percorriamo)* minimo.
Gli algoritmi di instradamento si classificano secondo diversi fattori:

1. Centralizzato: calcola il percorso avendo una conoscenza globale e completa della rete, vengono chiamati algoritmi link state;
2. Decentralizzato: calcola il percorso in maniera distribuita ed iterativa, ogni nodo conosce solo il costo degli archi dei nodi adiacenti, vengono chiamati algoritmi distance vector;
3. Statici: applicabili su percorsi che cambiano raramente e sono di fatto statici;
4. Dinamici: determinano gli instradamenti al variare del volume di traffico o della topologia della rete;
5. Sensibile al carico: il costo degli archi viene determinato dalla congestione di rete, *sono sensibili al traffico*;
6. Insensibili al carico: l'opposto del punto precedente.

Link State Routing - LS

Questo tipo di algoritmo deve conoscere il tipo della rete in cui si trovo e i costi di tutti i collegamenti all'interno di essa; il più importante è l' algoritmo di Dijkstra…





Al fine avremo una situazione del genere:

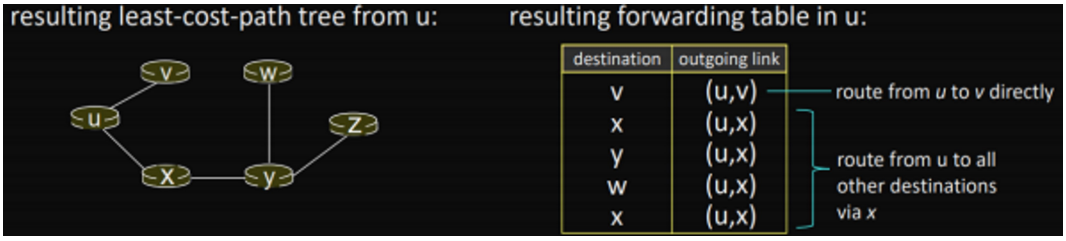
S = {(A, 0, None),
(B, 1, A)
(C, 3, A),
(D, 3, B),
(G, 4, B),
(G, 4, D),
(F, 5, B),
(I, 6, G)}
Q = {(E, 7, D)}

L'algoritmo termina una volta che ha esplorato tutti i nodi nel grafo. Al fine si prende in considerazione l'insieme S e si

esegue l'operazione di backtracking *(scorro l'insieme in direzione opposta)*. Notiamo che G che ha due precedenti B e D, può essere raggiunto in due modi, il percorso minimo può essere quindi più di uno, infatti alla fine del backtracking avremo i seguenti percorsi:

- *A → B → G → I*
- *A → B → D→G → I*

Questi due percorsi sono equivalenti a livello di costo, entrambi i percorsi hanno costo 6, tutti gli altri percorsi saranno sicuramente con un costo maggiore.
Nell'esempio abbiamo utilizzato come supporto una coda di priorità come struttura questo fa si che l'algoritmo termini con una complessità di O(N log(N)), usare strutture come array peggiora la complessità arrivando ad unna funzione di O(N^2).



Esempio di tabella d'inoltro costruita tramite l'algoritmo di Dijkstra prendendo in considerazione u come nodo iniziale

Negli algoritmi LS ogni router monitora e mantiene aggiornato lo stato dei suoi link incidenti, questo avviene tramite il protocollo **Hello Protocol** in 2 passi:

1. Ogni router invia in broadcast lo stato dei propri collegamenti, necessita ovviamente di un meccanismo di *flooding* Affidabile;
2. Ogni router calcola i cammini minimi tra esso e tutti gli altri nodi, utilizzando l'algoritmo descritto precedentemente.

Dato che i router eseguono l'algoritmo in maniera sincrona sorge un problema di oscillazioni: i costi dei collegamenti riflettono il traffico sullo stesso collegamento in quanto pare a tutti in contemporaneamente la strada migliore, tale collegamento tenderà a saturarsi lasciando gli altri con costo maggiore ma senza ritardi liberi…
Questo problema si risolve desincronizzando l'esecuzione dell'algoritmo, un aspetto buffo è dato dal fatto che i router tenderanno ad auto-sincronizzarsi e di conseguenza si necessita di una randomicità periodica per cui l'algoritmo deve essere eseguito.

Distance Vector Routing - DV

Tale algoritmo è iterativo, ovvero non termina ma si blocca, si ripete fin tanto che avvengono scambi d'informazioni con i vicini, è asincrono, i nodi non operano contemporaneamente ed è distribuito, ogni nodo non riceve informazioni globali ma solo relativi ai propri vicini.
L'algoritmo DV viene anche chiamato **Bellman-Ford**, poiché basato proprio sulle formule e sull'algoritmo di quest'ultimo:

FORMULA : $d_x(y) = \min_v \{c(x,v) + d_v(y)\}$

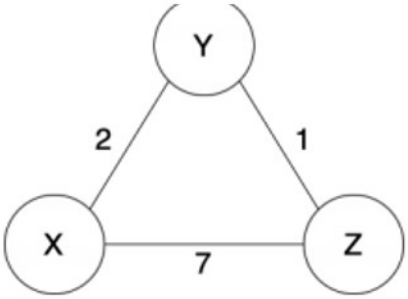
il minimo tra tutti i vicini v di x

costo del singolo arco individuato con i suoi vicini

costo del percorso minimo dal nodo padre al nodo parentesi

L'idea di base è quella di partire dal nodo sorgente ed esplorare i nodi adiacenti per assegnare a loro il costo per raggiungerli, anche qui il costo è determinato dal costo iniziale dal nodo sommato al costo dell'arco per raggiungere l'adiacente. Oltre al valore per raggiungere il nodo si salva anche il predecessore *(->il nodo correlato al valore immediatamente precedente)* questo permette back tracking.
L'algoritmo Bellman-Ford ci consente la dinamicità: non necessito di iterare l'intero algoritmo su tutto il grafo se si modificano i costi di un sottoinsieme degli archi (in cui io nodo non c'entro) ma basta iterarlo sui nodi interessati.
Vediamo ora il suo funzionamento…

L'insieme dei costi degli archi di x è:
$D_x = \{0, 2, +\infty\}$;



è:

$D_X = \{0, 2, 7\}$;

Anche Y e Z calcolano i loro:

$D_Y = \{2, 0, 1\}$

$D_Z = \{7, 1, 0\}$

Infine ogni nodo comunica il suo insieme ai suoi adiacenti...

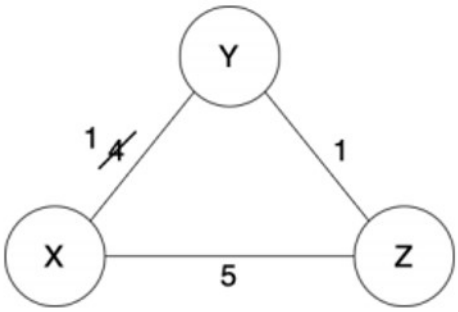
Tabella calcolata dopo tutti gli aggiornamenti

	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

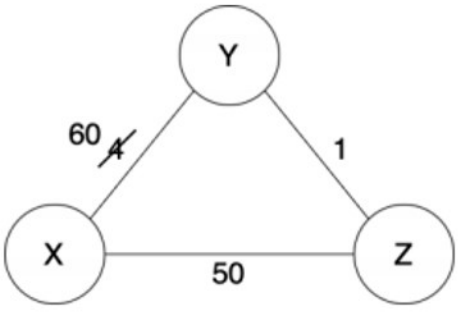
Alla prima iterazione era ∞ (il percorso diretto che X sapeva senza comunicare con i vicini) poi si è accorto che passando da Y costa 2+1 per raggiungere Z

"Le buone notizie si propagano velocemente, le cattive notizie si propagano lentamente"

I costi dei collegamenti possono variare: aumentano o diminuiscono...



Denotiamo che all'istante *t0* il nodo Y rileva un cambiamento del costo di un suo collegamento, quindi Y aggiorna la sua tabella e invia i suoi valori ai vicini all'istante *t1*; all'istante *t2* il nodo X sa che può raggiungere Z con costo minimo 2 al posto che 5. Questo concretizza la prima parte del titolo di questo paragrafo, Vediamo ora il caso opposto...



Questa volta Y rileva un aumento di costo del suo collegamento y quindi ricalcola il suo insieme con la formula Bellman-Ford, quindi:

$C_Y(X) = min\ \{c(Y, X) + C_X(X), c(Y, Z) + C_Z(X)\} = min\{60 + 0, 1 + 5\}$ ← vecchio valore per raggiungere X da Z

Il costo evidenziato calcolato è però errato! Verrà calcolato errato fino a che il valore che prima era minimo non raggiunge quello che adesso è il nuovo minimo, nel nostro esempio 50, per ricalcolare correttamente il costo si dovrebbero fare quindi 44 iterazioni..Questo problema si denomina **problema del conteggio all'infinito**.
Quest'ultimo ci spinge ad avere un **instradamento ciclico**, nel nostro esempio infatti i pacchetti verranno instradati in maniera compulsiva tra Y e Z senza mai raggiungere X.

Aggiunta dell'inversione avvelenata all'instradamento DV

L'inversione avvelenata permette di evitare lo scenario con i cicli. Il funzionamento è il seguente: se Z instrada tramite Y per giungere alla destinazione X, allora Z avvertirà Y che la sua destinazione verso X è infinita *(ovvero gli comunica che* $Dz(x) = \infty$ *anche se non è vero)*, dirà questo fino a quando Z instrada verso X passando per Y.
Ora dato che Y crede che Z non abbia un percorso verso X, non tenterà mai di instradare verso X passando per Z.
L'inversione avvelenata tuttavia non risolve il problema in generale del conteggio all'infinito, i cicli che non riguardano due nodi adiacenti non verranno rilevati dalla tecnica dell'inversione avvelenata.