

SIMULAZIONE DEL PROTOCOLLO DI ROUTING DISTANCE VECTOR

1. INTRODUZIONE

Questo progetto ha l'obiettivo di simulare il funzionamento del protocollo di routing Distance Vector Routing, un metodo semplice per comprendere i principi base dei protocolli di routing dinamico. Il protocollo si basa sull'algoritmo di Bellman-Ford, che consente a ogni nodo della rete di calcolare il percorso più breve verso tutti gli altri nodi scambiando informazioni con i propri vicini.

DESCRIZIONE DEL PROTOCOLLO DISTANCE VECTOR ROUTING

Ogni nodo mantiene una tabella di routing contenente:

- Le destinazioni raggiungibili.
- Il costo del percorso (distanza minima).
- Il prossimo nodo per raggiungere la destinazione.
- I nodi scambiano periodicamente le loro tabelle di routing con i vicini.
- Quando un nodo riceve una nuova tabella da un vicino, confronta i costi e aggiorna la propria tabella se trova un percorso più breve.

2. GRAFO E FUNZIONI UTILIZZATE

A. Graph

-Si dichiara un grafo iniziale nel quale vengono determinati i nodi e i costi degli archi.

```
graph = {  
    'A': {'B': 2, 'C': 5},  
    'B': {'A': 2, 'D': 8},  
    'C': {'A': 5, 'D': 3, 'E': 7},  
    'D': {'B': 8, 'C': 3, 'F': 2},  
    'E': {'C': 7, 'F': 4, 'G': 6},  
    'F': {'D': 2, 'E': 4, 'G': 1},  
    'G': {'E': 6, 'F': 1}  
}
```

B. init_routing_tables(graph)

- Inizializza le tabelle di routing per ogni nodo.
- Ogni nodo conosce solo le distanze ai propri vicini diretti.

C. update_routing_table(node, graph, routing_tables)

- Aggiorna la tabella di routing di un nodo sulla base delle informazioni ricevute dai vicini.
- Implementa l'algoritmo di Bellman-Ford.

D. distance_vector_routing(graph, max_iterations=10)

- Simula l'algoritmo Distance Vector Routing.
- Mostra l'evoluzione delle tabelle di routing ad ogni iterazione.
- Si arresta quando la convergenza è raggiunta o il numero massimo di iterazioni è superato.

E. print_routing_tables(routing_tables)

- Visualizza le tabelle di routing in un formato leggibile per l'utente.

3. OUTPUT

A. TABELLE DI ROUTING INIZIALI

-Ogni nodo conosce solo i costi dei collegamenti diretti.

Routing Tables iniziali:	
Routing Table per il Nodo A:	
To A: Cost = 0, Path = A	
To B: Cost = 2, Path = A → B	
To C: Cost = 5, Path = A → C	
To D: Cost = ∞, Path = N/A	
To E: Cost = ∞, Path = N/A	
To F: Cost = ∞, Path = N/A	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo B:	
To A: Cost = 2, Path = B → A	
To B: Cost = 0, Path = B	
To C: Cost = ∞, Path = N/A	
To D: Cost = 8, Path = B → D	
To E: Cost = ∞, Path = N/A	
To F: Cost = ∞, Path = N/A	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo C:	
To A: Cost = 5, Path = C → A	
To B: Cost = ∞, Path = N/A	
To C: Cost = 0, Path = C	
To D: Cost = 3, Path = C → D	
To E: Cost = 7, Path = C → E	
To F: Cost = ∞, Path = N/A	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo D:	
To A: Cost = ∞, Path = N/A	
To B: Cost = 8, Path = D → B	
To C: Cost = 3, Path = D → C	
To D: Cost = 0, Path = D	
To E: Cost = ∞, Path = N/A	
To F: Cost = 2, Path = D → F	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo E:	
To A: Cost = ∞, Path = N/A	
To B: Cost = ∞, Path = N/A	
To C: Cost = 7, Path = E → C	
To D: Cost = ∞, Path = N/A	
To E: Cost = 0, Path = E	
To F: Cost = 4, Path = E → F	
To G: Cost = 6, Path = E → G	
Routing Table per il Nodo F:	
To A: Cost = ∞, Path = N/A	
To B: Cost = ∞, Path = N/A	
To C: Cost = ∞, Path = N/A	
To D: Cost = 2, Path = F → D	
To E: Cost = 4, Path = F → E	
To F: Cost = 0, Path = F	
To G: Cost = 1, Path = F → G	
Routing Table per il Nodo G:	
To A: Cost = ∞, Path = N/A	
To B: Cost = ∞, Path = N/A	
To C: Cost = ∞, Path = N/A	
To D: Cost = ∞, Path = N/A	
To E: Cost = 6, Path = G → E	
To F: Cost = 1, Path = G → F	
To G: Cost = 0, Path = G	

B. AGGIORNAMENTI ITERATIVI

-Ad ogni iterazione, i nodi aggiornano le loro tabelle in base alle informazioni ricevute dai vicini (nodo sorgente, nodo destinazione, costo totale percorso, tutti i nodi percorsi).

Iterazione 1:	
Routing Table per il Nodo A:	
To A: Cost = 0, Path = A	
To B: Cost = 2, Path = A → B	
To C: Cost = 5, Path = A → C	
To D: Cost = 8, Path = A → C → D	
To E: Cost = 12, Path = A → C → E	
To F: Cost = ∞, Path = N/A	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo B:	
To A: Cost = 2, Path = B → A	
To B: Cost = 0, Path = B	
To C: Cost = 7, Path = B → A → C	
To D: Cost = 8, Path = B → D	
To E: Cost = 14, Path = B → A → C → E	
To F: Cost = 10, Path = B → D → F	
To G: Cost = ∞, Path = N/A	
Routing Table per il Nodo C:	
To A: Cost = 5, Path = C → A	
To B: Cost = 7, Path = C → A → B	
To C: Cost = 0, Path = C	
To D: Cost = 3, Path = C → D	
To E: Cost = 7, Path = C → E	
To F: Cost = 5, Path = C → D → F	
To G: Cost = 13, Path = C → E → G	
Routing Table per il Nodo D:	
To A: Cost = 8, Path = D → C → A	
To B: Cost = 8, Path = D → B	
To C: Cost = 3, Path = D → C	
To D: Cost = 0, Path = D	
To E: Cost = 6, Path = D → F → E	
To F: Cost = 2, Path = D → F	
To G: Cost = 3, Path = D → F → G	
Routing Table per il Nodo E:	
To A: Cost = 12, Path = E → C → A	
To B: Cost = 14, Path = E → C → A → B	
To C: Cost = 7, Path = E → C	
To D: Cost = 6, Path = E → F → D	
To E: Cost = 0, Path = E	
To F: Cost = 4, Path = E → F	
To G: Cost = 5, Path = E → F → G	
Routing Table per il Nodo F:	
To A: Cost = 10, Path = F → D → C → A	
To B: Cost = 10, Path = F → D → B	
To C: Cost = 5, Path = F → D → C	
To D: Cost = 2, Path = F → D	
To E: Cost = 4, Path = F → E	
To F: Cost = 0, Path = F	
To G: Cost = 1, Path = F → G	
Routing Table per il Nodo G:	
To A: Cost = 11, Path = G → F → D → C → A	
To B: Cost = 11, Path = G → F → D → B	
To C: Cost = 6, Path = G → F → D → C	
To D: Cost = 3, Path = G → F → D	
To E: Cost = 5, Path = G → F → E	
To F: Cost = 1, Path = G → F	
To G: Cost = 0, Path = G	

C. CONVERGENZA

-Le tabelle di routing convergono quando non ci sono più aggiornamenti.

<p>Iterazione 3:</p> <p>Routing Table per il Nodo A:</p> <p>To A: Cost = 0, Path = A</p> <p>To B: Cost = 2, Path = A → B</p> <p>To C: Cost = 5, Path = A → C</p> <p>To D: Cost = 8, Path = A → C → D</p> <p>To E: Cost = 12, Path = A → C → D → E</p> <p>To F: Cost = 10, Path = A → C → D → F</p> <p>To G: Cost = 11, Path = A → C → D → F → G</p> <p>Routing Table per il Nodo B:</p> <p>To A: Cost = 2, Path = B → A</p> <p>To B: Cost = 0, Path = B</p> <p>To C: Cost = 7, Path = B → A → C</p> <p>To D: Cost = 8, Path = B → D</p> <p>To E: Cost = 14, Path = B → A → C → E</p> <p>To F: Cost = 10, Path = B → D → F</p> <p>To G: Cost = 11, Path = B → D → F → G</p> <p>Routing Table per il Nodo C:</p> <p>To A: Cost = 5, Path = C → A</p> <p>To B: Cost = 7, Path = C → A → B</p> <p>To C: Cost = 0, Path = C</p> <p>To D: Cost = 3, Path = C → D</p> <p>To E: Cost = 7, Path = C → D → E</p> <p>To F: Cost = 5, Path = C → D → F</p> <p>To G: Cost = 6, Path = C → D → F → G</p> <p>Routing Table per il Nodo D:</p> <p>To A: Cost = 8, Path = D → C → A</p> <p>To B: Cost = 8, Path = D → B</p> <p>To C: Cost = 3, Path = D → C</p> <p>To D: Cost = 0, Path = D</p> <p>To E: Cost = 6, Path = D → F → E</p> <p>To F: Cost = 2, Path = D → F</p> <p>To G: Cost = 3, Path = D → F → G</p> <p>Routing Table per il Nodo E:</p> <p>To A: Cost = 12, Path = E → C → A</p> <p>To B: Cost = 14, Path = E → C → A → B</p> <p>To C: Cost = 7, Path = E → C</p> <p>To D: Cost = 6, Path = E → F → D</p> <p>To E: Cost = 0, Path = E</p> <p>To F: Cost = 4, Path = E → F</p> <p>To G: Cost = 5, Path = E → F → G</p> <p>Routing Table per il Nodo F:</p> <p>To A: Cost = 10, Path = F → D → C → A</p> <p>To B: Cost = 10, Path = F → D → B</p> <p>To C: Cost = 5, Path = F → D → C</p> <p>To D: Cost = 2, Path = F → D</p> <p>To E: Cost = 4, Path = F → E</p> <p>To F: Cost = 0, Path = F</p> <p>To G: Cost = 1, Path = F → G</p> <p>Routing Table per il Nodo G:</p> <p>To A: Cost = 11, Path = G → F → D → C → A</p> <p>To B: Cost = 11, Path = G → F → D → B</p> <p>To C: Cost = 6, Path = G → F → D → C</p> <p>To D: Cost = 3, Path = G → F → D</p> <p>To E: Cost = 5, Path = G → F → E</p> <p>To F: Cost = 1, Path = G → F</p> <p>To G: Cost = 0, Path = G</p>	<p>Iterazione 4:</p> <p>Routing Table per il Nodo A:</p> <p>To A: Cost = 0, Path = A</p> <p>To B: Cost = 2, Path = A → B</p> <p>To C: Cost = 5, Path = A → C</p> <p>To D: Cost = 8, Path = A → C → D</p> <p>To E: Cost = 12, Path = A → C → D → E</p> <p>To F: Cost = 10, Path = A → C → D → F</p> <p>To G: Cost = 11, Path = A → C → D → F → G</p> <p>Routing Table per il Nodo B:</p> <p>To A: Cost = 2, Path = B → A</p> <p>To B: Cost = 0, Path = B</p> <p>To C: Cost = 7, Path = B → A → C</p> <p>To D: Cost = 8, Path = B → D</p> <p>To E: Cost = 14, Path = B → A → C → E</p> <p>To F: Cost = 10, Path = B → D → F</p> <p>To G: Cost = 11, Path = B → D → F → G</p> <p>Routing Table per il Nodo C:</p> <p>To A: Cost = 5, Path = C → A</p> <p>To B: Cost = 7, Path = C → A → B</p> <p>To C: Cost = 0, Path = C</p> <p>To D: Cost = 3, Path = C → D</p> <p>To E: Cost = 7, Path = C → D → E</p> <p>To F: Cost = 5, Path = C → D → F</p> <p>To G: Cost = 6, Path = C → D → F → G</p> <p>Routing Table per il Nodo D:</p> <p>To A: Cost = 8, Path = D → C → A</p> <p>To B: Cost = 8, Path = D → B</p> <p>To C: Cost = 3, Path = D → C</p> <p>To D: Cost = 0, Path = D</p> <p>To E: Cost = 6, Path = D → F → E</p> <p>To F: Cost = 2, Path = D → F</p> <p>To G: Cost = 3, Path = D → F → G</p> <p>Routing Table per il Nodo E:</p> <p>To A: Cost = 12, Path = E → C → A</p> <p>To B: Cost = 14, Path = E → C → A → B</p> <p>To C: Cost = 7, Path = E → C</p> <p>To D: Cost = 6, Path = E → F → D</p> <p>To E: Cost = 0, Path = E</p> <p>To F: Cost = 4, Path = E → F</p> <p>To G: Cost = 5, Path = E → F → G</p> <p>Routing Table per il Nodo F:</p> <p>To A: Cost = 10, Path = F → D → C → A</p> <p>To B: Cost = 10, Path = F → D → B</p> <p>To C: Cost = 5, Path = F → D → C</p> <p>To D: Cost = 2, Path = F → D</p> <p>To E: Cost = 4, Path = F → E</p> <p>To F: Cost = 0, Path = F</p> <p>To G: Cost = 1, Path = F → G</p> <p>Routing Table per il Nodo G:</p> <p>To A: Cost = 11, Path = G → F → D → C → A</p> <p>To B: Cost = 11, Path = G → F → D → B</p> <p>To C: Cost = 6, Path = G → F → D → C</p> <p>To D: Cost = 3, Path = G → F → D</p> <p>To E: Cost = 5, Path = G → F → E</p> <p>To F: Cost = 1, Path = G → F</p> <p>To G: Cost = 0, Path = G</p> <p>Convergenza raggiunta.</p>
--	--

4. CONCLUSIONE E CONSIDERAZIONI

A. CONFRONTO CON ALTRI PROTOCOLLI

Il Distance Vector Routing è più semplice da implementare rispetto a protocolli come OSPF.

Può essere più lento rispetto ai protocolli basati sullo stato del collegamento questo lo rende adatto per reti piccole o didattiche, ma meno efficiente in reti grandi.

B. ASPETTI POSITIVI

-Ogni nodo aggiorna la propria tabella di routing basandosi solo sulle informazioni ricevute dai vicini.

-Si adatta automaticamente ai cambiamenti della rete, aggiornando le tabelle di routing quando si verificano modifiche nella topologia.

-Non richiede una visione globale della rete. Ogni nodo opera indipendentemente scambiando informazioni con i propri vicini.

-Molti protocolli di routing tradizionali, come RIP (Routing Information Protocol), si basano sul Distance Vector, dimostrando la sua robustezza storica.

C. ASPETTI NEGATIVI

-Quando si verifica un cambiamento nella rete, il processo di aggiornamento delle tabelle può richiedere molte iterazioni per raggiungere la convergenza, specialmente in reti di grandi dimensioni.

-È vulnerabile a problemi di instradamento ciclico (routing loops), dove i pacchetti possono continuare a circolare indefinitamente.

-Mitigazioni come split horizon, route poisoning e hold-down timers devono essere introdotte per ridurre questo rischio.

-Non è adatto per reti di grandi dimensioni, poiché la complessità della convergenza e il traffico di aggiornamento aumentano rapidamente.

-In alcune topologie, possono essere inviati aggiornamenti non necessari, aumentando l'overhead della comunicazione.

-Ogni nodo si basa solo sulle informazioni fornite dai vicini e non ha una visione completa della rete, il che può portare a decisioni di instradamento subottimali.

-In situazioni di fallimento (ad esempio, un nodo disconnesso), i costi verso quel nodo possono aumentare indefinitamente finché non si raggiunge la convergenza.

D. CONSIDERAZIONI FINALI

Questo progetto ha simulato con successo il protocollo di routing Distance Vector, dimostrando come i nodi aggiornano le tabelle di routing per trovare i percorsi più brevi in una rete.