Simulazione del Protocollo di Routing Distance Vector

Mattia Mencaccini

1 Introduzione

Il protocollo di routing *Distance Vector* (DV) è un algoritmo utilizzato nei sistemi di comunicazione per determinare i percorsi più brevi tra i nodi di una rete. Questo protocollo si basa sul principio della propagazione delle distanze: ogni nodo invia ai suoi vicini informazioni sulla propria tabella di routing, consentendo loro di aggiornare i propri percorsi in modo iterativo.

2 Descrizione dell'Algoritmo

L'algoritmo DV si basa sui seguenti principi:

1. Ogni nodo mantiene una tabella di routing che contiene:
   - La distanza minima verso ogni altro nodo della rete.
   - Il *next hop*, ossia il nodo attraverso cui passare per raggiungere la destinazione.
2. Ogni nodo conosce inizialmente solo i costi dei collegamenti diretti ai suoi vicini.
3. I nodi scambiano le loro tabelle di routing con i vicini in modo iterativo.
4. Ogni nodo aggiorna la propria tabella di routing in base alle informazioni ricevute, utilizzando la formula:
   $$d_{A,C} = d_{A,B} + d_{B,C}$$
   dove $d_{A,C}$ è la distanza stimata dal nodo A al nodo C, passando attraverso B.
5. Il processo continua finché le tabelle non si stabilizzano (convergenza).

3 Struttura del Codice

Il codice Python è organizzato in modo modulare per rappresentare il comportamento del protocollo DV. La struttura principale è suddivisa come segue:

3.1 Classe `Node`

La classe `Node` rappresenta un nodo della rete. Ogni istanza della classe ha i seguenti attributi:

- `name`: Il nome del nodo
- `routing_table`: Un dizionario che memorizza le distanze minime verso ogni destinazione e il *next hop*.

La classe include il metodo `update_routing_table`, che implementa la logica per aggiornare la tabella di routing in base alle informazioni ricevute dai vicini.

3.2 Funzione `simulate_network`

Questa funzione coordina la simulazione dell'algoritmo DV. Il suo scopo è:

- Eseguire iterazioni successive in cui ogni nodo aggiorna la propria tabella di routing in base alle tabelle ricevute dai vicini.
- Rilevare la convergenza del sistema, terminando il processo quando nessun nodo modifica più la propria tabella.
- Stampare le tabelle di routing intermedie per visualizzare il progresso del processo di convergenza.

3.3 Funzione `print_routing_table`

Questa funzione stampa in modo leggibile la tabella di routing di un nodo. Ogni riga contiene:

- La destinazione.
- La distanza minima per raggiungerla.
- Il *next hop*.

4 Esempio di Rete

La rete simulata è composta da 4 nodi (`A`, `B`, `C`, `D`) con la seguente matrice di adiacenza:

$$\texttt{adjacency\_matrix} = \begin{bmatrix} 0 & 1 & \infty & 4 \\ 1 & 0 & 2 & \infty \\ \infty & 2 & 0 & \infty \\ 4 & \infty & \infty & 0 \end{bmatrix}$$

5 Esempio di Output

Lo script produce l'output delle tabelle di routing dopo ogni iterazione. Di seguito mostriamo lo stato iniziale e finale.

5.1 Stato Iniziale

```
Routing Table for A:
Destination     Distance        Next Hop
A               0               A
B               1               B
D               4               D

Routing Table for B:
Destination     Distance        Next Hop
A               1               A
B               0               B
C               2               C

Routing Table for C:
Destination     Distance        Next Hop
B               2               B
C               0               C

Routing Table for D:
Destination     Distance        Next Hop
A               4               A
D               0               D
```

5.2 Stato Finale

Dopo la convergenza, le tabelle di routing sono complete:

```
Routing Table for A:
Destination     Distance        Next Hop
A               0               A
B               1               B
C               3               B
D               4               D

Routing Table for B:
Destination     Distance        Next Hop
A               1               A
B               0               B
C               2               C
D               5               A
...
```

6 Conclusioni

Lo script mostra l'applicazione pratica del protocollo DV, evidenziando il processo iterativo di aggiornamento delle tabelle di routing e la convergenza al risultato ottimale. Questo approccio può essere utilizzato come base per implementazioni più avanzate, come la gestione dinamica delle modifiche nella topologia della rete.