Simulazione del Protocollo di Routing Distance Vector

Carlo Michele Nicastro

matricola: 0001078418

email: carlo.nicastro2@studio.unibo.it

Università di Bologna - Campus di Cesena

OBIETTIVO DEL PROGETTO

L’obiettivo del progetto è stato quello di sviluppare una simulazione del protocollo **Distance Vector Routing** (DVR) utilizzando il linguaggio Python. Questo protocollo è ampiamente adottato nelle reti di comunicazione per determinare i percorsi più brevi tra i nodi di una rete. La simulazione consente di analizzare il comportamento dell’algoritmo durante l'aggiornamento delle tabelle di routing e il processo di convergenza verso uno stato stabile.

DESCRIZIONE DELL'ALGORITMO

Il protocollo Distance Vector opera attraverso i seguenti passaggi:

1. Inizializzazione della Tabella di Routing: Ogni nodo crea inizialmente una tabella di routing in cui:
   - La distanza verso se stesso è pari a 0.
   - La distanza verso tutti gli altri nodi è impostata a infinito (∞).
   - La distanza verso i nodi vicini corrisponde al costo diretto del collegamento.
2. Scambio di Informazioni: I nodi inviano periodicamente la propria tabella di routing ai vicini.
3. Aggiornamento delle Tabelle: Quando un nodo riceve una tabella di routing da un vicino, aggiorna la propria calcolando la distanza minima per ciascun nodo della rete. Questo avviene sommando il costo del collegamento al vicino con la distanza riportata nella tabella ricevuta.
4. Convergenza: L'algoritmo termina quando le tabelle di routing non subiscono più modifiche, raggiungendo così uno stato stabile.

IMPLEMENTAZIONE

La simulazione è stata implementata in Python e si basa sulle seguenti componenti principali:

- **Inizializzazione della Rete**: La funzione `initialize_routing_table` crea una tabella di routing per ciascun nodo. Ogni tabella include:
  - I costi iniziali: 0 per se stesso, il costo diretto per i nodi vicini e infinito (∞) per tutti gli altri nodi.
  - Il next hop per ogni destinazione, inizialmente impostato sul nodo vicino diretto o su `None` per i nodi non raggiungibili.
- **Aggiornamento Iterativo delle Tabelle**: La funzione `distance_vector_routing` implementa l'algoritmo principale, che:
  - Scansiona iterativamente tutti i nodi e i rispettivi vicini.
  - Calcola il costo dei percorsi considerando il passaggio attraverso ciascun vicino.
  - Aggiorna la tabella di routing solo se viene individuato un percorso più conveniente.
  - Termina l'esecuzione quando le tabelle di routing non subiscono più modifiche, segnalando la convergenza dell'algoritmo.

- **Visualizzazione**: La funzione print routing table permette di osservare lo stato della tabella di routing per ciascun nodo. Stampa i costi e i next hop, aiutando a tracciare l'evoluzione dell'algoritmo.

ESEMPIO DI SIMULAZIONE

Input:

- Nodi: Nodo A, B, C, D

- Collegamenti:

    - Nodo A <> Nodo B , costo 1
    - Nodo B <> Nodo C , costo 2
    - Nodo A <> Nodo C , costo 4
    - Nodo C <> Nodo D , costo 1

Output:

Le tabelle di routing aggiornate ad ogni iterazione e quelle finali sono come segue:

Nodo A:
    Nodo A : 0
    Nodo B : 1
    Nodo C : 3
    Nodo D : 4
Nodo B:
    Nodo A : 1
    Nodo B : 0
    Nodo C : 2
    Nodo D : 3
Nodo C:
    Nodo A : 3
    Nodo B : 2
    Nodo C : 0
    Nodo D : 1
Nodo D:
    Nodo A : 4
    Nodo B : 3
    Nodo C : 1
    Nodo D : 0

ANALISI E CONCLUSIONI

Questo simulatore dimostra il funzionamento del protocollo di routing Distance Vector in una rete semplice. Può essere esteso per supportare altre funzionalità come la prevenzione di loop o la gestione di metriche più avanzate.

• Convergenza:
L'algoritmo converge rapidamente, con un numero limitato di iterazioni.
• Efficienza:
La complessità computazionale è O(N3), dove N è il numero di nodi, poiché ogni nodo aggiorna le proprie distanze per tutti gli altri nodi.
• Robustezza:
Il protocollo gestisce bene le variazioni dei costi, ma è suscettibile al problema dei count-to-infinity, non affrontato in questa implementazione.