Simulazione di Protocollo di Routing: Distance Vector Routing in Python

Descrizione

Questo script Python permette di simulare il protocollo di routing Distance Vector Routing all'interno di una rete di nodi. Attraverso la creazione di nodi interconnessi con costi specifici, ogni nodo mantiene una tabella di routing che contiene le informazioni sulle distanze e i prossimi salti verso tutte le altre destinazioni nella rete. I nodi scambiano periodicamente le loro tabelle di routing con i nodi vicini, consentendo l'aggiornamento dinamico delle rotte basato sulle informazioni ricevute. Utilizzando l'algoritmo di Bellman-Ford, lo script calcola iterativamente le rotte più brevi fino a raggiungere la convergenza, ovvero quando nessun nodo apporta ulteriori modifiche alle proprie tabelle di routing. Al termine della simulazione, lo script stampa le tabelle di routing finali per ogni nodo, mostrando i percorsi ottimali determinati per ogni destinazione nella rete.

Requisiti

- Python 3.x
- Libreria standard di Python:
    - Copy

Struttura dello script

Per simulare il protocollo di routing, definisco due classi principali Node e Network.

La classe **Node** rappresenta un nodo individuale nella rete che mantiene nome, vicinanze, cioè l'elenco dei nodi adiacenti e relativi costi, e la tabella di routing.

Node ha gli attributi:

- *name* stringa che rappresenta il nome,
- *neighbors* un dizionario (dict) con struttura *{nome_vicino: costo_link}*,
- *routing_table* dizionario con struttura *{destinazione: (costo, next_hop)}*

e metodi principali:

- *add_neighbor(neighbor, cost)* Aggiunge un nodo vicino con il costo specificato e aggiorna la tabella di routing,
- *initialize_routing_table(all_nodes)* Inizializza la tabella di routing per tutte le destinazioni nella rete,
- *send_routing_table* Restituisce una copia della tabella di routing per inviarla ai vicini

- *update_routing_table(neighbor, neighbor_table)* Aggiorna la tabella di routing basandosi sulle informazioni ricevute da un vicino,
- *print_routing_table* Stampa la tabella di routing in modo leggibile.

La classe **Network** rappresenta l'intera rete composta da più nodi. Gestisce l'aggiunta di nodi e link, l'inizializzazione delle tabelle di routing e l'esecuzione dell'algoritmo di routing.

Network ha l'attributo:

- *nodes* dizionario con struttura *{nome_nodo: oggetto_Node}*

e metodi principali:

- *add_node(node_name)* Aggiunge un nodo alla rete se non esiste già,
- *add_link(node1, node2, cost)* Aggiunge un collegamento bidirezionale tra due nodi con il costo specificato,
- *initialize_routing_tables* Inizializza le tabelle di routing per tutti i nodi nella rete,
- *distance_vector_routing* Esegue l'algoritmo di Distance Vector Routing fino alla convergenza,
- *print_final_routing_tables* Stampa le tabelle di routing finali per tutti i nodi nella rete.

Simulazione

Per testare il codice, nel codice è definita il metodo *main* dove definisco:

```
network = Network()
```

che rappresenta l'intera rete di nodi e che gestisce l'aggiunta dei nodi, dei collegamenti e l'esecuzione dell'algoritmo di routing.

Poi definisco la topologia della rete aggiungendo i nodi:

```
network.add_link('A', 'B', 1)
network.add_link('A', 'C', 4)
network.add_link('B', 'C', 2)
network.add_link('B', 'D', 5)
network.add_link('C', 'D', 1)
```

dove ad ogni chiamata di *add_link* assicura che entrambi i nodi coinvolti nel collegamento siano presenti nella rete e aggiorna le rispettive tabelle di routing iniziali con i costi dei collegamenti diretti.

Eseguo poi il protocollo di routing con:

```
network.distance_vector_routing()
```

dove viene avviato l'algoritmo di Distance Vector Routing.

Infine, dopo aver raggiunto la convergenza, viene stampata la tabella di routing finale per ogni nodo nella rete che permette di osservare come i nodi in una rete scambiano informazioni e aggiornano le proprie tabelle di routing per determinare i percorsi più efficienti verso tutte le destinazioni.

Utilizzo

1. Salvare lo script in una certa directory,
2. Aprire un terminale e navigare fino alla cartella in cui il file è salvato,
3. Eseguire lo script con 'python3 routing_protocol.py' (eventualmente cambiando il nome dello script se è stato salvato con un altro nome)

Dopo questi passaggi il programma inizierà l'esecuzione e stamperà le tabelle di routing per ogni nodo ad ogni iterazione fino a raggiungere la convergenza.