Progetto Python: Simulazione di Protocollo di Routing

MEMBRI DEL GRUPPO:

- Bartolini Riccardo / riccardobartolini@studio.unibo.it / Matricola: 0001068901
- Fabbri Gianmarco / gianmarcofabbri3@studio.unibo.it / Matricola: 0001069003

Introduzione

Il progetto simula un semplice protocollo di routing basato sull'algoritmo Distance Vector Routing. L'obiettivo è implementare la logica di aggiornamento delle tabelle di routing tra nodi in una rete, calcolando le rotte più brevi tra di essi. La simulazione converge quando tutte le tabelle di routing diventano stabili.

Struttura del Codice

Il progetto è suddiviso in tre moduli principali:

1. NODE.PY

Contiene la classe Node, che rappresenta un nodo nella rete. Ogni nodo ha:
- Una lista di vicini diretti con il costo associato.
- Una tabella di routing che memorizza:
- La destinazione.
- Il costo per raggiungere la destinazione.
- Il prossimo hop verso la destinazione.

Metodi:

- add_neighbor(neighbor, cost)
  Aggiunge un vicino diretto al nodo, aggiornando la tabella di routing.

- remove_neighbor(neighbor)
  Seppur nella simulazione non venga mostrato il caso di perdita di connessione abbiamo comunque offerto questo metodo che mostra il comportamento e l'aggiornamento della tabella di routing di questo scenario.

- update_routing_table()
  Aggiorna la tabella di routing basandosi sulle informazioni ricevute dai vicini. Restituisce True se la tabella è stata modificata, False in caso contrario.

- print_routing_table()
  Stampa la tabella di routing del nodo in formato leggibile.

2. NETWORK.PY

Contiene un'unica funzione:

- simulate_network(nodes).
  Gestisce il processo iterativo in cui ogni nodo aggiorna la propria tabella di routing basandosi sulle informazioni ricevute dai vicini.
  Durante ogni iterazione, le tabelle di routing aggiornate vengono stampate per monitorare l'evoluzione della rete.
  La simulazione termina quando tutte le tabelle diventano stabili, indicando che ogni nodo ha calcolato i percorsi più brevi verso tutte le destinazioni. L'output viene salvato in un file di testo per essere visualizzato o ulteriori analisi.

3. MAIN.PY

È il punto di ingresso del programma. Configura i nodi, definisce le connessioni tra di essi e avvia la simulazione.

L'algoritmo Distance Vector Routing si basa sui seguenti principi:

1. Inizializzazione, ogni nodo conosce solo i costi per raggiungere i suoi vicini diretti.

2. Scambio di informazioni, ad ogni iterazione, i nodi condividono le proprie tabelle di routing con i vicini.

3. Aggiornamento delle tabelle, per ogni destinazione ricevuta dai vicini, il nodo valuta se:
   - Esiste un percorso più economico.
   - È necessario aggiornare la tabella di routing.

4. Convergenza, la simulazione si ferma quando le tabelle di routing diventano stabili, ovvero non si verificano ulteriori aggiornamenti.

Simulazione

Nell'esempio fornito, la rete è costituita da quattro nodi: A, B, C e D. Le connessioni dirette tra i nodi sono definite con i seguenti costi:

- A → B (1)
- A → C (5)
- B → C (2)
- B → D (4)
- C → D (1)

Il programma esegue iterazioni fino alla convergenza:
Alla prima iterazione (Iterazione 0), ogni nodo conosce solo i vicini diretti.
Nelle iterazioni successive: i nodi iniziano a scoprire percorsi indiretti verso altre destinazioni, aggiornando gradualmente le loro tabelle di routing. Quando le tabelle non cambiano più, la simulazione si ferma.

OUTPUT

L'output della simulazione è costituito dalle tabelle di routing aggiornate a ogni iterazione, salvate in un file di testo.

Di seguito un esempio dimostrativo di una tabella di routing generata durante la simulazione.
Tabella di routing per il nodo A:

Destinazione	Costo	Prossimo Hop
A	0	A
B	1	B
C	3	B
D	5	B

Conclusioni

La simulazione dimostra come il protocollo Distance Vector Routing consenta ai nodi di calcolare percorsi ottimali in modo distribuito, senza la necessità di una visione globale della rete. Ogni nodo utilizza esclusivamente informazioni locali scambiate con i vicini diretti per costruire progressivamente una tabella di routing completa, che rappresenta il costo minimo e il percorso ottimale verso ogni altra destinazione nella rete.

Il progetto mette in evidenza i seguenti aspetti chiave del protocollo:

- Semplicità dell'algoritmo:
  Il Distance Vector Routing è un algoritmo basato su un meccanismo iterativo di scambio di informazioni tra nodi. Ogni nodo aggiorna la propria tabella di routing confrontando i costi ricevuti dai vicini con i propri. Questo approccio distribuito è facile da implementare e si presta bene a reti decentralizzate.

- Convergenza progressiva:
  La simulazione mostra che sono necessarie più iterazioni affinché le informazioni si propagano in tutta la rete. Durante ogni iterazione, le tabelle di routing si avvicinano progressivamente alla stabilità, poiché ogni nodo aggiorna i percorsi basandosi sulle informazioni più recenti ricevute dai vicini.

- Sensibilità alle modifiche dinamiche:
  La rete è influenzata da eventi dinamici come aggiunte, rimozioni di collegamenti o modifiche nei costi. In tali situazioni, i nodi sono in grado di adattare le proprie tabelle di routing in modo iterativo, ma la convergenza può richiedere più tempo o generare temporaneamente percorsi sub-ottimali. Questo dimostra la flessibilità del protocollo ma sottolinea anche i limiti in termini di rapidità di adattamento.

- Scalabilità:
  Sebbene il protocollo funzioni bene per reti di dimensioni moderate, un numero elevato di nodi o cambiamenti frequenti nella rete può aumentare significativamente il numero di iterazioni necessarie per la convergenza. Seppur sia un algoritmo efficace e semplice da implementare, ideale per reti decentralizzate, la convergenza non è immediata e la sensibilità a cambiamenti frequenti nella rete rappresentano fattori critici da considerare in applicazioni reali.