Relazione sulla Simulazione di Protocollo di Routing

Corso di Reti di Telecomunicazione - A.A. 2024/2025

Elisa Yan - 0001070882
elisa.yan@studio.unibo.it

11 dicembre 2024

1 Introduzione

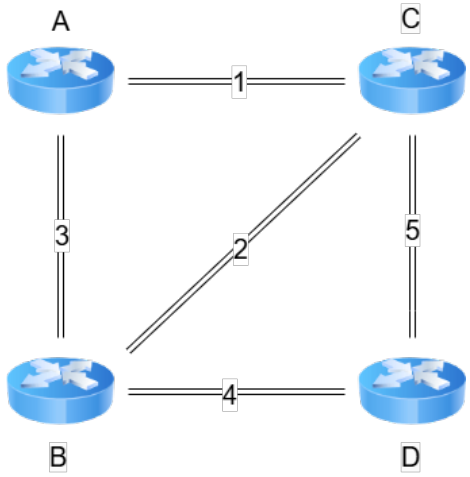

Figura 1: Esempio di distance vector routing

Tale progetto, richiede di implementare una simulazione di un protocollo di routing semplice, ispirato al protocollo Distance Vector Routing, utilizzando il linguaggio Python.

La rete è composta da diversi nodi ed è modellata con collegamenti bidirezionali, ognuno con un costo associato. Il sistema si concentra sulla simulazione dell'aggiornamento dinamico delle tabelle di routing che viene ottimizzata sulla base delle informazioni condivise. Ogni nodo del sistema rappresenta un dispositivo nella rete, come un router. Ognuno possiede una tabella di routing

che viene utilizzata per mantenere le informazioni sui costi minimi necessari per raggiungere gli altri nodi.

2 Funzionamento del sistema

Inizialmente, ogni nodo conosce solo il costo per raggiungere i propri vicini diretti, con un costo di zero per sé stesso e infinito per le altre destinazioni.

```
Initial Routing Tables:
Routing table for Node A:
  A --> A (Cost: 0, Next Hop: None)
  A --> B (Cost: inf, Next Hop: None)
  A --> C (Cost: inf, Next Hop: None)
  A --> D (Cost: inf, Next Hop: None)

Routing table for Node B:
  B --> A (Cost: inf, Next Hop: None)
  B --> B (Cost: 0, Next Hop: None)
  B --> C (Cost: inf, Next Hop: None)
  B --> D (Cost: inf, Next Hop: None)

Routing table for Node C:
  C --> A (Cost: inf, Next Hop: None)
  C --> B (Cost: inf, Next Hop: None)
  C --> C (Cost: 0, Next Hop: None)
  C --> D (Cost: inf, Next Hop: None)

Routing table for Node D:
  D --> A (Cost: inf, Next Hop: None)
  D --> B (Cost: inf, Next Hop: None)
  D --> C (Cost: inf, Next Hop: None)
  D --> D (Cost: 0, Next Hop: None)
```

Figura 2: Tabella di routing iniziale

A intervalli regolari, ogni nodo comunica la propria tabella di instradamento aggiornata ai vicini. Una volta ricevuto, le informazioni vengono confrontate con quelle già presenti nella propria tabella. Se vengono trovati più percorsi, si utilizza quello più economico verso la destinazione e la tabella di instradamento viene aggiornata di conseguenza.

3 Implementazione del codice

La struttura del programma è stata suddivisa in due classi principali: `Node` e `Network`. Tale separazione delle classi facilita la gestione del codice. La classe `Network` gestisce gli aspetti globali della rete, mentre `Node` si occupa del comportamento locale di ogni nodo.

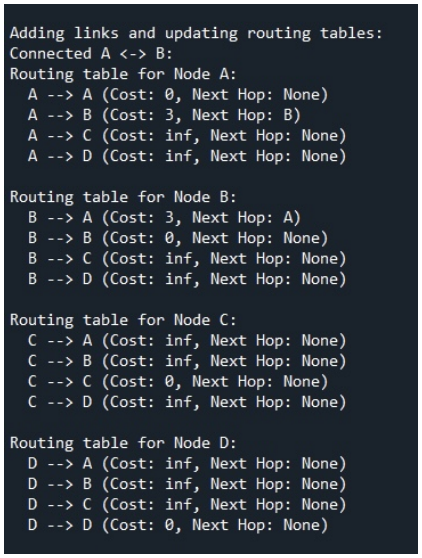

Figura 3: Connessione dei nodi A-B

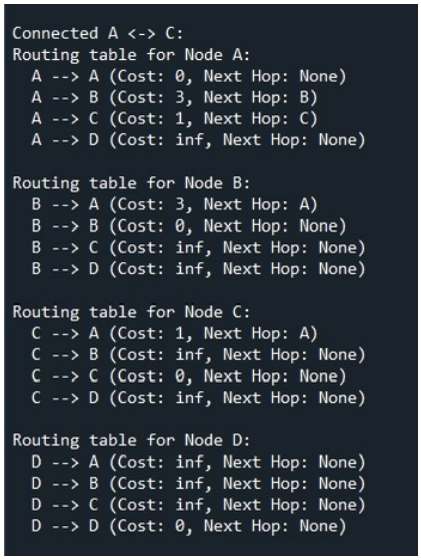

Figura 4: Connessione dei nodi A-C

3.1 Node

La classe Node rappresenta un nodo della rete e gestisce la sua tabella di routing, che contiene i costi e i prossimi hop verso ciascun nodo.

In particolare, i principali metodi sono:

- add_neighbor(), il quale aggiunge un nodo vicino alla lista dei vicini del nodo corrente, inserendo anche il corrispettivo costo,
- il metodo `update_routing table()`, aggiorna la tabella di routing del nodo basandosi sulle informazioni ricevute dai suoi vicini. Per ogni vicino, il nodo verifica i percorsi verso tutte le destinazioni nella sua tabela di routing. Se trova un percorso più conveniente allora aggiorna modifica la propria tabella con i costi aggiornati.

3.2 Network

La classe Network rappresenta l'intera rete, si occupa di collegare i nodi tra di loro e simula il processo di scambio di informazioni di routing tra di essi. Questa classe fornisce metodi per aggiungere nodi, creare delle connessioni tra i nodi e simulare il comportamento del protocollo Distance Vector Routing. I principali metodi sono:

- `add_node()`, il quale permette di aggiugere un nuovo nodo alla rete,
- `add_link()`, crea un collegamento bidirezionale tra due nodi con un costo specificato,

- `simulate routing()`, ha lo scopo di modellare il comportamento iterativo e distribuito dell'algoritmo. Se durante un'iterazione non viene più aggiornato le informazioni in nessun nodo, la simulazione viene interrotta in anticipo.

4 Esecuzione dell'applicazione

Per eseguire l'applicazione, è necessario avere Python installato sul proprio sistema. Una volta effettuata l'installazione, si deve posizionare nella directory in cui si trova il file dello script e avviarlo utilizzando il seguente comando:

```
python ./dv_routing_simulator.py
```

Una volta eseguito il programma verrà stampato sul terminale le seguenti informazioni: le tabelle di routing iniziali per ogni nodo e gli aggiornamenti delle tabelle di routing dopo ogni iterazione.

5 Difficoltà incontrate

Durante lo sviluppo del progetto, sono stati affrontati problemi come l'inizializzazione corretta delle tabelle di instradamento, la convergenza del protocollo e la gestione dei possibili cicli di instradamento.

6 Lavori futuri

Attualmente, il sistema implementa una versione semplificata del protocollo Distance Vector Routing, ma ci sono molte possibilità di miglioramento. Tra queste, si potrebbero aggiungere il supporto per i thread per simulare una rete in cui i nodi operano in modo asincrono e indipendente, e una visualizzazione grafica interattiva della rete per facilitarne l'analisi e la comprensione.