

Progetto Laboratorio Reti di Telecomunicazione

Traccia 2 - Simulazione di Protocollo di Routing

Marilia Merendi - matricola 0001071190
marilia.merendi@studio.unibo.it

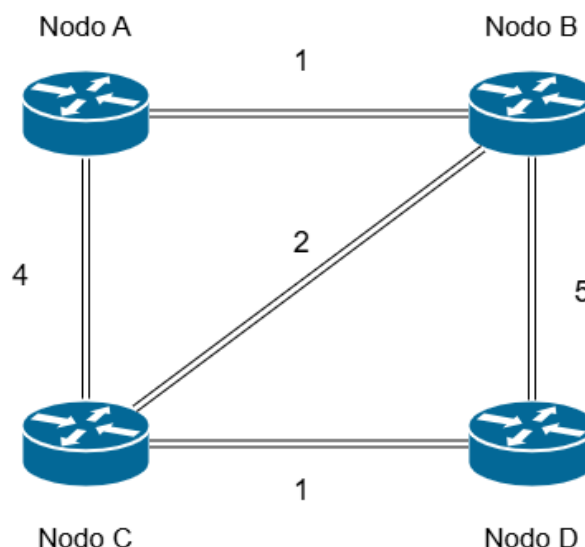
Funzionamento del sistema

Questo progetto simula tramite un semplice script Python il Distance Vector Routing: ogni nodo della rete ha una tabella di routing contenente la distanza più breve verso ogni altro nodo e lo script si occupa dell'aggiornamento di routing tra i nodi delle rete e del calcolo delle rotte più brevi.

Struttura della Rete

La rete è composta da un insieme di nodi interconnessi tramite collegamenti diretti. Ogni collegamento ha una distanza associata, che può essere vista come un "peso", ovvero una metrica che determina il costo di trasferimento dati da un nodo a un altro.

In questo progetto, è stata definita una rete composta da quattro nodi, A, B, C e D, con le seguenti connessioni e distanze rappresentate tramite un **grafo pesato**:



nota: per semplicità si è scelto di utilizzare archi non-orientati

Classe Node

Ogni nodo è rappresentato dalla classe **Node**, che include:

- Un **nome** identificativo.
- Un **dizionario di vicini** dove ogni vicino è associato a una **distanza**.
- Una **tabella di routing**, che inizialmente contiene solo le distanze verso se stesso (distanza 0) e verso i vicini diretti.

Il sistema opera tramite queste due funzioni principali:

- **update_routing_table()** : si occupa di aggiornare la tabella di routing del nodo in base alle tabelle ricevute dai vicini. Se una nuova rotta è più corta, la tabella di routing viene aggiornata.
-
- **distance_vector_routing()** : simula il funzionamento dell'algoritmo Distance Vector Routing. Ogni nodo invia la propria tabella ai vicini e aggiorna la sua tabella con i dati ricevuti. La funzione ripete questo processo per un numero prefissato di iterazioni. Durante ogni iterazione, ogni nodo invia la propria tabella di routing agli altri nodi, e le tabelle vengono aggiornate. Al termine di ogni iterazione, la funzione stampa le tabelle di routing dei nodi per monitorare il progresso dell'algoritmo.

Esecuzione dell'Algoritmo

L'algoritmo viene eseguito per un totale di 5 iterazioni. Ogni iterazione simula uno scambio di informazioni di routing tra i nodi, durante il quale le tabelle vengono aggiornate. Di seguito sono riportati i risultati per ciascun nodo durante l'esecuzione dell'algoritmo.

Tabella di Routing (Esempio per Nodo A)

All'inizio, la tabella di routing del nodo A è la seguente:

DESTINAZIONE	DISTANZA
A	0
B	1
C	3
D	5

Dopo ogni iterazione, la tabella viene aggiornata. Alla fine delle 5 iterazioni, la tabella di routing del nodo appare così:

DESTINAZIONE	DISTANZA
A	0
B	1
C	3
D	4

Risultati delle Iterazioni

Durante le 5 iterazioni, la rete converge verso una **configurazione stabile**. Dopo la prima iterazione, i nodi iniziano a scambiare informazioni sulle rotte. Successivamente, le tabelle di routing vengono aggiornate, con i nodi che trovano le rotte più brevi verso i destinatari.

Alla fine, le tabelle di routing si stabilizzano e non ci sono più aggiornamenti. Ogni nodo ha determinato la distanza minima per raggiungere gli altri nodi della rete.

Discussione dei Risultati

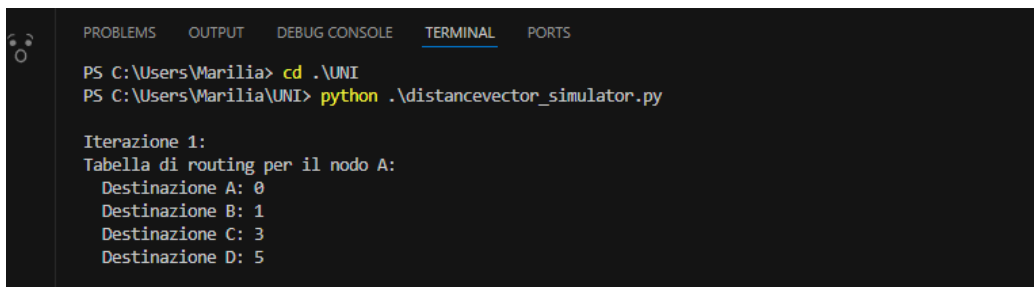
Il protocollo DVR converge rapidamente in reti di **piccole dimensioni**.

Tuttavia, in reti più grandi o con topologie complesse, l'algoritmo potrebbe risentire di problemi come il **count-to-infinity**. Andrebbero introdotti meccanismi che qui non sono presenti per semplicità di esecuzione, quali la **limitazione del numero di aggiornamenti** o l'introduzione di **vettori di distanza** con **poison reverse** e **split horizon**.

Requisiti per eseguire il codice

Per il corretto funzionamento del programma, basta lanciare lo script da terminale:

```
python .\distancevector_simulator.py
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Marilia> cd .\UNI
PS C:\Users\Marilia\UNI> python .\distancevector_simulator.py

Iterazione 1:
Tabella di routing per il nodo A:
  Destinazione A: 0
  Destinazione B: 1
  Destinazione C: 3
  Destinazione D: 5
```