Simulazione di Protocollo di Routing: Distance Vector Routing

Alessandro Ambrogiani
alessandr.ambrogian4@studio.unibo.it
0001080493

November 20, 2024

1 Introduzione

Questa relazione descrive il progetto di simulazione del protocollo di routing Distance Vector Routing in Python. L'obiettivo del progetto è simulare il funzionamento del protocollo attraverso una rete di nodi che condividono informazioni di routing per determinare i percorsi più brevi verso tutti gli altri nodi della rete.

2 Obiettivi

L'obiettivo principale è implementare un programma Python che:

- Gestisca la costruzione di una rete di nodi.
- Implementi la logica di aggiornamento delle tabelle di routing per il protocollo Distance Vector.
- Mostri le tabelle di routing finali per ogni nodo.

3 Struttura del Codice

Il programma è strutturato attorno alla classe `Nodo`, che rappresenta ogni nodo della rete. Ogni istanza della classe ha una tabella di routing, una lista di vicini e dei metodi per gestire gli aggiornamenti.

3.1 Classe Nodo

La classe `Nodo` contiene i seguenti attributi e metodi:

- `nome`: identificativo del nodo.

- `tavola_routing`: un dizionario che memorizza le rotte e le loro distanze.
- `vicini`: un dizionario che memorizza le distanze verso i nodi vicini.

3.1.1 Metodo `__init__`

Il metodo `__init__` inizializza un nodo con un nome specifico, la propria tabella di routing e un dizionario dei vicini. Il nodo conosce inizialmente solo la distanza verso se stesso.

```
class Nodo:
    def __init__(self, nome):
        self.nome = nome
        self.tavola_routing = {nome: 0}
        self.vicini = {}
```

3.1.2 Metodo `aggiungi_vicino`

Il metodo `aggiungi_vicino` consente di aggiungere un nodo vicino con una distanza specificata. Aggiorna la lista dei vicini e inizializza la tavola di routing del nodo corrente per includere il vicino.

```
def aggiungi_vicino(self, vicino, distanza):
    self.vicini[vicino] = distanza
    self.tavola_routing[vicino.nome] = distanza
```

3.1.3 Metodo `invia_aggiornamenti`

Il metodo `invia_aggiornamenti` permette al nodo di inviare la propria tabella di routing a tutti i vicini. Questo metodo simula la fase di invio degli aggiornamenti nel protocollo Distance Vector.

```
def invia_aggiornamenti(self):
    for vicino in self.vicini:
        vicino.ricevi_aggiornamento(self.nome, self.tavola_routing)
```

3.1.4 Metodo `ricevi_aggiornamento`

Il metodo `ricevi_aggiornamento` riceve la tabella di routing di un nodo vicino e aggiorna la propria tabella di routing se trova un percorso più breve. Questo è il cuore del protocollo Distance Vector, che si basa sul confronto delle distanze ricevute.

```
def ricevi_aggiornamento(self, nodo_vicino, tavola_vicino
    ):
    aggiornato = False
    distanza_vicino = self.tavola_routing[nodo_vicino]
```

```
        for destinazione, distanza in tavola_vicino.items():
            nuova_distanza = distanza_vicino + distanza
            if destinazione not in self.tavola_routing or
                nuova_distanza < self.tavola_routing[
                destinazione]:
                 self.tavola_routing[destinazione] =
                     nuova_distanza
                 aggiornato = True

        return aggiornato
```

4 Funzionamento del Programma

Il programma crea i nodi e li connette secondo una struttura di rete definita. Utilizzando il metodo `invia_aggiornamenti`, ogni nodo invia la propria tabella di routing ai vicini, che a loro volta aggiornano le proprie tabelle di routing usando il metodo `ricevi_aggiornamento`. Questo processo continua fino alla convergenza, ossia finché le tabelle di routing non vengono più aggiornate.

5 Output del Programma

Alla fine della simulazione, viene stampata la tavola di routing per ciascun nodo, mostrando la rotta più breve verso ogni altro nodo. Di seguito un esempio di output per una rete con quattro nodi (`A`, `B`, `C`, `D`).

```
Tavola di routing per A:
  A: 0
  B: 1
  C: 3
  D: 4

Tavola di routing per B:
  A: 1
  B: 0
  C: 2
  D: 3

Tavola di routing per C:
  A: 3
  B: 2
  C: 0
  D: 1

Tavola di routing per D:
```

```
A: 4
B: 3
C: 1
D: 0
```

6 Conclusioni

Il programma dimostra con successo il funzionamento del protocollo Distance Vector Routing, permettendo a ciascun nodo di apprendere la rotta più breve verso tutti gli altri nodi in una rete. Il processo di scambio delle tabelle di routing con i vicini permette una convergenza rapida verso il percorso minimo, rendendo questo protocollo efficace per reti di piccole e medie dimensioni.