

# Simulazione di Protocollo di Routing

Morri Lorenzo

10 dicembre 2024

# Indice

1	Obiettivo	3
2	Funzionamento	3
3	Probleme soluzioni	3
4	Conclusioni	4
5	Codice	4

# 1 Obiettivo

In questo progetto ho implementato una simulazione del protocollo Distance Vector Routing. Ogni nodo della rete scambia informazioni con i propri vicini e aggiorna iterativamente la propria tabella di routing per calcolare i costi minimi verso le altre destinazioni.

Nel codice ho utilizzato due classi principali: **Nodo**, che rappresenta un dispositivo della rete, e **Rete**, che gestisce l'insieme dei nodi e delle loro connessioni. Ogni nodo mantiene una tabella di routing con costi e prossimo hop per ogni destinazione, mentre la rete collega i nodi, simula gli aggiornamenti e verifica che sia stabile.

# 2 Funzionamento

Ogni nodo inizia con una tabella di routing base, dove conosce solo se stesso. Quando si collega a un altro nodo, aggiorna la tabella. Durante la simulazione, ogni nodo scambia informazioni con i propri vicini e aggiorna la tabella calcolando i percorsi minimi. Questo processo continua fino a quando le tabelle non vengono più modificate.

Ad esempio, se nella rete ci sono i nodi A, B, C e D con collegamenti e costi definiti, al termine della simulazione ogni nodo avrà il percorso migliore per raggiungere qualsiasi destinazione, indicando costo che prossimo hop da seguire.

# 3 Problemie soluzioni

Durante lo sviluppo del codice sono emersi alcuni problemi:

- **Esistenza dei nodi:** il metodo per collegare i nodi non verificava se i nodi specificati esistessero già e questo poteva portare a errori durante l'esecuzione. Ho risolto introducendo un controllo che solleva un'eccezione se uno dei nodi non è presente nella rete.
- **Propagazione inefficiente:** durante la simulazione, alcuni nodi continuavano a propagare dati già aggiornati, allungando il processo. Per risolvere, ho aggiunto un modo che interrompe la simulazione non appena le tabelle di routing non vengono modificate ulteriormente.

## 4 Conclusioni

Al termine della simulazione, il programma stampa le tabelle di routing finali per ciascun nodo. Ogni tabella mostra il costo minimo per raggiungere ogni destinazione e il prossimo hop da seguire. I risultati confermano che l'algoritmo lavora correttamente.

## 5 Codice

```
1 # Simulazione del protocollo di Routing
2
3 class Nodo:
4     def __init__(self, nome):
5         # Inizializza un nodo con il proprio nome e una
6         # tabella di routing
7         # La tabella di routing parte con un unico elemento:
8         # se stesso con costo 0
9         self.nome = nome
10        self.tabella_di_routing = {nome: (0, nome)} # {
11            destinazione: (costo, prossimo_hop)}
12
13    def aggiorna_tabella_di_routing(self, vicino,
14        tabella_vicino):
15        # Aggiorna la tabella di routing in base ai dati
16        # ricevuti dal nodo vicino
17        aggiornato = False
18        for destinazione, (costo_destinazione, prossimo_hop)
19        in tabella_vicino.items():
20            # Calcola il nuovo costo passando attraverso il
21            # nodo vicino
22            nuovo_costo = vicino.tabella_di_routing[self.nome
23                ][0] + costo_destinazione
24            # Aggiorna solo se la destinazione non e'
25            # presente o il nuovo costo e' inferiore
26            if destinazione not in self.tabella_di_routing or
27            nuovo_costo < self.tabella_di_routing[
28                destinazione][0]:
29                self.tabella_di_routing[destinazione] = (
30                    nuovo_costo, vicino.nome)
31            aggiornato = True
32        return aggiornato
33
34    def stampa_tabella_di_routing(self):
35        # Stampa in modo leggibile la tabella di routing del
36        # nodo
37        print(f"Tabella di routing per {self.nome}:")
```

```

25     print("Destinazione\tCosto\tProssimo Hop")
26     for destinazione, (costo, prossimo_hop) in sorted(
27         self.tabella_di_routing.items()):
28         print(f"{destinazione}\t\t{costo}\t{prossimo_hop}")
29     print()
30 class Rete:
31     def __init__(self):
32         # Inizializza una rete vuota
33         self.nodi = {}
34
35     def aggiungi_nodo(self, nome_nodo):
36         # Aggiunge un nuovo nodo alla rete
37         self.nodi[nome_nodo] = Nodo(nome_nodo)
38
39     def collega_nodi(self, nome_nodo1, nome_nodo2, costo):
40         # Collega due nodi con un costo specificato
41         if nome_nodo1 not in self.nodi or nome_nodo2 not in
42             self.nodi:
43             raise ValueError("Entrambi i nodi devono esistere
44                 nella rete.")
45
46         nodo1, nodo2 = self.nodi[nome_nodo1], self.nodi[
47             nome_nodo2]
48         # Aggiorna le tabelle di routing per riflettere la
49             connessione diretta
50         nodo1.tabella_di_routing[nome_nodo2] = (costo,
51             nome_nodo2)
52         nodo2.tabella_di_routing[nome_nodo1] = (costo,
53             nome_nodo1)
54
55     def simula_aggiornamenti_routing(self):
56         # Simula iterativamente gli aggiornamenti delle
57             tabelle di routing fino a stabilita'
58         aggiornato = True
59         while aggiornato:
60             aggiornato = False
61             for nodo in self.nodi.values():
62                 for nome_vicino, (costo, _) in list(nodo.
63                     tabella_di_routing.items()): # Itera su
64                     una copia
65                     if nome_vicino != nodo.nome: # Evita di
66                         aggiornare verso se stesso
67                         vicino = self.nodi[nome_vicino]
68                         # Propaga le informazioni di routing
69                             dai vicini
70                         if nodo.aggiorna_tabella_di_routing(
71                             vicino, vicino.tabella_di_routing)

```

```

60         :
61         aggiornato = True
62
63     def stampa_tutte_tabelle_di_routing(self):
64         # Stampa le tabelle di routing di tutti i nodi della
65         rete
66         for nodo in self.nodi.values():
67             nodo.stampa_tabella_di_routing()
68
69 # Esempio di utilizzo:
70 if __name__ == "__main__":
71     rete = Rete()
72
73     # Aggiungi nodi alla rete
74     for nome_nodo in ["A", "B", "C", "D"]:
75         rete.aggiungi_nodo(nome_nodo)
76
77     # Collega i nodi con i relativi costi
78     rete.collega_nodi("A", "B", 1)
79     rete.collega_nodi("A", "C", 4)
80     rete.collega_nodi("B", "C", 2)
81     rete.collega_nodi("C", "D", 1)
82
83     # Simula gli aggiornamenti delle tabelle di routing
84     rete.simula_aggiornamenti_routing()
85
86     # Stampa le tabelle di routing finali
87     rete.stampa_tutte_tabelle_di_routing()

```