DistanceVector - Programmazione di Reti 2024

Lorenzo Bergami

December 2024

1 Introduzione

Il progetto simula il comportamento di una rete di router che utilizzano il protocollo *Distance Vector*, un metodo di routing in cui ogni nodo di rete mantiene una tabella delle distanze verso tutti gli altri router. All'inizio, ogni router ha solo la conoscenza di se stesso, ma attraverso l'iterazione e lo scambio di informazioni con i vicini, ogni router acquisisce progressivamente la conoscenza dell'intera rete. Questo processo di aggiornamento delle tabelle avviene fino a quando ogni router non conosce il percorso migliore per raggiungere ogni altra destinazione.

Il progetto è stato sviluppato in Python utilizzando la libreria `Tkinter` per la realizzazione di un'interfaccia grafica interattiva, che consente all'utente di visualizzare e monitorare il comportamento dinamico della rete e delle tabelle di routing.

2 Funzionamento dell'applicazione

L'applicazione simula il protocollo *Distance Vector* all'interno di una rete di router. Ogni router inizia con una tabella di routing che contiene solo informazioni triviali (distanza zero verso se stesso). Quando l'utente preme il pulsante *Next*, ogni router invia e riceve informazioni dai suoi vicini, aggiornando le tabelle di routing con i percorsi più brevi (con il costo minore).

2.1 Struttura della GUI

L'interfaccia grafica permette all'utente di osservare la rete simulata. È composta da:

- **Canvas**: Mostra i router come cerchi e i collegamenti come linee.
- **Logger**: Mostra le tabelle di routing aggiornate, permettendo all'utente di monitorare lo stato della rete ad ogni iterazione.

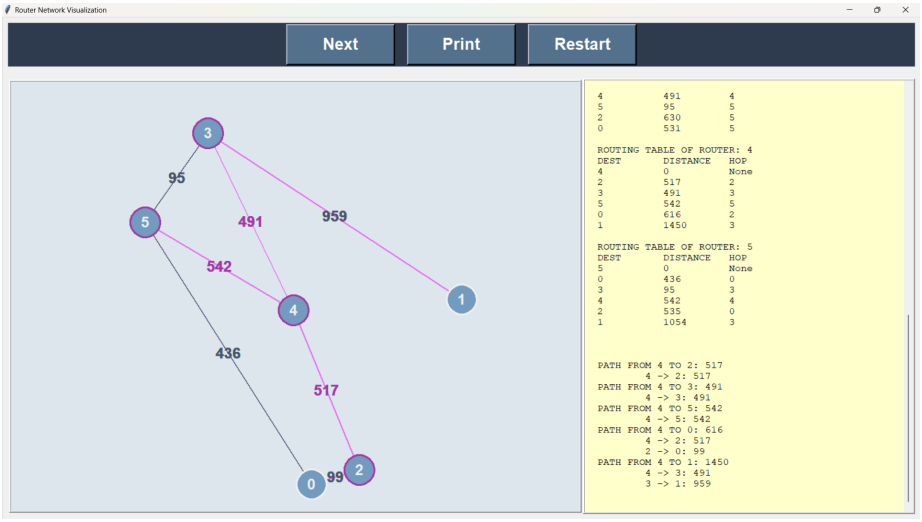

Figure 1: GUI esempio

2.2 Interazione con la GUI

L'utente può interagire con l'applicazione tramite tre pulsanti:

- **Next**: Ogni volta che l'utente preme questo pulsante, i router scambiano informazioni sui loro percorsi e aggiornano le rispettive tabelle di routing. Questo processo simula il comportamento del protocollo *Distance Vector*.
- **Print**: Questo pulsante stampa sul logger lo stato corrente delle tabelle di routing di tutti i router, permettendo di osservare i cambiamenti dopo ogni iterazione.
- **Restart**: Rende possibile ripristinare l'applicazione, mantenendo la stessa configurazione ma generando casualmente nuove posizioni per i router e i collegamenti. In questo modo è possibile osservare come la rete evolve in condizioni diverse.

Inoltre, quando un router viene cliccato, vengono evidenziati:

- Il router selezionato.
- I suoi vicini diretti e i relativi collegamenti.
- Se il router ha altri router nella sua tabella di routing, vengono evidenziati e stampati sul *logger* anche i percorsi che permettono di raggiungere questi router.

2.3 File di Configurazione

Il comportamento della rete è definito da un file di configurazione che specifica:

- Il numero di router nella rete.
- I collegamenti tra i router, generati casualmente in base ai parametri della configurazione.
- Parametri grafici come le dimensioni della finestra, i colori dei router e delle linee di connessione.

La configurazione consente di generare topologie di rete diverse ad ogni esecuzione del programma, in quanto la posizione dei router e i collegamenti vengono generati casualmente.

3 Struttura del Codice

Il programma è suddiviso in diverse classi che gestiscono vari aspetti della simulazione:

3.1 Classe Router

La classe `Router` rappresenta un singolo router nella rete. Ogni router ha una tabella di routing che contiene le informazioni sui router raggiungibili e il costo per arrivarci. Inoltre, la classe tiene traccia dei suoi vicini e dei collegamenti. Le principali funzionalità della classe includono:

- `update_dv`: aggiorna la tabella di routing in base alle informazioni ricevute dai vicini.
- `share_dv`: consente a un router di inviare la sua tabella di routing agli altri router connessi.

3.2 Classe Edge

La classe `Edge` rappresenta un collegamento tra due router. Ogni edge ha un peso che rappresenta il costo del collegamento tra i due router. Questo peso viene utilizzato per determinare i percorsi più brevi nella tabella di routing.

3.3 Classe NetworkVisualization

La classe `NetworkVisualization` è responsabile della creazione dell'interfaccia grafica. Utilizza la libreria `Tkinter` per disegnare i router e i collegamenti sulla finestra. La classe gestisce anche l'interazione dell'utente con la GUI e l'aggiornamento dinamico della visualizzazione ad ogni iterazione del protocollo.

4 Conclusioni

Il progetto ha dimostrato il funzionamento del protocollo *Distance Vector* all'interno di una rete di router simulata. L'applicazione permette di osservare come i router si aggiornano reciprocamente le loro tabelle di routing, evolvendo da una conoscenza limitata a una rete completamente connessa. Grazie alla possibilità di interagire con la rete tramite l'interfaccia grafica, l'utente può osservare chiaramente come i percorsi vengano ottimizzati iterativamente.

In futuro, il programma potrebbe essere esteso per supportare altre topologie di rete più complesse o per includere altri algoritmi di routing.