Introduzione

Ho svolto la seconda traccia, consistente nella simulazione di un protocollo di routing. In particolare, ho deciso di simulare l'algoritmo Distance Vector (DV).

Struttura e funzionamento

Router

Il file `router.py` contiene la rappresentazione di un router capace di implementare DV.

Metodi

- `add_neighbor`: aggiunge un collegamento diretto con un altro router
- `send`: invia le proprie informazioni di routing ai vicini
- `recv`: riceve le informazioni di routing da un vicino, salvandole in un buffer
- `update`: aggiorna le rotte

Attributi

I campi dei router sono così organizzati:

- `name`: stringa contenente un nome identificativo arbitrario (potrebbe rappresentare il MAC address)
- `address`: stringa contenente l'indirizzo IP (fittizio, in quanto si tratta di una simulazione)
- `neighbors`: lista contenente i router vicini
- `paths`: coppie chiavi-valori rappresentante i percorsi e quindi la tabella di routing; la chiave è il router di destinazione, il valore è una lista di 2 elementi: il primo è il gateway (vicino da contattare per raggiungere il router destinazione), il secondo la distanza stimata
- `received_paths`: buffer contenente le informazioni di routing ricevute ma non ancora processate; la struttura è la stessa di `paths`

Main

`main.py` contiene il codice per inizializzare i router e i loro collegamenti e per eseguire la simulazione vera e propria.

Setup

La parte di setup consiste nell'inizializzazione dei vari router e nella creazione dei collegamenti. La funzione `create_link()` si occupa di creare un collegamento (bidirezionale) tra due router con una certa distanza.

Simulazione

La simulazione è divisa in round (il numero di essi è il numero di router, cioè il limite superiore) ed ognuno è a sua volta composto di 3 fasi:

- output delle tabelle di routing
- scambio dei percorsi
- aggiornamento delle rotte

Viene offerta la possibilità di eseguire la simulazione manualmente tramite il flag `--manual`, altrimenti di default la simulazione viene svolta completamente in automatico (`--auto`).

Considerazioni aggiuntive

- ho utilizzato lo split horizon per risolvere parte dei problemi noti del distance vector (vedi `send()`)
- ho scelto di simulare la comunicazione tra i router utilizzando un buffer temporaneo come coda dei messaggi perchè volevo mantenere uno stato del sistema che fosse indipendente dall'ordine in cui vengono simulati gli invii dei percorsi (ad ogni "round" tutti i router inviano i percorsi ai vicini ed aggiornano le rotte solo dopo aver a loro volta ricevuto tutto)
- i collegamenti tra i router sono bidirezionali ma il programma è facilmente modificabile per permettere collegamenti unidirezionali