Informatica – Prova di laboratorio, 23 febbraio 2022

Reti opportunistiche

Negli ultimi anni si sono diffusi protocolli di distribuzione dell'informazione tramite l'uso di "Reti Opportunistiche". L'idea di base è quella di sfuttare i dispositivi mobili (smartphone) per creare reti mobili effimere capaci di diffondere messaggi/informazioni di cui non si deve garantire né la consegna, né il tempo di arrivo. Fanno parte di questa categoria di informazioni, per esempio, le pubblicità. L'uso di reti opportunistiche offre il vantaggio di alleggerire il carico delle reti di comunicazione convenzionali e consente un "targeting" più mirato basato sulla localizzazione geografica dei dispositivi.

Il meccanismo di trasferimento dell'informazione si basa sulla prossimità dei dispositivi mobili: se un dispositivo che ha una copia del messaggio transita vicino a un altro dispositivo, esso trasmette il messaggio a quel dispositivo. Il dispositivo che ha appena ricevuto una copia del messaggio, può transitare, muovendosi, vicino ad altri dispositivi e trasmettere a sua volta il messaggio a quelli e così via... .

In questo progetto cercheremo di implementare una simulazione (semplificata) della diffusione di un messaggio attraverso una rete opportunistica.

Parametri simulazione. La definizione di uno scenario richiede di definire alcuni parametri. Innanzitutto il numero di dispositivi, che indichiamo con K, abilitati al trasferimento. Un altro parametro è il numero n di *passi*: assumiamo infatti che il movimento dei dispositivi sia rilevato a tempi discreti equidistanziati a partire da un tempo 0 sino ad un tempo n. L'ultimo parametro da specificare è la distanza massima d di trasmissione tra due dispositivi. Il valore di d determina quindi se un dispositivo, diciamo w_i, che ha una copia del messaggio può trasferirlo ad un altro dispositivo w_j, con $i \neq j$: questo può accadere solo se la distanza tra i due dispositivi è $\leq d$, ovvero $\sqrt{(w_i.x - w_j.x)^2 + (w_i.y - w_j.y)^2} \leq d$, dove $(w_i.x, w_i.y)$ sono le coordinate della posizione dell'i-esimo dispositivo sul piano cartesiano.

Mosse Ad ognuno degli n passi, i dispositivi fanno uno spostamento di ampiezza 1 lungo una direzione parallela agli assi del piano cartesiano. Le *mosse* possibili sono quindi 4 e le indicheremo coi caratteri `'N','S','O'` e `'E'`, caratteri che stanno a indicare rispettivamente: fai un passo a `Nord`, a `Sud`, a `Ovest` e a `Est`. Per intenderci, se $(w_i.x, w_i.y)$ è la posizione del dispositivo i-esimo ad un certo istante e la mossa da compiere a quell'istante è `'N'`, la posizione del dispositivo al prossimo istante diventa $(w_i.x, w_i.y) \to (w_i.x, w_i.y + 1)$. Analogamente:

$$\texttt{'S'}\colon (w_i.x, w_i.y) \to (w_i.x, w_i.y - 1),$$
$$\texttt{'O'}\colon (w_i.x, w_i.y) \to (w_i.x - 1, w_i.y),$$
$$\texttt{'E'}\colon (w_i.x, w_i.y) \to (w_i.x + 1, w_i.y).$$

Gli spostamenti di un dispositivo, quindi, saranno forniti in termini di una n-upla di mosse, con n numero di *passi*, ciascun elemento della n-upla nell'insieme {`'N','S','O','E'`}.

Nota: per semplicità, saranno intere anche le coordinate della posizione iniziale dei dispositivi, così che la posizione dei dispositivi sia sempre data da una coppia di interi.

Trasferimento messaggio. Se ad un certo istante due dispositivi sono *vicini* (entro distanza d, vedi sopra) possono succedere diverse cose. Nota: da qui in avanti chiameremo *messaggero* un dispositivo che ha una copia del messaggio. **Scenario (a)** nessuno dei due dispositivi è messaggero: non succede nulla. **Scenario (b)** uno dei due dispositivi è messaggero: anche l'altro dispositvo diventa messaggero. **Scenario (c)** tutti e due dispositivi sono messaggeri: non succede nulla. È quindi chiaro che, se la proprietà *essere messaggero* fosse codificata con un valore booleano in un campo s (ad esempio $w_i.s =$ `true` se w_i è messaggero, $w_i.s =$ `false` altrimenti), allora due w_i e w_j dispositivi vicini diventano entrambe messaggeri se anche solo uno lo era. Per intenderci, in termini di codice:

```
if ( w[i].s == true || w[j].s == true ) {
    w[i].s = true;
    w[j].s = true;
}
```

Specifiche del progetto, leggete attentamente ⇒

Specifiche del progetto

La specifica dei parametri della simulazione che andremo ad implementare è fornita nel file `parametri.dat` nella cartella `/home/comune/20220223_Dati/` sulla macchina `tolab.fisica.unimi.it`. In particolare, il file contiene una sola riga con 3 dati: il primo dato è un valore **intero** che indica il numero K di dispositivi utilizzati nella simulazione; il secondo dato è un altro valore **intero** che specifca il numero n di passi della simulazione; il terzo dato è invece il valore **razionale** (`float`) che determina la distanza massima d di trasmissione del messaggio tra due dispositivi.

Il file `walkers.dat` contiene invece, riga per riga, la descrizione dello stato dei K dispositivi all'inizio della simulazione. In particolare, per ciascun dispositivo (cioè, riga del file) vengono fornite: la *posizione iniziale* (coordinate), nella forma di due valori **interi**, seguiti da n **caratteri** che specificano le n *mosse* che ciascun dispositivo farà negli n *passi* della simulazione; l'ultimo valore di ciascuna riga è ancora un valore **intero**: questo valore è 1 se all'inizio della simulazione il dispositivo è *messaggero*, 0 altrimenti.

Copiati i file `parametri.dat` e `walkers.dat` sulla vostra macchina, e definita la struttura:

```
struct walker {
   int x;       // ascissa iniziale del dispositivo
   int y;       // ordinata iniziale del dispositivo
   int n;       // numero mosse della simulazione
   char *moves; // array delle n mosse (n caratteri)
   bool s;      // true: dispositivo messaggero, false: altrimenti
};
```

1. Caricare dal file `parametri.dat` i parametri della simulazione. Stampare a video i 3 parametri letti.
2. Caricare dal file `walker.dat` la descrizione dei K dispositivi in un vettore di `walker`, allocato dinamicamente, di dimensione K. Osserviamo che per ogni dispositivo sarà necessario allocare dinamicamente il vettore delle mosse (campo `*moves`), di dimensione n pari al numero di mosse estrapolato al punto precedente, e riempirlo con gli n caratteri forniti nella corrispondente riga del file. Determinare e stampare a video:
   - **(i)** il numero di dispositivi *messaggeri* all'istante iniziale,
   - **(ii)** la posizione e la sequenza delle n mosse dei soli dispositivi *messaggeri* all'istante iniziale,
   - **(iii)** la distanza massima tra due dispositivi *messaggeri* all'istante iniziale.
3. Procedere ora con gli n passi della simulazione. Per ciascun passo eseguire, nell'ordine qui specificato, le seguenti operazioni:
   - **(i)** aggiornare la posizione di **tutti i dispositivi** usando le corrispondenze mossa $\leftrightarrow$ cambio coordinata descritte nel foglio precedente (ad esempio `'O'`: $(w_i.x, w_i.y) \to (w_i.x - 1, w_i.y)$),
   - **(ii)** determinare lo status (*messaggero/non messaggero*) di ciascun dispositivo, usando la posizione aggiornata, e la definizione di *vicino* data nella pagina precedente,
   - **(iii)** determinare e stampare a video il numero di *messaggeri*.
4. Dopo aver eseguito tutte le mosse, ovvero a simulazione terminata, determinare e stampare a video:
   - **(i)** il numero di dispositivi *messaggeri*,
   - **(ii)** la posizione finale e la sequenza delle n mosse dei soli dispositivi *messaggeri*,
   - **(iii)** la distanza massima tra due dispositivi *messaggeri*.

ATTENZIONE! Tutti i risultati stampati a video devono essere anche registrati su un file `risultati.out` e devono avere *opportune diciture*, che consentano di capire il significato di quanto stampato/registrato.

Istruzioni per la consegna del progetto *e per la copia in remoto di file e cartelle* $\Rightarrow$

Istruzioni per la consegna del progetto

Il vostro software deve essere predisposto in una cartella denominata `cognome_matricola` che deve essere copiata in `/home/comune/20220223_Risultati` sulla macchina `tolab.fisica.unimi.it`

Nella cartella `cognome_matricola` devono essere inclusi:

- un `makefile` che tramite i comandi `make compila` e `make esegui` consenta rispettivamente di compilare e di eseguire il programma,
- i file `parametri.dat` e `walkers.dat` dei dati di input del progetto,
- il file `risultati.out` prodotto dal programma,
- tutti e soli i `.C` `.cpp` `.cxx` e `.h` `.hpp` utili alla soluzione del problema.

Valutazione del progetto. La valutazione terrà conto sia della qualità dei risultati sia della struttura e dell'organizzazione del codice; per chiarire, sono graditi uso di funzioni e compilazione separata, mentre non è gradito un `main` omnicomprensivo. I progetti che non compilano o che entrano in loop dopo il lancio verranno immediatamente classificati come insufficienti.

Istruzioni per la copia in remoto di file e cartelle

Per copiare i file dati da `tolab` al vostro computer usate il comando

```
scp  username@tolab.fisica.unimi.it:<sorgente>   <destinazione>
```

Per copiare la cartella contenente il vostro svolgimento su `tolab` usate il comando

```
scp -r  <cartella>   username@tolab.fisica.unimi.it:<destinazione>
```

ATTENZIONE!

I docenti NON forniranno chiarimenti o indicazioni in merito all'uso del comando `scp` durante l'intero svolgimento della prova.