Gossiping

Progetto di
"Calcolo della Probabilità e
Statistica Matematica"

di Cavenaghi Mattia
matricola 640926

Indice:

1.Schema della rete

La rete scelta per la simulazione e, riportata nell'*immagine 1*, è costituita da un grafo, avente 14 nodi, collegati tra di loro tramite degli archi orientati. Come possiamo notare, tali archi sono, ad esclusione di 8-11 e 10-12, unidirezionali. Il nodo *sorgente* ed il nodo *destinatario*, sono stati etichettati ed evidenziati con colori differenti: rispettivamente verde e rosso.

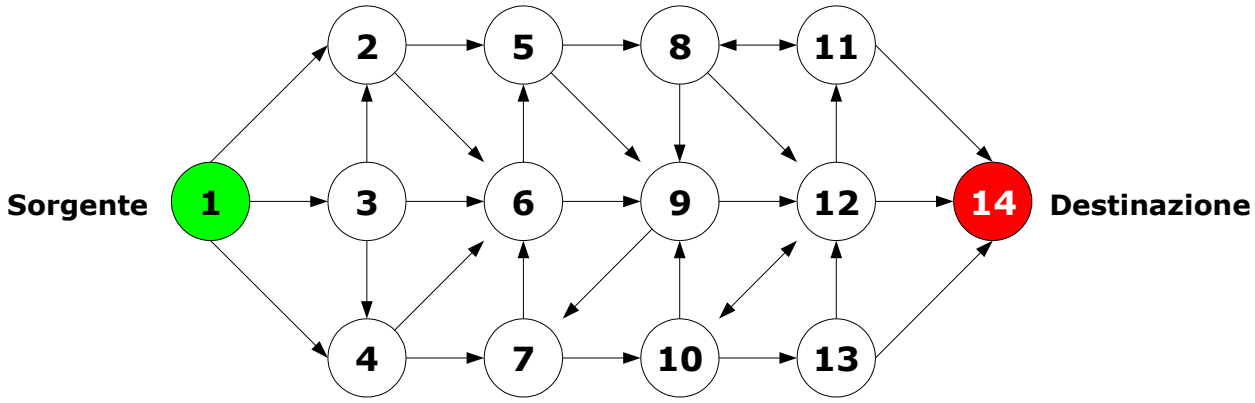

Immagine 1: Topologia della rete utilizzata per la simulazione

Ogni nodo, ha un numero minimo di 2 archi e, può averne un massimo 3, collegati ad altrettanti nodi: la numerazione di tali archi, è stata assegnata in senso orario: nella seguente *immagine 2*, ne viene riportato un esempio:

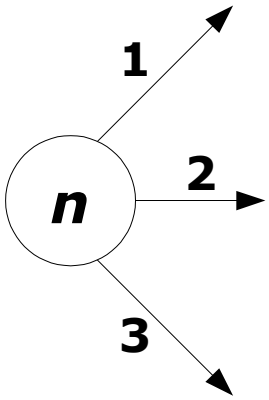

Immagine 2: Esempio di numerazione degli archi

Per meglio comprendere il sistema di numerazione, ho riportato nella tabella 1, i *primi vicini*, ossia dato un nodo di partenza, vengono riportati tutti i nodi raggiungibili tramite un solo hop:

Numero di nodo	Arco 1	Arco 2	Arco 3
1	2	3	4
2	5	6	
3	2	6	4
4	6	7	
5	8	9	
6	5	9	

Numero di nodo	*Arco 1*	*Arco 2*	*Arco 3*
7	6	10	
8	11	12	9
9	12	7	
10	9	12	13
11	14	8	
12	11	14	10
13	12	14	
14			

Tabella 1: Tabella dei nodi "primi vicini"

Questa tabella, è stata inoltre memorizzata nel file `tabella_nodi.xls` (formato Excel), poiché necessaria per la corretta esecuzione della simulazione, attraverso il programma da me realizzato (per il codice e l'analisi dello stesso, si rimanda al capitolo 5).

2.Esempio di cammino realizzabile

Di seguito, ho riportato un esempio di cammino che un pacchetto, tramite un processo di generazione casuale dei nodi, può intraprendere:

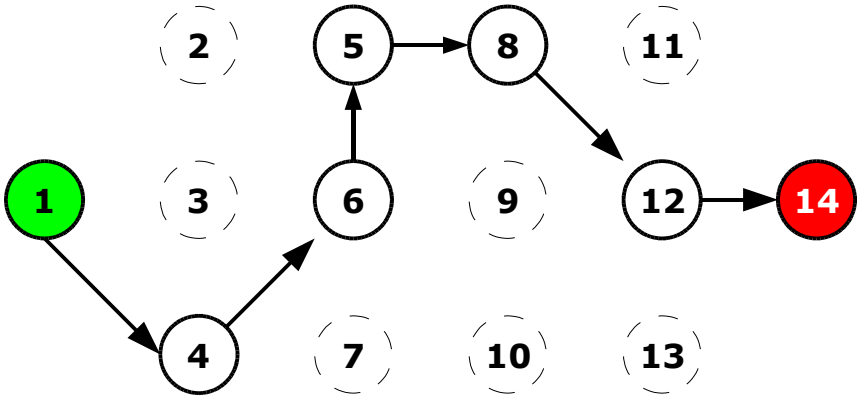

Immagine 3: Esempio di cammino

Il pacchetto, come accennato in precedenza, parte dal nodo numero 1 e, compiendo 6 hop (il numero minimo di hop richiesto è 4 ma, guardando la topologia della rete simulata, si può notare che sono necessari come minimo 5 hop, per compiere un cammino completo), arriva a destinazione (nodo 14).

Nodo di partenza	***Nodi disponibili***	***Nodo di arrivo***	***N. hop effettuati***
1	2, 3, 4	4	1
4	6, 7	6	2
6	5, 9	5	3
5	8, 9	8	4
8	9, 11, 12	12	5
12	11, 14	**14**	6
Cammino effettuato		***N. totali di hop***	
1, 4, 6, 5, 8, 12, 14		6	

Tabella 2: Esempio di cammino

Durante la computazione, è possibile che si verifichino dei loop tra più nodi (*immagine 4*) ma, durante la progettazione della rete ed eseguendo diverse simulazioni, sono riuscito ad evitare i fenomeni di *loop cieco* (*immagine 5*).

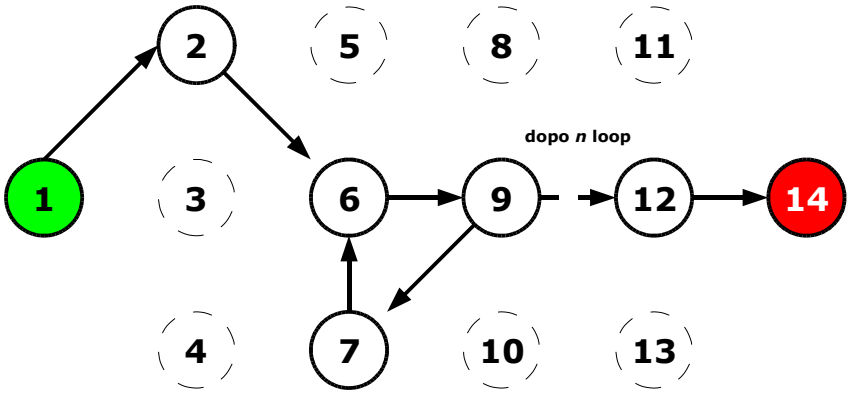

Immagine 4: Esempio di loop tra più nodi

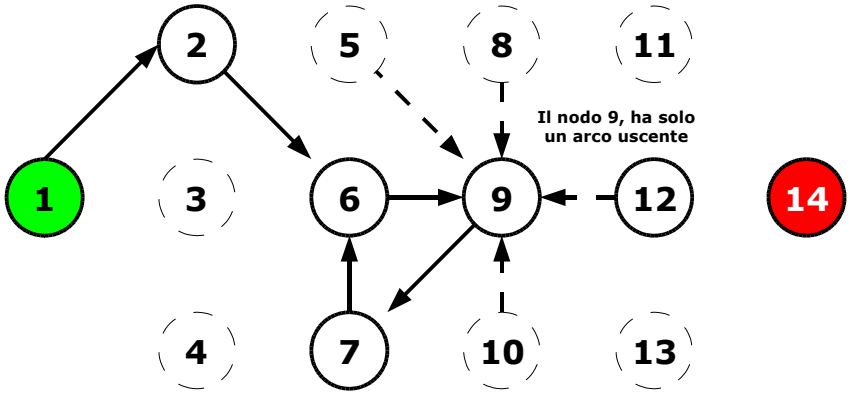

Immagine 5: Esempio di loop cieco

3.L'istogramma della simulazione

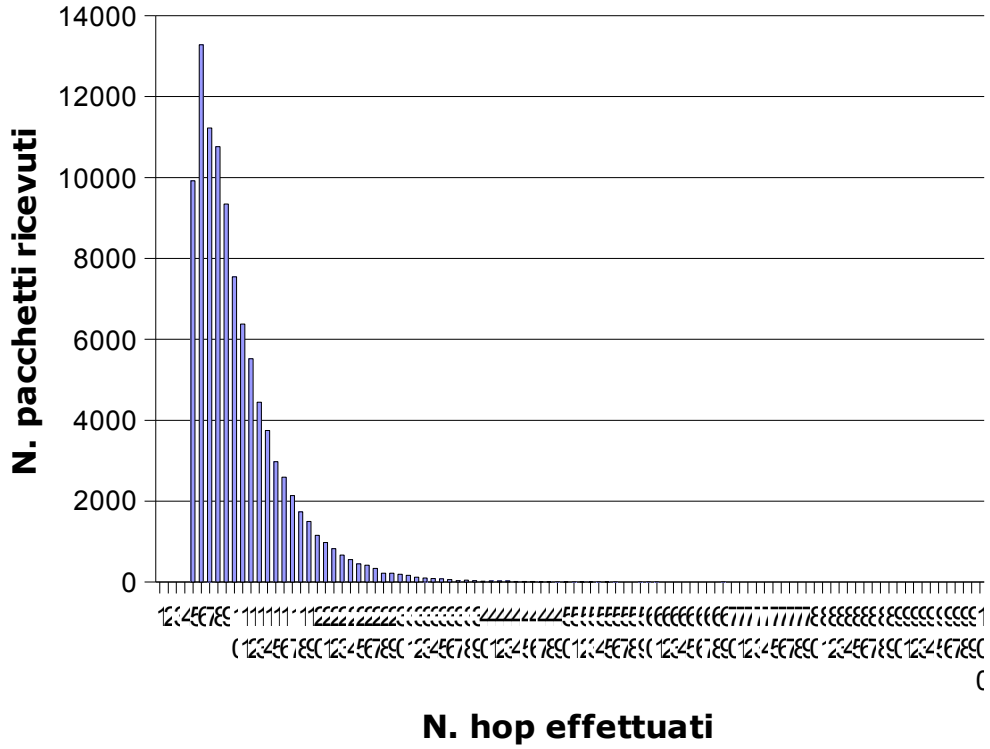

Immagine 6: Istogramma della distribuzione dei pacchetti, in base al numero di hop

Questo grafico, è stato realizzato tramite i dati presenti nel file `distribuzione_pacchetti.csv`, riformattato poi, nel file `distribuzione_pacchetti.xls`.

La simulazione, comporta il lancio di 100.000 pacchetti ognuno, avente la capacità di effettuare 1.000 hop. Per questioni di visibilità, ho scelto di rappresentare i pacchetti che hanno compiuto il loro cammino entro 100 passi: scorrendo il file `distribuzione_pacchetti.csv`, è facile notare che oltre questo valore, non sono presenti pacchetti recapitati al nodo di destinazione.

Osservando il grafico (in particolare, concentrando l'attenzione sul picco), la maggioranza dei pacchetti, ha compiuto il proprio cammino con un numero compreso tra i 5 ed i 20 hop (circa). Per le considerazioni relative al TTL dei pacchetti, si rimanda al seguente capitolo.

4.Scelta del TTL (time to live) dei pacchetti

Sempre grazie al file `distribuzione_pacchetti.csv`, possiamo calcolare la cumulativa, tramite l'ausilio di un foglio di calcolo elettronico: per questioni di spazio, riporto i valori corrispondenti ai soli cammini composti dal massimo 30 hop:

N.Hop	N.Pacchetti ricevuti	Cumulativa
1	0	0
2	0	0
3	0	0
4	0	0
5	9921	9921
6	13280	23201
7	11225	34426
8	10759	45185
9	9341	54526
10	7543	62069
11	6376	68445
12	5522	73967
13	4446	78413
14	3745	82158
15	2974	85132
16	2593	87725
17	2136	89861
18	1737	91598
19	1498	93096
20	1153	94249
21	**978**	**95227**
22	823	96050
23	665	96715
24	552	97267
25	449	97716
26	416	98132
27	340	98472
28	220	98692
29	215	98907
30	190	99097
...	...	...

Tabella 3: Distribuzione dei pacchetti e degli hop: evidenziato in giallo, compare il valore del TTL ottimale corrispondente al 95% dei pacchetti recapitati

I requisiti progettuali, richiedevano che si calcolasse il valore del TTL, in corrispondenza del 95% della massa della distribuzione. Tale valore, nel mio caso, corrisponde ad un cammino di 21 hop, valore ottimo per garantire la sopravvivenza dei pacchetti, all'interno della rete realizzata.

5.Il programma di simulazione

Veniamo ora alla parte finale di questa relazione ovvero la discussione del codice, relativo al programma che ho realizzato.

Il corpo principale, è rappresentato dal file `Progetto_CPSM.m`, realizzato attraverso il linguaggio Matlab e, di cui ne riporto il codice:

```
1  hop = 1000;
2  pacchetti = 100000;
3
4  for a = 1:hop
5      for b = 1:2
6          distribuzione(a, 1) = a;
7          distribuzione(b, 2) = 0;
8      end
9  end
10
11 pacchetti_ricevuti = 0;
12 pacchetti_persi = 0;
13
14 file1_destinazione = fopen('risultati_simulazione.txt', 'wt');
15 file2_destinazione = fopen('distribuzione_pacchetti.csv', 'wt');
16
17
18 fprintf(file1_destinazione, 'Inizio della simulazione del: %d-%d-%d; ore:
   %d:%d:%d\n\nN.Pacchetto\t\tN.Hop\t\tRisultato\n\n', fix(clock));
19 fprintf(file2_destinazione, 'Simulazione del: %d-%d-%d; ore: %d:%d:
   %d\nDistribuzione numero di hop e pacchetti ricevuti\n\nN.Hop,N.Pacchetti
   ricevuti\n', fix(clock));
20
21 acquisizione_matrice;
22
23 for i = 1:pacchetti
24
25     nodo_corrente = 1;
26     nodo_successivo = 1;
27     hop_effettuati = 0;
28
29     fprintf(file1_destinazione, '%d\t\t\t', i);
30
31     instradamento;
32
33     fprintf(file1_destinazione, '%d\t\t', hop_effettuati);
34     if nodo_corrente ~= y
35         fprintf(file1_destinazione, 'Perso\n');
36         pacchetti_persi = pacchetti_persi + 1;
37     else
```

```
38          fprintf(file1_destinazione, 'Ricevuto\n');
39          pacchetti_ricevuti = pacchetti_ricevuti + 1;
40          distribuzione(hop_effettuati, 2) = (distribuzione(hop_effettuati,
   2) + 1);
41      end
42  end
43
44  fprintf(file1_destinazione, '\n\n');
45  fprintf(file1_destinazione, 'Pacchetti arrivati a destinazione: %d\n',
   pacchetti_ricevuti);
46  fprintf(file1_destinazione, 'Pacchetti persi: %d\n', pacchetti_persi);
47  for l = 1:hop
48      fprintf(file2_destinazione,'%d,', distribuzione(l, 1));
49      fprintf(file2_destinazione,'%d\n', distribuzione(l, 2));
50  end
51
52  fclose('all');
```

Tralasciando le linee relative alla creazione e formattazione dei files di output, osserviamo che le linee, dalla 1 alla 12 e dalla 25 alla 27, sono comandi di inizializzazione delle variabili necessarie al corretto funzionamento della simulazione: in particolare la variabile `distribuzione`, è una matrice 1000 x 2, contenete i dati utili al tracciamento dell'istogramma ed alla scelta del valore di TTL, visti nel capitolo 4 e 5.

Dalla linea 23, parte la simulazione vera e propria: con un ciclo `for`, viene simulato il lancio dei singoli pacchetti: se (`if`, linea 34) il peso del nodo generato casualmente, tramite la funzione `instradamento`, è diverso dal nodo di destinazione (nodo 14), il pacchetto verrà conteggiato nei `pacchetti_ricevuti` altrimenti, verrà inserito tra i `pacchetti_persi` (linea 36 e 39).

Verso la fine del ciclo (linea 40), si memorizza nella matrice `distribuzione`, il numero degli hop effettuati dal pacchetto *i-esimo*.

Prima di procedere all'analisi del processo di *instradamento*, è opportuno capire come i dati della rete (peso del nodo e nodi *primi vicini*), vengono inseriti nella simulazione. La linea 21, riporta la funzione `acquisizione_matrice`, implementata nel file `acquisizione_matrice.m`:

```
1   matrice_grafo = xlsread('tabella_nodi.xls');
2   [y, x] = size(matrice_grafo);
3
4   for m = 1:y
5       for n = 1:x
6           if (isnan(matrice_grafo(m, n)))
7               matrice_grafo(m, n) = matrice_grafo(m, 1);
8           end
9       end
10  end
```

In sostanza, i dati presi da un foglio di calcolo (`tabella_nodi.xls`), il quale deve essere

sempre presente nella medesima cartella dei files del programma, vengono letti da Matlab ed inseriti in una matrice denominata `matrice_grafo`.

In tale matrice, vi è la possibilità della presenza di celle vuote o contenenti testo che, il linguaggio Matlab, interpreta come valori `NaN` (Not a Number). Dopo l'acquisizione e tramite il ciclo `for` (linea 4), risolvo tale problema, sostituendo ai valori `NaN` (non utilizzabili nella simulazione) il valore del peso del nodo, la cui riga corrente, fa riferimento.

Supponiamo che la seguente tabella 4, sia una porzione della `matrice_grafo` del programma:

Peso del nodo	***Arco 1***	***Arco 2***	***Arco 3***
...	...	...	...
8	11	12	9
9	12	7	***NaN***
10	9	12	13
11	14	8	***Nan***
12	11	14	10
...	...	...	...

Tabella 4: Esempio di applicazione della funzione "acquisizione_matrice": prima

La funzione `acquisizione_matrice`, trasforma la matrice nel modo seguente:

Peso del nodo	***Arco 1***	***Arco 2***	***Arco 3***
...	...	...	...
8	11	12	9
9	12	7	***9***
10	9	12	13
11	14	8	***11***
12	11	14	10
...	...	...	...

Tabella 5: Esempio di applicazione della funzione "acquisizione_matrice": dopo

Avvenuta l'acquisizione dei dati e l'inizializzazione delle variabili, la linea 31 del file `Progetto_CPSM.m` tramite la funzione `instradamento`, realizzata nel file `instradamento.m`, effettua il cammino dell'*i-esimo* pacchetto:

```
1   for k = 1:hop
2       if (nodo_corrente ~= y)
3
4           riga_matrice = matrice_grafo(nodo_corrente, :);
5
6           nodo_1 = riga_matrice(:, 2);
7           nodo_2 = riga_matrice(:, 3);
```

```
8           nodo_3 = riga_matrice(:, 4);
9
10          while (nodo_successivo == nodo_corrente)
11              numero_casuale = randint(1, 1, [1 5]);
12              if numero_casuale == 1
13                  nodo_successivo = nodo_1;
14              elseif numero_casuale == 2
15                  nodo_successivo = nodo_2;
16              elseif numero_casuale == 3
17                  nodo_successivo = nodo_3;
18              end
19          end
20
21          nodo_corrente = nodo_successivo;
22
23          hop_effettuati = hop_effettuati + 1;
24      else
25          break
26      end
27  end
```

La riga della matrice (`riga_matrice`, linea 4), ossia quella corrispondente al nodo corrente (per il primo hop sarà il nodo 1, per l'*n-esimo* hop sarà uno qualsiasi dei nodi della rete, memorizzato nella variabile `nodo_corrente`), viene scomposta nei nodi *primi vicini* (variabili `nodo_1`, `nodo_2`, `nodo_3`) e, tramite il ciclo `while` (linee da 10 a 19), viene randomizzato un numero intero (`randint`) compreso tra 1 e tre 3 ed a cui è associato un nodo *primo vicino*.

Successivamente (linea 21), se il nodo non è pari ne al nodo di destinazione ne al `nodo_corrente`, viene memorizzato come `nodo_successivo` e il numero di hop aumentato di 1 (`hop_effettuati`, variabile che da specifiche, doveva essere `nhop`).

Terminato il ciclo, si riparte col passo (hop) successivo, fino al *1.000-esimo*; concluso il cammino, si ripete l'intero procedimento col pacchetto successivo, fino al *100.000-esimo*.