

Peersim Epidemic Protocol

Fabio Giuseppe Strozzi

`fstrozzi@cs.unibo.it`

1 Obiettivi

Lo scopo di questo progetto è scrivere un protocollo epidemico in grado di generare una rete con topologia random partendo da un overlay arbitrario.

In generale, un protocollo epidemico permette di diffondere lo stato di un nodo della rete ai suoi nodi vicini. In questo caso specifico, lo stato di un nodo è rappresentato dalla vista locale dei suoi vicini, ovvero un sottoinsieme di dimensione limitata dei nodi della rete coi quali esso può comunicare. Il protocollo quindi si basa sullo scambio di peer tra peer collegati da un arco. La progettazione del protocollo ha mirato al conseguimento delle seguenti proprietà della topologia random della rete che si voleva ottenere:

1. grado medio dei nodi tendente a quello stabilito dall'utente (`VIEWSIZE`);
2. coefficiente di clustering possibilmente basso;
3. basso valore della lunghezza media del cammino tra i nodi (average path length).

Il grado dei nodi è un aspetto importante in una rete perché è direttamente collegato alla robustezza ai guasti: maggiore è il suo valore, maggiore sarà la capacità della rete di tollerare la perdita di archi. In secondo luogo è cruciale (insieme alla dimensione dei messaggi scambiati `MSGSIZE`) per garantire rapidità nel diffondersi delle informazioni (in questo caso le viste di nodi).

Il coefficiente di clustering è un valore che tenta di misurare il numero di “agglomerati” di nodi presenti nella rete, ovvero di sottoinsiemi di nodi maggiormente connessi tra di loro. È stato ritenuto importante minimizzare questo valore per due motivi: un alto coefficiente di clustering in una rete comporta l'aumento di messaggi ridondanti (dovuti alle eccessive connessioni tra nodi vicini di vicini); inoltre, rende più soggetta la rete al partizionamento se si verificano guasti negli archi che connettono i cluster al resto della rete.

Infine, avere un basso valore dell'average path length influisce positivamente sui costi per accedere ai vari peer e sulla scalabilità della rete.

2 Il protocollo

Periodicamente, ogni nodo della rete scambia con uno dei suoi vicini un sottoinsieme dei peer delle rispettive viste (possibilmente di dimensione `MSGSIZE`). La scelta del vicino e dei nodi della vista scambiata dipendono da una euristica che cerca di valutare la “notorietà” e la “ridondanza” dei nodi. Minimizzare il coefficiente di clustering significa ridurre la possibilità che i vicini di un peer siano vicini tra di loro. Ogni nodo può tentare di dare una stima di questo coefficiente valutando quanto i suoi vicini forniscono, in cicli successivi, nodi già conosciuti (cioè già appartenenti alla vista locale). In particolare, ogni peer, quando riceve un vista da un suo vicino, tiene traccia delle seguenti informazioni:

notorietà : un valore intero associato ad ogni nodo della propria vista che viene incrementato ogni volta che questo nodo è presente in una vista ricevuta da un altro peer; cerca di stimare quanto un vicino è conosciuto tra i propri vicini.

ridondanza : un valore decimale positivo associato ad ogni nodo della propria vista che viene aggiornato ogni volta che da quel nodo sono ricevuti peer che sono già dei vicini; cerca di stimare la “qualità” di un vicino, ovvero quanto esso è in grado di fornire peer non conosciuti.

Avendo a disposizione questo tipo di metriche sarebbe possibile definire diverse strategie per minimizzare il coefficiente di clustering. Nella strategia scelta, ogni nodo avvia il protocollo di scambio col vicino dal più alto valore di ridondanza e gli spedisce i nodi col più basso valore di notorietà. Il peer ricevente, dopo aver aggiornato le metriche, aggiunge alla propria vista i nodi nuovi ricevuti (se ce ne sono). Nel caso in cui la dimensione massima della vista sia stata raggiunta, vengono rimpiazzati i nodi più noti (tanti quanti bastano per aggiungere i nodi nuovi). Infine, il peer risponderà al mittente inviando una sotto-vista random, che risulta molto utile per valutare il livello di “ridondanza” raggiunto dal peer ricevente.

L’idea che è alla base di questo protocollo è che un valore basso del livello di clustering può essere raggiunto se ogni nodo provvede a ridistribuire intelligentemente le informazioni tra i propri vicini. In questo modo e grazie ad una dimensione della vista appropriata in relazione al numero di nodi della rete, ci si aspetta che anche l’average path length tenda a diminuire nel tempo.

Per quanto riguarda il grado medio, il protocollo fa sì che esso aumenti progressivamente ad ogni scambio fino a raggiungere il limite stabilito dall’utente (`VIEWSIZE`); i vicini, infatti, non vengono mai rimossi dalla vista a

meno che non sia necessario rimpiazzarli con altri nodi, ma anche in questo caso il numero di connessioni non cala.

3 Implementazione

La formulazione del problema ha messo in luce i punti focali del protocollo:

1. scelta del nodo con cui comunicare;
2. scelta dei nodi che compongono la sotto-vista da inviare;
3. come effettuare lo scambio delle viste ricevute.

Variando anche solo una di queste tre politiche si può modificare drasticamente il comportamento del protocollo. Esiste, però, un algoritmo abbastanza generico che struttura in maniera semplice il problema:

```
// scegli un nodo con cui comunicare
Node node = pickNode();
// scegli la vista da spedire
Node[] mySubView = getSubView();
// spedisci la vista e ricevine un'altra
Node[] subView = node.exchange(mySubView);
// combina la vista ricevuta con la propria
mergeView(subView);
```

Questo ha suggerito l'idea di spezzare l'implementazione in due classi secondo il design pattern *Template*. Nella prima classe, astratta, (*Randomizer*) viene implementato solo lo scheletro dell'algoritmo generico mentre le funzionalità più importanti vengono solo definite tramite metodi astratti. La loro implementazione è lasciata ad un'altra classe (in questo caso *FreshRandomizer*). Grazie a questa separazione logica diventa più semplice modificare il protocollo; una volta costruito lo scheletro, infatti, le modifiche vengono apportate esclusivamente alle classi che implementano le politiche. Il diagramma UML in figura 1 schematizza questa separazione.

Per quanto riguarda il livello di “ridondanza” dei nodi, esso viene aggiornato nel seguente modo: una volta ricevuta una vista da un nodo, per prima cosa bisogna calcolare quanti nodi, tra quelli ricevuti, sono già conosciuti (questo numero è memorizzato nella variabile *duplicates*); a questo punto la ridondanza del nodo mittente al ciclo $t + 1$ esimo è definita dall'espressione

$$redundance_{t+1} = \alpha \times redundance_t + (\alpha - 1) \times duplicates$$

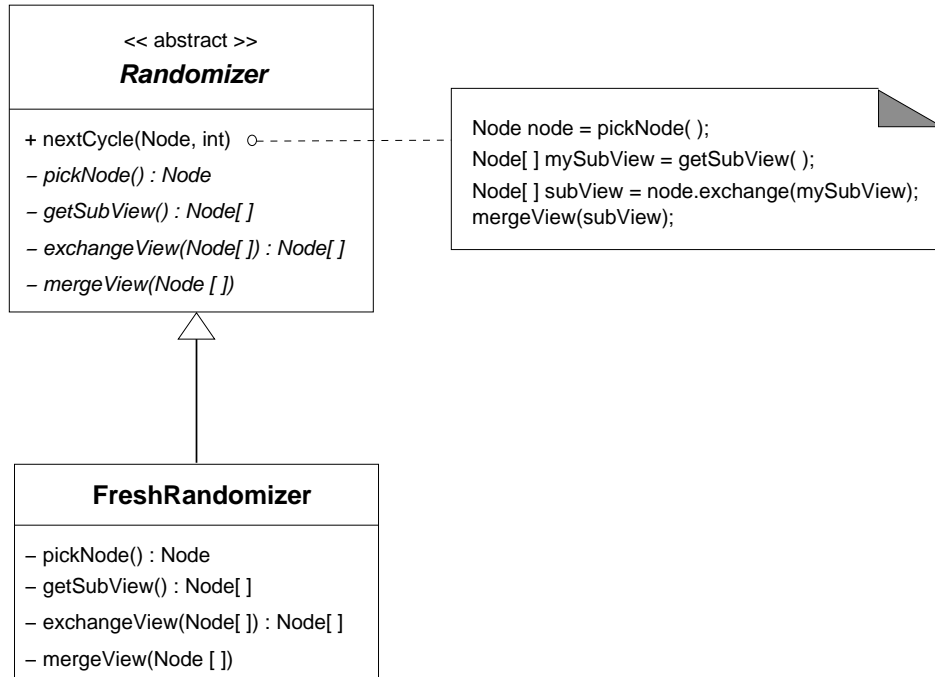


Figura 1: Applicazione del design pattern *Template*.

dove α è un valore compreso tra 0 e 1 (escluso) ed è configurabile dall'utente (tramite il parametro `alpha` nel file di configurazione). In questo modo la ridondanza di un vicino può crescere in maniera più o meno conservativa a seconda che α tenda rispettivamente a 1 o a 0 (di default α è uguale a 0.4).

4 Risultati

Le simulazioni sono state eseguite utilizzando come configurazioni iniziali della rete una topologia a stella (`WireStar`) ed una topologia scale free (`WireScaleFreeBA`). Le dimensioni scelte sono state di 500 e 2000 nodi; il parametro `VIEWSIZE` ha assunto i valori di 20 e 60 peer, mentre `MSGSIZE` di 15 e 40 peer. Le simulazioni hanno avuto una durata di 100 cicli ciascuna.

Come si può osservare dai grafici in figura 2 e 3, la curva del grado medio dei nodi ha una ripida ascesa e in meno di cinque cicli si stabilizza sul valore costante definito dall'utente (`VIEWSIZE`). Questo comportamento è indipendente dalla dimensione della rete mentre dipende dal numero di peer scambiati ad ogni ciclo in relazione alla dimensione della vista. Anche nei grafici successivi si intuisce che il fattore discriminante nell'andamento dei

vari parametri è soprattutto la dimensione della vista, mentre il numero di nodi scambiati influenza esclusivamente la rapidità di convergenza delle curve ad un punto fisso.

Riguardo il coefficiente di clustering, osservando attentamente, si può affermare che il comportamento è pressoché identico con entrambe le tipologie di rete. Infatti, in tutti e quattro i grafici (figure 4 e 5), le curve dopo un certo numero di cicli tendono a linearizzarsi: con 500 nodi ed una `VIEWSIZE` di 20 peer, la curva (rossa) tende ad un valore compreso tra 0.1 e 0.15 mentre se `VIEWSIZE` è pari a 60 peer la curva (verde) rimane compresa tra 0.2 e 0.25; discorso analogo vale se la rete ha dimensione di 2000 nodi e in tal caso le curve rossa e verde sono entrambe comprese approssimativamente tra 0.05 e 0.08. La distanza tra le curve, perciò, dipende esclusivamente dalle dimensioni della vista e della rete e non dalla tipologia (né da `MSGSIZE`). Inoltre, la curva verde (`VIEWSIZE` pari a 60) è sempre superiore alla curva rossa. Ciò è spiegabile dal fatto che il coefficiente di clustering cerca di quantificare la presenza di archi ridondanti, cioè di archi tra nodi che sono già vicini di un altro nodo; quindi, all'aumentare della dimensione della vista il numero di collegamenti "superflui" aumenta perché si ottiene un grafo maggiormente connesso. Guardando solo al coefficiente di clustering, il vantaggio maggiore si ottiene partendo da una tipologia a stella con un elevato numero di nodi. Nel caso della rete scale free con vista di dimensione pari a 60 nodi per peer vanno fatte due considerazioni: innanzitutto, nei cicli prefissati il coefficiente di clustering non migliora il valore iniziale; in secondo luogo, la curva non mostra immediatamente un andamento decrescente e anzi nei primi cicli il coefficiente di clustering tende a crescere molto rapidamente. Questo effetto rivela il reale comportamento del protocollo: inizialmente quello che avviene è una crescita indiscriminata delle viste di ogni singolo nodo che porta ad un aumento del coefficiente di clustering; solo in un momento successivo, quando tutte le viste sono sature, inizia lo scambio mirato dei nodi che permette di ridistribuire intelligentemente le conoscenze locali.

L'average path length dipende in maniera preponderante dal valore medio del grado dei nodi. Per questo motivo, nelle figure 6 e 7 i risultati migliori si ottengono esclusivamente col valore di `VIEWSIZE` più alto. Va notato come, anche in questo caso, i valori a cui tendono le curve non dipendono dalla configurazione iniziale della rete ma piuttosto dalle sue dimensioni e dalla dimensione della vista.

I grafici in figura 8 descrivono il numero di peer scambiati ad ogni ciclo. Non sono state fatte distinzioni in base alla topologia dato che i risultati si equivalgono; infatti, una volta raggiunto il grado stabilito tutti i nodi scambiano una quantità di informazioni costante e precisamente pari a $2 \times \text{MSGSIZE} \times n$, dove n è la dimensione in nodi della rete. Per sem-

plicità, la misurazione dei nodi scambiati è stata eseguita dal lato dei peer che instaurano la comunicazione: a regime, questi nodi spediscono ciclo per ciclo un numero di peer pari a `MSGSIZE` e ne ricevono altrettanti dal peer destinatario.

Gli ultimi grafici sono solo un esempio dell'evoluzione di due reti di 500 nodi configurate inizialmente con topologia star (figure 9) e scale free (figure 10) e valori di `VIEWSIZE` e `MSGSIZE` rispettivamente pari a 20 e 15. Si noti come nel caso della rete a stella la topologia muti immediatamente (bastano due soli cicli).

5 Conclusioni

Il protocollo implementato ha permesso di ottenere le caratteristiche di rete richieste. Considerando i grafici nelle figure 4 e 5, si può supporre che risultati migliori dal punto di vista del clustering possano essere ottenuti, a parità di dimensioni della vista, aumentando il numero di peer scambiati (perché si contribuirebbe a migliorare la conoscenza dei nodi a cui sono connessi i propri vicini). D'altra parte, con questo protocollo, esiste un limite inferiore al coefficiente di clustering che è stabilito dalla dimensione delle viste.

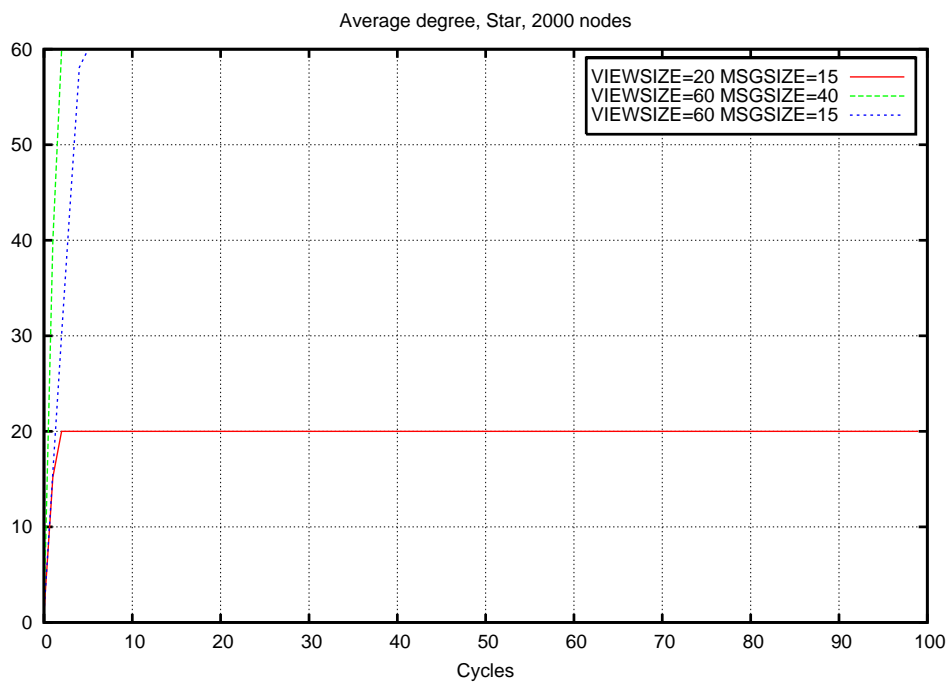
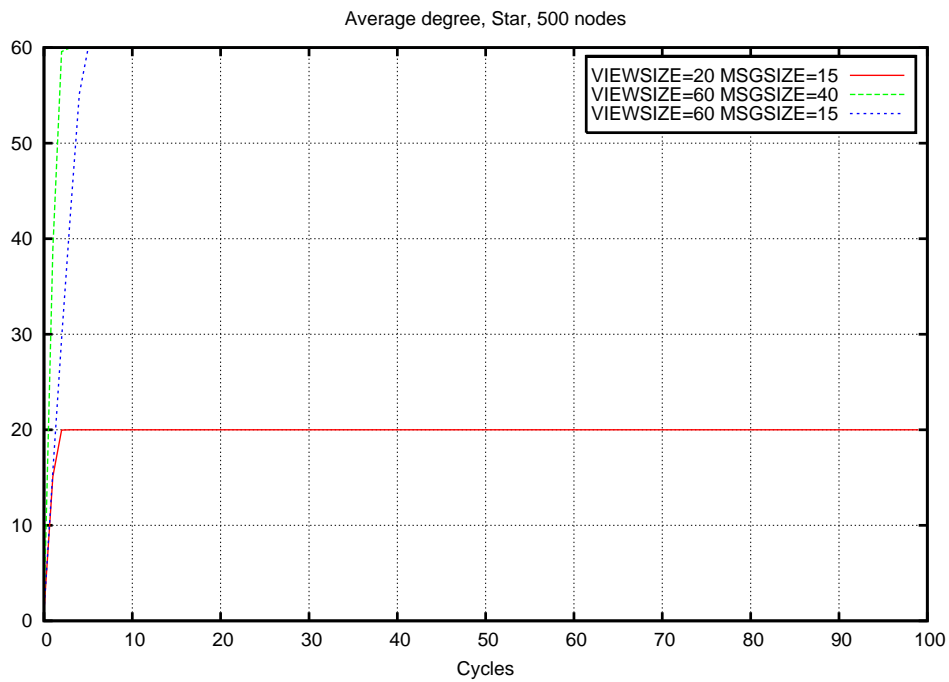


Figura 2: Andamento nel tempo del grado medio dei nodi per una tipologia di rete a stella con 500 e 2000 nodi.

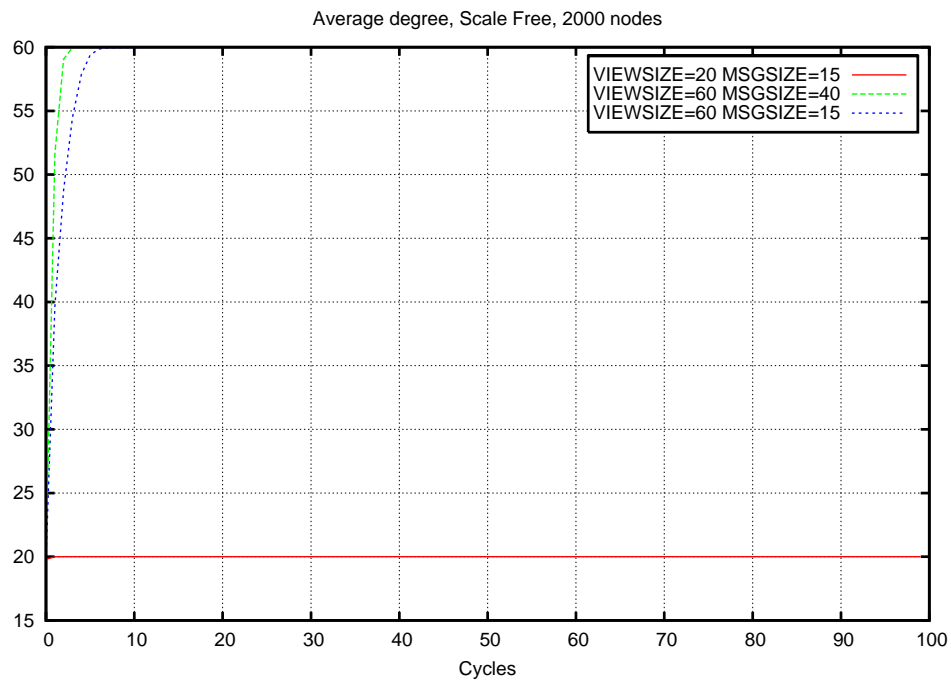
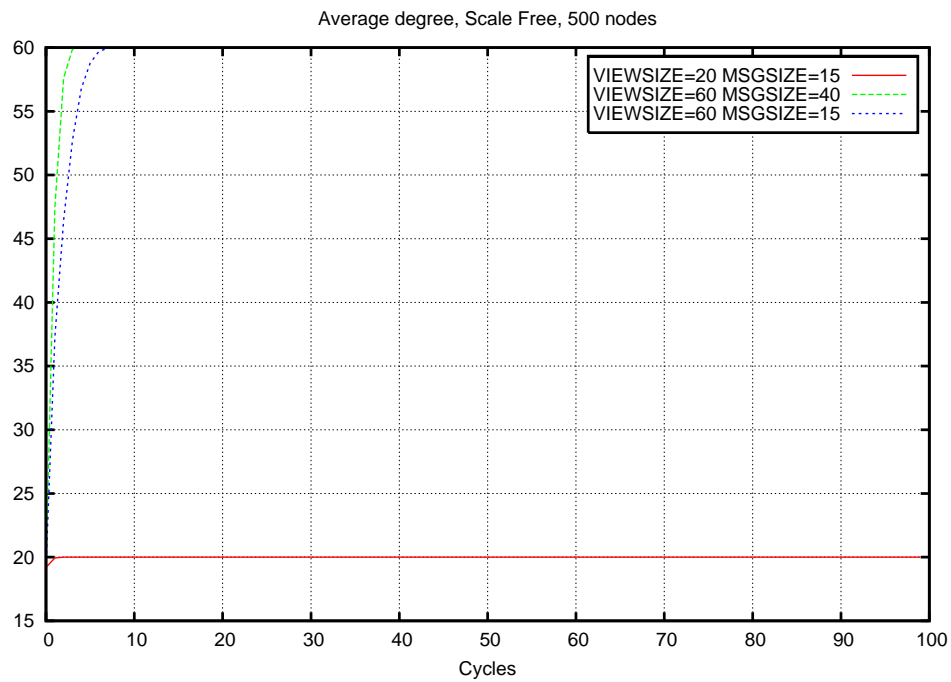


Figura 3: Andamento nel tempo del grado medio dei nodi per una tipologia di rete scale free con 500 e 2000 nodi.

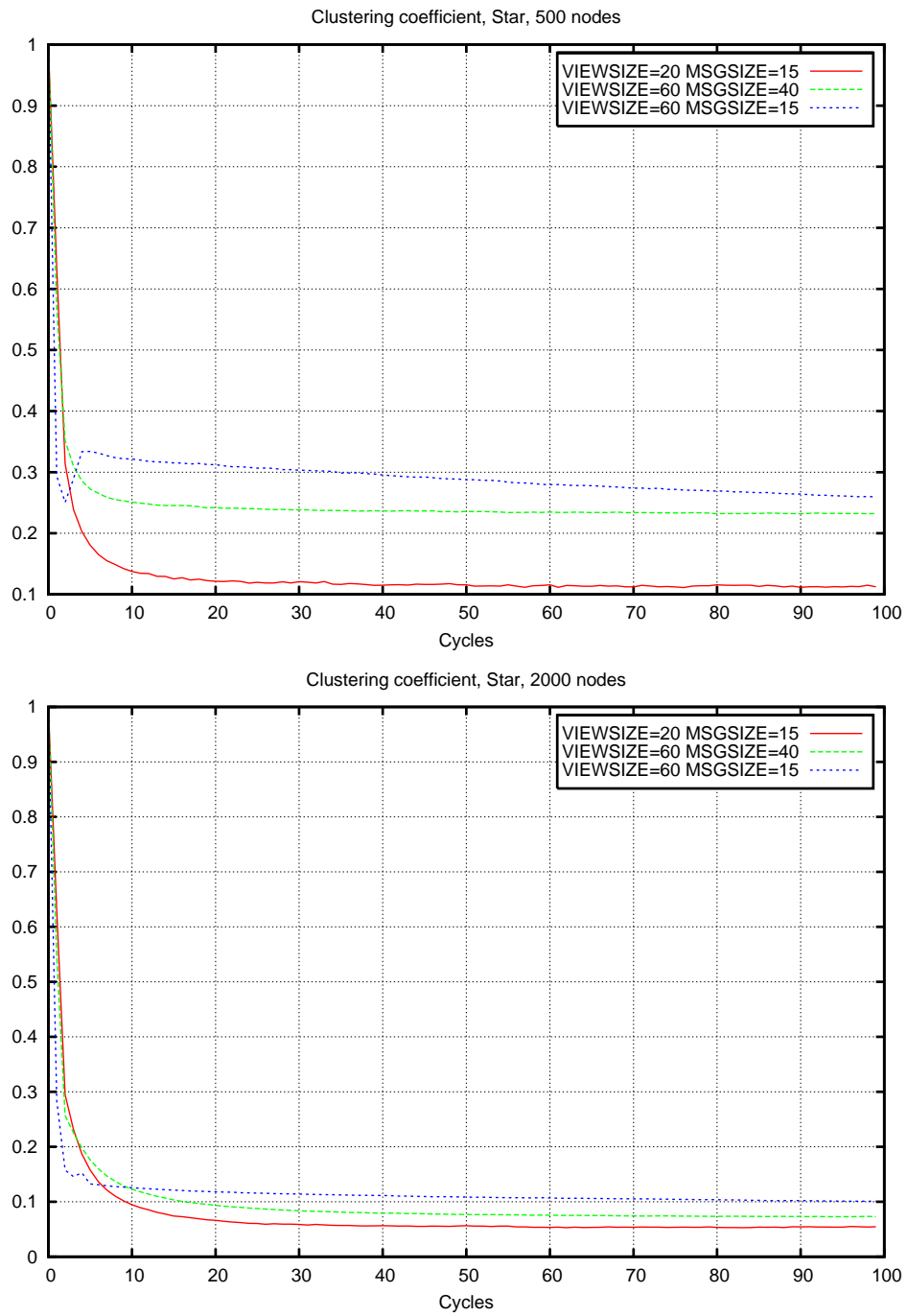


Figura 4: Andamento nel tempo del coefficiente di clustering per una tipologia di rete a stella con 500 e 2000 nodi.

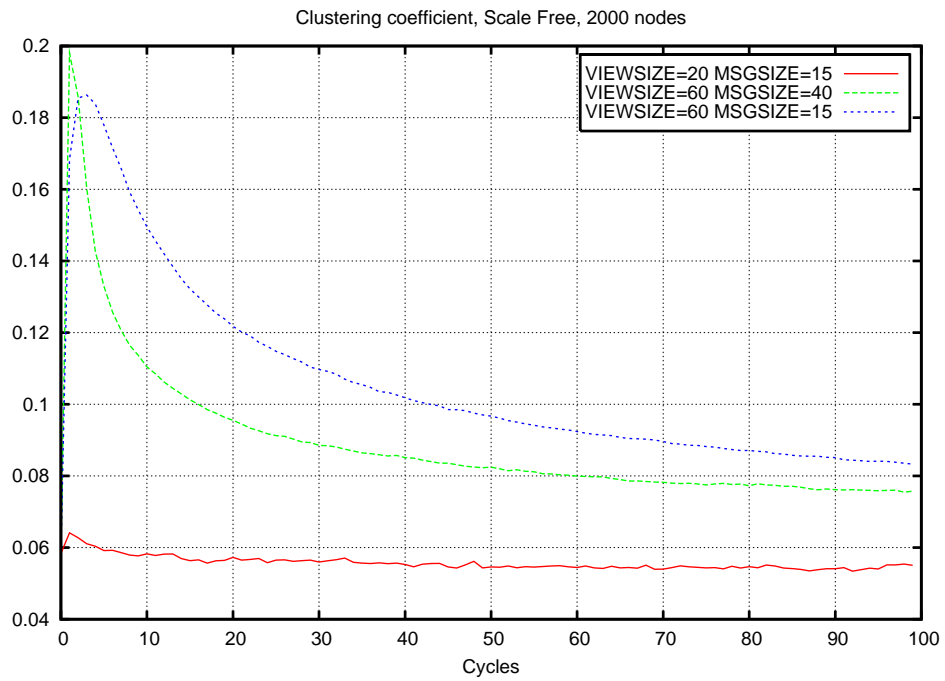
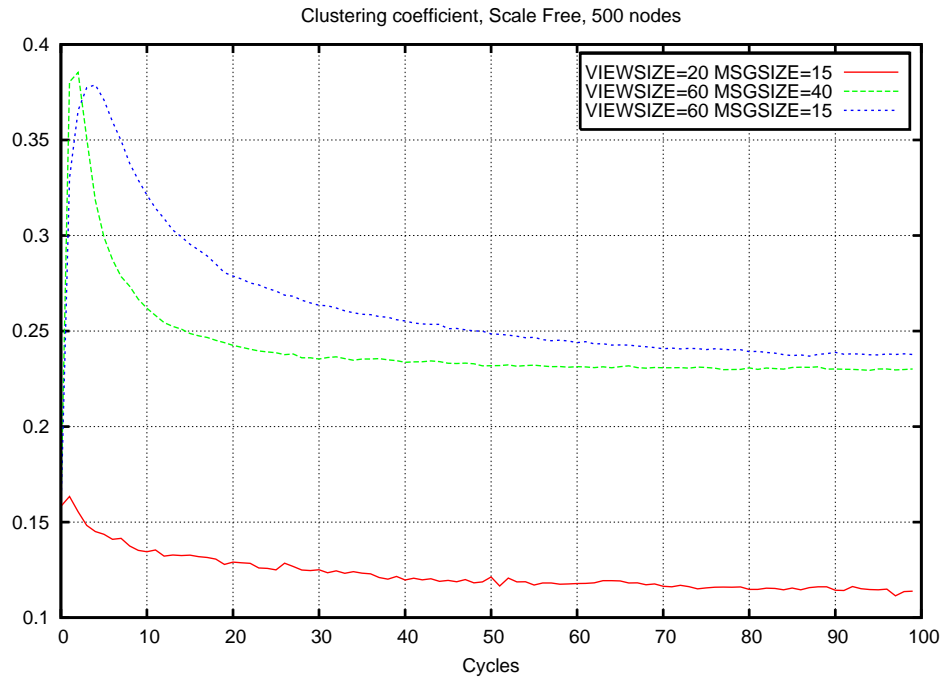


Figura 5: Andamento nel tempo del coefficiente di clustering per una tipologia di rete scale free con 500 e 2000 nodi.

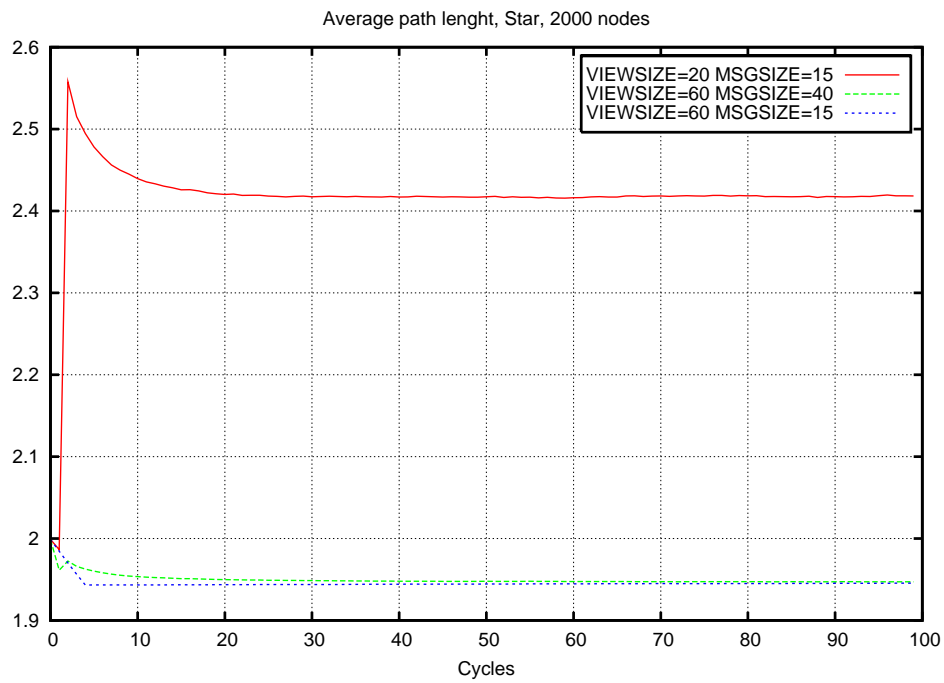
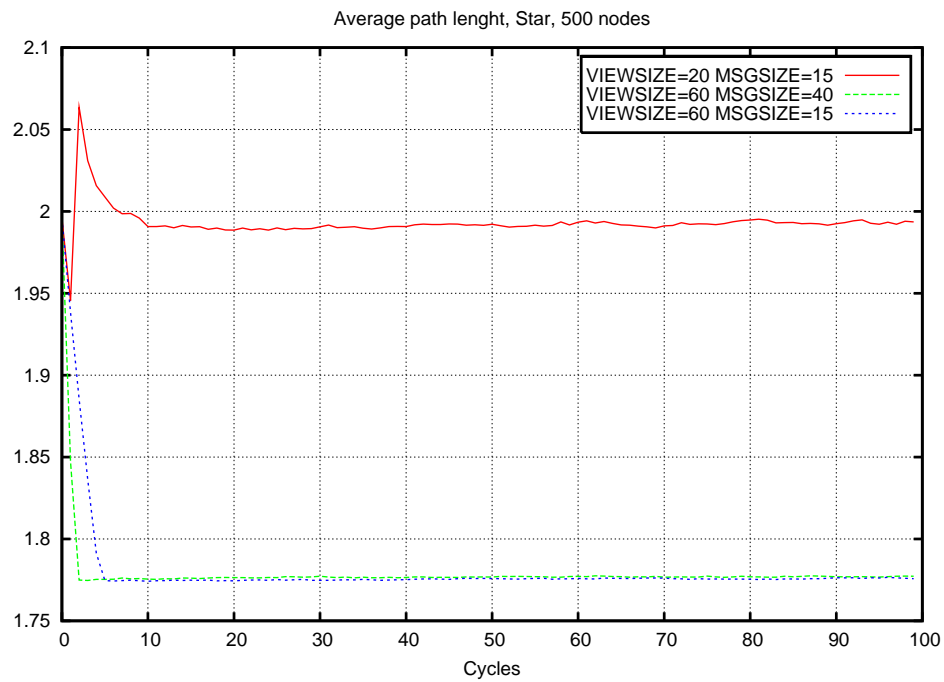


Figura 6: Andamento nel tempo dell'average path length per una tipologia di rete a stella con 500 e 2000 nodi.

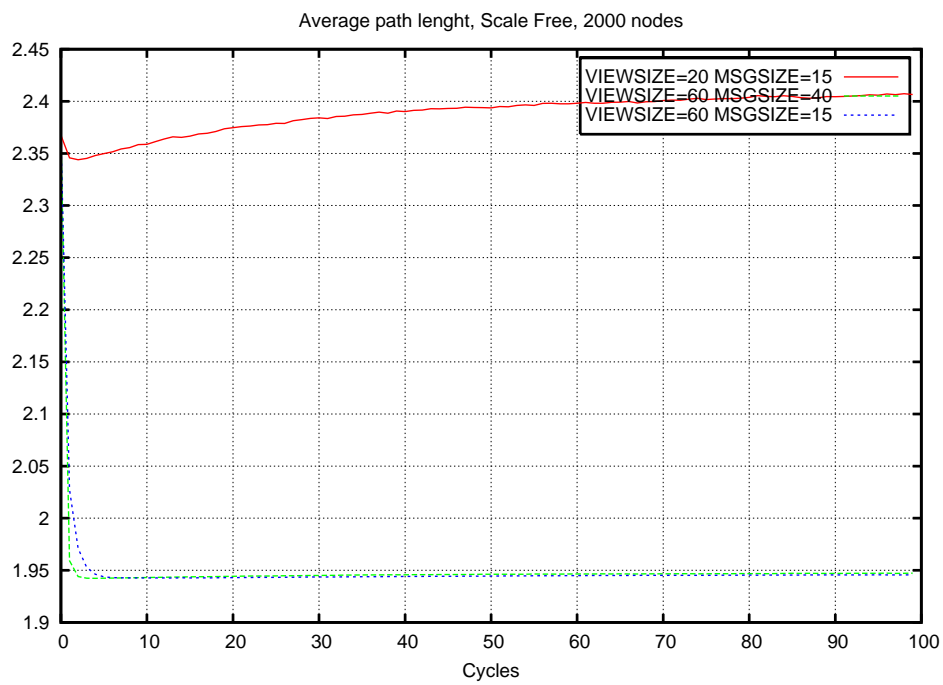
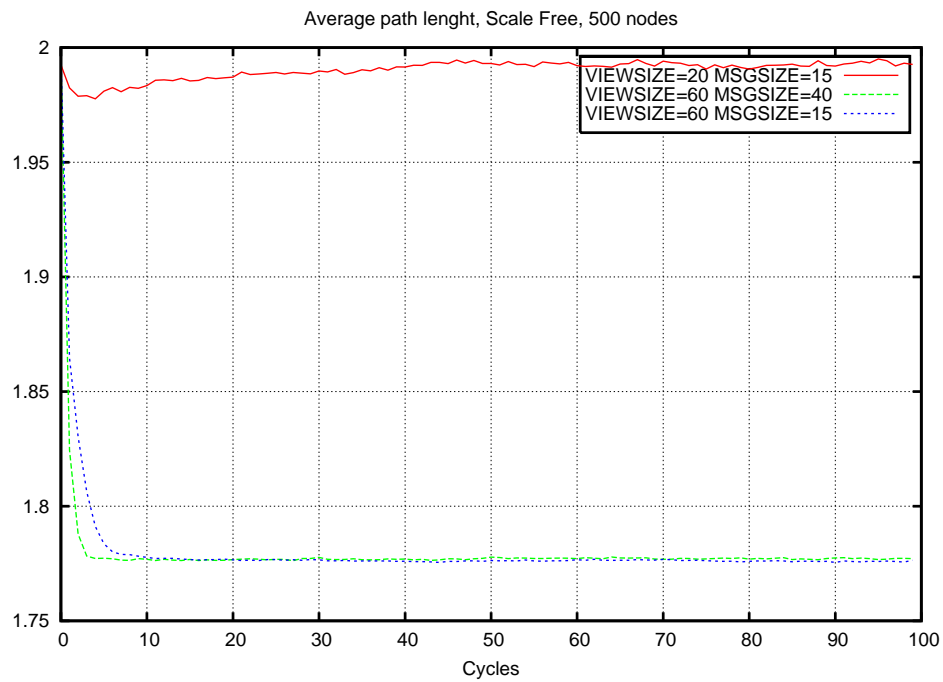


Figura 7: Andamento nel tempo dell'average path length per una tipologia di rete scale free con 500 e 2000 nodi.

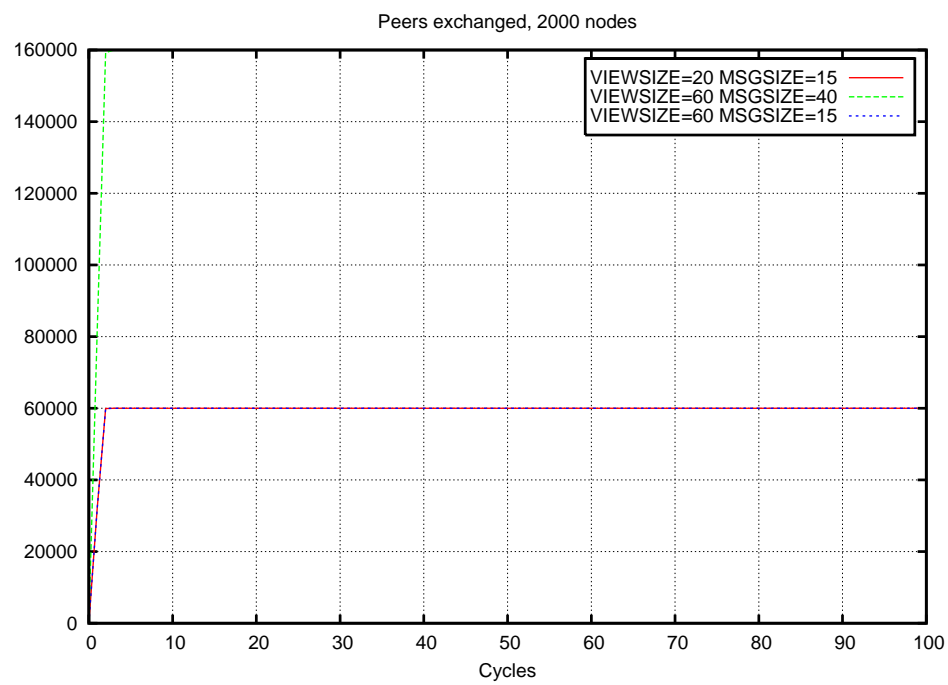
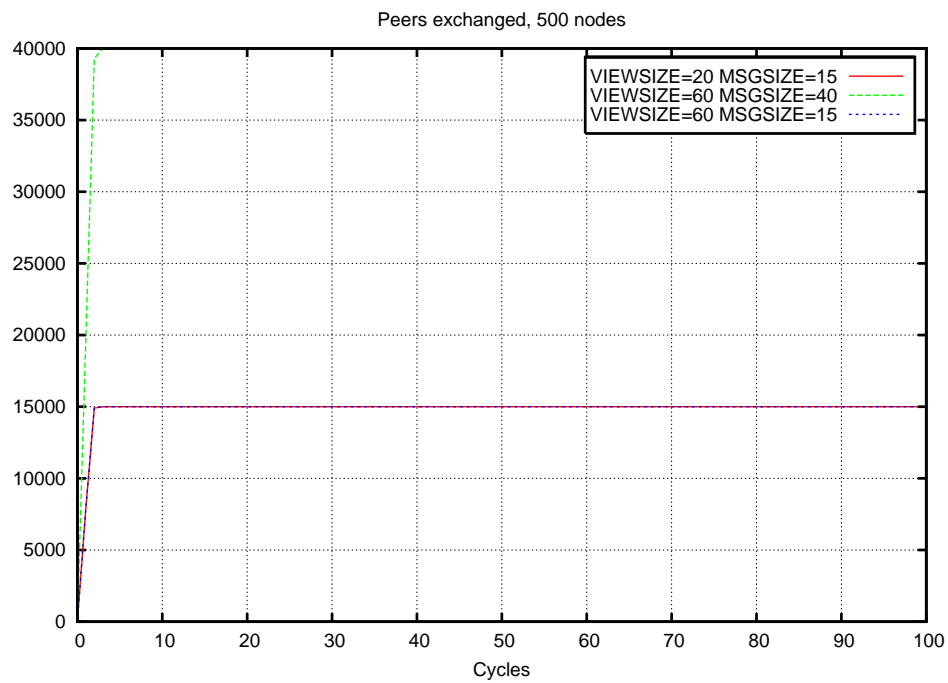
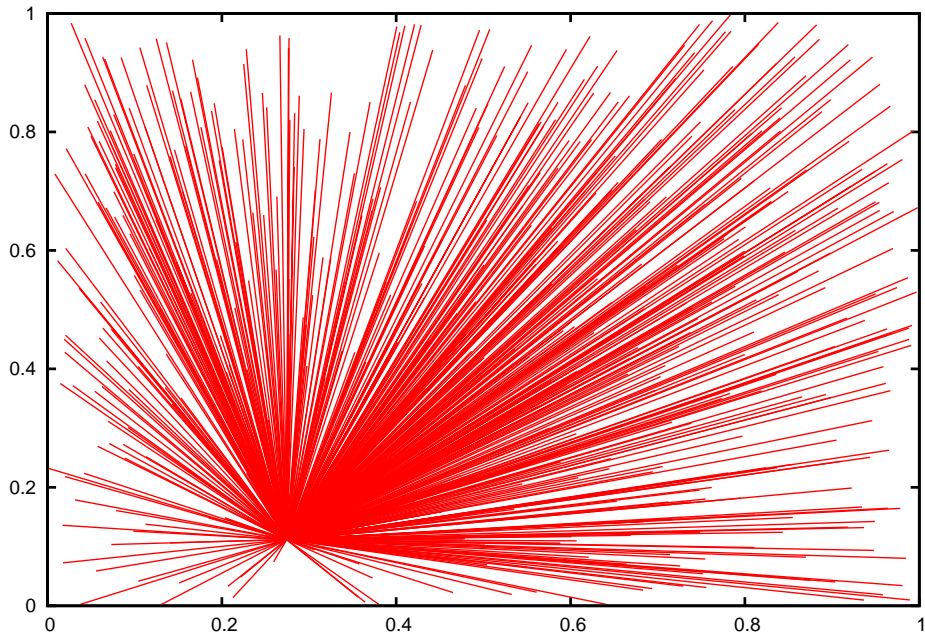
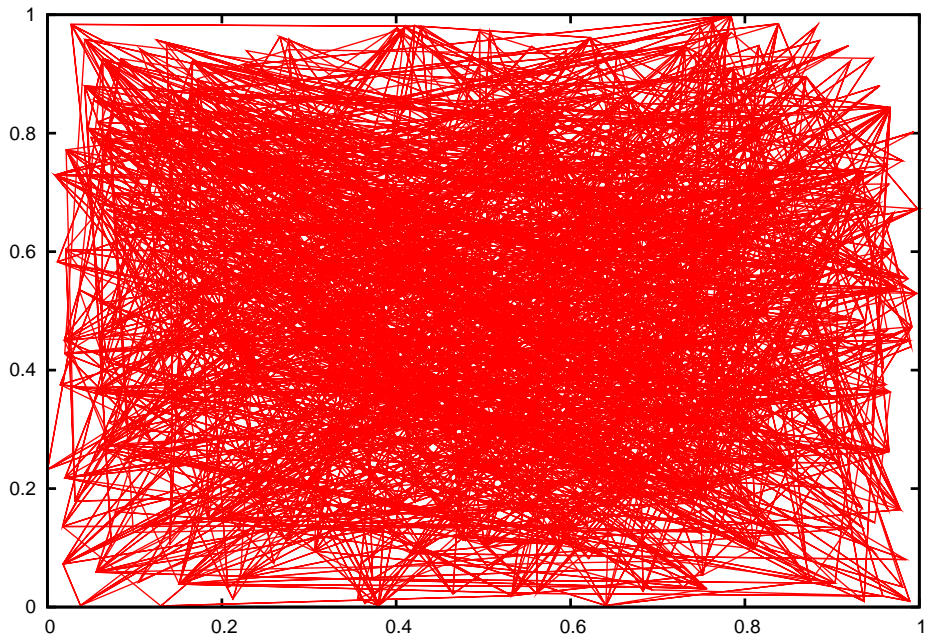


Figura 8: Traffico generato con 500 e 2000 nodi; questi risultati coincidono per entrambe le tipologie di rete.

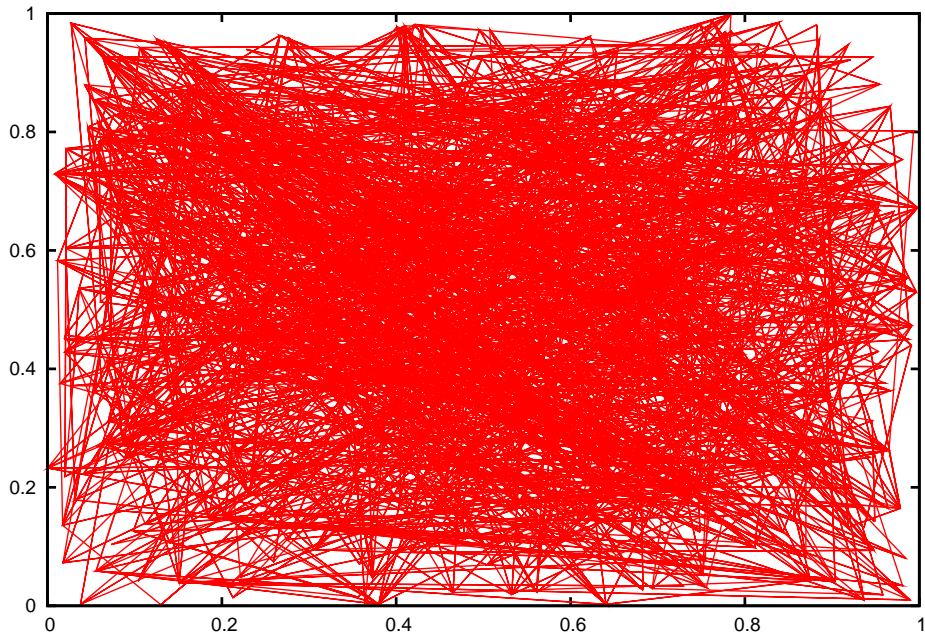
Star topology, 500 nodes, Cycle 0



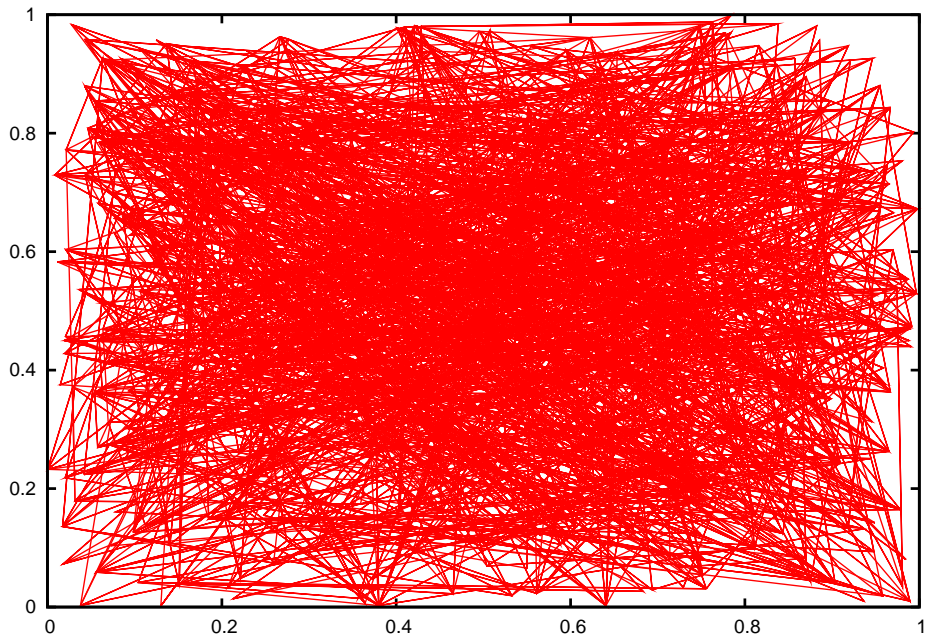
Star topology, 500 nodes, Cycle 2



Star topology, 500 nodes, Cycle 4



Star topology, 500 nodes, Cycle 10



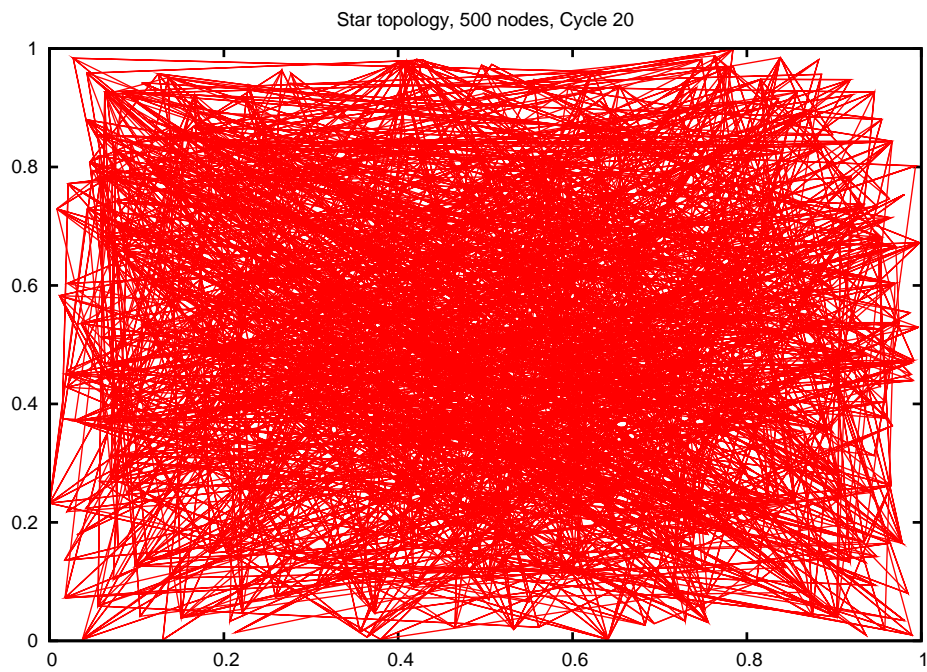
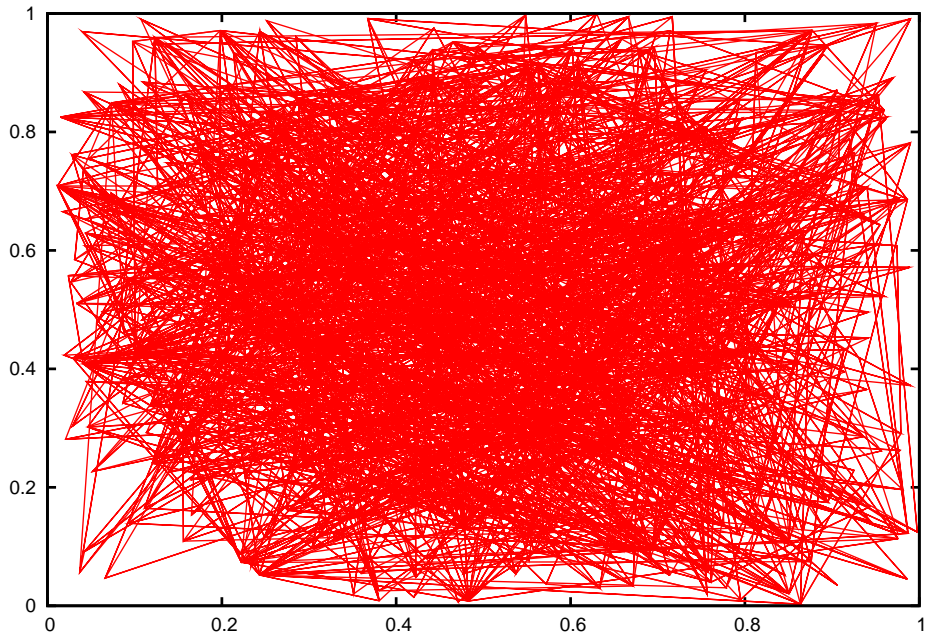
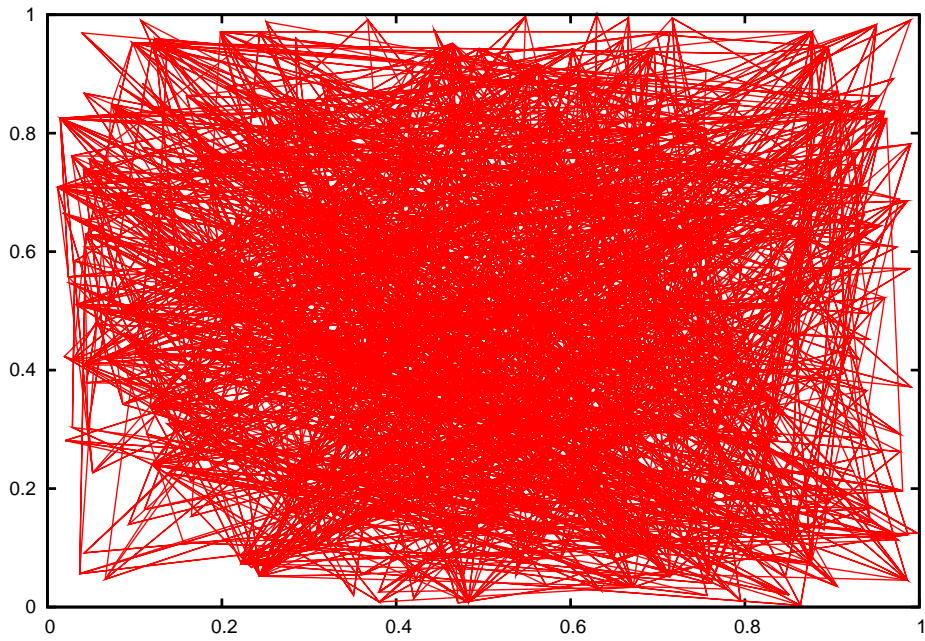


Figura 9: Esempio di evoluzione di una topologia a stella di 500 nodi, con viste di dimensioni pari a 20 peer ed un numero di peer scambiati per nodo per ciclo pari a 15.

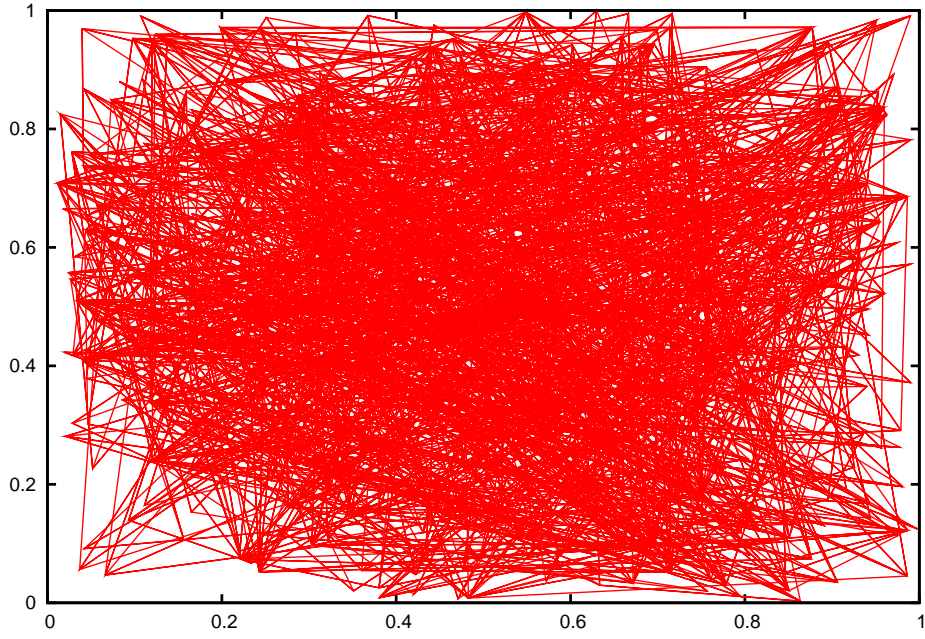
Scale Free topology, 500 nodes, Cycle 0



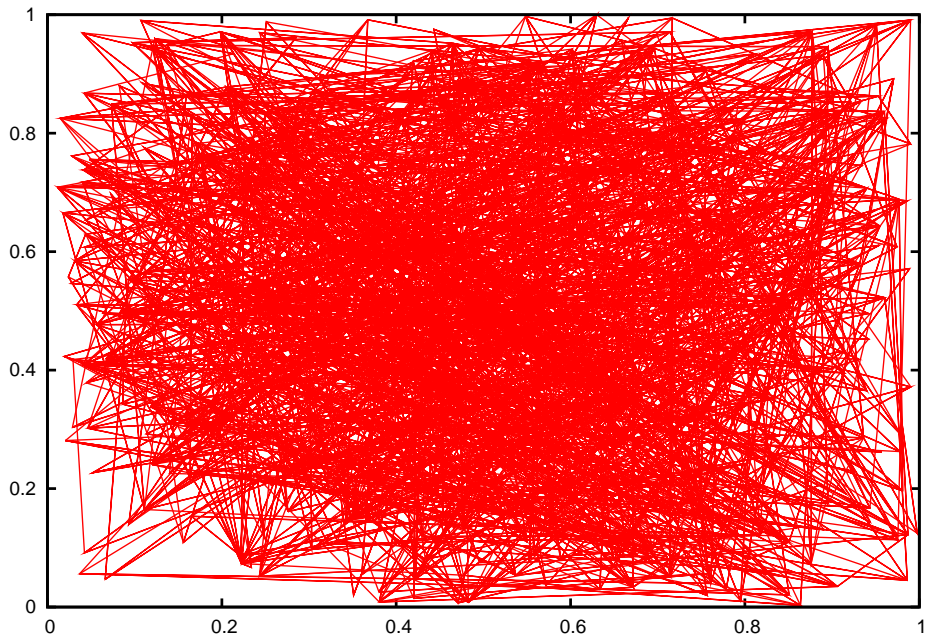
Scale Free topology, 500 nodes, Cycle 2



Scale Free topology, 500 nodes, Cycle 4



Scale Free topology, 500 nodes, Cycle 10



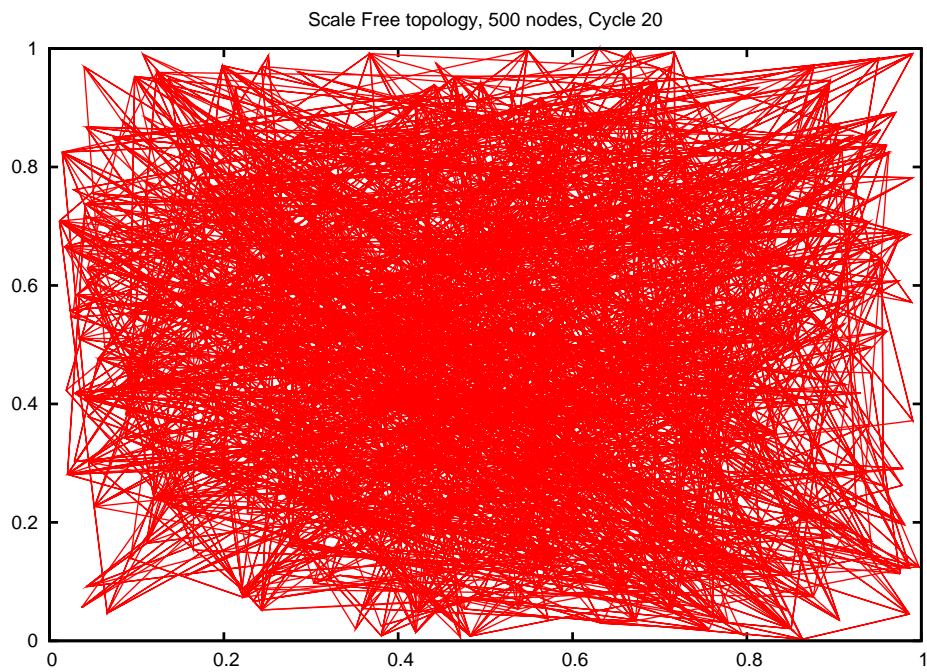


Figura 10: Esempio di evoluzione di una topologia a scale free di 500 nodi, con viste di dimensioni pari a 20 peer ed un numero di peer scambiati per nodo per ciclo pari a 15.