# Capitolo 5 – Architettura del sistema

## Simulatore

### File di Inizializzazione

### File Network Description

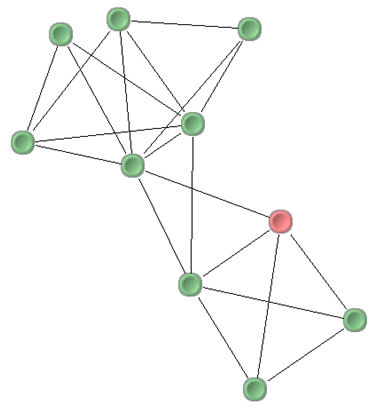
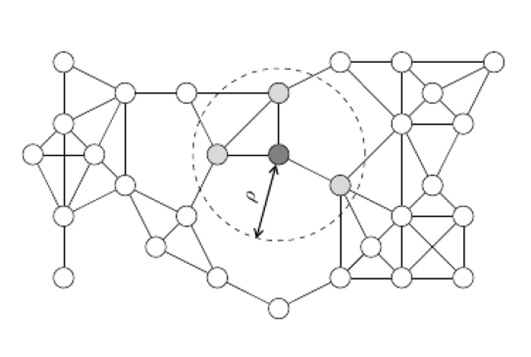
## Modello di rete

Il modello di rete utilizzato è quello del Random Geometric Graph, una rete di nodi rappresentata da un grafo connesso. I nodi sono connessi tra loro utilizzando il criterio di distanza geometrica, quindi ogni nodo sarà collegato con tutti quei nodi che sono ad una distanza uguale o inferiore di un dato raggio ρ da essi. Il raggio d’azione ρ sarà una variabile del nostro problema. Nella figura 5.1 sono rappresentati due esempi di RGG; il grafico di destra è una visualizzazione di un esempio di rete rappresentato dal tool grafico Tkenv di OMNeT++.

La generazione della nostra rete è dipendente da:

* Densità dei nodi.
* Numero di nodi che compongono la rete.
* Raggio d’azione del BLE (ρ).

La scelta di operare a diversi livelli di densità è dovuta al fatto di aver voluto studiare il comportamento della nostra soluzione in alcuni scenari urbani possibili per diverse distribuzioni geografiche sul territorio, in altre parole poter studiare il variare del comportamento del nostro algoritmo sia in situazioni di centri urbani normalmente molto densamente popolati, ad esempio medie/grandi città, sia in centri urbani con una più bassa concentrazione urbana che rappresentano la maggioranza dei paesi italiani.



*Figura 5.2: esempio di Random Geometric Graph.*

Per queste considerazioni abbiamo ipotizzato di associare a un nodo un abitante, perché mediamente ogni persona possiede uno smartphone. Abbiamo scelto come prima e più grande densità 0.02 abitanti per metro quadro perché ci è sembrato una densità già sufficientemente alta per avere ottimi valori in termini di perfomance, quindi simulare a densità superiori non ci è sembrato utile. Quello che invece abbiamo voluto studiare è come si degradano le prestazioni al diradarsi della concentrazione urbana, ed ecco il perché della scelta delle alte densità.

Le densità scelte sono:

1. D = 0.02 nodi/m2,
2. D = 0.01 nodi/m2,
3. D = 0.008 nodi/m2,
4. D = 0.005 nodi/m2,
5. D = 0.001 nodi/m2.

La generazione dell’area dipende anche dal numero di nodi in quanto, per mantenere un certo valore di densità è necessario che il rapporto tra il numero di utenti e l’area analizzata sia corretto. Quindi per ogni densità abbiamo scelto di fare molteplici prove con diversi valori di utenti. I valori di N che vanno da 2 a 100, sono valori pensati per simulare situazioni reali, mentre i valori più grandi sono stati scelti per vedere la scalabilità della nostra selezione in casi estremi.

I valori per il numero di nodi scelti sono:

1. N = 2,
2. N = 5,
3. N = 10,
4. N = 30,
5. N = 50,
6. N = 80,
7. N = 100,
8. N = 200,
9. N = 500,
10. N = 1000.

Come già anticipato, una volta fissata la densità e il numero di nodi si ricavano le misure dell’area in analisi. Abbiamo scelto di operare in questo modo in quanto aver scelto direttamente la grandezza dell’area avrebbe portato a scegliere aree o troppo piccole per il numero di nodi usato, con la conseguenza di avere prestazioni perfette sempre, oppure scegliere aree troppo grandi nelle quali la dispersione dei nodi sarebbe stata talmente alta da rendere inutilizzabile la soluzione, in quanto gli scenari così generati sarebbero stati scenari per i quali il nostro algoritmo non è stato pensato e risulti quindi inutilizzabile.

Infine l’ultimo parametro che abbiamo voluto far variare è stato il raggio d’azione ρ del BLE. Le specifiche tecniche dicono che il raggio d’azione è di circa 50 m, ma chi ha definito il protocollo BLE (chi è? Azienda Blutooth?) non ha né imposto né definito, alcuna distanza specifica che le case costruttrici di trasmettitori BLE devono seguire; cosa che fu fatta nel caso del suo predecessore Bluetooth Classic. Ciò fa sì che non si abbia un valore unico di ρ, ma ogni costruttore implementa il proprio trasmettitore in modo che sia più o meno potente. In media è stato rilevato che il raggio d’azione è circa 50 metri (non trovo + la *cit)*. Dato che i nodi della nostra rete sono smartphone e trasmettitori installati dalle varie case produttrici sono tutti diversi, abbiamo scelto di simulare per due valori di raggio ρ, uno pessimistico e uno ottimistico.

I valori di raggio ρ scelti sono:

* R = 15 m (pessimistico),
* R = 50 m (ottimistica, come la media rilevata).

Abbiamo scelto di non spingerci oltre col valore di raggio poiché ci è sembrato difficile che dispositivi mobili come i cellulari possano arrivare a 100 m o anche oltre come alcuni chip per impiego industriale.

### Costruzione della rete

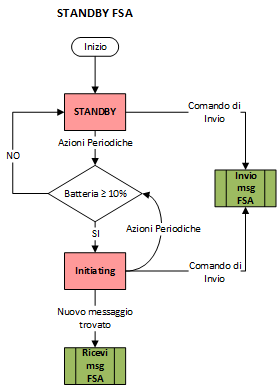
In fase di simulazione, la generazione della rete viene fatta in modo automatico dal simulatore come la distribuzione dei nodi nell’area e la costruzione dei collegamenti tra i nodi. I vari parametri necessari, grandezza dell’area, numero dei nodi e raggio ρ sono inseriti negli appositi file d’inizializzazione. Quando viene eseguito lo scrip di lancio, viene caricato il file di inizializzazione corrispondente al caso di simulazione e nel Network Definition file, che contiene i metodi per la costruzione della rete, vengono prima posizionati tutti i nodi della rete in maniera casuale e poi nella sezione *connections* vi è il metodo che controlla e inserice i collegamenti tra i nodi. Per ogni nodo, controlla quali altri nodi si trovano ad una distanza geometrica inferiore al valore di ρ specificato nel file di inizializzazione e tra essi inserisce un canale di comunicazione con le prestazioni di trasmissione del BLE.

*(ci vuole uno screen della pezzo di codice)*

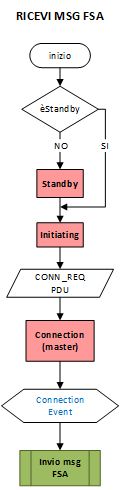
## Algoritmo Dynamic Fanout

Come detto nella Sezione 4.4 la nostra soluzione è una estensione dell’algoritmo di gossip Fixed Fanout, presentato nella Sezione 2.4.4. Abbiamo modificato il metodo di calcolo del limite del numero di trasmissioni che ogni nodo può effettuare, inserendo nuove variabili dinamiche che hanno reso lo stesso fattore dinamico e reattivo ai cambiamenti sia ambientali sia interni al dispositivo. Per questo motivo abbiamo chiamato la nostra soluzione: Dynamic Fanout. Abbiamo esteso inoltre la parte di controllo per la terminazione delle trasmissioni, aggiunto due nuovi metodi che permettano di gestire la nuova forte dinamicità dell’algoritmo, per poter cercare un compromesso tra efficienza e risparmio energetico nella maggior parte degli scenari studiati. I due principali fattori che abbiamo progettato sono il Dynamic Fanout e l’Advertising Limit. Sono due parametri contatore utilizzati per la continua valutazione dello sforzo utile compiuto dal dispositivo in termini di trasmissioni fatte o di tentativi di diffusione dell’informazione fatte.

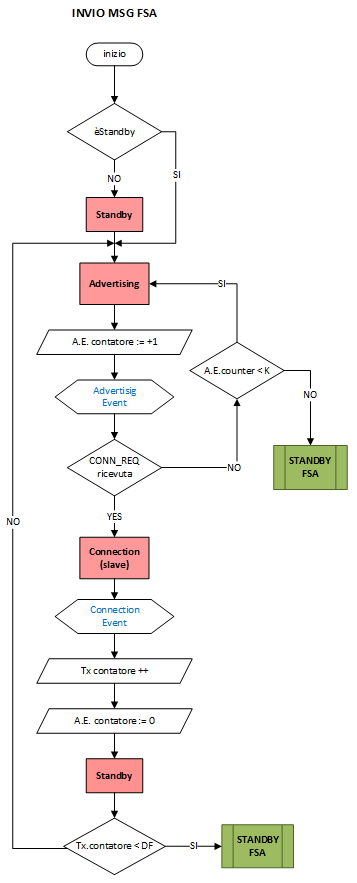
Il nostro algoritmo agisce fondamentalmente basandosi sulla macchina a stati del BLE, cambiando stato all’avvenire di particolari eventi, quali per esempio la presenza di un nuovo messaggio da inviare o ricevere, aggiornando periodicamente lo stato dell’algoritmo con i nuovi parametri aggiornati, controllando che eventuali trasmissioni rimangano nei limiti prefissati dai parametri e quando uno dei due viene raggiunto agire di conseguenza. Abbiamo progettato l’algoritmo in modo che usi il dispositivo finché la sua batteria è più del 10%, poiché non volevamo che l’algoritmo stesso potesse consumare le ultime riserve di energia del dispositivo, lasciando il tempo all’utente di poter raggiungere un eventuale fonte di ricarica o permettergli di continuare a usare servizi più vitali che il dispositivo stesso può offrire. Per questo motivo l’algoritmo se nei suoi controlli periodici rileva che la batteria del dispositivo è inferiore al 10%, ferma il BLE in stato di standby disabilitando la ricezione e l’invio di nuovi messaggi finché il livello di batteria non risale sopra il 10%. L’algoritmo garantisce comunque che qualunque azione o attività in fase di esecuzione, venga terminata anche se durante la sua esecuzione la batteria decresca sotto il 10%. Questo perché abbiamo voluto che l’eventuale trasmissione a ridosso del 10% di batteria venga portata a termine e permetta l’utente di ricevere anche questa informazione. Nelle figure 5.3, 5.4 e 5.5 sono riportati i diagrammi di flusso che descrivono il generale comportamento dell’algoritmo di Dynamic Fanout e sono rispettivamente quello di *Standby*, quello di *Ricezione di un nuovo messaggio* e quello di *Invio di un messaggio*.



*Figura 5.3: Diagramma di flusso della macchina a stati Standby dell'algoritmo DF.*



*Figura 5.4: Diagramma di flusso della macchina a stati Ricevi Messaggio dell'algoritmo DF.*



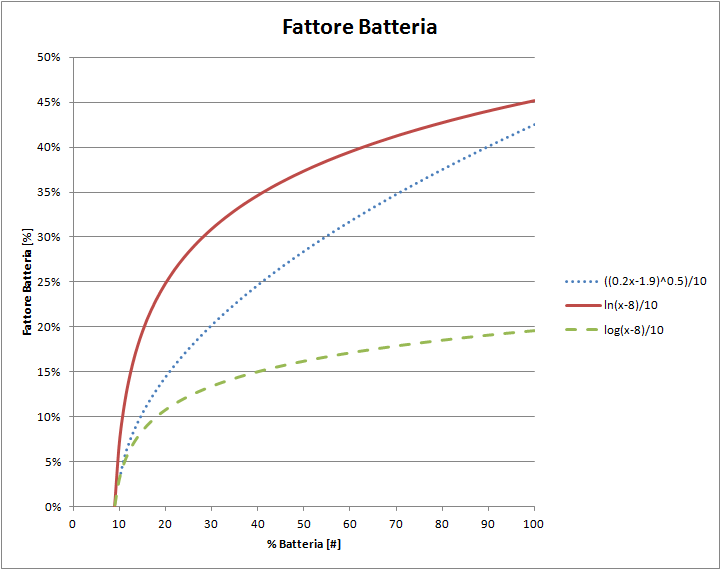
*Figura 5.5: Diagramma di flusso della macchina a stati Invia Messaggio dell'algoritmo DF.*

### Dynamic Fanout

Il Dynamic Fanout ha il compito di fermare il dispositivo dopo un certo numero di trasmissioni effettuate con successo per una certa informazione. Questo valore limite è calcolato in modo dinamico, dipendente dal livello di batteria del dispositivo e dal numero di nodi che il dispositivo riesce a percepire, quindi al numero di nodi ai quali può potenzialmente connettersi. Abbiamo voluto che questo parametro avesse un particolare andamento e rapidità di risposta in particolari situazioni e altri andamenti in altri momenti. Per ottenere ciò abbiamo provato a modellare il comportamento di questo parametro attraverso diverse funzioni ottenendo quindi comportamenti più o meno conservativi, secondo la funzione scelta.

Il calcolo del DF è composto da due fattori: un primo fattore che tiene conto del livello di batteria del dispositivo, chiamato *Fattore Batteria*, un secondo fattore che ha la funzione di correggere l’andamento globale della funzione risultante all’aumentare del numero di nodi. La nostra idea è di ottenere dal Fattore Batteria una percentuale che rappresenta la quantità di nodi, tra quelli percepiti, che il dispositivo prenderà in considerazione come suo limite di trasmissione cioè come suo DF. Più batteria un dispositivo ha più alta sarà la percentuale di nodi cui potremo trasmettere l’informazione. In figura 5.6 riportiamo in grafico tre funzioni batteria studiate, ognuna con un andamento diverso; esse sono rispettivamente:

* (curva 1)
* (curva 2)
* (curva 3)

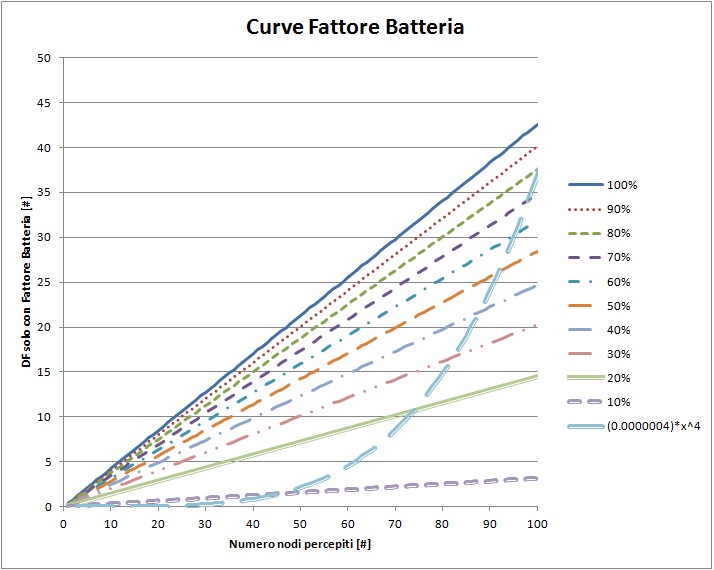


*Figura 5.6: Andamenti Funzioni Batteria.*

Tutte e tre le funzioni sono normalizzate per necessità di ottenere un valore percentuale. Dalla figura 5.6 si nota come diversi tipi di funzioni diano differenti curve di risposta, più o meno conservative e più o meno reattive per valori tra 10% e il 20% di batteria. Noi abbiamo scelto di utilizzare la funzione con la radice quadrata (curva 1) rispetto alla funzione col logaritmo naturale (curva 2), poiché non troppo aggressiva per valori bassi di batteria ma nemmeno troppo conservativa e perché al crescere della percentuale di batteria le due funzioni tendono allo stesso valore massimo o quasi. La curva 3 risulta essere troppo conservativa e risulta non scalare bene con la crescita della rete, non permettendo una buona diffusione delle informazioni.

Fattore Batteria:

Abbiamo progettato il Fattore Batteria come un fattore di “scelta” di una porzione del numero totale di nodi. Un andamento fortemente crescente tra i 10% e il 30% (come in curva 2) fa si che dispositivi con relativamente poca batteria, tentino di inviare l’informazione ad un numero di nodi troppo alto per la quantità di energia che essi possiedono, causando un potenziale sforzo eccessivo e rapida decadenza di durata. Se il numero di nodi percepiti è basso, ciò non crea problemi di efficienza, ma se il numero di dispositivi aumenta, si ha che questi dispositivi abbiamo un fattore di scelta vicino a quello di dispositivi col 60-70% di batteria. Il Fattore Batteria quindi, indica la percentuale della totalità dei nodi percepiti da utilizzare come limite di trasmissioni. Si nota che il FB restituisce una componente che applicata alla quantità di nodi risulta essere linearmente crescente. Ciò è ovviamente inutilizzabile per un numero di nodi che cresce, quindi abbiamo pensato di introdurre un fattore limitante a questa crescita: un Fattore di Correzione. In figura 5.7 sono rappresentate le rette relative ai dieci stadi di batteria e anche la curva rappresentate il fattore di correzione.



*Figura 5.7: Curve Fattore Batteria più cura Fattore di Correzione.*

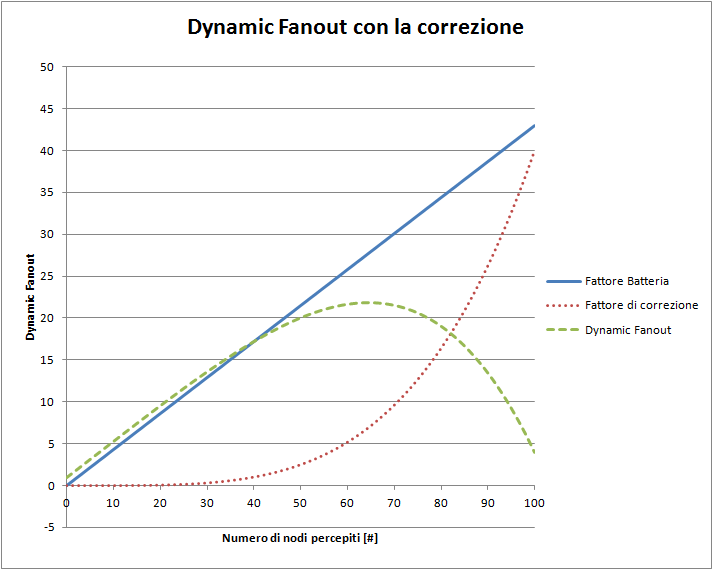
Fattore di Correzione:

Abbiamo progettato il Fattore di Correzione in modo che potesse, come poi si vede nella formula finale, dare il suo contributo limitante per i valori grandi di numero di nodi e risultare trascurabile per i valori più bassi. Abbiamo fatto questa scelta in modo che partendo da valori bassi di numero di nodi, la componente prevalente fosse quella della batteria, mentre al crescere della popolazione la componente prevalente fosse quella dipendente dal numero di nodi. La nostra scelta è basata sull’idea che più nodi ci sono più il carico di lavoro debba essere distribuito tra tutti i nodi senza sovraccaricare un singolo nodo. Inoltre, dividendo il lavoro tra tutti i nodi, si cerca di non avare pochi nodi che intasano i canali di comunicazione disponibili.

Il DF è quindi calcolato unendo il fattore dato dalla batteria e il Fattore di Correzione, tramite la seguente formula:

Dynamic Fanout:

In figura 5.8 è riportato su grafico l’andamento del DF, di uno dei possibili Fattore Batteria e del Fattore di Correzione. Ciò che si ottiene come DF è un andamento crescente fino a raggiungere il massimo, che varia al variare del livello di batteria, e poi decresce a causa della forte prevalenza del Fattore di Correzione. Com’è si può notare dal grafico, per valori superiori a 100 nodi o per bassi livelli di batteria, il valore del DF può risultare nullo o negativo. Per questo motivo abbiamo aggiunto un asintoto orizzontale, valutato solo per valori di nodi oltre quello relativo al massimo DF, così da garantire un DF sempre positivo e sempre dipendente dalla batteria.



*Figura 5.8: Andamento Dynamic Fanout con correzione.*

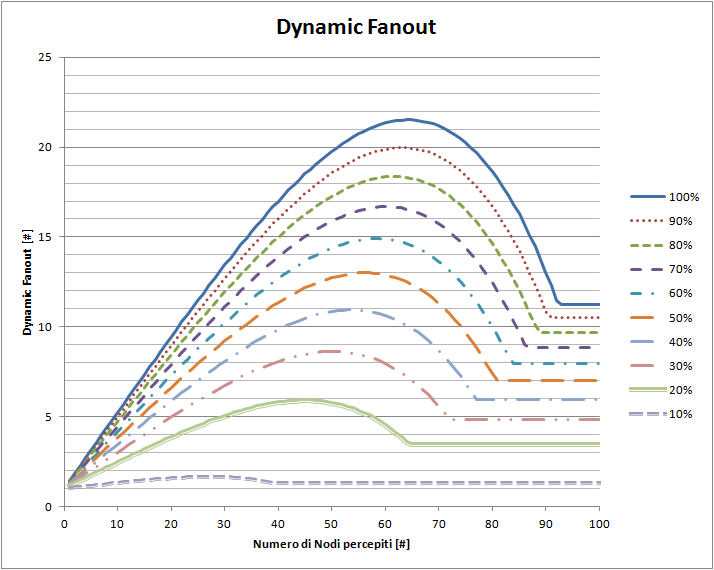
La nostra scelta è basata sull’idea che oltre un certo numero di nodi, si possa trascurare la dipendenza dalla numero di nodi percepiti in quanto ci troveremmo in uno scenario di altissima densità. Quindi abbiamo pensato che è sufficiente avere un limite asintotico dipendente dal livello della batteria in modo da garantire sempre che ogni nodo non compia sforzi eccessivi consumando troppa energia.

Asintoto:

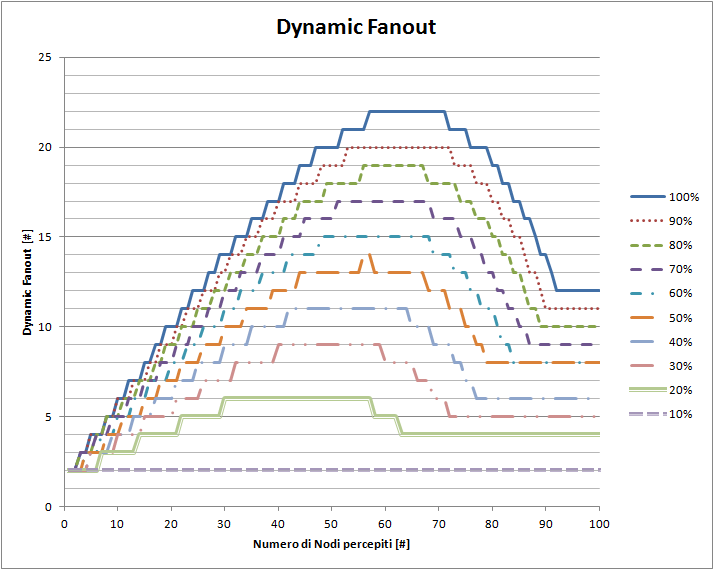
Quest’asintoto è preso in considerazione solo dopo aver superato il massimo della funzione in modo che non generi interferenze per basso numero di nodi. Ricordiamo che il DFMax è dipendente dalla curva, quindi dal livello di batteria, confermando la non dipendenza dell’asintoto rispetto al numero di nodi percepiti.

L’algoritmo valuta due casi: per numero di nodi minore di Xmax, con Xmax il valore di nodi corrispondente al DFmax, prende in considerazione solo la curva, per numero di nodi maggiore valuta il massimo tra la curva e l’asintoto.

Infine in figura 5.9 è riportato su grafico il risultato finale del calcolo del Dyanmic Fanout con asintoti orizzontali. Come prima, sono mostrate solo le dieci curve per i dieci principali livelli di operatività.



*Figura 5.9: Curve del Dynamic Fanout.*



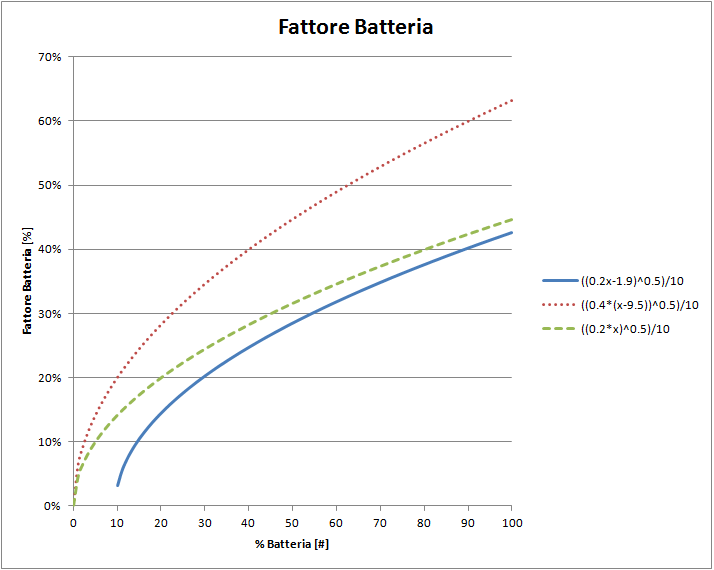
*Figura 5.10: Curve del Dynamic Fanout arrotondate per eccesso.*

Dato che il DF esprime il limite di trasmissioni che un nodo può fare per ogni informazione, non ha senso avere valori razionali perché è il contatore di trasmissioni conti solo per numeri interi. Abbiamo applicato un arrotondamento per eccesso alle curve mostrate in precedenza in figura 5.9. Mostriamo il risultato in figura 5.10. Applicando tale arrotondamento abbiamo ottenuto un innalzamento del valore minimo a 2. Ciò non crea alcun problema, anzi rende l’algoritmo ancora più reattivo per valori piccoli di numero nodi percepiti, mentre resta trascurabile il suo effetto per valori medio - grandi.

Quello riportato in figura 5.10 è il risultato scelto e implementato dall’algoritmo. Abbiamo studiato anche molte alte funzioni con andamenti più o meno conservativi e/o permissivi e/o reattivi. Di seguito riportiamo solo due casi: uno più permissivo e uno più conservativo rispetto alla soluzione scelta, con le relative formule delle funzioni utilizzate. Riporteremo solo i grafici finali, quelli degli andamenti del DF con asintoti senza gli arrotondamenti, per comprendere meglio le caratteristiche delle curve

In figura 5.11 sono riportate una funzione più permissiva e reattiva e una funzione più conservativa, più la funzione utilizzata nella soluzione per poter fare un confronto. Le tre funzioni sono:

* (curva 1)
* (curva 2)
* (curva 3)



*Figura 5.11: Funzioni Fattore Batteria: blu) soluzione, rosso) permissiva, verde) conservativa.*

#### Caso Permissivo

Uno dei casi di DF permissivo nasce dalla sola manipolazione dei termini della funzione utilizzata nella nostra soluzione. Il Fattore Batteria ha coefficiente sulle x pari a 0.4, più grande di quello utilizzato nella soluzione (0,2) e inoltre non ha nemmeno il fattore di traslazione. Ciò lo fa crescere più velocemente, come si può vedere nella figura 5.11.

Fattore Batteria:

Ovviamente aumentando tale coefficiente si ottiene una crescita sempre più ripida, quindi una reattività sempre maggiore. Forzare la reattività del sistema sì può giovare alle prestazioni, ma fino ad un certo punto perché alla fine il limite DF sarà talmente alto per ogni situazione di nodi percepiti che tale limite non sarà mai raggiunto diventando pressoché inutilizzato.

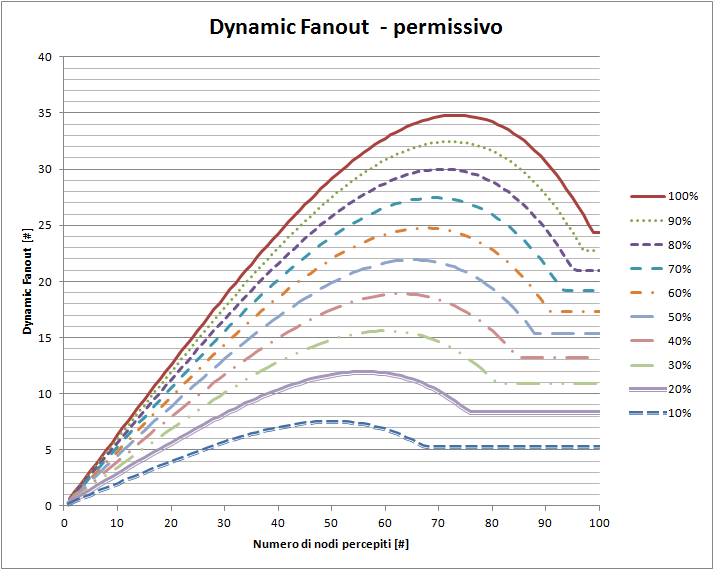
Il Fattore di Correzione rimane lo stesso.

Fattore di Correzione:

L’asintoto invece è stato alzato al 70% del DF massimo:

Asintoto:

Il risultato finale è riportato nel grafico in figura 5.12. Come si può notare dal grafico, abbiamo ottenuto un incremento del valore massimo di circa 60% o addirittura superiore, dipende dalla curva.



*Figura 5.12: Dynamic Fanout permissivo.*

#### Caso Conservativo

Tra le varie casistiche studiate uno dei casi di DF conservativo trovato ha un Fattore Batteria ha simile a quello utilizzato nella soluzione, infatti, il suo andamento come si vede in figura 5.11 (curva 3) è molto simile, ma implementa un Fattore di Correzione diverso che agisce prima.

Fattore Batteria:

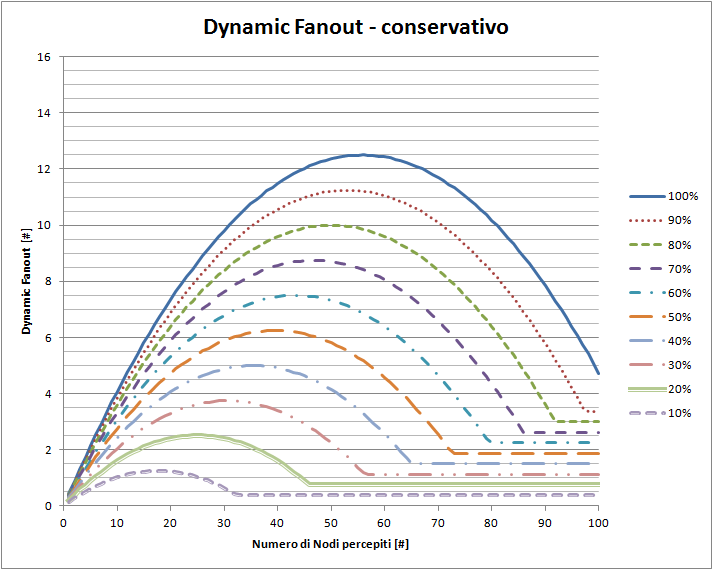
Il Fattore di Correzione è cambiato; ora il suo coefficiente è stato aumentato e il grado della funzione diminuito, per rendere percepibile la sua correzione molto prima.

Fattore di Correzione:

Anche l’asintoto è stato cambiato e scelto più basso, solo il 30% del massimo delle curve.

Asintoto:

Il risultato finale è riportato nel grafico in figura 5.13. Come si può notare dal grafico, abbiamo ottenuto una diminuzione del valore dei massimi da un 40% a un 50% rispetto alla soluzione implementata.



*Figura 5.13: Dynamic Fanout conservativo.*

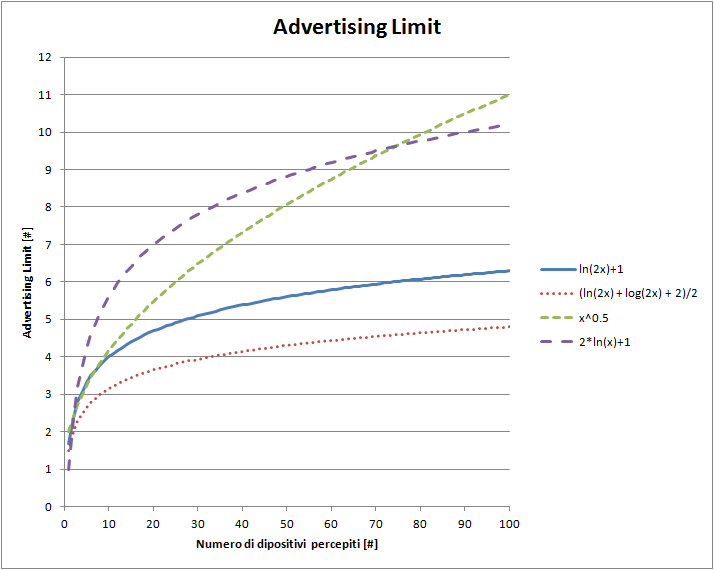
### Advertising Limit

L’Advertising Limit ha il compito di segnalare al dispositivo quando smettere di pubblicizzare un’informazione dopo un certo numero di tentativi di advertising consecutivi senza successo. Questo valore limite è calcolato in modo dinamico, dipendente solamente dal numero di nodi che il dispositivo riesce a percepire, quindi al numero di nodi cui può potenzialmente connettersi. Non abbiamo inserito una dipendenza dal livello della batteria perché la richiesta energetica di un singolo messaggio di advertising è molto bassa, tale da essere trascurabile rispetto al consumo energetico medio del generico dispositivo. Comunque continuare all’infinito a trasmettere qualcosa, anche se richiede poca energia, si traduce in un considerevole consumo. L’obiettivo di questo parametro è di far capire al dispositivo quanto i nodi intorno a lui non sono più interessati all’informazione che esso ha da trasmettere e che quindi può fermarsi e mettersi in ascolto in attesa di altri nuovi messaggi da diffondere.

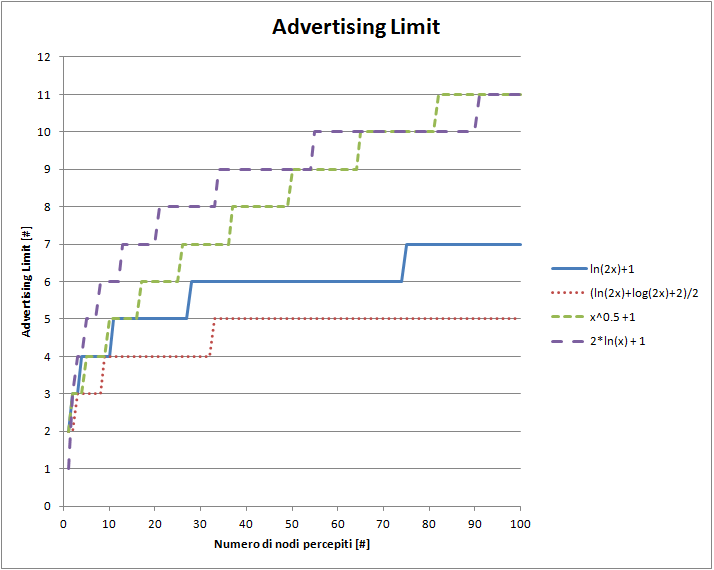
Abbiamo ricercato delle funzioni con cui modellare l’Advertising Limit, che potessero dare a questo parametro il comportamento desiderato. Allo stesso modo del DF, abbiamo pensato che fosse necessario che per valori bassi di nodi percepiti, il parametro avesse una buona reattività nell’aumentare e adattarsi alla rapida crescita della popolazione. Dato che il processo di advertising richiede poca energia, non abbiamo trovato la necessità di cercare forti comportamenti conservativi per valori grandi di nodi percepiti, ma lasciando la funzione col suo comportamento di crescita normale. Questo perché vi sono dei timeout di attesa connessione da parte di chi sta richiedendo l’informazione. Quando un dispositivo pubblicizza un’informazione, tutti i nodi che ricevono la pubblicità e non hanno l’informazione ne faranno richiesta, ma solo il primo richiedente sarà accontentato mentre tutti gli altri staranno in attesa di connessione finché il rispettivo timeout scade. In altre parole mentre un dispositivo è in attesa di connessione, non può sapere se il mittente ha scelto lui come destinatario, l’unica cosa che può fare è aspettare che arrivi il messaggio entro il timeout di connessione; se ciò avviene, avviene la trasmissione dell’informazione, altrimenti una volta scaduto il timeout, il nodo in attesa torna in stato Initiating e ascolta per altri pacchetti di advertising. Durante quest’attesa però, il dispositivo rimane “bloccato” sul possibile mittente ed ignora qualsiasi altra comunicazione come altre pubblicità della stessa informazione fatte da altri nodi. Ci siamo resi subito conto che un solo evento di advertising andato a vuoto non significa assolutamente che tutti i nodi attorno al dispositivo sono già stati tutti infettati. Per questo motivo abbiamo progettato l’Advertising Limit in modo che sia sufficientemente ridondante rispetto al numero di nodi percepiti.

Abbiamo valutato anche per l’AL più funzioni, dai comportamenti più o meno permissivi. In figura 5.14 sono riportate su grafico le seguenti funzioni:

* (curva 1)
* (curva 2)
* (curva 3)
* (curva 4)



*Figura 5.14: Funzioni dell'Advertising Limit.*



*Figura 5.15: Advertising Limit arrotondato per eccesso.*

In figura 5.15 sono riportate le funzioni rappresentate nella figura 5.14, arrotondate per eccesso, utilizzate dall’algoritmo. Come si può vedere da entrambe le figure 5.14 e 5.15 abbiamo studiato quattro funzioni di cui due molto rapide nel crescere per valori bassi di numero di nodi e due meno reattive. La funzione che abbiamo scelto di implementare nell’algoritmo è la seguente (figura 5.14 curva 1):

Advertising Limit:

L’idea generale nel modellare questo parametro è stata quella di cercare di dare reattività nell’adattamento al crescere del numero di nodi percepiti, quando i nodi percepiti sono pochi. Questo perché in questa situazione non si hanno abbastanza nodi nel grafico da poter abbassare il proprio AL, supponendo che altri nodi coprano dove il nodo stesso non arriva. Per ovviare a questa mancanza abbiamo deciso di modellare l’AL in modo che abbia una rapida crescita al crescere nel numero di nodi, fino ai 10-15 nodi, e poi avesse una crescita più lenta e calma in modo da non esagerare per valori alti di nodi. Abbiamo comunque studiato funzioni più aggressive o con comportamento più lineare, rispettivamente rappresentate in figura 5.14 dalle curve 4 e curva 3. La prima è una funzione logaritmica che reagisce molto rapidamente nella parte bassa del grafico per poi rallentare per valori medio grandi, mentre la seconda è una funzione con meno reattività ma che mantiene lo stesso tasso d’incremento.