# Capitolo 6 – Simulazioni e valutazione risultati

### Organizzazione simulazioni

* In cosa consiste la realizzazione
* Come sono strutturate le simulazioni
* I Principali parametri che differenziano e caratterizzano i risultati delle simulazioni

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

La parte di realizzazione sperimentale del nostro progetto, è stata fatta tramite simulazioni con lo scopo di verificare il comportamento del nostro algoritmo e raccogliere dati per fare un’analisi più oggettiva delle sue prestazioni al variare delle condizioni. Per avere una sufficiente base statistica, abbiamo impostato nel file d’inizializzazione il numero di simulazioni a venti. Significa che per ogni configurazione il simulatore eseguirà venti simulazioni consecutive senza reimpostare il generatore di numeri casuali, infatti, in questo modo abbiamo ottenuto ogni volta, venti configurazioni di rete differenti. Per impostare il numero di ripetizioni da fare per ogni configurazione, abbiamo inserito nella sezione *General* di ogni file d’inizializzazione utilizzato, l’istruzione “*repeat = 20*”.

Le simulazioni sono caratterizzate da due parametri principalmente: la densità di nodi e il raggio di trasmissione del BT. Come presentato nella Sezione 5.2, le densità che abbiamo scelto sono:

1. D = 0.02 nodi/m2,
2. D = 0.01 nodi/m2,
3. D = 0.008 nodi/m2,
4. D = 0.001 nodi/m2,
5. D = 0.0005 nodi/m2,
6. D = 0.0001 nodi/m2.

E i raggi ρ dei trasmettitori sono:

* ρ = 15 m,
* ρ = 50 m.

Le densità da D=0.02 nodi/m2 a D=0.001nodi/m2 sono state pensate per simulare ambienti urbani che possono essere dal mediamente popolati al densamente popolati. Situazioni simili possono essere il caso di medie città, grandi città o per le densità più elevate, situazioni di forte concentrazione di persone in un’area ristretta; quest’ultimo tipo di situazione può essere generata da forte concentrazione abitativa come una serie di palazzi o condomini vicini tra loro, ma anche da motivazioni esterne alla semplice concentrazione abitativa come possono essere eventi di qualunque genere. Le densità più piccole, D=0.0005 nodi/m2 e D=0.0001nodi/m2 sono state pensate per studiare la scalabilità del sistema a fronte della dispersione dei nodi su una vasta area, ma anche per simulare situazioni urbanistiche tipiche di piccoli comuni, con un relativo basso numero di abitanti come ad esempio i piccoli paesi in pianure o ampie vallate. Abbiamo quindi organizzato il processo di realizzazione sperimentale in modo da eseguire, per ogni densità, simulazioni per entrambi i raggi ρ e collezionare dati per un’analisi a posteriori delle performance.

Il principale risultato che abbiamo voluto monitorare è stato la percentuale di diffusione di un messaggio, al diminuire della densità, per studiare quali fossero i limiti di applicabilità in termini di efficienza sistema e per avere un insieme di scenari in cui l’algoritmo da noi proposto presentasse buoni livelli di prestazioni. Parallelamente abbiamo anche raccolto dati sul tempo totale di trasmissione, inteso come l’istante di tempo in cui l’ultima trasmissione è conclusa. Abbiamo scelto di raccogliere anche quest’ultimo dato perché abbiamo cercato di capire con che tempistica il messaggio è diffuso. Il motivo è stato di voler dare un corrispettivo valore temporale alla percentuale di nodi che l’algoritmo raggiunge. Da solo un valore di tempo massimo di trasmissione non è valutabile, poiché è molto dipendente dalla dispersione spaziale della rete, infatti, esso deve essere analizzato in coppia con il valore percentuale di rete coperta durante la diffusione per la rispettiva simulazione presa in analisi. In questo modo possiamo avere un’idea di come il tempo di propagazione evolva al variare della copertura e della densità della rete.

### Algoritmo Dynamic Fanout: funzionamento

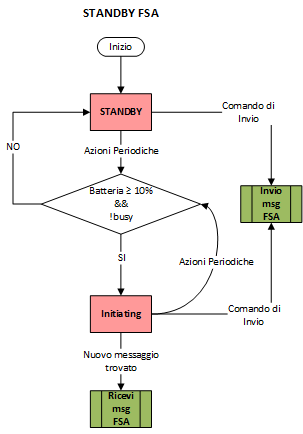
* Caratteristiche dell’algoritmo, come funziona e quali/come vengono eseguite certe operazione in fase di simulazione
* Diagrammi di flusso dell’algoritmo

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In questa sezione diamo una breve spiegazione sul funzionamento dell’algoritmo all’atto pratico dell’esecuzione. Presenteremo il suo comportamento tramite diagrammi di flusso, uno per ognuna delle possibili situazioni che l’algoritmo si può trovare ad affrontare. Ogni diagramma rappresenta il comportamento della relativa macchina a stati virtuale che descrive il comportamento dell’algoritmo in quella specifica situazione. I tre principali automi sono i seguenti: *Standby*, *Invio Messaggio* e *Ricevi Messaggio*. Vi è anche una quarta parte che è eseguita in parallelo ai primi tre, che si occupa solo di organizzare e temporizzare l’esecuzione delle azioni periodiche.

#### Standby fsa

Partiamo con la *Standby fsa*, il diagramma che descrive il comportamento dell’algoritmo durante i periodi di attesa e ascolto. Il diagramma riportato in Figura 6.1, descrive il generale comportamento del dispositivo quando esso non è impegnato in nessun tipo di scambio dati attivo. Dato che lo stato di Standby è raggiungibile da qualsiasi stato, l’algoritmo forza senza alcun controllo l’entrata in quello stato. Poi, in maniera temporizzata, l’algoritmo esegue delle azioni periodiche che hanno un duplice scopo: aggiornare la visione dello stato circostante e aggiornare i parametri di conseguenza e simulare il consumo energetico del dispositivo nel tempo. Dopo aver eseguito queste operazioni periodiche, l’algoritmo compie un controllo sulla batteria del dispositivo. Se essa è superiore al 10%, il sistema si porta in stato di ascolto (Initiating) in attesa di eventuali nuovi messaggi, mentre se inferiore alla soglia limite l’algoritmo considera il dispositivo fuori dalla zona di lavoro e lo riporta in stato di Standby. Continuerà comunque a eseguire le azioni periodiche e qualora la batteria salisse nuovamente sopra il 10%, il sistema tornerebbe operativo.



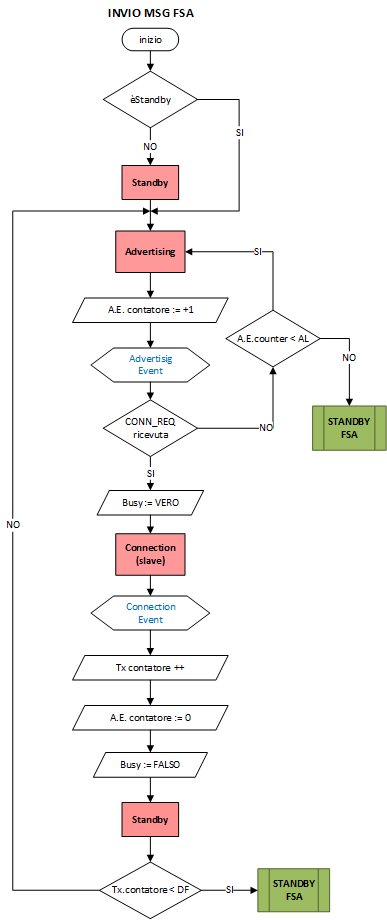
*Figura 6.1: Diagramma di flusso della Standby fsa.*

Il sistema, oltre al livello della batteria, controlla anche che il dispositivo non sia “*busy”* (occupato) in quale operazione o scambio d’informazione. Quest’ulteriore controllo serve a garantire il completamento delle operazioni o transazioni iniziate quando la batteria era sopra il 10% e durante l’esecuzione è scesa sotto la soglia limite. E’ anche vero che lo standard BLE da solo implementa un insieme di controlli per rilevare connessioni perse, ma in questo facciamo si che l’algoritmo garantisca che anche quest’ultima operazione d’invio o ricezione sia completata e non interrotta.

Sia se il dispositivo si trovi in stato di Initiating sia se si trovi in quello di Standby, se l’utente intende inviare un messaggio, il sistema si sposta a eseguire un’altra serie di operazioni. Il sistema permette un invio di un nuovo messaggio generato dall’utente, anche se la batteria è inferiore alla soglia limite, poiché crediamo che negli scenari da noi ipotizzati, se l’utente vuole iniziare a diffondere un nuovo messaggio, sia una valida ragione per eseguire l’operazione anche in condizioni di scarso livello energetico. In questa particolare situazione, l’algoritmo avrà comunque un DF pari a uno o al massimo pari a due, quindi si cerca solo di far compiere uno sforzo al dispositivo per iniziare la diffusione del messaggio e non molto di più.

#### Invio Messaggio fsa

Il diagramma in figura 6.2 illustra il funzionamento del sistema quando esso cerca di inviare un nuovo messaggio arrivatogli dalla rete o che l’utente vuole inviare. Per prima il sistema controlla di essere nello stato di standby e nel caso non lo fosse, ci si porta. Una volta in Standby è possibile passare allo stato di Advertising.



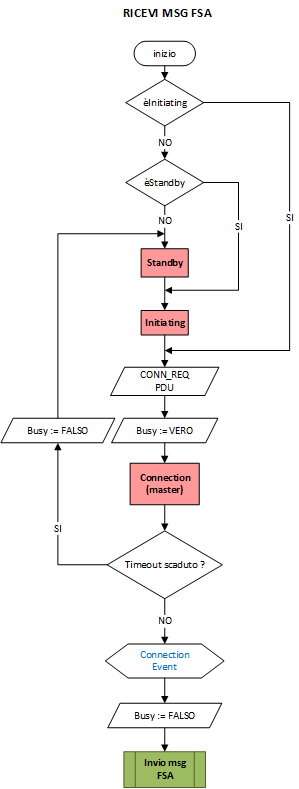
*Figura 6.2: Diagramma di flusso Invio Messaggio fsa.*

Dopo essere entrato in Advertising, è incrementato il contatore degli advertising e poi è eseguito l’advertising event vero e proprio. Il sistema si mette quindi in attesa di eventuali pacchetti di tipo CONN\_REQ in riposta ai suoi pacchetti. L’attesa di un pacchetto di CONN\_REQ è regolata da un timeout, quindi l’attesa non sarà mai infinita. Il significato di quel preciso nodo decisionale significa appunto se è stata ricevuta una richiesta di connessione entro il timeout. Se il sistema non riceve una richiesta di connessione entro lo scadere del timeout allora controlla se il contatore di advertising è inferiore all’AL (Advertising Limit); se lo è si riporta in stato di Advertising, incrementa nuovamente il contatore e ricomincia, se invece il contatore è maggiore o uguale all’AL, il sistema si ferma e ritorna a seguire il flusso d’istruzioni della Standby fsa, mettendosi poi in ascolto di eventuali nuove informazioni. Questo pezzo di flusso esecutivo rappresenta una delle due parti di controllo dinamico che abbiamo progettato per estendere e adattare l’algoritmo originale al nostro problema.

Se invece il sistema riceve una richiesta di connessione entro il timeout, esso si marca come occupato e passa nello stato di Connection in *Slave Role*. A questo punto arriverà il primo pacchetto di *pool* da parte del *Master* *Role*, inizierà il connection event e al suo interno verrà trasmetta tutta l’informazione. Una volta terminato il connection event, il sistema incrementa il contatore di trasmissioni effettuate con successo, azzera il contatore degli advertising a vuotoconsecutivi e resetta lo stato di busy a falso. In seguito si porta in Standby e controlla se il contatore di trasmissioni effettuate è minore del DF. Se sì, il sistema ricomincia tutta la procedura d’invio messaggio, altrimenti vuol dire che ha completato il suo lavoro per quell’informazione e può tornare a eseguire la Standby fsa e mettersi in ascolto per la prossima informazione.

#### Ricevi Messaggio fsa

Nel caso vi sia uno o più dispositivi che stanno tentando di diffondere informazioni che il nodo in questione non ha ancora ricevuto, esso inizia a eseguire la sequenza d’istruzioni del Ricevi Messaggio fsa. Dallo stato di Initiating del diagramma della Standby fsa, si passa al diagramma in figura 6.3. All’inizio il dispositivo si porta correttamente in stato di Initiating se già non lo era e poi invia una richiesta di connessione al mittente dei pacchetti di advertising. Poiché sta tentando di instaurare una nuova connessione con un determinato dispositivo, dopo aver mandato la richiesta di connessione, è messo a vero il flag busy. Dopo esser entrato in stato di Connection in *Master Role*, il sistema invia il pacchetto di *pool* necessario ad autorizzare lo slave a iniziare il trasferimento dati. Dopo aver inviato la pool request, il dispositivo rimane in attesa dell’informazione; se scatta il timeout vuol dire che il mittente non ha accettato la sua richiesta oppure qualche altro dispositivo ha inviato una richiesta di connessione ed è stato scelto (la politica di accettazione è FIFO, anche se non vi è una vera e propria coda). Nel caso il timeout scade, il dispositivo capisce di non essere stato scelto, quindi imposta a falso il flag busy e si riporta in stato Initiating, di ascolto. Nel caso invece la sua richiesta sia la prima ad arrivare e venga accettata, dopo la pool request, il dispositivo slave manda al master l’informazione durante il Connection Event. Al termine del trasferimento la connessione è chiusa e il sistema automaticamente inizia a eseguire la sequenza d’istruzioni dell’Invio Messaggio fsa.



*Figura 6.3: Diagramma di flusso Ricevi Messaggio fsa.*

#### Azioni Periodiche

Il sistema ha un processo in background, che schedula delle istruzioni da eseguire periodicamente e queste azioni servono per aggiornare i parametri utilizzati in modo che dinamicamente si adattino all’ambiente esterno e allo stato attuale del dispositivo stesso, ma anche per compiere dei controlli. Il controllo principale è quello fatto sulla batteria. Il sistema periodicamente rileva lo stato della batteria e se il livello è sopra la soglia limite (10% soglia limite), il sistema, se può, porta il dispositivo in stato di Standby rendendolo inabile a partecipare alla diffusione dei messaggi. Il motivo è di non lasciare al sistema la possibilità di consumare quell’ultima porzione di batteria e di lasciarla all’utente per le sue necessità. L’unica eccezione è di permettere all’utente, anche se la batteria è sotto la soglia limite, di inviare un messaggio. Abbiamo ritenuto necessario lasciare questa possibilità, perché negli scenari di lavoro da noi analizzati, potrebbe essere utile per diramare qualche informazione d’emergenza molto utile, anche a costo della rimanente poca batteria rimasta.

Questi controlli avvengono periodicamente in background, anche durante eventi di trasmissione o ricezione. Per questo motivo abbiamo inserito il flag busy, in modo da segnalare qualora il sistema sia impegnato in una transazione e quindi non sia possibile da parte del controllo periodico forzare uno stato Standby. Questo perché non vogliamo che vada a intralciare il sistema mentre è in corso una transazione, ma può comunque agire tra una trasmissione e l’altra.

Anche quando la batteria è sotto la soglia limite, i controlli sono sempre effettuati, grazie alla loro esigua richiesta energetica e anche per riabilitare il sistema al lavoro attivo qualora la batteria sia ricaricata e torni sopra soglia.

Una parte dei controlli periodici è già stata inserita e rappresentati nel diagramma di flusso della Standby fsa, per dare una semplice idea di quale sia il compito di questi controlli, ma non sono vincolati a nessun stato né macchina a stati; come detto prima queste azioni periodiche sono eseguite sempre e comunque in background nel sistema.

### Valutazione risultati

* Valutazione risultati
* Sottolineare che il decadimento delle prestazioni avviene per densità più basse di un paesino di campagna
* Riferimento al documento delle prestazioni (da ritrovare) in cui dice che per algoritmi rumor mongering non è possibile stabilire una complessità, a causa delle possibili scelte nei metodi di terminazione.
* Il nostro caso presenta estrema variabilità e quindi non è possibile stabilirne con che legge si diffondano le informazioni né in quanto tempo.
* Dire che In una ipotesi ideale avremmo certi tipi di comportamenti
* Ribadire che vi sono altri fattori non considerati, quali la mobilità dei nodi, che possono aumentare l’efficacia dell’algoritmo.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

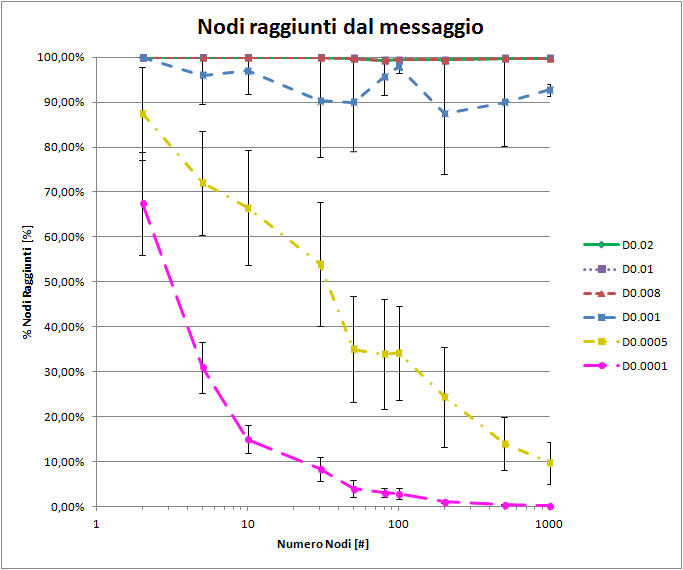
Dopo aver terminato le simulazioni, abbiamo raccolto i dati ed eseguito analisi su di essi. I parametri che abbiamo monitorato come risultati, a fronte dell’invio di un singolo messaggio sono stati:

* Percentuale di contagio che il messaggio ha raggiunto.
* Tempo totale impiegato per raggiungere tale copertura di rete.

Grazie alle funzionalità offerte dal framework di simulazione OMNeT++, è stato possibile raccogliere i dati in maniera efficiente. Con una manipolazione dei dati appena raccolti, il framework è stato in grado di fornirci anche il valor medio, la varianza, la deviazione standard e anche l’intervallo di confidenza al 95%. Abbiamo anche estratto dai dati raccolti, gli stessi indici tramite un foglio di calcolo Excel, in modo da poter validare ed eventualmente correggere eventuali errori o approssimazioni del software di simulazione.

Prima di parlare dei singoli risultati, ripetiamo alcuni aspetti che hanno caratterizzato l’algoritmo, i modelli e la simulazione stessa. Per avere una maggior variabilità nella simulazione, abbiamo assegnato molti valori in maniera casuale, ove possibile, come ad esempio la disposizione dei nodi nella rete, il livello di batteria iniziale di ogni dispositivo e, a ogni ripetizione, era scelto causalmente anche il nodo che possedeva l’informazione da divulgare. Per ogni densità, sono stati simulati i due range ρ scelti, ρ=15m e ρ=50m, e poi è stata fatta la media tra i due. Inoltre la diversità di prestazioni dovuta alla differenza di raggio ρ utilizzato è elevata. Questi aspetti saranno ripresi quando illustreremo i risultati ottenuti.

Il primo parametro che presentiamo è quello della percentuale di rete contagiata dal messaggio. In figura 6.4 sono riportati su grafico i valori medi dei dati raccolti relativi alla copertura raggiunta dal messaggio, per ogni densità simulata. Per ogni curva abbiamo anche riportato sul grafico, il suo intervallo di confidenza al 95%, per comprendere la distribuzione attorno ai valori medi. La prima cosa che si nota è che per le densità d=0.02, d=0.01 e d=0.008 la percentuale di contagio è intorno al 100% ed è indicativo di un’ottima efficienza prestazionale della soluzione da noi proposta in ambienti densamente popolati o in momenti di alta concentrazione urbana per motivi particolari. Per queste tre densità molto alte entrambi i raggi ρ hanno dato ottimi valori. Le tre densità più piccole invece hanno evidenziato i limiti di operabilità del sistema. Per la densità d=0.001, abbiamo ottenuto ancora degli ottimi valori in termini di copertura, ma in questo caso, il raggio ρ=15m ha cominciato a evidenziare problemi nello stabilire collegamenti tra i nodi e quindi raggiungere un’ottima copertura. Per questo motivo abbiamo ottenuto un intervallo di confidenza già ben più ampio rispetto alle tre casistiche precedenti nelle quali era pressoché nullo o molto piccolo. Nel complesso rimane comunque stabile in un intorno del 90% al crescere del numero dei nodi nella rete.

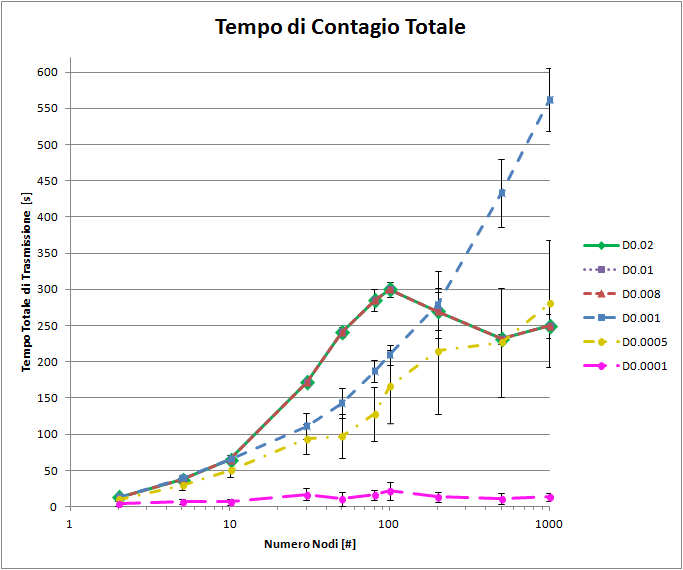


*Figura 6.4: Risultati della percentuale di contagio raggiunta.*

Poi abbiamo la curva per la densità d=0.0005. Questo valore di densità, rispetto al precedente, presenta un andamento decrescente, al crescere del numero di nodi. Già a questo livello di densità, abbiamo riscontrato che al crescere della rete, cresce la probabilità che si formino sottoreti isolate che causano solo una diffusione parziale del messaggio in rete. Rispetto alla densità precedente che era il doppio, abbiamo avuto un forte deterioramento nella copertura della rete e un più ampio intervallo di confidenza. Con questo valore di densità, si ha che al crescere del numero di nodi, essi siano troppo diradati nella rete e nel complesso si vengano a creare sottoreti sempre più piccole col crescere del numero di nodi. Con le ipotesi fatte, non vi è modo per l’algoritmo di eludere o superare quest’ostacolo. Nonostante quest’andamento decrescente, possiamo vedere che con mille nodi nella rete, il sistema sarebbe in grado di coprire una porzione di rete intorno al 10%; per questo motivo la densità d=0.0005 può essere considerata un limite inferiore di operatività. Infine l’ultima densità d=0.0001 si dimostra essere oltre le possibilità operative del sistema, infatti, mostra un degrado prestazionale forte già con reti piccole composte di soli dieci nodi.

Dalle nostre simulazioni è emerso che il sistema è in grado di gestire e operare a livelli ottimi, copertura intorno al 90% circa, fino a densità d=0.001 ma può operare, se pur con un degrado prestazionale linearmente dipendente dalla grandezza della rete, anche fino a una densità minima di d=0.0005.

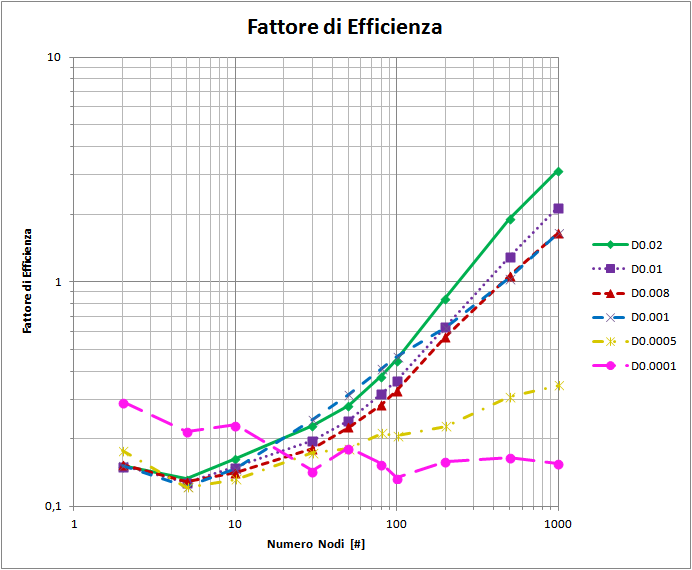
In figura 6.5 riportiamo i risultati riguardanti il tempo che l’algoritmo ha impiegato per completare l’ultima trasmissione di contagio a lui possibile. Dalle nostre analisi è risultato che per densità basse come d=0.02, d=0.01 e d=0.008, l’alta vicinanza a permesso di contenere il tempo necessario e dopo un picco in corrispondenza di cento nodi, esso diminuisce pur garantendo una copertura totale o quasi. Questo grazie all’alta ridondanza di collegamenti creatasi dalla vicinanza dei nodi; in questo modo la rete stessa riesce a evitare la formazione di colli di bottiglia tra gruppi di nodi. Per d=0.001 invece, abbiamo un andamento crescente col numero di nodi, anche se in figura 6.4 si vede come resti sempre vicino alla copertura totale. In questo caso abbiamo un andamento di tipo quadratico, al crescere del numero di nodi nella rete. Già passando da d=0.008 a d=0.001 si può notare come cambi il tempo necessario all’algoritmo per raggiungere la sua massima copertura. Infine, le densità d=0.0005 e d=0.0001 presentano curve inferiori alle precedenti ma vanno lette in combinazione con i risultati mostrati in figura 6.4. Notiamo che l’andamento, almeno per la curva d=0.0005, riamane di tipo quadratico, indice di una rete sempre più diradata nella quale si perdono sempre più i collegamenti ridondanti tra i nodi e compaiono sempre più sottoreti e singoli collegamenti tra essi, che sono il punto debole dell’algoritmo.



*Figura 6.5: Risultati del Tempo Totale di Contagio.*

Se al crescere del numero di nodi, la percentuale di rete coperta diminuisce, significa che il sistema esegue sempre meno trasmissioni, per questa ragione le curve per d=0.0005 e d=0.0001 sono inferiori alle altre. Per la curva di densità più bassa d=0.0001, possiamo solo da dire che conferma i dati rilevati della copertura. L’impossibilità eseguire trasmissioni a causa di una rete troppo sparsa, fa sì che l’algoritmo perda efficacia e si fermi dopo pochi secondi di operatività.

Per valutare meglio l’efficacia della propagazione, abbiamo anche analizzato il rapporto tra la copertura raggiunta, in termini di numero di nodi, e il tempo impiegato per raggiungerla. In figura 6.6 riportiamo i risultati su grafico. Dal grafico si nota come tutte le densità tra d=0.02 a d=0.001 siano molto simili in termini di efficienza. L’estrema vicinanza tra i nodi per d=0.02, rende l’algoritmo molto efficace, ma anche scendendo fino a d=0.001 l’efficienza dell’algoritmo rimane buona e l’andamento rimane lo stesso delle densità più alte. Per le due densità limite invece si nota come l’efficienza per la d=0.0005 fatichi ad aumentare all’aumentare del numero di nodi, mentre per la d=0.0001 si noti come continui a decrescere, fatta eccezione per qualche piccolo picco e per il valore superiore per n=2. Quest’ultimo fatto è dovuto a una scelta di progettazione, per agevolare l’analisi dei dati. Quando il primo nodo è inizializzato col messaggio da inviare, il sistema registra il tempo di consegna come l’istante del primo contagio che è sicuramente inferiore al tempo necessario per contagiare l’altro nodo, in una rete formata da due soli nodi. Per questo motivo, per reti piccole risulta avere un’efficienza più alta. All’aumentare della grandezza della rete la curva per d=0.0001 si stabilizza e non presenta più il problema, rappresentando il corretto andamento.



*Figura 6.6: Rapporto tra numero di nodi contagiati e tempo impiegato per farlo.*

Dopo l’analisi dei dati raccolti, è difficile poter dire con che legge si comporta il nostro algoritmo, per vari motivi. Il primo tra tutti è che non abbiamo sempre una situazione in cui ogni nodo sia connesso in un unico grafo. Le diverse densità possono creare più sottografi isolati senza la possibilità che questi ultimi possano comunicare. Un altro fattore importante è che i nodi del grafo sono dispositivi con una batteria e l’energia a disposizione si consuma nel tempo. Nodi in posizioni critiche della rete possono esaurire la batteria e così escludere parte della rete. Sul fattore batteria sono calibrati i parametri del sistema, e i limiti di trasmissione. Un nodo con limiti di trasmissione bassi, potrebbe completare il suo lavoro di trasmissione, ma verso una zona di rete già abbastanza informata e non in una zona di rete in cui l’informazione non è ancora arrivata. Un altro fattore molto importante e influente, è il metodo (o i metodi) di terminazione che si è scelto di utilizzare. Differenti metodi con differenti criteri possono portare ad avere prestazioni molto differenti.

In condizioni ideali, tutti i nodi connessi in un unico grafo e batteria infinita per tutti i dispositivi, come presentato in [*articolo\_complessità\_algoritmi.pdf*], gli algoritmi di rumor mongering con strategia di diffusione push&pull hanno queste caratteristiche:

* L’informazione viene diffusa a tutti dopo *O(ln n)* cicli.
* La copertura completa richiede *O(nln ln n)* trasmissioni.

Per gli algoritmo di rumor mongering, le complessità appena elencate sono completamente indipendenti dal modo in cui vengono scegli i nodi con cui comunicare e le distribuzioni di probabilità con cui i nodi vengono scelti. Per questo motivo, questi algoritmi lavorano allo stesso modo sia se hanno una visione completa della rete sia se hanno una visione parziale della rete.

Aspetti che non sono stati considerati in questo studio e che possono avere un impatto sulle prestazioni sono: la mobilità dei nodi e una distribuzione dei nodi nell’area non più totalmente uniforme, ma con una logica più vicina al centro abitativo. Le zone abitative sono costruite vicino ad altre già presenti e non in maniera uniforme su un determinato territorio. La mobilità può creare situazioni riconducibili ai veri contagi epidemici. Un nodo a conoscenza di un’informazione può spostarsi in una zona, anche lontana dal punto di partenza, dove l’informazione sarebbe impossibilitata ad arrivare e iniziarne la diffusione. Questo porterebbe a ridurre i problemi generati dalla bassa densità e dalla conseguente formazione di sottoreti isolate.