# Capitolo 2 – Stato dell’arte

## Risparmio energetico

## Bluetooth 4.0 Low Energy

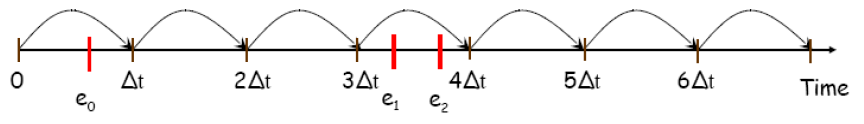
## Algoritmi di Gossip

## Modelli di Rete

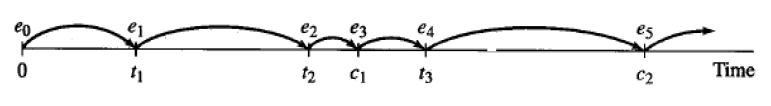
## Simulatori

La nostra ricerca si è focalizzata su simulatori di reti Peer-2-Peer, reti in cui tutti i nodi della rete sono allo stesso livello, sono paritetici e non vi è presenza di gerarchie tra nodi, e anche su simulatori di protocolli che fossero sia basati su cicli sia su eventi. In generale simulare qualcosa prevede l’esecuzione di una sequenza di operazioni nel tempo e magari le entità che eseguono tali operazioni sono molteplici. Per poter gestire l’esecuzione contemporanea e/o concorrente di più entità il simulatore deve stabilire l’ordine in cui eseguire quale istruzione di quale nodo. La scelta sul come venga stabilito l’ordine di esecuzione caratterizza il motore del simulatore che potrà essere basato su cicli, su eventi o su entrambi. Una simulazione basata su cicli prevede che si definisca uno “time step” di simulazione, che è l’incremento del tempo virtuale di simulazione che definisce la cadenza di ogni ciclo. Tutte le operazioni definite come periodiche verranno eseguite ad ogni ciclo, contemporaneamente. Inoltre qualsiasi evento non periodico che occorre tra un ciclo e un altro verrà schedulato al ciclo successivo. Come si vede in figura 2.1, gli eventi e0, e1 ed e2 accadono tra un ciclo e il successivo, con un passo di simulazione di Δt. L’evento e0 verrà quindi processato all’instante Δt, mentre gli eventi e1 ed e2 verranno processati all’instante 4Δt. In questo modo si perde la distribuzione degli eventi, avendo lo scheduler che appiattisce la distribuzione temporale su un unico ciclo, ad ogni iterazione. Questo tipo di simulazioni sono molto vincolanti e non permettono una simulazione molto realistica di una generica classe di nodi. Si adatta molto bene a simulazioni di operazioni batch e/o periodiche. Per poter avere più profondità nell’esecuzione degli eventi vi è lo scheduling guidato dagli eventi che, come dice il nome, ordinerà le operazioni da eseguire in base al momento in cui esse hanno generato o genereranno un evento.

In figura 2.2 è riportato un esempio di scheduling guidato dagli eventi. Il tempo di simulazione non sarà più una progressione costante, ma seguirà il susseguirsi degli eventi, nell’ordine in cui essi avvengono. In questo caso la simulazione avrà inizio con l’evento e0, poi il simulatore salterà all’instante di tempo t1 in quanto lo scheduler prima di t1 non ha messo nessuna operazione da eseguire.



*Figura 2.1: Scheduling basato su cicli.*



*Figura 2.2: Scheduling guidato dagli eventi.*

All’istante t1 avviene l’evento e1 e viene processato, poi si passerà all’istante di tempo t2 perché l’evento e2 è previsto che avvenga in quell’istante e così via tutti gli altri. In generale gli eventi sono relativi per esempio alla ricezione di un messaggio da parte di un nodo. In figura 2.2 vediamo che vi sono gli eventi e3 ed e5 che sono schedulati agli istanti di tempo c1 e c2. Questo è un esempio di integrazione di eventi ciclici in un ambiente guidato dagli eventi. Gli eventi e3 ed e5 sono azioni periodiche e quindi devono essere eseguite ogni “time step” specificato, ma come fare se non vi è un “time step” di simulazione? Il simulatore genererà un evento ogni “time step” ti tempo di simulazione associando come evento l’azione periodica. Ciò garantisce di poter avere con un unico motore di scheduling, sia eventi aperiodici che azioni periodiche. In questo caso abbiamo un simulatore basato sia sugli eventi sia sui cicli.

Navigando sul web, abbiamo trovato tanti simulatori di reti come Mosquite [http://www.mesquite.com/] che è una libreria per CSIM, PADS [http://pads.cs.unibo.it/doku.php?id=start] sviluppato dal Dipartimento di Ingegneria Informatica dell’Università di Bologna in grado di simulare protocolli complessi su reti di larga scala o senza una ben definita struttura. Vi è anche PeerSim [http://peersim.sourceforge.net/], una libreria Java che fornisce gli strumenti di simulazione basata sia su cicli sia su eventi. Abbiamo infine trovato anche OMNeT++ [http://omnetpp.org/], un framework costruito sulla piattaforma Eclipse ma scritto in C++. OMNeT++ è uno simulatore basato solo su eventi ma dato il suo intenso sviluppo in ambito commerciale, vi sono disponibili diverse librerie e tool in grado di offrire molte funzionalità di simulazione di protocolli di rete e diverse tipologie di reti.

## PeerSim

PeerSim è un software sviluppato in ambiente accademico che consente di simulare reti con una alta flessibilità sul numero di nodi che compongono la rete, rendendolo molto adatto a simulare reti che scalano molto facilmente. PeerSim poi necessita che venga definito lo stack di protocolli che ogni nodo della rete implementerà; infatti la logica con cui questo simulatore costruisce la rete da simulare è quella di istanziare solo un nodo e poi esso venga copiato, o clonato, tante volte quanto fino a raggiungere il numero di nodi nella rete. Per questo motivo si nota subito che PeerSim è uno strumento ottimizzato per la simulazione di reti anche di grandi dimensioni e soprattutto, come si può dedurre dal nome, reti soprattutto P2P. L’output deve essere creato dall’utente, rendendo quindi lo stream in output su console ristretto alle informazioni di simulazione più le informazioni volute dall’utente. Non presenta alcun tipo di strumento grafico per la rappresentazione della rete o dei pacchetti in transito tra i nodi, la simulazione è solo computazione e stampa di messaggi su console. Vi è la possibilità di utilizzare o estendere alcune funzioni di alcuni componenti del sistema di simulazione che permettono di stampare su file di testo le coordinate dei nodi della rete e/o i rami del grafo della rete nel formato usato da GNUplot [http://www.gnuplot.info/] e quindi poi visualizzare il layout di rete. PeerSim necessita che l’utente definisca in un apposito file di configurazione le specifiche di simulazione, della rete, dei protocolli implementati sui nodi, delle dipendenze tra i protocolli, i componenti di inizializzazione, i componenti di controllo e i componenti di osservazione. I componenti di inizializzazione sono componenti che il simulatore eseguirà solo all’inizio del processo di simulazione e hanno il compito di inizializzare i nodi della rete e la rete stessa, ad esempio stabilire le connessioni tra i nodi. I componenti di controllo sono “agenti” che vengono eseguiti ciclicamente e hanno il compito di agire sulla rete durante la simulazione, per introdurre dinamicità nella stessa come per esempio accendere o spegnere nodi o canali di trasmissione per simulare malfunzionamenti oppure aggiungere o togliere nodi e collegamenti. Questi componenti di controllo sono opzionali. Infine i componenti di osservazione detti anche osservatori sono componenti che operano alla fine della simulazione oppure ciclicamente se resi cycle-based impostando il loro time step. Il loro compito è quello appunto di osservare la rete e permettere di raccogliere dati per analisi. I dati devono raccogliere devono essere specificati dall’utente nell’apposita funzione di osservazione che viene eseguita dal motore di simulazione. Un esempio di osservatore è il componente che permette di scrivere su file il layout della rete in formato compatibile con GNUplot. Punto forte di questo simulare quindi è la flessibilità sul numero di nodi della rete, le ottime prestazioni di simulazione anche per grandi reti e la scelta di lasciare all’utente l’estensione dei componenti che governano la simulazione.

## OMNeT++

OMNeT++ è un framework e libreria di simulazione basata sul linguaggio C++, estendibile e modulare. OMNeT++ è un software di simulazione molto diffuso sia nel settore commerciale sia nel settore scientifico per la simulazione di reti e protocolli di trasmissione. OMNeT++ è un software che offre un editor di sviluppo basato su Eclipse [https://eclipse.org/]. Sono disponibili molte una gran varietà di strutture base come reti wired e wireless, ma è possibile aggiungere estensioni che permettono di ampliare la gamma di reti supportate. INET è una delle estensioni del framework più corpose e contiene una grossa quantità di reti e protocolli delle più diffuse strutture di rete utilizzate.

OMNeT++ è un simulatore basato solo su eventi, ma come spiegato nella Sezione 2.5, è possibile coprire anche situazioni di esecuzione di azioni periodiche, organizzandole come eventi programmati. OMNeT++ contiene anche un ambiente grafico, Tkenv, per la rappresentazione grafica della struttura della rete e degli eventuali link fisici tra i nodi; è in grado inoltre di eseguire un ri-ordinamento spaziale dei nodi della rete con l’obiettivo di rappresentare il grafico della rete col minor numero di intersezioni tra i collegamenti per rendere più leggibile lo schema ed eventuali animazioni date dalla simulazione.

OMNeT++ offre una architettura per i componenti che compongono la rete, detti moduli. I moduli sono anch’essi implementati col linguaggio C++ e successivamente assemblati in componenti più grandi, utilizzando un linguaggio a più alto livello, il linguaggio NED che sta per Network Definition. Grazie a questa sua struttura modulare, è possibile riutilizzare facilmente i moduli, sia semplici sia più complessi, già presenti nelle librerie di sistema oppure quelli creati dall’utente.

Con OMNeT++, come per PeerSim, è possibile specificare in appositi file di inizializzazione una lista di parametri necessari a configurare la rete durante la sua fase di inizializzazione prima di iniziare la simulazione vera e propria. In un singolo file di inizializzazione è possibile specificare più scenari di esecuzione e per ognuno specificare quindi valori diversi.

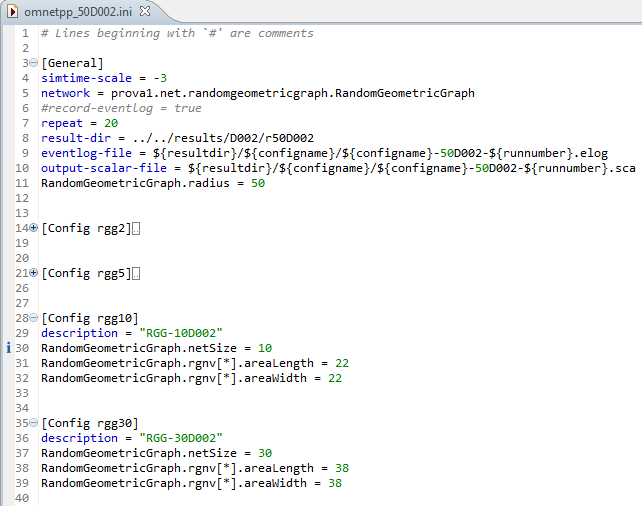
I parametri specificati possono essere letti dal file di Network Definition e usati per creare e impostare la rete secondo i parametri specificati dall’utente nel file di inizializzazione. Tramite il file di Network Definition si può propagare la lettura dei parametri fino ai file C++ che implementano i metodi che modellano il comportamento del componente.

OMNeT++ offre anche un sistema di raccolta dati, col quale si possono poi analizzare i dati raccolti, filtrarli e ottenere differenti rappresentazioni grafiche dei dati stessi. La raccolta dati può essere sia la sequenza temporale dei risultati, sia una raccolta delle statistiche su dati specifici oppure entrambe le cose.

### File di Inizializzazione

Un file di inizializzazione è un file necessario alla corretta configurazione della rete, dei suoi componenti e dell’ambiente di simulazione. Un file di inizializzazione ha estensione “.ini” e per comodità verrà anche chiamato “INI file”. In un file di inizializzazione è possibile fare una definizione dei parametri organizzata in una struttura ad albero, in cui ogni ramo è uno scenario di simulazione indipendente dagli altri ma eredita tutti i parametri coi rispettivi valori, se non ridefiniti, dello scenario padre da cui discente. Al momento della simulazione vera e propria, il sistema chiederà all’utente di scegliere quale scenario di simulazione utilizzare se non ve ne è uno specificato direttamente nello script di lancio.

Nella figura2.3 abbiamo un esempio di file d’inizializzazione. Vediamo come sotto la sezione “General” vengano specificati tutti quei parametri comuni a tutte le configurazioni e di seguito vengono specificate tutte le altre configurazioni coi parametri specifici per quelle simulazioni.

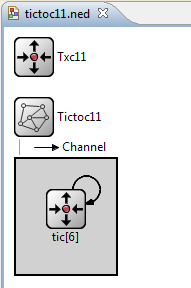


*Figura 2.3: Esempio file di inizializzazione.*

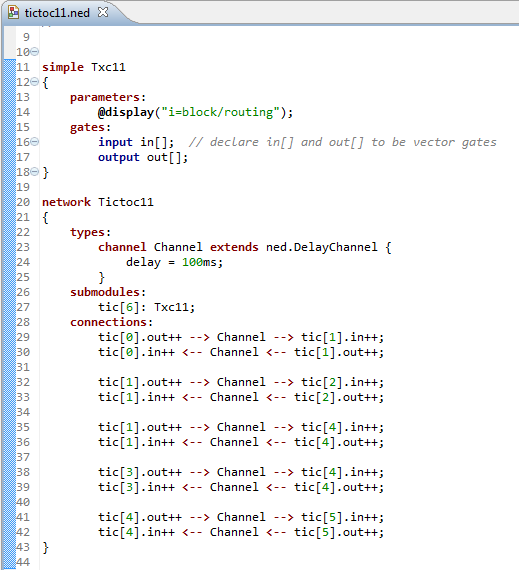
In generale ogni file di inizializzazione caratterizza un set di scenari da simulare, quindi si tende a mettere nella sezione “General” anche tutti quei parametri che servono a impostare correttamente la simulazione o la gestione del framework quali per esempio l’unità di misura con a quale viene scandito il tempo di simulazione, il numero di ripetizioni per ogni casistica che si desidera fare, le varie directory in cui si vogliono salvare i risultati o i dati raccolti e altri ancora.

### Linguaggio NED

OMNeT++ offre, tramite un linguaggio ad alto livello chiamato Network Definition la possibilità di definire il layout della rete e la struttura dei singoli componenti e della composizione dei componenti più grandi. Il linguaggio di Network Definition viene usato per creare file di specifica con estensione “.ned”, che abbrevieremo con NED file. IL linguaggio NED è stato pensato per poter implementare singoli moduli in maniera indipendente dall’ambiente, rendendo così facile il loro riutilizzo o estensione. Il sistema ci offre due visibilità del layout di rete ad alto livello: Design e Source. Con la vista Design ci fornisce una rappresentazione grafica della rete, se pur sempre ad alto livello, mentre la vista Source permette di vedere e implementare il file come puro codice sorgente. In figura 2.4 è riportato un esempio di NED file in vista Design, dove il sistema dice che nel file in questione sono stati usati componenti di tipo “Txc11”, i canali di comunicazioni utilizzati sono dei “Channel” e la rete si può riassumere come un vettore di nodi collegati tra di essi tramite diversi “Channel”. La vera rappresentazione del layout di rete viene fatto solo a runtime in fase di simulazione nel caso venga utilizzato lo strumento grafico Tkenv. In figura 2.5 invece è riportato lo stesso file della figura 2.4 ma in vista Source. Come si può vedere nella figura 2.5, in vista sorgente possiamo vedere tutto il codice che sta dietro alla vista Design. Si nota che nella prima parte viene definito il singolo componente “Txc11”, poi viene definita la rete come un vettore di componenti “Txc11” e tra essi vi sono dei collegamenti di tipo “Channel”, definito dentro la rete.



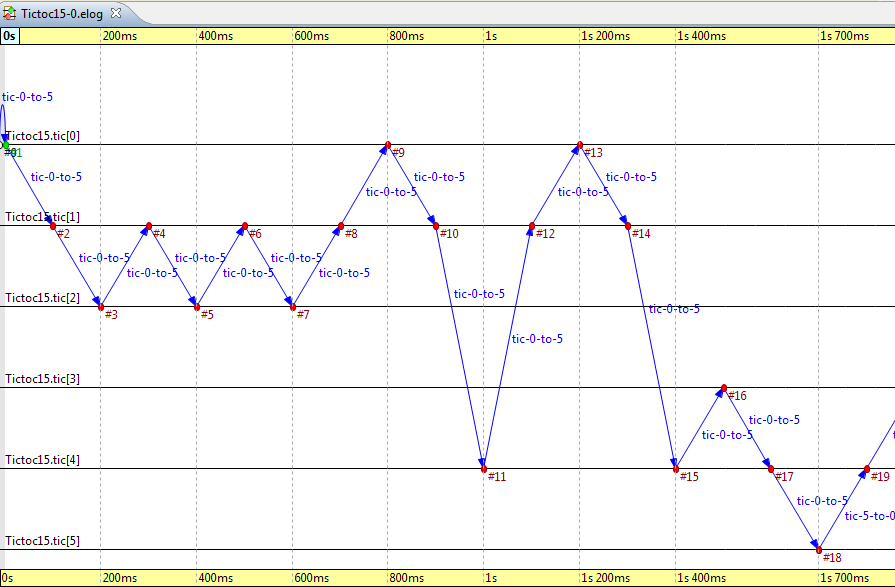
*Figura 2.4: Esempio di NED file, vista Design.*



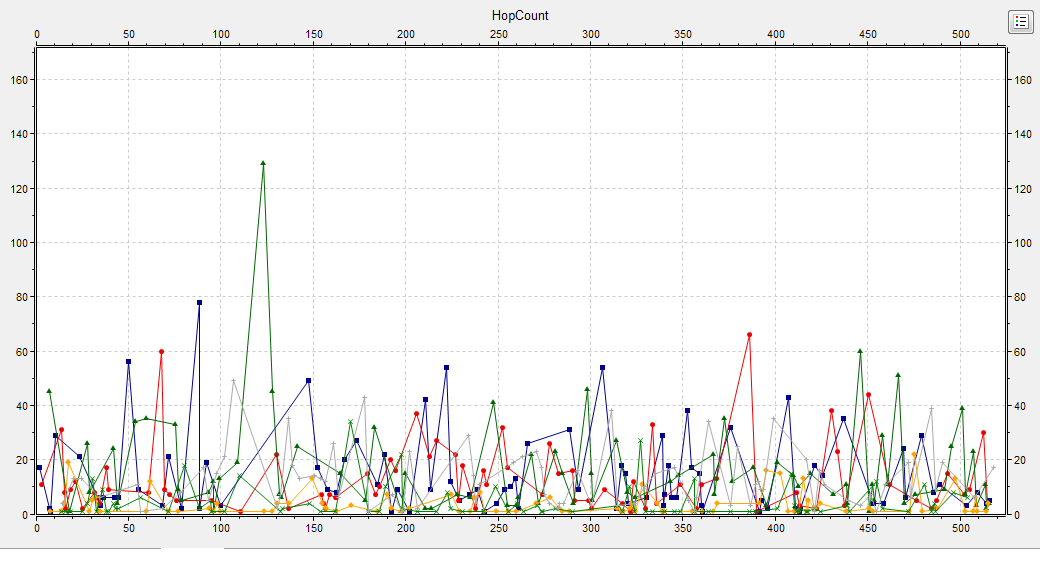
*Figura 2.5: Esemprio di NED file, vista Source.*

### Raccolta Dati

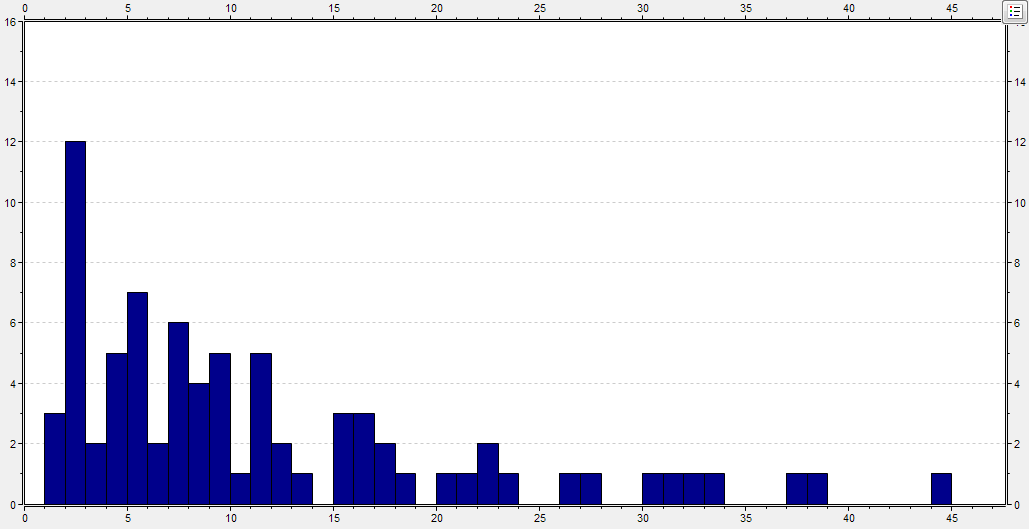
OMNeT++ fornisce metodi per la raccolta dati e di statistiche e la possibilità di analizzarli al termine della simulazione. Si possono raccogliere due categorie di dati: i log temporali, chiamati anche event-log e i dati relativi a statistiche. I log temporali non sono altro che rappresentazioni grafiche della successione degli eventi tra i nodi della rete. Molto utile per analizzare la sequenza temporale degli eventi e la loro durata, soprattutto quando si hanno parecchi eventi che avvengono in contemporanea; in figura 2.6 è riportato uno frammento di un esempio di event-log. Per quanto riguarda le statistiche, si possono raccogliere dati con riferimento temporale, chiamati vettori, oppure statistiche su dati, chiamati scalari, quali media, varianza, deviazione standard, somma, minimo, massimo ecc… . Sia sui vettori che sugli scalari è possibile poi fare delle operazioni di manipolazione dati, come raggruppamenti, applicare filtri, applicare operazioni ai dati o a gruppi di dati come l’operatore media; ad esempio si può applicare l’operatore media a un grafico temporale di vettori, per ottenere l’andamento temporale della media, utile per visualizzare il comportamento asintotico del sistema. In figura 2.7 è riportato un esempio di rappresentazione grafica di vettori temporali. In figura 2.8 invece è riportato un esempio di istogramma come rappresentazione della distribuzione del conteggio di hop necessari per raggiungere l’host destinatario.



*Figura 2.6: Esempio di event-log.*



*Figura 2.7: Esempio di grafico di vettori temporali.*



*Figura 2.8: Esemprio di grafico di statistiche.*