Configurazione a minima energia per reti di sensori wireless

Enrico Magon
magonenr@dei.unipd.it

Michele Nicetto
mnicetto@dei.unipd.it

Luca Parolini
lparolin@dei.unipd.it

Gruppo 10

Sommario

Una rete di sensori wireless (WSN - Wireless Sensor Network) in genere non dispone di riserve energetiche illimitate, ne segue che la longevità di funzionamento è limitata dalla ridotta capacità della sorgente di alimentazione (tipicamente batterie). E' quindi importante per queste reti l'uso di protocolli di comunicazione che abbiamo come scopo la minimizzazione del consumo delle riserve energetiche a disposizione dei nodi, senza influire sulla connettività e quindi sulla funzionalità della rete. In questo progetto sono stati analizzati i tre grandi filoni di ricerca che, separatamente, contribuiscono alla minimizzazione dei consumi energetici della rete, essi sono: Controllo della topologia, Power aware routing e Sleep management.

I tre approcci, mirando a minimizzare l'energia dissipata solamente in un particolare stato del nodo (trasmissione, ricezione e idle) non riescono a minimizzare il consumo energetico globale e quindi non raggiungono l'ottimo. Nel progetto si studia quindi il problema della minimizzazione globale dell'energia utilizzata dai nodi di una rete, dimostrando che questo problema appartiene alla classe dei problemi NP-completi. Nel progetto vengono quindi proposti degli algoritmi approssimanti la soluzione ottima ed è svolto un confronto fra le energie dissipate dalla rete quando si utilizzino gli algoritmi proposti od algoritmi di tipo greedy.

Il confronto degli algoritmi è stato realizzato su un simulatore di reti (Rmase) ideato da Zhang Ying (yzhang@parc.com) e basato su progetto precedente (Prowler) ideato da Gyula Simon (gyula.simon@vanderbilt.edu).

Indice

Figura 1: Esempi di nodi wireless, da sinistra: TUTWSN node, Tmote-Sky e Mica2

1 Introduzione e motivazioni del problema

Le Wireless Sensor Network sono reti in cui i sensori, detti anche nodi, sono collegati fra loro tramite link radio. Ogni nodo consiste di tre unità fondamentali: controllo, comunicazione e analisi dell'ambiente. L'idea che muove lo sviluppo industriale in questo settore è quella di ottenere degli strumenti di analisi ambientale molto robusti, di costo ridotto e facilmente collocabili anche in zone inaccessibili direttamente. Si pensi ad esempio al rilevamento delle condizioni climatiche di un vasto territorio: tale compito potrebbe essere svolto facilmente lasciando cadere i sensori da un elicottero sorvolante la zona, senza richiedere un posizionamento manuale da terra dei singoli nodi, con un notevole risparmio di tempo e costi. La robustezza della rete risiede nella numerosità dei nodi e nella loro capacità di modificare dinamicamente la topologia della rete variando la potenza di trasmissione. Ciò permette ad esempio di mantenere connessa la rete, accettando un dispendio energetico maggiore, anche in presenza di perdite di più del 50% dei nodi. Allo scopo sono in fase di studio algoritmi di localizzazione ed analisi dei dati che sfruttino pienamente la capacità di calcolo distribuito della rete. Se infatti la rete è gestita da un unico nodo centrale, la perdita di quell'unico nodo rende la rete non più operativa.

Generalmente i sensori montati a bordo dei nodi forniscono delle prestazioni ridotte. Questo è dovuto al fatto che si cerca di mantenere il più possibile basso il costo del singolo nodo, sfruttando l'idea secondo cui una volta raccolti i dati di interesse dai nodi della rete, l'indipendenza degli errori generati dai diversi sensori, rende la media delle misure un risultato di precisione elevata, anche maggiore di quanto si sarebbe potuto ottenere con la misura di un unico sensore di maggior precisione.

Esempi di alcuni nodi wireless attualmente utilizzati sono proposti in figura 1. Si osservi che il primo nodo a sinistra è delle dimensioni di 5 €cent.

I nodi di reti wireless sono generalmente alimentati da batterie, o comunque da generatori a bassa potenza ed è quindi necessario ottimizzare il consumo energetico dei nodi per massimizzare il tempo di vita dell'intera rete. Si pensi ad esempio che per alcune applicazioni il tempo di vita richiesto per i nodi è dell'ordine di qualche anno.

Un aspetto importante da tenere in considerazione nella ricerca del minor dispendio energetico da parte dei nodi è l'utilizzo di hardware a bassissimo consumo energetico. Dal punto di vista dell'hardware di controllo negli ultimi anni sono stati raggiunti dei risultati ragguardevoli, si vedano ad esempio i microcontrollori AVR-picoPower (www.atmel.com) che arrivano a consumare 1 [μW] lasciando un real time clock atti-

vo[1]. Ciò che invece è il grande punto debole, ma anche conferisce alla rete la sua grande flessibilità, è la comunicazione radio fra i nodi. I chip radio attualmente utilizzati richiedono infatti circa 28[mW] per una trasmissione a minima potenza, si vede dunque che la principale fonte di consumo energetico in una WSN è proprio la comunicazione radio fra i nodi e su di essa, si è svolta negli ultimi anni un'intensa attività di ricerca i cui risultati sono brevemente introdotti nella sezione 2. In tale sezione si evidenzia anche come tali linee di ricerca intervengano su un unico aspetto del consumo energetico dei nodi e che quindi, non possano raggiungere il consumo energetico minimo globale. Nella sezione 3.2 è perciò impostato in modo formale il problema di configurazione a potenza minima di una generica rete wireless, dimostrando che tale problema appartiene alla classe dei problemi NP-completi. In tale sezione, il nostro gruppo ha anche impostato il problema nel caso di canale soggetto ad errore nelle trasmissioni, parte non trattata negli articoli trovati in letteratura. Nelle successive sezioni sono introdotti alcuni approcci per la risoluzione del problema e, nella sezione 5, è presentato l'algoritmo risolutivo scelto dal nostro gruppo fra quelli proposti in letteratura, è indicato come esso sia stato implementato e quali siano le possibili migliorie apportabili. Nella sezione 6 sono illustrati i risultati delle simulazioni svolte su Rmase, mentre la sezione 7 conclude la relazione riassumendo quanto è stato prodotto durante il progetto.

In appendice sono stati inserite anche delle piccole introduzioni ai software Prowler ed Rmase, software che il gruppo ha studiato ed utilizzato per lo svolgimento del progetto.

2 Sviluppi di ricerca attuali

A differenza delle reti cablate, in una rete wireless ciascun nodo ha la possibilità di cambiare l'insieme dei nodi raggiungibili direttamente (one-hop neighborhood), semplicemente variando la potenza di trasmissione. Una conseguenza di tale fatto è che la topologia della rete dipende dalla potenza di trasmissione dei nodi e quindi si deve prestare attenzione a non causare, nel tentativo di ridurre l'energia dissipata dalla rete, la suddivisione della rete in cluster fra loro sconnessi. Un esempio di rete wireless in cui sono stati evidenziati i link fra i nodi è riportata in figura 2 nella pagina seguente.

Sono stati provati numerosi approcci per ottenere protocolli di comunicazione energeticamente efficienti ed in grado di evitare l'insorgenza dei problemi appena descritti. Tali approcci possono essere grossolanamente classificati in tre categorie: Controllo della topologia, Power aware routing e Sleep Management.

Di seguito sono descritte queste tre tipologie, complete di alcuni esempi che evidenziano pregi e difetti di ciascuna.

2.1 Sleep Management

Questo ramo di ricerca si occupa di ottimizzare gli istanti temporali ottimi di accensione e spegnimento dei nodi al fine di minimizzare il consumo energetico globale della rete senza condizionarne le capacità trasmissive. Si veda ad esempio [**?**]. Dato che i nodi possono operare in diversi stati, legati a diversi consumi energetici e che, per il

[1] Dati rilevati da [**?**].

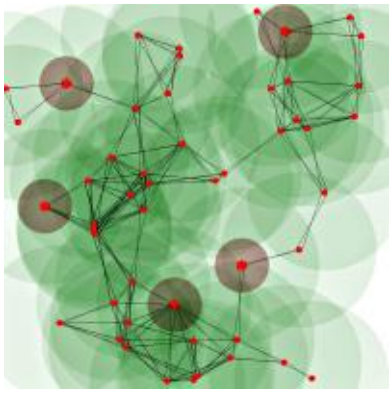


Figura 2: Esempio di rete wireless in cui sono evidenziate le connessioni fra i nodi.

State description	Power [mW]	Time [μs]
CPU on, Radio idle	6	
CPU idle, Radio off	0.18	
CPU standby, Radio off	0.017	6[μs]

Tabella 1: Stati, consumi di potenza e tempi di ripristino di un nodo Tmote-sky, dati rilevati da [**?**].

passaggio da uno stato ad un altro, vi è un certo dispendio energetico ed un intervallo temporale di transizione durante il quale il nodo non può compiere alcuna operazione, è necessario scegliere in modo intelligente quando ed in quale stato porre ogni singolo nodo della rete. Un elenco degli stati, dei consumi energetici e dei tempi di transizione relativo ai nodi Tmote-Sky è riportato in tabella 1.

La scelta del passaggio da uno stato ad alto consumo energetico ad uno inferiore non è banale in quanto il passaggio da e verso lo stato a consumo energetico inferiore richiede del tempo e soprattuto energia. Inoltre se il nodo si trovasse in uno stato a basso consumo energetico potrebbe non essere in grado di rispondere alle richieste della rete all'interno di un margine temporale prestabilito o addirittura, potrebbe essere completamente isolato dal resto della rete e quindi non in grado di rilevare i messaggi che transitano in essa (deep-sleep). E' chiaro dunque che per applicare questo tipo di tecnica è necessario conoscere la statistica di interarrivo dei pacchetti e, in base a questa, decidere ad ogni istante quale stato energetico sia ottimo per il nodo. E' possibile dunque determinare delle soglie temporali, una per ogni livello energetico, tali per cui se il tempo di interarrivo dei pacchetti è maggiore della soglia $k$esima, allora la transizione alla stato k risulta energicamente vantaggiosa.

Si supponga che i pacchetti giungano ad un nodo A ad intervalli temporali di almeno T_1[s] e sia P_k la potenza dissipata nello stato k da A e P_h la potenza dissipata nello stato h, con $P_k > P_h$. Detti $\tau_{h,k}$ il tempo necessario per passare dallo stato P_h allo stato P_k e $\tau_{k,h}$ il tempo necessario per la transizione inversa, si vuole determinare una soglia temporale $T_{k,h}$ tale per cui, se $T_1 > T_{k,h}$, allora per il nodo è conveniente eseguire la transizione dallo stato k allora stato h, se invece, $T_1 \leq T_{k,h}$ allora la transizione implicherebbe una perdita di energia. E' chiaro che per il nodo converrà porsi nello stato h a minima energia tale per cui $T_{k,h-1} \leq T_1 \leq T_{k,h}$.

Per calcolare quale sia la soglia temporale $T_{k,h}$ si segua il seguente ragionamento: l'energia conservata dalla transizione è data dalla differenza fra l'energia dissipata nei due stati nel tempo T_1 e l'energia richiesta per eseguire le transizioni fra i due stati ottenendo:

$$\Delta_E = E_k - E_{k,h} = (P_k - P_h) \cdot T_1 - \frac{P_k - P_h}{2}\tau_{h,k} - \frac{P_k - p_h}{2}\tau_{k,h} \tag{1}$$

Il nodo effettuerà dunque una transizione solo se $\Delta_E > 0$ e quindi:

$$T_1 > T_{Th_{h,k}} = \frac{1}{2}\left[\tau_{k,h} + \frac{P_k + P_h}{P_k - P_h}\tau_{h,k}\right] \tag{2}$$

Si osservi che questa tecnica ritorna dei buoni risultati solo per quanto riguarda il traffico di rete periodico o comunque con un tempo di interarrivo a bassa varianza. Quando invece nella rete transitano anche pacchetti aperiodici o che richiedono una latenza minima (ad esempio contenenti messaggi d'allarme) i nodi sono costretti ad accendersi frequentemente per verificare che non vi siano tali pacchetti all'interno della rete, aumentando così notevolmente il consumo energetico.

La variazione del consumo energetico nei vari stati di un nodo è generalmente attuata spegnendo le periferiche del nodo come ADC, timer o il chip radio, anche se in quest'ultimo caso si causa la disconnessione del nodo dalla rete. Spesso è inoltre possibile anche ridurre la frequenza di clock e la tensione di alimentazione della CPU ottenendo, nel primo caso, una riduzione lineare dell'energia consumata e nel secondo caso, una diminuzione di tipo quadratico. Si osservi comunque che la scelta di transizione dello stato del chip suppone l'uso di sistemi operativi sufficientemente complessi all'interno dei nodi. In realtà tali sistemi, a causa delle limitazioni sulla memoria a bordo del chip, spesso si riducono ad un semplice insieme di routine che raramente implementano algoritmi evoluti di gestione dei consumi energetici. Recentemente comunque, si sono prodotti nodi a ridotto consumo energetico ma con una elevata quantità di memoria a bordo e, all'interno di questi nuovi nodi, possono essere utilizzati sistemi operativi anche complessi.

Questo tipo di tecniche si applicano sostanzialmente ai livelli più bassi della rete e possono quindi essere convenientemente poste tra il livello uno e due del modello OSI.

2.2 Controllo della topologia

Gli algoritmi di tipo Topology Control intervengono nella minimizzazione del consumo energetico nella fase di comunicazione fra i nodi. In particolare, scopo di tali algoritmi, è determinare la potenza di comunicazione ad hoc per ogni nodo al fine di ottenere una topologia di rete con proprietà prestabilite.

Si consideri ad esempio il caso di una rete con una topologia molto fitta, allora è possibile che la comunicazione fra una qualunque coppia di nodi, se fatta in modo poco attento, impedisca una contemporanea comunicazione fra altri nodi della rete, riducendo così il throughput della rete. Al contrario, in una rete sparsa lo spegnimento di un nodo può causare un partizionamento della rete se i nodi non adeguano la potenza di trasmissione o se preventivamente non si erano creati dei cammini ridondanti.

L'ultimo esempio introduce quindi il concetto di k-connettività (secondo i vertici) del grafo di una rete. Per definire correttamente la nozione di k-connettività sono necessarie le seguenti definizioni[2]:

Definizione 2.1 (Grafo connesso) *Un grafo $G = (V, E)$ si definisce connesso se per ogni coppia di nodi $u, v \in V$ esiste un cammino da u a v.*

Definizione 2.2 (Insieme di Taglio) *Si definisce insieme di taglio di un grafo connesso $G = (V, E)$, un sottoinsieme C di V la cui rimozione rende il grafo non connesso.*

Definizione 2.3 (Connettività) *Si definisce connettività secondo i vertici (per brevità connettività) di un grafo $G = (V, E)$ la norma del più piccolo insieme di taglio di G.*

Definizione 2.4 (k-connettività) *Un grafo $G = (V, E)$ è detto k-connesso se la sua connettività è almeno k.*

Lemma 2.1 *Se un grafo $G = (V, E)$ è k-connesso, allora è possibile rimuovere fino a $k - 1$ nodi senza influire sulla connettività del grafo.*

Dim.
Sia dato un grafo $G = (V, E)$ k-connesso e supponiamo di eliminare un nodo v_1 di G. In forza della k-connettività di G, il nuovo grafo G' è ancora connesso. Si deve ora dimostrare che G' è $k - 1$ connesso. Si supponga dunque che G' sia h-connesso con $h < k - 1$ e dimostriamo che questo comporta un assurdo.

Se G' è h connesso allora esiste una combinazione di h nodi la cui rimozione comporta la perdita di connettività di G', indichiamo con G'' il nuovo grafo così ottenuto. Si noti che G'' è stato ottenuto a partire dal grafo G k-connesso, con la rimozione di $h+1$ nodi. Il grafo G per ipotesi è $k > h + 1$ connesso e quindi la rimozione di qualunque $h + 1$ nodi non modifica la connettività del grafo e quindi, G'' è un grafo connesso e quindi, G' è $k - 1$-connesso. ■

In base al lemma ora esposto, si nota che la proprietà di k-connettività è una proprietà notevole per un grafo descrivente la topologia di una rete di sensori dato che è garantito che la perdita di $k - 1$ nodi consente ancora le comunicazioni multi-hop tra due qualsiasi dei nodi rimanenti. Gli algoritmi in questo filone di ricerca si concentrano quindi per il più a determinare le potenze di trasmissione in modo da ottenere reti con proprietà di k-connettività dove k è un valore prestabilito.

Un articolo che rientra in questa linea di ricerca è [**?**], nel quale sono indicati vari algoritmi per minimizzare il consumo energetico pur mantenendo le proprietà di k-connettività della rete. In particolare lo scopo degli algoritmi proposti nell'articolo, non è la minimizzazione globale delle potenze di comunicazione, ma la minimizzazione del massimo delle potenze di comunicazione fra i diversi nodi.

Dato che la riserva di energia di un nodo è un caratteristica intrinsecamente locale, una minimizzazione globale dell'energia potrebbe generare un utilizzo irregolare e

[2]Si suppongono note le nozioni basilari sui grafi. Per un'introduzione all'argomento si veda ad esempio [**?**]

disomogeneo dei nodi della rete, inducendo un rapido spegnimento di alcuni in favore di uno scarso utilizzo degli altri. Scegliendo invece di minimizzare il massimo delle energie, è assicurato che ogni nodo al più utilizzerà una potenza P_{max} per comunicare con gli altri nodi.

Per quanto esposto, gli algoritmi di tipo topology control possono essere posti fra il livello due e tre di una rete, in quanto modificano la potenza di comunicazione fra le singole coppie di nodi (livello Data Link) per creare una predeterminata topologia di rete (livello di Rete).

2.3 Power aware routing

In questo caso lo studio è volto all'individuazione di un percorso di instradamento (routing path) a costo energetico minimo. Tale risultato viene ottenuto tramite algoritmi di routing che sfruttano metriche basate sulla potenza di trasmissione fra i nodi, anziché basate sul numero di hop. Si osservi che la sostanziale differenza con la precedente classe di algoritmi risiede nel fatto che in questo caso non si modifica la potenza di trasmissione dei singoli nodi e quindi, indirettamente, la topologia della rete, bensì, data una topologia di rete, si determinano dei cammini per i pacchetti che minimizzino l'indice di costo desiderato. Si può dunque dire che questa classe di algoritmi appartiene ad un livello OSI più alto di tutte e tre le precedenti: il livello di rete, terzo livello nel modello OSI. Algoritmi che utilizzano questa strategia sono descritti ad esempio in [**?**] oppure [**?**].

Di seguito sono riportati alcuni esempi di indici di costo per gli algoritmi di routing, ciascuno dei quali comporta determinati vantaggi e svantaggi.

2.3.1 Minimizzazione dell'energia di trasmissione globale

Un primo esempio di indice di costo è l'energia spesa per l'invio di tutti i pacchetti della rete. Indicato con $E(a, b)$ l'energia spesa per l'invio di un pacchetto da un nodo a ad uno b, con N_{pck} il numero di pacchetti che la rete deve inviare e con $n_{1,j}, \ldots, n_{k,j}$ i nodi interessati per la trasmissione del pacchetto j-esimo si ha:

$$\min \sum_{j=1}^{N_{pck}} \sum_{h=1}^{k_j-1} E(n_{h,j}, n_{h+1,j}) \tag{3}$$

Si noti che in questo caso il cammino determinato dall'equazione (3) può creare delle asimmetrie nel consumo energetico, ottenendo così nodi che dissipano molta energia e nodi che restano per lo più inutilizzati e dissipano quindi una energia irrisoria. In tal caso potrebbe dunque verificarsi lo spegnimento dei nodi maggiormente utilizzati in un tempo molto breve, causando una modifica nella topologia della rete che può anche perdere le sue caratteristiche di connettività.

Per risolvere tale problema, come indicato nel paragrafo 2.2, si potrebbe scegliere la minimizzazione del seguente indice:

$$\min \max_{j\in[1,N_{pck}]} \max_{h\in[1,k_j-1]} E(n_{h,j}, n_{h+1,j}) \tag{4}$$

In questo caso si otterrebbe la minimizzazione del consumo energetico fra tutte le potenze di trasmissione utilizzate nei diversi hop.

2.3.2 Massimizzazione del tempo di vita dei nodi

Data una coppia di nodi sorgente e ricevente (s, t) l'idea è in questo caso di determinare un insieme minimo di nodi $\mathcal{N}_{s,t}$ la cui rimozione rende la rete sconnessa, lasciando ricevitore e sorgente in due sottografi distinti. E' chiaro dunque che una qualunque comunicazione fra il nodo s ed il nodo t, dovrà necessariamente passare per un elemento di $\mathcal{N}_{s,t}$. Al fine di massimizzare il tempo di vita della rete è dunque bene dividere in modo ottimo l'energia consumata dai nodi in $\mathcal{N}_{s,t}$. Gli algoritmi che utilizzando questo tipo di indici però, risultano essere molto complessi e sono raramente utilizzati in quanto la ricerca dell'insieme $\mathcal{N}_{s,t}$ è un problema NP-completo, [**?**].

3 Il problema di configurazione a minima potenza

Nessuno dei tre approcci esposti nella precedente sezione è in grado di minimizzare il consumo energetico richiesto globalmente nei diversi stati dei nodi: trasmissione, ricezione, attesa (idle). Il controllo di topologia ed il power aware routing infatti, intervengono durante le fasi di trasmissione e ricezione ma non considerano la potenza dissipata nello stato di idle. Al contrario lo sleep management può ridurre la potenza dissipata dai nodi in stato di attesa portandoli al livello di consumo energetico minimo (deep sleep), ma non ottimizza la potenza utilizzata nella trasmissione radio. E' chiaro quindi che per il raggiungimento del minimo consumo energetico di una WSN, detto anche problema di configurazione a potenza minima (**MPC**), è necessario implementare tutte e tre queste strategie che, si noti, intervengono su diversi layer del modello OSI.

E' meno palese invece, che la scelta del cammino ottimo, e quindi della potenza di trasmissione, dipenda anche dal rate trasmissivo dei dati. E' possibile mostrare questa dipendenza con un esempio tratto da [**?**].

3.1 Dipendenza del cammino ottimo dal rate trasmissivo dei dati

Sia data una WSN descritta dal grafo di figura 3 e si supponga che il nodo a debba comunicare con il nodo c.

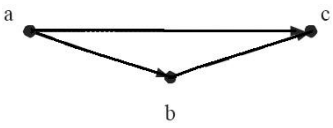


Figura 3: Grafo $G = (V, E)$, $V = \{a, b, c\}$, $E = \{(a, b), (b, c)(a, c)\}$.

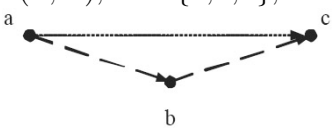


Figura 4: Due possibili soluzioni per una comunicazione da a a b.

In generale sono possibili due soluzioni:

- una comunicazione diretta nodo-nodo;

- una comunicazione multi-hop.

Assunto che il nodo a generi D[bit] dati ogni T[s], ad un rate quindi di $R = \frac{D}{T}$[bps], si vuole studiare come vari l'energia dissipata dalla rete per la trasmissione dei dati utilizzando la prima o la seconda soluzione.

Si studi inizialmente il caso della trasmissione diretta dal nodo a al nodo c, situazione che permette di lasciare il nodo b in stato di riposo (sleep).

Sia $B \geq R$ la banda trasmissiva dei nodi della rete. L'energia dissipata dalla rete in T è data dalla somma dell'energia[3] dissipata dal nodo a per trasmettere i dati al nodo c:

$$P_{tx}(a,c) \cdot T_{tx}$$

dall'energia dissipata dal nodo c per ricevere i dati:

$$P_{rx} \cdot T_{rx}$$

ed, infine, dall'energia dissipata dai nodi a e c durante il rimanente periodo in attesa di una nuova trasmissione dei dati:

$$P_{id}(1 - T_{tx}) + P_{id}(1 - T_{rx})$$

Si noti che utilizzando un algoritmo di sleep management, discusso nel paragrafo 2.1, i nodi potrebbero anche essere posti in uno stato di consumo energetico inferiore. Il nodo b può essere convenientemente posto nello stato di minimo consumo energetico (sleep) e quindi:

$$P_s \cdot T$$

L'energia complessivamente richiesta dalla rete è data quindi da:

$$\begin{aligned} E = P_1 \cdot T &= P_{tx}(a,c) \cdot T_{tx} + P_{rx} \cdot T_{rx} + P_{id}(1 - T_{tx}) + P_{id}(1 - T_{rx}) + P_s \cdot T \\ &= \frac{D}{B} P_{tx}(a,c) + \frac{D}{B} P_{rx} + P_{id}(1 - \frac{D}{B}) + P_{id}(1 - \frac{D}{B}) + P_s \frac{D}{R} \end{aligned} \tag{5}$$

dove nell'ultimo passaggio si è utilizzata la seguente eguaglianza: $T_{tx} = T_{rx} = \frac{D}{B}$

Risulta comodo per le successive analisi osservare la potenza media, P_1 richiesta dalla rete durante il periodo T:

$$\begin{aligned} P_1 &= \frac{D}{B}\frac{R}{D} P_{tx}(a,c) + \frac{D}{B}\frac{R}{D} P_{rx} + P_{id}(1 - \frac{D}{B}\frac{R}{D}) + P_{id}(1 - \frac{D}{B}\frac{R}{D}) + P_s \frac{D}{R}\frac{R}{D} \\ &= \frac{R}{B} P_{tx}(a,c) + \frac{R}{B} P_{rx} + 2P_{id}(1 - \frac{R}{B}) + P_s \end{aligned} \tag{6}$$

E' chiaro che minimizzare l'energia dissipata o la potenza media richiesta dai nodi della rete è equivalente e quindi, nel seguito ci si concentrerà sulle potenze richieste per le comunicazioni attraverso la rete.

[3] Si osservi che nelle formule si trascura l'energia richiesta dai nodi per eseguire i processi computazionali di generazione ed elaborazione dei dati in quanto si ritiene che tale energia sia irrilevante rispetto a quella dissipata per la ricezione e la trasmissione dei dati.

Nel caso invece si utilizzasse una comunicazione multi-hop coinvolgente il nodo b, al posto del termine $Ps \cdot T$, si dovrebbe inserire:

$$P(b,c)T_{tx} + P_{rx}T_{rx} + P_{id}(T - T_{rx} - T_{tx})$$

ed inoltre sostituire alla potenza $P(a,c)$ la potenza di trasmissione dal nodo a al nodo b, $P(a,b)$, ottenendo dopo pochi passaggi:

$$P_2 = \frac{R}{B}P_{tx}(a,b) + P_{tx}(b,c) + 2\frac{R}{B}P_{rx} + \left(3 - 4\frac{R}{B}\right)P_{id} \tag{7}$$

Appare ora chiaro che la potenza media richiesta dalla rete durante una trasmissione di dati dipende dal rate trasmissivo e che, quando il valore di potenza P_1 è inferiore a P_2 è bene utilizzare una comunicazione diretta punto punto, nel caso invece di rate trasmissivi superiori, è consigliabile una comunicazione di tipo multi-hop. Il valore di soglia R_0 del rate trasmissivo che determina quale sia la scelta ottima è dato dalla seguente formula:

$$R_0 = B\frac{P_{id} - P_s}{P_{tx}(a,c) - P_{tx}(b,c) - P_{tx}(a,b) + 2P_{id} - P_{rx}} \tag{8}$$

Si riportano di seguito due esempi di potenze medie, il primo nel caso di nodi di tipo Mica2 il secondo, invece, suppone l'uso di nodi Tmote-Sky.

es.

Nodi Mica2 Nel caso dei nodi di tipo Mica2, si supponga che la comunicazione fra i nodi a e c avvenga ad una potenza di 10[dBm] e che la comunicazione fra a e b e fra b e c avvenga a -12[dBm]. Le potenze di trasmissione, ricezione e sleep assumono i seguenti valori[4]:

- $P_{tx}(a,c) = 80.14$[mW];
- $P_{tx}(a,b) = P_{tx}(b,c) = 24.6$[mW];
- $P_{rx} = 24$[mW];
- $P_s = 6[\mu$W];
- $P_{id} = 24$[mW];
- $B = 38.4$ [Kbps].

Da cui:

$$R_0 = 38.4 \cdot \frac{24}{54.94} = 16.77[\text{Kbps}] \tag{9}$$

Il grafico di figura 5 nella pagina successiva mostra come varino le potenze $P1$ e $P2$ al variare del rate trasmissivo R.

Nodi Tmote-Sky

Nel caso dei nodi Tmote-sky, si assuma una trasmissione fra i nodi a e c a massima potenza: 0[dBm] e una comunicazione fra i nodi a e b, e b e c, a potenza minima: -25[dBm], in questo caso si otterrebbe[5]:

[4] Dati rilevati da [?] e da [?].
[5] Dati rilevati da [?] e [?].

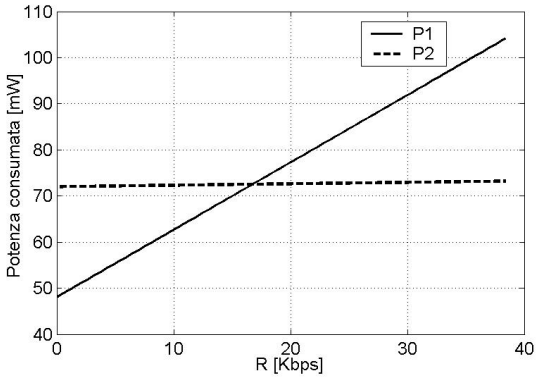


Figura 5: Consumo di potenza globale rispetto al data rate. Nodo Mica2

- $P_{tx}(a,c) = 57.42[\text{mW}]$;
- $P_{tx}(a,b) = P_{tx}(b,c) = 28[\text{mW}]$;
- $P_{rx} = 76[\text{mW}]$;
- $P_s = 17[\mu\text{W}]$;
- $P_{id} = 6[\text{mW}]$;
- $B = 250$ [Kbps].

Da cui:

$$R_0 = 250 \cdot \frac{6}{-62.58} = -24[Kbps] \tag{10}$$

Come è osservabile dal grafico di figura 6, in questo tipo di nodi la potenza richiesta durante la fase di ricezione è talmente elevata che, quando possibile, la comunicazione diretta fra i nodi è la comunicazione a minor consumo energetico.

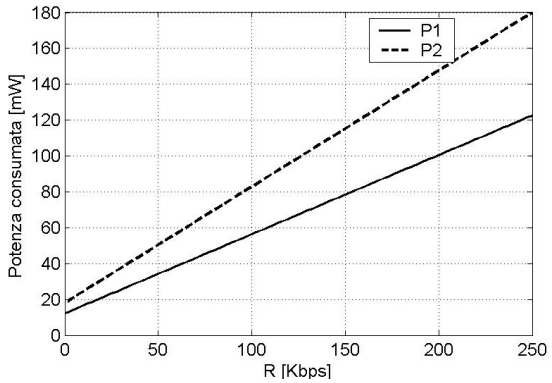


Figura 6: Consumo di potenza globale rispetto al data rate. Nodo Tmote-Sky

3.2 Formalizzazione del problema

Sia data una rete WSN[6] e si supponga che i nodi possano lavorare nei seguenti stati: *ricezione, trasmissione, idle* e *riposo*. I primi tre stati corrispondono ad un'impostazione di nodo attivo nei quali il nodo è in grado di ricevere dati dalla rete. Il quarto stato è lo stato a minor consumo energetico, in tale stato il nodo è incapace di ricevere dati dalla rete e si può supporre che il consumo energetico in tale stato sia nullo[7].

Dati due nodi u e v della rete si indica con $P_{tx}(u,v)$ la *minima* potenza di trasmissione necessaria per una corretta trasmissione di dati dal nodo u al nodo v. Si ricordi che in generale, a causa di link asimmetrici, $P_{tx}(u,v) \neq P_{tx}(v,u)$. Si indica inoltre con P_{MAX} la massima potenza di trasmissione dei nodi[8].

La rete di sensori è descritta dal grafo $G = (V, E)$ nel quale ad ogni nodo della rete è associato un vertice v ed esiste un arco diretto dal vertice u al vertice v se e solo se $P_{tx}(u,v) \leq P_{MAX}$. Data la corrispondenza biunivoca fra i nodi della rete ed i vertici del grafo, nel seguito si useranno indifferentemente i termini nodo e vertice risultando chiaro dal contesto a cosa ci si riferisca.

Si assume che esistano dei nodi sorgenti s_i, appartenenti a $S \subseteq V$, che trasmettono dati ad alcuni nodi riceventi t_j, appartenenti a $T \subseteq V \setminus S$, con un rate $R_{i,j}$ noto al nodo sorgente[9]. Si assume che la banda trasmissiva B sia identica per ogni nodo e si indica con $r_{i,j}$ il rapporto fra $R_{i,j}$ e la banda trasmissiva B. Si definisce infine insieme di richieste di comunicazione l'insieme $\mathcal{I} = \{(s_i, t_j, r_{i,j}) : s_i \in S, t_j \in T\}$.

Si assume che le richieste di comunicazione fra ogni coppia di nodi sorgente e ricevente siano soddisfacibili dai nodi della rete indipendentemente dal cammino intrapreso dai flussi di dati. La formalizzazione esatta di quest'ipotesi è molto complessa e può essere convenientemente semplificata richiedendo che la somma di tutti i rate trasmissivi normalizzati sia minore di $\frac{1}{2}$:

$$\sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} r_{i,j} \leq \frac{1}{2} \tag{11}$$

Potrebbe inizialmente apparire sufficiente richiedere che tale sommatoria sia minore dell'unità. Dato però che i nodi intermedi di hop devono prima ricevere un pacchetto e poi ritrasmetterlo, assumendo anche un tempo nullo per l'elaborazione del pacchetto, è chiaro che il canale rilevato dalla sorgente ha banda dimezzata.

E' utile ora introdurre le seguenti definizioni. Dato un grafo $G = (V, E)$ ed un insieme di richieste di comunicazione $\mathcal{I}$

Definizione 3.1 *Si indica con $\Gamma^-(u)$ l'insieme dei nodi v tali per cui esiste un arco diretto dal nodo v al nodo u.*

[6] L'impostazione che verrà data al problema è presa da [**?**].

[7] Questo è verificato anche coi nodi reali. Il consumo energetico degli attuali nodi posti in stato di riposo è infatti paragonabile all'energia dispersa dalle batterie di alimentazione a causa dell'invecchiamento.

[8] In generale si potrebbe anche non richiedere che P_{MAX} sia un valore identico per ogni nodo, dato però che questa estensione è banale rispetto a quanto verrà detto e che, una tale assunzione complicherebbe solamente la notazione, si preferisce per ora trascurare questa possibilità.

[9] Nei casi reali è possibile ottenere una valida stima di questo valore anche con degli algoritmi di tipo greedy.

Definizione 3.2 *Si indica con* $\Gamma^+(u)$ *l'insieme dei nodi* v *tali per cui esiste un arco diretto dal nodo* u *al nodo* v*.*

Definizione 3.3 *Si indica con* $\Gamma(u)$ *l'unione di* $\Gamma^+(u)$ *e* $\Gamma^-(u)$*.*

Definizione 3.4 *Si indica con* $f(s_i, t_j)$ *l'insieme dei nodi utilizzati per la comunicazione fra un nodo sorgente* s_i *ed uno ricevente* t_j *(*s_i *e* t_j *compresi) tali per cui esiste un* $r_{i,j}$ *tale che* $(s_i, t_j, r_{i,j}) \in \mathcal{I}$*.*

Dato che in generale un flusso dati può essere suddiviso in più sottoflussi instradati lungo percorsi indipendenti, $f(s_i, t_j)$ potrebbe essere l'unione di più cammini fra i nodi s_i e t_j. Risulta quindi utile dare le seguenti definizioni:

Definizione 3.5 *Si indica con* $R_{i,j}(u)$ *la parte di rate trasmissivo, relativo alla comunicazione fra il nodo* s_i *ed il nodo* t_j*, che attraversa il nodo* u*.*

Definizione 3.6 *Si indica con* $r_{i,j}(u)$ *la parte di rate trasmissivo normalizzato relativo alla comunicazione fra il nodo* s_i *ed il nodo* t_j *che interessa il nodo* u*.*

Definizione 3.7 *Si indica con* $r_{i,j}(u, v)$ *la parte di rate* $r_{i,j}(u)$ *che il nodo* u *invia al nodo* v*.*

Definizione 3.8 *Si indica con* $\Gamma^-_{f(s_i,t_j)}(u)$ *l'insieme* $\Gamma^-(u) \cap f(s_i, r_j)$*.*

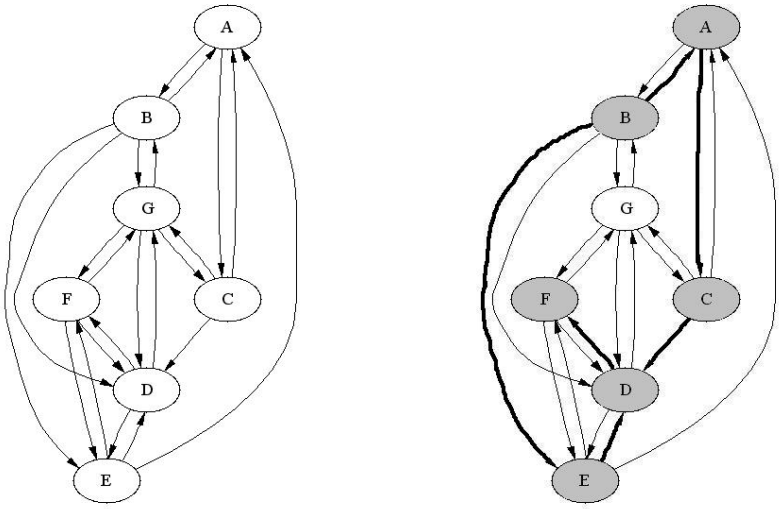
Definizione 3.9 *Si indica con* $\Gamma^+_{f(s_i,t_j)}(u)$ *l'insieme* $\Gamma^-(u) \cap f(s_i, r_j)$*.*

$\Gamma^-_{f(s_i,t_j)}(u)$ e $\Gamma^+_{f(s_i,t_j)}(u)$ rappresentano rispettivamente l'insieme dei nodi che trasmettono dati a u e dei nodi che ricevono dati da u relativamente al solo flusso di dati da s_i a t_j.

Per chiarire le definizioni date è possibile osservare il grafo di figura 7 nella pagina successiva. In esso si ha $\Gamma^-(D) = \{B, C, E, F, G\}$ e $\Gamma^+(F) = \{E, F, G\}$ ed inoltre $f(B, F) = \{A, C, D, E, B, F\}$, $\Gamma^+_{f(B,F)}(c) = \{D\}$ e $\Gamma^-_{f(B,F)}(C) = \{A\}$. In questo caso appare evidente che $f(B, F)$ sia l'unione di due cammini, il primo dato da $\{B, A, C, D, F\}$ ed il secondo dato da $\{B, E, D, F\}$. Nel grafo sia ha inoltre $r_{B,F}(A) = r_{B,F}(C)$ e, $r_{B,F}(A) + r_{B,F}(E) = \frac{r_{B,F}(B)}{2} = \frac{r_{B,F}(D)}{2} = \frac{r_{B,F}(F)}{2}$.

Per analizzare il problema è ora necessario studiare quale sia il consumo di potenza di un generico nodo u durante una comunicazione fra un nodo sorgente s_i ed un nodo ricevente t_j. Si noti che la potenza richiesta da un nodo di hop e quella richiesta da un nodo sorgente o da uno ricevente ha una diversa formulazione. Nel primo caso infatti il nodo deve prima ricevere i dati e poi ritrasmetterli, dimezzando quindi il periodo di idle rispetto ad un nodo sorgente o ricevente che esegue una sola di queste operazioni.

Allo scopo di semplificare il calcolo della potenza richiesta dai nodi della rete, si aggiungono quindi due nodi virtuali, uno ricevente t^* ed uno sorgente s^* che ricevono e inviano dati da e verso tutti i nodi sorgenti s_i e riceventi t_j ad un rate di $r_{i,j}$. In questo modo anche i nodi sorgenti e riceventi divengono nodi di hop, semplificando notevolmente le formule. E' da notare che l'aggiunta alla rete dei due nodi virtuali non influenza la scelta della configurazione ottima a potenza minima dato che le potenza richieste dai nodi s^* e t^* sono indipendenti dalla configurazione scelta e possono essere escluse dai successivi ragionamenti.



(a)

(b) Nodi utilizzati per la comunicazione fra i nodi B e F.

Figura 7: Esempio di grafo con link asimmetrici.

3.2.1 Potenza richiesta da un generico nodo $u \in V$

Per quanto visto in precedenza, durante una comunicazione fra i soli nodi s_i e t_j la potenza media richiesta da un generico nodo u è data dalla combinazione lineare convessa della somma della potenza richiesta negli istanti di comunicazione, ricezione e idle:

$$\begin{aligned} P(u) \quad = \quad & \sum_{v \in \Gamma^{+}_{f(s_i,r_j)}(u)} r_{i,j}(u,v) \cdot P_{tx}(u,v) + \sum_{v \in \Gamma^{-}_{f(s_i,r_j)}(u)} r_{i,j}(v,u) \cdot P_{rx} + \\ & + P_{id} \left(1 - \sum_{v \in \Gamma^{+}_{f(s_i,r_j)}(u)} r_{i,j}(u,v) - \sum_{v \in \Gamma^{-}_{f(s_i,r_j)}(u)} r_{i,j}(v,u) \right) \end{aligned} \tag{12}$$

Estendendo ora quanto espresso nell'equazione (12) al caso in cui il nodo u sia

interessato da più flussi di dati, si ottiene:

$$\begin{aligned} P(u) &= \sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} \left(\sum_{v\in\Gamma^+_{f(s_i,r_j)}(u)} r_{i,j}(u,v)\cdot P_{tx}(u,v) + \sum_{v\in\Gamma^-_{f(s_i,r_j)}(u)} r_{i,j}(v,u)\cdot P_{rx} \right) + \\ &\quad + P_{id}\left[1 - \sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} \left(\sum_{v\in\Gamma^+_{f(s_i,r_j)}(u)} r_{i,j}(u,v) + \sum_{v\in\Gamma^-_{f(s_i,r_j)}(u)} r_{i,j}(v,u) \right) \right] \\ &= P_{id} + \sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} \left[\sum_{v\in\Gamma^+_{f(s_i,r_j)}(u)} r_{i,j}(u,v)\cdot(P_{tx}(u,v) - P_{id}) + \right. \\ &\quad \left. + \sum_{v\in\Gamma^-_{f(s_i,r_j)}(u)} r_{i,j}(v,u)\cdot(P_{rx} - P_{id}) \right] \end{aligned} \tag{13}$$

3.2.2 Il problema MPC

Grazie all'analisi della potenza richiesta da un nodo u interessato dalle comunicazioni fra più nodi sorgenti e riceventi, è possibile definire il problema di configurazione a potenza minima nel modo seguente:

Definizione 3.10 (MPC) *Dato un grafo $G = (V, E)$ che descrive la topologia di una WSN e un insieme $\mathcal{I}$ di richieste di comunicazione, trovare un sottografo $G' = (V', E')$ di G ed un cammino $f(s_i, r_j)$ per ogni elemento di $\mathcal{I}$, tale per cui la potenza totale consumata $P(G')$ è minima:*

$$P(G') = \min \sum_{u\in V'} P(u) \tag{14}$$

con $P(u)$ dato da:

$$\begin{aligned} P(u) &= P_{id} + \sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} \left[\sum_{v\in\Gamma^+_{f(s_i,r_j)}(u)} r_{i,j}(u,v)\cdot(P_{tx}(u,v) - P_{id}) + \right. \\ &\quad \left. + \sum_{v\in\Gamma^-_{f(s_i,r_j)}(u)} r_{i,j}(v,u)\cdot(P_{rx} - P_{id}) \right] \end{aligned} \tag{15}$$

Il problema ora esposto è estremamente complesso in quanto è consentito che uno stesso flusso di dati possa essere ripartito lungo più cammini. Per semplificare la notazione e ottenere dei validi risultati ci concentreremo ora sul caso in cui un flusso di dati generato da un nodo sorgente può seguire un unico percorso.

3.2.3 Il problema MPC2

Si supponga dunque che il flusso dati dal nodo sorgente al ricevente non possa essere suddiviso lungo più cammini, allora l'equazione (12) può essere riscritta come:

$$P(u) = \sum_{v \in \Gamma^{+}_{f(s_i,r_j)}(u)} r_{i,j} \cdot P_{tx}(u,v) + \sum_{v \in \Gamma^{-}_{f(s_i,r_j)}(u)} r_{i,j} \cdot P_{rx} + P_{id}(1 - 2r_{i,j}) \tag{16}$$

nella quale le sommatorie comprendono un unico termine. Semplificando ulteriormente la notazione si può scrivere:

$$P(u) \stackrel{\triangle}{=} r_{i,j} \cdot P_{tx}(u,v) + r_{i,j} \cdot P_{rx} + P_{id}(1 - 2r_{i,j}) \tag{17}$$

dove il nodo v è il nodo a cui sono ritrasmessi i dati da u secondo il cammino $f(s_i, t_j)$.

Estendendo come in precedenza la formula al caso in cui il nodo u sia interessato da più flussi, si ottiene:

$$P(u) = \sum_{(s_i,t_j,r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot (P_{tx}(u,v) + P_{rx}) + P_{id} \left(1 - 2 \sum_{(s_i,t_j,r_{i,j}) \in \mathcal{I}} r_{i,j} \right) \tag{18}$$

$$P(u) = P_{id} + \sum_{(s_i,t_j,r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot (P_{tx}(u,v) + P_{rx} - 2P_{id}) \tag{19}$$

In questo caso il problema di configurazione a potenza minima diviene più trattabile ed assume la seguente forma:

Definizione 3.11 (MPCII) *Dato un grafo $G = (V, E)$ che descrive la topologia di una WSN ed un insieme $\mathcal{I}$ di richieste di comunicazione tale per cui il singolo flusso di dati non sia suddivisibile lungo più cammini, trovare un sottografo $G' = (V', E')$ di G ed un cammino $f(s_i, r_j)$ per ogni elemento di $\mathcal{I}$, tale per cui la potenza totale consumata $P(G')$ è minima:*

$$P(G') = \min \sum_{u \in V'} P(u) = |V'| P_{id} + \sum_{u \in V'} \sum_{(s_i,t_j,r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot C_{u,v} \tag{20}$$

dove:

$$C_{u,v} = P_{tx}(u,v) + P_{rx} - 2P_{id} \tag{21}$$

Nel seguito l'attenzione sarà volta solo al problema **MPC2** e per questo saranno proposti algoritmi risolutivi.

3.2.4 Proprietà del problema MPC2

Nel caso particolare del problema **MPC2** è possibile dimostrare che l'insieme di nodi $f(s_i, t_j)$ è il cammino a costo minimo in V' dal nodo s_i al nodo t_j secondo i pesi $C_{u,v}$.

Proposizione 3.1 *Se il flusso di dati generato da un nodo s_i non può essere diviso in più flussi instradati su cammini distinti, allora il cammino ottimo per il problema MPC $f(s_i, t_j)$ è il cammino minimo in G' dal nodo s_i al nodo t_j secondo i pesi $C_{u,v}$.*

Dim.
Si osservi inizialmente che, non essendo possibile suddividere il flusso in più sottoflussi, l'insieme $f(s_i, t_j)$ è effettivamente un cammino.

Supponiamo dunque che esista un cammino ottimo $f(s_1, t_1)$ per il problema MPC ma che non sia il cammino a costo minimo dal nodo s_1 al nodo t_1 secondo i pesi $C_{u,v}$. Si deve allora dimostrare che il costo in G', $P(G')_1$, utilizzando tale cammino, non è minimo, dove

$$P(G')_1 = |V'|P_{id} + \sum_{u \in V'} \sum_{(s_i, t_j, r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot C_{u,v}$$

Se $f(s_1, r_1)$ non è il cammino minimo in $|V'|$, esisterà allora un insieme di nodi $h_1, \ldots, h_m$ appartenenti a V' tali per cui, il nuovo cammino $s_1, h_1, h_2, \ldots, h_n, t_1$ è minimo. Si chiami $f_2(s_1, t_1)$ il nuovo cammino.

Riscrivendo la formula per il calcolo di $P(G')_1$ si ottiene:

$$\begin{aligned} P(G)_1 &= |V'|P_{id} + \sum_{(s_i, t_j, r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot \sum_{u \in V'} C_{u,v} \\ &= |V'|P_{id} + \sum_{(s_i, t_j, r_{i,j}) \in \mathcal{I}} r_{i,j} \cdot C_{f(s_i,t_j)} \end{aligned} \tag{22}$$

dove $C_{f(s_i,t_j)}$ è proprio il costo del cammino $f(s_i, t_j)$ secondo i pesi $C_{u,v}$.

Dato che $C_{f(s_1,t_1)} > C_{f_2(s_1,t_1)}$ si ha:

$$\begin{aligned} P(G)_1 &= |V'|P_{id} + \sum_{(s_i, t_j, r_{i,j}) \in \mathcal{I} \setminus (s_1, t_1, r_{1,1})} r_{i,j} \cdot C_{f(s_i,t_j)} + r_{1,1} \cdot C_{f(s_1,t_1)} \quad > \\ P(G)_2 &= |V'|P_{id} + \sum_{(s_i, t_j, r_{i,j}) \in \mathcal{I} \setminus (s_1, t_1, r_{1,1})} r_{i,j} \cdot C_{f(s_i,t_j)} + r_{1,1} \cdot C_{f_2(s_1,t_1)} \end{aligned} \tag{23}$$

E quindi se $f(s_1, t_1)$ è un cammino ottimo allora è il cammino di costo minimo in V' fra il nodo s_1 ed il nodo t_1. ■

Si può dimostrare che il problema **MPC2** e quindi, il problema **MPC** , appartengono alla classe dei problemi NP-completi[10]. Esistono però casi particolari in cui il problema **MPC2** appartiene alla classe dei problemi risolubili in tempo polinomiale anche su macchine deterministiche, essi sono:

[10] Si veda ad esempio [**?**].

1. Quando l'insieme dei nodi è dato dall'unione dell'insieme dei nodi sorgenti S e quello dei nodi riceventi T;

2. Quando la potenza di richiesta in stato di idle è nulla, $P_{id} = 0$.

Nel primo caso infatti $V' = V$ ed il problema diventa la ricerca dei cammini minimi su G fra i nodi s_i e t_j con gli archi pesati secondo le quantità $r_{i,j} \cdot C_{u,v}$. Analogamente nel secondo caso, il costo nullo dei nodi nello stato di idle, sposta nuovamente il problema alla ricerca dei cammini minimi secondo quanto precedente espresso.

Esiste infine un caso molto particolare, ma anche estremamente importante, con cui si riesce a dimostrare facilmente l'appartenenza del problema alla classe dei problemi NP-completi. Tale è quando il costo dei nodi $C_{u,v}$ è nullo e vi è un unico nodo ricevente. In tal caso il problema, che possiamo indicare come **MPC3** richiede la minimizzazione del seguente indice:

$$P(G') = \min |V'| P_{id} \tag{24}$$

e si può dimostrare che il problema equivale alla ricerca dell'albero di Steiner[11] su di un opportuno grafo, come espresso dalla seguente proposizione.

Proposizione 3.2 *Il problema* **MPC3** *coincide con la ricerca dell'albero di Steiner a costo minimo sul grafo $G' = (V, E')$, in cui gli archi di G' pesano tutti P_{id} e in cui si prende $H = S \cup \{t\}$ come insieme dei nodi che si desidera connettere con l'albero.*

Dim.
Sia ST un algoritmo di ricerca dell'albero di Steiner a costo minimo e sia $T = (V_T, E_T)$ il grafo determinato applicando ST a G'. Si supponga che esista un grafo connesso $G'' = (V'', E'')$ che collega i nodi in H al nodo t con un costo inferiore a T e si ricordi che, nel caso del problema **MPC3**, gli archi hanno costo nullo. Secondo l'equazione (20) i grafi T e G'' hanno costo:

$$P(T) = |V_T| \cdot P_{id} = (|E_T| + 1) \cdot P_{id}$$

$$P(G'') = |V''| \cdot P_{id}$$

Se fosse $P(G'') < P(T)$ allora si avrebbe $|V''| \cdot P_{id} < |V_T| \cdot P_{id}$ e quindi

$$|V''| < |V_T| = |E_T| + 1 \tag{25}$$

D'altronde G_T è stato determinato applicando un algoritmo di ricerca dell'albero di Steiner sul grafo G' e quindi, E_T è l'insieme di archi di cardinalità minima che connette i nodi dell'insieme H e, dato che l'albero è il grafo connesso col minor numero possibile di archi rispetto ai nodi, si ha:

$$|E_T| < |V''| - 1 \leq |E''| \tag{26}$$

Confrontando ora l'equazione (25) e l'equazione (26) si ottiene

$$\begin{cases} |E_T| > |V''| - 1 \\ |E_T| < |V''| - 1 \end{cases}$$

[11] Per la definizione di Albero di Steiner si veda l'appendice A.

ecco quindi che $|V''|$ non può essere inferiore a $|V_T|$ e quindi, $P(T) \leq P(G'')$. ■

Dato che il problema **MPC2** appartiene alla classe dei problemi NP-completi e dato che non ne è noto l'algoritmo ottimo risolutivo, mentre si conoscono algoritmi ottimi che risolvono il problema nei casi particolari citati, l'idea è di applicare tali algoritmi al problema **MPC2** e determinare quale sia l'errore massimo commesso rispetto ad un ipotetico algoritmo ottimo. Nei prossimi paragrafi quindi, si studieranno quali sono gli errori massimi che si commettono applicando algoritmi di ricerca di cammini minimi o di ricerca dell'albero di Steiner al problema **MPC2**.

3.3 Estensione del problema MPC2 al caso di errori nelle comunicazioni fra i nodi

In questo paragrafo si imposterà formalmente il problema **MPC2** nel caso in cui i link fra i nodi siano soggetti a perdita di pacchetti. Si noti infatti che finora è sempre valsa l'ipotesi che la comunicazione fra i nodi avvenisse senza errori.

Misure svolte su reti wireless reali indicano invece che è frequente il caso di ritrasmissioni di pacchetti e questo modifica profondamente le formule espresse in precedenza in merito al consumo energetico di un generico nodo u (eq. (13), 19). Per un'analisi più aderente alla realtà è dunque necessario considerare, oltre al rate trasmissivo dei dati ed alla potenza minima di comunicazione, anche la qualità del canale. Una possibile soluzione, che riconduce il problema a quanto esposto in precedenza, potrebbe essere la definizione della potenza minima di trasmissione fra due nodi u e v secondo la seguente formula[12]:

$$P_{tx}(u,v) = arg\min_{P_{tx}} \frac{P_{tx}}{PRR(u,v,P_{tx})} \tag{27}$$

dove $PRR(u,v,P_{tx})$ è il rapporto di ricezione dei pacchetti fra il nodo u e v alla potenza di trasmissione P_{tx}.

Una scelta alternativa, proposta dal nostro gruppo, è l'utilizzo dell'informazione sulla statistica di ritrasmissione dei pacchetti per modellare la banda di trasmissione dei nodi come una variabile aleatoria. Se in fatti un nodo u, mediamente, dovesse ritrasmettere un pacchetto n volte, allora la banda del nodo u la si potrebbe assumere ridotta di un fattore n. In base a questo è possibile, nota la statistica di ritrasmissione dei pacchetti per ogni link, modellare la banda trasmissiva di ogni connessione come una variabile aleatoria $\mathcal{B}$ a valori in $(0, B]$. Il nuovo rate $r_{i,j}(u)$, che ora è funzione di variabile aleatoria, viene a dipendere, secondo le definizioni date, anche dal link su cui è calcolato e lo si indicherà quindi con $r_{i,j}(u,v;B)$, dove v è il nodo a cui u trasmette i dati.

Può trarre in inganno il fatto che in generale, secondo le nuove definizioni date, si possa avere per un generico nodo u un data rate relativo entrante diverso da quello uscente:

$$\sum_{v \in \Gamma^{-}_{f(s_i,t_j)}(u)} r_{i,j}(v,u;\mathcal{B}) \neq \sum_{v \in \Gamma^{+}_{f(s_i,t_j)}(u)} r_{i,j}(u,v;\mathcal{B})$$

[12] Idea introdotta anche in [?].

ma questo deriva dal fatto che la banda trasmissiva d'ingresso e d'uscita del nodo u è differente. Resta comunque valida la seguente eguaglianza:

$$\sum_{v\in\Gamma^-_{f(s_i,t_j)}(u)} r_{i,j}(v,u;\mathcal{B})\cdot\mathcal{B} = \sum_{v\in\Gamma^-_{f(s_i,t_j)}(u)} R_{i,j}(v) = R_{i,j}(u) =$$
$$= \sum_{v\in\Gamma^+_{f(s_i,t_j)}(u)} R_{i,j}(v) = \sum_{v\in\Gamma^+_{f(s_i,t_j)}(u)} r_{i,j}(u,v;\mathcal{B})\cdot\mathcal{B}$$

Il problema **MPC2** è ora esprimibile come:

Definizione 3.12 *Sia dato un grafo $G=(V,E)$ che descrive la topologia di una WSN, un insieme $\mathcal{I}$ di richieste di comunicazione, tale per cui il singolo flusso di dati non sia suddivisibile lungo più cammini e siano note le statistiche di ritrasmissione dei pacchetti per ogni arco della rete. Determinare un sottografo G' di G ed un cammino $f(s_i,r_j)$ per ogni elemento di $\mathcal{I}$, tale per cui la potenza media totale consumata $P(G')$ è minima:*

$$\begin{aligned}\bar{P}(G') &= \min E_{\mathcal{B}}\left[\sum_{u\in V'} P(u;\mathcal{B})\right] = |V'|P_{id} + \sum_{u\in V'}\sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} E_{\mathcal{B}}[r_{i,j}(u,v;\mathcal{B})]\cdot C_{u,v} \\ &= |V|P_{id} + \sum_{u\in V'}\sum_{(s_i,t_j,r_{i,j})\in\mathcal{I}} \bar{r}_{i,j}(u,v)\cdot C_{u,v}\end{aligned} \tag{28}$$

dove:

$$C_{u,v} = P_{tx}(u,v) + P_{rx} - 2P_{id} \tag{29}$$

E' utile osservare che, anche sotto l'ipotesi di non separabilità dei flussi, a causa della differente qualità dei link fra i nodi di hop, non è più vero che $\bar{r}_{i,j}$ coincide per tutti i nodi di un cammino $f(s_i,t_j)$ e quindi, questo problema coincide formalmente con quanto espresso in (14) nella quale però, il rate dei dati $r_{i,j}(u)$ è sostituito dall'aspettazione di $r_{i,j}(u,v;\mathcal{B})$.

4 Algoritmi risolutivi

In questa sezione si analizzeranno gli algoritmi di ricerca di cammino minimo o di ricerca dell'albero di Steiner quando sono applicati al caso **MPC2** nell'ipotesi di *simmetria dei costi*: $C_{u,v} = C_{v,u}$ e di *unicità del nodo di sink*. Si dimostrerà quindi che, l'utilizzo di algoritmi di ricerca di cammini minimi introducono un errore relativo, rispetto alla soluzione ottima, non superiore al numero massimo di nodi sorgenti e che, gli algoritmi basati sulla ricerca dell'albero di Steiner introducono errori non superiori ad una predeterminabile costante indipendente dal numero di nodi sorgenti.

4.1 Errore relativo commesso da algoritmi di ricerca del cammino minimo

Si consideri il problema **MPC2** e si aggiunga l'ipotesi di archi simmetrici e di unicità del nodo ricevente: $T=\{t_1\}$. Con riferimento alla figura 8 a fronte, sia G il grafo che

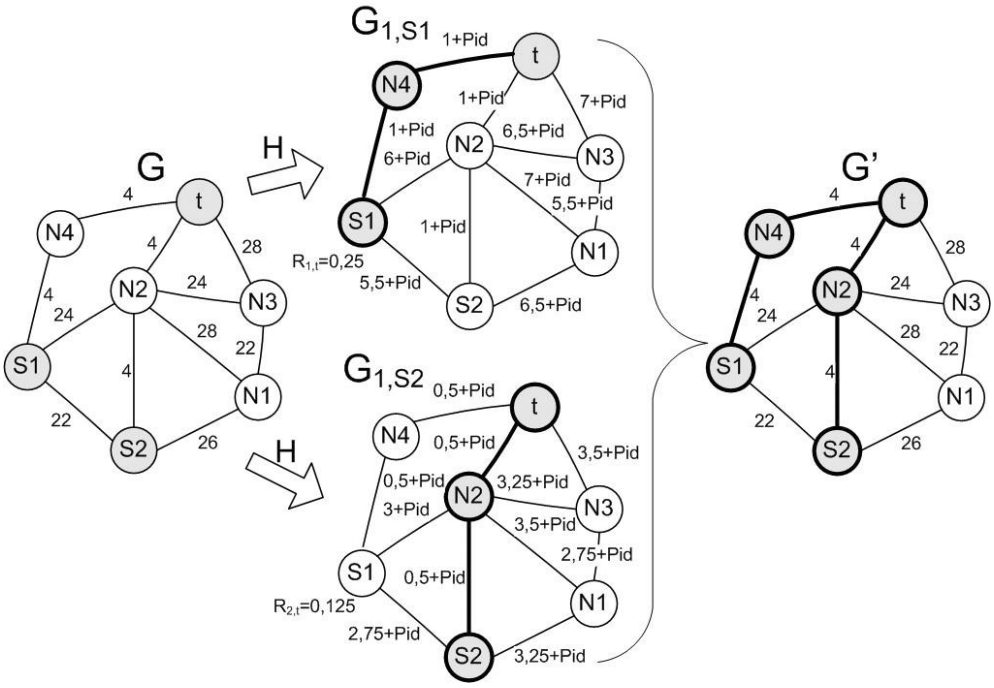


Figura 8: Esempio di applicazione dell'algoritmo H al grafo G.

descrive la topologia della rete in esame e si consideri $G_1 = (V, E_1)$ nel quale il costo degli archi $\tilde{C}_{u,v}$ è così calcolato:

$$\tilde{C}_{u,v} = C_{u,v} \cdot r_{i,1} + P_{id} \,, \qquad \forall u, v \in V$$

Sia H un algoritmo di ricerca di cammini minimi[13] su di un grafo e sia G' il grafo determinato dall'algoritmo H applicato ai grafi G_1, vale allora la seguente:

Proposizione 4.1 *Indicato con $G_{\min}$ il grafo ottimo e con $P(G_{min})$ il suo costo e con $P(G')$ il costo del grafo G', determinato applicando l'algoritmo H sul grafo G_1, si ha che $P(G') \leq |S| \cdot P(G_{min})$.*

Dim.
Dato che un flusso di dati non è suddivisibile lungo più cammini, si è dimostrato precedentemente che il costo del grafo può essere calcolato anche con la seguente formula:

$$P(G') = |V'|P_{id} + \sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} r_{i,1} \cdot C_{f_{s_i,t_1}}$$

nella quale $C_{f_{s_i,t_1}}$ è il costo del cammino minimo in G' da s_i a t_1.

Il costo della somma di tutti i cammini minimi presenti in G', secondo i pesi $\tilde{C}_{u,v}$, è invece dato da:

$$SP(G') = \sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} \tilde{C}_{f_{s_i,t_1}} = \sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} \left(r_{i,1} \cdot C_{f_{s_i,t_1}} + |f_{s_i,t_1}|P_{id} \right) \tag{30}$$

[13] Ad esempio l'algoritmo di Dijkstra.

Dove $|f_{s_i,t_1}|$ rappresenta il numero di nodi presenti nel cammino f_{s_i,t_1}.

Dato che il costo della potenza di idle nella formula (30) compare più di $|V'|$ volte, si ha $P(G') \leq SP(G')$. Si osservi che al più il costo di idle per ogni nodo, può essere contato $|S|$ volte. Si costruisca ora un grafo G_2, similmente a quanto fatto per G_1 prendendo però come grafo iniziale G_{min} e si indichi con G'' il grafo così ottenuto. Essendo G_{min} contenuto in G certamente si ha $SP(G') \leq SP(G'')$ e dunque:

$$\begin{aligned} SP(G'') &= \sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} \tilde{C}_{f_{s_i,t_1}} \leq \sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} r_{i,1}\cdot C_{f_{s_i,t_1}} + |S|\cdot|V'|\cdot P_{id} \\ &\leq |S|\left(\sum_{(s_i,t_1,r_{i,1})\in\mathcal{I}} r_{i,1}\cdot C_{f_{s_i,t_1}} + |S|\cdot|V'|\cdot P_{id}\right) \\ &= |S|P(G_{min}) \end{aligned} \tag{31}$$

e quindi, dato che $P(G') \leq SP(G') \leq SP(H(G_{min}))$, si ha:

$$P(G') \leq |S|P(G_{min}) \tag{32}$$

■

4.2 Errore relativo commesso da algoritmi di ricerca dell'albero di Steiner a costo minimo

Si osservi inizialmente che, data l'appartenenza del problema di Steiner alla classe dei problemi NP-completi, difficilmente ne esisterà una soluzione polinomiale e quindi si utilizzano degli algoritmi approssimanti la soluzione ottima, che presentano però un tempo di calcolo polinomiale rispetto alla dimensione dell'input. Sia dunque H un algoritmo che risolve il problema MST in un tempo polinomiale con un'approssimazione β, e sia $G' = (V, E')$ il grafo costruito a partire dal grafo G descrivente la topologia della WSN G. L'insieme degli archi E' coincide con E ma gli archi in E' hanno tutti costo identico e pari a P_{id}. Vale allora la seguente:

Proposizione 4.2 *Dato un algoritmo H che risolva il problema di Steiner sul grafo $G' = (V, E')$, se per ogni $(u, v) \in E$ si ha $C_{u,v} \leq \alpha P_{id}$, allora il costo del grafo determinato da H è inferiore a $(1 + \frac{\alpha}{2})\beta \cdot P(G_{min})$ dove $P(G_{min})$ è il costo del grafo determinato con l'algoritmo di ricerca dell'albero di Steiner.*

Dim.
Con riferimento alla figura 9 nella pagina successiva, sia $\bar{G} = (\bar{V}, \bar{E})$ il grafo ottenuto applicando un algoritmo ottimo di ricerca dell'albero di Steiner su G' e $G_H = (V_H, E_H)$ il grafo ottenuto applicando l'algoritmo H su G'. Dato che gli algoritmi sono stati eseguiti su di un grafo avente archi tutti dello stesso peso e, dato che il numero di nodi in un albero coincide col numero di archi meno uno, si ha:

$$|V_H| - 1 = |E_H| \leq \beta|\bar{E}| = \beta(|\bar{V}| - 1)$$

In particolare, essendo $\beta \leq 1$ si ottiene:

$$|V_H| \leq |\bar{V}| \tag{33}$$

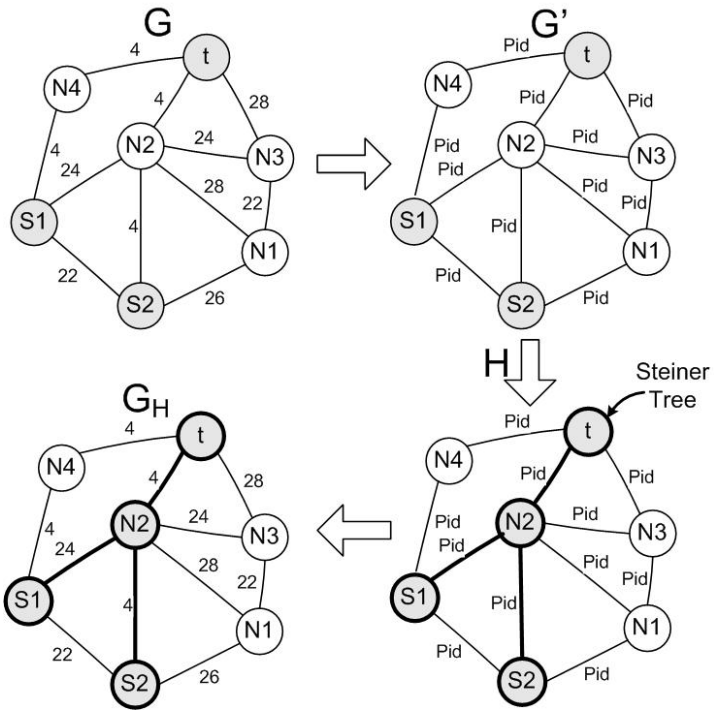


Figura 9: Passi per l'applicazione dell'algoritmo H al grafo G.

Indicato con $P(G_H)$ e con $P(\bar{G})$ la potenza media richiesta dalla rete relativa al grafo G_H ed al grafo $\bar{G}$, si ha:

$$\begin{aligned} P(G_H) &= |V_H|P_{id} + \sum_{u \in V_H} \sum_{(s_i,t_1,r_{i,1}) \in \mathcal{I}} r_{i,1} C_{u,v} \\ &\leq |V_H|P_{id} + \sum_{(u,v) \in E_H} \sum_{(s_i,t_1,r_{i,1})} \alpha P_{id} \cdot r_{i,1} \in \mathcal{I} \\ &\leq |V_H|P_{id} + \sum_{(u,v) \in E_H} \alpha P_{id} \cdot \sum_{(s_i,t_1,r_{i,1})} r_{i,1} \in \mathcal{I} \end{aligned} \tag{34}$$

Dato che per ipotesi $\sum_{(s_i,t_1,r_{i,1}) \in \mathcal{I}} r_{i,1} \leq \frac{1}{2}$ si ha:

$$\begin{aligned} P(G_H) &\leq |V'|P_{id} + \sum_{(u,v) \in E_H} \frac{\alpha P_{id}}{2} \\ &= |V_H|P_{id} + |E_H|\frac{\alpha P_{id}}{2} = |V_H|P_{id} + (|V_H| - 1)\frac{\alpha P_{id}}{2} \\ &= P_{id}\left[|V_H|\left(1 + \frac{\alpha}{2}\right) - \frac{\alpha}{2}\right] \leq P_{id}|V_H|\left(1 + \frac{\alpha}{2}\right) \end{aligned} \tag{35}$$

Ricordando ora l'equazione (33) si ottiene:

$$\begin{aligned} P(G_H) &\leq \beta|\bar{V}|\left(1 + \frac{\alpha}{2}\right) P_{id} \leq \beta\left(1 + \frac{\alpha}{2}\right)\left(|\bar{V}|P_{id} + \sum_{(s_i,t_1,r_{i,1} \in \mathcal{I})} r_{i,1} \cdot C_{f_{s_i,t_1}}\right) = \\ &= \left(1 + \frac{\alpha}{2}\right)\beta P(\bar{G}) \end{aligned}$$

(36)

Ora, l'algoritmo che ha determinato il grafo $\bar{G}$, essendo stato eseguito sul grafo G', ha determinato il numero minore di archi in modo da connettere tutti i nodi di $S \cup t$. ■

Per quanto visto, è dunque chiaro che gli algoritmi di ricerca del cammino a costo minimo raggiungono una soluzione più vicina all'ottimo quando il numero di sorgenti è ridotto e la potenza dissipata nello stato di idle è trascurabile rispetto alla potenza necessaria per una trasmissione o ricezione dei dati. Al contrario, gli algoritmi basati sulla ricerca dell'albero di Steiner sono maggiormente vicini all'algoritmo ottimo quando la potenza di idle è maggiore della potenza di trasmissione o di ricezione dei nodi. Osservando le caratteristiche elettriche dei nodi attualmente sul mercato, si veda ad esempio [**?**] o [**?**], si osserva che generalmente è la potenza in stato di idle ad essere trascurabile rispetto a quella di trasmissione o di ricezione via radio e quindi, con i nodi attuali, gli algoritmi di ricerca del cammino minimo risultano essere più promettenti.

A seguito di quanto appena visto, il nostro gruppo ha scelto di implementare ed implementare in simulazione alcuni algoritmi basati sulla ricerca di cammini ed una loro descrizione è riportata nella sezione 5.

5 Algoritmo scelto

L'algoritmo scelto dal nostro gruppo è un algoritmo di ricerca di cammini minimi, simile a quello descritto nel paragrafo 4.1 e l'idea base è presa da [**?**]. L'algoritmo scelto, si applica principalmente al caso in cui nella rete sia presente un unico nodo di sink. Questo particolare scenario non è da vedere come una limitazione in quanto sono numerosi i casi di interesse pratico in cui la tipologia di comunicazione è del tipo del tipo many-to-one, si pensi ad esempio a tutti i casi in cui la rete di sensori viene utilizzata per l'analisi o il monitoraggio dei dati raccolti da sensori sparsi in una zona di interesse inoltre, l'estensione al caso generale aggiungerebbe poche difficoltà oltre quelle incontrare per l'implementazione dell'algoritmo scelto e, utilizzando quanto da noi prodotto, sarebbe possibile in breve tempo creare un nuovo algoritmo per il caso generale.

Si osservi come avviene il calcolo del cammino minimo in 4.1. Quando per un nodo sorgente s_i è scelto un cammino verso il sink, allora tutti i nodi lungo il cammino risultano attivi. Ciò nonostante, se un nodo s_h deve scegliere una nuova path, non si cura di quanti nodi siano già attivi e quindi può scegliere di attivare degli altri nodi su un cammino differente che magari, considerando l'informazione relativa ai nodi già attivi, risulta poi essere a costo non minimo.

Una soluzione a questo problema può essere quella di modificare il costo degli archi fra i nodi in base all'informazione sullo stato (attivo o spento) di un generico nodo u. In questo modo si favoriscono i cammini che utilizzano dei nodi già attivi, riducendo il costo complessivo del grafo rispetto al semplice algoritmo descritto in 4.1.

Indicato dunque con $h_i(u, v)$ il costo dell'arco (u, v) secondo questo nuovo algoritmo, che possiamo chiamare, seguendo anche quanto indicato nell'articolo originale, ISTH

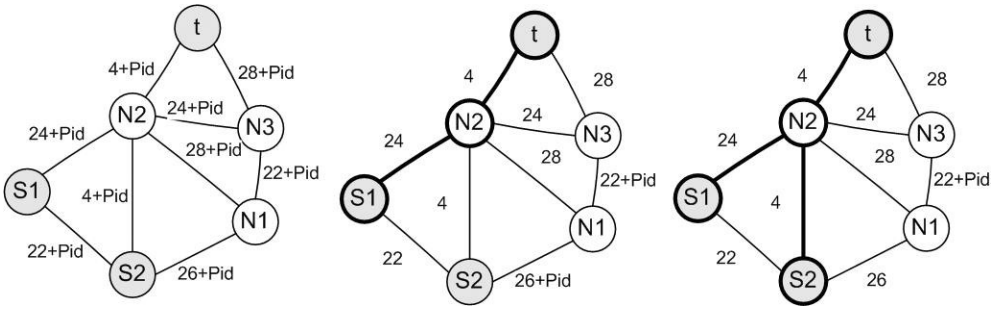


Figura 10: Esempio di esecuzione dell'algoritmo ISTH su di un grafo G.

(Incremental Shortest-path Tree Heuristic), si può scrivere:

$$h_i(u,v) = \begin{cases} r_{i,1} \cdot C_{u,v} + P_{id} & \text{se } u \text{ è spento} \\ r_{i,1} \cdot C_{u,v} & \text{se il nodo } u \text{ è attivo} \end{cases} \tag{37}$$

Di seguito sono indicati i singoli passi svolti dall'algoritmo per generare il grafo di potenza minima G', supposto noto il grafo $G = (V, E)$, l'insieme di potenze minime di comunicazione, l'insieme delle richieste di comunicazione $\mathcal{I}$, l'insieme S ed il nodo di sink t.

1. Poni $V' = E' = \phi$;
2. Segna tutti i nodi come spenti;
3. Per ogni nodo $s_i \in S$;
4. Trova il cammino minimo da s_i a t secondo i cammini $h_i(u, v)$;
5. Poni nello stato attivo tutti i nodi del cammino

In figura 10 è indicata una possibile esecuzione dell'algoritmo.

E' facile ora vedere che, essendo l'algoritmo di ricerca del cammino minimo eseguibile in un tempo $O(|E| \log |V|)$, questo algoritmo termina in un tempo $O(|S||E| \log |V|)$.

L'algoritmo studiato, ISTH, è stato implementato in Rmase. Una breve introduzione al simulatore la si trova in C. L'algoritmo per come è stato ideato si adatta bene sia a realizzazioni di tipo centralizzato che di tipo distribuito. L'implementazione centralizzata è abbastanza semplice, ma soffre ovviamente di tutti i problemi legati alla centralizzazione delle routine di gestione, con conseguente blocco delle comunicazioni nel caso di fault del nodo base/centrale. Per cui, sebbene più complessa da un punto di vista dell'implementazione, si è preferito sviluppare la versione distribuita che meglio si adatta alla caratteristiche di una WSN. L'applicazione è costituita da due layer principali più una serie di layer di supporto. Il primo dei layer principali, `MPC_hello`, implementa la ricerca delle potenze di trasmissione minime fra i singoli nodi della rete, mentre il secondo, `MPC_routing` si occupa del routing dei pacchetti scegliendo i cammini minimi secondo l'algoritmo ISTH che è in esso implementato.

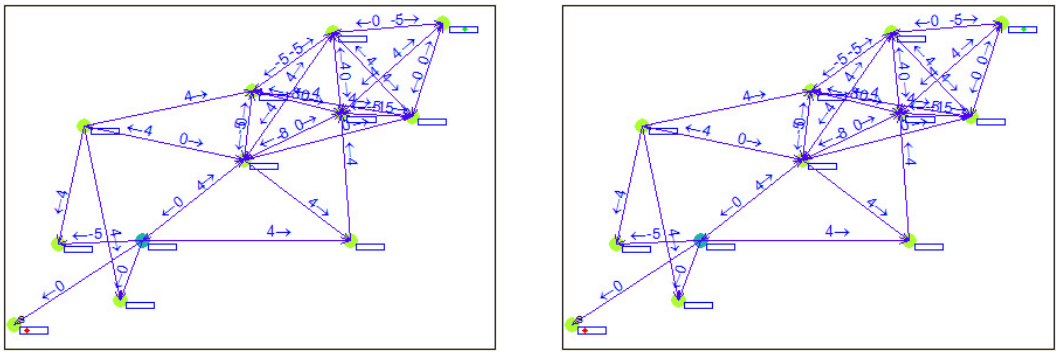


Figura 11: Screen shot durante l'esecuzione di alcune fasi dal layer MPC_hello.

5.1 Layer MPC_hello

Questo layer si occupa di determinare la potenza di trasmissione minima fra i nodi. Il suo corretto funzionamento è dunque fondamentale per il secondo layer di routing. Di questo layer sono state create due versioni, la prima, più semplice, è stata utilizzata per generare le simulazioni della sezione 6, la seconda versione, cerca di risolvere i problemi legati alle comunicazioni con un canale non ideale nel quale i pacchetti possono essere soggetti ad errori nella trasmissione o collidere.

Si noti comunque che, anche se la seconda versione dell'algoritmo esegue una ricerca della potenza minima in modo estremamente raffinato, ancora è ben lontana da una soluzione ottima. In particolare essa trascura completamente quanto esposto nel paragrafo 3.3. Questo è dovuto principalmente al fatto che tale impostazione suppone di conoscere la statistica di ritrasmissione dei pacchetti, cosa che, anche se possibile in simulazione, richiede di modificare moltissime routine interne di Rmase, software che al momento dell'ideazione del nuovo layer era ancora in fase di studio. Tale versione è stata quindi allegata a quanto prodotto dal gruppo ma non è stata utilizzata per la generazione delle simulazioni di seguito riportate, prodotte modificando i parametri del canale radio in modo da renderlo ideale.

Il funzionamento del layer MPC_hello, nel caso di canale ideale ed assenza di errori nella trasmissione dei pacchetti, è abbastanza semplice e si riduce all'invio di pacchetti di hello in broadcast a tutte le potenze di trasmissione. Quando un nodo rileva un messaggio di hello indica al nodo trasmittente la ricezione del messaggio e nell'interfaccia grafica è tracciata una freccia diretta, dal nodo che ha trasmesso verso il nodo che ha ricevuto, con sovrascritta la potenza di trasmissione con cui è stato instaurato il link. Si osservi però che sebbene il canale sia ideale è possibile che dei pacchetti di hello vengano comunque persi a causa di fenomeni di collisione, per cui in questa fase si è optato per far trasmettere i nodi secondo un meccanismo di scheduling di tipo TDMA.

5.2 Layer MPC_routing

Questo layer si occupa di soddisfare le richieste di instradamento dei pacchetti provenienti dai livelli superiori: livello di applicazione (OSI Layer 6) nel caso di nodi sorgenti e livello di trasporto (OSI Layer 4) nel caso di hop intermedi. Il meccanismo di

routing si basa su delle tabelle di instradamento che ogni nodo conserva nella propria memoria. Una tabella di esempio è riportata in 2. Poichè il costo di trasmissione varia con il rate della sorgente, in ciascuna tabella di routing sono elencati una serie di rate trasmissivi ad ognuno dei quali corrisponde un determinato percorso di instradamento verso il sink della rete. In particolare ad ogni rate è associato il prossimo nodo a cui inviare il pacchetto (il vicino con minor costo verso il sink), il costo per raggiungere il sink a partire dalla locazione attuale attraverso il vicino specificato e il numero di hop ancora da effettuare prima di raggiungere la destinazione. Inoltre è presente un campo sequenziale generato dal sink e utilizzato per la gestione degli aggiornamenti delle tabelle. In una prima fase di set-up il layer di routing attende che il calcolo delle potenze di trasmissione minimi da parte del layer MPC_hello termini. Il layer rileva questa informazione grazie alla ricezione di un pacchetto recante un flag di - fase terminata inviato dal layer MPC_hello verso il sink della rete. Una volta ricevuto questo pacchetto, il sink provvede a generare la prima tabella di routing, che conterrà tutta la serie di rate trasmissivi richiesti dall'utente e avrà ovviamente tutti i costi nulli essendo lui stesso il sink della rete(si ricordi infatti che il caso trattato è solo quello di comunicazioni many-to-one e che quindi, i costi di comunicazione sono sempre intesi come costi dal nodo attuale verso il sink). La tabella così creata viene poi tramandata di nodo in nodo fino a coprire tutti i nodi della rete. Ad ogni passaggio la tabella viene aggiornata con i costi di trasmissione ed eventualmente di idle del nodo ricevente. I costi sono calcolati secondo l'algoritmo ISTH e quindi il costo di idle può non essere incluso nel calcolo del costo totale qualora il nodo che ha appena ricevuto la tabella sia un nodo già attivo (il costo di accensione del nodo viene contato una volta sola). Il procedimento prosegue a passi di 1 hop sino a che i nodi sorgenti acquisiscono una prima tabella di routing. Quando un nuovo nodo sorgente comincia a spedire pacchetti, il vicino a cui inviarli viene prelevato dalla tabella andando a selezionare il record relativo al rate trasmissivo che più si avvicina a quello attuale della sorgente. Nel caso in cui il rate corrente si discosti oltre una certa soglia da quelli menzionati nella tabella di routing iniziale, il nodo sorgente provvede a marcare i pacchetti inviati con un flag che, una volta rilevato dal nodo sink, produce l'immissione in rete, da parte di quest'ultimo, di una nuova tabella (con nuovo sequenceID) aggiornata con il rate richiesto dalla sorgente. Un'altra situazione in cui avviene un aggiornamento delle tabelle di routing si ha quando il flusso dati proveniente da una sorgente viene completato. In questo caso infatti i nodi che facevano parte della path sorgente-sink, dopo un timeout dipendente dal rate del flusso che li attraversava, passano in stato di riposo, il che, indirettamente, alza i costi di routing, che devono quindi essere resi noti ai vicini tramite un adeguato aggiornamento delle tabelle. Per limitare l'overhead introdotto dalla propagazione delle tabelle di routing, queste vengono ritrasmesse dai nodi soltanto nel caso in cui il beneficio, in termini di costo, portato dalla nuova tabella ricevuta, sia superiore ad una soglia impostabile dall'utente.

5.3 Layer MPC_routing (Enhanced-ISTH)

Il layer di routing appena descritto rispecchia abbastanza fedelmente il comportamento dell'algoritmo ISTH distribuito presentato in [**?**]. Durante la scrittura del codice sorgente e nelle successive fasi di simulazione si è notato che, per come la strategia di routing è implementata, le path seguite dai flussi di dati non sono deterministica-

rate	next hop	cost to sink	hops to sink	seqID
0.1	19	59.67	6	1
0.5	19	130.35	6	1
1	28	200.4	4	1
2	28	316.8	4	1
4	28	549.6	4	1

Tabella 2: Tabella di routing relativa ad un nodo di una rete composta da 36 nodi.

mente collegate al set di sorgenti selezionato. In particolare supponendo di fissare un determinato set di sorgenti ed i loro corrispettivi rate trasmissivi, lanciando più volte la simulazione non è garantito il fatto che i percorsi scelti (e quindi il set di nodi attivi e il set di nodi spenti) rimangano gli stessi di volta in volta. Il motivo di questo comportamento è da individuare nel fatto che i percorsi che si instaurano dipendono dall'ordine con cui le sorgenti cominciano a spedire i dati nella rete. Supposto infatti che una prima sorgente abbia spedito dei dati (e quindi instaurato una path verso il sink), la sorgente successiva che inizierà a trasmettere, cercherà, per quanto possibile, di sfruttare i nodi già attivati dalla prima. Risulta quindi chiaro che la sorgente che trasmette per prima, individua una direzione preferenziale verso cui tutte le altre path tenteranno di confluire. Si è notato che questo comportamento alle volte può portare a configurazioni che sebbene soddisfacenti, non sono ottime dal punto di vista del costo totale sviluppato.

Per migliorare questo aspetto è stato ulteriormente sviluppato il layer analizzato in precedenza aggiungendo un meccanismo di update periodico (con cadenza relativamente lenta) delle tabelle di routing. Il periodo di questi update è liberamente impostabile dall'utente e la bassa cadenza in relazione al rate trasmissivo delle sorgenti, fa si che l'overhead introdotto sia trascurabile.Questo meccanismo permette tuttavia di risolvere il problema esposto, in quanto sebbene in una prima fase le sorgenti continuino ad utilizzare le path originarie, al primo passo di update, le path vengono ricalcolate e reimpostate in modo da trovare la configurazione ottima per quel particolare set di nodi sorgenti. Ovviamente il set di nodi sorgenti è soggetto a continui cambiamenti, per cui la configurazione ottima trovata è soltanto temporanea, ma se si suppone che la variazione del set di nodi sorgenti sia dell'ordine dell'intervallo di update, si ottiene un algoritmo che in genere si comporta meglio del precedente. Le prestazioni di questo algoritmo, che verrà d'ora in poi chiamato semplicemente ISTH, nella sezione 6 verranno confrontate con quelle date da algoritmi di tipo MinHop e STH di seguito discussi.

5.4 Layer MinHop

Il protocollo di routing MinHop è stato preso come mezzo di paragone nelle simulazioni effettuate, visto la sua semplicità di implementazione che ne permette l'uso anche in nodi con hardware dalle capacità estremamente limitate. L'idea alla base dell'algoritmo è quella di scegliere la path verso il sink che coinvolge il minor numero di nodi possibili. In questo modo gli altri nodi, non partecipanti alla trasmissione possono essere convenientemente spenti. Dal punti di vista del consumo energetico questo protocollo è da tenere in considerazione quando il numero di sorgenti è esiguo. Nelle reti con molto sorgenti attive contemporaneamente infatti c'è il rischio che la maggior

parte dei nodi venga attivata, visto che un nodo sceglie la propria path minima verso il sink indipendentemente dai cammini (e quindi dai nodi) già attivati.

5.5 Layer STH

Il protocollo STH è del tutto simile alla versione incrementale ISTH che ne rappresenta l'evoluzione. La differenza sostanziale fra i due protocolli è concentrata nella fase di calcolo dei costi di comunicazione fra i nodi. Nell'STH infatti il costo di idle di un nodo viene sempre aggiunto al costo totale di comunicazione verso il vicino, indipendentemente dal fatto che il nodo sia o non sia già attivo. Ciò comporta un consumo energetico che generalmente è superiore rispetto a quello sviluppato dall'ISTH ogni qualvolta vi sia più di una sorgente attiva contemporaneamente nella rete, visto che l'STH non tenta di sfruttare a pieno le path già instaurate, e quindi attiva dei nodi supplementari che potrebbe invece essere convenientemente lasciati spenti, cosa che solo l'ISTH è in grado di fare.

5.6 Layer MPC_hello, seconda versione

In questa sottosezione verrà spiegato brevemente il funzionamento della seconda versione del layer di hello.

Questo layer, contrariamente ai precedenti, utilizza un sistema di Ack per determinare se vi siano state collisioni od errori di trasmissione durante lo scambio di informazioni con il nodo di cui si sta attualmente studiando la potenza minima di trasmissione. Il layer di Ack gestisce la ritrasmissione dei pacchetti di ack ed i tempi di timeout secondo dei parametri scelti dall'utente. In particolare, solo i messaggi con identificatore maggiore di zero presuppongono l'invio di un ack da parte del nodo ricevente ed è così possibile scegliere per quali tipi di messaggi richiedere l'invio di pacchetti di ack.

Nel layer di hello, inizialmente, un nodo con ID prescelto comincia a trasmettere in broadcast i messaggi di hello a intervalli regolari partendo dalla minima potenza di trasmissione e aumentandola di volta in volta. Per i pacchetti di hello non è richiesta la generazione di Ack. Il nodo che riceve un pacchetto di hello, al livello del proprio layer di hello, genera un pacchetto di risposta per indicare l'avvenuta ricezione del pacchetto alla potenza indicata nel campo dati del pacchetto. Per il pacchetto di risposta, che può essere chiamato, helloReceived, è invece richiesta la conferma di ricezione attraverso l'invio di un pacchetto di ack.

Si è scelto di non richiedere l'invio di un ack per i pacchetti di hello in quanto, quando la potenza di trasmissione dell'ipotetico nodo trasmettente A scende al di sotto della potenza minima per la comunicazione con B, il layer di ack cercherebbe di reinviare più volte il messaggio al nodo B fino a quando non fosse ricevuto un ack da B relativo a quel pacchetto di hello, causando un rallentamento nella ricerca delle minime potenze di comunicazione, dato che B non sarebbe in grado di ricevere mai quel particolare pacchetto di hello.

L'invio del pacchetto helloReceived è inviato alla massima potenza e solo relativamente al primo pacchetto di hello che si riceve[14]. In particolare si osservi che, essendo i pacchetti di hello inviati a potenze crescenti, il primo pacchetto di hello ricevuto,

[14]Si ricordi che i pacchetti contengono all'interno un campo dati in cui è indicata la potenza a cui sono stati inviati e, su tale valore è svolto il confronto delle potenze.

coincide con il pacchetto di hello a minima potenza ricevuto. E' richiesto l'invio di un messaggio di ack per il pacchetto di helloReceived in quanto una sua non ricezione da parte del nodo che inizialmente aveva generato il pacchetto di hello, a causa dell'invio a massima potenza del pacchetto di helloReceived, può essere solamente dovuta ad una collisione o ad un errore di trasmissione.

Come si è detto, si suppone in questo caso che l'invio dei pacchetti di hello avvenga attraverso un canale rumoroso. Si è dunque scelto di far eseguire l'invio dei pacchetti di hello ad uno stesso nodo più volte ma in istanti temporali differenti. Il nodo A ad esempio, comincerà ad inviare pacchetti di hello in broadcast a partire dalla minima potenza di trasmissione fino alla massima per poi ricominciare questo ciclo di trasmissioni `counter` volte. Al termine, invierà un pacchetto particolare, di cui ancora si richiede la conferma di avvenuta ricezione tramite un pacchetto di ack, per indicare al nodo B di cominciare ad inviare pacchetti di hello secondo quanto fatto dal nodo A. Quando ogni nodo avrà terminato il proprio ciclo di hello, ricomincerà nuovamente la trasmissione del nodo A ed in seguito, di tutti gli altri nodi, esattamente come esposto precedentemente. Questo superciclo di trasmissioni avviene `hello_times` volte. La scelta di una tale successione di invii di pacchetti di hello è dovuta a due motivi precisi. Il primo è che, all'interno dello stesso superciclo, un nodo B potrebbe aver rilevato un messaggio di hello di A ad una potenza maggiore di quanto necessario in quanto esso si trovava in fase di trasmissione durante l'invio dei precedenti pacchetti di hello e non era quindi in grado di ricevere i pacchetti di hello. Il secondo motivo, è la presenza nel canale di disturbi tempo varianti che potrebbero inficiare la misura della potenza minima di trasmissione e perdurare per un tempo maggiore del solo ciclo di ritrasmissione che, si noti, in nodi con banda a 250[Kbps], può risultare anche solamente di qualche frazione di secondo. Ripetendo la trasmissione dei pacchetti di hello, a distanza di un superciclo, è possibile annullare tali effetti. Un superciclo infatti, in una rete numerosa, può divenire un intervallo temporale considerevole dell'ordine anche dei minuti. Si noti che nei casi reali i disturbi tempo varianti, con durata dell'ordine del secondo, sono frequenti e sono dovuti per lo più al movimento di oggetti o persone all'interno dell'ambiente in cui avvengono le misure.

Al termine dei super cicli, il layer di hello termina e passa le informazioni rilevate al layer di gestione della rete.

E' bene precisare che, questo algoritmo per l'identificazione delle minime potenze di comunicazione, richiede un gran numero di comunicazioni ad alta potenza (i pacchetti di ack) e quindi richiede alla rete un alto consumo energetico. Tale algoritmo quindi, nonostante esegua un'attenta ricerca della minima potenza di comunicazione fra i nodi, risulta pessimo per gli scopi di questo progetto, tanto più se si considera che tutti i nodi della rete sono mantenuti attivi durante l'esecuzione dei diversi supercicli.

Tale layer non è quindi stato utilizzato nelle simulazioni, come detto in precedenza, ma lo si è inserito nella relazione in quanto il gruppo ritiene che possa essere un valido punto di partenza per lo sviluppo di un valido algoritmo di ricerca delle minime potenze di comunicazione.

6 Risultati delle simulazioni

Di seguito sono riportati i risultati delle simulazioni svolte su differenti topologie di reti, indicate di volta in volta come topologia A,B,C,..., con differenti data rate e numero di nodi sorgenti.

In tutte le simulazioni che verrano esposte, i nodi sorgenti si trovano nell'angolo in basso a sinistra della rete ed il nodo di sink in alto a destra. Gli algoritmi di routing testati per ogni topologia sono: ISTH, STH e minHOP. Il modello di nodo utilizzato nelle simulazioni è relativo ai nodi MICA2. Si è scelto di utilizzare tale modello sia perché già presente in Rmase, ma sopratutto perché i risultati che si sarebbero ottenuti da un modello di nodo più recente, ad esempio un Tmote-Sky, scritto da noi, non sarebbero stati ritenuti affidabili, dato che si sarebbe dovuto modificare molte righe di codice di un simulatore che è tutt'ora in fase di studio.

Il confronto fra le prestazioni dei tre algoritmi è svolto in base ai grafici dell'energia, E, globalmente dissipata dalla rete e dalla potenza media P_m richiesta dalla rete a partire dall'istante iniziale. Per una maggiore chiarezza espositiva, per ogni topologia è anche riportata la potenza media richiesta dalla rete ad ogni secondo di funzionamento. Tale potenza, in forza del fatto che le misure relative all'energia dissipata dalla rete avvengono ad intervalli di un secondo, verrà chiamata impropriamente potenza istantanea.

6.1 Topologia A

La topologia illustrata in questa sottosezione, riportata in figura 12, si riferisce ad una rete distribuita su un'area di 100x100[m^2], formata da $N = 30$ nodi, nella quale i quattro nodi sorgenti inviano dati ad rate di 8[Kbps].

Dai grafici di figura 13 nella pagina seguente e 14 nella pagina successiva Si vede chiaramente come, in questa configurazione, l'algoritmo minHOP ritorni i risultati peggiori, mentre lo ISTH i migliori. Questo è dovuto principalmente al fatto che, come osservato precedentemente, per valori elevati di data rate sono preferibili comunicazioni a bassa potenza e quindi con molti hop.

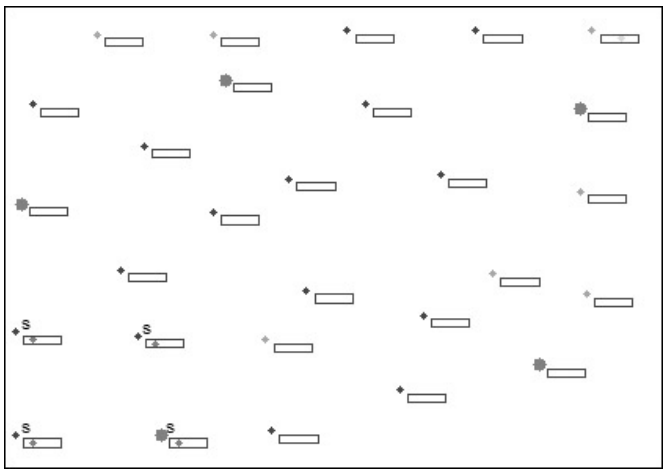


Figura 12: Topologia A.

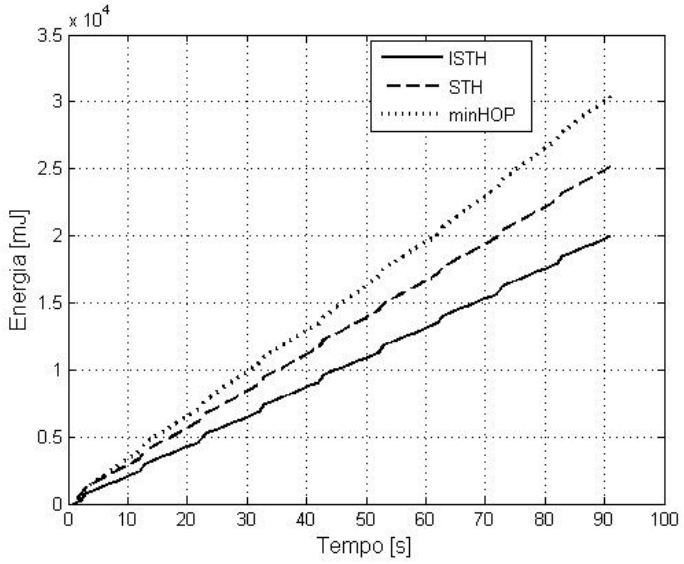


Figura 13: Energia globale dissipata dalla rete, topologia A.

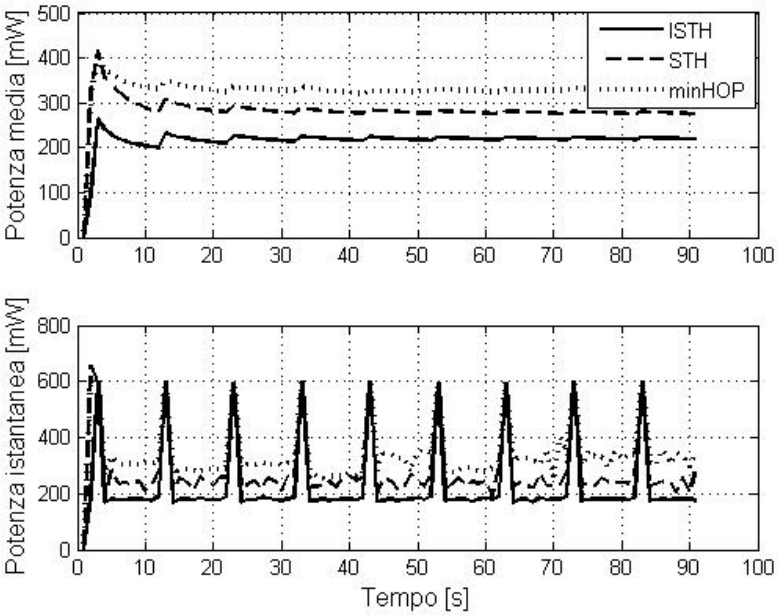


Figura 14: Potenza media e potenza istantanea relativa alla rete di topologia A.

I picchi di potenza visibili nel secondo grafico di figura 14 corrispondono agli istanti in cui tutti i nodi della rete si attivano per verificare se vi siano variazioni alle tabelle di routing e, il maggior consumo energetico dovuto a tali accensioni, è osservabile anche nella figura 13, in cui i consumi energetici procedono per gradini distanziati di 10[s], il periodo di duty cicle, l'uno dall'altro.

6.2 Topologia B

In questo caso la rete, formata da 36 nodi, è distribuita in un'area di dimensioni 100x100[cm], presenta undici nodi sorgenti ed il rate trasmissivo dei pacchetti è di 23 [bps]. La rete è stata simulata per un tempo $t = 1000$[s].

Osservando i grafici di figura 16 e 17 nella pagina successiva, si nota come le prestazioni dell'algoritmo minHOP ed STH siano pressoché identiche e si nota anche, che la loro potenza media converge al valore dato dall'algoritmo ISTH in circa 400[s]. Tale convergenza è dovuta al bassimo rate di trasmissione dei dati che porta l'algoritmo ISTH a scegliere i percorsi di comunicazione diretti fra i nodi e quindi, andando a realizzare le stesse scelte dell'algoritmo minHOP.

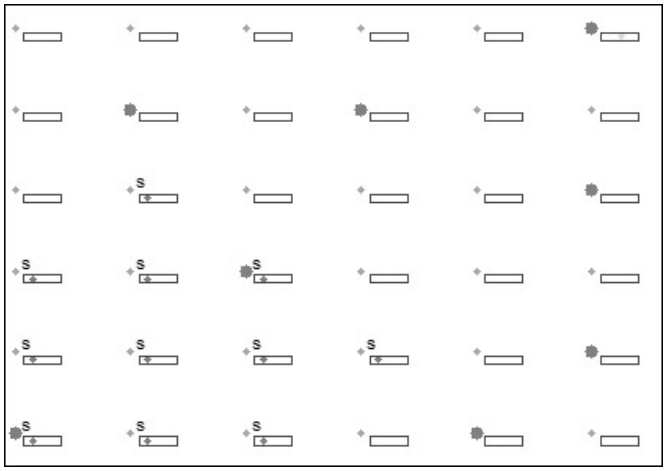


Figura 15: Topologia B.

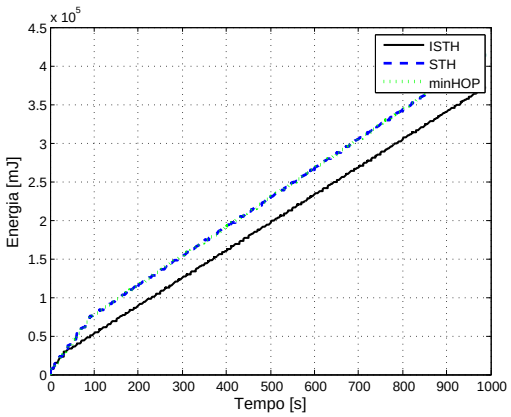


Figura 16: Energia globale dissipata dalla rete, topologia B.

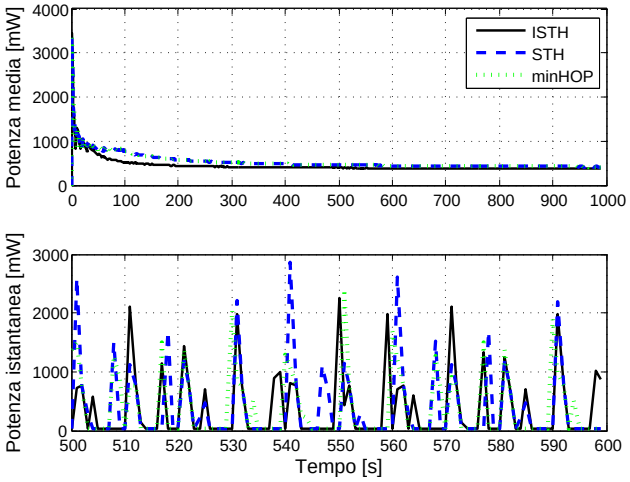


Figura 17: Potenza media e potenza istantanea relativa alla rete di topologia B.

6.3 Topologia C

La topologia illustrata in questa sottosezione, riportata in figura 12 a pagina 33, si riferisce ad una rete distribuita su un'area di 100x100[m^2], formata da $N = 22$ nodi, in cui i quattro nodi sorgenti inviano dati ad rate di 1.5[Kbps].

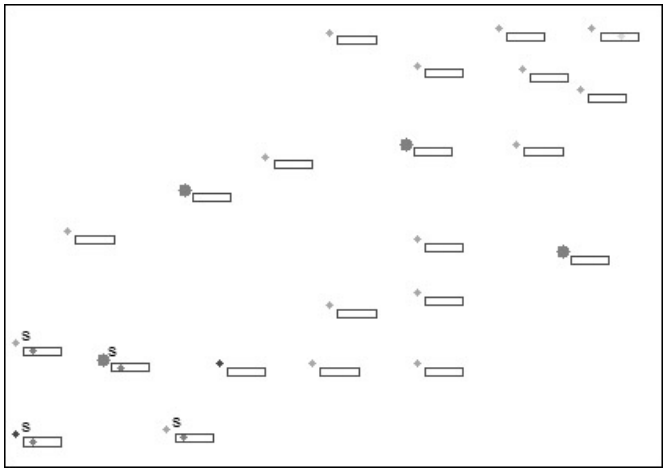


Figura 18: Topologia C.

Questa particolare topologia, in cui sono evidenziate sostanzialmente due percorsi per i dati diretti al sink, uno sopra alla diagonale che congiunge l'angolo in cui è posizionato il sink e l'angolo in cui si trovano i nodi sorgenti, ed uno al di sotto di

essa, rende l'algoritmo di minHOP migliore rispetto all'algoritmo STH. Dai grafici di figura 19 e 20, si vede infatti che la potenza media richiesta e l'energia dissipata dall'algoritmo STH sono superiori a quelle relative all'algoritmo minHOP.

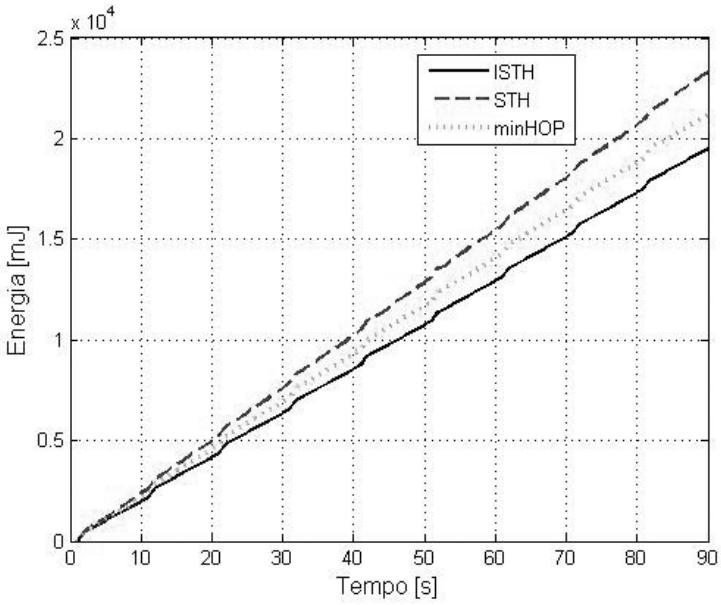


Figura 19: Energia globale dissipata dalla rete, topologia C.

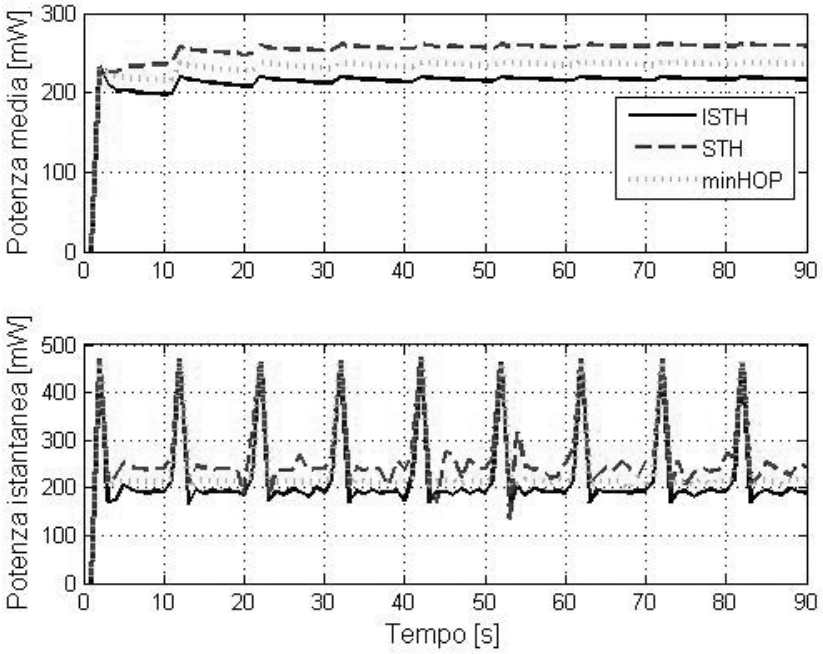


Figura 20: Potenza media e potenza istantanea relativa alla rete di topologia C.

7 Conclusioni

Nel progetto si sono presentati i principali filoni di ricerca che mirano ad una minimizzazione dell'energia dissipata da una WSN. Si sono indicati pregi e difetti delle singole linee di ricerca, verificando che da sole, queste non riescono a raggiungere il minimo consumo energetico e quindi, l'ottimo. Nella relazione si è successivamente introdotto il concetto di configurazione a potenza minima di una rete, dimostrando che, tale problema appartiene alla classe dei problemi NP-completi.

Ponendo diverse ipotesi semplificative al problema iniziale si è dimostrato che il problema **MPC** in alcuni casi particolari è risolubile in tempo polinomiale anche su macchine deterministiche. Questo ha permesso di studiare e determinare alcuni algoritmi in grado di approssimare la soluzione ottima e, in particolare, di determinare un bound per l'errore relativo commesso da tali algoritmi.

L'algoritmo proposto nella relazione nasce appunto come algoritmo di ricerca di cammini minimi su di un grafo i cui archi possiedono dei pesi che variano durante l'esecuzione dell'algoritmo. L'algoritmo si applica al solo caso in cui vi sia un unico nodo di sink nella rete, ma una sua estensione, utilizzando quanto prodotto dal nostro gruppo, non presenta particolari difficoltà. Per l'implementazione dell'algoritmo in Rmase si è dovuto anche creare un layer che determinasse le potenze minime di comunicazione fra i nodi: il layer di hello. Nel progetto, grazie all'ipotesi di idealità del canale, si è potuto utilizzare una versione estremamente semplice di tale layer. Nei casi reali invece, molta accortezza dovrebbe essere utilizzata per la ricerca della potenza minima di comunicazione fra due coppie di nodi in quanto gli errori di comunicazione fra i nodi, o disturbi esterni, possono modificare fortemente tale valore nel tempo. Si noti comunque che il gruppo ha cercato di produrre un layer di hello che sia utilizzabile anche nel caso di canale non ideale. Tale versione è allegata al progetto assieme agli altri file prodotti dal gruppo.

L'algoritmo ISTH utilizzato per la minimizzazione del consumo energetico della rete è stato confrontato con altri due algoritmi, uno, lo STH, di semplice ricerca del cammino minimo fra il nodo sorgente ed il nodo ricevente ed un'altro, minHOP, basato sulla minimizzazione del numero di HOP fra source e sink.

I risultati delle simulazioni svolte in Rmase indicano chiaramente che il protocollo di gestione della rete ISTH è il migliore fra quelli proposti.

Quanto presentato in questo progetto può essere visto come una prima introduzione al vasto problema della minimizzazione dell'energia di una WSN, esso quindi può essere utilizzato come un primo passo per una tesi di ricerca mirata allo studio ed alla creazione di algoritmi di minimizzazione energetica per reti wireless.

In appendice sono riportate la formalizzazione del problema dell'albero di Steiner (appendice A) e due piccole guide introduttive al software utilizzato per lo sviluppo del progetto, Prowler (appendice B) ed Rmase(appendice C).

A Il problema dell'albero di Steiner

Il problema dell'albero di Steiner, detto anche Minimum Steiner Tree (MST), nasce come problema geometrico nel quale un insieme di punti P, giacenti su una superficie, devono essere collegati da linee in modo che la lunghezza totale delle linee sia minima. Il problema diventa non banale se è possibile aggiungere un numero finito di punti del piano, detti appunto punti di Steiner e si richiede che due linee possano incontrarsi solo nei P punti o nei punti di Steiner. Un esempio di problema dell'albero di Steiner è riportato in figura 21.

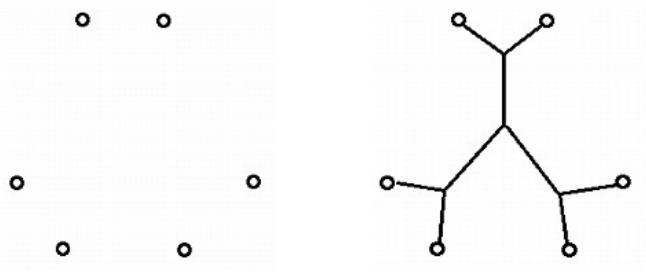


Figura 21: Esempio di creazione dell'albero di Steiner su di un piano.

Se i punti di Steiner sono un numero finito e predeterminato, si ottiene la seguente definizione:

Definizione A.1 (MST) *Dato un grafo $G = (V, E)$ i cui archi siano pesati, ed un sottoinsieme di nodi $H \subseteq V$, determinare un sottografo connesso a costo minimo che connetta tutti i nodi v appartenenti a H.*

Se gli archi in E hanno tutti costo maggiore o uguale a zero, allora il sottografo che si determina è un albero: l'albero di Steiner.

Si noti che il problema è apparentemente simile al problema di ricerca dell'albero coprente a costo minimo, in questo caso però, la possibilità di aggiungere all'albero i nodi appartenenti a $V \setminus H$, detti nodi di Steiner, rende il problema NP-completo. Per una dimostrazione dell'appartenenza del problema alla classe dei problemi NP-completi si veda [**?**].

B Prowler - Probabilistic Wireless Network Simulator

Proweler (http://www.isis.vanderbilt.edu/projects/nest/prowler/) è un simulatore per reti di sensori wireless ideato da Gyula Simon (http://home.mit.bme.hu/ simon/), inizialmente basato sulla sola piattaforma Matlab, recentemente è stato esteso anche in java sotto il nome di JProwler. Il simulatore è molto semplice da utilizzare ed una breve, ma completa guida è fornita nel file `Readme_vXX.txt`.

Ancorché semplice, questo simulatore è un ottimo tool per la simulazione di algoritmi per reti wireless in quanto costruito in modo modulare e quindi facilmente modificabile in ogni sua funzionalità. I livelli di rete implementati nel simulatore sono solamente i primi due del modello OSI: il livello fisico ed il livello di Mac. Il simulatore sfrutta inoltre una complessa ed utilissima interfaccia grafica, riportata in figura 22,

per la visualizzazione dello stato della rete. E' possibile ad esempio configurare i nodi affinché accendano un led (rosso, giallo o verde) in risposta a determinati eventi, oppure far tracciare delle frecce fra i nodi quando è ricevuto un messaggio.

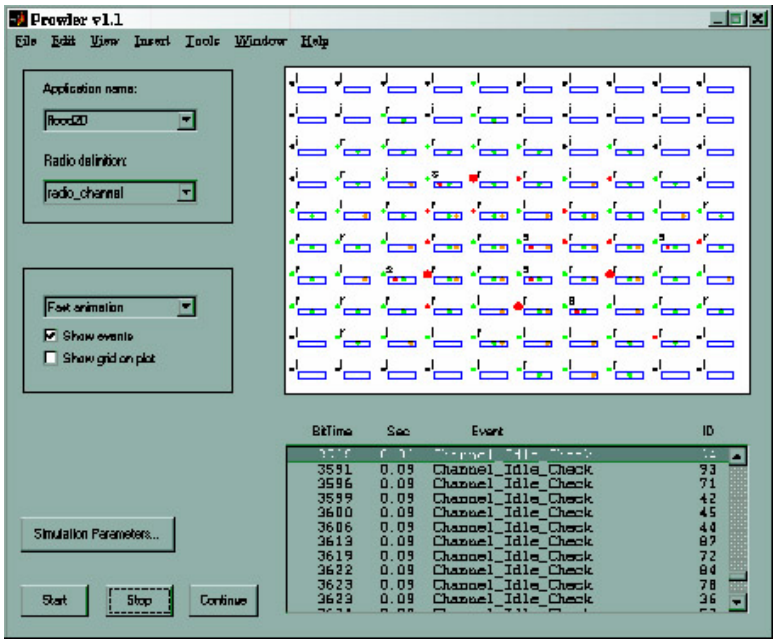


Figura 22: Interfaccia grafica di Prowler.

In merito all'implementazione del livello fisico, in Prowler esistono tre diversi modelli per la propagazione del segnale elettromagnetico. In tutti e tre i casi, il canale è modellato come un processo aleatorio a variabili indipendenti ed identicamente distribuite, ciò che cambia fra i diversi modelli, sono le distribuzioni di probabilità. E' comunque possibile, con un pò di accortezza, creare dei nuovi modelli per la propagazione del campo. Per il test degli algoritmi, ad esempio, è molto comodo utilizzare un modello di propagazione del campo elettromagnetico ideale e quindi, con decadimento quadratico della potenza del segnale elettromagnetico ed assenza di errori di trasmissione.

Fra le diverse funzionalità del livello fisico, vi è il calcolo del tempo necessario per la trasmissione del singolo pacchetto e, anche se il calcolo è svolto in maniera estremamente semplicistica (tempo costante per ogni pacchetto), tale funzionalità è facilmente modificabile ottenendo un conteggio del tempo necessario alla trasmissione differenziato sulla base della effettiva lunghezza dei pacchetti. Infine, sempre a livello fisico, sono previste le collisioni di pacchetti ed errori aleatori di trasmissione.

Per quanto riguarda il livello di Mac si deve prestare attenzione al fatto che il simulatore prevede un semplice meccanismo di tipo CSMA, mentre algoritmi più raffinati come il collision avoidance devono essere implementati dall'utente. L'uso di un semplice algoritmo di mac, ricalca però quanto avviene in molti nodi reali basati sulla piattaforma TinyOs e quindi il simulatore ricrea, in questo caso, il comportamento reale dei nodi in fase di trasmissione. Un esempio di sequenze per l'accesso al mezzo in Prowler è dato in figura 23 a fronte dalla quale si osserva lo stretto legame fra il livello di Mac

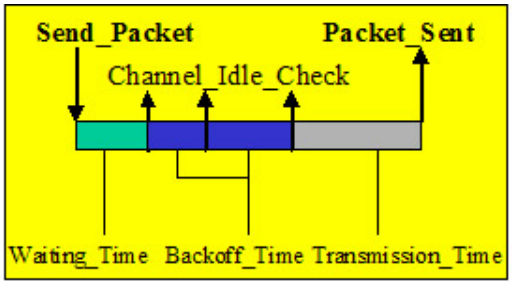


Figura 23: Sequenze per la trasmissione di un pacchetto in Prowler.

simulato in Prowler e quello presente in TinyOs. Prowler, è bene osservarlo, è stato ideato avendo come idea di nodo tipo, un nodo basato su TinyOs.

Per realizzare una nuova simulazione in Prowler, nominata ad esempio `Prova1`, è sufficiente creare tre file: `Prova1_topology`, `Prova1_animation` e `Prova1_application` e copiarli nella directory in cui si trovano gli altri file di Prowler. Il primo file indica la topologia della rete, il secondo come l'interfaccia grafica debba segnalare l'insorgere di un evento ed il terzo file contiene l'algoritmo di gestione dei nodi. Dei tre file il più interessante è certamente l'ultimo e su di esso si darà qualche ulteriore indicazione.

Essendo Prowler un simulatore basato su eventi, ad ogni ciclo esso verifica se debba lanciare una qualche evento fra: `Init_Application`, `Packet_Sent`, `Packet_Received`, `Collided_Packet_Received`, `Clock_Tick`, `Application_Finished` e `Application_Stopped`, od eseguire un qualche comando dei nodi: `Set_Clock, Send_Packet`. La simulazione termina quando non vi sono più eventi nella coda di eventi. Per eseguire un programma sui nodi è quindi sufficiente indicare nel file `Prova1_application` cosa far eseguire al nodo nel caso si verifichi uno degli eventi sopra citati. Un esempio di possibile codice all'interno del file `Prova1_application` è il seguente:

```
[...]
switch event
case 'Init_Application'
    source = 0;
    rate = 250000;
    Npower = 10;
    [...]


case 'Packet_Sent'

case 'Packet_Received'
    if ((data.data.dest == ID) || (data.data.dest == 0)
                                && (data.data.source~=ID))
    [...]

case 'Collided_Packet_Received'

case 'Clock_Tick'
```

```
case 'GuiInfoRequest'
   disp(sprintf('Memory Dump of mote ID# \%d:\n',ID)); disp(memory)

case 'Application_Stopped'
   % this event is called when simulation is stopped/suspended

case 'Application_Finished'
   % this event is called when simulation is finished
   disp(strcat('Node: ',num2str(ID)));
   disp(memory.power);

otherwise
    error(['Bad event name for application: ' event])
end
[...]
```

E' possibile far eseguire ad ogni nodo due comandi: `Set_Clock` e `Send_Packet`, la cui sintassi è la seguente:

```
Set_Clock(time_to_weak_up);

Send_Packet(radiostream(struct('type','ping', ...
                                'source',ID, ...
                                'dest',0, ...
                                'power',memory.signalstrength), ...
                                memory.signalstrength));
```

Si presti attenzione al fatto che il comando `Set_Clock` prevede l'indicazione dell'istante di risveglio del nodo, non il periodo del timer.

Un pacchetto, inviato col comando `Send_Packet`, può contenere qualunque campo in quanto è un oggetto di tipo struct. Nell'esempio si sono inseriti i campi `'type'` che identifica il tipo di pacchetto, il campo `'source'` il nodo che ha generato il pacchetto, il campo `'dest'` che indica a chi è indirizzato il nodo, zero generalmente indica un messaggio in boradcast ed infine `'power'`, che indica la potenza a cui il messaggio sarà inviato.

Il simulatore infine gestisce e consente la definizione di una zona di memoria indipendente per ogni nodo. I nodi possono dunque memorizzare qualunque cosa si desideri e possono rispondere agli eventi secondo quanto memorizzato. La definizione della memoria dei nodi avviene con la seguente sintassi:

```
memory=struct('signalstrength',signalstrength, ...
                  'source', source, ...
                  'power',power_to_tx, ...
                  'timesToTry', times_to_try, ...
                  'timeToSleep',time_to_sleep, ...
                  [...]
              );
```

Una leggera complicazione si ha nel caso che più nodi debbano gestire differenti parti di codice, in quel caso è però sufficiente inserire le diverse parti di codice all'interno di un blocco `switch, case` come nel seguente esempio:

```
case 'Packet_Received'
    switch ID
       case 1
          disp('Primo Nodo');
       case 2
          disp('Secondo Nodo');
       case 3
          [...]

     end
[...]
```

Il simulatore è quindi efficiente per la simulazione di semplici algoritmi di reti, ma risulta insufficiente nel caso si vogliano implementare algoritmi complessi che magari, necessitino di livelli di rete superiori a quelli di MAC.

C RMASE - Routing Modeling Application Simulation Environment

The Tao that can be taught is not the everlasting Tao.
The Name that can be named is not the everlasting Name.
That which has no name is the origin of heaven and earth.
That which has a name is the Mother of all things.
Tao Teh Ching, Lao Tzu

A system that can be modeled is not the system itself.
A model that can be made is not the absolute model.
That which has no model is the origin of a system.
That which has a model is the understanding of the system.
Zhang Ying

Rmase, contrariamente a Prowler, non è un simulatore di reti, bensi è ideato per valutare la qualità di diversi algoritmi di routing in base a parametri scelti dall'utente. Rmase, ideato e sviluppato da Zhang Ying (http://www2.parc.com/spl/members/yzhang/) membro del team Palo Alto Research Center dal 1999, è basato su Prowler ed attualmente implementato sulla sola piattaforma Matlab. Per valutare la qualità degli algoritmi di routing Rmase necessita di un simulatore, Prowler, ma si ricordi che, di per se, Rmase non è un simulatore di reti.

Rmase è distribuito con una documentazione alquanto scarna ed insufficiente per la complessità del tool. Per dare un'idea della complessità del tool si consideri che esso è formato da più di 180 file, ciascuno dei quali contiene diverse funzioni. Ad esempio, uno dei primissimi problemi che il nostro gruppo ha avuto con questo tool, risultato fondamentale per lo svoglimento del progetto, è stato proprio capire quali comandi

fossero necessari per l'esecuzione dello stesso, almeno su di uno dei tanti file di prova rilasciati dalla dott.sa Zhang.

Contrariamente a Prowler, Rmase fornisce all'utente tutti i layer del modello ISO/OSI di una rete e consente una loro modifica per testare la qualità di diversi algoritmi di routing. Nonostante il tool sia stato ideato specificatamente per l'esecuzione di test su algoritmi di routing, la sua costruzione modulare lo rende un tool estremamente flessibile ed utile per qualsiasi applicazione. E' ben precisare che Rmase non è un programma vero e proprio, bensì un insieme di routine collegate fra loro che preparano e settano le opzioni del simulatore Prowler rilasciato assieme al tool[15]. Prima di spiegare quali passi seguire per creare una nuova applicazione in Rmase, è necessario spiegare quale sia la sua architettura.

Si osservi la figura 24 a pagina 46. L'applicazione `FirstApp` è il file creato dall'utente per testare le proprie routine. Essa quindi setta per prima cosa i parametri di simulazione e, nel caso si vogliano utilizzare i parametri standard di utilizzerà il comando `sim_params('set_app_default')`, altrimenti è possibile impostare i diversi parametri di simulazione, ad esempio:

```
sim_params('set', 'RADIO_NAME','radio_channel_power');
sim_params('set_app','UseTopologyFile',1);
sim_params('set_app','TopologyFileName','MPC_topology');
```

In seguito è necessario indicare quali layer utilizzare con un comando del tipo:

```
set_layers({'mac','neighborhood','MPC_hello','MPC_routing','app'})
```

I layer obbligatori che devono essere utilizzati sono quelli di mac, di routing ed il layer app. Per eseguire la simulazione è infine necessario dare i seguenti comandi:

```
prowler('Init');
prowler('StartSimulation');
```

La prima istruzione inizializza il software di simulazione Prowler con le opzioni date dall'utente, la seconda istruzione invece, da avvio alla simulazione.

Similmente a Prowler, in Rmase ogni nodo ha a disposizione dei comandi, che viaggiano dai layer superiori a quelli inferiori ed è in grado di rilevare degli eventi, i quali seguono la direzione opposta. Sia i comandi che gli eventi possono essere bloccati in un determinato layer settando l'opzione `pass=0` relativamente a quel layer. Tale comando va però utilizzato con accortezza, in quanto agisce su tutti i comandi e su tutti gli eventi che transitano per il layer.

La generazione di un comando in Rmase è leggermente più complessa rispetto a quanto accadeva in Prowler in quanto vi sono diversi layer interconnessi fra di loro e quindi, comandi ed eventi, devono essere gestiti in modo più articolato. Un tipico comando per il settaggio di un clock in un nodo è ad esempio il seguente:

```
clock.type = 'hello_send';
prowler('InsertEvents2Q', make_event(alarm_time, 'Clock_Tick', ID, clock));
```

[15] Il programma Prowler inserito in Rmase è leggermente diverso dal Prowler del paragrafo precedente.

Con questo comando si richiede a Prowler di inserire nella coda di eventi un evento di tipo `'Clock_Tick'` per il nodo ID ed il tipo di clock da generare è dato da clock.type. Il campo clock.type risulta estremamente utile quando uno stesso nodo utilizza più clock per motivi differenti, ad esempio per algoritmi di sleep management e per l'intervallo di accensione e spegnimento di un led.

Nel caso invece si voglia comandare l'invio di un pacchetto, le istruzioni da scrivere sarebbero le seguenti:

```
pck.msgID = pckidentifier;
pck.address = IDnote_to_send;
pck.strength = index_strength;
pck.source = ID;
pck.forward = 0; %no multihop
[...]

status = MPC_hello_layer(N, make_event(t, 'Send_Packet', ID, pck));
```

I pacchetti di Rmase, similmente a Prowler, possono contenere qualunque campo, è importante però inserire nei pacchetti i campi necessari all'esecuzione di ogni layer. Nell'esempio si è inserito il campo forward in quanto il layer di routing, che si utilizzava nell'algoritmo da cui l'esempio è stato tratto, utilizzava tale informazione. Si noti la particolarità dell'ulimo comando. Esso richiama il layer su cui viene eseguito (nell'esempio: MPC_hello_layer) passandogli il numero di layer a cui esso stesso corrisponde (N) e l'evento 'Send_Packet', generato attraverso la funzione make_event.

Tale sintassi, può apparire inizialmente estremamente complessa. In realtà essa risulta spesso utilissima per semplificare in simulazione delle operazioni complesse che risulterebbero estremamente complesse su una rete reale. Con una tale sintassi infatti, si può generare un qualunque evento in un qualunque layer di un nodo scelto. Ad esempio è possibile segnalare l'avvenuta ricezione di un pacchetto al nodo B, anche se questo effettivamente non ha ricevuto nulla:

```
status = MPC_hello_layer(N, make_event(t, 'Packet_Received', B_ID, pck));
```

Al di sopra del layer app vi sono due layer invisibili all'utente, essi sono il layer di stats e di log. Il primo serve per generare le statistiche della rete, il secondo crea invece i dati di log della rete.

C.1 Creazione di un nuovo layer

Per creare un nuovo layer, nominato ad esempio `NewL` è necessario seguire i seguenti passi:

- Creare un file nominato `NewL_layer.m`
- Modificare il file `app_layers.m` e inserire `NewL` all'altezza corretta. `app_layers.m` contiene in ordine di livello OSI tutti i possibili layer utilizzabili da un'applicazione in Rmase e, alla lista in esso contenuta fanno riferimento le funzioni di Rmase per determinare l'ordine dei diversi layer. Ad esempio se `NewL_layer.m` fosse un nuovo layer di mac, esso andrebbe posto subito prima o subito dopo il layer di mac esistente.

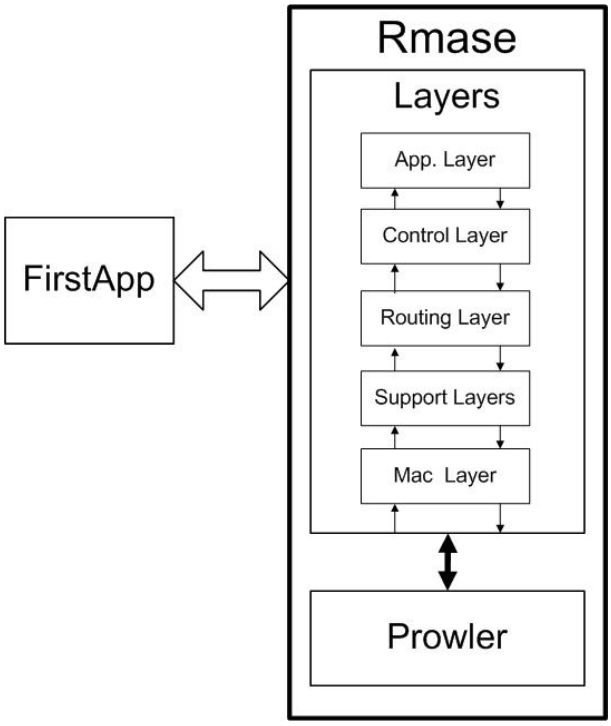


Figura 24: Struttura del software Rmase

- Se `NewL` contenesse dei parametri si dovrebbe creare anche un file `NewL_set_params.m` scrivendo per ogni parametro la seguente sintassi:

```
gId = gId + 1;
[param, i] = X_set_params(param, i, gId);
```

- Infine, per utilizzare tali parametri sarebbe necessario modificare il file `Rmase_params.m` aggiungendo il seguente codice:

```
i=i+1; x
param(i).name = 'NewL';
param(i).default = default;
param(i).group = 7;
param(i).type='checkbox';
```

Per quanto Rmase sia un software estremamente utile, talvolta ritorna degli errori che, probabilmente sono generati da bachi presenti nel codice. Il nostro gruppo non ha avuto il tempo di studiare in modo approfondito il codice sorgente di Rmase, ma l'esperienza con questo tool, suggerisce di verificarne la robustezza delle diverse routine da utilizzare prima di scrivere un nuovo layer. Un problema da noi riscontrato, anche se forse dovuto alla sola inesperienza del gruppo nell'uso del tool, è la perdita di pacchetti quando questi giungono ai nodi. E' capitato talvolta che dei nodi ricevessero una sequenza di pacchetti da inoltrare e che, alcuni, fra i pacchetti ricevuti andassero persi.

Purtroppo l'errore si è verificato troppo sporadicamente per essere studiato in modo approfondito e, esami sul codice da noi scritto, indicano che molto probabilmente il problema risieda nella gestione degli eventi di Rmase.