Localizzazione e *tracking* di agenti mobili mediante una rete di sensori *wireless*

Marco Bertinato, Giulia Ortolan, Patrizio Zambotto

***Abstract*— Questo documento è la relazione finale del progetto svolto per il corso di Progettazione di Sistemi di Controllo tenuto dal prof. L. Schenato dell'Università di Padova. L'oggetto è l'implementazione di una rete *wireless* per la localizzazione ed il *tracking* di nodi mobili; a questo proposito si è adottato un algoritmo che, mediante un filtro particellare con perdita di pacchetto, stima la posizione del nodo basandosi su una mappa di potenza dell'ambiente, ricavata a priori. Il progetto ha richiesto l'analisi di varie problematiche, quali la scelta delle soluzioni tecniche da adottare, lo studio della posizione per i nodi fissi (àncora) che costituiscono la rete e la programmazione degli stessi e dei nodi mobili che vanno localizzati. Per ogni problema sono state cercate soluzioni ottimali, secondo il criterio specificato di volta in volta. Una parte preponderante del progetto ha visto la realizzazione di un *testbed* presso i laboratori dell'Istituto Nazionale di Fisica Nucleare di Legnaro; una dettagliata documentazione dell'attività svolta conclude il lavoro presentato.**

***Index Terms*— Localizzazione, *tracking*, *WSN*, connettività, interpolazione, filtro particellare, perdita di pacchetto, TinyOS.**

I. Introduzione

L'OBIETTIVO del progetto è la realizzazione di un *testbed* per lo sviluppo di una rete wireless all'interno di un ambiente *indoor* di ampie dimensioni quale i Laboratori dell'Istituto Nazionale di Fisica Nucleare (INFN) di Legnaro[1]. Questa rete deve poter permettere, in caso di emergenza, la localizzazione ed il *tracking* in tempo reale degli operatori di soccorso che si introdurranno nei locali colpiti, in modo da poterne guidare le mosse dall'esterno e quindi facilitarli nell'intervento. Ogni operatore verrà dotato di un terminale (ad esempio, un palmare) in cui potrà visualizzare la sua posizione attuale ed il luogo da raggiungere; in una versione futura del progetto, inoltre, potrà ottenere informazioni aggiornate sul locale in cui si trova, in modo da essere a conoscenza della presenza di eventuali materiali esplosivi o radioattivi.

La volontà di realizzare un sistema di questo tipo trae origine da un incendio sviluppatosi nel 2005 a causa di un problema ad un gruppo di continuità elettrica. L'incidente, accaduto nei sotterranei dei Laboratori, ha messo in funzione tutte le opportune misure di sicurezza, ed i locali colpiti sono stati sommersi dalla schiuma antincendio; in quelle condizioni, tuttavia, l'intervento della squadra dei Vigili del Fuoco e dei Responsabili della Sicurezza dell'INFN è stato quantomai difficoltoso, dato che alla dedalea planimetria dei locali si era aggiunta la scarsa visibilità dovuta alla schiuma.

Documento consegnato il 23 Luglio 2007.

[1]INFN - Istituto Nazionale di Fisica Nucleare, Laboratori Nazionali di Legnaro, viale dell'Università, 2, 35020 Legnaro (Pd) Italia. Sito: `http://www.lnl.infn.it`.

Un progetto analogo, chiamato *Fire Information and Rescue Equipment (FIRE)* [1], è stato sviluppato dall'Università di Berkeley, e dalla primavera del 2006 è stato adottato dal *Chicago Fire Dept.*. Il *FIRE system* è composto di due elementi: *SmokeNet*, una rete di sensori wireless che permette di rilevare lo stato di ciascun ambiente (temperatura, presenza di fumo etc.) e la posizione di ciascun pompiere, per poi trasmettere ogni informazione alla centrale operativa esterna, in grado di stabilire con precisione la posizione di ogni agente grazie alla conoscenza della planimetria dei locali coperti dalla rete; *FireEye*, un particolare equipaggiamento per i pompieri tra cui compare anche un elmetto con un piccolo display LCD rivolto verso l'interno (si veda Fig. 1), che permette di visualizzare la planimetria dell'ambiente e la posizione propria e dei compagni. Il progetto, ancora in fase di test, sembra ottenere buoni risultati e la speranza è quella di realizzare una rete che copra l'intera città di Chicago.

Fig. 1: L'elmetto del progetto *FireEye*.

Il progetto presentato mira a risolvere il problema della localizzazione e del *tracking* di un nodo mobile sfruttando la misura della potenza dei segnali ricevuti dai moduli wireless fissi, opportunamente piazzati in modo da garantire la totale copertura dell'ambiente, ed un'elaborazione basata sull'impiego di un filtro particellare con perdita di pacchetto. A differenza della maggior parte delle soluzioni adottate per la localizzazione con reti wireless, qui non si fa uso di metodi geometrici, ma ci si basa sul confronto con una mappa di potenza del segnale ricavata da rilevazioni a priori. L'intero progetto è stato svolto con un occhio di riguardo verso la scalabilità e l'espandibilità delle funzioni svolte, cercando soluzioni ottimizzate che non richiedano ingenti risorse. Come mostrato dal progetto *FIRE*, il sistema che si vuole implementare può prestarsi agli scopi più diversi, dalla localizzazione al monitoraggio di ambienti; esso inoltre può essere visto come

un punto di partenza verso lo sviluppo di agenti di intervento artificiali completamente autonomi, che possano introdursi senza problemi anche in ambienti a rischio per l'uomo e siano in grado di navigare autonomamente.

Questo documento è organizzato nel modo che segue. Nella Sezione II è contenuta una rassegna di alcuni metodi di localizzazione e *tracking* basati su reti wireless. La Sezione III presenta l'organizzazione generale del lavoro svolto, evidenziando le varie problematiche affrontate. Nella Sezione IV sono riportati gli studi di connettività fatti per l'implementazione della rete wireless, mentre nella Sezione V vengono descritti i moduli hardware ed il codice sviluppato per il loro funzionamento. L'algoritmo di localizzazione è spiegato dettagliatamente in Sezione VI; infine, la Sezione VII mostra la realizzazione del *testbed* svolta in due ambienti diversi (un sotterraneo dei Laboratori dell'INFN di Legnaro e l'Auditorium del Collegio "S. Cuore" sito in Padova) e la Sezione VIII indica possibili sviluppi futuri del sistema implementato.

II. Stato dell'arte

Wireless sensor networks (WSN)

Una rete di sensori wireless *(Wireless Sensor Network, WSN)* è una rete senza fili formata da un insieme molto numeroso di dispositivi autonomi, detti *nodi* o *mote*, dotati di sensori per il monitoraggio dell'ambiente circostante. Tipicamente, ogni nodo monta anche un microcontrollore ed un *chip* radio che consente lo scambio di dati con gli altri nodi e con altri dispositivi wireless. Le peculiarità di una WSN possono essere riassunte come:

- nodi di dimensioni estremamente contenute e dalle risorse di memoria e calcolo limitate;
- interazione con l'ambiente circostante tramite sensori;
- scambio di dati mediante rete wireless;
- lunga autonomia energetica (dell'ordine di mesi o anni);
- contesti operativi eterogenei ed ostili, anche inaccessibili all'uomo;
- topologia dinamica di rete;
- funzionamento autonomo, non costantemente monitorato, e conseguente politica di gestione dei guasti.

Le WSN, nate originariamente in ambito militare per la sorveglianza del campo di battaglia, sono ora destinate agli usi più diversi; una rassegna degli attuali scenari applicativi nonchè delle caratteristiche di una WSN si può trovare in [2]. In particolare, reti di questo tipo si rivelano utili in applicazioni che richiedono la localizzazione ed il *tracking* in *real-time* di nodi mobili (distinti dai nodi àncora o *beacon*, di solito fissi, che costituiscono la rete). Le tecniche implementate a questo scopo possono essere classificate in base al mezzo fisico utilizzato per la propagazione del segnale (onde radio, impulsi sonori, infrarossi, ecc.) oppure in base a quale parametro viene valutato tra:

- *Angle Of Arrival (AOA)*, dove le informazioni sulla distanza sono ottenute da una stima degli angoli relativi del segnale trasmesso tra nodi vicini;
- *Time Of Arrival (TOA)*, in cui viene considerato il tempo di propagazione del segnale dalla sorgente alla destinazione; una stima basata sul TOA richiede una sincronizzazione dell'hardware, con conseguente aumento del costo della strumentazione. Il sistema di localizzazione *GPS (Global Positioning System)*[2] è un noto esempio di questo approccio;

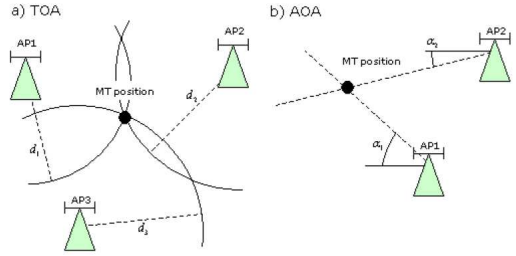


Fig. 2: Tecniche di posizionamento con i parametri TOA e AOA.

- *Time Difference Of Arrival (TDOA)*, dove vengono utilizzati segnali ad ultrasuoni per stimare la distanza tra un nodo mobile ed un nodo sorgente. Come per il TOA, anche il TDOA necessita dell'uso di hardware speciale, risultando a volte troppo costoso per le WSN;
- *Received Signal Strength Indicator (RSSI)*, in cui viene usato un modello per tradurre le informazioni sulla potenza del segnale nella distanza tra due nodi.

É possibile, inoltre, distinguere le WSN destinate alla localizzazione in base alla loro architettura, ovvero la filosofia di gestione dei nodi; l'architettura di un sistema di localizzazione ne influenza la scalabilità, l'abilità nel preservare la privacy dell'utente, la facilità nel posizionamento dei nodi e le prestazioni globali. Esistono due tipi di architettura: *active mobile* e *passive mobile*. L'architettura *active mobile*, illustrata in Fig. 3, è composta da un trasmettitore attivo che periodicamente invia dei messaggi sul canale wireless; i ricevitori disposti nell'ambiente ascoltano tali messaggi ed operano una prima stima della distanza dal nodo mobile ascoltato. Poi ogni ricevitore trasmette la stima ad una centrale operativa che stabilisce la posizione di ogni dispositivo mobile presente. Esempi di questa architettura sono i sistemi *Active Badge*, *Active Bat* e *Ubisense*, sviluppati dai Laboratori dell'AT&T di Cambridge.

Di contro, l'architettura *passive mobile*, mostrata in Fig. 4, inverte i ruoli dei dispositivi di trasmissione e di ricezione. I nodi àncora sono posizionati in locazioni note dell'ambiente e periodicamente inviano alla rete wireless la loro posizione o un loro identificativo; ogni dispositivo mobile ascolta "passivamente" queste informazioni, stima la distanza da ogni singolo nodo àncora e determina la sua posizione globale. Un esempio di questa architettura è il sistema *Cricket*, descritto in II-B.

Un'ulteriore suddivisione delle tecniche di localizzazione può essere eseguita in base alla tipologia di algoritmo utilizzato: *centralizzato* e *decentralizzato*. Nel primo caso vi è la presenza di un unico nodo centrale in grado di elaborare le informazioni provenienti dall'intera rete, mentre nel secondo

[2]Si veda `http://www.gps.gov/`; il sistema è stato commissionato dal governo degli Stati Uniti d'America (U.S.A.), che lo gestisce attualmente.

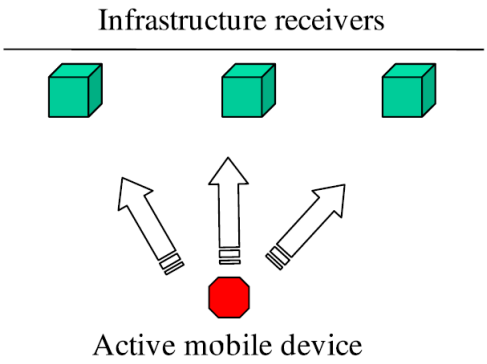


Fig. 3: Schema di principio dell'architettura *active mobile*.

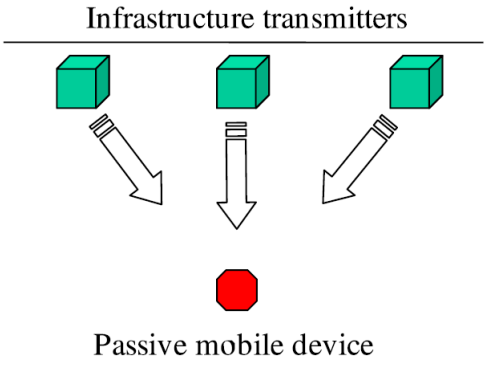


Fig. 4: Schema di principio dell'architettura *passive mobile*.

caso le mansioni di calcolo vengono svolte a livello locale nei singoli nodi della rete.

Nel seguito vengono riportati alcuni fra i più noti sistemi di localizzazione presenti in letteratura.

A. Il progetto RADAR

Il progetto RADAR [3] è uno dei primi progetti mirati alla realizzazione di un sistema di localizzazione e di *tracking* di un nodo mobile in ambiente *indoor*, basato sull'utilizzo del segnale a radiofrequenza *(Radio Frequency, RF)*. RADAR opera essenzialmente in due fasi:

1) *fase off-line*, in cui si effettua una raccolta dati che descriva la distribuzione nell'ambiente della potenza del segnale emesso dai nodi àncora *(Received Signal Strength, RSS)*;
2) *fase on-line*, in cui le misure rilevate vengono confrontate con i dati a priori ed elaborate per di determinare la posizione del nodo mobile.

La prima operazione può essere svolta in modo completamente empirico, effettuando una serie di misurazioni sul campo, oppure ci si può basare su un modello di propagazione del segnale radio. Nel secondo caso tipicamente si utilizza il *Wall Attenuation Factor model (WAF)*, in cui la potenza rilevata ad una distanza d dal trasmettitore è data da:

$$P(d)[dBm] = P(d_0)[dBm] - 10n \log\left(\frac{d}{d_0}\right) - K, \qquad (1)$$

nel quale n indica un coefficiente di attenuazione in funzione della distanza, $P(d_0)$ è la potenza del segnale ad una distanza di riferimento d_0, e K è dato da

$$K = \begin{cases} nW \cdot WAF & \text{se } nW < C \\ C \cdot WAF & \text{se } nW \geq C. \end{cases}$$

C rappresenta il massimo numero di ostruzioni (muri) in grado di far variare il fattore di attenuazione, nW è il numero di ostruzioni tra il ricevitore ed il trasmettitore, e WAF è l'attenuazione del segnale attribuibile ad un muro. I valori di nW e di WAF dipendono in modo complesso dalla struttura dell'ambiente e dai materiali di costruzione, per cui vengono di solito calcolati empiricamente.

Per la fase *on-line*, l'idea fondamentale è quella di confrontare le misure con i dati a priori, cercando la posizione che fornisce la corrispondenza migliore. È necessario allora definire uno spazio di ricerca, detto *spazio dei segnali (signal spaces)* ed una metrica: il primo è costituito dai dati raccolti nella fase *off-line*, mentre la seconda può essere ad esempio la norma euclidea. Avendo una misura, la tecnica *Nearest Neighbor(s) in Signal Space (NNSS)* calcola la distanza nello spazio dei segnali per un certo insieme di posizioni, e sceglie quella a distanza minima, ottenendo la stima della posizione del nodo mobile.

Anche in presenza di ambienti complessi in cui vi siano molte interferenze agenti sulla propagazione del segnale radio, il sistema RADAR permette di avere un notevole grado di accuratezza (lo scarto medio si aggira sui 2/3 metri), e consente indifferentemente localizzazione e l*tracking* del nodo mobile.

B. Cricket

Cricket è un sistema di localizzazione per interni concepito per integrare sistemi come il GPS nelle aree in cui la copertura non è garantita. Nella WSN implementata vengono utilizzati due tipi di segnali: segnale elettromagnetico a radiofrequenza, che si propaga alla velocità della luce (ca. $3 \cdot 10^8$ m/s), e segnale ultrasonico, che viaggia alla velocità del suono (ca. 343 m/s). I *beacon* inviano in sequenza un segnale radio (contenente l'identificativo del mittente) ed un impulso ultrasonico: misurando la differenza tra i TOA dei due segnali, il nodo mobile riesce a stimare la distanza che lo separa dal nodo àncora mittente. Ripetendo questa operazione per ogni *beacon* rilevato, il nodo mobile riesce a ricavare la sua posizione; la precisione raggiunta con questa tecnica è molto alta, con un errore dell'ordine del cm. Non è richiesta alcuna raccolta dati a priori: una volta installati e programmati i nodi, il sistema è pronto per eseguire la localizzazione.

Il progetto del sistema di localizzazione Cricket [4] si pone quattro obiettivi:

1) *scalabilità*: possibilità di aumentare facilmente l'area coperta dalla WSN senza un peggioramento apprezzabile delle prestazioni;
2) *privacy*: minima intrusione nella privacy degli utilizzatori;
3) *low cost*: utilizzo di sensori di costo ridotto, dell'ordine dei 10 $ a componente;

4) *robustezza*: mantenimento di una buona precisione, anche in caso di perdita accidentale di alcuni nodi trasmittenti;

in particolare, l'ultimo punto è soddisfatto mediante un algoritmo di tipo decentralizzato, che richiede un nodo mobile con un'adeguata potenza di calcolo.

Nonostante le buone prestazioni questa soluzione viene poco utilizzata in pratica, poichè i componenti preposti alla ricezione degli impulsi ultrasonici sono molto sensibili alle vibrazioni meccaniche, inevitabili conseguenze degli spostamenti del nodo mobile. In Fig. 5 si riportano alcuni nodi utilizzati nello sviluppo del progetto Cricket.

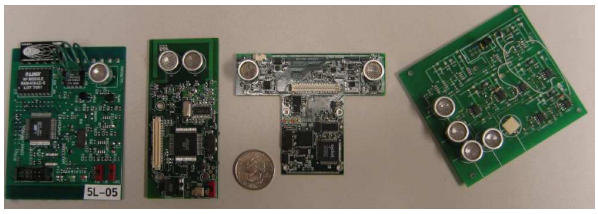


Fig. 5: Esempio di nodi utilizzati nello sviluppo del progetto Cricket.

C. MoteTrack

Come RADAR, Motetrack [5] è un sistema di localizzazione decentralizzato basato sulla misura della potenza del segnale elettromagnetico emesso da alcuni nodi àncora. Per determinare la posizione di un nodo mobile T il sistema utilizza una mappa, costruita tramite una raccolta di misure svolte direttamente sul campo, che associa ad ogni punto dell'ambiente un set di valori di potenza del campo elettromagnetico rilevato da un nodo mobile. L'insieme dei valori di RSS rilevati da un nodo mobile, assieme all'identificativo (ID) del nodo àncora che ha inviato il segnale radio a cui si riferiscono i dati, è detta *signature*. L'insieme delle *signature* e delle coordinate spaziali in cui vengono raccolte (ovvero le mappe a priori) è chiamato *reference signature*.

Durante la fase di localizzazione, un nodo mobile M, dopo aver raccolto una *signature*, utilizza le *reference signature* dei nodi àncora per stimare la propria posizione: questa è determinata attraverso una media pesata delle distanze dalle coordinate contenute nelle *reference signature* utilizzate, dove i pesi utilizzati nella media sono proporzionali alle distanze fra le *signature* rilevate dal nodo M e quelle contenute nelle *reference signature*. La tecnica di localizzazione del progetto MoteTrack presuppone quindi che due *signature* siano tanto più simili fra loro quanto più vicine sono le coordinate in cui queste sono state raccolte. Viene riportato in Fig. 6 un esempio di raccolta di una *signature* da parte di un nodo mobile M e l'invio di alcune reference signature da parte dei nodi àncora.

Lo scopo per cui è nato il progetto MoteTrack era la realizzazione di un sistema di localizzazione robusto, in grado di mantenere buone prestazioni anche nel caso di perdita di alcuni nodi àncora e di variazioni dell'ambiente con conseguente alterazione della potenza de. Per aumentare la robustezza,

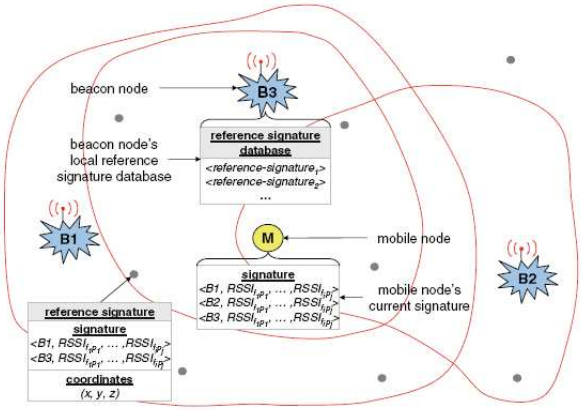


Fig. 6: Esempio di raccolta di *signature* di un nodo mobile M.

il MoteTrack adotta le seguenti 3 soluzioni, differenti da RADAR:

1) un approccio decentralizzato, in cui la localizzazione viene effettuata dai *beacon* senza essere completamente affidata ad un solo dispositivo (il nodo mobile o l'elaboratore ad esso collegato);
2) le *reference signature* sono memorizzate nei nodi àncora in modo da minimizzare le ripetizioni nel *database* completo ma garantendo il funzionamento del sistema anche in caso di danni;
3) le variazioni delle *signature* rispetto alle *reference signature* raccolte, dovute a mutamenti dell'ambiente o al malfunzionamento di alcuni nodi trasmettitori, sono compensate mediante la scelta di una metrica adattativa: eventuali differenze fra *beacon* rilevati e *beacon* memorizzati nella corrispondente *reference signature* non vengono ignorate ma aggiungono un opportuno contributo al calcolo della distanza, diverso (adattativo, appunto) a seconda del numero di nodi àncora danneggiati.

Nonostante il progetto MoteTrack sia uno dei pochi sistemi di localizzazione basato sull'RSS che sia considerato affidabile e robusto, il fatto che esso richieda una lunga fase di *setup* per la raccolta delle *reference signature* ne limita la diffusione.

D. RSS-based localization with particle filters

Analogamente a RADAR, questo algoritmo di localizzazione, di tipo centralizzato, si basa sulle informazioni a priori relative alla topologia della WSN e su un modello log-normale dell'RSS [6]. L'idea innovativa consiste nel modellizzare il movimento $\mathbf{x}_t$ del nodo mobile come un processo markoviano omogeneo del primo ordine,

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_t,$$

dove $\mathbf{v}_t$ rappresenta il processo di guida: questo consente di implementare un filtraggio bayesiano per diminuire il problema della *false localization* che può sorgere dallo stimare staticamente la posizione minimizzando istante per istante la distanza tra dati misurati e dati memorizzati.

In pratica la realizzazione di questo metodo si articola in due fasi:

1) *Calibrazione off-line*: il nodo mobile (NM) viene posto in M differenti posizioni: in ognuna vengono calcolati media e varianza dell'RSSI relativo al segnale ricevuto da ciascun nodo àncora (NA); viene poi operata un'interpolazione delle misure effettuate per ricavare delle mappe bidimensionali *(mappe a priori)* relative all'intero ambiente;
2) *Tracking on-line*: ad ogni istante temporale t ogni NA misura il valore di RSS del segnale ricevuto dal nodo mobile; una tecnica di *Sequential Importance Resampling (SIR) Particle Filtering (PF)* confronta le misure con le mappe a priori e fornisce la stima della posizione del nodo mobile.

Il funzionamento suppone che la potenza di trasmissione sia la stessa per i NA e per il NM.

Allo stato attuale, questo algoritmo è stato valutato soltanto in simulazione, fornendo una precisione nella stima dell'ordine di 0.5 m.

E. Stima della distanza basata sulla potenza del segnale ricevuto

L'utilizzo dell'RSSI è uno dei metodi più comuni utilizzati negli algoritmi di localizzazione presenti in letteratura: nonostante una forte variabilità, deve il suo successo alla semplicità ed all'economicità implementativa.

É possibile affermare come, in una fase di propagazione di un segnale, l'energia di quest'ultimo diminuisca in base alla distanza percorsa. Se al ricevitore è nota la potenza con la quale è stato trasmesso il segnale, è possibile ottenere una stima della distanza tra trasmettitore e ricevitore. Il modello e la tecnica utilizzati allo scopo sono relativamente semplici, ma richiedono la determinazione di diversi parametri; generalmente, il modello di propagazione del segnale che si utilizza è il seguente modello log-normale

$$\text{RSSI}(d) = P_T - PL(d_0) - 10\eta \log_{10} \frac{d}{d_0} + X_\sigma \qquad (2)$$

I termini che compaiono in (2) corrispondono a:

- RSSI(d), livello di RSSI ricevuto ad una distanza d dal trasmettitore [dB];
- P_T, potenza di trasmissione [dB];
- $PL(d_0)$, perdita di potenza in ricezione ad una distanza di riferimento d_0;
- η, fattore di decrescenza della potenza del campo elettromagnetico;
- X_σ, variabile aleatoria gaussiana di media nulla e varianza σ^2, in grado di modellizzare le variazioni casuali del valore di RSSI.

1) Variabilità dell'RSSI: In aggiunta ai vari fenomeneni di propagazione del campo elettromagnetico descritti in IV-B, la misura della potenza del segnale è afflitta anche dai seguenti fattori:

- *Variabilità del trasmettitore*: differenti trasmettitori si comportano in maniera differente anche se sono configurati allo stesso modo. In pratica, questo significa che quando un trasmettitore è impostato per inviare pacchetti ad un livello di potenza *d* [dBm] si avrà che il trasmettitore invierà questi pacchetti con un livello di potenza "molto vicino" a *d* [dBm], ma non necessariamente uguale a *d* [dBm]. Questo può alterare l'RSSI e può portare ad una inesattezza nella stima della posizione.
- *Variabilità del ricevitore*: la sensibilità dei ricevitori montati su differenti chip radio è differente. In pratica, questo porta ad avere diversi valori di RSSI memorizzati in ricevitori diversi, anche se tutti i parametri che influenzano l'RSSI sono mantenuti costanti.
- *Orientazione dell'antenna*: ogni antenna ha un proprio diagramma di radiazione, ed esso non è perfettamente uniforme. In pratica, questo significa che il valore di RSSI memorizzato in un ricevitore, dalla comunicazione con un nodo posto ad una certa distanza, varia a seconda dell'orientazione dell'antenna del trasmettitore e del ricevitore.

Si noti che questo metodo fornisce indicazioni soltanto sulla determinazione della distanza dal nodo trasmettitore; per la stima della posizione devono essere implementate altre tecniche.

F. Tecniche di triangolazione

Conoscendo il valore di potenza del segnale ricevuto dal nodo mobile al variare della distanza dei nodi àncora, la posizione può essere calcolata in modo geometrico mediante una tecnica di triangolazione [7]. Tenendo conto degli errori di stima delle distanze, il problema viene allora risolto adottando un sistema di equazioni sovraparametrizzato del tipo:

$$\begin{cases} (x_1 - T_x)^2 + (y_1 - T_y)^2 = r_1^2 \\ (x_2 - T_x)^2 + (y_2 - T_y)^2 = r_2^2 \\ \qquad\qquad \vdots \\ (x_n - T_x)^2 + (y_n - T_y)^2 = r_n^2, \end{cases}$$

dove $(T_x, T_y) = T$) sono le coordinate del nodo mobile T, (x_i,y_i) sono le coordinate del nodo àncora i-esimo ed r_i è la stima della distanza fra il nodo T ed il nodo àncora i-esimo, calcolata in base all'RSS. É possibile riportare il sistema di equazioni considerato in un sistema lineare nelle incognite T_x e T_y, ottenendo

$$AT = b$$

allora il vettore $\mathbf{T}$ si trova da

$$\mathbf{T} = \arg\min_T \|\mathbf{A}T - \mathbf{b}\| = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}.$$

Si noti che $\mathbf{T}$ è il vettore in grado di minimizzare la distanza fra il vettore $u = \mathbf{AT}$ ed il vettore $\mathbf{B}$, e quindi $\mathbf{T}$ è la proiezione ortogonale di $\mathbf{b}$ sul sottospazio $\mathcal{A}$ generato dalle colonne di $\mathbf{A}$.

Anche tenendo conto della rumorosità della stima della distanza, le prestazioni di questo metodo degradano fortemente, riportando notevoli errori di stima, soprattutto quando il nodo mobile è posizionato nelle vicinanze del bordo dell'area sottoposta al controllo. Questo algoritmo è quindi ancora poco accurato richiede un'ulteriore fase di studio.

In linea generale, il processo di localizzazione per via geometrica di un nodo può essere suddiviso in due passi: nel primo un nodo stima la propria distanza dagli altri nodi nelle vicinanze sfruttando una o più caratteristiche del segnale ricevuto; nel secondo, il nodo utilizza tutte le distanze stimate per calcolare la sua posizione attuale. Questo calcolo può essere svolto in tre modi diversi:

- *Triangolazione*: consiste nella raccolta di misure del parametro AOA al sensore da almeno 3 sorgenti. Utilizzando il riferimento di AOA la localizzazione viene effettuata applicando semplici proprietà di natura geometrica. Un gran numero di algoritmi di localizzazione viene incluso in questa classe;
- *Trilaterazione*: utilizza un insieme di terne del tipo (x, y, d), dove d rappresenta una stima della distanza tra la sorgente posta in (x, y) ed il nodo sensore. Per determinare in maniera accurata e univoca la posizione relativa di un punto nel piano 2D utilizzando la trilaterazione sono necessari almeno 3 punti di riferimento.
- *Multilaterazione*: sfrutta le intersezioni di diverse circonferenze (che modellizzano il segnale emesso dai nodi àncora) sulla base del parametro TDOA; il segnale inviato da un nodo trasmettitore a tre o più ricevitori deve pertanto essere accuratamente sincronizzato. Quando vengono utilizzati N ricevitori, la presenza di $N-1$ circonferenze permette di individuare univocamente un punto nello spazio 3D. Quando invece vengono utilizzati $N > 4$ ricevitori, il problema può essere visto come un problema di ottimizzazione utilizzando, per esempio, un metodo di risoluzione ai minimi quadrati.

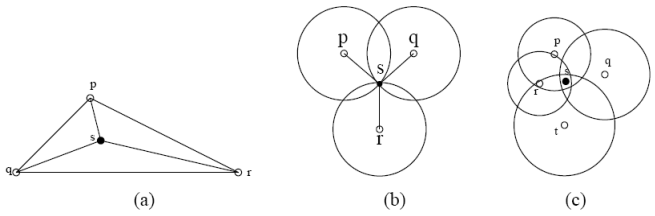


Fig. 7: (a) Triangolazione (b) Trilaterazione (c) Multilaterazione

G. Bounding Box algorithm

L'algoritmo Bounding Box [8] utilizza un approccio di tipo geometrico che sfrutta pesantemente informazioni sulla connettività della rete. Una prima implementazione modellizza il campo di copertura di un *beacon* con una *bounding box* (letteralmente, scatola) quadrata, di lato pari al doppio della distanza coperta dal segnale *(beaconing range, br)*. Note le posizioni (x_B^i, y_B^i) di ciascun nodo àncora, allora le coordinate del nodo mobile ubbidiscono alle relazioni

$$x_N \in \left[x_B^i - br^i;\ x_B^i + br^i\right]; \quad y_N \in \left[y_B^i - br^i;\ y_B^i + br^i\right];$$

per ogni nodo ancora i rilevato. Quindi la posizione del nodo mobile si trova nell'intersezione delle *bounding box*: questo insieme è anch'esso una "scatola", rettangolare, le cui dimensioni diminuiscono all'aumentare dei *beacon* processati. Infine, la posizione stimata per il nodo mobile è data dal centro della regione di intersezione, come mostrato in Fig. 8.

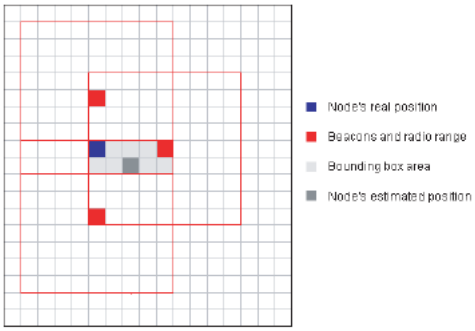


Fig. 8: Principio della tecnica *Bounding Box*.

É possibile migliorare questa implementazione utilizzando al posto come *bounding box* dei cerchi anzichè dei quadrati, ovvero servendosi di un modello più realistico di copertura del segnale radio emesso dai nodi àncora.

Il problema di questo approccio, comune a tutti gli algoritmi che utilizzano metodi geometrici per risolvere il problema della localizzazione, è il richiedere un elevato numero di nodi àncora, che lo rende poco indicato in progetti che hanno degli imprescindibili vincoli di costo.

III. Approntamento del progetto

L'obiettivo che viene raggiunto da questo progetto è la costruzione di uno studio preliminare di fattibilità per l'implementazione di un algoritmo di localizzazione basato su una rete wireless in un ambiente ad alta rumorosità, in grado di identificare in *real-time* la posizione di un operatore opportunamente dotato di un nodo mobile con una precisione inferiore ai 5 m. Le problematiche da risolvere sono quindi diverse, sia di ordine teorico che di ordine pratico, e sono raggruppabili in tre macro-aree:

1) studio dell'ambiente destinato al *testbed* e delle risorse a disposizione; posizionamento dei nodi della WSN in modo opportuno (IV);
2) studio di un algoritmo di localizzazione e sua implementazione (VI);
3) sviluppo del codice necessario all'interazione fra i componenti della WSN e l'elaboratore, per garantire un funzionamento in tempo reale dell'algoritmo (V).

I tre punti, in realtà, non sono distinti ma procedono in modo parallelo, condizionandosi reciprocamente: ad esempio, la scelta della posizione dei mote àncora è determinata dalla scelta dell'algoritmo, che impone un certo tipo di copertura dell'area interessata, così come un funzionamento dell'algoritmo che rispetti le specifiche fissa dei limiti alle soluzioni implementative adottate.

A. Considerazioni preliminari

Nell'implementazione della WSN ci si è orientati verso la minimizzazione delle risorse impiegate: una rete wireless

composta di pochi nodi comporta sforzi minori in termini di consumo energetico e di manutenzione; inoltre la quantità di dati circolante rimane contenuta. Tutto ciò è a favore della scalabilità del progetto, dato che risulta possibile estendere l'area di copertura della WSN mantenendo accessibili i costi e con modifiche minime al codice appositamente sviluppato.

Nello studio dell'algoritmo di localizzazione, è stato necessario decidere preliminarmente se adottare un approccio di calcolo centralizzato oppure distribuito: nel primo caso si tratta di convogliare tutti i dati della WSN ad un elaboratore esterno, che svolge le operazioni richieste per il calcolo della posizione dei nodi mobili e si preoccupa di restituire questo valore al nodo mobile; nel secondo caso, invece, tutti i nodi della rete convergono alla conoscenza della posizione, poi inviata all'esterno per il monitoraggio. Un approccio centralizzato consente l'impiego di risorse hardware con potenza di calcolo pressochè nulla e minimo dispendio energetico; un'architettura distribuita si rivela più efficiente ma richiede l'utilizzo di *smart sensor*. In realtà, poichè ogni operatore sarà dotato di un nodo mobile interfacciato ad un palmare, si può pensare di sfruttare questa capacità di elaborazione (sicuramente maggiore rispetto a quella dei mote) per i calcoli necessari alla localizzazione; l'approccio adottato è quindi centralizzato e prevede che il nodo mobile raccolga i dati dalla WSN per poi inoltrarli all'elaboratore (che, in questa fase di *testbed*, è rappresentato da un normale *laptop*): questo provvederà poi a restituire la posizione stimata in modo che essa venga inviata ad un elaboratore esterno alla rete.

Infine, lo sviluppo del codice per il funzionamento dell'algoritmo richiede di affrontare diverse problematiche, che riguardano la programmazione di ogni singolo agente ma anche la negoziazione delle interazioni tra i vari agenti. Particolarmente critica la parte relativa allo scambio di dati fra mote e pc, in cui è previsto che il nodo mobile inoltri i dati ricevuti dalla WSN, il pc li elabori e restituisca la stima della posizione al nodo. Si presentano a questo punto due alternative: tradurre l'intero algoritmo in un linguaggio di programmazione (ad esempio C) oppure utilizzare strumenti come MATLAB™, sicuramente non ottimizzati ma molto duttili e potenti. La prima scelta, se da un lato facilita la comunicazione poichè agisce a basso livello ed aumenta il controllo del programmatore, richiede la costruzione *in toto* dei componenti necessari al funzionamento, mentre la seconda implica l'esatto contrario. Si è optato per la seconda soluzione per due motivi: la possibilità di servirsi di buoni strumenti grafici e la constatazione che la fase di ingegnerizzazione che dovrà seguire questo *testbed* comporterà inevitabilmente una riscrittura del codice.

B. Algoritmo implementato: BOZ algorithm

Gli sforzi congiunti hanno portato alla realizzazione di un algoritmo di localizzazione, denominato *BOZ algorithm* dal nome degli Autori, che è un'evoluzione dell'algoritmo descritto nella II-D: esso prevede innanzitutto la costruzione, per ogni mote àncora, di mappe contenenti media e varianza della potenza del segnale ricevuto in un certo insieme di posizioni, cui si aggiunge la probabilità di ricezione di pacchetti; queste mappe vengono poi interpolate in modo che una densità maggiore renda più precisa la successiva localizzazione. Il comportamento dell'operatore è poi modellizzato con un'opportuna passeggiata aleatoria; questo consente di ottenere la stima della sua posizione mediante un filtraggio di tipo particellare con perdita di pacchetto, in cui il peso delle particelle è determinato tenendo conto non solo dalla distanza fra misura e valore atteso (ricavato dalla mappa), ma anche dell'affidabilità dei pacchetti ricevuti. I risultati ottenuti mostrano che le scelte fatte sono particolarmente adatte all'ambiente del *testbed*, garantendo la localizzazione con un errore medio di 3 m: in generale, l'approccio adottato si dimostra efficiente nel momento in cui ogni posizione è caratterizzata il più univocamente possibile da una combinazione (nodi àncora ricevuti-RSSI rilevati), in modo che risulti facile associare una misura ricevuta ad un particolare punto dell'ambiente. Prove in ambienti profondamente diversi, mostrate in VII, rivelano come le prestazioni del *BOZ algorithm* peggiorino quanto più uniformi sono le mappe dell'RSS: è allora sufficiente impostare opportunamente i parametri della WSN (quali numero di nodi, posizione, potenza di trasmissione) per avere garanzia del buon funzionamento dell'algoritmo.

Nelle Sezioni che seguono è documentato in dettaglio tutto il lavoro svolto, motivando ogni singola scelta e descrivendone il significato. Il fatto che i contenuti siano presentati in modo sequenziale risponde semplicemente ad un'esigenza di stesura: in realtà la dimensione di indagine ed implementazione è stata piuttosto parallela che seriale, poichè alcune scelte in corso d'opera hanno reso più volte necessario un adattamento dell'intero lavoro. Una documentazione di tipo cronologico, tuttavia, avrebbe reso ostica la comprensione della relazione, e si è scelto di esporre i contenuti per area tematica.

IV. IMPLEMENTAZIONE DELLA RETE WIRELESS

A. Studio dell'ambiente ed analisi di connettività

La particolarità dell'ambiente di lavoro obbliga ad uno studio preliminare del comportamento del segnale radio, sottoposto a fenomeni di diversa natura (rifrazione, ostruzione, ecc.), accentuati alla presenza di numerosi ostacoli come pilastri e recinzioni metalliche; la foto in Fig. 9 dà un'idea dei sotterranei dell'INFN, dove verrà installata la rete wireless per il sistema di localizzazione.

Fig. 9: Scorcio dei sotterranei dei laboratori dell'INFN.

In Fig. 10 viene riportata la mappa 2D relativa ai sotterranei dei laboratori: la larghezza totale dell'ambiente è pari a 37 m e la lunghezza pari a 31, per un'area totale di 1150 mq circa. L'altezza del soffitto misura mediamente 3.10 m, ma è da evidenziare la presenza di numerose tubature idrauliche ed elettriche che riducono l'effettiva altezza di lavoro di circa 2 m.

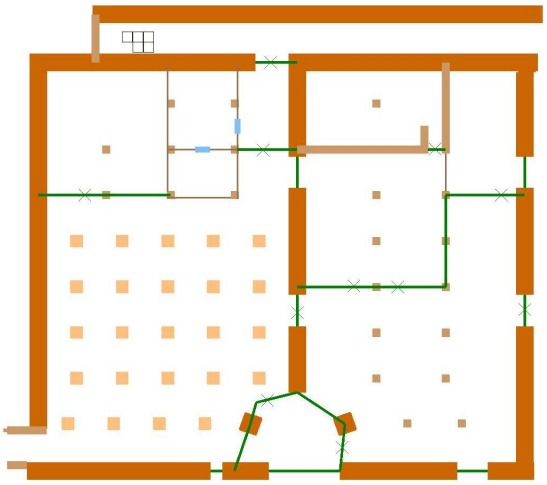


Fig. 10: Mappa dei laboratori sotterranei dell'INFN - scala 1:210.

Si nota che i muri principali, in cemento armato, hanno un notevole spessore, pari a 1.4 m: si rileverà come essi siano in grado di impedire il passaggio delle onde elettromagnetiche generate da un nodo àncora con il livello di potenza considerato negli esperimenti. In questi sotterranei vi sono inoltre numerosi gruppi elettrici di continuità e strumentazione elettronica che possono interferire in maniera non trascurabile nella propagazione delle onde radio emesse dai mote àncora.

Un primo sopralluogo nei sotterranei dell'INFN ha permesso un'analisi sommaria sulla reale capacità di propagazione del segnale radio nel particolare ambiente, con lo scopo di disegnare una copertura ottimale dell'intera area. Impostando un solo nodo come trasmettitore e disponendo alcuni mote nell'ambiente si è notato come in assenza di ostacoli (nei due corridoi paralleli) la distanza percorsa dal segnale possa arrivare ai 30 m. Quando però un nodo mobile viene spostato in una zona irta di ostacoli, per esempio i pilastri, il segnale trasmesso o non viene captato dal ricevitore, oppure lo raggiunge solamente grazie a fenomeni di riflessione delle onde elettromagnetiche: tale comportamento sottolinea l'esigenza di un adeguata modellizzazione stocastica. Queste prime prove sono state effettuate con l'ausilio dell'applicazione `Trawler-Moteiv` (vedi Fig. 11), in grado di indicare il parametro *Link Quality Indicator (LQI)* dei collegamenti che si instaurano tra i nodi in fase di comunicazione: il valore LQI è tanto più basso quanto più stabile è il collegamento tra i nodi.

Sebbene il valore di LQI dipenda dalla quantità di rumore presente nel canale nella banda in cui avviene la trasmissione, non lo si considera di grande utilità a causa del suo comportamento molto variabile con la distanza e con le condizioni ambientali.

Si evidenzia il fatto che, per una ricezione ottimale, i nodi

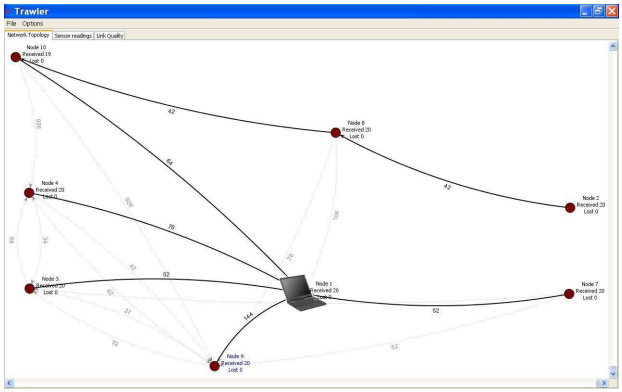


Fig. 11: Esempio di utilizzo del Trawler.

trasmettitore e ricevitore dovrebbero trovarsi alla medesima altezza, in quanto (vedi [9]) una piccola variazione nell'altezza dei sensori da terra può portare forti conseguenze sulla potenza del segnale ricevuto. Inoltre, si dovrebbero mantenere tutti i nodi in una posizione più alta possibile da terra, sfruttando il fatto che, normalmente, le ostruzioni di natura variabile come la presenza umana diminuiscono all'aumentare dell'altezza. In pratica però, date la finalità del progetto e le reali condizioni dell'ambiente, i nodi fissi sono posti ad una altezza di 2 m da terra ed il nodo mobile viene supposto ad un'altezza di 1 m circa; il particolare algoritmo scelto permetterà di non considerare la differente altezza di NM e NA.

B. Propagazione del campo elettromagnetico

La propagazione delle onde elettromagnetiche (sia all'aperto che, soprattutto, negli ambienti chiusi) è soggetta a molteplici fenomeni e risulta quindi di difficile modellizzazione. In particolare, la velocità di propagazione delle onde dipende dalle proprietà fisiche del mezzo in cui avviene la propagazione: ci si deve quindi aspettare che nel passaggio di un'onda da un mezzo ad un altro la velocità di propagazione cambi.

I fenomeni più evidenti che intervengono durante la propagazione di un'onda elettromagnetica risultano essere: *riflessione*, *diffrazione* e *scattering*, oltre all'*interferenza*. Essi si verificano in misura maggiore o minore al variare del rapporto fra la dimensione della lunghezza dell'onda del campo elettromagnetico e le dimensioni dell'oggetto su cui il campo impatta. Il protocollo di comunicazione utilizzato dai nodi impone che la frequenza del segnale trasmesso sia compresa nell'intervallo [2400,2483.5] MHz e quindi la lunghezza d'onda del campo elettromagnetico generato è compresa fra 12.08 cm e 12.5 cm. Nell'ambiente di interesse esistono numerosi oggetti aventi dimensioni maggiori o paragonabili alla lunghezza d'onda del campo elettromagnetico considerato e quindi è possibile osservare tutti i fenomeni sopra citati.

1) Riflessione: Quando l'onda raggiunge un oggetto riflettente con un angolo i rispetto alla normale, essa modifica la propria traiettoria tornando indietro secondo un angolo $r = -i$. Un esempio di onda riflessa ed incidente è riportata in Fig. 12.

É bene fare una precisazione riguardante le caratteristiche della superficie su cui avviene la riflessione. Se la superficie

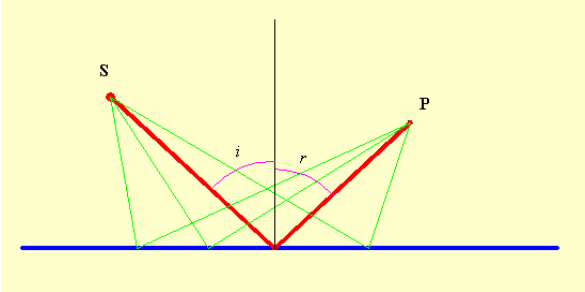


Fig. 12: Riflessione di un'onda su una superficie totalmente riflettente.

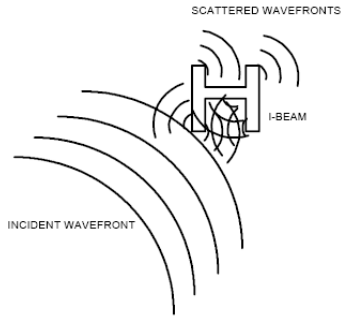


Fig. 14: Esempio del fenomeno di *scattering* su di una trave.

è liscia, un fascio di raggi paralleli che incide sulla superficie dopo la riflessione si presenterà ancora come un fascio di raggi paralleli; se invece la superficie è scabra i raggi saranno riflessi in varie direzioni, per cui i raggi riflessi avranno perso la loro caratteristica di direzionalità.

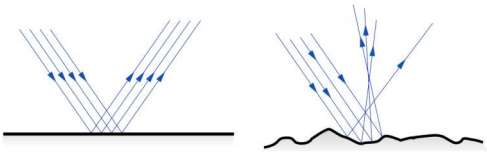


Fig. 13: Rifrazione su superficie liscia e su superficie scabra.

2) Diffrazione: La diffrazione è un particolare fenomeno di interferenza che si verifica quando un'onda incontra nel suo percorso un ostacolo o un'apertura. Ad esempio, l'apertura può essere costituita da un foro circolare o rettangolare praticato in uno schermo assorbente per le onde in esame, un ostacolo da un filo, un disco assorbente o più in generale da un qualsiasi oggetto. Qualitativamente, nello spazio oltre l'ostacolo o l'apertura le onde si propagano anche lungo direzioni *diverse* da quella di incidenza e si generano differenze di percorso tra onde che si sovrappongono in un dato punto; possono quindi avvenire fenomeni di interferenza con conseguente ridistribuzione dell'energia nei punti dello spazio. Gli effetti della diffrazione sono di norma tanto più vistosi quanto più le dimensioni dell'apertura o dell'ostacolo sono vicine al valore della lunghezza d'onda delle onde incidenti.

Il fenomeno della diffrazione è anche detto *shadowing* in quanto consente la trasmissione del campo elettromagnetico fra trasmettitore e ricevitore anche quando fra i due manca un cammino diretto (*Line Of Sight, LOS*) a causa della presenza dei corpi impenetrabili dall'onda.

3) Scattering: Il fenomeno dello *scattering* è dovuto alla natura corpuscolare del campo elettromagnetico e si manifesta in presenza di oggetti impenetrabili aventi dimensioni paragonabili a quelli della lunghezza dell'onda. In questo caso l'energia dell'onda che insiste su tali oggetti viene disseminata (dall'inglese *scatter*) in moltissime direzioni che nei casi reali sono impredicibili e dipendono dalla lunghezza d'onda del campo incidente (vedi Fig. 14).

4) Interferenza: Il termine interferenza è riferito propriamente a quei fenomeni di sovrapposizione ottenuti con onde emesse da due o più sorgenti coerenti[3]. La possibilità di produrre interferenza è una caratteristica generale delle grandezze che si propagano per onde; ne è conferma il fatto che l'interferenza si verifica con ogni tipo di onda e che la trattazione analitica è indipendente dalla natura delle onde in esame. Anzi, l'interferenza è considerata una proprietà così caratteristica delle onde che la sua osservazione viene presa come prova definitiva della natura ondulatoria di un grandezza.

Un esempio di *diffrazione* e di *interferenza* viene visualizzato in Fig. 15.

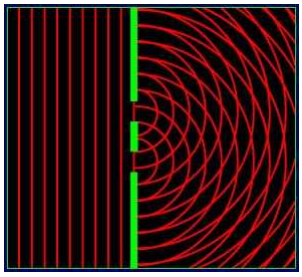


Fig. 15: Esempio di diffrazione e di interferenza.

I fenomeni ora introdotti causano la ricezione, al ricevitore, di innumerevoli onde elettromagnetiche provenienti da molteplici direzioni. Ricordando che tali fenomeni dipendono fortemente dalla lunghezza d'onda del segnale trasmesso, si può osservare come, al variare della frequenza di trasmissione, il ricevitore possa rilevare variazioni nella potenza del segnale elettromagnetico anche se il mezzo in cui si propaga il segnale resta costante. In maniera analoga, supponendo di mantenere costante la frequenza di trasmissione e di spostare il ricevitore a distanze dell'ordine di mezza lunghezza d'onda è possibile registrare notevoli variazioni nella potenza del campo elettromagnetico.

C. Simulazione con RPS

Il software RPS[4] consente di procedere con alcune simulazioni sulla propagazione del segnale trasmesso dai nodi àncora nell'ambiente, riprodotto in maniera sufficientemente accurata.

[3]Sono coerenti le sorgenti che presentano una differenza di fase fra due onde emesse costante nel tempo, in qualsiasi punto dello spazio.

[4]RPS Version 5.3 Build (440). Copyright(C) 1997-2005 Radioplan GmbH - email: `support@radioplan.com` - Internet: `http://www.radioplan.com`

Tale programma utilizza un algoritmo di *ray-tracing*, una tecnica generale di geometria ottica che si basa sul calcolo del percorso fatto dalla luce, seguendone i raggi attraverso l'interazione con le superfici [11].

Assumendo che il campo elettromagnetico sia trasmesso da un solo nodo sorgente, la potenza ricevuta in un qualsiasi punto dello spazio può essere calcolata utilizzando la nota *formula di Friis* per lo spazio libero [12]:

$$P_{r\,[dB]} = P_{t\,[dB]} - 32.5 - 20\log\left(f_{[GHz]}\right) - \\ - 20\log\left(d_{[m]}\right) + G_{t\,[dB]} + G_{r\,[dB]}.$$

In aggiunta all'attenuazione nello spazio libero, le onde elettromagnetiche vengono riflesse dagli ostacoli, attenuate a causa della penetrazione nei muri o nelle finestre e sottoposte al fenomeno della diffrazione in corrispondenza dei bordi. L'attenuazione risultante dai fenomeni di riflessione e di penetrazione può essere determinata attraverso l'angolo di incidenza, la polarizzazione, le caratteristiche dei materiali e lo spessore degli ostacoli; i bordi di diffrazione sono confrontabili con le linee sorgenti che definiscono il punto uscente di nuovi raggi.

Il particolare algoritmo utilizzato in RPS è stato il *2.5D Ray Tracing Algorithm*, il quale considera solamente i raggi riflessi/trasmessi da ostacoli in posizione verticale: nel particolare caso di ambienti *indoor* questa restrizione è più che accettabile. Da un confronto con gli algoritmi operanti in 3D, l'algoritmo 2.5D permette inoltre una riduzione drastica dello sforzo di calcolo [13].

Come primo passo nell'analisi della propagazione del segnale trasmesso da un nodo àncora, si guarda il percorso LOS, che è il modo più semplice e più facile per capire quali traiettorie percorre il segnale da un'antenna all'altra. La propagazione LOS richiede che le antenne del trasmettitore e del ricevitore siano visibili tra loro, senza la presenza di alcuna ostruzione. Un esempio è riportato in Fig. 16.

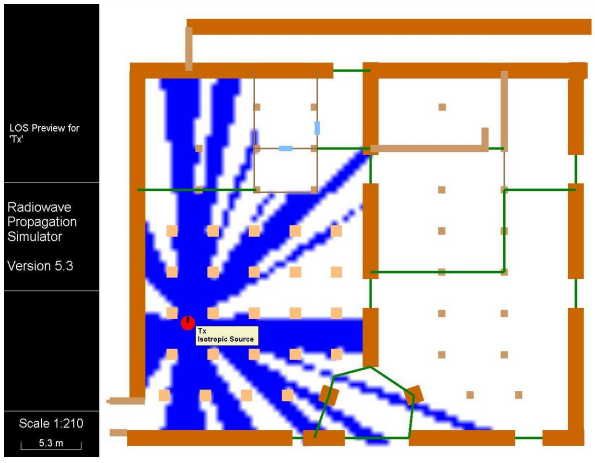


Fig. 16: Esempio di percorso LOS di un nodo trasmettitore.

Ripetendo la stessa procedura con altri nodi trasmittenti disseminati nell'ambiente si ricavano delle informazioni preliminari riguardo la propagazione del segnale all'interno dei sotterranei; poichè, però, nel particolare ambiente in esame le onde elettromagnetiche sono influenzate da molteplici disturbi, è necessario condurre una simulazione più accurata.

Si crea una griglia di ricevitori con passo di 0.5 m posta ad un'altezza di 1 m da terra: questi due valori corrispondono rispettivamente al passo di interpolazione che verrà considerato nella costruzione della mappa reale ed all'altezza alla quale si supporrà posizionato il nodo mobile soggetto al *tracking*. Le impostazioni per i nodi trasmettitori sono essenzialmente:

- antenna di tipo omnidirezionale (*isotropic source*), vedi Fig. 17;
- potenza di trasmissione di 0 dBm, pari a 1.0 mW[5];
- frequenza della portante, $f = 2.4$ GHz.

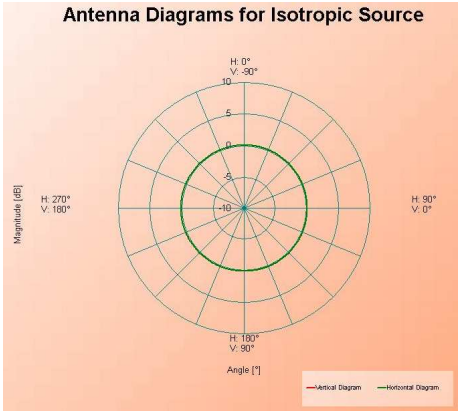


Fig. 17: Diagramma di radiazione di un'antenna isotropica (omnidirezionale).

Un primo esempio di simulazione con un solo trasmettitore inserito nell'ambiente è riportato in Fig. 18. In ogni punto della griglia è possibile ricavare la potenza ricevuta da un nodo ricevitore: si nota che i pilastri ed i divisori presenti attenuano od addirittura bloccano il segnale elettromagnetico emesso dal nodo trasmittente.

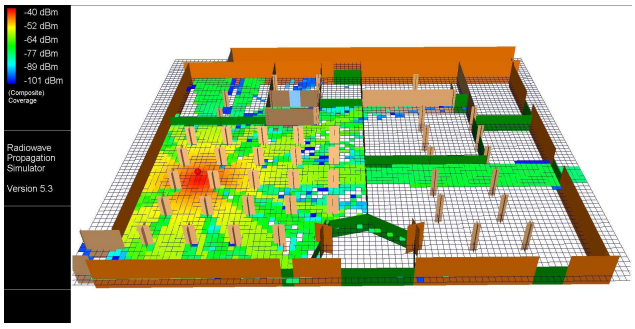


Fig. 18: Simulazione di un nodo àncora trasmettitore.

Di notevole interesse l'osservazione delle traiettorie percorse dal campo elettromagnetico, visualizzate in Fig. 19. Anche se non visualizzato per motivi grafici, si osserva che il soffitto causa di frequente la riflessione del segnale radio.

Si cerca ora di determinare un numero e successivamente una disposizione ottimali dei nodi àncora in grado di offrire

[5]Per esprimere una arbitraria potenza P[mW] come x[dBm] o viceversa, sono valide le relazioni:

$$x = 10 \cdot \log_{10}(P), \quad P = 10^{(x/10)}.$$

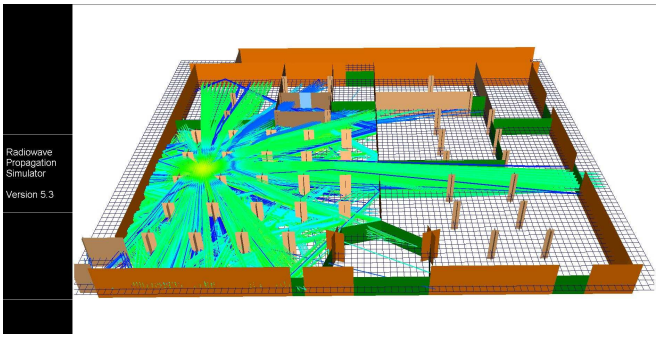


Fig. 19: Simulazione di un nodo àncora trasmettitore.

una adeguata copertura di rete: entrambe queste due richieste dipendono pesantemente dal tipo di algoritmo implementato. Esso, sfruttando una mappa di rilevazioni fatte *a priori*, necessita di un numero di *beacon* in grado di assicurare una copertura dell'area sotto esame senza specificare un numero minimo di nodi che devono essere rilevati in ogni posizione. Considerando i risultati ottenuti dal primo sopralluogo, in cui si è notato come il segnale riuscisse a propagarsi per buone distanze (mediamente il campo era circolare con un raggio di 20 m)ed impostando il problema da un punto di vista strettamente geometrico tralasciando l'effetto degli ostacoli, si potrebbe coprire l'intera area con un numero veramente esiguo di nodi trasmettitori dato da

$$\text{n° nodiTx} = \frac{\text{area totale[mq]}}{\text{area nodoTx[mq]}} \simeq 4$$

dove si è ipotizzato che un nodo àncora riesca a far giungere il proprio segnale per un'area circolare con raggio di 10 m. Tuttavia, analizzando con maggior dettaglio il particolare ambiente, si può osservare come alcune aree (i.e. il corridoio-nord e le due stanzette) siano "isolate" rispetto al resto: è quindi realistico supporre che il segnale non riesca a propagarsi all'esterno di esse. Decidendo di voler ricevere in ogni punto dell'area totale (con l'esclusione delle micro-aree citate come "isolate") almeno 4 nodi trasmettitori, per assicurare una buona accuratezza ai calcoli implementati dall'algoritmo di localizzazione, il numero di nodi àncora sufficienti a garantire una buona copertura della rete sale a 12 unità.

Per la disposizione ci si affida in buona parte ai risultati ottenuti in fase di simulazione con il software RPS; per avere in ogni punto una buona copertura una serie di tentativi ha portato alla disposizione di Fig. 20 le cui coordinate esatte sono specificate in Tab. 1. Si noti come il sistema di riferimento fissi l'origine degli assi nell'angolo in basso a sinistra.

La disposizione fissa 5 nodi àncora rispettivamente per le due aree maggiori e 2 nodi trasmettitori per il solo corridoio; in una successiva fase di ottimizzazione che voglia ridurre il numero totale dei nodi si può pensare di eliminare un nodo per stanza, mentre non è pensabile di rimuovere alcun nodo dal corridoio senza compromettere la funzionalità dell'algoritmo; il numero totale di NA diverrebbe così 10. In ogni caso, poichè l'intero sistema è concepito per funzionare in situazioni di emergenza in cui non è da escludersi il danneggiamento di uno o più nodi àncora, è preferibile tenere un certo grado di

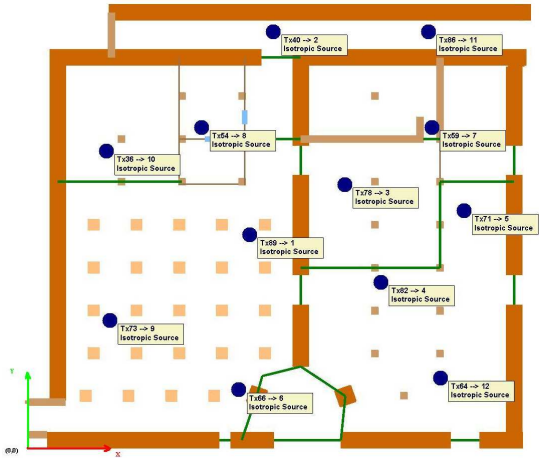


Fig. 20: Disposizione nodi àncora nell'ambiente.

nodo àncora	coord. x[m]	coord. y[m]
Tx89 → 1	18.5	18
Tx40 → 2	20.5	35
Tx78 → 3	26.5	22.2
Tx82 → 4	29.5	14
Tx71 → 5	36.5	20
Tx66 → 6	17.6	5
Tx59 → 7	33.8	27
Tx54 → 8	14.5	27
Tx73 → 9	6.8	10.8
Tx36 → 10	6.5	25
Tx86 → 11	33.5	35
Tx64 → 12	34.5	6

Tab. 1: Coordinate dei nodi àncora nella disposizione adottata.

ridondanza nella copertura. Da notare il fatto che il numero esiguo di nodi àncora utilizzati (rispetto a quelli necessari utilizzando un approccio alternativo, i.e. mediante triangolazione), permette di snellire notevolmente le procedure di cablaggio ed installazione della rete stessa: questa possibilità permette di incrementare l'economicità dell'intero sistema.

Con la disposizione appena determinata è possibile simulare nuovamente il comportamento del segnale radio all'interno dell'ambiente; quanto ottenuto viene riportato in Fig. 21.

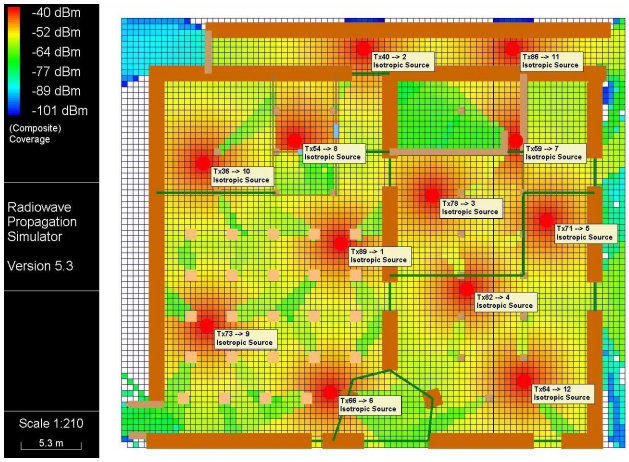


Fig. 21: Simulazione del segnale radio di tutti i nodi àncora utilizzati.

In ogni posizione dell'area in esame la copertura (composita) assicuri una potenza del segnale ricevuto non inferiore a -65 dBm: ciò si vede anche dal grafico della distribuzione della copertura, presente in Fig. 22.

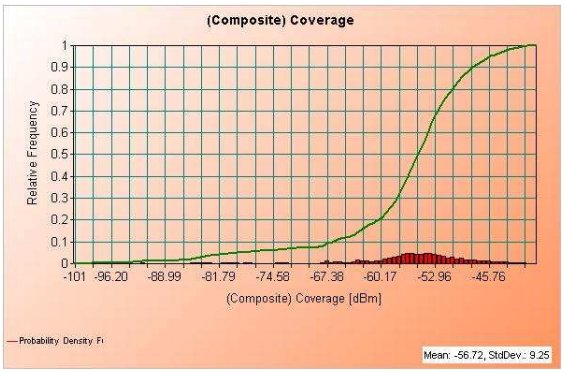


Fig. 22: Grafico della distribuzione della copertura (composita).

V. Componenti hardware e software

A. Mote *utilizzati*

Una rete wireless per la localizzazione in un ambiente lavorativo come quello dei Laboratori dell'INFN richiederà l'impiego di strumenti non invasivi nè appariscenti, ed in grado di sopravvivere con manutenzione minima anche in condizioni molto variabili. Nel progetto i nodi sono i *T*mote *Sky* prodotti dalla Moteiv Corporation [14], messi a disposizione dalla Facoltà di Ingegneria, che offrono buone prestazioni di base[6].

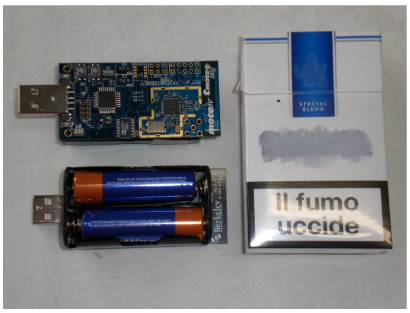


Fig. 23: Un nodo T*mote* Sky confrontato con un pacchetto di sigarette per dare un'idea delle dimensioni.

I Tmote Sky sono mote a bassissimo consumo su cui è montato un microcontrollore MSP430 prodotto dalla Texas Instruments (8 MHz, 10 kB di RAM, 48 kB di memoria Flash) ed un *chip* radio CC2420 prodotto dalla Chipcon: di seguito il dettaglio delle componenti più rilevanti per l'applicazione svolta.

1) Chip radio: Le comunicazioni radio sono rese possibili dal *chip* radio CC2420 della Chipcon [15] (250 kbps, 2.4 GHz, implementazione del protocollo IEEE 802.15.4 [16]). Tramite il microcontrollore MSP430 è possibile spegnere ed accendere il *chip* e modulare la potenza di trasmissione, programmabile su 31 livelli compresi in [0,-25] dBm. Ad ogni segnale ricevuto è possibile associare un valore, l'RSSI, legato alla potenza del segnale ricevuto *(Received Signal Strength, RSS)* come mostrato in Fig. 24. In generale, si ha che con ottima approssimazione vale:

$$RSSI_{[dBm]} = RSS_{[dBm]} + RSSOFFSET_{[dBm]} \quad (3)$$

dove $RSSOFFSET_{[dBm]}$ è una costante che varia da mote a mote, pur rimanendo nell'intorno dei -45 dBm. La funzionalità del calcolo dell'RSSI è determinante per lo svolgimento dell'intero progetto, poichè gli algoritmi proposti si basano soltanto su questa grandezza: il *range* di valori dell'RSSI restituiti dal *chip* è [-60, 40] dBm; segnali che arrivano al ricevitore con potenza inferiore alla soglia non vengono rilevati.

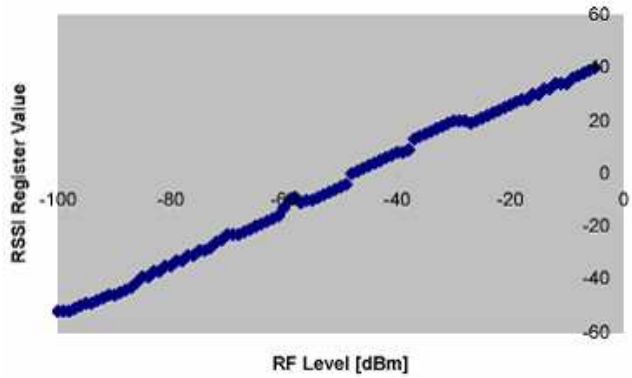


Fig. 24: Mappa tipica tra la potenza del segnale ricevuto e l'RSSI.

2) Antenna radio: I T*mote* Sky offrono un'antenna integrata di tipo *inverted-F microstrip design* (antenna a banda stretta a forma di F rovesciata), composta da una pista di rame posta su un lato della scheda coperta da uno strato di vernice che funge da dielettrico, mentre sul lato opposto della scheda un substrato di rame ha potenziale di terra. L'antenna offre un campo di irradiamento pressochè circolare con poche irregolarità; verifiche sperimentali hanno confermato che in assenza di ostacoli (*LOS condition*) il raggio di trasmissione è di ca. 50 m, come specificato in [14], dove si mostra anche che la presenza del vano batterie abbassa leggermente le prestazioni dell'antenna. Le specifiche del progetto, tuttavia, non hanno reso necessario ricorrere all'antenna opzionale esterna prevista per i T*mote*.

3) Alimentazione: L'alimentazione può essere fornita da due batterie di tipo AA oppure tramite il collegamento USB; per un funzionamento conforme alle direttive di programmazione, è necessario che la tensione di alimentazione sia compresa tra i 2.1 V ed i 3.6 V, e non sia inferiore ai 2.7 V in fase di riprogrammazione. Il collegamento alla porta USB di un pc alimenta il mote a 3 V. È fondamentale che sia sempre superato il limite di alimentazione minima per consentire una corretta interpretazione dei dati forniti da ogni nodo: un accurato controllo del livello di carica delle batterie è stato indispensabile prima di ogni operazione.

[6]Sebbene tali nodi siano, nella versione utilizzata, privi di sensoristica, è uso comune riferirsi ad essi come sensori, e tale abitudine verrà adottata anche in questo documento.

In Tab. 2 sono riportate le condizioni operative di maggiore rilevanza.

	MIN	MAX
Tensione di alimentazione richiesta	2.1 V	3.6 V
Tensione di alimentazione richiesta durante la riprogrammazione	2.7 V	3.6 V
Temperatura esterna che consente il funzionamento	-40°C	85°C

Tab. 2: Condizioni operative del T*mote* Sky.

B. Protocolli di comunicazione

1) Il protocollo 802.15.4 per la trasmissione radio: Le reti wireless, utilizzate per trasmissioni senza fili a distanza limitata, sono supportate dallo standard IEEE 802.15.4. Questo protocollo, approvato nel 2003, definisce le specifiche di interconnessione tra diversi dispositivi che vanno a comporre una *Low Rate-Wireless Personal Area Network (LR-WPAN)*: tali reti sono costituite da dispositivi alimentati tramite batterie che non possono essere sostituite frequentemente come, ad esempio, i sensori. Il protocollo 802.15.4 si prefigge le seguenti finalità:

- basso bit-rate;
- basso consumo di energia;
- basso costo.

Come tutti i membri della famiglia IEEE 802, anch'esso si preoccupa di definire soltanto i primi due livelli corrispondenti nel modello ISO/OSI, ovvero il *Physical layer (PHY)* ed il *Medium Access Control (MAC)*, come visualizzato in Fig. 25. La Fig. 25 permette inoltre di chiarire un errore alquanto

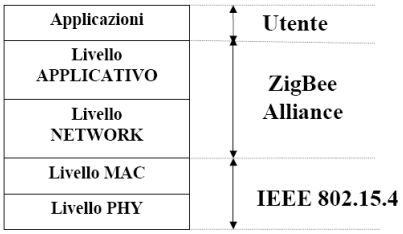


Fig. 25: Suddivisione dei livelli nel modello ISO/OSI.

diffuso: il prodotto commerciale ZigBee della ZigBee® Alliance[7] e lo standard IEEE 802.15.4 non sono sinonimi, ma ensì complementari, in quanto ZigBee è in grado di specificare i livelli superiori, a partire dal livello di rete fino al livello di applicazione, esclusi il PHY e il MAC.

a) Physical layer: Il livello fisico dello standard 802.15.4 fornisce due servizi: il *PHY data service* che abilita la ricezione e la trasmissione dei *Protocol Data Unit (PDU)*, pacchetti spediti sulla radio), ed il *PHY management service* che si interfaccia con il *Physical Layer Management Entity (PLME)* fornendo alcuni servizi di controllo. Le caratteristiche ed i compiti del livello fisico possono essere sintetizzati come:

- attivazione e spegnimento del trasduttore radio *(radio transceiver)*;
- *Energy Detection (ED)*, rilevamento della potenza nel canale radio;
- *Link Quality Indicator (LQI)*, indicazione della qualità del canale radio;
- selezione del canale di comunicazione;
- *Clear Channel Assessment (CCA)*, riconoscimento di canale libero o occupato;
- trasmissione e ricezione di pacchetti nel canale radio.

Lo standard offre due bande di frequenze, operanti a *bitrate* e modulazioni diverse: i valori principali sono riportati in Fig.26.

PHY	Banda di frequenza	Numerazione canali	Parametri di spreading		Parametri dato		
			Chip rate	Modulazione	Bit rate	Symbol rate	Mappatura bit a simbolo
800/915 MHz	868-870 MHz	0	300 kchip/s	BPSK	20 kb/s	20 kbaud	Binaria
	902- 928 MHz	Da 1 a 10	600 kchip/s	BPSK	40 kb/s	40 kbaud	Binaria
2.4 GHz	2.4-2.4835 GHz	Da 11 a 26	2.0 Mchip/s	O-QPSK	250 kb/s	62.5 kbaud	16-ary Orthogonal

Fig. 26: Tabella riassuntiva della descrizione del livello fisico.

b) Medium Access Control: Il livello di accesso al mezzo rappresenta il livello più interessante e più complesso del protocollo 802.15.4. Questo livello fornisce due servizi: il *MAC data service*, che abilita la ricezione e la trasmissione dei MAC-PDU attraverso i servizi del livello fisico, ed il *MAC management service*, che si interfaccia con il *MAC subLayer Management Entity (MLME)* fornendo alcuni servizi di controllo. Le caratteristiche ed i compiti del livello MAC sono:

- gestione dei *beacon*;
- accesso al canale;
- controllo sui frame;
- invio dei pacchetti di conferma *acknowledgement (ACK)*;
- associazione e dissociazione da una rete.

Il cuore del livello MAC di IEEE 802.15.4 è l'algoritmo che gestisce la contesa del canale e l'accesso al mezzo. Il metodo di accesso a contesa è il CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*, accesso multiplo mediante ascolto della portante con meccanismo per evitare le collisioni); questo metodo consiste nel mettersi in ascolto sul canale ed utilizzarlo solo se libero. In caso risultasse occupato, si attende un tempo casuale e viene ripetuto il tentativo di accesso. Il suffisso *Collision Avoidance* indica proprio il fatto che si evitano le collisioni monitorando il livello di potenza sul canale prima di trasmettere.

Nelle reti previste dal protocollo IEEE 802.15.4 possono essere distinti due tipi di dispositivi: i *full-function devices* (FFD), dispositivi con funzionalità completa, e i *reduced-function devices* (RFD), dispositivi con funzionalità ridotta. Gli FFD implementano tutte le funzionalità dello standard 802.15.4 e quindi possono operare in tre modi: come coordinatori della PAN, come coordinatori semplici o come normali *device*. Gli RFD non implementano tutte le specifiche dello standard 802.15.4, e non possono ricoprire il ruolo di *PAN coordinator*; per questo sono utilizzati in applicazioni semplici che non prevedano la trasmissione di grandi quantità di dati e

[7]Si veda il sito web: `http://www.zigbee.org/`

quindi hanno bisogno di ridotte risorse e limitata disponibilità di memoria. Un FFD può comunicare sia con gli FFD che con gli RFD, mentre un RFD può comunicare solo con un FFD.

Lo standard in esame supporta due tipologie di rete, mostrate in Fig.27:

- *Stella*: in questa tipologia tutti gli elementi della rete dipendono da un coordinatore e dialogano direttamente con esso. Spesso il coordinatore viene alimentato da una rete elettrica mentre gli altri dispositivi sono alimentati da batterie. Un FFD viene eletto a coordinatore della rete e fornisce il controllo di quell'insieme di dispositivi, rendendo la rete indipendente dalle altre eventualmente presenti. Ogni dispositivo vede soltanto il coordinatore e per dialogare con altri dispositivi deve passare attraverso di esso.
- *Peer-to-peer*: anche in questo caso esiste un solo coordinatore, ma ogni dispositivo può comunicare con gli altri senza dover passare dal coordinatore: la rete assume una topologia magliata. Questo tipo di topologia è meno gerarchica e rigida e ben si adatta a reti autoconfiguranti o reti ad-hoc. Inoltre, questo tipo di rete permette anche la comunicazione *multi-hop* e fornisce una maggiore affidabilità della rete mediante percorsi multipli tra sorgente e destinazione. La rete sviluppata per questo progetto ha una topologia di questo tipo, anche se diversi sono i ruoli ricoperti dai vari nodi.

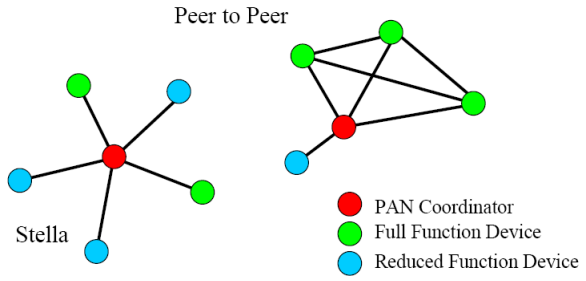


Fig. 27: Tipologie di rete.

2) Protocollo per la trasmissione seriale: La trasmissione seriale di TinyOS fa riferimento ad un protocollo *point-to-point (PPP)* in *HDLC (High-Level Data Link Controller)-like framing* [17] [18]. PPP si riconduce ai principi descritti nell'"ISO 3309-1991 *HDLC frame structure*"; un pacchetto *(frame)* di dati, escludendo i bit inviati per la sincronizzazione, vede l'invio, nell'ordine, di:

- *Flag* (01111110) (0x7e): segnala l'inizio del pacchetto, e serve per avere una corretta interpretazione dei dati trasmessi; tra un pacchetto e l'altro deve essere presente una sola *flag sequence*;
- *Address* (11111111) (0xff): questa sequenza rappresenta di norma l'indirizzo generico corrispondente a tutti i dispositivi in grado di ricevere;
- *Control* (00000011) (0x03): contiene il comando *Unnumbered Information* con il bit *Poll/Final*[8] posto a 0;
- *Protocol*: 1 byte il cui valore identifica la struttura dati contenuta nel campo *Information*: i valori ammessi da TinyOS sono `PROTO_ACK` (0x40), per un pacchetto di *acknowledgement*, `PROTO_PACKET_ACK` (0x41) per una trasmissione che attende l'invio di un *acknowledgement* dal ricevente, `PROTO_PACKET_NOACK` (0x42) per un pacchetto di dati che non richiede *acknowledgement*, `PROTO_UNKNOWN` (0xFF) quando un componente segnala di aver ricevuto un pacchetto con un protocollo non supportato o non riconosciuto.
- *Information*: i dati da trasmettere; questo campo consta di un numero di byte compreso fra 0 e il valore di *Maximum Receive Unit (MRU)*[9], tipicamente pari a 1500;
- *Frame Check Sequence (FCS)*: sono 2 byte trasmessi in *little-endian*; il valore di questo campo è determinato in base ai bit dei campi *Address, Control, Protocol, Information* e *Padding*;
- *Flag* (01111110)(0x7e): indica la conclusione del *frame*; l'invio consecutivo di un altro *frame* non prevede l'invio di un nuovo *flag*, anzi due *flag* consecutivi vengono interpretati come un pacchetto vuoto, e quindi ignorati.

È sempre attuata la compressione dei campi *Address* e *Control*: nella fase di trasmissione, i due campi sono semplicemente omessi, per essere aggiunti automaticamente in fase di ricezione. Inoltre, è previsto un *Cyclic Redundancy Check (CRC)* a 16 bit, per la verifica dell'integrità dei dati.

C. Il sistema operativo: TinyOS

L'hardware impiegato nelle WSN è caratterizzato da dimensioni contenute, struttura modulare e risorse limitate; poichè il funzionamento richiesto è di tipo continuativo e in *real-time*, diventa fondamentale poter disporre di un software di controllo veloce, con un'ottima gestione degli eventi in concorrenza ed orientato al risparmio energetico. La soluzione attualmente più diffusa è costituita dal *TinyOS* [19] [20], sistema operativo *open-source* sviluppato dall'Università di Berkeley (California) e scritto nel linguaggio di programmazione *nesC*. Le tre caratteristiche più importanti di TinyOS sono l'architettura *component-based*, il modello di gestione della concorrenza e le operazioni *split-phase*.

1) Architettura a componenti: TinyOS fornisce una serie di componenti di sistema da utilizzare nei programmi, che possono far riferimento ad elementi hardware o software; un'applicazione si occupa innanzitutto di collegare i componenti mediante *wiring specification*, e poi sceglie (o fornisce) un'adeguata implementazione per gli stessi. La separazione in componenti permette di utilizzare soltanto gli strumenti effettivamente necessari all'esecuzione del programma, risparmiando le risorse.

2) Gestione di eventi concorrenti: TinyOS è un sistema operativo *single-tasking*, cioè consente l'esecuzione di una singola operazione alla volta. Tuttavia le funzioni eseguite sono di due tipi: i processi (*task*) e gli eventi (*event* o *hardware*

[8]Si parla di bit di *poll* quando la stazione che trasmette chiede una risposta, è invece un bit di *final* quando indica che il *frame* corrente rappresenta una risposta oppure il termine di una trasmissione.

[9]Anche se non è il caso dei Tmote Sky, può capitare che i byte di dati siano seguiti da un certo numero di byte di *padding*, in modo che i byte totali del campo *Information* siano pari alla *MRU*.

interrupt), che vengono gestiti da uno *scheduler*. I *task* sono richieste di esecuzione, e sono gestiti con una politica FIFO *run-to-completion*[10]: vengono eseguiti nello stesso ordine con cui sono stati chiamati e, una volta cominciati, non possono essere interrotti da altri *task*. Gli eventi sono delle richieste di *interrupt* e rispetto ai *task* sono asincroni ed hanno una priorità più alta: il *concurrency model* prevede che il verificarsi di un evento provochi l'immediata esecuzione del codice associato, e non possono essere interrotti se non da altri eventi. Di fatto, l'esecuzione di un programma in TinyOS è governata dal verificarsi di eventi, corrispondenti ad *interrupt* hardware (si veda anche V-D.3 per maggiori dettagli).

3) Operazioni split-phase*:* In TinyOS ogni operazione non banale, come ad esempio l'invio di un pacchetto via radio, diventa un'operazione *split-phase*: la chiamata alla funzione e la conclusione della stessa diventano due operazioni separate. In questo modo la chiamata ad un comando diventa una semplice richiesta di esecuzione: il componente chiamato si limita ad avviare un *task*, al compimento dell'operazione verrà poi segnalato un evento, specificato dal componente chiamante. Questa filosofia, contrapposta a quella di tipo *blocking* che tiene occupato il sistema finchè un'operazione non giunge al termine, permette un notevole livello di concorrenza ed una gestione molto snella dei *task*.

TinyOS è un progetto in continua evoluzione: l'ultima release è la 2.0.1 (maggio 2007). Per lo svolgimento del progetto, tuttavia, è stata installata *Boomerang 2.0.4*, una distribuzione di TinyOS messa a disposizione gratuitamente dalla Moteiv Corporation che contiene principalmente librerie di TinyOS 1.1.15, oltre ad una versione di *Cygwin* con le estensioni necessarie ed il Java Runtime Environment 1.5.0.10. Questa scelta è stata guidata dal desiderio di usare strumenti ampiamente collaudati, anche se più limitati, e che consentissero, in caso di necessità, di poter contare sul supporto della Moteiv. Inoltre, considerando che il *porting* di applicazioni scritte in TinyOS 1.x a TinyOS 2.x può essere fatto con un modesto sforzo, tale soluzione non è apparsa così restrittiva.

D. Il linguaggio di programmazione: nesC

Network embedded system C (nesC) è un linguaggio di programmazione sviluppato in collaborazione tra l'Università di Berkeley e Intel Research. Si tratta di un dialetto derivato dal linguaggio C che implementa un'architettura a componenti e un *concurrency model* basato sugli eventi, esattamente com'è nella filosofia del TinyOS; inoltre l'allocazione statica della memoria e la possibilità di analizzare a priori il flusso di esecuzione lo rendono particolarmente adatto alla programmazione dei mote di una WSN. La comprensione di un applicazione in nesC non può prescindere dalla descrizione dei *componenti*, delle *interfacce* e delle *priorità di esecuzione* del codice *(concurrency model)*.

1) Descrizione dei componenti: I componenti in nesC possono essere indifferentemente elementi hardware o software; le applicazioni vengono costruite collegando fra loro dei componenti. Un componente può *implementare (provides)* o *usare (uses)* le interfacce, che costituiscono l'unico modo per avere accesso ai componenti. Le interfacce sono bidirezionali, poichè specificano dei *comandi (command)* e degli *eventi (event)*: i comandi sono legati alle funzionalità messe a disposizione al programmatore, mentre gli eventi contengono le operazioni da svolgere quando si verificano determinate condizioni. Si veda ad esempio l'interfaccia `Timer` per l'invio di messaggi:

```
interface Timer {
 command result_t start(char type, uint32_t interval);
 command result_t stop();
 event result_t fired();
}
```

`start` fa partire il timer nella modalità `type` (continuo, o *one-shot*) e per il tempo `interval`, `stop` ferma il timer, nell'implementazione di `fired` verranno specificate le operazioni da compiere al raggiungimento dell'intervallo prefissato. Un componente che implementa un'interfaccia ne implementa i comandi, mentre un componente che usa un'interfaccia ne implementa gli eventi.

Questa definizione di interfaccia facilita la comprensione delle operazioni *split-phase*, punto di forza del TinyOS: il comando rappresenta la richiesta di esecuzione, mentre l'evento viene segnalato alla conclusione del comando; questa struttura rende particolarmente semplice la gestione degli *interrupt hardware*.

2) Implementazione dei componenti: Ci sono due tipi di componenti in nesC: le *configurazioni (configuration)* e i *moduli (module)*; entrambi hanno la seguente struttura:

```
tipo nome_componente { \\ tipo = module,configuration
 provides {
  \\ interfacce implementate
  }
 uses {
  \\ interfacce usate
  }
}
implementation {
 \\ codice
}
```

Le configurazioni contengono i legami *(wiring specification)* fra i componenti, specificando le implementazioni dei comandi delle interfacce usate dal componente; spesso le configurazioni sono dette *top-level configuration* perchè permettono di avere un quadro complessivo del funzionamento dell'applicazione. Si veda ad esempio `Blink`:

```
configuration Blink {
}
implementation {
  components Main, BlinkM, SingleTimer, LedsC;
  Main.StdControl -> SingleTimer.StdControl;
  Main.StdControl -> BlinkM.StdControl;
  BlinkM.Timer -> SingleTimer.Timer;
  BlinkM.Leds -> LedsC;
}
```

i componenti utilizzati per l'applicazione sono `Main, BlinkM, SingleTimer, LedsC`; per indicare le interfacce legate ad un componente si usa la sintassi `componente.interfaccia`. Le frecce indicano dove trovare l'implementazione delle interfacce usate: ad esempio i comandi dell'interfaccia `Timer` usata

[10]Quanto qui riportato vale soltanto per TinyOS 1.x: le versioni successive prevedono una modifica allo *scheduler* che realizza una scala di priorità per i *task.*

da `BlinkM` fanno riferimento all'implementazione fornita da `SingleTimer`. `Main` è il componente che viene eseguito per primo in un'applicazione nesC, e l'interfaccia `StdControl` contiene le istruzioni di base per l'inizializzazione del mote e del programma: in ogni applicazione non deve quindi mancare l'implementazione di quell'interfaccia. In generale, è possibile collegare più comandi alla stessa implementazione *(multiple fan-in)* oppure lo stesso comando può far riferimento a più implementazioni *(multiple fan-out)*: ciò consente di compiere più operazioni con un notevole risparmio di codice scritto.

Il modulo contiene invece il codice che effettivamente implementa i comandi o gli eventi delle interfacce che il componente implementa o usa. Il linguaggio dei moduli è molto somigliante al C: per chiamare `comando` si usa la sintassi
`call interfaccia.comando`,
mentre un evento viene segnalato con
`signal interfaccia.evento`;
c'è inoltre la possibilità di fare riferimento a C-file per la definizione di nuove strutture dati e la definizione di costanti globali.

È una convenzione da adottare quella di usare la lettera maiuscola per il nome delle interfacce e dei componenti; in caso di ambiguità la lettera finale M indicherà che il file è un modulo, mentre la C finale fa riferimento alle configurazioni. Ad esempio:

- `Timer.nc`: interfaccia Timer;
- `TimerC.nc`: configurazione del componente che implementa l'interfaccia Timer;
- `TimerM.nc`: modulo che contiene l'implementazione dell'interfaccia Timer.

3) Concurrency model: Come già visto nella descrizione di TinyOS, il codice si divide in *task* ed *event*: i primi sono codice sincrono, eseguito con una politica *FIFO run-to-completion*, mentre i secondi sono asincroni, corrispondono a richieste di *interrupt* ed interrompono il processo corrente. Gli eventi asincroni non possono contenere chiamate dirette a funzioni sincrone: possono però richiedere l'esecuzione di un *task* mediante il comando
`post nome_task()`
che accoda `nome_task` nello *scheduler* che guida il flusso di esecuzione. L'operazione di *posting* rappresenta l'unico modo possibile per passare da istruzioni asincrone a sincrone. È anche possibile rendere asincroni comandi o eventi dichiarandoli `async`: le chiamate ad essi ne provocano l'esecuzione immediata, bloccando l'istruzione corrente (sia essa riferita ad un *task* o ad un *event*). Codice `async` può essere invocato direttamente dagli *event*; poichè nesC prevede che funzioni asincrone possono chiamare comandi o segnalare eventi solo se asincroni, mentre possono essere chiamate anche da codice sincrono. Ovviamente è una buona regola quella di mantenere limitate le dimensioni del codice asincrono, per non avere un eccessivo tempo di latenza nell'esecuzione dei processi. Riassumendo, l'intero codice può essere formalmente suddiviso tra:

- codice sincrono (SC): eseguito soltanto all'interno di *task*;
- codice asincrono(AC)[11]: esiste almeno un *interrupt* che causa l'immediata esecuzione di questo codice.

Questa distinzione può dar luogo a *data race*, ovvero ad un conflitto sull'accesso ai dati: in generale, qualsiasi accesso ai dati fatto tramite AC può dare origine a conflitto. Si veda a questo proposito il seguente esempio:

```
bool busy;
async command result_t do() {
 if (!busy) {
  busy = TRUE;
  \\ esecuzione
  return SUCCESS;
  }
 return FAILED;
}
```

e si supponga che `busy = TRUE` al momento di una chiamata a `do()`. Se capita un'altra chiamata a `do()` subito dopo il controllo `if (!busy)`, entrambe le istruzioni contenute nell'`if` verranno considerate eseguite, mentre in realtà soltanto una delle due chiamate sarà andata a buon fine. Per risolvere questi conflitti nesC mette a disposizione la dichiarazione `atomic`, che blocca l'esecuzione degli interrupt: come per `async`, il codice atomico non deve contenere troppe istruzioni, per non inibire troppo a lungo la possibilità di generare *interrupt*. Questa organizzazione genera così un'invariante che il compilatore nesC usa per individuare la presenza di potenziali *data race*: se l'accesso ad una variabile condivisa non avviene solo all'interno di SC, allora deve essere contenuto all'interno di codice atomico. Questa invariante, se non è soddisfatta, causa un errore di compilazione; tuttavia, se certo dell'impossibilità di un conflitto, il programmatore può aggirare questa condizione senza far ricorso ad *atomic statements* dichiarando una variabile `norace`, e forzando il compilatore a tralasciare qualsiasi analisi di conflitto di accessi a quella variabile. In generale [20][cap. 2], è consigliabile ricorrere con molta parsimonia al codice asincrono, dichiarandolo solo per operazioni per cui il tempo di latenza deve essere necessariamente minimo. La caratteristica `async`, se fondamentale per il corretto funzionamento di un'applicazione, è indicata direttamente nella dichiarazione dell'interfaccia.

È molto importante (e, purtroppo, spesso non viene fatto) non confondere gli *event* legati agli *interrupt* con gli *event* specificati nelle interfacce: di default, tutto il codice che viene scritto (comandi, eventi, *task*) è sincrono. Illuminante a questo proposito è l'esempio del funzionamento della trasmissione su porta seriale [20][cap. 4.5]: alla ricezione di dati, viene segnalata un'interruzione hardware (asincrona), che costringe il mote a leggere i dati dal registro ed a spostarli in un buffer. Dopo aver spostato l'ultimo byte, bisogna segnalare che la ricezione è conclusa: ciò viene fatto postando un *task*, che verrà eseguito non appena i processi pendenti saranno stati eseguiti.

E. La comunicazione tra i nodi

L'algoritmo implementato prevede vari scambi di dati tra i componenti della rete. Il diagramma di flusso in Fig. 28

[11] I termini "sincrono" e "asincrono" fanno qui riferimento all'esecuzione rispetto ai *task*: l'intervallo di tempo che intercorre fra l'esecuzione di un processo e di un'operazione asincrona non è predicibile.

permette di visualizzare le operazioni che devono essere svolte per ottenere la localizzazione del nodo mobile:

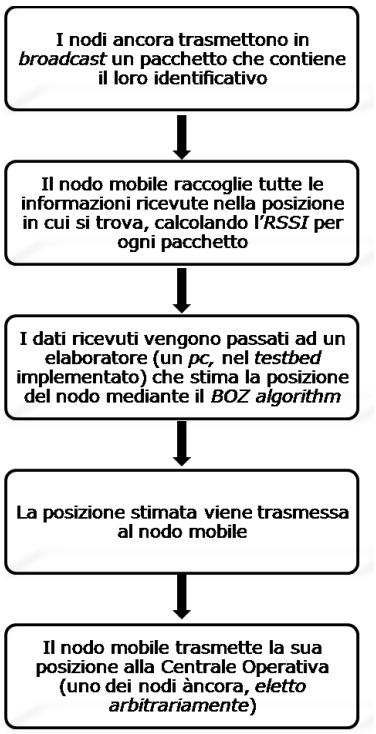


Fig. 28: Sequenza delle operazioni svolte per la localizzazione.

ogni trasmissione che vede mote sia come mittente che come destinatario avviene in modo wireless, mentre le comunicazioni da e verso l'elaboratore passano attraverso la porta seriale, dopo aver opportunamente collegato il mote. Si noti che la maggior parte del codice sviluppato è stato mantenuto sincrono, onde evitare conflitti o rendere esclusivo l'accesso alle risorse.

1) Definizione delle costanti e delle strutture dati: Nelle varie fasi dell'algoritmo vengono scambiati messaggi con dati diversi; per semplificare la lettura del codice, sono state create delle strutture dati apposite, contenute, assieme alle costanti, nei file `param.h` e `stimaMsg.h`, incluso in ogni componente sviluppato. La struttura di `Tos_Msg` permette, impostando adeguatamente i puntatori, di includere queste strutture dati nel campo `data` e di facilitare così le operazioni di invio. I messaggi che circolano devono fondamentalmente veicolare tre tipi di contenuti: l'identificativo dei nodi àncora, l'RSSI dei pacchetti ricevuti (con mittente associato) e la posizione stimata. Le strutture corrispondenti sono:

- `msgAnc`: messaggio trasmesso dai nodi àncora; i campi sono `id` (1 byte), identificativo del nodo mittente (un numero intero compreso tra 1 e 12, poichè 12 sono i nodi), `coordX` (1 byte) e `coordY` (1 byte), le coordinate x e y che esprimono la posizione del mittente nel sistema di riferimento, `potTx` (1 byte), la potenza di trasmissione;
- `totalData`: messaggio trasmesso dal nodo mobile all'elaboratore; i campi sono `rss` (2 byte in formato *big-endian*), RSSI del pacchetto ricevuto dal nodo àncora, `receivedMessage` (4 byte), i dati contenuti nel `msgAnc` ricevuto;
- `stimaMsg`: messaggio trasmesso dall'elaboratore al nodo mobile, che lo inoltra alla centrale operativa; il campo è `data` (3 byte), contenente un numero identificativo della stima (1 byte) e le coordinate x e y stimate che esprimono la posizione del mittente nel sistema di riferimento (rispettivamente 1 byte)

2) Raccolta dati per la stima della posizione: I nodi àncora devono essere programmati in modo da trasmettere ripetutamente e senza preoccuparsi dell'eventuale ricezione del pacchetto. È stata perciò sviluppata l'applicazione `TxAncora`, composta dall'omonima configurazione e dal modulo `TxAncoraM`, che consente di inviare con una frequenza predefinita ed in *broadcast* un messaggio di tipo `msgAnc`. La frequenza di trasmissione utilizzata, considerando l'assenza di disturbi nel *range* di funzionamento del protocollo 802.15.4, è quella del primo canale disponibile, 2405 MHz. In realtà, al *BOZ algorithm* non serve la posizione dei nodi àncora; tuttavia tale informazione è stata inserita nel messaggio da inviare pensando all'eventuale implementazione di tecniche di triangolazione per una maggiore accuratezza nella localizzazione. Allo stesso modo, in tutti gli esperimenti compiuti la potenza di trasmissione è stata sempre posta pari alla massima potenza disponibile (0 dBm); poichè, però, future esigenze di risparmio energetico possono comportare una potenza di trasmissione variabile (ad esempio, in funzione della carica delle batterie), si è ritenuto opportuno provvedere alla trasmissione di questa informazione[12]. L'invio in *broadcast* radio è ottenuto semplicemente indicando come indirizzo del destinatario la costante `TOS_BCAST_ADDRESS` definita in `AM.h` (si veda l'appendice III-B). Si osservi che la scelta di una frequenza di invio uguale per tutti i mote non impedisce il corretto funzionamento dell'algoritmo: il protocollo 802.14.5, grazie al meccanismo (CSMA-CA) descritto in V-B.1 conferisce robustezza all'invio e minimizza la perdita di pacchetti.

Il nodo mobile, dal canto suo, deve ricevere i pacchetti dai nodi àncora ed associare ad ognuno l'RSSI, prima di inviare questi dati all'elaboratore. L'applicazione `RxMobile`, composta dall'omonima configurazione e dal modulo `RxMobileM`, si preoccupa della ricezione dei dati `msgAnc`, della creazione di un messaggio `totalData` e dell'inoltro al pc. In realtà le applicazioni sviluppate per il nodo mobile sono più d'una, e leggermente diverse tra loro, poichè diverse sono le funzioni che devono svolgere.

Durante la fase di raccolta dati per la costruzione della mappa a priori di RSS e varianza, il nodo mobile compie 100 rilevazioni per ogni posizione stabilita: durante ogni rilevazione l'unica cosa importante è salvare un solo pacchetto per mote mittente. A questo proposito in `idArray` si memorizzano gli identificativi dei nodi àncora da cui è già arrivato un pacchetto; ad ogni ricezione, prima di ogni altra operazione, viene effettuato un controllo sull'`id` del mittente: solo se in quella rilevazione non è già stato ricevuto nulla da quel mote si

[12]Considerando che il *chip* radio scambia dati a 250 kbps, l'invio di 3 byte in più rispetto a quelli strettamente necessari comporta un aumento nel tempo di invio e di ricezione pari a 2×96 μs = 192 μs, valore che appare assolutamente trascurabile - si veda V-E.4.

procede per l'invio del pacchetto al pc. Si osservi che, in virtù di questo controllo sui mittenti, non è necessario segnalare il termine di ogni singola rilevazione: tutti i dati utili sono deducibili dal numero totale di pacchetti ricevuti da ogni nodo àncora, che non può superare il valore 100.

Nella raccolta dati per la localizzazione, invece, è necessario costruire un pacchetto delimitatore da inviare al termine di ogni singolo tempo di ascolto; in caso contrario, non è possibile capire quanti e quali nodi àncora sono stati rilevati dal nodo mobile. Il delimitatore è un pacchetto `totalData` con tutti i campi posti a 0. Per sicurezza, dal termine del tempo di ascolto fino all'inizio della rilevazione successiva, viene spento il ricevitore radio; inoltre, a differenza del caso precedente, non è prevista alcuna durata massima di funzionamento dell'algoritmo: per bloccare l'elaborazione è necessario chiudere manualmente il processo.

È poi necessario fare in modo che il pc raccolga i dati che riceve sulla porta seriale. A questo scopo sono stati utilizzati due tipi di strumenti: un *tool* Java e la libreria MATLAB™ per lo scambio dati sulla seriale, di seguito descritti. La seconda soluzione è stata sviluppata per poter garantire un funzionamento in tempo reale dell'algoritmo, dato che i pacchetti ricevuti possono essere dati direttamente come ingresso al filtro particellare; tuttavia la perdita di pacchetti a cui MATLAB™ andava incontro ha richiesto l'implementazione di alcune modifiche al codice originario destinato al nodo mobile, come si vedrà. Il pacchetto inviato sulla seriale ha struttura rappresentata in Fig. 29: il messaggio si apre con tre byte, contenenti rispettivamente il *Flag* (0x7e) di inizio spedizione, il *Protocol* (0x42) che caratterizza il messaggio come non richiedente *acknowledgement* dal destinatario e la lunghezza (in byte) del campo dati (0x06 nel caso `totalData`). Seguono 5 byte di *padding*, sempre nulli, per chiudere l'ottetto iniziale; poi 3 byte sono dedicati all'indirizzo, uno contiene l'identificativo del mittente (0x02, trattandosi di un nodo mobile) e un altro ancora riporta il gruppo di appartenenza (0x7d, `GROUP_ID` standard). Nei 6 byte successivi trova posto il dato, seguito da 2 byte di *FCS*; il pacchetto è concluso dalla *Flag* di fine trasmissione, seguito da 2 byte di *CRC*.

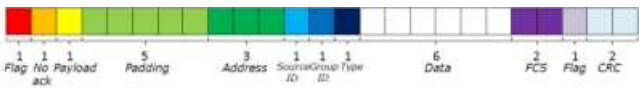


Fig. 29: Pacchetto inviato sulla seriale dal nodo mobile: la lunghezza del messaggio è di 25 byte.

TinyOS mette a disposizione alcune librerie Java per consentire una gestione più completa dei *Tmote*; tra queste l'applicazione `Listen.java`, che si limita a stampare sullo schermo ogni pacchetto ricevuto sulla porta seriale specificata[13] e l'applicazione `LogMsg.java` che traduce i dati ricevuti in formato stringa. Il programma è stato leggermente modificato in modo da salvare i byte di dato dei pacchetti ricevuti in un file `rilevazioni.txt` con un *header* appropriato. Il programma fa uso delle classi `FileOutputStream` e `PrintStream`, appartenenti al *package* standard Java di I/O (`package java.io.*`). Il file `rilevazioni.txt` è scritto in modo da essere facilmente elaborabile in modo automatico con uno *script* MATLAB™, e contiene soltanto i campi della struttura `totalData`.

Poichè MATLAB™ mette a disposizione degli strumenti per la comunicazione di base sulla porta seriale, servirsi di quelli permette di avere a disposizione i dati e procedere con la loro elaborazione in tempo reale. Nel file `serialListen.m` viene creato un oggetto `serial` con la proprietà *baud-rate* impostata a 57600 bps, come richiesto dai Tmote Sky. Gli oggetti `serial` prevedono un *buffer* in cui salvare i dati ricevuti: poichè un singolo pacchetto consta di 25 byte, ed a priori non è possibile sapere quanti pacchetti arriveranno durante la fase di ascolto corrente (dipende dal numero di mote àncora rilevati in quella particolare posizione), l'idea è quella di leggere i dati a mano a mano che arrivano. Si fa uso a questo proposito di una funzione di *callback* che venga chiamata non appena il numero di byte presenti nel *buffer* di ingresso, accessibili dalla proprietà `BytesAvailable`, sia maggiore di 25: al verificarsi dell'evento viene eseguita `rxcallback.m`, che tramite `saveData.m` provvede al salvataggio in una opportuna matrice dei byte che interessano (l'identificativo del nodo àncora mittente, contenuto al byte 17, e l'*RSSI* del pacchetto ricevuto dal nodo mobile, contenuto al byte 16, salvati in riga) e verifica l'arrivo del delimitatore che segna il termine della rilevazione corrente[14]. Quando ciò accade, si fa partire lo *script* del filtraggio particellare, per ottenere la stima della posizione corrente del nodo mobile.

Si è verificato sperimentalmente, tuttavia, che il codice *nesC* implementato per la trasmissione dei dati via seriale dal nodo mobile non era adatto alla comunicazione con MATLAB: circa il 90% delle rilevazioni vedeva la perdita di 1-3 pacchetti; prestazioni assolutamente insoddisfacenti per consentire un buon funzionamento del *BOZ algorithm*. Questo comportamento va attribuito al fatto che il nodo mobile invia i pacchetti a mano a mano che li riceve dai mote àncora, e può capitare che un certo numero di pacchetti venga inoltrato sulla seriale in rapidissima successione; in situazioni di questo genere MATLAB non si rivela sufficientemente pronto e si ha perdita di pacchetti. La soluzione a questo problema giace nel regolarizzare l'invio dei dati sulla seriale: il nodo mobile, alla ricezione di pacchetti via radio, provvede a salvarli in una certa variabile; l'inoltro sulla seriale è scandito da un timer. Queste modifiche permettono di azzerare la perdita di pacchetti dovuta alla lentezza di MATLAB™; la scelta della durata dei vari timer è discussa alla V-E.4.

È da aggiungere che entrambi gli strumenti descritti vanno incontro a degli errori nella ricezione, dovuti ad una mancata

[13] La porta seriale su cui scambiare dati è impostata tramite la variabile `MOTECOM`:
`export MOTECOM=serial@COMxx:baud`
dove `xx` è la seriale a cui è collegato il mote, facilmente individuabile tramite il comando `motelist`. `baud` rappresenta il *baud-rate* di trasmissione; è possibile anche usare delle costanti di sistema. Ad esempio `export MOTECOM=serial@COM1:tmote` imposta la velocità di trasmissione sulla COM1 a 57600 bps.

[14] In realtà, può capitare che due pacchetti arrivino consecutivamente, senza che MATLAB™riesca ad eseguire la funzione si *callback*. Poichè dopo l'esecuzione di tale funzione il *buffer* di ingresso viene in ogni caso svuotato, per evitare di perdere pacchetti `rxcallback` verifica il numero di pacchetti da elaborare con il comando `ceil(BytesAvailable/25)`, e si comporta di conseguenza.

sincronia tra mittente e destinatario. Questo fenomeno è tuttavia piuttosto limitato (nella costruzione della mappa a priori dei sotterranei, su 68583 pacchetti totali ricevuti da tutti i mote soltanto 253 si sono rivelati non validi, lo 0.37%), pertanto ci si limita a scartare i dati corrotti, come descritto nella VI-C: tale decisione non inficia le prestazioni complessive del *BOZ algorithm*.

3) Invio della posizione stimata alla centrale operativa: Una volta che il filtraggio particellare ha calcolato la posizione stimata all'istante corrente, le finalità del progetto richiedono che tale stima sia inviata ad una centrale operativa esterna. Date le condizioni operative della WSN, non è pensabile l'invio dei dati in wireless su protocollo TCP/IP direttamente dal palmare: in generale, non è detto che l'ambiente sia coperto e, quand'anche ciò si verificasse, in condizioni di emergenza la corrente potrebbe essere tolta vanificando il funzionamento della rete Internet. Si può pensare allora che un solo nodo àncora (o un sottoinsieme) possa svolgere la funzione di raccolta dati, per poi inviarli mediante cavo ad un elaboratore esterno, sfruttando così la WSN e la comunicazione seriale. Con questo approccio il percorso della posizione stimata si divide in due parti:

1) restituzione della stima al nodo mobile, mediante porta seriale;
2) invio della stima dal nodo mobile al nodo àncora collegato all'elaboratore esterno (nodo àncora eletto).

Per trasmettere dei dati da MATLAB™ al *mote* su porta seriale non è possibile usare i comandi `fwrite` a disposizione per gli oggetti di tipo `serial`: il messaggio che si genera in questo modo non si accorda al protocollo *PPP HDLC-like framing* e viene ignorato dal *mote*. Tuttavia, TinyOS mette a disposizione degli *M-file* per risolvere questo problema: essi, utilizzando degli opportuni oggetti Java, creano dei messaggi interpretabili secondo il protocollo di comunicazione del *mote* e rendono quindi possibile l'invio di dati da MATLAB™. Una volta configurato opportunamente MATLAB™ per una corretto funzionamento con le risorse Java, è stato allora utilizzato il *tool Message Interface Generator (MIG)* che consente, a partire da una struttura dati definita in C, di creare un'omonima classe Java che crei un messaggio leggibile da un dispositivo programmato con TinyOS e con il campo dati articolato come la struttura dati definita. Questo procedimento è stato applicato alla struttura dati `stimaMsg`, ottenendo la classe `stimaMsg.java` e `stimaMsg.class`. Un oggetto di questa classe viene istanziato da MATLAB™nello *script* `serialListen.m`, e i campi dati vengono opportunamente scritti al termine di ogni stima. Il messaggio viene così trasmesso tramite seriale al nodo mobile collegato all'elaboratore, che provvede ad inoltrarlo per farlo giungere alla centrale operativa.

Questo procedimento, nel funzionamento in tempo reale, richiede che MATLAB™ chiuda l'oggetto `serial` creato per la registrazione dei dati inviati dal *mote* per collegarsi (con lo *script*`connect.m`) sulla stessa porta mediante Java: la mancata esecuzione di questa operazione provoca un conflitto di risorse ed il conseguente *crash* dell'elaboratore. Allo stesso modo, è necessario interrompere l'ascolto sulla seriale tramite Java per riaprire l'oggetto `serial`[15]. Questa sequenza di istruzioni, però, richiede un intervallo di tempo abbastanza ingente per essere eseguita, portando a ca. 2.5 s il tempo complessivo necessario all'elaborazione dei dati raccolti ed all'invio della stima. La nuova temporizzazione ha richiesto una veloce ritaratura dei parametri dell'algoritmo, che ha di molto contenuto il peggioramento delle prestazioni; per questo, anche tenendo conto che i problemi sono legati all'utilizzo di strumenti che verranno abbandonati dopo la fase di ingegnerizzazione che seguirà questo *testbed*, non si è cercato di trovare strade alternative per velocizzare la trasmissione dei dati da pc a nodo.

Una volta inviata dal nodo mobile, la stima deve arrivare al NA (o ai NA) collegati con la centrale operativa, affinchè la posizione stimata dell'operatore possa essere monitorata dall'esterno. La disposizione scelta per i *beacon* assicura che ciascuno di essi comunichi con almeno un altro *beacon*, ed è sempre possibile trovare un cammino che in un numero finito di passi conduca ad un nodo arbitrario, qualsiasi sia il punto di partenza. Considerando il numero esiguo di NA, per far arrivare la stima della posizione al *mote* designato (chiamato Centrale Operativa, CO) è stato implementato un semplice meccanismo di *multi-hop*: il nodo mobile invia in *broadcast* la stima che ha ricevuto dall'elaboratore; ogni nodo àncora che la riceve, se è diverso dalla CO, si comporta allo stesso modo. Per evitare che si formino *loop* di comunicazione tra i *mote* àncora, e conseguente saturazione del canale di trasmissione, ogni messaggio è contrassegnato da un numero crescente: ogni *beacon* si limiterà ad ignorare messaggi con un identificativo minore rispetto all'ultimo messaggio inoltrato. La CO, ricevuto il messaggio, si limita a ritrasmetterlo ad un elaboratore in ascolto sulla porta seriale; in quest'ultimo, un opportuno *script* MATLAB™ estrae i dati e visualizza la posizione stimata dell'operatore sulla pianta dei sotterranei.

4) Temporizzazione delle operazioni: Poichè il funzionamento complessivo prevede che gli stessi dispositivi compiano operazioni diverse in un determinato ordine, un punto cruciale è costituito dalla scelta della temporizzazione. Il primo parametro da decidere è la frequenza con cui raccogliere i dati trasmessi dai nodi àncora: deve trattarsi di un intervallo di tempo sufficientemente grande da consentire lo svolgimento dei calcoli previsti dall'algoritmo ma abbastanza contenuto da consentire un *tracking* significativo dell'operatore. Le simulazioni hanno mostrato che, anche con un numero elevato di particelle (> 500) il tempo necessario all'elaborazione con MATLAB™è inferiore agli 0.25 s; un valore ragionevole relativo alla frequenza per compiere le rilevazioni è di 2000 ms_b: stimando una velocità media dell'operatore pari a 5 m/s questo implica una differenza di posizione tra un passo e l'altro di ca. 10 m. Bisogna poi decidere il tempo di ascolto del nodo mobile, ovvero l'intervallo in cui vengono raccolte le informazioni trasmesse dei nodi àncora. Tale valore è strettamente legato alla frequenza di trasmissione dei pacchetti dei nodi àncora; una scelta ottimale appare quella di minimizzare il numero di trasmissioni nell'unità di tempo, per minimizzare

[15]Si decide di mantenere distinti i metodi di ricezione ed invio dati da e verso un *mote*, per facilitare le operazioni necessarie all'elaborazione dati.

il consumo energetico e lasciare i nodi liberi di svolgere altre eventuali attività, quale l'invio della posizione stimata alla centrale operativa. Un buon compromesso si raggiunge ponendo a 500 ms$_b$ il periodo di invio dei nodi fissi, e tenendo in ascolto il nodo mobile per 1000 ms$_b$. In questo modo, quasi un secondo è lasciato per i calcoli e la ritrasmissione della posizione stimata. In realtà, il tempo di ascolto minimo richiesto, nel caso peggiore e supponendo che i dati raccolti siano trasmessi all'elaboratore in una soluzione unica al termine del periodo di ascolto, si può trovare dalla formula

$$\begin{aligned} 488 \text{ ms } &+ 10 \cdot 8 \cdot 0.032 \text{ ms} + \\ &+ \text{ (latenza dovuta alla sincronizzazione)} + \\ &+ \text{ (latenza dovuta all'elaborazione)} + \\ &+ \text{ (latenza dovuta alla trasmissione)} > 491\text{ms}, \end{aligned} \quad (4)$$

in cui si suppone che in ogni posizione possano essere "ascoltati" al più 10 *mote* (massimo valore rilevato sperimentalmente) ed ogni messaggio sia composto da 8 byte (esclusi i bit di sincronizzazione). Pertanto, ogni durata del tempo di ascolto maggiore di 500 ms appare sufficiente a scongiurare la perdita di pacchetti.

Per quanto riguarda la temporizzazione dell'invio dei pacchetti sulla seriale, il periodo scelto è di 48.8 ms. Si osservi innanzitutto che, poichè la velocità di trasmissione sulla seriale è di 57600 bps ed il pacchetto da trasmettere è lungo 25 byte (si veda Fig. 29), il tempo necessario per la trasmissione di un messaggio è apri a $25 \cdot 8 \cdot (1/57600)$ s $\simeq 3.5$ ms, molto inferiore del valore scelto. Bisogna ora controllare se, complessivamente, il tempo necessario alla raccolta dei dati, allo svuotamento del *buffer* ed all'elaborazione della stima è inferiore al tempo che intercorre tra l'inizio di una rilevazione e la rilevazione successiva. A questo scopo si può effettuare un calcolo per eccesso: tenendo conto che il massimo numero di *mote* che possono essere avvertiti come mittenti nella stessa rilevazione non supera la decina, il tempo necessario per lo svuotamento del *buffer* è di ca. 500 ms$_b$. Sempre nel caso peggiore (si veda (4)), si può ragionevolmente supporre che i pacchetti vengano ricevuti trascorsi 550 ms$_b$ dall'inizio del tempo di ascolto; entro 1050 ms$_b$, quindi, tutti i dati sono arrivati correttamente all'elaboratore tramite seriale. In un caso del genere rimarrebbero 950 ms$_b$ per effettuare i calcoli e rispedire la stima alla centrale operativa: in realtà il tempo richiesto per queste operazioni è inferiore ai 0,5 s. Si vede dunque che i valori scelti soddisfano i vincoli del corretto funzionamento anche in un caso come questo, ben oltre il caso peggiore: garantiscono quindi la piena operatività dell'algoritmo in *real-time*. Vero è che la temporizzazione scelta non ottimizza le prestazioni; ma poichè questo progetto si propone come semplice *testbed*, una successiva fase di ingegnerizzazione provvederà alla scelta ottima per ogni parametro.

Una temporizzazione diversa è purtroppo richiesta quando si voglia inoltrare la posizione stimata alla CO. Come detto in precedenza, le operazioni di *connect* e *disconnect* sulla porta seriale per l'invio delle stime da MATLAB™ al NM richiedono oltre 2 s; qualora il NM riprendesse ad inoltrare i pacchetti ricevuti dai NA prima del termine di quel periodo, tali dati non verrebbero registrati. È quindi necessario alzare a 4000 ms$_b$ il periodo tra una rilevazione e la successiva; con una configurazione di questo tipo, lo spostamento previsto tra un passo e l'altro è di ca. 20 m. Si è verificato sperimentalmente che in questo caso le prestazioni dello stimatore peggiorano, aumentando l'errore medio; tuttavia il peggioramento non è stato così pesante da richiedere la ricerca di una soluzione alternativa.

VI. BOZ ALGORITHM PER LA LOCALIZZAZIONE

A. Modellizzazione matematica del problema

La regione dove il nodo mobile (NM) è libero di muoversi è $\mathcal{X} \subset \mathbb{R}^2$. Gli L nodi àncora (NA) sono invece in posizioni fisse e note:

$$\mathbf{z}^{(l)} = \begin{bmatrix} z_1^{(l)} & z_2^{(l)} \end{bmatrix} \in \mathcal{X} \quad l = 1, \ldots, L$$

dove $z_1^{(l)}$ e $z_2^{(l)}$ rappresentano le coordinate nello spazio 2D rispetto al sistema di riferimento scelto: tali coordinate sono misurate direttamente in loco e quindi note con sufficiente precisione. Ad ogni istante temporale discreto, che indicheremo per semplicità di notazione con un pedice $t \in \mathbb{N}$, la posizione del NM è indicata con

$$\mathbf{x}_t = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \in \mathcal{X}.$$

$\mathbf{x}$ rappresenta lo stato da stimare che non è ovviamente direttamente misurabile ma è *nascosto* nelle L misure scambiate tra il NM e i NA ad ogni istante temporale:

$$\mathbf{y}_t = \begin{bmatrix} y_{1,t}, \ldots, y_{L,t} \end{bmatrix}^T \in \mathcal{X}$$

dove ciascuna $y_{l,t}$ rappresenta la potenza in dBm del segnale ricevuto (RSS) lungo l'l-esimo collegamento NA→NM. In particolare la misura della potenza può essere affetta da un offset costante purché uguale per tutte le misure $l = 1, \ldots, L$ e indipendente dal tempo: essendo la misura fatta dal NM, queste condizioni sono soddisfatte e si userà direttamente il valore di RSSI. Ecco che non risulta necessaria la taratura della misura di potenza. Da una descrizione statistica affidabile dell'RSSI è quindi possibile stimare lo stato $\hat{\mathbf{x}}_t$: questo sta alla base dei convenzionali approci di stima a massima verosimiglianza applicata alle misure $\mathbf{y}_t$. Similmente, basandosi su di un modello in di stato Markoviano, si può utilizzare un metodo bayesiano per la stima dell'intero movimento $\mathbf{x}_{1:t}$ non direttamente osservabile: in questo caso tutta la storia delle misure $y_{1:t}$ fino all'istante t corrente viene sfruttata da un algoritmo di *tracking* che perfeziona la probabilità a posteriori degli stati. Si rende necessaria una descrizione precisa del modello dell'RSS[16] per l'accuratezza della localizzazione.

1) Modello dell'RSS: Per quanto discusso in II-E.1 sull'inaffidabilità dei modelli che legano il valore ricevuto di RSS alla distanza trasmettitore-ricevitore, viene qui utilizzato un modello statistico log-normale i cui parametri verranno ricavati da misure sperimentali raccolte in fase di calibrazione (si veda VI-B). Ciascuna misura $y_{l,t}$ di RSS [dBm] è espressa nel modo seguente:

$$y_{l,t} = \bar{y}_l(\mathbf{x}_t) + v_{l,t} \qquad l = 1, \ldots, L \quad (5)$$

[16]Nel seguito i termini RSS e RSSI verranno utilizzati indistintamente alla luce di quanto prima osservato sull'offset che li lega reciprocamente

Il termine $\bar{y}_l(\mathbf{x}_t)$ è deterministico e indica l'attenuazione dovuta alla propagazione e alle ostruzioni statiche quali muri, pilastri, macchinari fissi ecc.: dipende ovviamente dalla posizione ed è quindi funzione dello stato all'istante corrente. Il rumore additivo è gaussiano a media nulla $v_{l,t} \sim \mathcal{N}\left(0, \sigma_l^2(\mathbf{x}_t)\right)$. La deviazione standard $\sigma_l(\mathbf{x}_t)$ tiene conto dell'aleatorietà del fenomeno dello *shadowing* dovuto al movimento delle persone nell'ambiente, allo spostamento di piccoli oggetti ecc.; $\sigma_l(\mathbf{x}_t)$ è una quantità deterministica funzione della posizione, rilevata sperimentalmente. Questo tipo di modellizzazione è stato validato con delle tecniche di *ray-tracing* [21] in cui il modello di propagazione consta di una componente deterministica e di una stocastica dovuta alla presenza di oggetti inseriti in modo *random* nell'ambiente. Si noti che le rilevazioni multiple mediate nel tempo riducono gli errori dovuti ad altri rumori additivi nonché a generiche interferenze. Si assume quindi di avere a disposizione una mappa completa di $\bar{y}_l(\mathbf{x}_t)$ e $\sigma_l(\mathbf{x}_t)$ per ciascuno stato plausibile, ottenuta attraverso l'interpolazione delle rilevazioni sperimentali secondo quanto verrà descritto in VI-C.

2) Modello di stato: La sequenza delle posizioni assunte dal NM è modellizzata come un processo di Markov del primo ordine, omogeneo; l'equazione di aggiornamento dello stato è una passeggiata aleatoria bidimensionale

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_t \,, \tag{6}$$

dove $\mathbf{v}_t$ è il processo di guida. Il modello statistico della evoluzione dello stato è quindi in accordo con la seguente

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = f_{\mathbf{v}}(\mathbf{x}_t - \mathbf{x}_{t-1}) \,,$$

con $f_{\mathbf{v}}$ densità di probabilità di $\mathbf{v}$. L'inizializzazione della distribuzione dello stato $p(\mathbf{x}_1)\ \forall\, \mathbf{x}_1 \in \mathcal{X}$ può essere scelta in base all'informazione disponibile a priori sulla posizione del NM. In particolare, sapendo che l'operatore entrerà da una porta, si può concentrare (con una distribuzione impulsiva) la probabilità su tale posizione (stato); altrimenti, in mancanza di informazione a priori, $p(\mathbf{x}_1)$ avrà distribuzione uniforme. Lo stato $\mathbf{x}_t$ è nascosto dalle osservazioni $\mathbf{y}_t \in \mathbb{R}^L$. Sulla base alla modellizzazione data nella (5) è possibile definire la probabilità del vettore $\mathbf{y}_t$ condizionata allo stato $\mathbf{x}_t$ supponendo indipendenti tra di loro le L misure: essa è un vettore di media

$$\bar{\mathbf{y}}_t = \begin{bmatrix} \bar{y}_1(\mathbf{x}_t) \\ \vdots \\ \bar{y}_L(\mathbf{x}_t) \end{bmatrix} \in \mathbb{R}^L$$

e varianza (matrice diagonale)

$$C(\mathbf{x}_t) = \begin{bmatrix} \sigma_1^2(\mathbf{x}_t) & 0 & \cdots & \\ 0 & \sigma_2^2(\mathbf{x}_t) & & \\ \vdots & & \ddots & \\ & & & \sigma_L^2(\mathbf{x}_t) \end{bmatrix} \in \mathbb{R}^{L\times L} .$$

Quindi la densità di probabilità (ddp) condizionata è facilmente

$$p(\mathbf{y}_t \,|\, \mathbf{x}_t) = \frac{1}{(2\pi)^{L/2}\,|C(\mathbf{x}_t)|^{1/2}}\, \mathrm{e}^{-\frac{1}{2}||\,\mathbf{y}_t - \bar{\mathbf{y}}_t||^2_{C^{-1}(\mathbf{x}_t)}} \tag{7}$$

con $||\mathbf{h}||_A^2 = \mathbf{h}^T A \mathbf{h}$. Se al tempo t il segnale del NA l non venisse ricevuto per vari motivi quali momentanea occlusione o spegnimento del nodo stesso, si pone $y_{l,t} := \bar{y}_l(\mathbf{x}_t)$ e $\sigma_l^2(\mathbf{x}_t) = K$ (con K costante) al fine di annullare il contributo di tale NA nel calcolo della probabilità condizionata.

Come già anticipato, una stima locale dello stato $\mathbf{x}_t$ si ottiene ad esempio applicando il criterio di massima verosimiglianza: $\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t \in \mathcal{X}} p(\mathbf{y}_t \,|\, \mathbf{x}_t)$. Tuttavia, secondo l'approccio bayesiano, si vuole sfruttare il modello di stato della (6) al fine di valutare la ddp a posteriori $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t})$ dato il set di misure $\mathbf{y}_{1:t}$. La ddp a posteriori dipende dalla ddp $p(\mathbf{y}_t \,|\, \mathbf{x}_t)$ (senza memoria) e dalla ddp a priori (con memoria) $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})$

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t \,|\, \mathbf{x}_t)\, p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1}). \tag{8}$$

La ddp a priori, utilizzando l'equazione di Chapman-Kolmogorov, diventa

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1}) = \int_{\mathcal{X}} p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})\, d\,\mathbf{x}_{t-1} \quad t > 1 \tag{9}$$

inizializzata con $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})\big|_{t=1} = p(\mathbf{x}_1)$. Una volta calcolata la ddp a posteriori attraverso la (8), la stima dello stato è ottenibile con il criterio di massimizzazione della probabilità a posteriori (MAP) oppure con il criterio MMSE (*Minimum Mean Square Error*).

3) Filtro particellare: A seguire si trova l'applicazione del filtro particellare al problema della stima della posizione: non si vuole fare una trattazione rigorosa della teoria che sta alla base del filtro ma darne una presentazione funzionale al problema in esame. Alcuni particolari sul filtro, metodi di stima diversi da questo e alcune tecniche implementative sono riportate in Appendice II; per altre varianti del filtro particellare e una trattazione più approfondita si rimanda a [22].

Il cuore del filtro particellare (FP) è l'approssimazione della ddp a priori $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})$ con una somma di S impulsi delta di Dirac, equamente pesati, centrati su un set di *particelle* $\{\mathbf{x}_t^{(s)}\}_{s=1}^S$. La differenza con gli algoritmi di *Detecting/Tracking* (D/TA) sta nella griglia degli stati considerati per il calcolo della stima $\hat{\mathbf{x}}_t$: nei D/TA il campionamento è regolare e fisso, mentre nel FP il campionamento è non uniforme. Ciascuna particella viene campionata da una variabile aleatoria con ddp a priori $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})$: brevemente si indica $\mathbf{x}_t^{(s)} \sim p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})$. Si assume quindi che per S sufficientemente grande la seguente approssimazione sia valida

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1}) \approx \frac{1}{S} \sum_{s=1}^{S} \delta(\mathbf{x}_t - \mathbf{x}_t^{(s)}). \tag{10}$$

Dalla (8) e dalla (10) si ottiene per la ddp a posteriori

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) \approx \sum_{s=1}^{S} w_t^{(s)}\, \delta(\mathbf{x}_t - \mathbf{x}_t^{(s)}) \tag{11}$$

dove ciascun peso $w_t^{(s)}$ è proporzionale alla probabilità di ricevere la misura $\mathbf{y}_t$ condizionata alla particella $\mathbf{x}_t^{(s)}$: $p(\mathbf{y}_t \,|\, \mathbf{x}_t^{(s)})$. Per le proprietà delle ddp essa dovrà sommare

a uno, quindi i pesi vanno opportunamente normalizzati affinché $\sum_s w_t^{(s)} = 1$. Secondo la (11) la dinamica della ddp a posteriori sarà quindi completamente identificata dalle t quantità $\{\mathbf{x}_t^{(s)}, w_t^s\}_{s=1}^S$. All'istante t-esimo la stima dello stato $\hat{\mathbf{x}}_t$ può essere calcolata secondo il criterio MMSE tramite l'aspettazione

$$\hat{\mathbf{x}}_t = \mathrm{E}[\mathbf{x}_t] \approx \sum_{s=1}^{S} w_t^{(s)}\ \mathbf{x}_t^{(s)}, \tag{12}$$

ovvero come media pesata delle particelle considerate. Per quanto riguarda l'avanzamento delle particelle e dei pesi dallo stato dall'istante t all'istante $t+1$ si procede con:

- *predizione dello stato*: consiste nell'estrazione di $\mathbf{v}_t^{(s)}$ (processo di guida) dalla sua ddp ($\mathbf{v}_{t+1}^{(s)} \sim f_{\mathbf{v}}$) per *ciascuna particella*; la predizione sarà quindi in accordo con la seguente

$$\mathbf{x}_{t+1}^{(s)} = \mathbf{x}_t^{(s)} + \mathbf{v}_{t+1}^{(s)}; \tag{13}$$

- *ricampionamento*: le particelle $\{\mathbf{x}_t^{(s)}\}_{s=1}^S$ sono ridistribuite in un nuovo set $\{\bar{\mathbf{x}}_t^{(m)}\}_{m=1}^S$ in modo che la probabilità associata ad uno stato rimanga inalterata:

$$\mathrm{P}[\bar{\mathbf{x}}_t^{(m)} = \mathbf{x}_t^{(s)}] = w_t^{(s)} \quad \forall\ m, s = 1, \ldots, S.$$

  Con questa operazione si evita di giungere ad una situazione degenere nella quale rimangono poche particelle con peso molto elevato vanificando il principio di funzionamento del filtro (i.e. propagare l'intera descrizione statistica dello stato attraverso un *significativo* campionamento). Vengono quindi create più particelle *(split)* a partire da una con peso elevato[17] mentre quelle con peso trascurabile vengono scartate. I particolari implementativi di questo filtro particellare detto *Sequential Importance Resampling (SIR)* sono riportati nell'Appendice II.

B. Rilevazioni a priori

Per un corretto funzionamento dell'algoritmo è necessario procedere con una serie di rilevazioni da effettuare direttamente sul campo; utilizzando la disposizione dei nodi àncora ricavata in IV-C si memorizza il valore di RSSI ricevuto da un nodo posizionato in n diverse postazioni.

Considerando l'area totale dell'ambiente, si decide di distanziare ogni singola rilevazione di 3 m circa, ottenendo la mappa delle misure visualizzata in Fig. 30 (le coordinate delle posizioni si possono trovare in IV in Tab. 4); il numero delle rilevazioni risulta $n = 115$. Il nodo mobile viene lasciato in ogni posizione per 100 secondi, in cui effettua 100 intervalli di ascolto della durata di 1 s: le varie informazioni che giungono dalla rete vengono salvate e riportate al pc tramite l'applicazione `SaveLog` che permette la creazione di un file testo, come mostrato in Fig. 31 (per i dettagli si veda V-E.2).

Una successiva fase di interpolazione dei dati (vedi VI-C) permetterà di ottenere una griglia delle rilevazioni con un passo ridotto a soli 0.5 m.

[17]Questo introduce varietà poiché nella successiva fase di predizione ogni particella subirà un processo di guida diverso dalle altre in accordo con la (13).

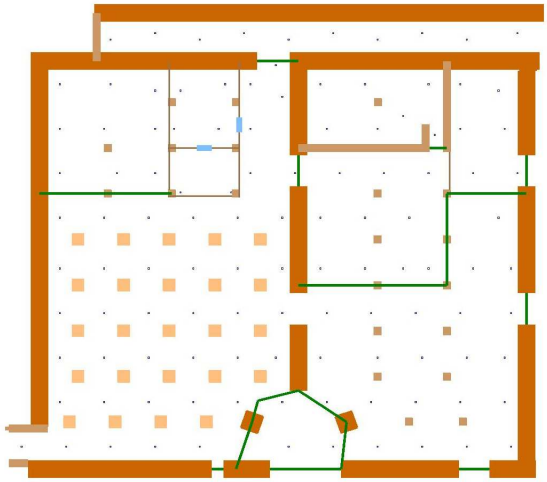


Fig. 30: Mappa dei sotterranei in cui sono evidenziate le posizioni delle singole rilevazioni effettuate per creare la mappa *a priori*.

logRx1 - Blocco note

File Modifica Formato Visualizza ?

RSSI	SENDER	X_COORD	Y_COORD	TX_POWER
'e7'	'4'	'3b'	'1c'	'1f'
'f7'	'3'	'35'	'2c'	'1f'
'd8'	'1'	'25'	'24'	'1f'
'd5'	'5'	'49'	'28'	'1f'
'd3'	'2'	'29'	'46'	'1f'
'e8'	'4'	'3b'	'1c'	'1f'
'd5'	'2'	'29'	'46'	'1f'
'f5'	'3'	'35'	'2c'	'1f'
'd6'	'1'	'25'	'24'	'1f'
'd4'	'5'	'49'	'28'	'1f'
'e7'	'4'	'3b'	'1c'	'1f'
'f2'	'3'	'35'	'2c'	'1f'

Fig. 31: Esempio di file testo con le informazioni sulle rilevazioni.

Si osserva che

$$\frac{\text{area totale}}{\text{numero rilevazioni}} \simeq 10$$

e

$$\frac{\textit{mote} \text{ utilizzati}}{\text{area totale}} \simeq 0.01$$

sono rapporti molto simili a quelli usati in [6]: il dimensionamento della soluzione è quindi adeguato al problema.

C. Elaborazione ed interpolazione dei dati delle rilevazioni

Per ciascun punto di rilevazione è quindi stato raccolto un file di log in formato testo contenente in ciascuna riga le informazioni del pacchetto ricevuto: in questo contesto interessano solamente l'identificativo (ID) del nodo trasmittente e il valore di RSSI associato a tale pacchetto. In MATLAB™ la funzione `elaborazLog`, rispettivamente:

1) per ciascuna riga (pacchetto) converte la stringa con caratteri in formato esadecimale il valore di RSSI ricevuto eliminando eventuali valori non ammissibili (i.e. fuori dal range definito in V-A.1);
2) raccoglie in un vettore gli ID dei NA ricevuti (non è detto che una particolare posizione siano ricevuti tutti i NA) scartando eventuali ID errati;

3) per ciascun NA ricevuto (ID raccolto) vengono memorizzati i valori di RSSI corrispondenti (calcolati al punto 1), calcolata la media e la varianza, calcolata la percentuale di pacchetti ricevuti sul totale delle rilevazioni;
4) calcola il numero totale di pacchetti raccolti e quello dei *fake*.

I dati forniti dalla funzione `elaborazLog` per ciascun punto di rilevazione (i.e. per ciascun file di log) vengono elaborati nella funzione `campioniIrregolari` e raccolti in una struttura dati tridimensionale (`mappa`) in cui le righe corrispondono all'ID dei NA, le colonne alle rilevazioni e in profondità sono collocate media e varianza (per il significato di questi dati si veda VI-A). Quando un NA non è ricevuto, i.e. la potenza del segnale è inferiore alla sensibilità del chip radio (ca. -60 dBm), due sono le possibili motivazioni: presenza di un ostacolo (ostruzione) che impedisce la propagazione (anche NLOS) o naturale decadenza della potenza per effetto dell'attenuazione dello spazio libero (distanza eccessiva). In situazioni di questo tipo si potrebbe procedere analiticamente calcolando il valore atteso di RSSI mediante un modello di propagazione del segnale, e.g. (1); analogamente per la varianza dopo aver identificato un modello sulla base delle misure a disposizione. Tuttavia tale approccio appare inutilmente dispendioso, anche tenendo conto della necessità di una calibrazione specifica per ogni ambiente. Per semplificare le cose, si utilizza un modello di propagazione troncato: ogni valore di potenza inferiore ai -60 dBm viene posto a -70 dBm, mentre la varianza viene posta a 25 dBm^2. Il valore di $\sigma_l^2(\mathbf{x})$ è stato scelto tra quelli più alti ottenuti dalle rilevazioni poiché la perdita d'informazione dovuta al troncamento non è trascurabile e allo stesso tempo valori ancora maggiori creerebbero problemi in fase di interpolazione.

In Fig. 32 è riportata la mappa dell'RSSI medio e della rispettiva varianza del segnale ricevuto dal NA 1.

Ora, a partire dalla `mappa`, che riporta i parametri di ricezione nelle 115 rilevazioni sul campo per ciascun NA, è necessario ricavare una mappa che contempli un valore di media e varianza *per ogni stato* $\mathbf{x} \in \mathcal{X}$. L'insieme $\mathcal{X}$, definito in VI-A, è una griglia 2D regolare con componenti

$$x_1 \in \{1, 2, \ldots, 82\} \quad x_2 \in \{1, 2, \ldots, 74\}, \tag{14}$$

costruita sulla base di considerazioni legate alla precisione richiesta dall'algoritmo. Lo stato rappresenta quindi la posizione in passi di 0.5 m nel sistema di riferimento con origine l'angolo Sud-Ovest e asse x_1 rivolto verso Est; l'ambiente coperto dalla griglia ha quindi estensione di longitudine 40.5 m e latitudine di 36.5 m. Quindi, dato lo stato, per calcolare la posizione secondo tale sistema di riferimento è sufficiente togliere 1 e dividere la componente per 2. L'interpolazione dei dati è stata condotta separatamente per la media dell'RSS ricevuto $\bar{y}_l(\mathbf{x})$ e per la varianza $\sigma_l^2(\mathbf{x})$ per ciascun mote ottenendo quindi un totale di 24 mappe. La tecnica d'interpolazione usata è presentata nel dettaglio nell'Appendice I: si usa uno stimatore lineare che per ciascuna posizione utilizza tutti i dati raccolti nelle rilevazioni. In questo paragrafo vengono invece riassunti i valori dei parametri scelti e le prove effettuate.

Attraverso lo script MATLAB™ `coordinateMote` vengono raccolte in due vettori le coordinate $[z_1^{(l)} \; z_2^{(l)}] \in \mathcal{X}$, $l = 1, \ldots, L$ dei NA secondo la notazione introdotta in VI-A e attenendosi alla formulazione del problema dell'interpolazione in Appendice I. In `campioniIrregolari` vengono definite le strutture tridimensionali che ospiteranno i dati stimati `RSS_stim` (e `var_stim`): considerando la profondità n-esima di tali matrici, si trova la mappa 2D - per righe la prima coordinata dello stato, per colonne la seconda coordinata - del segnale ricevuto (varianza) dal NA n-esimo.

Per la scelta della funzione di correlazione si è fatto riferimento all'articolo [23], supponendo, in aggiunta a quanto ipotizzato in nell'Appendice I, che la funzione di correlazione sia isotropica

$$r(\xi, \eta) = r(d) \quad \text{con} \quad d = \sqrt{\xi^2 + \eta^2}$$

. La superficie interpolata dipende molto dalla funzione scelta $r(d)$: sono state provate le seguenti

$$r(d) = \exp(-d/d_0), \tag{15}$$

$$r(d) = \exp\left[(-d/d_0)^2\right], \tag{16}$$

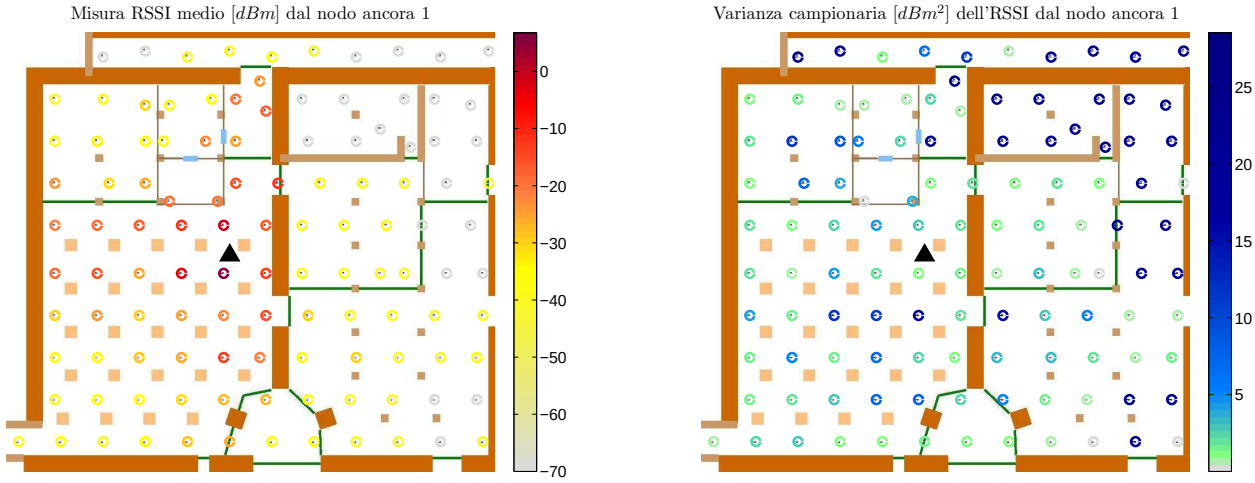
$$r(d) = 1/(1 + d^2/d_0^2)^{3/2}; \tag{17}$$

in particolare la (15) è quella che da i migliori risultati per il tipo di superficie che si sta interpolando. Il parametro d_0 sta ad indicare il peso che hanno nella stima i valori rilevati in relazione alla distanza dal punto in cui si vuole calcolare la superficie: valori elevati di d_0 producono superfici molto *smoothed* mentre valori bassi tendono a creare delle stime frastagliate (i dati "distanti" sono pesati molto poco). Tra le varie prove effettuate si riporta quanto osservato dalle più significative per la stima dell'RSS:

- con $d_0 \geq 30$ si stima una mappa troppo smoothed, che onora il dato[18] ma tende ad "appiattire" la superficie;
- con $d_0 \leq 10$ si ha la comparsa di un'attenuazione generalizzata della mappa fuori dai punti di misura (rappresentando la stima in un plot 3D si notano dei rialzamenti della superficie in corrispondenza di ciascun dato rilevato);
- con $d_0 = 20$ si ottiene la soluzione migliore.

Per quanto concerne la varianza si osservano dei comportamenti del tutto simili alla mappa dell'RSS in riferimento alla scelta di d_0; in particolare si è provato ad aggiungere della luce bianca ($\sigma_w^2 \neq 0$ nella (30)) come descritto in Appendice I e consigliato in [23]: valori accettabili di σ_w^2 sono nell'ordine di 0.1, 0.2 dBm^2. Si osserva in questi ultimi casi che la stima non onora più il dato, come ci si aspettava; l'aspetto della superficie non presenta grandi miglioramenti da motivare l'uso di σ_w^2. Per l'interpolazione della varianza si osserva inoltre che, per alcuni NA, la mappa risultante assume dei valori negativi che, da un punto di vista statistico, non hanno ovviamente senso. Tale fenomeno nasce dal fatto che in alcune zone le varianze rilevate assumono valori elevati (oltre 20 dBm^2) mentre in punti vicini è nell'ordine di qualche decimo: l'interpolazione produce una superficie che, per raccordare con

[18] Per quanto osservato nell'Appendice I.



(a) Rilevazioni RSSI medio ricevuto dal NA n.1 **(b)** Rilevazioni varianza RSSI ricevuto dal NA n.1

Fig. 32: Rilevazioni segnale dal NA numero 1

continuità i dati, ha una forte pendenza che può portare alla stima di valori negativi nei punti immediatamente prossimi. Per evitare questi risultati si è forzata la mappa ad assumere solamente valori non negativi ponendo $\sigma^2_{min} = 0.01$ dBm^2: σ^2_{min} è posta tale sulla base della varianza minima rilevata (dati effettivamente misurati). In Figg. 33 e 34 sono riportati i risultati dell'interpolazione dell'RSSI e della varianza relativi al *mote* 1; in I compaiono le mappe interpolate per tutti i NA.

Operando una interpolazione diretta dei dati memorizzati si è giunti, in un primo momento, a dei risultati non veritieri sul reale andamento della potenza del segnale trasmesso: in prossimità delle pareti perimetrali l'interpolazione portava ad avere valori di RSSI del segnale trasmesso da un nodo àncora non conformi al modello matematico di II-E che prevede una decadenza della potenza con la distanza. Infatti si era giunti ad avere una superficie interpolata che cresceva muovendosi dalla rilevazione in prossimità del perimetro verso il muro stesso. Una motivazione può essere ricercata nel fatto che l'interpolazione opera un raccordo globale dei dati, per cui un *trend* di crescita lieve può portare ad un aumento considerevole in una zona in cui mancano le misure. Per ovviare a questo si sono estesi i valori di RSSI e varianza delle rilevazioni in corrispondenza delle pareti sud ed ovest: semplicemente sono state create delle rilevazioni "fittizie" posizionate lungo l'asse x e l'asse y (delle coordinate della mappa) i cui valori sono gli stessi delle rilevazioni immeditamente prossime tra quelle sperimentali.

D. Processi di guida

La modellizzazione del problema fornita evidenzia che il processo di guida introdotto nella (6) assume notevole importanza nella costruzione di tutto il filtro particellare per il problema della localizzazione. Infatti, nel processo Markoviano dello stato il rumore $\mathbf{v_t}$ rappresenta il numero di passi[19] compiuti in un intervallo di campionamento nella componente x_1 e nella componente x_2: si vuole quindi che il processo $\mathbf{v}_t$ sia il più possibile rappresentativo del movimento del NM di cui si vuole stimare la posizione. Poichè il NM sarà solidale con il movimento dall'operatore di soccorso, la velocità massima supposta in una qualsiasi direzione è di ca. 5 m/s; rimane da dare una descrizione statistica della dislocazione spaziale. In questo contesto è più agile lavorare con le coordinate polari:

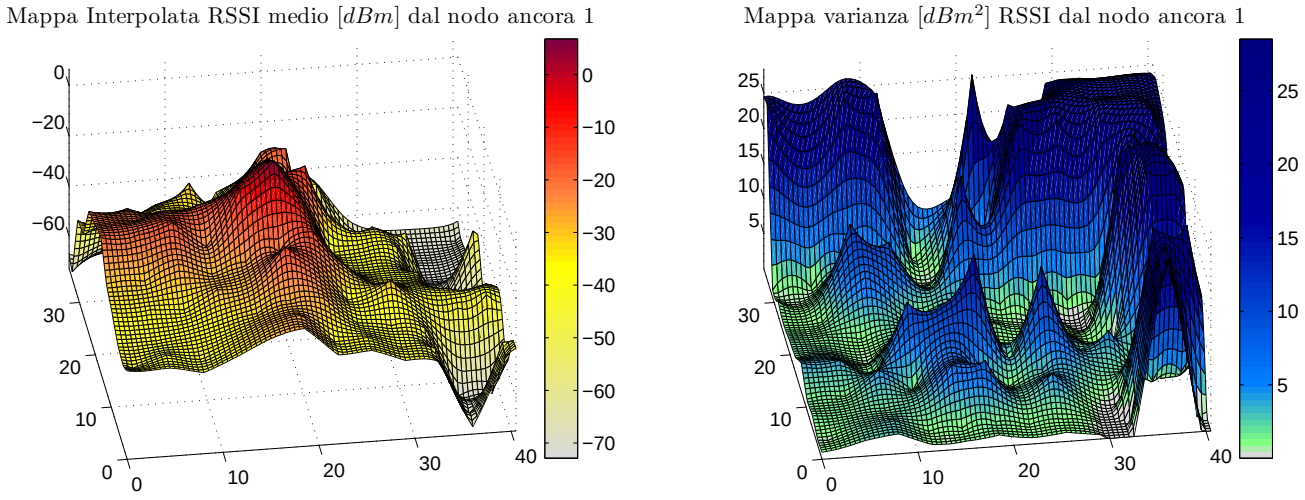
$$\rho = \sqrt{x_1^2 + x_2^2}, \qquad \theta = \arctan \frac{x_2}{x_1}.$$

La prima ddp considerata è una gaussiana 2D centrata nell'origine: il numero di passi aggiunto all'istante t è quindi con alta probabilità nullo e può assumere valori via via più elevati in ogni direzione con probabilità sempre minore; si veda la Fig. 35a. In coordinate polari è equivalente ad avere una distribuzione di tipo gaussiana unidimensionale (di media nulla e di varianza σ_v) lungo la direzione di ρ e una distribuzione uniforme per θ

$$f_\rho(x_1, x_2) = \frac{1}{\sqrt{2/\pi}\,\sigma_v} \mathrm{e}^{-\frac{\rho^2}{2\sigma_v^2}}, \qquad f_\theta = \frac{1}{2\pi}. \tag{18}$$

Prendendo spunto dal lavoro di Spagnolini *et al.* in [24] una distribuzione che per certi versi è più attinente alla modellizzazione del movimento del soccorritore è una *uniform ring*: per essa la gaussiana disposta lungo la direzione di ρ è centrata su di un valore non nullo detto raggio. Il raggio rappresenta la velocità media che ci si aspetta dal processo di guida che evidentemente ora avrà bassa probabilità di fornire valori di $\mathbf{v}_t$ nulli (operatore fermo). La ddp è del tutto identica alla (18) tranne per la media della normale che ora è pari al raggio dell'anello. Si veda Fig. 35b.

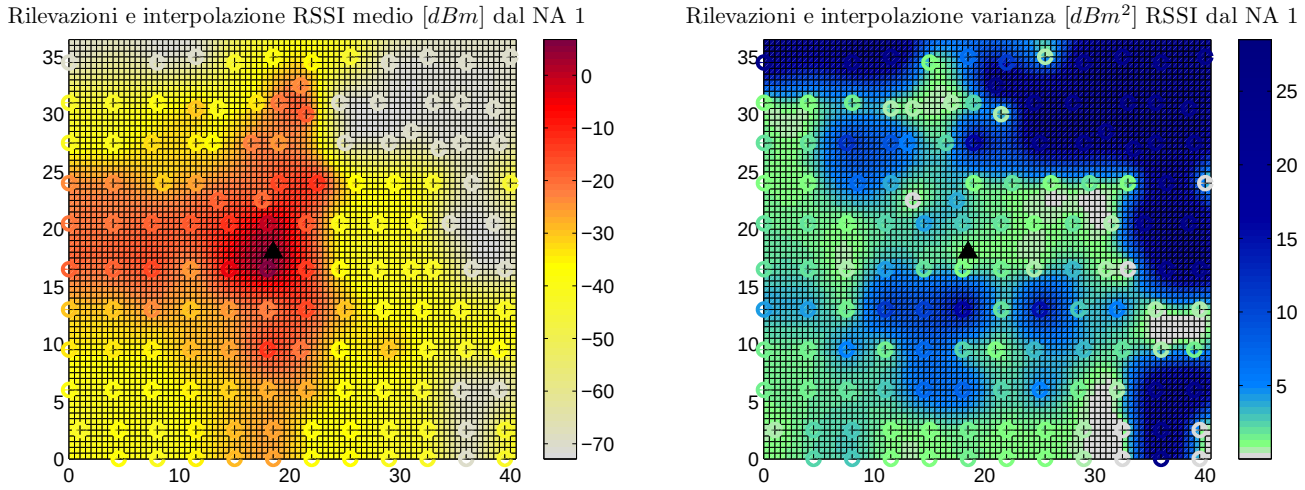
[19] Secondo la griglia $\mathcal{X}$ definita in (14).



(a) Mappa RSSI medio ricevuto dal NA n.1

(b) Mappa varianza RSSI ricevuto dal NA n.1

Fig. 33: Mappe ottenute da interpolazione dei dati ricevuti dal NA numero 1



(a) Mappa RSSI medio ricevuto dal NA n.1

(b) Mappa varianza RSSI ricevuto dal NA n.1

Fig. 34: Confronto dati rilevati e mappe ottenute per interpolazione.

Pensando al fatto che una normale ammette, per quanto a minima probabilità, valori anche molto elevati di ρ si è pensato di sostituire la gaussiana con una distribuzione che ammetta probabilità non nulla solamente in un intervallo prestabilito. La distribuzione beta è una famiglia di curve parametrizzate da a e b, valori che ne conferiscono la forma rispettivamente nel primo e nel secondo tratto.

$$p(\rho|a,b) = \frac{1}{\mathrm{B}(a,b)}\rho^{a-1}(1-\rho)^{b-1}\,\mathrm{I}_{(0,1)}(\rho), \qquad (19)$$

con $\mathrm{B}(a,b)$ funzione beta

$$\mathrm{B}(a,b) = \int_0^1 \tau^{a-1}\,(1-\tau)^{b-1}\,d\tau = \frac{\Gamma(a)\,\Gamma(b)}{\Gamma\,a+b}, \qquad (20)$$

$$\Gamma(c) = \int_0^\infty \mathrm{e}^{-\sigma}\;\sigma^{c-1}\,d\sigma,$$

e $\mathrm{I}_{(0,1)}(\rho)$ funzione rivelatrice:

$$\mathrm{I}_{(0,1)}(\rho) = \begin{cases} 1 & \text{se } \rho \in [0,1] \\ 0 & \text{altrimenti.} \end{cases}$$

La distribuzione beta è molto flessibile: può rappresentare le distribuzioni uniforme (beta degenere con $a = b = 1$), triangolare, Poisson, log-normale, gaussiana e altre semplicemente agendo sui parametri a e b per determinare la forma e sull'intervallo d'integrazione della funzione Beta (nella (20)) per modificare il range di definizione. In figura 35c è riportato il risultato della costruzione di una ddp 2D con una distribuzione beta lungo ρ opportunamente scalata tra $[0\ \rho_{MAX}]$ e una ddp uniforme lungo θ. Per la particolare conformazione della beta (i.e. selezione tra la famiglia parametrica) si è usato il toolbox di MATLAB™ `disttool`: si è voluto dare

una probabilità elevata all'evento $\mathbf{v}_t$ tra zero e ca 2 passi [20] e probabilità via via decrescente fino al valore massimo della beta (che rappresenta la massima velocità che si suppone possa avere l'operatore). Il risultato è in Fig. 35e: si noti che tale distribuzione è altamente non simmetrica.

Infine, considerando che il percorso compiuto da una squadra di soccorso avrà una bassa probabilità di avere cambi repentini di direzione, si è voluto riportare tale caratteristica alla ddp 2D di $f\mathbf{v}_(x_1, x_2)$: per contemplare una probabilità maggiore di estrazione di un $\mathbf{v}_t$ che mantenga la direzione che il NM aveva al passo $t-1$, è necessario che lungo la direzione di θ la distribuzione sia del tipo gaussiana (o eventualmente beta simmetrica) di media pari all'angolo della direzione al passo precedente θ_{-1}. Per la deviazione si è pensato inizialmente ad un valore che permettesse di ottenere ca il 99% dei valori tra $[-\pi, \pi]$: $\sigma_\theta = \pi/3$; tuttavia a livello grafico la probabilità di tutto il semicerchio opposto alla direzione di guida sembrava troppo penalizzante con tale valore di σ_θ, per cui si è tarata la deviazione ad un valore lievemente superiore (Fig. 35d).

Per la costruzione delle Fig. 35 si considera il prodotto delle densità lungo ρ e θ considerate indipendenti; in particolare si presenta il problema della normalizzazione (integrale della ddp 2D unitario) che si pone nel momento in cui vi possono essere più punti $(\mathbf{x})$ mappati dalla stessa coppia (ρ, θ)[21]. Tale problema non si verifica nel momento in cui, a livello algoritmico, si estrae dalla distribuzione scelta il valore di $\mathbf{v}_t$ attraverso le funzioni MATLAB™ preposte[22] (e opportunamente parametrizzate): si converte da coordinate polari in cartesiane (`pol2cart`) ottenendo direttamente il valore di $\mathbf{v}_t$ senza la necessità di normalizzare.

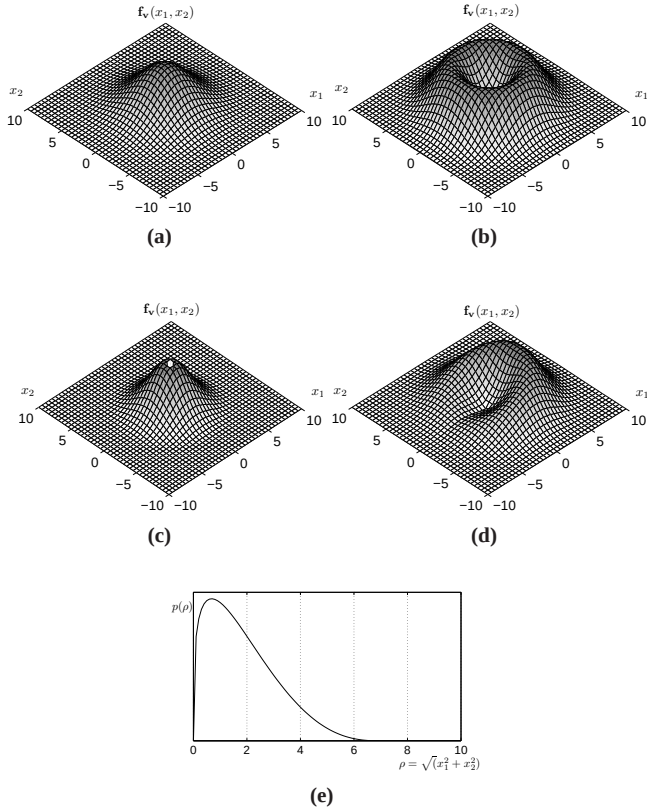


Fig. 35: Processi di guida 2D: densità di probabilità da cui viene estratto $\mathbf{v}_t$ (NB: le unità di misura delle coordinate sono in passi sulla griglia $\mathcal{X}$).
(a) Gaussiana circolare con $\sigma_v = 3$.
(b) Anello uniforme con $\sigma_v = 2$ e raggio $r = 4$.
(c) Beta circolare con parametri $a = 1.28$, $b = 3.6$, $\rho_{MAX} = 7$.
(d) Gaussiana lungo la direzione di ρ con $\sigma_v = 2$ e raggio 4 - come in (b)- e gaussiana lungo la direzione di θ con $\sigma_\theta = \pi/2$ e media (direzione al passo precedente) $\theta_{-1} = 0$.
(e) Sezione lungo la direzione di ρ della pdf Beta di fig.(c).

E. Simulazioni

Per la generazione di una traiettoria simulata in MATLAB™[23] si è implementata la passeggiata aleatoria definita nel modello markoviano in (6) con le varie distribuzioni di $f_\mathbf{v}$ definite nel paragrafo VI-D. A questo proposito si pone il problema, solo in fase di simulazione, di forzare la *random walk* a rimanere all'interno dell'ambiente considerato (o equivalentemente alla griglia $\mathcal{X}$ su cui è definita la mappa a priori con l'interpolazione delle misure sperimentali). Per non appesantire a livello computazionale le simulazioni si è adottata una soluzione semplice: alla generazione della componente i-esima dello stato in t, $x_{i,t} = x_{i,t-1} + v_{i,t}$, se essa dovesse fuoriuscire dalla griglia si inverte il segno del processo di guida ottenendo $x_{i,t} = x_{i,t-1} - v_{i,t}$.

Per la simulazione delle misure si utilizza la stessa mappa di RSSI e varianza ottenute dall'interpolazione in VI-C: per ciascun NA, dato lo stato vero $\mathbf{x}_t$, si accede alla mappa a priori leggendo il corrispondente valore di RSSI $\bar{y}_l(\mathbf{x}_t)$; ad esso viene aggiunto un rumore gaussiano di varianza $\sigma_l(\mathbf{x}_t)$ rilevata nella mappa. In tal modo si simula quindi il modello di misura definito nella (5) ottenendo il vettore delle misure $\mathbf{y}_t$.

Una prima prova[24] è stata condotta per valutare le prestazioni del filtro al variare del numero delle particelle usate per campionare la densità a posteriori (si veda l'Appendice II). Per quanto riguarda il ricampionamento, in prima istanza, si usa una versione SIR (algoritmo 1) i.e. con ricampionamento sistematico; successivamente sono state provate le versioni più "parsimoniose" dal punto di vista computazionale, come l'algoritmo 2 in cui il campionamento viene eseguito in vicinanza della degenerazione del FP. Tuttavia, impostando la soglia S_T anche a valori elevati, in alcune occasioni si arriva comunque alla degenerazione (molto probabilmente per l'approssimazione usata nel calcolo di $\hat{S}_{eff}$); per tale motivo, si è deciso di usare il SIR sistematicamente, anche considerando che il ricampionamento impiega un tempo di calcolo non elevato: $O(S)$. Per quanto riguarda la tecnica di ricampionamento, inizialmente si è implementata la versione dell'algoritmo 4 e i risultati sono stati soddisfacenti; migliori

[20] Il tutto è ovviamente riferito al periodo di campionamento.

[21] In seguito alla conversione con la funzione `cart2pol` della griglia su cui vengono rappresentate le $f_\mathbf{v}(x_1, x_2)$.

[22] `randn` per la densità normale e `betarnd` per la beta.

[23] Script `generaTraiett`.

[24] Script `algoritmo`.

prestazioni si ottengono dall'algoritmo 3 che sarà quello usato in tutte le prove seguenti. Il filtro particellare per le simulazioni è implementato nella funzione `particleFilter` che viene chiamata dallo script `algoritmo`; i parametri di cui necessita sono: la sequenza delle misure $\mathbf{y}_{1:T}$, la traiettoria vera (per produrre il plot di confronto con la stima), una stringa che indica la densità $f_{\mathbf{v}}$ da usare in predizione[25] e i relativi parametri. Seguendo l'ordine delle operazioni nell'algoritmo 1, le funzioni MATLAB™ a loro volta istanziate da `particleFilter` sono[26]:

- `predizioneOneSample`: esegue il passo di predizione di una particella;
- `pesatura`: assegna i pesi a tutte particelle e relativa normalizzazione;
- `aggiornamento`: ricampiona tutte le particelle secondo l'algoritmo 3;

Le particelle del filtro vengono inizializzate (in assenza di altra informazione) con distribuzione uniforme all'interno della griglia $\mathcal{X}$:

$$\mathbf{x}_{t=1}^{s} \sim \mathcal{U}[\mathcal{X}], \quad s = 1, \ldots, S.$$

Per la passeggiata aleatoria lo stato è inizializzato in un punto arbitrario circa nel mezzo dell'ambiente: $\mathbf{x}_1 = [40\ 40]'$.

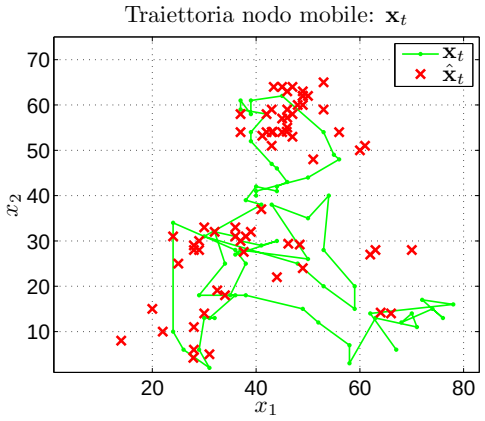
In Fig. 36 sono riportati i risultati ottenuti da una simulazione di 70 passi con processo di guida come in Fig. 35a con $\sigma_v = 7$: così facendo il 95.5% ca. dei passi aggiunti ad ogni istante ha norma $||\mathbf{v}_t||$ minore di $2\sigma_v = 14$ (i.e. 7 m). Se si suppone che il tempo di campionamento sia di 1 s, si ottiene una velocità massima di spostamento dell'operatore sui 7 m/s, con alta probabilità che invece sia prossima a zero. Il filtro predice la posizione al tempo $t+1$ con un modello accordato al processo di generazione della traiettoria, cioè con la stessa ddp di guida e con la tecnica di cui sopra per il contenimento dello stato all'interno della frontiera della griglia $\mathcal{X}$.

In riferimento alla Fig. 36 con $S = 10$ si vede chiaramente che la stima è inadeguata, con $S = 50$ la precisione nella posizione stimata è molto scarsa ma almeno il PF insegue la traiettoria. Migliori risultati si ottengono con almeno un centinaio di particelle: la differenza tra 200 e 500 campioni è molto lieve, quindi si intuisce che l'aumento ulteriore di S non comporta precisioni tali da giustificarne la complessità di calcolo. In Fig. 37 è riportato l'andamento nel tempo della distanza tra posizione vera e stimata, mediando su tre istanze del filtro sulla medesima traiettoria: il filtro ha prestazioni considerevolmente diverse in ogni realizzazione del processo di guida estratto nella fase di predizione; il confronto di Fig. 37 è quindi semplicemente qualitativo. Infine, per giustificare l'aumento di particelle usate per il campionamento della ddp a posteriori, si è confrontato in Fig. 38 l'errore medio (nei 70 passi di simulazione) al variare di S; la complessità di calcolo[27] si vede crescere approssimativamente in modo lineare.

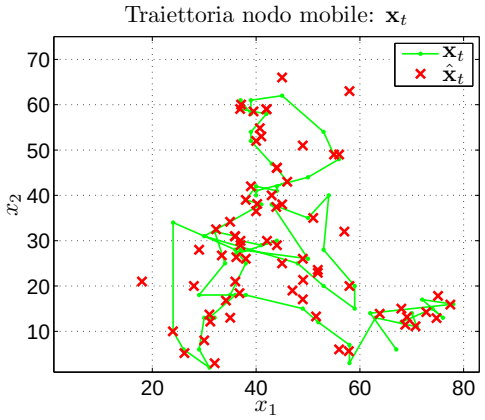
[25] Questo parametro permette di selezionare il processo di guida senza dover disporre di una funzione per ciascuna $f_{\mathbf{v}}$.

[26] Le funzioni di seguito indicate sono le versioni base, sono state fatte poi tutta una serie di varianti per gestire le simulazioni con NA guasti e le altre varianti.

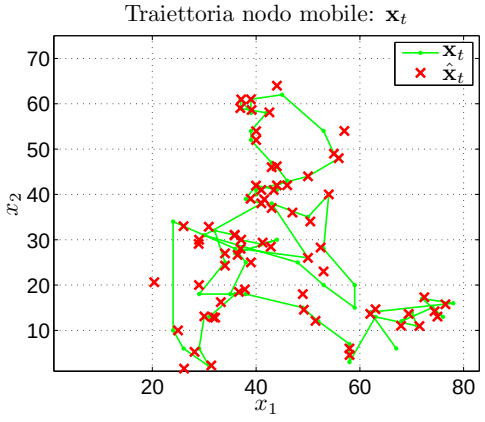
[27] I valori in figura si devono intendere solo indicativi (di confronto) poiché il filtro in MATLAB™ deve gestire anche delle figure che ne rallentano pesantemente l'esecuzione.



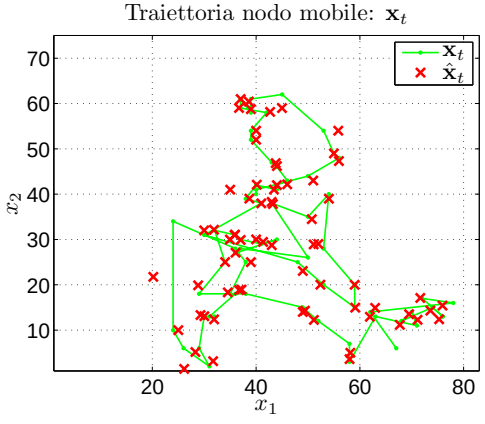
(a) $S = 10$



(b) $S = 50$



(c) $S = 200$



(d) $S = 500$

Fig. 36: Confronto traiettoria, generata con ddp 2D gaussiana (con $\sigma_v = 7$ e media nulla), con le posizioni stimate (con FP perfettamente accordato) al variare del numero di particelle S

Alla luce di queste considerazioni si sceglie di implementare un filtro a 500 *samples*. Si vuole sottolineare il fatto che (ad esempio in riferimento al caso con $S = 50$), nonostante alcune stime siano molto poco accurate (lontane più di 15 passi dalla posizione reale), il tracking è comunque garantito (i.e. nei passi successivi il filtro non diverge ma produce delle stime soddisfacenti), purché il numero di campioni sia sufficientemente elevato.

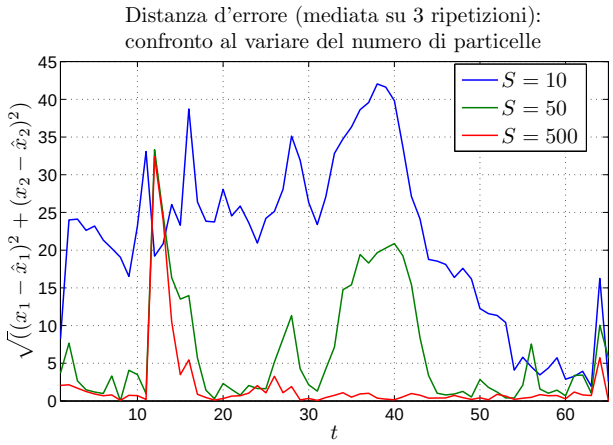


Fig. 37: Confronto errore (distanza euclidea mediata su tre istanze del filtro) di stima al variare del numero di particelle S.

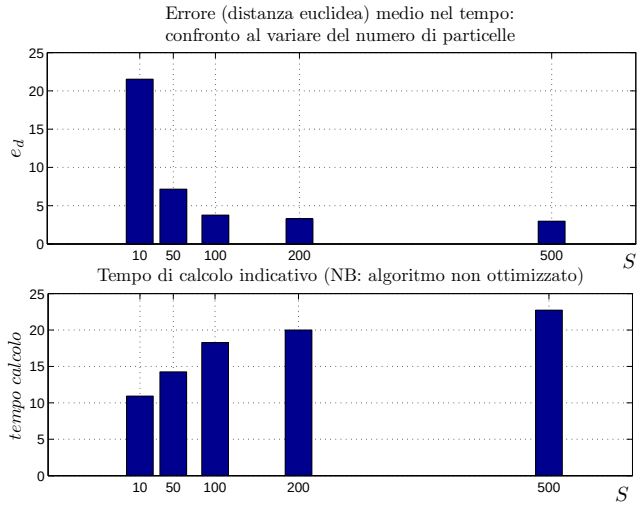


Fig. 38: Confronto errore medio (distanza euclidea) di stima e tempo di esecuzione del filtro al variare del numero di particelle S.

Si riporta, per meglio comprendere il funzionamento del FP, la Fig. 39 con la visualizzazione delle particelle $\{\mathbf{x}_{\bar{t}}^s\}_{s=1}^S$ al tempo $\bar{t}$ dopo la fase di pesatura, prima del ricampionamento; i pesi $\{w_{\bar{t}}\}_{s=1}^S$ normalizzati assegnati ai samples sono rappresentati mediante la colorazione dei campioni.

Per valutare gli effetti della densità $f_\mathbf{v}$ contenuta nel modello del FP sono state fatte delle stime con diversi filtri sulla stessa traiettoria: il risultato è riportato in Fig. 40: si vede che le prestazioni sono del tutto equivalenti per le distribuzioni 2D che non tengono memoria della direzione (anche perché sono parametrizzate in modo da avere una "forma" non molto dissimile tra loro). Il FP con densità di Fig. 35d ha "perso" la traiettoria in più occasioni, producendo infatti un errore medio molto elevato: ciò è giustificato dal fatto che la traiettoria vera era generata da un processo isotropo.

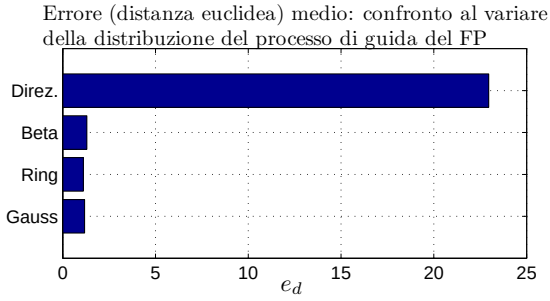


Fig. 40: Errore medio (distanza) con vari FP sulla stessa traiettoria generata con ddp $f_\mathbf{v}$ gaussiana 2D (come in Fig. 35a con $\sigma_v = 7$). Nel FP il processo usato in predizione è di tipo:
"Gauss": $f_\mathbf{v}$ come nella traiettoria ($\sigma_v = 7$);
"Ring": $f_\mathbf{v}$ come in Fig. 35b ($\sigma_v = 7$, raggio $r = 4$);
"Beta": $f_\mathbf{v}$ come in Fig. 35c ($a = 1.28$, $b = 3.6$, $\rho_{MAX} = 10$);
"Direz.":$f_\mathbf{v}$ con memoria della direzione, come in Fig. 35d ($\sigma_\theta = \pi/2$).

Nel momento in cui la traiettoria è generata con un processo di guida che ha memoria della direzione, come si potrebbe supporre sia il movimento di un operatore umano, il FP accordato con essa ha ovviamente prestazioni migliori rispetto a quello in cui $f_\mathbf{v}$ è isotropa. Tuttavia le differenze sono lievi e, considerando le già buone prestazioni mostrate in Fig. 41, si decide per l'uso di un processo di predizione con densità di probabilità isotropa.

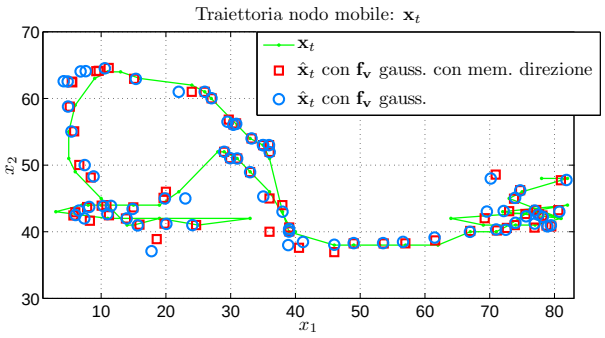


Fig. 41: Confronto prestazioni FP con $f_\mathbf{v}$ di Fig. 35a ($\sigma_v = 7$) e con $f_\mathbf{v}$ con memoria della direzione, come in Fig. 35d ($\sigma_\theta = \pi/2$); traiettoria generata con memoria della direzione.

Al fine di testare la robustezza dell'algoritmo nel caso di malfunzionamenti nella rete, sono state fatte delle simulazioni in cui venivano spenti alcuni NA; a tale scopo è stato sufficiente inserire nello script MATLAB™ `pesatura` l'ID dei mote spenti: in maniera analoga ai NA il cui segnale non è ricevuto, il loro contributo nel calcolo della (7) verrà annullato tramite la forzatura dell'RSSI letto dalla mappa a priori:

$$\bar{y}_j \leftarrow y_j\,, \quad j \in \{\text{NA non ricevuti}\}.$$

In Fig. 42 è riportato l'errore di stima medio ottenuto nelle diverse simulazioni: tutti i NA accesi, un nodo ancora spento

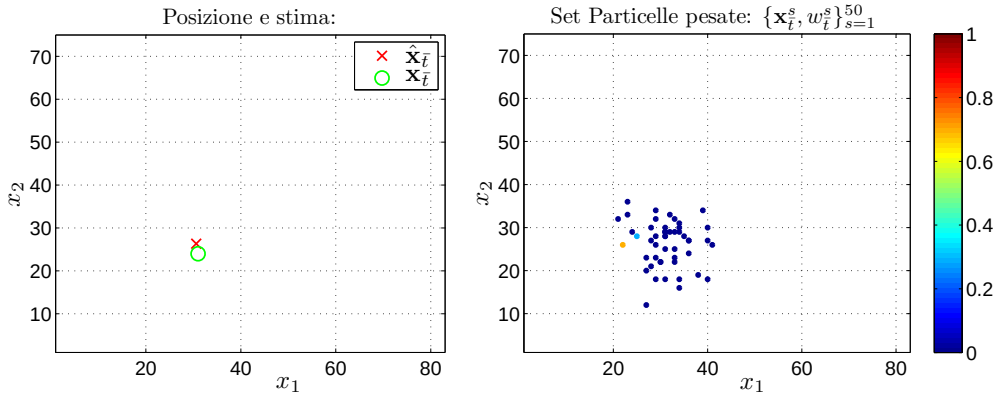


Fig. 39: Posizione e stima all'istante $\bar{t}$, $S = 50$; sia la traiettoria che il filtro usano un processo di guida gaussiana 2D.

(il numero 1 che è in posizione centrale: vedi Fig. 20), due NA spenti, tre NA spenti. Si è avuta cura di disattivare dei *beacon* vicini alla traiettoria in modo che le prove siano significative. L'aumento di errore tra la configurazione completa e quella con $1-2$ nodi spenti è minore di 1 m e addirittura di poco superiore è l'errore con ben tre nodi spenti. L'approccio implementato si dimostra quindi adatto a funzionare in contesti di emergenza in cui è molto probabile che l'hardware di appoggio abbia subito danni, che è esattamente ciò che viene richiesto dal progetto.

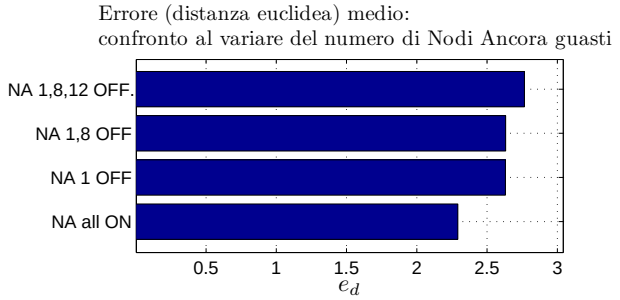


Fig. 42: Confronto errore (distanza euclidea in passi, mediata su tre istanze del filtro) di stima in seguito al guasto di un numero variabile di NA S; traiettoria e FP con $f_{\mathbf{v}}$ come in Fig. 35c ($a = 1.28$, $b = 3.6$, $\rho_{MAX} = 14$)

F. Prove sperimentali off-line

Prima di testare l'algoritmo on-line, sono state fatte delle prove off-line basate su dati reali: attraverso la stessa procedura utilizzata per la raccolta dei dati per la costruzione della mappa a priori, è stato creato un file di testo in formato `.txt` con una riga per ogni pacchetto ricevuto proveniente dai NA. La temporizzazione è stata affidata al timer del NM come descritto in V-E.4; all'interno del tempo d'ascolto veniva inviato in seriale e memorizzato da un'applicazione java un pacchetto per ciascun NA ricevuto, mentre la separazione tra un istante di campionamento e l'altro veniva segnalata da un pacchetto delimitatore che qui chiameremo *nullPck*. E' stato seguito un lungo percorso prestabilito con il NM e, al termine della rilevazione, l'elaborazione off-line del file di log è stata affidata allo script MATLAB™ `stimaPercorso`. Alla funzione `elaborazLogPercorso` viene passato come parametro il file di log, in essa viene letto dal file solo l'identificativo (ID) del nodo trasmittente e il valore di RSSI associato a tale pacchetto; in uscita è prodotta la matrice `y` che sulla colonna t-esima contiene il vettore delle misure $\mathbf{y}_t$ (vettore colonna in cui la posizione i-esima contiene l'RSSI ricevuto dal NA i come definito in (5)). La funzione `elaborazLogPercorso`, rispettivamente:

1) per ciascuna riga (pacchetto) converte il valore di RSSI ricevuto, contenuto in una stringa, in formato esadecimale eliminando eventuali valori non ammissibili (i.e. fuori dal range definito in V-A.1), particolare attenzione va usata per la gestione del formato stringa che può avere lunghezza diversa da pacchetto a pacchetto;
2) scandisce ciascuna riga del log, converte da stringa a numero decimale gli ID di cui controlla l'ammissibilità: vengono scartati ID inferiori a 1 o superiori a 12 (pacchetti *fake*);
3) ad ogni pacchetto nullPck incrementa l'indice t delle colonne della matrice `y` e, per i pacchetti inseriti tra l'istante t e il *nullPck* successivo, memorizza nella colonna t di `y` l'RSSI (calcolati al punto 1) dei NA ricevuti (non è detto che in una particolare posizione siano ricevuti tutti i NA);
4) calcola la percentuale di pacchetti validi sul totale dei ricevuti ad esclusione dei *nullPck*.

Considerando la colonna t esima della matrice MATLAB™ `y`, ai NA il cui segnale non è ricevuto durante il periodo di campionamento, viene assegnato un RSSI (a puro scopo identificativo) pari a -100 dBm.

La funzione `particleFilterSperim` istanziata dallo script `stimaPercorso` è del tutto analoga a `partcileFilter` riportata a pag. 27, con parametri: matrice `y` delle misure, tipo di $f_{\mathbf{v}}$ per la predizione e relativi parametri. Diversamente dalla simulazione, il filtro viene inizializzato con la conoscenza a priori sullo stato: visto che l'ambiente considerato ha quattro possibili entrate si "posizionano" le particelle distribuendole

nel modo seguente (si veda Fig. 10):

$$\mathbf{x}_{t=1}^{s} = \begin{cases} \begin{bmatrix} 82 & 28 \end{bmatrix}' & s = 1, \ldots, \frac{S}{4} \quad \text{(entrata EST)} \\ \begin{bmatrix} 82 & 71 \end{bmatrix}' & s = \frac{S}{4}+1, \ldots, \frac{2}{4}S \quad \text{(entrata N-E)} \\ \begin{bmatrix} 17 & 71 \end{bmatrix}' & s = \frac{2}{4}S+1, \ldots, \frac{2}{4}S \quad \text{(entrata N-O)} \\ \begin{bmatrix} 1 & 6 \end{bmatrix}' & s = \frac{3}{4}S+1, \ldots, S \quad \text{(entrata S-O)} \end{cases}$$

Si vuole anticipare che mediante tale inizializzazione, con l'arrivo delle misure il filtro *identifica al primo passo* la posizione con un piccolo errore (minore di 2 m); tale configurazione del filtro non è per nulla restrittiva in quanto, se l'operatore accendesse il dispositivo quando è già all'interno dell'ambiente, l'algoritmo identifica la posizione corretta dopo $2-3$ passi e raggiunge una precisione di circa un metro al terzo-quarto passo[28].

I percorsi utilizzati per provare l'algoritmo sono stati tracciati con l'intento di coprire tutta dell'area in questione comprese le stanze isolate, con una andatura sostenuta dell'operatore. In particolare il percorso in Fig. 44a ha una durata di 2'14", mentre il percorso in Fig. 44b ha durata 2'20"; l'intervallo di ascolto dei pacchetti è di 1 s mentre il tempo di campionamento per il FP è di 2 s. La densità di probabilità del processo di guida utilizzato nella fase di predizione del FP è un anello uniforme con $\sigma_v = 6$ e raggio $r = 2$. La sequenza delle posizioni stimate in Fig. 44a e 44b prodotte dalla funzione `particleFilterSperim` mostra che l'algoritmo in entrambi i casi produce una stima iniziale situata nel mezzo della mappa[29]: essa è dovuta all'inizializzazione delle particelle. Nel percorso 44a la stima all'arrivo della prima misura è posizionata già ad una distanza di ca. 5 m dalla posizione vera, tuttavia nei successivi passi si nota che la stima non riesce ad inseguire il tratto di cammino che si snoda verso Nord; le $\hat{\mathbf{x}}$ seguenti sono invece soddisfacenti con errore contenuto nei 5 m di specifica ad esclusione delle ultime due stime. In riferimento al percorso 44b si nota invece un errore di poco superiore alle specifiche nella stima iniziale; il tracking viene comunque recuperato entro 3-4 passi.

G. Filtro particellare con perdita di pacchetto

Durante l'elaborazione dei dati raccolti per la costruzione delle mappe a priori di RSSI e varianza (VI-C) di frequente è capitato che a NA distanti dalla posizione di rilevazione fosse associata una varianza prossima allo zero: questo, sulla base del modello (5), implicherebbe il possesso di un dato piuttosto sicuro relativo all'RSSI medio di quel collegamento. Un'attenta analisi dei log mostra tuttavia che durante le rilevazioni viene ricevuto un numero di pacchetti molto limitato (anche inferiore a 30, rispetto ai 100 totali) con RSSI minimo, e conseguentemente il calcolo della varianza campionaria restituisce un valore quasi nullo. In realtà questa informazione è fuorviante poiché non tiene conto di tutti i pacchetti arrivati con potenza inferiore alla soglia di sensibilità e per questo non registrati. Inoltre i controlli implementati nello *script* `elaborazLog` rivelano numerosi errori legati alla registrazione dell'RSSI salvato dal *mote*: il valore non appartiene al *range* ammissibile, probabilmente a causa di un problema al *chip* radio[30]. Misure di questo tipo non possono essere considerate valide per la stima della posizione.

Per risolvere i problemi descritti è allora stata aggiunta una modellizzazione del processo di arrivo dei pacchetti, rappresentato con la seguente variabile aleatoria:

$$\gamma_l(\mathbf{x}_t) = \begin{cases} 1 & \text{se il pacchetto inviato dal NA } l\text{-esimo} \\ & \text{è ricevuto in } \mathbf{x}_t \\ 0 & \text{altrimenti} \end{cases}$$

Data la particolare applicazione, non è necessario modellizzare il ritardo nell'arrivo dei pacchetti: è sufficiente conoscere la probabilità che un pacchetto giunga a destinazione. Si fa la tipica assunzione che il processo di arrivo sia stazionario e i.i.d., e valga

$$\mathrm{P}[\gamma_l(\mathbf{x}_t) = 1] = \lambda_l(\mathbf{x}_t), \quad \lambda_l(\mathbf{x}_t) \in [0,1], \ \forall\, t, l. \tag{21}$$

Ciò equivale a modellizzare il processo di arrivo come un processo bernoulliano con probabilità di successo $\lambda_l(\mathbf{x}_t)$. Le misure sono quindi descritte da

$$\tilde{y}_{l,t} = \begin{cases} y_{l,t} & \text{se } \gamma_l(\mathbf{x}_t) = 1 \\ -70 & \text{se } \gamma_l(\mathbf{x}_t) = 0, \end{cases} \tag{22}$$

dove la scelta del valore -70 dBm è motivato in VI-C. Con questa assunzione, il funzionamento del filtro particellare rimane invariato, ad esclusione della fase di pesatura che viene modificata nel modo seguente. Supponiamo per il momento di considerare un solo *mote*: la probabilità a posteriori diventa:

$$p(y_{l,t} | \, \mathbf{x}_t) = \\ = \begin{cases} \lambda_l(\mathbf{x}_t) \frac{1}{(2\pi)^{1/2}\sigma_l(\mathbf{x}_t)} \, \mathrm{e}^{-\frac{(y_{l,t}-\bar{y}_{l,t})^2}{2\sigma_l^2(\mathbf{x}_t)}} & \text{se } \gamma_l(\mathbf{x}_t) = 1 \\ 1 - \lambda_l(\mathbf{x}_t) & \text{se } \gamma_l(\mathbf{x}_t) = 0 \end{cases}$$

Quindi, tenendo conto dell'indipendenza della trasmissione di ogni singolo nodo àncora, la probabilità dell'intero vettore delle misure condizionata allo stato si ottiene mediante produttoria:

$$p(\mathbf{y}_t \, | \, \mathbf{x}_t) \propto \\ \propto \prod_{l \in N} \big(1 - \lambda_l(\mathbf{x}_t)\big) \cdot \prod_{l \in R} \lambda_l(\mathbf{x}_t) \frac{1}{(2\pi)^{1/2}\sigma_l(\mathbf{x}_t)} \, \mathrm{e}^{-\frac{(y_{l,t}-\bar{y}_{l,t})^2}{2\sigma_l^2(\mathbf{x}_t)}},$$

dove N è l'insieme dei NA non rilevati nella misura corrente e R è l'insieme complementare (NA ricevuti).

I valori $\lambda_l(\mathbf{x}_t)$ sono calcolati dalle rilevazioni a priori come:

$$\lambda_l(\mathbf{x}_t) = \max\left[0.03, \min\left(\frac{\text{numero pck ricevuti}}{100}, 0.97\right)\right],$$

dove 100 è il numero totale di ascolti durante una rilevazione (si veda VI-B). La formula mostra che $\lambda_l(\mathbf{x}_t) \in [0.03, 0.97]$, a differenza di quanto indicato in (21): il fatto è che la durata di 100 s scelta per le rilevazioni a priori appare un intervallo

[28] Considerando che l'operatore stia fermo per i primi istanti, altrimenti è comunque assicurato il tracking con ovviamente precisioni inferiori.

[29] La stima iniziale si distingue dalle altre perché è posizionata nello stesso punto al centro della mappa, per entrambe le prove.

[30] Si è portati ad attribuire la causa di questo fenomeno al *chip* radio piuttosto che a problemi di trasmissione seriale osservando che i valori errati non sono disposti in modo casuale ma ricorre il valore 0x41 (65 dBm).

di tempo troppo limitato per la valutazione di un parametro di un processo aleatorio. Pertanto appare ragionevole restringere i valori ammissibili per $\lambda_l(\mathbf{x}_t)$ fissando una probabilità minima di arrivo del pacchetto pari al 3% ed una eguale probabilità minima di perdita.

Le mappe a priori da costruire, ora, diventano 3: RSSI medio, varianza e probabilità di ricezione di un pacchetto. Si osservi che, per i *mote* non rilevati in una determinata posizione, si ha cura di porre RSSI medio $\overline{y}_{l,t}$ a -70 dBm, come mostrato in (22), e la varianza $\sigma^2_{l,t}$ a 25 dBm^2. Questo valore è stato scelto in modo tale che, modellizzando la potenza del segnale ricevuto come una gaussiana $\mathcal{N}(\overline{y}_{l,t}, \sigma^2_{l,t})$, la probabilità di ricevere un pacchetto[31] sia ca. pari al 4%, vicina alla probabilità minima di arrivo fissata. Le mappe a priori di RSSI e varianza sono le stesse costruite in VI-C; per costruire la mappa di $\lambda_l(\mathbf{x}_t)$ si procede nello stesso modo, ponendo $d_0 = 10$ e successivamente restringendo i valori interpolati al *range* ammissibile stabilito per l'applicazione. In Fig. 43 è riportata la mappa di probabilità di ricezione dal NA 9.

H. Prove sperimentali off-line con FP con perdita di pacchetto

I percorsi utilizzati per provare l'algoritmo off-line con l'implementazione del PF con modello delle misure che contempla la perdita di pacchetto sono gli stessi utilizzati in VI-F. La densità di probabilità del processo di guida utilizzato nella fase di predizione del FP è ancora un anello uniforme però con parametri modificati rispetto al caso che non contempla la perdita di pacchetto: $\sigma_v = 5$, raggio $r = 0.5$. La sequenza delle posizioni stimate in Fig. 44c e 44d dalle funzione MATLAB™, analoghe al caso senza perdita di pacchetto ma con le dovute modifiche di cui in VI-G, mostra che la modellizzazione della perdita di pacchetto consente delle migliorie alle stime soprattutto nei casi in cui si aveva errore maggiore rispetto alla media. Tali stati sono stati evidenziati in colore rosso nelle Figg. 44c e 44d. Analoghe prestazioni si ottengono con $f_{\mathbf{v}}$ di tipo gaussiana 2D; prestazioni leggermente più scadenti si registrano con processo di guida di tipo "Beta" mentre l'utilizzo della memoria della direzione è da evitare viste le scadenti *performance*.

VII. TEST-DRIVE DELL'ALGORITMO

A. Implementazione WSN nell'ambiente di test

Questa sezione è dedicata alla descrizione della realizzazione pratica del *testbed* dell'algoritmo di localizzazione implementato presso un sotterraneo dei Laboratori dell'INFN di Legnaro.

Una volta decisa la disposizione dei nodi àncora riportata in IV-C, si è dovuto studiare un modo per posizionare i *mote* nel particolare ambiente dei sotterranei dell'INFN. La presenza di numerose canalette (elettriche ed idrauliche) ha permesso di fissare i trasmettitori ad una altezza di 2 m circa dal suolo, ben accettata per il corretto funzionamento dell'algoritmo; un posizionamento di questo tipo potrà facilitare il cablaggio dei NA qualora il sistema venisse effettivamente realizzato e messo in funzione. É però utile ricordare che la rete di localizzazione deve essere in grado di funzionare soprattutto in casi di emergenza, come un incendio, dove di norma la tensione di rete viene staccata: a questo proposito si consiglia di scegliere come NA dei sensori equipaggiati di una batteria tampone in grado di alimentarli e quindi di garantire il corretto funzionamento dell'intero sistema per un sufficiente arco di tempo. I NA, durante la fase sperimentale di questo progetto, sono stati fissati utilizzando una struttura particolarmente semplice ma in grado di non ostruire la propagazione del segnale elettromagnetico emesso, riportata in Fig. 45. Si osservi che i *mote* vi sono riposti con le batterie rivolte verso l'alto: ciò permette di diminuire, rispetto alla posizione opposta, l'ostruzione al segnale radio emesso dal trasmettitore stesso [25].

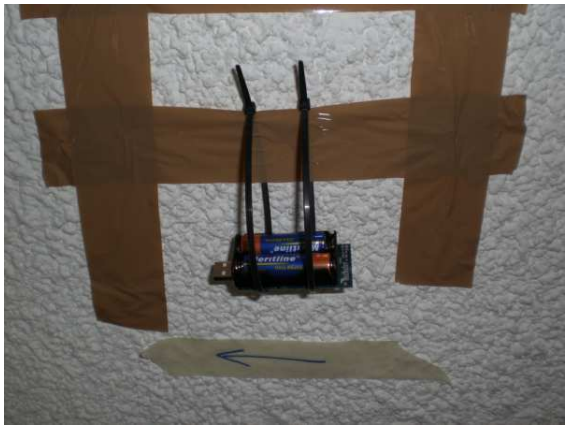


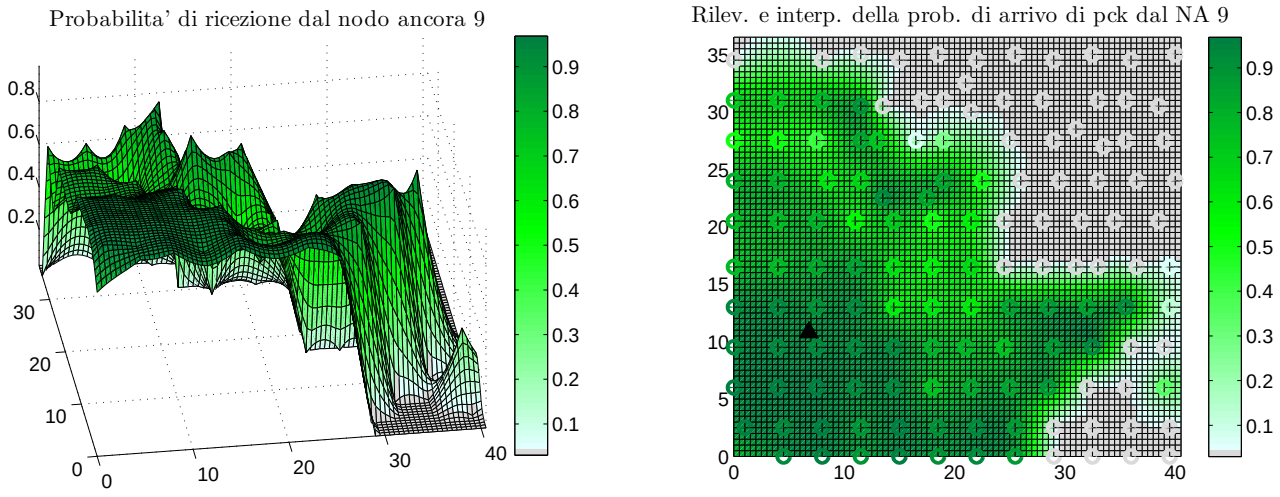
Fig. 45: Struttura utilizzata per fissare i nodi àncora nell'ambiente.

Per ogni *mote*, prima di effettuare qualsiasi operazione nella WSN, è stato verificato con un voltmetro che la tensione fornita dalla coppia di batterie fosse sufficiente (3 V $\pm$ 0.1 V) per garantire la potenza del segnale emesso fosse quella scelta in fase di studio. Si noti inoltre che l'algoritmo scelto non richiede la stima del valore di $RSSOFFSET$ di (3): ogni calcolo si basa sulla mappa a priori di RSSI medio raccolta. Una calibrazione si renderebbe tuttavia necessaria se come NM venisse usato un *mote* diverso da quello usato per effettuare le rilevazioni, dato che bisognerebbe trovare l'*offset* che lega l'RSSI dei due sensori; tale operazione non è stata effettuata, tuttavia, dato che ogni prova ha visto la presenza di un solo NM nella WSN.

B. Funzionamento in real-time del BOZ algorithm

Al termine del lavoro svolto è stato possibile effettuare un test realistico del funzionamento dell'intero sistema. Un mote è stato collegato ad un *laptop* per fungere da NM; per simulare l'invio delle stime all'esterno un *beacon* è stato designato a raccogliere i dati di posizione del nodo mobile ed è stato a sua volta collegato ad un secondo pc per poter visualizzare lo stato stimato dell'operatore. Un componente del gruppo, per simulare il comportamento dell'operatore, si è fisicamente

[31] Che corrisponde alla probabilità di ricevere un segnale con potenza superiore a -60 dBm.



(a) Probabilità di ricezione dal NA 9

(b) Probabilità interpolata di ricezione dal NA 9

Fig. 43: Esempio: mappa di λ_9.

spostato per tutta l'area di test portandosi appresso il NM: sul monitor egli aveva la possibilità di valutare la posizione stimata restituita dall'algoritmo (attraverso una figura con un marker mobile sullo sfondo della pianta dell'ambiente come in Fig. 44) e di confrontarla con la posizione reale in cui si trovava. Con un intervallo di campionamento pari a 2 s il ritardo nella visualizzazione dovuto all'elaborazione non è stato praticamente avvertibile, mentre non era così con un periodo di osservazione pari a 4 s, necessario però a che la stima arrivasse alla centrale operativa. Una verifica di questo tipo non consente di valutare in modo assoluto la precisione passo dopo passo della stima, ma l'errore sulla stima è facilmente valutabile facendo un confronto con i numerosi punti di riferimento disseminati per tutto l'ambiente. L'operatore ha avuto cura di muoversi variando più volte la velocità per essere il più simile possibile simile al comportamento di un agente di soccorso: in questo modo è stato possibile valutare che i parametri scelti[32] per la modellizzazione del processo di guida del filtro particellare hanno dato buoni risultati dissipando i dubbi su un possibile legame troppo stretto tra essi ed il reale movimento dell'operatore. A riprova dei più che soddisfacenti risultati ottenuti sono stati girati dei video reperibili sul sito web `www.dei.unipd.it\~ortolang`; è stato valutato sperimentalmente che l'errore medio sulla stima si aggira intorno ai 2.5 m. Nelle rare occasioni in cui la stima appariva divergere, sono stati sufficienti pochi passi affinché l'errore ritornasse nel suddetto intervallo.

C. Implementazione WSN in un diverso ambiente

Per verificare la robustezza dell'algoritmo e la sua indipendenza dall'ambiente di implementazione, si è scelto di testare l'algoritmo di localizzazione anche in diverso ambiente: si è optato per il piano terra del collegio "Sacro Cuore" situato a Padova in via Belzoni, la cui mappa viene visualizzata in Fig. 46 a) (si noti come l'origine degli assi di riferimento venga fissato in basso a sinistra). L'ambiente è completamente diverso da quello dei sotterranei dell'INFN: ampio, luminoso e praticamente privo di ostacoli.

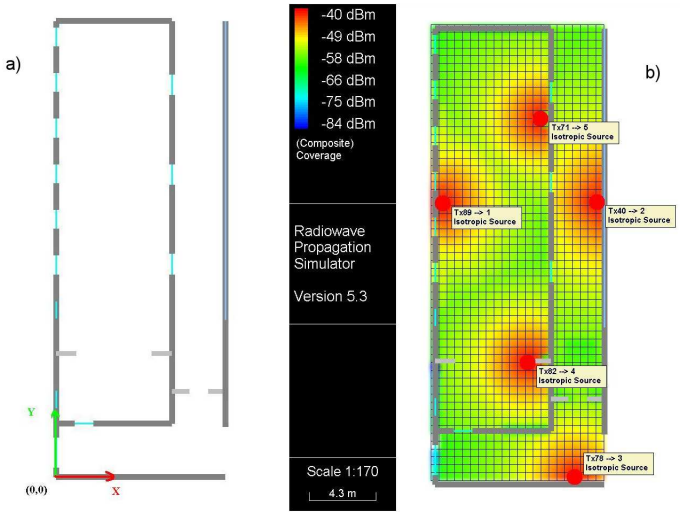


Fig. 46: a) Mappa planimetrica del piano terra del collegio "Sacro Cuore"; **b)** Simulazione della copertura di rete.

Le dimensioni dell'ambiente in esame sono 10.8 m $\times$ 29 m, con area totale di circa 320 mq; l'altezza media è di 3.30 m.

Con ragionamenti analoghi a quelli riportati in IV-C, si è deciso di posizionare un numero di nodi àncora pari a 5 nelle posizioni riportate in Tab. 3; con gli stessi parametri di rete già considerati per le prove nell'altro ambiente (0 dBm di potenza di trasmissione ad una frequenza di 2.4 GHz), la simulazione sulla potenza dei segnali è mostrata in Fig. 46 b).

Anche in questo contesto i nodi trasmettitori vengono posti

[32]In queste prove nello specifico è stato usato un processo di guida di tipo "Ring" con deviazione $\sigma_v = 5$ e raggio $\rho = 0.5$ nel caso di tempo di campionamento di 2 s; $\sigma_v = 6$, $\rho = 0.7$ con tempo di campionamento di 4 s.

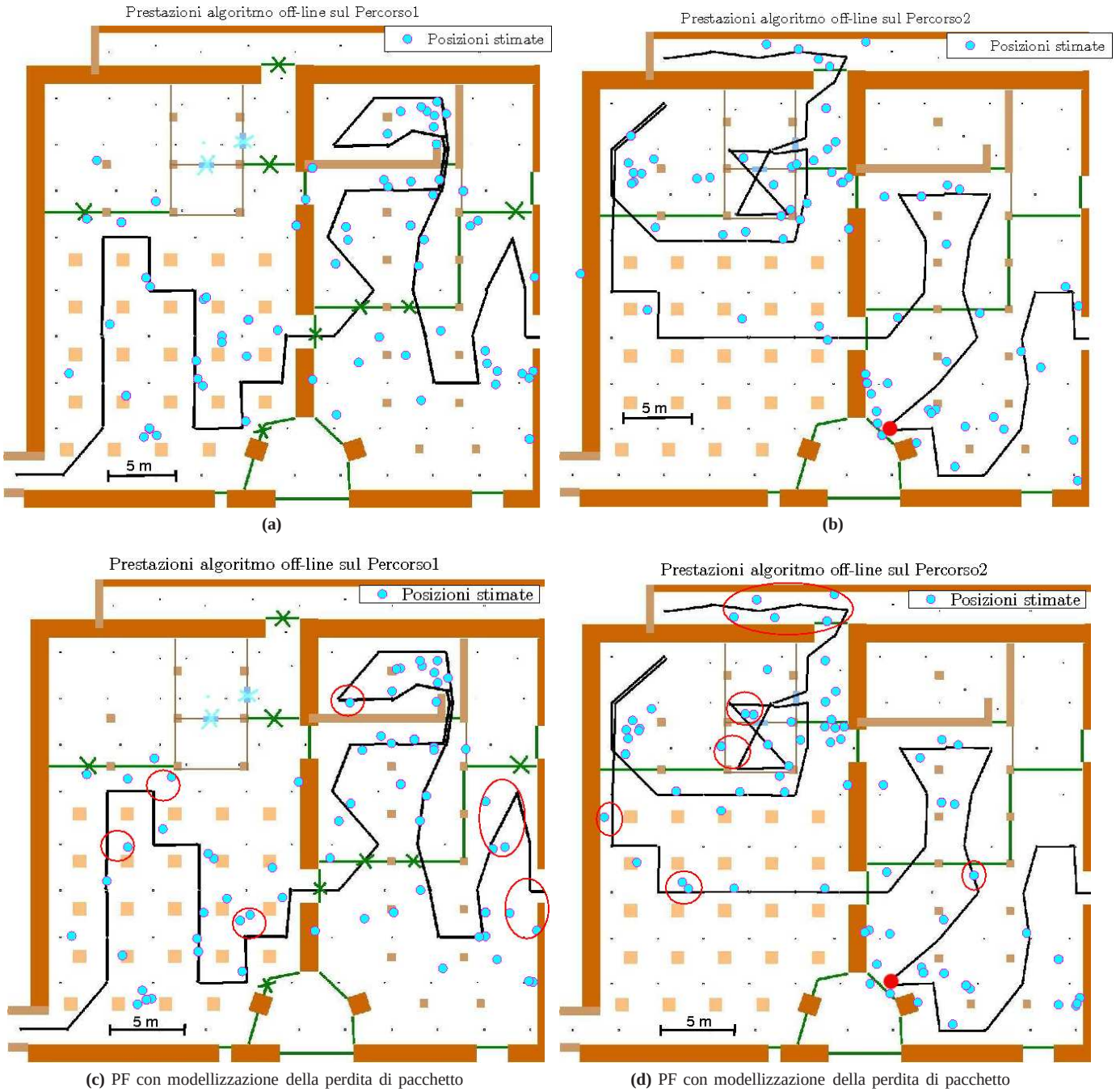


Fig. 44: Posizioni stimate off-line con dati raccolti sperimentalmente. I percorsi veri seguiti dal NM con l'operatore sono mostrati graficamente dalla spezzata di colore nero.
(a),(c): inizio percorso dall'entrata Est;
(b),(d): inizio percorso dall'entrata N-O;
il marker pieno di colore rosso indica una pausa di qualche secondo del cammino dell'operatore, le ellissi rosse indicano i punti dove si sono riscontrate prestazioni migliori tra la modellizzazione senza e quella con perdita di pacchetto.

nodo àncora	coord. x[m]	coord. y[m]
Tx89 → 1	0.5	17.9
Tx40 → 2	10.3	18
Tx78 → 3	8.9	0.5
Tx82 → 4	5.9	7.8
Tx71 → 5	6.7	23.3

Tab. 3: Coordinate dei nodi àncora nella disposizione adottata.

ad una altezza da terra pari a 2 m e la griglia di ricevitori, con passo pari a 0.5 m, a 1 m dal suolo. Dalla simulazione di Fig. 46 b) si nota come in ogni punto dell'ambiente la copertura (composita) assicuri una potenza ricevuta maggiore o uguale a -60 dBm.

Ovviamente vengono effettuate nuovamente tutte le rilevazioni atte a costruire la mappa *a priori* necessaria per il funzionamento dell'algoritmo per la localizzazione e l'asservimento

del nodo mobile; in questo caso sono pari a $n = 33$.

Per non appesantire la lettura, non vengono riportati grafici relativi ai risultati ottenuti in questo particolare ambiente; si può comunque affermare che le prestazioni dell'algoritmo di localizzazione non sono risultate rispettose delle specifiche di progetto. Questo si motiva notando che l'assenza di ostacoli nell'ambiente e la potenza di trasmissione troppo elevato costruiscono mappe a priori prive di varietà, da cui è difficile per il filtro particellare ricostruire in maniera univoca la posizione stimata: questo conferma una delle ipotesi poste per il buon funzionamento dell'algoritmo. Per rimediare a ciò è sufficiente abbassare il livello della potenza con cui viene trasmesso il segnale dai nodi àncora, in modo da ottenere una descrizione più varia dell'ambiente.

VIII. Sviluppi futuri e conclusioni

Il progetto svolto pone le basi per numerosi lavori futuri, che possono perfezionare il sistema di localizzazione implementato: ogni modifica non va comunque a sostituire il lavoro di ingegnerizzazione ed ottimizzazione, inevitabile soprattutto in vista di una commercializzazione del sistema.

Tali miglioramenti possono riguardare l'utilizzo di hardware appositamente studiato, dotato di una batteria tampone in grado di alimentare i *mote* per tutta la durata dell'evento di emergenza, oltre a *routine* di autodiagnosi per la verifica del livello della batteria e del buon funzionamento generale del nodo.

Alcune modifiche possono riguardare anche l'algoritmo stesso, pensando ad esempio di impedire che le posizioni stimate all'istante t ed all'istante $t + 1$ prevedano l'attraversamento di un muro da parte dell'operatore, o in generale un percorso materialmente inagibile. In vista dell'estensione alla copertura di un'area molto maggiore, può essere introdotto un meccanismo di *clustering*: data la posizione stimata del nodo mobile, solo un certo sottoinsieme di nodi àncora viene messo in funzione per la localizzazione; questo approccio consente un notevole risparmio energetico e la drastica riduzione del traffico dati sul canale di trasmissione radio. Può inoltre essere implementato un algoritmo di navigazione che provveda automaticamente alla guida dell'operatore verso una destinazione preimpostata da egli stesso oppure dalla centrale operativa esterna.

Innovazioni a parte, il progetto svolto si è rivelato enormemente interessante, e non solo per le soluzioni presentate e per i risultati ottenuti, ma anche per la completezza di aspetti che sono stati esaminati, dalla scelta di un'opportuna struttura di sostegno per i sensori alla taratura di un processo di guida per un filtro particellare. Esso ha dato la possibilità ad un gruppo di tre studenti di attivare la loro inventiva e di sperimentare una completa indipendenza operativa, rendendoli pienamente responsabili delle scelte fatte. Un'esperienza, si ritiene, assolutamente formativa e profonda, e tanto più significativa perchè vicina all'ingresso in un mondo, come quello del lavoro, in cui non è sempre possibile avere un *feedback* in tempo reale sulla bontà delle soluzioni adottate.

Appendix I
Interpolazione dei dati

La tecnica di seguito descritta utilizza uno stimatore lineare per la ricostruzione di un segnale campionato sui punti di un reticolo bidimensionale. Si riassumono preliminarmente alcuni concetti sui segnali casuali e sulla stima lineare.

A. Processi casuali a 2D

Si abbia un segnale bidimensionale (2D), $s(x, y)$ particolare realizzazione di un processo stocastico reale a media nulla, campionato su un reticolo regolare di lato Δ. Generalmente non si dispone direttamente del campione $s_{m,n}$ ma una sua versione disturbata da rumore bianco

$$s'_{m,n} = s(m\Delta, n\Delta) + z_{m,n} \quad \forall\, m, n; \tag{23}$$

dove la sequenza 2D disturbante è costituita da campioni indipendenti $z_{m,n}$. In particolare si ha quindi

$$E[z_{m,n} z_{h,k}] = \begin{cases} \sigma_z^2 & m = h,\ n = k \\ 0 & m \neq h,\ n \neq k \end{cases}$$

Il segnale può essere ottenuto dall'uscita di un filtro formante $h_{m,n}$ al cui ingresso è applicata una sequenza bianca $w_{m,n}$ di varianza σ_w^2

$$s_{m,n} = w_{m,n} \overset{2}{*} h_{m,n} = \sum_{k,l=-N/2}^{N/2} w_{m-k,n-l}\, h_{k,l}$$

dove con il simbolo $\overset{2}{*}$ si denota l'operazione di convoluzione sui segnali 2D[33]. La funzione di correlazione della sequenza $s'_{m,n}$ è definita come

$$r'_{k,l} = E[s'_{m,n} s'_{m+k,n+l}] = \\ = \sigma_w^2 \sum_{i,j=-N/2}^{N/2} h_{k,l} h_{i+k,j+l} + \sigma_z^2 \delta(k, l) \tag{24}$$

B. Stima lineare

Noti i valori campionati $s'_{m,n}$ definiti nella (23) si vuole stimare l'intera sequenza $s(x, y)$ in tutti i punti del piano; si impone inoltre che lo stimatore sia una combinazione lineare degli $(H+1)^2$ valori assunti dalla sequenza $s'_{m,n}$ in un intorno $H \times H$ del punto da stimare:

$$\hat{s}(x, y) = \sum_{n,m=H/2}^{H/2} a_{n,m}(x, y) s'_{n,m} \tag{25}$$

dove nella (25) i coefficienti $a_{n,m}(x, y)$ sono delle funzioni continue di x e y determinabili in base al principio di ortogonalità[34]

$$\mathrm{E}\left[\{s(x, y) - \hat{s}(x, y)\} s'_{n,m}\right] = 0 \quad |n|, |m| \leq H/2. \tag{26}$$

[33]La convoluzione per i segnali a tempo continuo è definita:

$$f(x, y) \overset{2}{*} h(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\xi, \eta) h(x - \xi, y - \eta) d\xi d\eta$$

[34]Secondo cui lo spazio dell'errore di stima deve essere ortogonale allo spazio delle misure al fine di estrarre tutta l'informazione possibile dai campioni.

La condizione di ortogonalità della (26) si traduce nel sistema di equazioni dette *normali* o di *Wiener-Hopf*:

$$\sum_{p,q=-N/2}^{N/2} a_{p,q}(x,y) r'\left((p-n)\Delta,\,(q-m)\Delta\right) = \\ = r\left(x-n\Delta,\,y-m\Delta\right) \quad |n|,|m| \leq H/2 \quad (27)$$

con

$$r'(p,q) = r\left(x=p\Delta,\,y=q\Delta\right) + \sigma_w^2 \delta(p,q),$$

dove ovviamente $r(x,y)$ è la correlazione definita sulla sequenza originale $s(x,y)$ analogamente alla definizione (24).

Generalizzando il problema al caso di reticolo di campionamento irregolare[35] si indica con un solo indice $i = 1 \ldots N$ gli N punti (i.e. misure) in cui è campionata la sequenza $s(x,y)$. Il vettore dei dati viene così definito:

$$\mathbf{s} = \begin{bmatrix} s(x_1,y_1) \\ s(x_2,y_2) \\ \vdots \\ s(x_N,y_N) \end{bmatrix}$$

La stima della superficie in un punto qualsiasi è quindi funzione lineare di tutti gli N punti con coefficienti $\mathbf{a}(x,y)$

$$\hat{s}(x,y) = \mathbf{s}^T \mathbf{a}(x,y). \quad (28)$$

I coefficienti $\mathbf{a}(x,y)$ dipendono dalla posizione del punto da stimare e devono soddisfare le equazioni *normali* definite in (27) che si riassumo in notazione vettoriale in:

$$\mathbf{R}\mathbf{a}(x,y) = \mathbf{r}(x,y) \quad (29)$$

dove gli elementi della matrice $\mathbf{R}$ sono dati da

$$r_{m,n} = r(x_m - x_n,\, y_m - y_n) + \sigma_w^2 \delta(m,n) \quad m,n = 1 \ldots N; \quad (30)$$

$\mathbf{r}$ è il vettore delle correlazioni calcolate tra il punto dove si vuole fare la stima e i vari punti in cui è assegnata

$$\mathbf{r} = \begin{bmatrix} r(x-x_1,y-y_1) \\ r(x-x_2,y-y_2) \\ \vdots \\ r(x-x_N,y-y_N) \end{bmatrix}$$

Si vede che l'interpolazione non onora il dato, tranne che per $\sigma_w^2 = 0$: quindi, l'interpolatore può funzionare anche come filtro passa basso, per smussare il segnale misurato qualora sia affetto da rumore a spettro bianco. Il vantaggio dell'interpolazione realizzata con la stima lineare è principalmente quello di poter usare misure provenienti da un campionamento irregolare. Non tutte le funzioni $r(x,y)$ sono possibili funzioni di autocorrelazione; è necessario che la loro trasformata sia tutta positiva: solo in questo caso è possibile garantire che, aggiungendo luce bianca σ_w^2, non compaiano zeri nello spettro degli autovalori di $\mathbf{R}$ e quindi il sistema delle equazioni normali sia ben condizionato.

[35]Per cui il punto di misura i-esimo è descritto dalla coppia di coordinate (x_i, y_i).

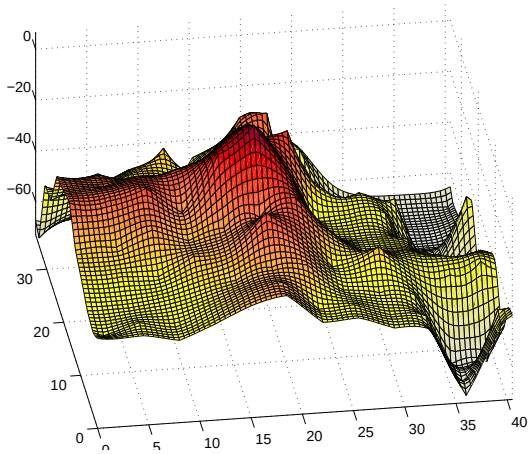
C. Interpolazione mediante stima lineare

Come anticipato, scegliendo $\sigma_w^2 = 0$, la stima lineare coincide con una interpolazione dei campioni che onora il dato: la funzione interpolata è somma di tante funzioni di autocorrelazione centrate sui campioni; i coefficienti sono tali che la loro somma rispetti il dato. Riprendendo la (29) si noti che $\mathbf{R}$ dipende dalla disposizione del reticolo e dalla funzione di correlazione ma non dai dati. Invertendo la relazione (29) e combinandola con la (28) si ottiene il risultato voluto

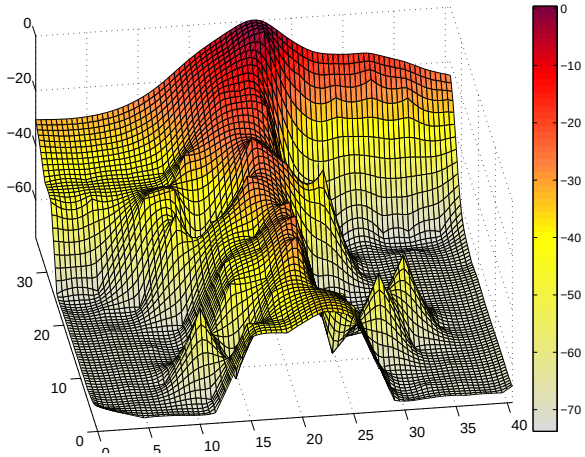
$$\hat{s}(x,y) = \mathbf{s}^T \mathbf{a}(x,y) = \mathbf{s}^T \mathbf{R}^{-1} \mathbf{r}(x,y) = \mathbf{q}^T \mathbf{r}(x,y) = \\ = \begin{bmatrix} q_1 & q_2 & \ldots & q_N \end{bmatrix} \begin{bmatrix} r(x-x_1,y-y_1) \\ r(x-x_2,y-y_2) \\ \vdots \\ r(x-x_N,y-y_N) \end{bmatrix} \quad (31)$$

con $\mathbf{q} = \mathbf{R}^{-1}\mathbf{s}$ vettore che dipende dai dati, dalla matrice $\mathbf{R}$ e dalle posizioni di campionamento.

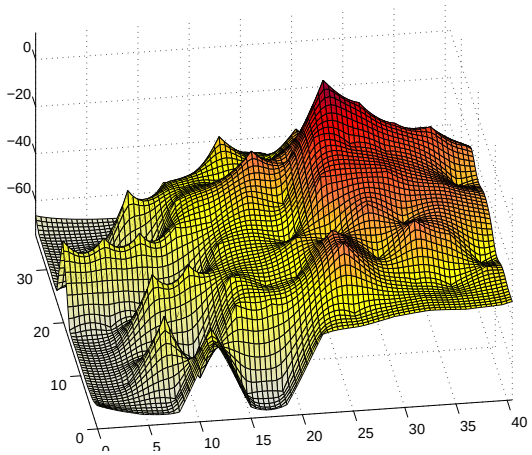
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 1



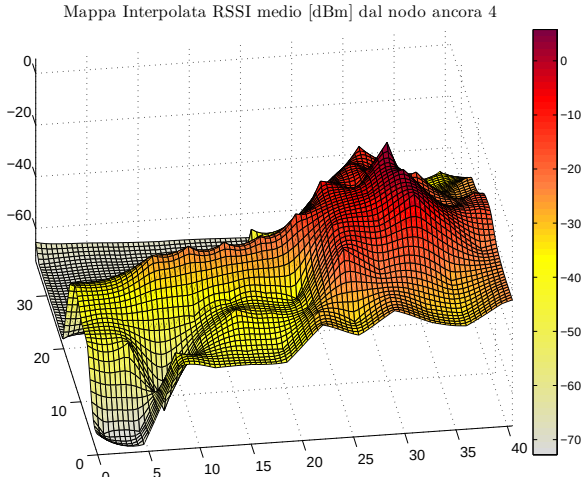
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 2



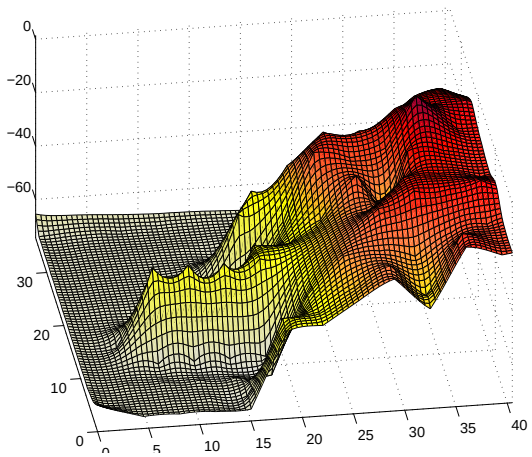
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 3



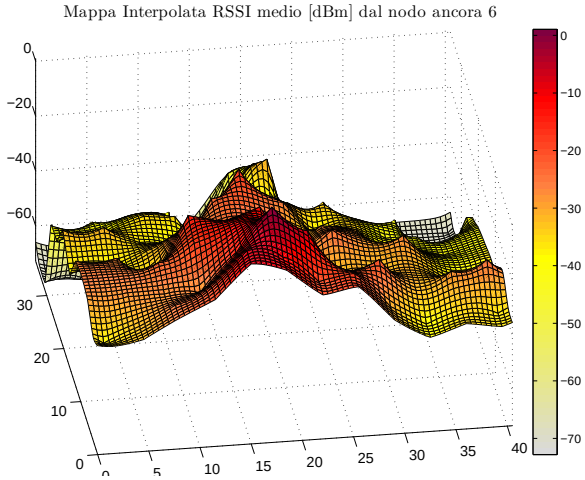
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 4



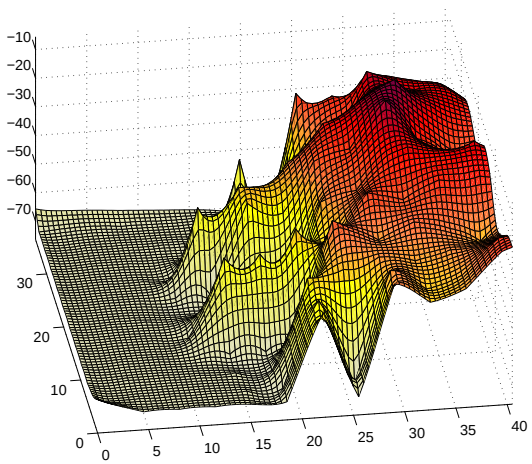
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 5



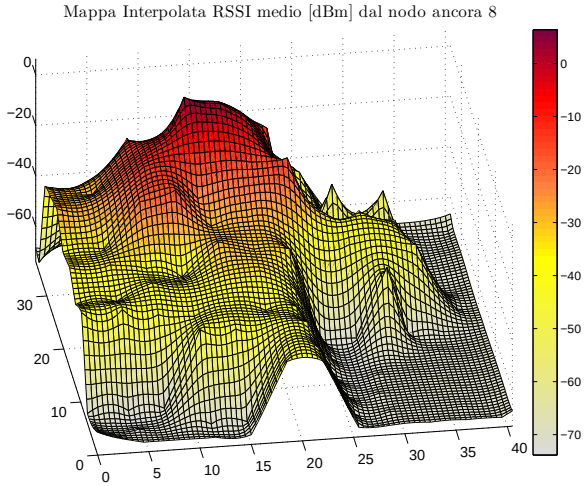
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 6



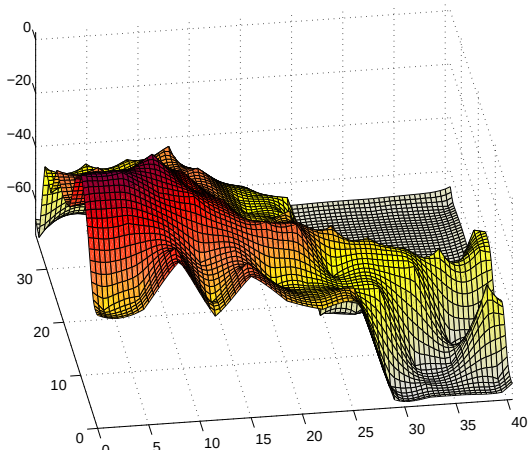
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 7



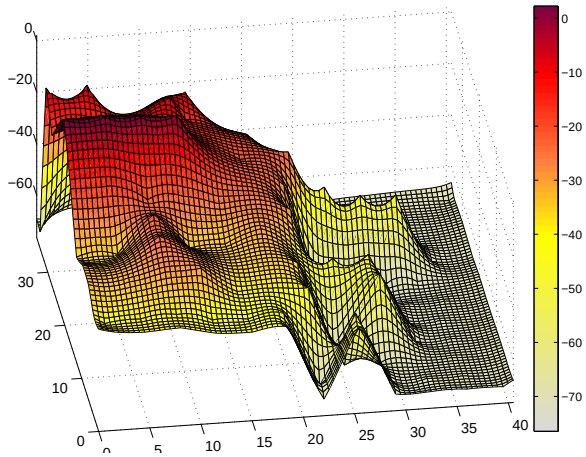
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 8



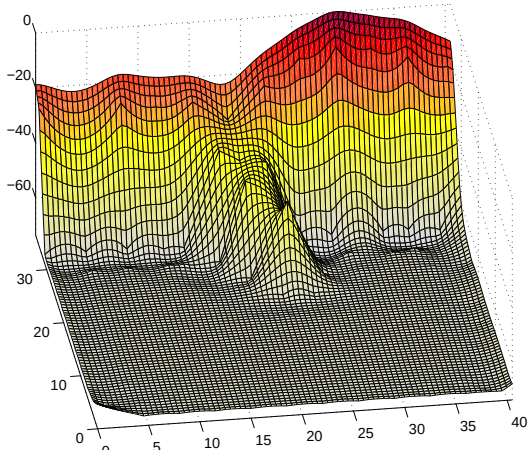
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 9



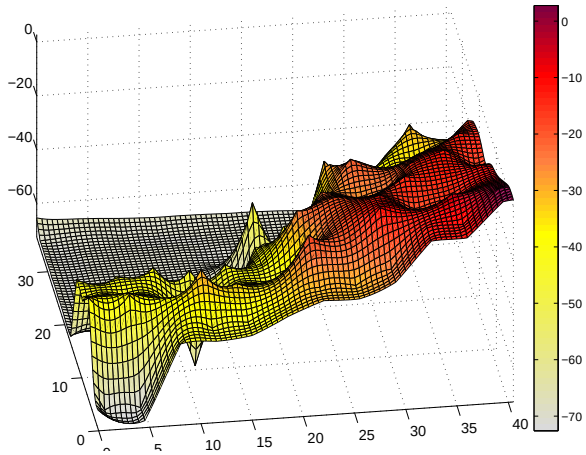
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 10



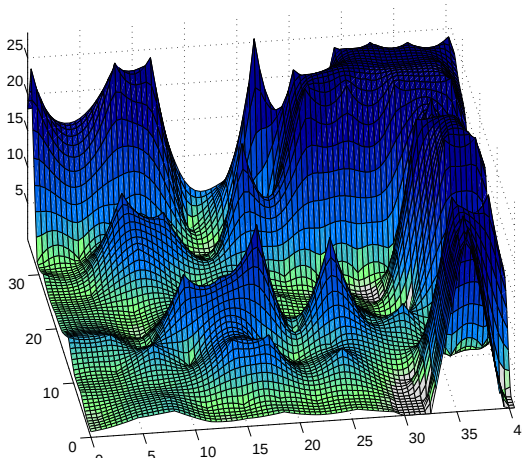
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 11



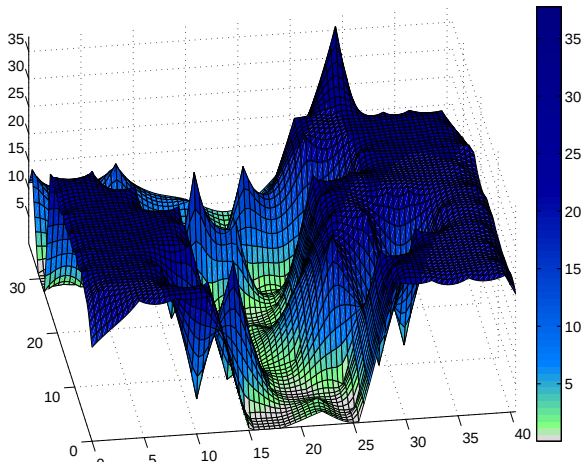
Mappa Interpolata RSSI medio [dBm] dal nodo ancora 12



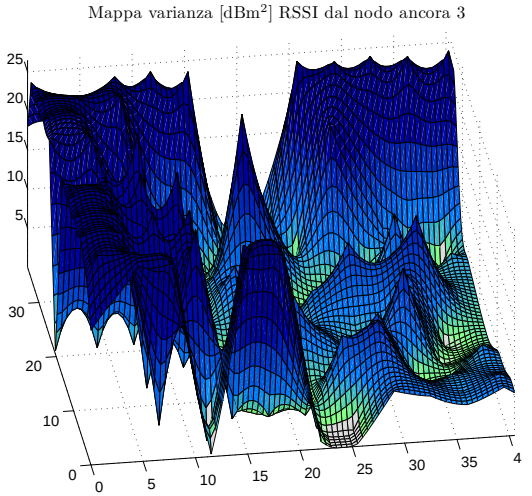
Mappa varianza [dBm^2] RSSI dal nodo ancora 1



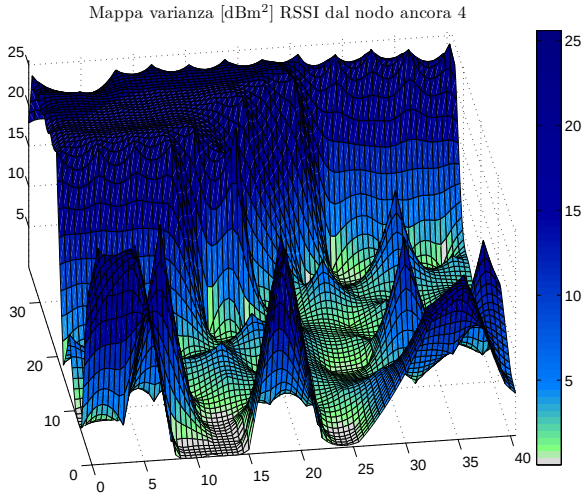
Mappa varianza [dBm^2] RSSI dal nodo ancora 2



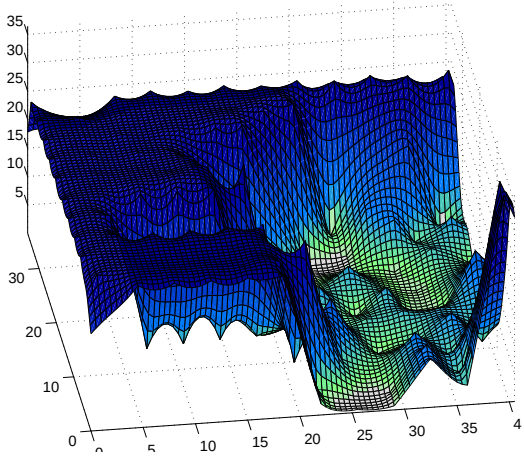
Mappa varianza [dBm^2] RSSI dal nodo ancora 3



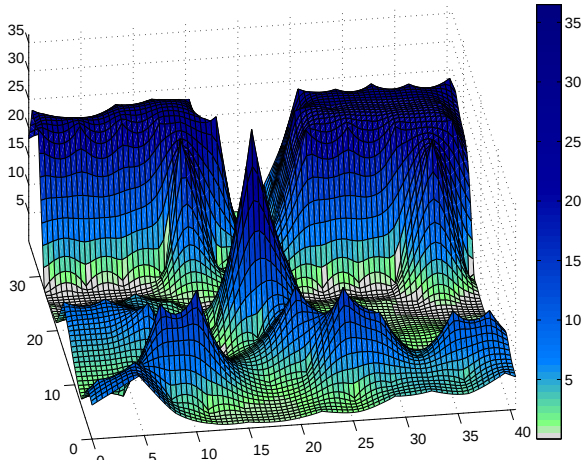
Mappa varianza [dBm^2] RSSI dal nodo ancora 4



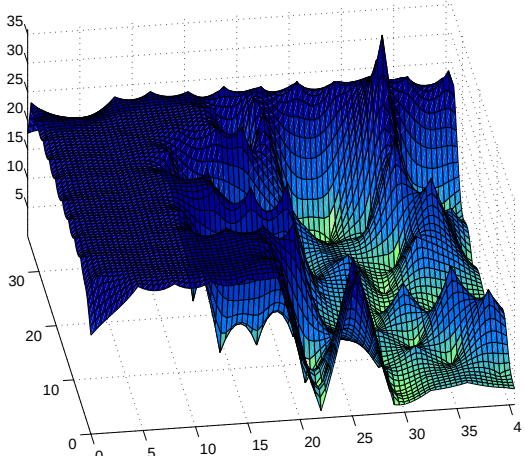
Mappa varianza [dBm^2] RSSI dal nodo ancora 5



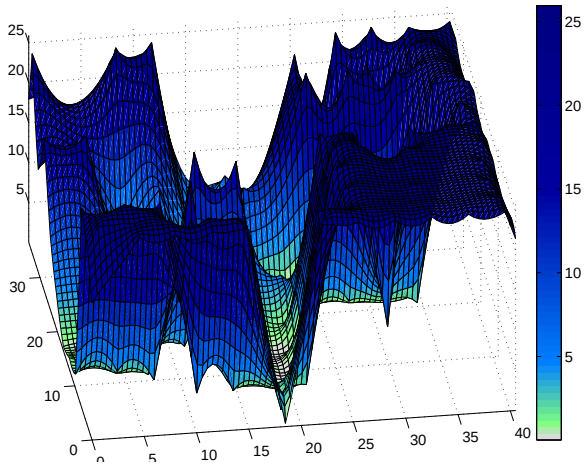
Mappa varianza [dBm^2] RSSI dal nodo ancora 6



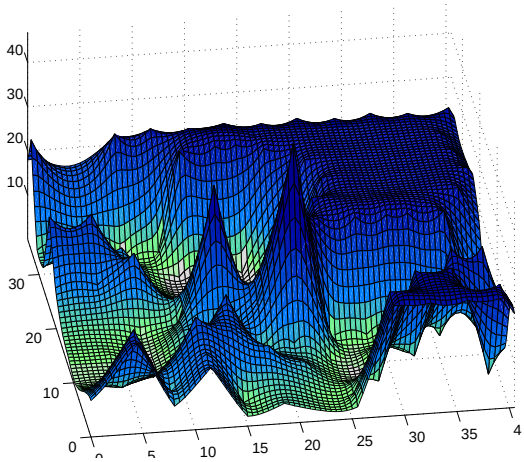
Mappa varianza [dBm^2] RSSI dal nodo ancora 7



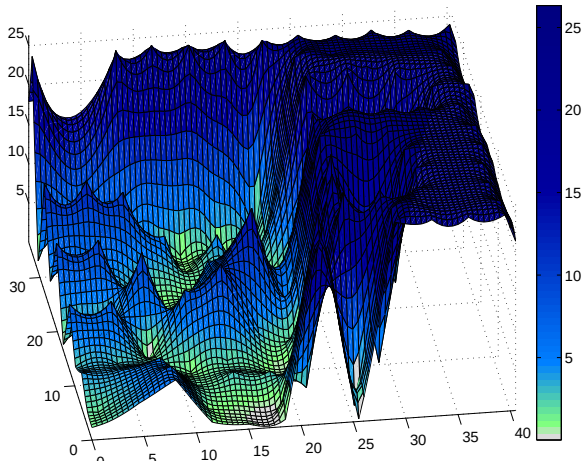
Mappa varianza [dBm^2] RSSI dal nodo ancora 8



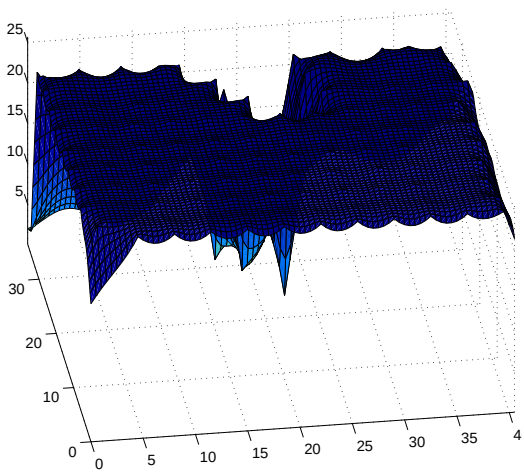
Mappa varianza [dBm^2] RSSI dal nodo ancora 9



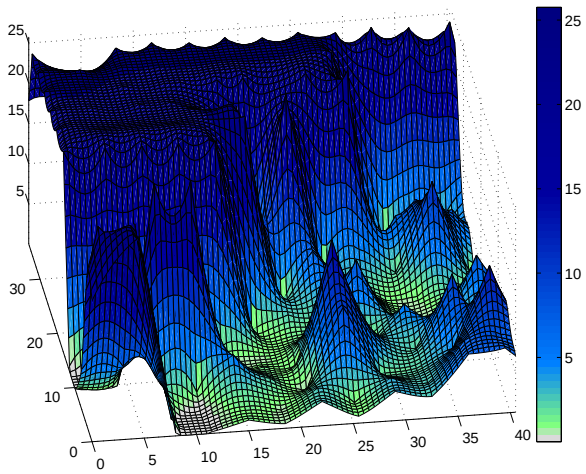
Mappa varianza [dBm^2] RSSI dal nodo ancora 10



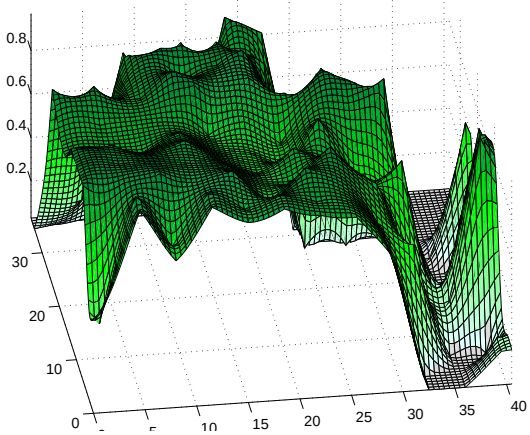
Mappa varianza [dBm^2] RSSI dal nodo ancora 11



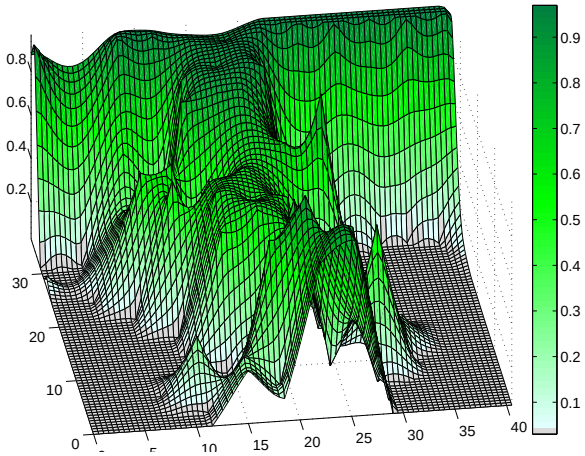
Mappa varianza [dBm^2] RSSI dal nodo ancora 12



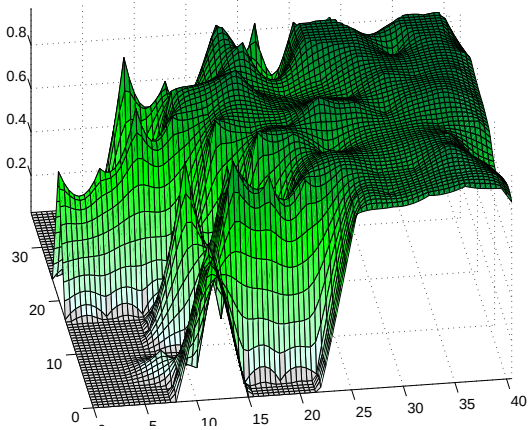
Probabilita' di ricezione dal nodo ancora 1



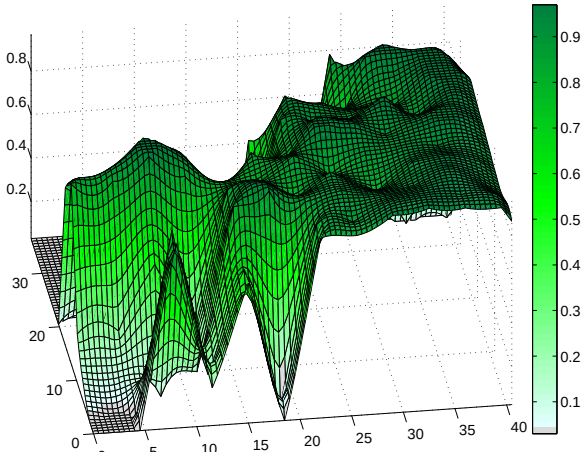
Probabilita' di ricezione dal nodo ancora 2



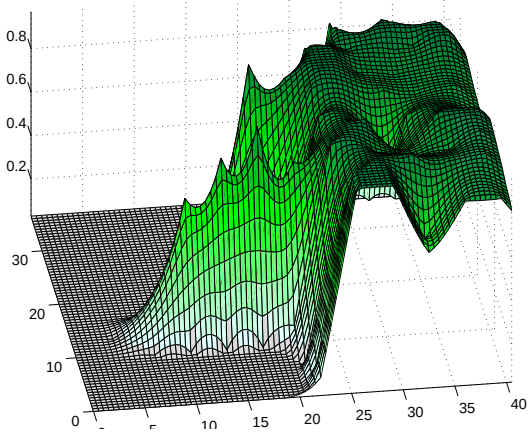
Probabilita' di ricezione dal nodo ancora 3



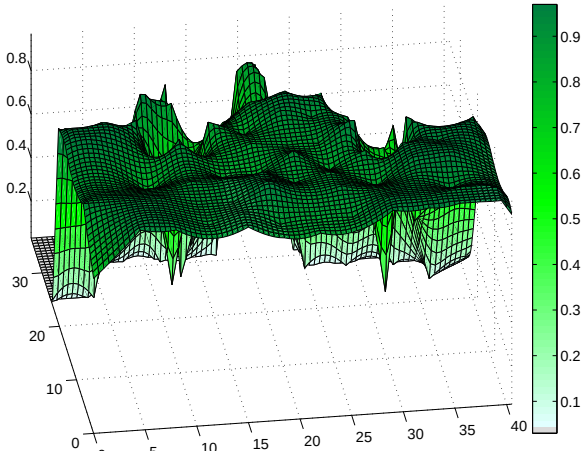
Probabilita' di ricezione dal nodo ancora 4



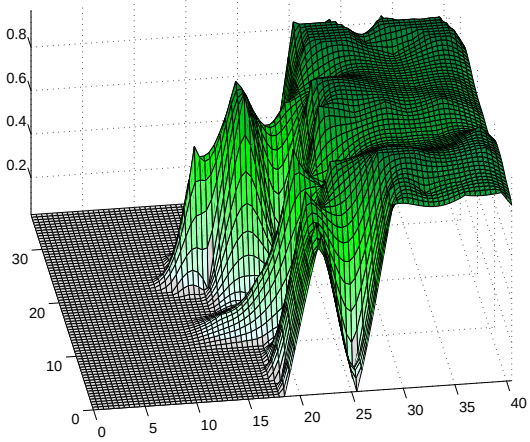
Probabilita' di ricezione dal nodo ancora 5



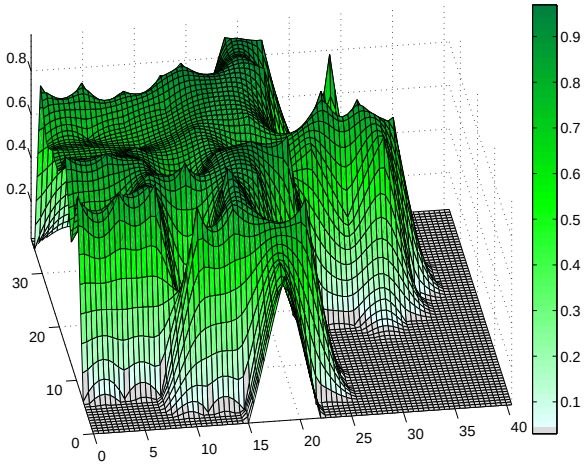
Probabilita' di ricezione dal nodo ancora 6

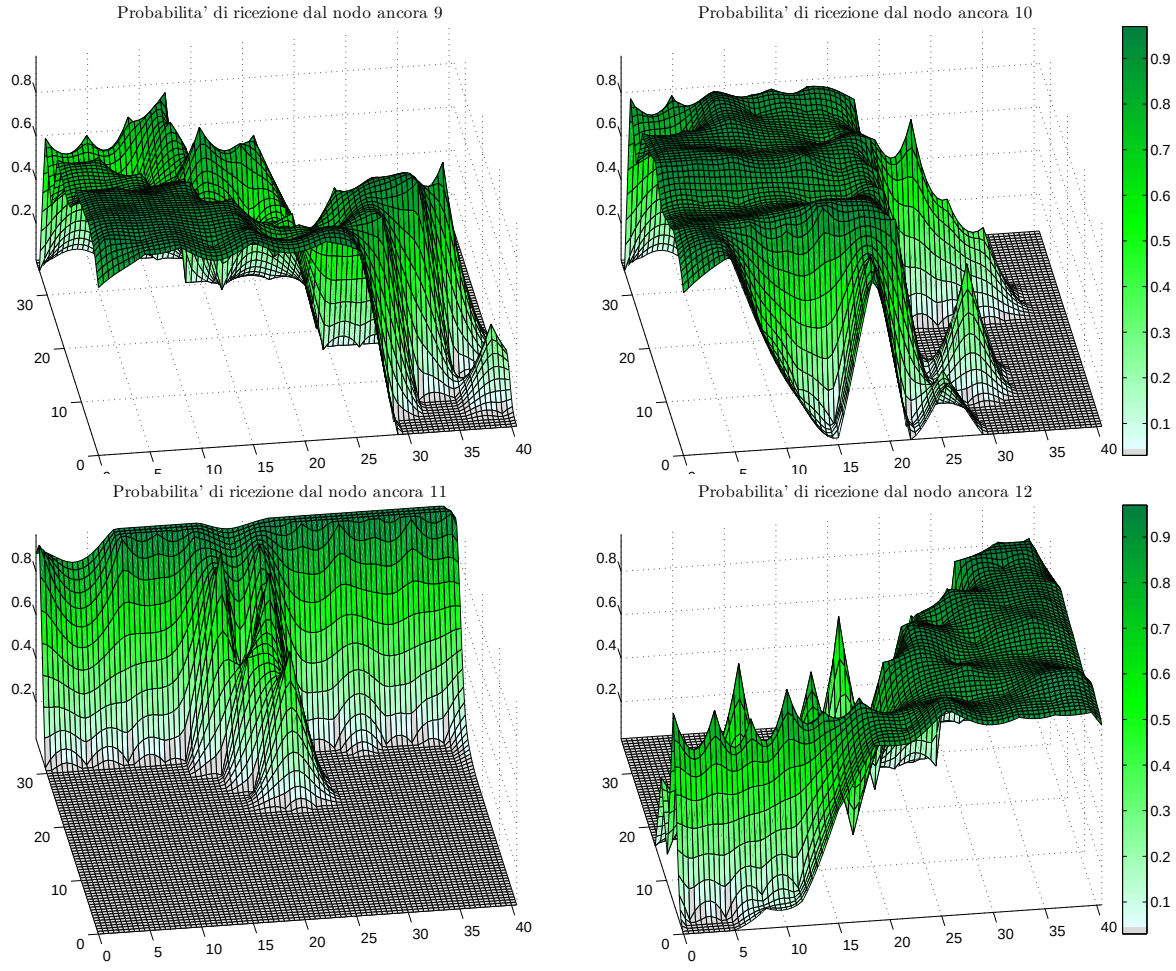


Probabilita' di ricezione dal nodo ancora 7



Probabilita' di ricezione dal nodo ancora 8




Probabilita' di ricezione dal nodo ancora 9
Probabilita' di ricezione dal nodo ancora 10
Probabilita' di ricezione dal nodo ancora 11
Probabilita' di ricezione dal nodo ancora 12

APPENDIX II
FILTRAGGIO BAYESIANO

Nell'approccio bayesiano l'attenzione è rivolta alla costruzione della densità di probabilità (ddp) a posteriori attraverso tutta l'informazione disponibile (comprese tutte le misure ricevute): quest'ultima rappresenta la soluzione completa al problema della stima. La soluzione ricorsiva permette di non dover memorizzare l'intero set di dati ricevuti ma di elaborare le misure appena disponibili; questi filtri constano essenzialmente di due passi. La predizione usa il modello dinamico per predire lo stato di un passo di misura avanti: siccome lo stato è soggetto ad errore di modello, tale operazione deforma e trasla la ddp dello stato. Il secondo passo è l'aggiornamento della ddp alla luce dell'informazione portata dai nuovi dati; richiamando quanto definito nel paragrafo VI-A.2, attraverso la formula di Bayes si ottiene l'aggiornamento della ddp a priori:

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \,|\, \mathbf{x}_t) p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})}{p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})} \tag{32}$$

dove $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1})$ è data dalla formula di Kolmogorov nella (9). La soluzione ricorsiva della (9) e della (32) è la soluzione ottima al problema della stima. Tuttavia tale procedimento è analiticamente improponibile.

Una prima soluzione è il *filtro di Kalman* (KF) dove si assume che la ddp a posteriori sia, per ogni t, gaussiana e quindi parametrizzabile da media e varianza. Questo porta alle note restrizioni sulla modellizzazione del problema quali la distribuzione gaussiana del rumore di modello, la linearità della funzione di aggiornamento dello stato e la linearità del modello di misura. A partire da queste premesse si derivano le formule ricorsive per il filtro ottimo (nelle condizioni appena elencate) di Kalman: [26].

Il metodo *Grid-Based*[36] calcola ricorsivamente la la densità $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t})$ se lo spazio è discreto e consiste di un numero finito di stati $\mathbf{x}^{i}{}_{i=1}^{N_s}$. L'idea è di assegnare per ciascuno stato la probabilità condizionata alle misure $\mathrm{P}[\mathbf{x}_{t-1} = \mathbf{x}_{t-1}^{i} \,|\, \mathbf{y}_{1:t-1}] = w_{t-1|t-1}^{i}$. La ddp a-posteriori diventa:

$$p(\mathbf{x}_{t-1} \,|\, \mathbf{y}_{1:t-1}) = \sum_{i=1}^{N_s} w_{t-1|t-1}^{i} \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{i}); \tag{33}$$

sostituendo nella (9) e nella (32) la (33), si ricavano le formule per la predizione e l'aggiornamento che compongono il filtro Grid-Based[37].

Se l'evoluzione dello stato $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$ e il modello di misura $\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$ non sono lineari si ricorre usualmente al filtro di Kalman esteso (EKF). Si linearizzano localmente le funzioni $\mathbf{f}$ e $\mathbf{h}$ presupponendo comunque che la $P(\mathbf{x}_t \,|y_{1:t})$ sia gaussiana. Un'ulteriore rifinimento dell'EKF porta all'*unscented Kalman filter* (UKF)[38] che fa uso dell'omonima trasformata migliorando le approssimazioni delle funzioni non lineari $\mathbf{f}$ e $\mathbf{h}$ rispetto allo sviluppo di Taylor usato nell'EKF.

[36]Da cui poi gli algoritmi D/TA (Detecting/Tracking).

[37]Per maggiori dettagli si veda [22].

[38]Si veda poi l'estensione al filtro particellare in [27].

Nel momento in cui la supposizione di gaussianità della ddp a posteriori diventa troppo restrittiva, cioè si abbia una distribuzione multimodale o pesantemente asimmetrica, si deve ricorrere ad un Grid-Based filter oppure ad un particellare. Il metodo Grid-Based approssimato è usato per modelli con spazio di stato continuo: esso viene suddiviso in celle similmente a quanto fatto nel metodo non approssimato. Si comprende che per avere delle prestazioni accettabili la griglia deve essere sufficientemente fitta e, nel caso di spazio dello stato di dimensione infinita, saranno necessari dei troncamenti. Gli Hidden Markov Model (HMM) filter sono una applicazione del metodo Grid-Based approssimato.

A. Filtri Particellari

I filtri particellari, detti anche *Sequential Monte Carlo (SMC), bootstrap filter, condensation algorithm, interacting particle approximations e Survival of The Fittest (SoTF)*, permettono di implementare il filtraggio ricorsivo alla Bayes. L'idea fondamentale su cui si basa il funzionamento del filtro particellare è quella già espressa nel paragrafo VI-A.3: usando la notazione ivi introdotta, la densità a posteriori è approssimata dal set $\{\mathbf{x} : t^s, w_t^s\}_{s=1}^{S}$. Qui, in aggiunta a quanto trattato nel paragrafo VI-A.3, si presenteranno delle varianti al filtro SIR usato per la localizzazione e chiarimenti implementativi sulle sue componenti salienti.

1) Sampling Importance Resampling Filter (SIR): Questo filtro è una estensione del filtro particellare di base detto Sequential Importance Sampling (SIS) in cui la *importance density* (una preposta distribuzione di probabilità da cui sono estratte le particelle[39]) viene a coincidere con la a-priori $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}^{s})$. Questo permette notevoli semplificazioni nelle equazioni del SIS, in particolare per i pesi vale

$$w_t^s \propto w_{t-1}^s p(\mathbf{y}_t \,|\, \mathbf{x}_t^s). \tag{34}$$

Il ricampionamento (introdotto in VI-A.3) viene eseguito ad ogni iterazione per cui si ha $w_{t-1}^s = 1/S \;\forall s$ e la (34) diventa:

$$w_t^s \propto p(\mathbf{y}_t \,|\, \mathbf{x}_t^s)$$

Secondo quanto trattato in VI-A.3 e qui precisato, una iterazione del SIR è descritta in pseudocodice dall'algoritmo 1.

Il ricampionamento sistematico, così come presentato nell'algoritmo 3 è abbastanza semplice poiché impiega $O(S)$ e minimizza la variazione MC come dimostrato in [28]. Questo può comunque comportare dei problemi come la perdita di diversità tra le particelle con il risultato di avere molti stati ripetuti; il problema noto come *sample impoverishment* si verifica nei casi di rumore di processo molto piccolo con il collasso, in poche iterazioni, di tutte le particelle in un punto.

Per non appesantire a livello computazionale il filtro si può adottare una tecnica che prevede di effettuare il ricampionamento solo nel momento in cui l'algoritmo sta degenerando controllando l'effettivo numero delle particelle

$$S_{eff} = \frac{S}{1 + \mathrm{var}(\bar{w}_t^s)} \tag{35}$$

[39]Per maggiori chiarimenti si veda [22].

Algorithm 1 Filtro Particellare SIR

1: $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S$ = SIR $(\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^S, \mathbf{y}_t)$
2: **for** $s = 1$ **to** S **do**
3: $\mathbf{x}_t^s \sim p(\mathbf{x}_t^s \mid \mathbf{x}_{t-1}^s)$ ▷ Predizione secondo il modello di stato (noto)
4: $w_t^s := p(\mathbf{y}_t \mid x_t^s)$ ▷ Assegno i pesi alle particelle attraverso la funzione di verosimiglianza
5: **end for**
6: $totWeight :=$ SUM $\left(\{w_t^s\}_{s=1}^S\right)$ ▷ Peso totale
7: **for** $s = 1$ **to** S **do**
8: $w_t^s := \frac{w_t^s}{totWeight}$ ▷ Normalizza i pesi
9: **end for**
10: $\{\bar{\mathbf{x}}_t^j, w_t^j\}_{j=1}^S =$ RICAMPIONAMENTO$\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S$ ▷ secondo l'algoritmo 3
11: **end** SIR

con $\bar{w}_t^s = p(\mathbf{x}_t^s \mid \mathbf{y}_{1:t})/q(\mathbf{x}_t^s \mid \mathbf{x}_t^s \mid \mathbf{x}_{t-1}^s, \mathbf{y}_t)$ peso "vero" secondo la definizione della importance density. La (35) non è calcolabile esattamente per cui si usa una sua stima $\hat{S}_{eff} = 1/\sum_s (w_t^s)^2$. Valori piccoli di $\hat{S}_{eff}$ denotano degenerazione grave. La variante del filtro SIR che utilizza il ricampionamento solo nelle iterazioni che stanno portando alla degenerazione (i.e. quando $\hat{S}_{eff}$ è inferiore ad un opportuno valore di soglia) è riportata nell'algoritmo 2

Algorithm 2 Filtro Particellare generico

1: $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S$ = PF $(\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^S, \mathbf{y}_t)$
2: **for** $s = 1$ **to** S **do**
3: $\mathbf{x}_t^s \sim p(\mathbf{x}_t^s \mid \mathbf{x}_{t-1}^s)$ ▷ Predizione secondo il modello di stato (noto)
4: $w_t^s := p(\mathbf{y}_t \mid x_t^s)$ ▷ Assegno i pesi alle particelle attraverso la funzione di verosimiglianza
5: **end for**
6: $totWeight :=$ SUM $\left(\{w_t^s\}_{s=1}^S\right)$ ▷ Peso totale
7: **for** $s = 1$ **to** S **do**
8: $w_t^s := \frac{w_t^s}{totWeight}$ ▷ Normalizza i pesi
9: **end for**
10: $\hat{S}_{eff} := \frac{1}{\text{SUM}\left(\{w_t^s\}_{s=1}^S\right)}$
11: **if** $\hat{S}_{eff} < S_T$ **then** ▷ Sotto soglia: degenerazione
12: $\{\bar{\mathbf{x}}_t^j, w_t^j\}_{j=1}^S =$ RICAMPIONAMENTO$(\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S)$
13: **end if**
14: **end** PF

Nello specifico, il ricampionamento, sia che venga effettuato sistematicamente che solo alla degenerazione, è un'operazione che può avere diverse implementazioni; una prima soluzione consiste nel costruire una distribuzione di probabilità (Cumulative Density Function) attraverso la somma cumulativa dei pesi delle particelle. Da come sono stati definiti i pesi e da quanto riportato nell'algoritmo 3 si vede che il vettore c_i indicizzato da 1 a S rappresenta proprio una funzione di ripartizione discreta poiché soddisfa le proprietà:

- $0 \leq c_i \leq 1 \ \forall i$
- è funzione non decrescente (somma di positivi)
- $c_1 = 0$, $c_S = 1$

La CDF così definita viene confrontata con una CDF di riferimento costruita invece come somma cumulativa di $1/S$. Questo permette di identificare le particelle associate a pesi elevati (per effetto del calcolo di $\mathrm{P}[\mathbf{y}_t \mid \mathbf{x}_t]$) e di continuare a replicarle fintantoché la CDF di riferimento abbia raggiunto la CDF dei pesi. Specularmente le particelle a peso trascurabile vengono scartate: si veda la Fig. 47.

Algorithm 3 Ricampionamento

1: $\{\bar{\mathbf{x}}_t^j, w_t^j\}_{j=1}^S$ = RICAMPIONAMENTO $(\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S)$
2: $c_1 := 0$ ▷ Inizializzazione della CDF
3: **for** $i = 2$ **to** S **do**
4: $c_i := c_{i-1} + w_t^i$ ▷ Costruzione della CDF
5: **end for**
6: $i := 1$ ▷ Comincio dall'inizio della CDF
7: $u_1 \sim \mathcal{U}[0, \frac{1}{S}]$ ▷ Estraggo punto di inizio per CDF di riferimento
8: **for** $j = 1$ **to** S **do**
9: $u_j := u_1 + \frac{1}{S}(j-1)$ ▷ Scorro la CDF di riferimento
10: **while** $u_j > c_i$ **do**
11: $i{+}{+}$ ▷ Scarta le particelle a peso trascurabile oppure continua a creare nuove particelle da una con peso elevato
12: **end while**
13: $\bar{\mathbf{x}}_t^j := \mathbf{x}_t^i$ ▷ Assegna particella
14: $w_t^j := \frac{1}{S}$ ▷ Assegna peso
15: **end for**
16: **end** RICAMPIONAMENTO

Un ricampionamento alternativo all'algoritmo 3 viene proposto da Nando de Freitas in un esempio di implementazione del SIR in `http://www.cs.ubc.ca/\%7Enando/software.html`: lo pseudocodice è riportato nell'algoritmo 4

APPENDIX III
COMPONENTI TINYOS UTILIZZATI

A. TimerC

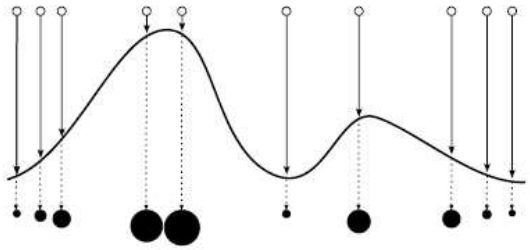
`TimerC` è il componente che realizza l'interfaccia `Timer`, riportata di seguito:

```
interface Timer {
 command result_t start(char type, uint32_t interval);
 command result_t stop();
 event result_t fired();
}
```

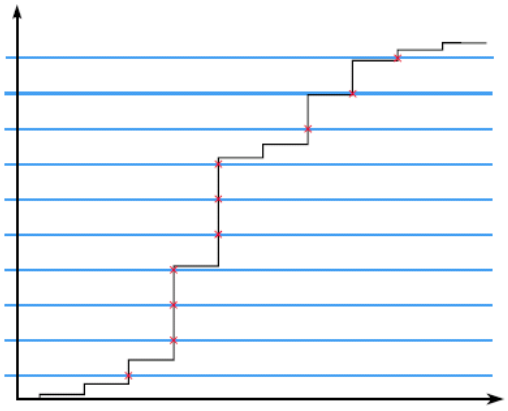
il significato dei comandi e degli eventi è facilmente deducibile dal loro nome; si noti altresì che tutti i comandi e gli eventi sono sincroni. `type` si riferisce alla modalità di funzionamento del timer, scelta fra `TIMER_REPEAT` (funzionamento a ciclo continuo) e `TIMER_ONE_SHOT` (viene effettuato un solo conteggio); `interval` si riferisce alla durata del timer espressa in *millisecondi binari (ms$_b$)*: 1 ms$_b = 1/1024$ s.

In realtà, in `TimerC` appare:

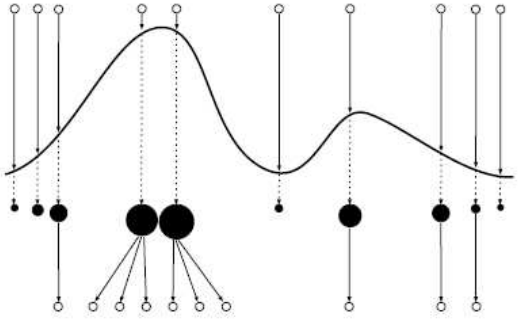
```
provides interface Timer[uint8_t id];
```



(a) Assegnazione del peso delle particelle sulla base delle osservazioni: $p(\mathbf{y}_t \mid \mathbf{x}_t)$



(b) Ricampionamento attraverso la CDF dei pesi e la CDF di riferimento: in nero la CDF dei pesi, in blu i livelli di quella di riferimento, in rosso gli indici delle particelle che verranno replicate



(c) Risultato: nuovo set di particelle, tutte con peso $1/S$

Fig. 47: Effetti del ricampionamento

Algorithm 4 Ricampionamento alternativo

$\{\bar{\mathbf{x}}_t^j, w_t^j\}_{j=1}^S = \text{RICAMP2}\ (\{\mathbf{x}_t^i, w_t^i\}_{i=1}^S)$
  for $i = 1$ **to** $S + 1$ **do**
    $u \sim \mathcal{U}(0,1)$ ▷ Estrae da una normale standard
    $l_i := -\ln(u)$ ▷ Vettore di numeri con distr. log-normale
  end for
  for $s = 2$ **to** S **do**
    $\bar{\mathbf{x}}_t^s := \mathbf{x}_c$ ▷ Inizializzazione nuove particelle in un punto predeterminato: alcune infatti possono non venire assegnate dal ciclo while
    $w_t^s := \frac{1}{S}$ ▷ Assegna gia il peso
  end for
  $c_1 := w_t^1$ ▷ Inizializzazione della CDF
  $q_1 := l_1$ ▷ Inizializzazione della CDF di riferimento
  for $i = 2$ **to** S **do**
    $c_i := c_{i-1} + w_t^i$ ▷ Costruzione della CDF
    $q_i := q_{i-1} + l_i$ ▷ Costruzione della CDF di riferimento
  end for
  while $j < S$ **do**
    if $c_j q_S > q_i$ **then**
      $\bar{\mathbf{x}}_t^i := \mathbf{x}_t^j$ ▷ Split delle particelle di peso elevato
      $i++$
    else
      $j++$ ▷ Particelle di peso trascurabile vengono saltate
    end if
  end while
end RICAMP2

ovvero viene implementata una versione parametrizzata dell'interfaccia; il valore `id` fa riferimento ad un valore identificativo per il timer. Ciò significa che la stessa applicazione può istanziare fino a 256 timer indipendenti, cambiando tutti i possibili valori di `id`. Per non gestire manualmente l'assegnazione di codici identificativi diversi, TinyOS mette a disposizione la funzione `unique()`, che genera costanti di 8 bit differenti a partire dalla stringa che è passata come parametro; ad esempio, per ciascun componente in cui compare `unique(String)` in fase di compilazione verrà generato un valore diverso. Si noti che `unique(String)` e `unique(anotherString)` possono dare luogo allo stesso valore, quindi all'interno della stessa applicazione l'univocità può essere mantenuta soltanto usando sempre la stessa stringa come parametro di `unique`. TinyOS prevede anche la funzione `uniqueCount(String)`, che genera il numero di valori generati da `unique(String)`.

B. GenericComm

La configurazione `GenericComm` implementa le interfacce `ReceiveMsg` e `SendMsg`, atte alla gestione primitiva della ricezione e dell'invio di dati:

```
interface SendMsg {
 command result_t send(uint16_t address,...
                 uint8_t length, TOS_MsgPtr msg);
 event result_t sendDone(TOS_MsgPtr msg,...
                result_t success);
}
interface ReceiveMsg {
   event TOS_MsgPtr receive(TOS_MsgPtr m);
}
```

Queste interfacce possono essere istanziate in modo parametrizzato, specificando l'*Active Message handler* del messaggio spedito/ricevuto; ciò consente di far viaggiare contemporaneamente nella rete messaggi con funzioni e destinatari diversi senza che vi siano collisioni.

Il file `AM.h` definisce tutte le costanti e le strutture dati che consentono lo scambio di dati; da notare la definizione delle strutture `TOS_Msg` e `TOS_MsgPtr`:

```
typedef struct TOS_Msg {
 uint16_t addr;
 uint8_t type;
 uint8_t group;
 uint8_t length;
 int8_t data[TOSH_DATA_LENGTH];
 uint16_t crc;
 uint16_t strength;
 uint8_t ack;
 uint16_t time;
 uint8_t sendSecurityMode;
 uint8_t receiveSecurityMode;
} TOS_Msg;
typedef TOS_Msg *TOS_MsgPtr;
```

i campi di `Tos_Msg` successivi a `data`, in realtà, non sono trasmessi, ma vengono riempiti dal dispositivo ricevente; in particolare, il campo `strength` contiene l'RSSI del messaggio ricevuto via radio. Si noti che il campo `data` ha una lunghezza definibile: questo permette una grande duttilità nell'invio delle informazioni. `TOS_MsgPtr` è un semplice puntatore a dati `TOS_Msg`: qualsiasi metodo di invio o ricezione implementato in TinyOS ha fra i parametri il puntatore al messaggio e non il messaggio stesso.

`GenericComm` si basa fondamentalmente sul modulo `AMStandard` che consente l'invio e la ricezione dei dati sia su porta seriale - con il componente `UARTFramedPacket` - sia tramite radio - con il componente `RadioCRCPacket`. In particolare, in fase di invio è semplicemente un controllo sull'indirizzo del destinatario che determina l'utilizzo della porta seriale o del *chip* radio; infatti l'implementazione del comando `send` posta il *task* `sendTask` che controlla:

```
if (buf->addr == TOS_UART_ADDR)
 ok = call UARTSend.send(buf);
else
 ok = call RadioSend.send(buf);
```

se l'indirizzo del destinatario è `TOS_UART_ADDR`, la costante che identifica la porta seriale, allora il pacchetto viene inviato sulla seriale, altrimenti viene inviato via radio. Il valore di `TOS_UART_ADDR` è definito sempre in `AM.h`.

C. CC2420ControlM

`CC2420ControlM` è il modulo che implementa i vari comandi di controllo al *chip* radio CC2420 della Chipcon, contenuti nell'interfaccia `CC2420RadioControl`:

```
interface CC2420Control {
 command result_t TunePreset(uint8_t rh,...
                  uint8_t channel );
 command result_t TuneManual(uint8_t rh,...
                  uint16_t freq );
 command uint8_t GetPreset();
 command uint16_t GetFrequency();
 async command result_t VREFOn();
 async command result_t VREFOff();
 async command result_t OscillatorOn(uint8_t rh);
 async command result_t OscillatorOff(uint8_t rh);
 async command result_t TxMode(uint8_t rh);
 async command result_t TxModeOnCCA(uint8_t rh);
 async command result_t RxMode(uint8_t rh);
 command result_t SetRFPower(uint8_t rh,...
                  uint8_t power );
 command uint8_t GetRFPower();
 async command result_t enableAutoAck(uint8_t rh);
 async command result_t disableAutoAck(uint8_t rh);
 async command result_t enableAddrDecode(uint8_t rh);
 async command result_t disableAddrDecode(uint8_t rh);
 command result_t setShortAddress(uint8_t rh,...
                  uint16_t addr );
}
```

si noti che la maggior parte dei comandi è `async`: ciò implica che tali comandi possono essere invocati direttamente dall'*interrupt*, senza postare un *task*. In particolare, nell'applicazione descritta si fa uso dei comandi `OscillatorOn`, `OscillatorOff` (per accendere e spegnere la radio), e `SetRFPower` (per impostare la potenza di trasmissione).

APPENDIX IV
GRIGLIA DELLE RILEVAZIONI

# rilev.	x[m]	y[m]	# rilev.	x[m]	y[m]
1	1	2.5	61	14.5	20.5
2	4.5	2.5	62	18	20.5
3	8	2.5	63	21.5	20.5
4	11.5	2.5	64	24.5	20.5
5	15	2.5	65	28	20.5
6	18.5	2.5	66	31.5	20.5
7	22	2.5	67	34.5	20.5
8	25.5	2.5	68	38.5	20.5
9	29	2.5	69	4	24
10	32.5	2.5	70	8.5	24
11	36	2.5	71	11.5	24
12	39.5	2.5	72	13.5	22.5
13	4	6	73	17.5	22.5
14	7.5	6	74	19	24
15	11	6	75	22.5	24
16	14.5	6	76	26	24
17	18	6	77	29.5	24
18	21.5	6	78	33	24
19	25	6	79	36.5	24
20	28.5	6	80	40	24
21	32	6	81	4	27.5
22	35.5	6	82	7.5	27.5
23	39	6	83	11.5	27.5
24	4	9.5	84	13	27.5
25	7.5	9.5	85	16.5	27.5
26	11	9.5	86	19	27.5
27	14.5	9.5	87	21.5	30
28	18	9.5	88	25	27.5
29	21	9.5	89	29	27.5
30	24.5	9.5	90	31	28.5
31	29	9.5	91	35.5	27.5
32	32.5	9.5	92	39	27.5
33	36	9.5	93	4	31
34	39	9.5	94	8	31
35	4	13	95	11.5	30.5
36	7.5	13	96	13.5	30.5
37	11	13	97	17	31
38	14.5	13	98	19	31
39	18	13	99	21	32.5
40	21.5	13	100	24.5	31
41	25	13	101	28	31
42	28.5	13	102	33	31
43	32	13	103	35.5	31
44	35.5	13	104	38.5	30.5
45	39.5	13	105	8	34.5
46	4	16.5	106	11.5	35
47	7.5	16.5	107	15	34.5
48	11	16.5	108	18.5	35
49	14.5	16.5	109	22	34.5
50	18	16.5	110	25.5	35
51	21.5	16.5	111	29	34.5
52	24.5	16.5	112	32.5	35
53	28	16.5	113	36	34.5
54	31	16.5	114	40	35
55	33	16.5	115	33.5	27
56	36.5	16.5			
57	39.5	16.5			
58	4	20.5			
59	7.5	20.5			
60	11	20.5			

Tab. 4: Coordinate delle rilevazioni per la costruzione della mappa a priori.

BIBLIOGRAFIA

[1] C. Swedberg, "Chicago Fire Dept. tests ZigBee-based RFID system," *RFID Journal*, October 2006.

[2] G. Gamba, "Sviluppo e test di una rete di sensori in tecnologia IEEE 802.15.4 per monitoraggio industriale," October 2006.

[3] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," Tech. Rep., Jan. 2000.

[4] M. G. Adam Smith, Hari Balakrishnan and N. Priyantha, "Tracking Moving Devices with the Cricket Location System," Tech. Rep., 2004.

[5] K. Lorincz and M. Welsh, "MoteTrack: A Robust, Decentralized Approach to RF-based location tracking," in *Division of Engineering and Applied Sciences, Harward University*, 2005.

[6] C.Morelli, V.Rampa, M.Nicoli, U.Spagnolini, and C.Alippi, "Particle Filters for RSS-based Localization in Wireless Sensor Networks: an Experimental Study," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2006.

[7] L. Parolini, "Metodi di localizzazione per reti di sensori wireless," 2005-2006.

[8] Y. Chraibi, "Localization in Wireless Sensor Network," in *Master's Degree Project, Stockholm, Sweden*, 2005.

[9] C. W. Lee Ee Foong and L. Xiao, "A Study of Radio Signal Behaviors in Complex Environments," Computer Science Department, Michigan State University, Michigan, Tech. Rep.

[10] M. N. Paolo Mazzoldi and C. Voci, *elementi di Fisica - Onde*. EdiSES, 2006.

[11] R. A. Valenzuela, "Ray Tracing Prediction of Indoor Radio Propagation."

[12] N. Benvenuto and G. Cherubini, *Algorithms for Communications Systems and their Applications*. WILEY, 2004.

[13] *Radioplan RPS User Manual - version 5.3*, Radioplan GmbH.

[14] *Tmote Sky: datasheet*, Moteiv Corporation, November 2006.

[15] *SmartRF CC2420 datasheet (revision 1.3)*, Chipcon AS, October 2005.

[16] I. P802.15™, "802.15.4™: Part 15.4: Wireless Medium Access Control (MAC) and Physical layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Computer Society, Tech. Rep., 2003.

[17] *PPP in HDLC-like Framing*, Daydreamer ed., Network Working Group.

[18] *The Point-to-Point Protocol (PPP)*, Daydreamer ed., Network Working Group.

[19] D. Gay, P. Lewis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: a holistic approach to network embedded systems," in *PLDI'03*, June 2003.

[20] P. Lewis, "TinyOS programming," www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf, October 2006.

[21] U. Spagnolini and A. Bosisio, "Indoor localization by attenuation maps: model-based interpolation for random medium," in *ICEAA Internation Conference on Electromagnetics in Advanced Application*, Sept. 2005.

[22] M. S. Arulampalam, S. Maskell, N. Gornod, and T. Clapp, "A tutorial on Particle Filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE transaction on signal processing*, vol. 50, no. 2, 2002.

[23] G. Bolondi, F. Rocco, and S. Zanoletti, "MAutomatic contouring of faulted subsurfaces," *Geophysics*, vol. 41, no. 6, pp. 1377–1393, 1976.

[24] C. Morelli, M. Nicoli, V. Rampa, and U. Spanolini, "Hidden Markov Models for radio localization in Mixed LOS/NLOS conditions," *IEEE transaction on signal processing*, vol. 55, no. 4, 2007.

[25] F. Roveron, "Analisi sperimentale di connettività per una rete di sensori wireless," 2006-2007.

[26] G. Picci, *Fitraggio statistico (Wiener, Levinson, Kalman) e applicazioni*. Libreria Progetto Padova, 2006.

[27] R. van der Merve, A. Doucet, N. de Freitas, and E. Wan, "The Unscented Particle Filter," Cambridge University Engineering Department, U.K., Tech. Rep. CUED/F-INFENG/TR 380, 2000.

[28] G. Kitagawa, "Monte Carlo filter and smoother for non-gaussian non-linear state space models," *J. Comput. Graph. Statist.*, vol. 5, no. 1, 1996.

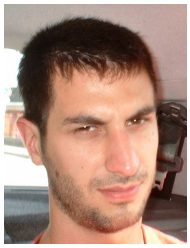
RINGRAZIAMENTI

Gli Autori hanno il piacere di ringraziare il prof. Luca Schenato ed il prof. Angelo Cenedese, che hanno reso possibile questo progetto, il prof. Alessandro Chiuso per i suoi preziosi suggerimenti, il dott. Sergio Sartor e lo staff dell'Ufficio tecnico dell'INFN per l'accoglienza dimostrata nei loro confronti e tutti i ragazzi del NAVLAB del DEI per la disponibilità offerta.

Vorrebbero inoltre qui ricordare: le mail dei proff. Spagnolini e Natali del Politecnico di Milano, Mastrolindo di Guardiania, le concubine del Collegio S. Cuore, i condimenti della mensa dell'INFN, il divano scomodo di Contino Bricchio e i vari *bear* e *twink* incontrati lungo il cammino.

Marco Bertinato (Cittadella, 12/10/1983) Marco, universalmente riconosciuto come Berty, nasconde un torbido passato di maltrattamenti familiari ad opera delle sorelle più anziane, da cui è uscito solo grazie alla fede incrollabile in S. Patata Vergine ed in Bamma. Nel tempo libero ama girare in macchina come passeggero, attendendo un incidente a colpa di terzi che gli permetta di pagarsi delle insostenibili spese sanitarie.

Giulia Ortolan (Ivrea, 09/11/1984) La più piccola del gruppo è senza ombra di dubbio tra le migliori studentesse di tutta la facoltà. E' latinista-spacca-maroni-correggo-tutto-quello-che-voi-scrivete; è il supremo sys-admin del collegio femminile ("dalle suore") in cui alloggia, nonché imperatrice incontrastata di tutto quel pollaio in cui il sacro si fonde discutibilmente al profano. Nel tempo libero tampona a destra e a manca in autostrada... forse perché è anche una bevitrice di birra stile catino senza fondo.



Patrizio Zambotto (Cologna Veneta, 21/12/1983) Patrizio, dopo una brillante carriera universitaria in cui ha maturato il valore del *rispetto* verso la figura generica del Docente, si appresta ad una sfavillante carriera nel campo della gestione di locali. A questo proposito è già alla ricerca di un'avvenente barista trentacinquenne, meglio se divorziata e con un'ottima predisposizione alle pubbliche relazioni.

Gli Autori desiderano inoltre inserire una breve descrizione delle figure ispiratrici che li hanno accompagnati durante tutto questo percorso:

Bamma Di professione sosia, coltiva l'hobby della prostituzione e crede fermamente a Babbo Natale. Arrabbiata dalla nascita a causa di un malessere, ha partecipato al reality *Il divano scomodo* per burla.

Contino Bricchio (detto Amando) Emette soltanto insulti a mezza bocca; arrabbiato per una questione di danaro, ha partecipato al reality *Il divano scomodo* sennò a casa si annoiava.

Gli Autori, infine, vogliono ricordare che Donald Knuth[40] non utilizza la posta elettronica dal 1 gennaio 1990, sostenendo che 15 anni di utilizzo dell'*e-mail* sono più che sufficienti nell'arco di una vita.

[40] Pron. Ka-nuuth.