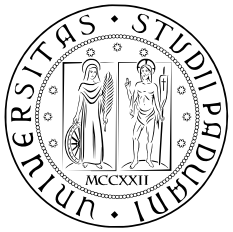


UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria Elettronica

Tesina di Laurea Triennale

Simulazione cross-layer di tecniche di ritrasmissione in reti wireless interferenti 802.11 e 802.15.4

Relatore:
Prof. Matteo Bertocco
Correlatore:
Dott. Federico Tramarin

Laureando:
Riccardo Bersan
580511/IL

Padova, 21 Febbraio 2011
ANNO ACCADEMICO 2010 - 2011

Prefazione

Questa tesi si propone come continuazione ed estensione del lavoro di tesi magistrale del dott. Federico Tramarin [1] sulla base delle informazioni tratte dall'articolo "*Retransmission strategies for cyclic polling over wireless channels in the presence of interference*", [2], vincitore del premio *Best Paper Award* della IEEE-ETFA (*Emerging Technology and Factory Automation*).

Nell'ottica di continuazione, alcune parti di questa tesi prendono spunto dai contenuti del lavoro precedente; in particolare, nel Capitolo 1 e nel Capitolo 2, è possibile trovare alcuni passaggi che si ispirano ad altrettante sezioni della tesi stessa.

Questa scelta è stata ponderata ritenendo superflua, in questa sede, una riscrittura ex-novo dei medesimi contenuti, privilegiando l'originalità dei contenuti della seconda parte della tesi.

L'autore.

Indice

Introduzione

L'uso di reti di sensori senza fili è un campo di interesse che, negli ultimi anni, sta assumendo un ruolo di primaria importanza in molti ambiti di applicazione. Queste reti, denominate usualmente Wireless Sensor Network (WSN), trovano impiego nell'automazione industriale, nel controllo di processi produttivi, nel monitoraggio ambientale, nella domotica e in molti altri settori [3] [4] [5].
Il principale fattore di successo delle WSN sta nella flessibilità di utilizzo che si ottiene eliminando i cablaggi, nella conseguente riduzione dei costi e nell'estrema scalabilità della rete formata, che può quindi aumentare o ridurre le proprie dimensioni a seconda delle necessità.
Una WSN è composta da un insieme di *nodi* che, tramite l'uso di sensori, sono in grado di raccogliere informazioni, che poi potranno essere elaborate e trasmesse via radio all'interno della rete: qui, saranno presenti altri nodi in grado di raccogliere le informazioni provenienti dalle rete e di utilizzarle secondo i casi.

Problematiche connesse alle WSN

L'uso di reti non cablate introduce, per contro, alcune nuove criticità, legate principalmente ai consumi energetici e ai fenomeni di interferenza legati alla coesistenza con altre sorgenti elettromagnetiche.

Consumi energetici Il consumo energetico della WSN riveste un ruolo di fondamentale importanza per caratterizzare l'affidabilità del sistema poiché, affinché la rete sia veramente wireless, è necessario che ogni nodo disponga di una sorgente di energia elettrica indipendente. La limitatezza dell'energia disponibile impone da un lato, lo studio di tecniche di stoccaggio dell'energia (associata, a volte, a tecniche di auto-alimentazione dei nodi), dall'altro la messa a punto di elaborazioni numeriche a basso consumo energetico. Mentre il primo aspetto prevede uno studio dal punto di vista architetturale del nodo, per quanto riguarda il secondo fattore è importante che le operazioni svolte e il processing delle informazioni siano poco gravose dal punto di vista energetico: questo si traduce nella scelta di protocolli che prevedano un basso livello di elaborazione dati da parte dei

nodi e una modesta potenza di trasmissione. É proprio in quest'ottica che si colloca la scelta del protocollo IEEE 802.15.4 per definire le caratteristiche dei nodi sensori: come sarà discusso nel Capitolo 1 questo protocollo indica una serie di procedure standardizzate che consentono di rispettare le specifiche sui consumi energetici e sulle capacità di elaborazione.

Interferenza La seconda criticità che si incontra nell'uso di reti wireless è l'insorgenza di fenomeni di interferenza che si manifestano quanto si verifica una coesistenza *temporale* tra due segnali che occupino la stessa banda di *frequenze*. Infatti, quando due segnali vengono trasmessi contemporaneamente nel mezzo, quello che viene visto dal ricevitore può essere pensato come la sovrapposizione dei due, con un possibile degrado della qualità del segnale utile e la perdita del contenuto informativo. Tuttavia, nell'ipotesi semplificativa che i dispositivi di ricezione siano provvisti di filtri selettivi che consentono di isolare la banda di interesse, solo i segnali interferenti che occupano la stessa banda del segnale utile possono realmente compromettere la qualità della trasmissione.

La gestione di questo tipo di fenomeni è già prevista dal protocollo IEEE 802.15.4 tramite il meccanismo di ascolto del canale prima di ogni trasmissione: questo algoritmo, che prende il nome di CSMA-CA, impone ai nodi di controllare se il mezzo è già impegnato da un'altra trasmissione prima di iniziare la propria. Tuttavia in alcune applicazioni, come nel caso di sistemi di controllo, il meccanismo CSMA-CA può ritardare l'invio del messaggio e la trasmissione può avvenire fuori tempo utile compromettendo l'affidabilità complessiva del sistema. Si rivela necessario, quindi, investigare il caso dell'interferenza per studiare in che modo e in che misura questa possa alterare il normale funzionamento della rete di sensori.

Di particolare interesse è lo studio di fenomeni interferenziali dovuto alla *coesistenza di reti* diverse che condividano lo stesso insieme di frequenze: in queste situazioni infatti, l'interferenza non è più un fattore trascurabile poiché la sovrapposizione temporale di pacchetti informativi appartenenti alle due reti rappresenta una condizione non sporadica. Preso in esame il protocollo IEEE 802.15.4 per la rete di sensori, il caso di maggior interesse pratico è la condivisione del mezzo con una rete che si basi sulla famiglia di protocolli IEEE 802.11, utilizzato normalmente, anche in ambito industriale, per la distribuzione di connessioni a banda larga nella zona di osservazione.

Entrambi i protocolli consentono la trasmissione su diverse bande ma si evidenzia una loro sovrapposizione quando le due reti realizzano trasmissioni a frequenze comprese tra 2.4 e 2.5 GHz: questa banda di frequenze è una porzione della banda Industrial Scientific and Medical (ISM) ed ha la caratteristica di essere utilizzabile liberamente all'interno di proprietà private, per applicazioni radio non commerciali o per uso industriale, scientifico e medico. Una rappresentazione grafica di questa sovrapposizione può essere osservata Figura 1.

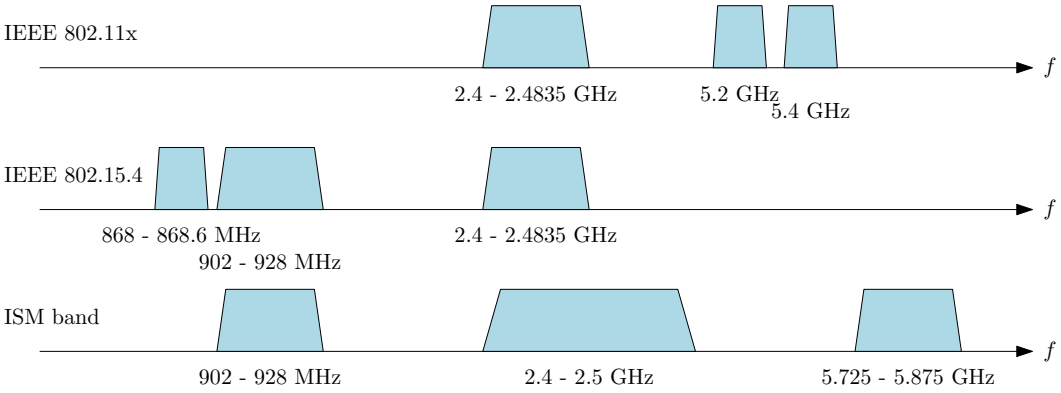


Figura 1: Rappresentazione delle frequenze utilizzate dai protocolli: si osserva una sovrapposizione nella banda ISM centrata in 2.45 GHz

La simulazione come strumento d'indagine

L'osservazione di ciò che accade in corrispondenza della sovrapposizione di due reti indipendenti e interferenti, come descritte sopra, può essere svolta in via analitica, in via sperimentale, come descritto in [6], oppure ricorrendo ad un software di simulazione. Tuttavia, la simulazione congiunta del funzionamento delle due reti è un'operazione complessa che richiede di definire nel dettaglio il comportamento dei dispositivi reali, tenendo conto di tutti i parametri fisici di interesse. Appare evidente che la simulazione non potrà mai porgere *esattamente* i risultati che si ottengono dalle verifiche sperimentali proprio perché il modello utilizzato si discosta necessariamente da quello reale. Se ne deduce che, tanto più accurato è il modello che si utilizza, tanto minore sarà il divario tra i risultati che si ottengono e quelli sperimentali.
Per la simulazione delle due reti si è utilizzato OMNeT++, un software open-source di simulazione di sistemi a eventi discreti, che sarà descritto in dettaglio nel Capitolo 2. Il principale punto di forza di questa applicazione è sicuramente l'elevata espandibilità, grazie ad un sistema basato sulla creazione di *moduli* tra loro interconnessi, che l'ha reso il simulatore più utilizzato dalla comunità scientifica globale per la simulazione e l'analisi di reti di telecomunicazioni. Grazie alla sua diffusione e alla modularità che lo contraddistingue, è possibile trovare on-line le implementazioni dei protocolli più diffusi tra cui anche le implementazioni per protocolli IEEE 802.11 e IEEE 802.15.4 rilasciati sotto licenza pubblica, ed utilizzarle per gli scopi prefissati.
Detto questo, la possibilità di effettuare simulazioni che coinvolgano *più* reti con protocolli diversi non è nativamente parte dei progetti considerati, che si limitano a sintetizzare il comportamento di una rete: questa possibilità è offerta da un modulo apposito [7], [8], che consente di valutare gli effetti di interferenza di una rete sull'altra, come sarà descritto nel Capitolo 2.

Grazie a questo modulo e alle implementazioni già realizzate da altre università e

gruppi di ricerca nel mondo, è possibile focalizzare la propria attenzione sull'aspetto che si desidera analizzare, limitandosi ad apportare le modifiche e le aggiunte necessarie. Questo modo di procedere si rivela estremamente vantaggioso, evidenziando l'importanza del lavoro *d'equipe* e l'efficacia di una mentalità orientata alla condivisione del lavoro e della conoscenza, per il raggiungimento di un vantaggio comune.

L'obiettivo della tesi

Come osservato, i fenomeni di interferenza possono causare la perdita di pacchetti di informazione con un conseguente calo delle prestazioni generali; per ridurre l'entità di questi eventi, i protocolli considerati sfruttano degli accorgimenti che prendono il nome di *tecniche di ritrasmissione*. Queste consistono nel rinvio dei pacchetti non ricevuti, secondo regole che saranno esposte nel Capitolo 3; tali tecniche prendono spunto dall'articolo citato nella Prefazione ([2]) in cui si analizzano e si comparano tra loro quattro strategie al fine di evidenziarne le caratteristiche peculiari. Lo scopo di questa tesi è quindi implementare nell'ambiente di simulatore OMNeT++ le tecniche di ritrasmissione presentate in questo documento, osservando come variano le prestazioni della comunicazione al variare della tecnica utilizzata, ottenendo così uno strumento di analisi flessibile ed adattabile a diverse configurazioni di rete. Nel Capitolo 4 verranno, poi, presentate le implementazioni delle strategie di ritrasmissione considerate; infine nel Capitolo 5 verranno descritte alcune simulazioni eseguite e i risultati ottenuti.

Capitolo 1

Il protocollo IEEE 802.15.4

Introduzione

Il protocollo IEEE 802.15.4 [9] definisce le caratteristiche del livello fisico, o PHY, e del livello MAC del modello ISO/OSI, pensato per reti wireless a basso bit rate e con una copertura spaziale poco estesa, classificabile come Personal Area Network (PAN). Questo protocollo offre una descrizione dei livelli considerati che coniuga:

- un basso bit-rate;
- un basso costo;
- una copertura ridotta;
- una bassa potenza;
- la possibilità di formare reti;
- un ridottissimo consumo energetico.

Come già osservato nell'introduzione, sono proprio queste caratteristiche a rendere l'IEEE 802.15.4 una promettente via alla realizzazione di reti di sensori wireless.

1.1 Introduzione ai protocolli

In generale un protocollo definisce in modo formale delle norme che due o più dispositivi tra loro interconnessi devono seguire, al fine di stabilire tra loro una comunicazione. L'aderenza ad un comune protocollo consente il dialogo tra due apparecchi che possono essere costruiti in modo indipendente da produttori diversi, garantendo una maggiore flessibilità della rete.

Il modello di riferimento nelle reti di calcolatori è il modello ISO/OSI, osservabile in Figura 1.1: esso definisce una precisa suddivisione in sette livelli o *layer* costruiti uno sopra all'altro, che ogni dispositivo della rete implementa. Lo scopo di ogni strato è quello di fornire servizi agli strati di livello superiore, in modo che lo strato n-esimo di un dispositivo di rete sia in comunicazione con lo stesso strato di tutti i dispositivi della rete tramite delle regole e delle convenzioni comuni denominate *protocolli*. Lo scambio materiale di informazioni avviene sempre tra due livelli contigui attraverso la trasmissione di pacchetti: l'informazione viaggia dal livello superiore al livello inferiore fino al raggiungimento del livello fisico, che permette una comunicazione diretta tra dispositivi. Durante questo percorso ogni layer modifica il pacchetto ricevuto dal livello superiore aggiungendo delle informazioni proprie in testa e in coda al pacchetto ricevuto oppure rimuovendole se il pacchetto proviene dal layer inferiore; in questo modo ogni layer comunica con i layer di pari livello leggendo le informazioni di testa e di coda, inviando eventualmente il payload al livello superiore.

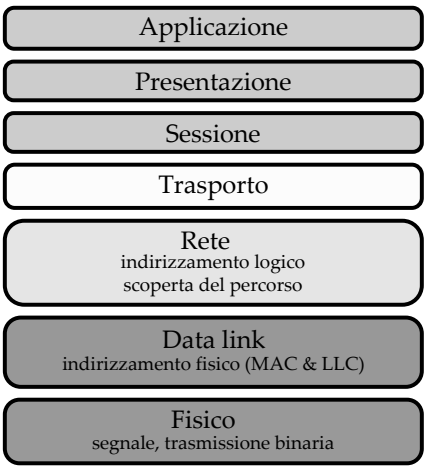
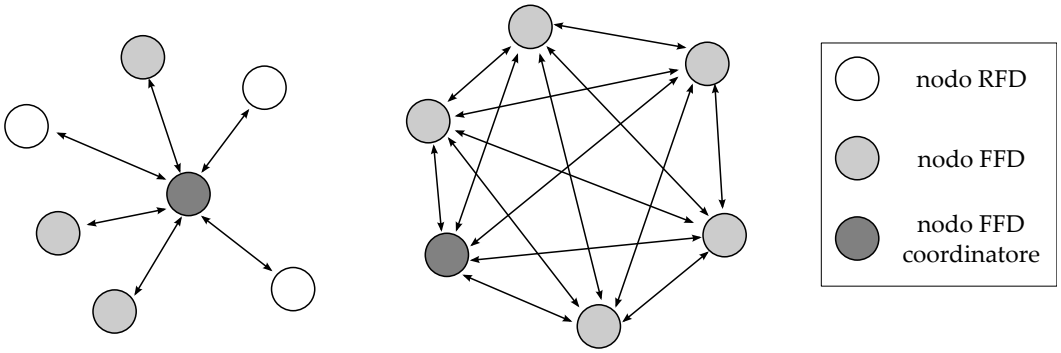


Figura 1.1: Rappresentazione dello stack ISO/OSI

Dopo aver osservato questa stratificazione, è doveroso fare una distinzione tra il protocollo IEEE 802.15.4 e lo standard Zigbee: la Zigbee Alliance è un'associazione di compagnie che lavorano assieme per formare uno standard globale per dispositivi wireless che siano affidabili, economici e a basso consumo energetico, mentre il gruppo 4 dell'IEEE 802.15 è un gruppo di ricerca che ha messo a punto un protocollo con specifiche di basso livello sugli stessi argomenti. Anche se il nome commerciale di questa tecnologia rimane ZigBee, al giorno d'oggi si utilizza il protocollo IEEE 802.15.4 per la progettazione dei livello fisico e del sublayer MAC, mentre lo standard ZigBee specifica il funzionamento dei layer superiori.

1.2 Componenti e topologia di rete

Il protocollo IEEE 802.15.4 specifica l'esistenza di due diversi tipi di nodi: essi possono essere di tipo Reduced-function Device (RFD) oppure Full-function Device (FFD) ma ogni rete deve includere almeno un nodo di tipo FFD che svolga la funzione di coordinatore di rete. La differenza tra questi due tipi di nodi risiede proprio nella capacità o meno di svolgere la funzione di coordinatore; resta inteso che, ogni nodo di tipo FFD può, all'occorrenza comportarsi da nodo RFD, utilizzando un insieme ridotto delle sue



a) topologia a stella b) topologia a peer-to-peer

Figura 1.2: Esempi di topologie per le reti di sensori.

funzioni. Con questi nodi è possibile ottenere diverse topologie, come quelli illustrate in Figura 1.2.

1.3 Il PHY Layer

Secondo quanto specificato nel protocollo, il layer Physical (PHY) ha il compito di fornire agli strati superiori due tipi di primitive: la prima è il *PHY data service* che si occupa della trasmissione e della ricezione di pacchetti, denominati Packet Protocol Data Unit (PPDU); la seconda è il *PHY menagement service*, il cui compito è fornire servizi ai livelli superiori, dietro loro richiesta. L'unità logica che si occupa della gestione dei servizi prende il nome di Physical Layer Management Entity (PLME) e rende accessibili agli strati superiori i seguenti servizi:

- Attivazione e disattivazione dell'unità ricetrasmittente;
- Energy Detection sul canale corrente;
- Link Quality Indication per i pacchetti ricevuti;
- Clear Channel Assessment per il Carrier Sense Multiple Access with Collision Avoidance;
- Selezione del canale su cui trasmettere;
- Ricezione e trasmissione di dati.

Queste funzionalità, che saranno descritte in modo generale nelle sezioni successive, vengono generalmente invocate con delle chiamate identificate da opportuni codici, i cui nomi sono indicati nel protocollo.

Tabella 1.1: Identificazione dei canali tramite pagina di canale e numero di canale

Channel page	Channel number	Banda occupata	Modulazione
0	0	banda a 868 MHz	BPSK
	1-10	banda a 915 MHz	BPSK
	11-26	banda a 2.4 GHz	O-QPSK
1	0	banda a 868 MHz	ASK
	1-10	banda a 915 MHz	ASK
	11-26	Canali riservati	
2	0	banda a 868 MHz	O-QPSK
	1-10	banda a 915 MHz	O-QPSK
	11-26	Canali riservati	
3-13	Canali riservati per uso futuro		

1.3.1 Frequenze utilizzate

Alla definizione di livello fisico è importante associare una specificazione delle frequenze in gioco. Le frequenze di trasmissione e le modulazioni impiegate sono definite attraverso una combinazione di un numero di canale e di una pagina di canale: in particolare il concetto di *channel page* è stato introdotto per distinguere i tipi di PHY supportati in modo da consentire ulteriori usi delle stesse bande con tecniche di modulazione diverse. L'unico riferimento alla banda ISM da 2.4 GHz si trova nella channel page 0, che sarà quindi l'unica considerata in questa tesi: uno schema riassuntivo con tutte le bande utilizzabili e le modulazioni associate è comunque osservabile in Tabella 1.1.

Nella channel page considerata, sono disponibili 27 canali, numerati da 0 a 26: un canale è disponibile nella banda da 868 MHz, dieci sono disponibili nella banda da 915 MHz mentre sono sedici i canali nella banda ISM da 2.4 GHz, come osservabile in Figura 1.3. Questi canali hanno frequenza centrale così definita:

$$
\begin{aligned}
F_c &= 868.3 \text{ MHz}, && \text{per } k = 0 \\
F_c &= 906 + 2(k-1) \text{ MHz}, && \text{per } k = 1, 2, \ldots, 10 \\
F_c &= 2405 + 5(k-11) \text{ MHz}, && \text{per } k = 11, 12, \ldots, 26.
\end{aligned}
$$

1.3.2 Energy Detection

Il servizio di rilevazione della quantità di energia, prevista da questo protocollo, consente di selezionare in modo dinamico il canale da utilizzare, ricercando sempre quello in cui è presente il minor livello di energia. Questo servizio sta alla base dell'algoritmo per la selezione del canale e consente di realizzare la trasmissione sempre nelle condizioni ottimali. Il servizio utilizza un tempo pari a 8 periodi di simbolo per produrre un'indicazione

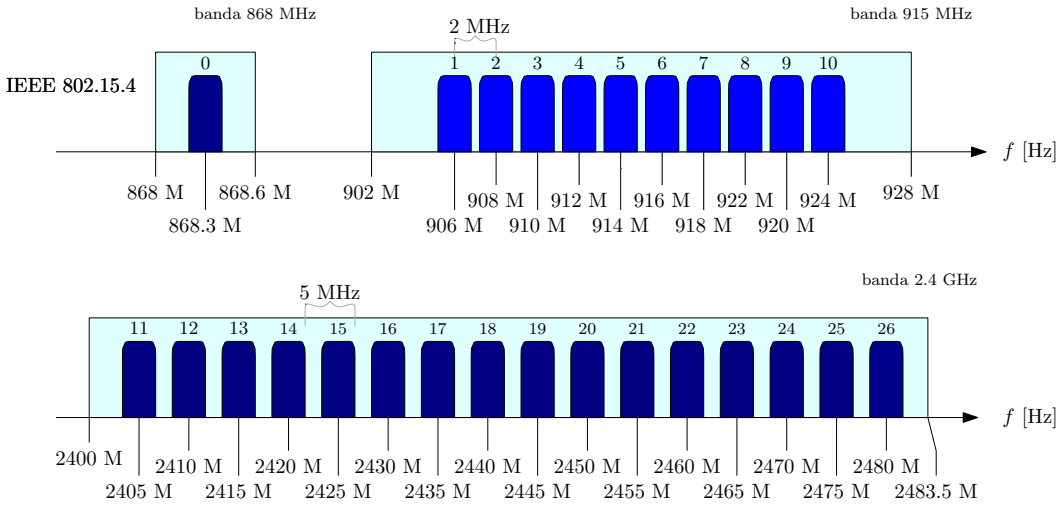


Figura 1.3: Bande occupate dai canali del protocollo IEEE 802.15.4 (con riferimento alla *channel page* 0).

della potenza rilevata su un determinato canale e memorizzarla in un registro ad 8 bit il cui valore sia proporzionale alla grandezza osservata. Il protocollo prevede che un valore pari a zero rappresenti la rilevazione di una quantità di potenza che sia sopra al livello di sensibilità di meno di 10 dB, mentre l'intero range deve coprire linearmente una dinamica di 40 dB.

1.3.3 Link Quality Indication

L'indicatore di livello di qualità è un sevizio che valuta la forza e la qualità di ogni pacchetto ricevuto, utilizzando il servizio di Energy Detection (ED), un estimatore del rapporto segnale rumore o una combinazione di queste due tecniche. Tale misura deve poi essere comunicata al Medium Access Control (MAC) layer, per una lunghezza di 8 bit, in cui i valori di massimo e di minimo sono relazionati alla massima e minima qualità individuabile e ripartiti tra questi su scala lineare.

1.3.4 Clear Channel Assessment

Il PHY layer fornisce la possibilità di realizzare una *dichiarazione di canale libero*, secondo una delle modalità previste:

CCA mode 1 dichiarazione di canale libero in base alla quantità di energia rilevata, paragonata con una soglia prestabilita;

CCA mode 2 dichiarazione di canale libero su rilevamento della portante, ovvero solo se non è rilevata la presenza di altre trasmissioni afferenti o compatibili al protocollo della rete;

bytes : 4	1	1		Variabile
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY Payload

Figura 1.4: Struttura del PPDU per il protocollo IEEE 802.15.4

CCA mode 3 dichiarazione di canale libero su rilevamento della portante con energia rilevata sopra una determinata soglia, ovvero una combinazione logica delle due tecniche precedenti.

Per ciascuna delle modalità scelte, il compito di questa primitiva è identificare se il mezzo è occupato e/o se si sta ricevendo un pacchetto, servendosi anche degli altri servizi del layer; la misura della quantità di energia, ad esempio, può essere realizzata dalla primitiva ED, descritta precedentemente. La ricezione è considerata *in corso* a partire dalla ricezione dello start-frame fino alla ricezione di tutti gli ottetti indicati nell'apposito campo nella testa del frame.

1.3.5 Descrizione del PHY frame

In una rete di comunicazione dello standard ISO/OSI, un pacchetto "viaggia" attraverso il nodo prima di raggiungere il mezzo fisico ed essere trasmesso. Questo comporta che il pacchetto, la cui trasmissione è usualmente generata a livello Applicazione, deve attraversare tutto lo stack protocollare e, ad ogni passaggio, qualche informazione viene aggiunta al contenuto informativo ricevuto dal livello superiore, che è denominato *payload*. Queste aggiunte vengono solitamente poste davanti ai dati ricevuti e prendono quindi il nome di *header*, mentre se vengono aggiunte nella parte finale, vengono identificate con il nome di *tail*. L'ultimo layer a cui il pacchetto appartiene è il livello fisico; qui è aggiunto l'ultimo header e viene formato il frame da inviare, che prende il nome di PPDU. Questo pacchetto sarà infine modulato e trasmesso via radio. Come illustrato in Figura 1.4 la prima parte del pacchetto è l'haeder di sincronizzazione o Synchronization Header (SHR), suddiviso in preambolo e delimitatore di sincronismo — Synchronization Frame Delimiter (SFD).
Questa è una parte essenziale del pacchetto poiché il ricevitore fa riferimento a questa parte per ottenere la sincronizzazione di cui ha bisogno; per ottimizzare il processo, il preambolo è generalmente composto di una sequenza pseudo-casuale con correlazione quasi impulsiva. La seconda sezione del PPDU contiene informazioni sulla lunghezza del pacchetto ed è seguita dal contenuto informativo del livello superiore.

1.3.6 Trasmissione

La trasmissione vera e propria dei pacchetti di bit avviene in tre fasi: in una prima fase i bit vengono convertiti in simboli, successivamente ad ogni simbolo viene associata una sequenza di bit, detta *chip*, ed infine ciascun chip viene modulato per essere trasmesso. Uno schema delle tre fasi è rappresentato in Figura 1.5.

Da bit a chip La conversione tra bit e simboli avviene a gruppi di quattro: i bit vengono processati a partire dall'header del PPDU fino all'ultimo bit della coda, in modo che da ogni ottetto si possano ricavare due simboli, fino ad un massimo di 127 bytes. Durante la seconda fase a ciascuno di questi simboli viene associata una sequenza pseudo-casuale con lunghezza 32 bit, che consente di ottenere segnali con una forte auto-correlazione e tra loro quasi ortogonali, con i conseguenti vantaggi per quanto riguarda l'immunità al rumore e la riconoscibilità.

Modulazione La sequenza di bit così ottenuta viene modulata con modulazione Offset - Quadrature Phase-Shift Keying (O-QPSK) e impulso a mezzo arco di seno: questa può essere vista come una sovrapposizione di due modulazioni BPSK tra loro sfasate di mezzo periodo (*chip period*). Dal punto di vista pratico i chip bit con indice pari e quelli con indice dispari vengono separati e modulati BPSK in modo indipendente. Successivamente una delle due componenti viene ritardata ed infine si procede alla somma e alla trasmissione su antenna. Dal diagramma della costellazione notiamo che lo sfasamento temporale tra le due componenti impone che la transizione tra due simboli avvenga sempre tra due punti contigui: il segnale trasmesso mantiene, quindi, energia quasi costante per tutta la trasmissione, consentendo una maggior facilità di realizzazione dei dispositivi ricetrasmettenti.

1.4 Il MAC Layer

Il MAC Layer è senza dubbio il più critico livello dello stack poiché le sue caratteristiche determinano le prestazioni e le funzionalità dell'interno standard. La complessità di questo layer nello standard IEEE 802.15.4 impedisce di esporre in modo completo tutte le sue caratteristiche in questa sede, in cui considereremo solo le principali capacità. Il layer fornisce due tipi di servizi: il primo tipo di servizi è fornito dal *MAC Data Service*, che permette la trasmissione e la ricezione di pacchetti MAC (o MPDU) verso il livello fisico; il secondo tipo di servizio è erogato dal *MAC Management Service* grazie al quale si può accedere ad alcuni utili servizi presso il gestore di livello denominato MAC Layer Management Entity (MLME). I principali servizi offerti sono:

- gestione dei pacchetti di sincronismo (*beacon*);

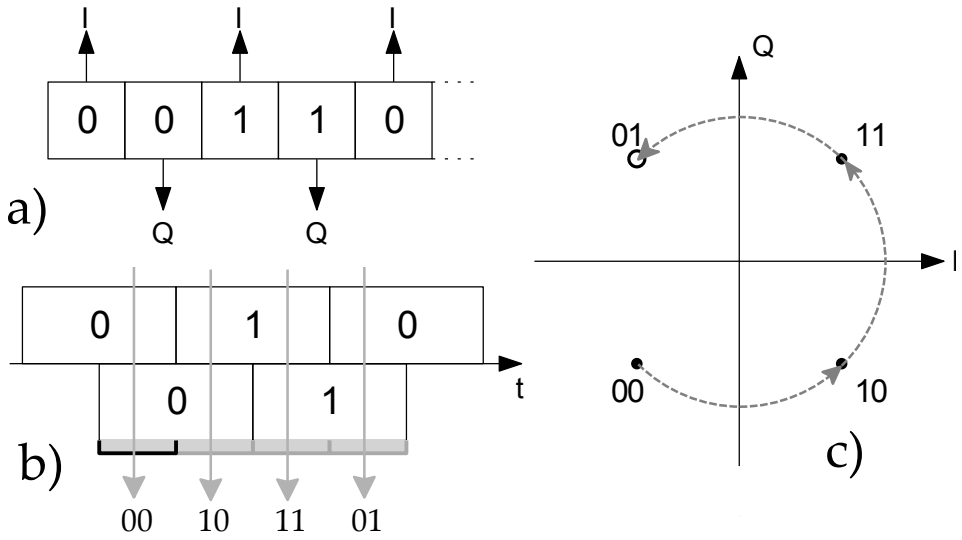


Figura 1.5: Schema funzionale della modulazione O-QPSK. Possiamo osservare: a) la suddivisione dei bit del PPDU nelle due componenti, in fase (I) e in quadratura (Q); b) le due componenti non cambiano valore simultaneamente, ma in modo alternato; c) la costellazione della modulazione, con esempio.

- accesso al canale;
- gestione dei tempi con suddivisione in slot temporali;
- invio dei pacchetti di Acknowledgement;
- de/associazione ad una rete PAN;
- servizi legati alla sicurezza e all'autenticazione.

Per garantire l'affidabilità della comunicazione, i pacchetti possono richiedere delle conferme di corretta ricezione da parte dei nodi riceventi, sotto forma di ACK frame. Questo protocollo supporta la gestione automatica delle ritrasmissioni come algoritmo di correzione degli errori.

1.4.1 L'algoritmo CSMA-CA

Nelle comunicazioni wireless, la difficoltà principale è gestire l'accesso al mezzo; un nodo che deve comunicare con un altro non ha una via dedicata per raggiungere la sua destinazione ma deve inviare le sue informazioni attraverso il canale — l'aria — che è condiviso con tutti gli altri nodi della rete. Se non consideriamo la presenza degli altri dispositivi è probabile che la trasmissione sia ostacolata da altre trasmissioni che, a loro volta, non conoscono l'esistenza della nostra trasmissione. É quindi necessario provvedere ad un algoritmo con il quale tutti i nodi possano raggiungere alti livelli di

probabilità di trasmissione, sebbene sia illusorio pensare di annullare l'eventualità di trasmissioni non riuscite.

In alcuni casi possono essere utilizzati metodi di accesso di tipo *deterministico*, ripartendo tra i nodi determinati slot temporali — Time Division Multiple Access (TDMA) — o particolari frequenze di esercizio — Frequency Division Multiple Access (FDMA). Tuttavia questi metodi non sono ottimali dal punto di vista delle prestazioni, poiché conducono ad un dispendio di risorse se il nodo non ha bisogno di trasmettere. Nei metodi di accesso di tipo *probabilistico*, invece, nessuna risorsa è assegnata in modo predefinito ma queste sono fornite utilizzando un algoritmo di tipo aleatorio: questo conduce ad un uso migliore del mezzo a disposizione, concedendolo solo a chi ne ha un reale bisogno. L'algoritmo Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) è un esempio di questo tipo di accesso, largamente utilizzato nell'ambito delle reti wireless.

Beaconless CSMA-CA In questo algoritmo, un nodo che deve trasmettere deve osservare il canale: se il canale è libero, allora la trasmissione può avvenire, altrimenti il nodo attenderà un tempo aleatorio prima di ritentare la trasmissione con un nuovo ascolto. Questo semplice metodo di accesso al canale, denominato *beaconless* può essere riassunto nell'algoritmo che segue:

1. attendo un tempo aleatorio $\tau \in \mathbb{U}[0, 2^{\mathrm{BE}} - 1]$, in cui BE è una variabile, denominata *esponente di backoff*;

2. ascolto il canale — ad esempio con il servizio Clear Channel Assessment (CCA) offerto dal livello PHY;

3. se il canale è occupato e, allo stesso tempo, il numero di tentativi fatti è inferiore ad un valore predefinito memorizzato con il nome di `macMaxCSMABackoffs`, allora scelgo un nuovo valore per l'esponente di backoff, dalla relazione $\mathrm{BE} = \min(\mathrm{BE} + 1, \mathrm{macMaxBE})$, e riprendo dal punto 1. Il valore di `macMaxBE`, definito a priori, è il valore massimo dell'esponente di backoff;

4. se il canale è libero, trasmetto.

Beacon CSMA-CA Il protocollo prevede una seconda modalità d'uso dell'algoritmo, basato su una più rigida ripartizione temporale gestita dal coordinatore della rete PAN. Questa struttura è utilizzata nella modalità *beacon* in cui il coordinatore usa dei particolari pacchetti di sincronismo (denominati, appunto, beacon) con lo scopo di descrivere una struttura temporale di contenimento, detta superframe. Questa può essere idealmente suddivisa in due parti: la prima, detta "parte attiva", è la sezione in cui avvengono le trasmissioni mentre la seconda, detta "parte inattiva" è riservata per l'inattività dei nodi.

A sua volta la prima parte è suddivisa in 16 slot temporali, raggruppati in Contention Access Period (CAP) — 9 slot — e in Contention Free Period (CFP) – 7 slots.

Nel periodo di contenimento (CAP) viene utilizzato l'algoritmo CSMA-CA in una sua versione leggermente modificata mentre nel periodo ad accesso garantito (CFP) la possibilità di trasmettere è riservata in modo esclusivo ad alcuni nodi, che utilizzano quindi un metodo di accesso a divisione temporale e non l'algoritmo CSMA-CA.

1.4.2 MAC frame

Il protocollo IEEE 802.15.4 prevede che la lunghezza massima di ogni pacchetto (a livello MAC) non debba superare i 127 byte. Come è possibile osservare operando un confronto con altri standard wirelss, questo valore è decisamente basso, il che implica la necessità di limitare la lunghezza delle intestazioni, per garantire comunque una buona quantità di dati trasferiti. Sono previsti quattro tipologie di frame:

Beacon frame utilizzati dal coordinatore nella modalità *beacon* per la distribuzione di informazioni relative alle temporizzazioni;

Data frame utilizzati per il trasferimento dati;

Acknowledgement frame utilizzati come frame di conferma di avvenuta ricezione;

Command frame utilizzati dal coordinatore per inviare comandi di controllo e di impostazione ai nodi della rete.

Tutti questi tipi di pacchetti hanno la medesima struttura. La prima parte del pacchetto è denominata Frame Control Field (FCF) e porta diverse informazioni riguardo al pacchetto, come ad esempio, il tipo, la modalità di indirizzamento, le impostazioni per la sicurezza, etc. Gli 8 bit che seguono contengono un valore numerico progressivo, che prende il nome di Sequence Number, che identifica in modo univoco il pacchetto inviato. Seguono quattro campi che dichiarano gli indirizzi del nodo sorgente, del nodo destinatario e gli indirizzi delle rispettive PAN di appartenenza. I campi di indirizzamento possono omessi nel caso in cui si stia trattando un frame di tipo Acknowledgement e la loro lunghezza può essere variata (16 o 64 bit) secondo le situazioni.

Dopo queste sezioni, che costituiscono l'*header* per il pacchetto MAC, segue la sezione contenente il *payload*, ovvero il contenuto informativo proveniente dal livello network mentre, in coda, è presente un ultimo campo che prende il nome di Frame Check Sequence (FCS). Quest'ultimo è un campo di controllo per la ridondanza ciclica che permette di stabilire se la trasmissione ha introdotto errori in qualche porzione del pacchetto ricevuto. Per spiegare in maniera più dettagliata la composizione dei frame, si può osservare la Figura 1.6 che riporta lo schema generale di ogni frame, mentre la

Bytes: 2	1	0 / 2	0 / 2/ 8	0 / 2	0 / 2/ 8	0 / 5/ 6/ 10/ 14	variabile	2
Frame Control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Campi ausiliari	Frame Payload	FCS
		Campi di indirizzamento						
MHR							MAC payload	MFR

Figura 1.6: Struttura di base di un MAC frame per il protocollo 802.15.4.

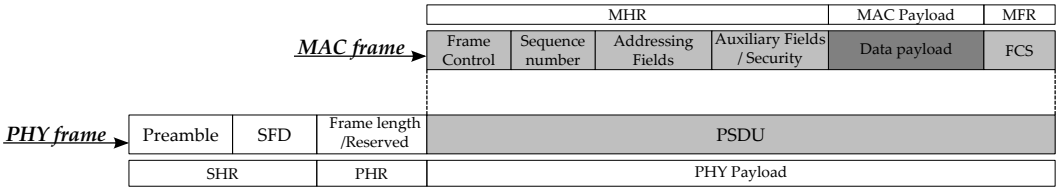


Figura 1.7: Esempio di come avviene l'incapsulamento del MAC frame all'interno del PPDU.

Figura 1.7 illustra una schematizzazione del processo di incapsulamento del MAC frame all'interno del pacchetto di trasmissione.

Capitolo 2

Il simulatore

Introduzione

Come indicato nell'introduzione, questa tesi è stata sviluppata utilizzando OMNeT++, un ambiente di simulazione ad eventi discreti, basato su una struttura modulare orientata agli oggetti. É importante sottolineare che la modularità di realizzazione fornisce un elevata flessibilità a questo software che può essere impiegato negli ambiti più diversi come nella modellizzazione di sistemi multiprocessore, di sistemi ad hardware distribuito o, più frequentemente, nello studio di reti di comunicazione a pacchetti.

É realizzato in linguaggio C++, combinato con il Tcl/Tk per l'interfaccia grafica: al suo interno il software utilizza il linguaggio C++ per definire il modello comportamentale dei moduli ed in linguaggio interno NED per dichiarare le strutture di rete, i suoi aspetti topologici e per definire le struttura dei messaggi.

OMNeT++ integra al suo interno:

- le librerie del kernel di simulazione;
- un compilatore per la descrizione topologica NED;
- una IDE basata sulla piattaforma Eclipse;
- una interfaccia grafica per l'esecuzione delle simulazione (TkEnv);
- un'interfaccia utente per eseguire le simulazione da riga di comando (CmdEnv);
- varie utilities, come il tool per la creazione di makefile;
- una nutrita documentazione, con una vasta gamma di esempi e tutorial.

Interfacce Le simulazioni possono essere seguite e controllate tramite un'interfaccia grafica, denominata *TkEnv*, che rappresenta le varie entità definite sotto forma di icone e i collegamenti dichiarati con delle linee, segnalando lo spostamento di pacchetti con

l'uso di animazioni. Questa interfaccia consente inoltre di osservare lo stato di alcune variabili interne per controllarne l'evoluzione temporale, utile soprattutto in fase di debug. All'occorrenza è sempre possibile sopprimere questa rappresentazione ed eseguire velocemente lunghe simulazioni da riga di comando.

Una seconda interfaccia grafica è GNED, che consente di creare i file di descrizione topologica (Network Description Language (NED)) in modo visuale, inserendo simboli che rappresentano i vari moduli e realizzando le *connessioni* collegando i simboli stessi sul piano di lavoro: in questo modo è possibile produrre codice più velocemente pur limitando il livello di definizione utilizzato.

2.1 Struttura a moduli

Il funzionamento dell'ambiente di simulazione si basa sull'uso di unità fondamentali, che prendono il nome di moduli semplici (*simple modules*). A ciascun modulo è sempre assegnata una coppia di file che ne delineano la una descrizione comportamentale, scritta in linguaggio C++, e la una descrizione topologica, basata sul linguaggio NED. I moduli singoli possono essere raggruppati all'interno di moduli di dimensioni maggiori, denominati moduli composti (*compound modules*), che li contengono e li collegano, fornendo un livello di astrazione superiore. La descrizione topologica definisce la struttura di rete del modulo specificando il numero di porte (*gates*) di cui dispone per la comunicazione, le connessioni esistenti, il tipo di sub-moduli contenuti (nel caso di moduli composti) e la descrizione comportamentale a cui il modulo corrisponde (nel caso di moduli semplici). Possono anche essere definiti dei parametri specifici del modulo, che possono essere utilizzati nel corso della simulazione: una rappresentazione delle relazioni tra i moduli e osservabile in Figura 2.1. La descrizione comportamentale, invece, fa uso delle funzioni di libreria e del linguaggio C++ per simulare il funzionamento del modulo: tale descrizione è fornita separatamente rispetto alla descrizione topologica, consentendo un maggior grado di riusabilità del codice. Alla descrizione comportamentale è usualmente affiancato un header file che riassume i metodi implementati dal modulo e consente di utilizzare l'ereditarietà del codice, rendendo il codice ancor più flessibile e facile da modificare.
In particolare osserviamo che ogni modulo è definito come estensione del simple module, e quindi deve essere sempre definita una funzione di inizializzazione (denominata `initialize()`) e la funzione di gestione di messaggi, chiamata `handleMessage()`.

2.2 Il punto di partenza

La maggior parte dell'ambiente di simulazione si basa sul lavoro prodotto da l'ing. Tramarin nel corso della sua tesi magistrale [1]: partendo da un ambiente che supportava la

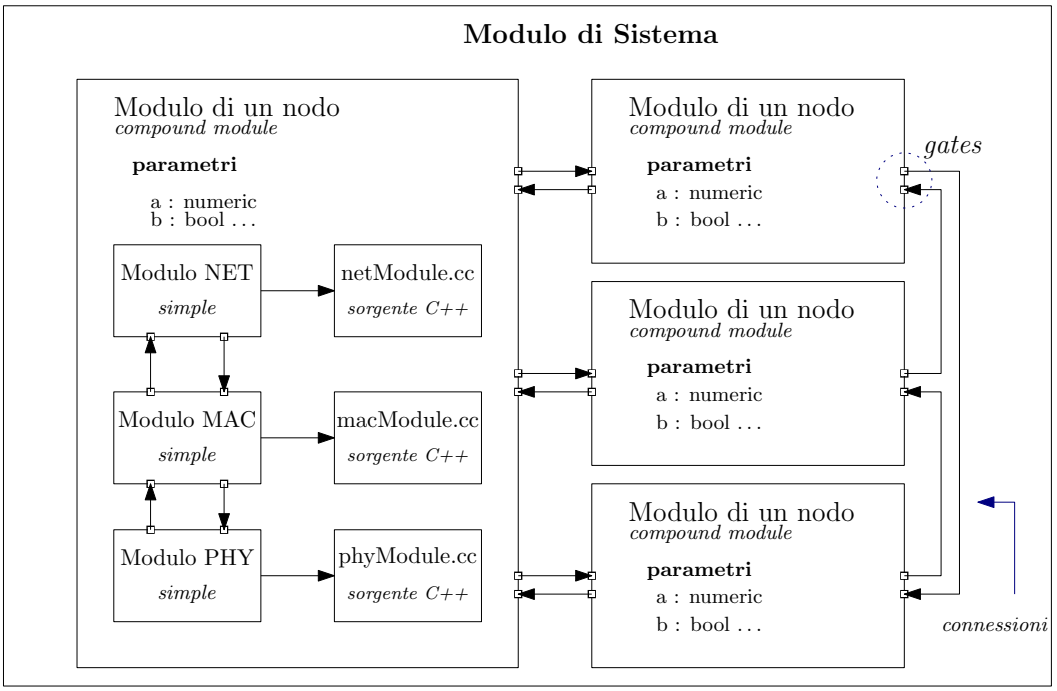


Figura 2.1: Schema delle relazioni tra i vari moduli della rete dell'ambiente di simulazione

simulazione di reti wireless mobili, denominato *Mobility Framework*, ha implementato la possibilità valutare in modo efficiente le problematiche collegate alla coesistenza di due reti wireless che utilizzano entrambe la banda ISM. L'ambiente di partenza di questa tesi contiene già un'implementazione base del protocollo 802.15.4 e del protocollo 802.11, che si concretizza in un insieme di numerosi file sorgenti, file di descrizione topologica e file di configurazione.

2.2.1 Mobility Framework

Il Mobility Framework è un'estensione di OMNeT++ che permette la simulazione di reti wireless mobili. Il codice è suddiviso in cartelle che permettono di riconoscere la sezione a cui ciascun modulo appartiene. La parte centrale dell'implementazione supporta la mobilità dei nodi, la gestione delle connessioni dinamiche e un modello per la comunicazione wireless; è inoltre presente una cartella dedicata alle utilità, porzioni di codice C++ che realizzano funzioni di interesse generale come, ad esempio, calcoli matematici. Il modulo *Channel Control* ha il compito di mantenere connessi tutti i dispositivi della rete che possono tra loro creare interferenza; resta inteso che due nodi connessi non sempre possono effettuare uno scambio di dati e comunicare tra di loro. Questo *framework* dispone inoltre di una serie di moduli di base che possono essere utilizzati per derivare le implementazioni dei propri nodi. Con questo concetto un programmatore può facilmente sviluppare la propria implementazione protocollare, senza doversi

preoccupare delle interfacce necessarie e dello studio dell'interoperabilità. Il modulo *Blackboard* consente lo scambio di informazioni sulla variazione dello stato interno dei moduli che afferiscono ad uno stesso protocollo; un modulo le cui informazioni devono essere monitorate, *pubblica* ogni cambio di stato sulla Blackboard, che gli assegna una categoria. A sua volta, il nodo che necessita di monitorare cambiamenti di stato di altre unità, si *iscrive* alla Blackboard indicando a quali categorie di informazioni è interessato. Ogni volta che si verifica un cambio di stato in uno dei moduli registrati, è cura della Blackboard verificare quali moduli sono iscritti a tale evento ed informarli del cambio di stato. In particolare l'uso della Blackboard consente lo scambio di informazioni tra i layer evitando di dover utilizzare puntatori ai moduli.

La sezione *contrib* include i moduli che forniscono funzioni più specifiche rispetto a quelle degli altri moduli presentati. Tutti i moduli che rappresentano i layer dello stack protocollare sono raccolti in cartelle: possiamo trovare la cartella `netwLayer`, la cartella `applLayer` e la cartella `nic`, che unisce al suo interno i layer MAC e PHY. Oltre a questi fondamentali moduli, ci sono cartelle dedicate ai messaggi e alla gestione della mobilità. La cartella *messages* contiene tutti i messaggi scambiati dai moduli indicati nella sezione `contrib`. La struttura dei messaggi è specificata attraverso il *linguaggio di definizione dei messaggi* e sono tutti sottoclassi della definizione base di messaggio, presente nella sezione principale.

L'altra fondamentale caratteristica offerta è la mobilità dei nodi, da cui il nome del framework. Ogni nodo può muoversi all'interno di un ambiente in diversi modi ed è necessario determinare sia dove collocare le informazioni relative al moto, sia come gestire le connessioni dinamiche. Le informazioni di movimento sono processate dal modulo stesso, mentre per quanto riguarda le connessioni, il modulo `ChannelControl` descritto precedentemente ha proprio il compito di mantenere aggiornate queste informazioni. In particolare il controllore di canale deve verificare se la distanza tra due nodi è tale da permettere una trasmissione di segnale sopra la soglia di sensitività richiesta oppure se il segnale ricevuto può essere confuso con il rumore; questo si traduce in una distanza-soglia sopra la quale le connessioni devo essere eliminate. É importante sottolineare che, quando un nodo è connesso agli altri non è detto che si possa stabilire una reale comunicazione: questo è deciso dal nodo stesso, a livello di PHY layer, in funzione del rapporto segnale rumore rilevato. Il framework dispone di diversi algoritmi che implementano cammini su cui i dispositivi possono muoversi: il più semplice è quello a velocità costante, che nelle simulazioni realizzate, è stato utilizzato con velocità nulla per ottenere il comportamento statico desiderato.

Ogni modulo è quindi composto da più moduli che rappresentano i layer e, oltre a questi, è presente un modulo per la mobilità (*mobility*) e un modulo blackboard, come illustrato in Figura 2.2.

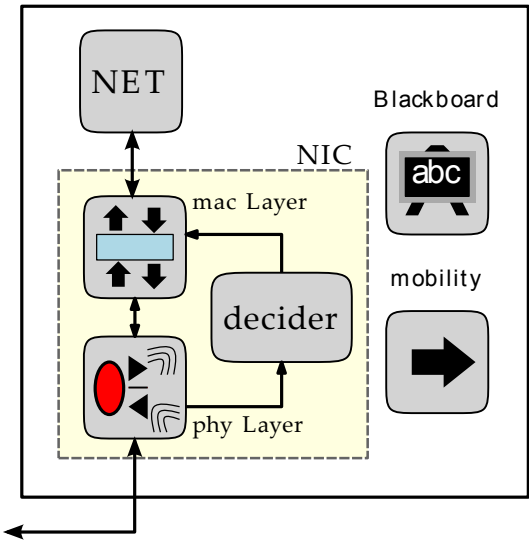


Figura 2.2: Struttura interna ad ogni nodo della rete, in cui sono evidenziati i moduli corrispondenti ad ogni layer.

2.3 Da modulo a layer

Dall'espansione del modulo base è stato possibile derivare un nuovo modulo che rappresenta la cellula fondamentale per la creazione dei layer dello stack ISO/OSI, chiamato *BasicLayer*. Questa classe non fa che fornire una base per l'implementazione di tutte le classi che definiscono i livelli che compongono i nodi. Il BasicLayer prevede un totale di sei porte di comunicazione con cui saranno formati altrettanti collegamenti unidirezionali da e per questo modulo. Di conseguenza vengono definite due porte dirette verso il layer gerarchicamente superiore, denominate *upperGateIn* e *upperGateOut*, due per la comunicazione verso il livello inferiore, dette *lowerGateIn* e *lowerGateOut*, e infine due porte per la trasmissione di informazioni di controllo una dal layer inferiore — *LowerControlIn* — ed un verso il layer superiore — *upperControlOut*.

Il metodo predefinito per la gestione dei messaggi in arrivo, presentato nella Sezione 2.1, viene ora diversificato a seconda della porta coinvolta, portando alla realizzazione di funzioni specifiche per ogni porta (es. `handleLowerControl`). L'unica alternativa a queste opzioni è che il messaggio non provenga da una porta ma sia un *selfMessage* che sarà gestito dalla funzione apposita per le temporizzazioni *handleSelfMessage()*. In particolare i *self-message* sono particolari tipi di messaggi inviati da un modulo a se stesso, che arrivano dopo un determinato intervallo di tempo, consentono di gestire le temporizzazioni necessarie.

Una specificazione del modulo `BasicLayer` può essere quella offerta dal modulo *BasicMacLayer* utilizzata appunto per implementare tutti i layer MAC. La prima differenza è la definizione di un nuovo parametro al momento dell'inizializzazione del

nodo, l'indirizzo MAC che viene assegnato in modo univoco ad ogni macLayer della rete. Una seconda differenza è la presenza di una funzione di incapsulamento, denominata `encapsMsg()`, incaricata di raccogliere i messaggi provenienti dal livello superiore per aggiungervi un proprio header, secondo la specificazione del protocollo usato. In realtà possono esservi delle differenze dovute al fatto che l'ambiente in cui si opera è simulato e alcune informazioni, come la lunghezza del pacchetto, possono essere già presenti altrove. Per dualità è fornita una funzione simmetrica, denominata `decapsMsg()` che estrae il contenuto informativo dai pacchetti provenienti dal livello inferiore prima di inviare l'informazione utile al livello superiore.

2.4 Moduli della rete 802.15.4

I moduli della rete 802.15.4 hanno la struttura osservata in Figura 2.2; qui è possibile identificare un insieme di 3 moduli, che costituiscono il Network Interface Controller (NIC), il modulo che implementa il layer NET e una coppia di moduli accessori — Blackboard e Mobility — che non fanno riferimento a reali componenti di rete ma sono utilizzati in fase di simulazione. Questa rappresentazione semplificata è in perfetto accordo con quanto detto inizialmente sulle caratteristiche dei nodi sensori, che devono essere economici e di facile realizzazione. Come prima cosa saranno descritti i moduli che compongono il NIC, ovvero il `PhyLayer`, il modulo `decider` e il `MacLayer` e solo successivamente sarà presentata la descrizione del layer NET.

Il PHY Layer Il modulo *phyLayer* è il legame tra il nodo e il mezzo trasmissivo quindi, tra i suoi parametri possiamo trovarne molti che servono appunto per caratterizzare il canale. All'interno di questo modulo sono definite un gran numero di primitive, a causa del gran numero di servizi offerti dal layer, tra cui i più importanti sono la primitiva CCA e la primitiva ED descritte nel Capitolo 1. In questa implementazione è stata definita solo la prima modalità di CCA, che fornisce una dichiarazione di canale libero valutando l'energia ricevuta. Nel momento in cui la funzione `cca()` viene chiamata, questa controlla se la trasmittente è in condizione di riposo e, in caso affermativo, esegue una comparazione tra il livello di energia ricevuta a la soglia memorizzata, ritornando lo stato del canale. Anche la funzione `ed()` presenta un funzionamento del tutto simile a quello della funzione `cca()`, con la differenza che in questo secondo caso non vengono eseguite comparazioni ma viene direttamente restituito il valore rilevato. Dal punto di vista implementativo il funzionamento di questi metodi coinvolge le classi che stanno sotto al modulo considerato (dal punto di vista dell'ereditarietà), in particolare la classe `SnrEval_802154`. Questa dispone di due funzioni per la gestione dei messaggi — `handleLowerMsgStart()` e `handleLowerMsgEnd()` — che vengono attivate rispettivamente quando un pacchetto inizia ad essere ricevuto e quando la sua ricezione termina. La funzione di inizio ricezione

del frame viene attivata all'arrivo di un messaggio ma prima del suo completamento; la prima operazione eseguita è un controllo sulla validità del pacchetto — eseguendo, ad esempio, dei controlli sul canale di provenienza. Subito dopo, la potenza ricevuta è calcolata e memorizzata in un buffer per le successive elaborazioni. Poi la funzione valuta se è avvenuta una collisione con altri pacchetti provenienti da nodi della stessa rete e, in caso affermativo, viene virtualmente aumentato il livello di rumore del canale di una quantità pari alla potenza ricevuta. Questo cambiamento è pubblicato via Blackboard per permettere agli altri nodi di agire di conseguenza. Se, invece, non ci sono collisioni, la potenza ricevuta viene comparata con il valore di sensibilità e, se non ne supera il valore, il messaggio viene considerato un frame di rumore mentre in caso contrario è considerato valido; in entrambi i casi il livello di potenza ricevuta viene aggiornato. Tuttavia, questo metodo non può rifiutare di riceve i messaggi poiché è invocato all'inizio della ricezione, quando il messaggio è ancora nel canale. La funzione `handleLowerMsgEnd()` viene quindi chiamata, con l'uso di un self-message, dopo che il tempo necessario alla ricezione del messaggio è trascorso. Questa va ad agire sul messaggio memorizzato dalla prima funzione, decidendo se è stato ricevuto correttamente, se è un frame di rumore o se deve essere scartato a causa dell'interferenza. Per raggiungere lo scopo, viene contattato il modulo di *interference*, richiedendo se il pacchetto ha subìto collisioni con i pacchetti della rete IEEE 802.11. La funzione utilizzata è denominata `collisionOccurred()` e risponde utilizzando le informazioni note al modulo di interferenza sui tempi di invio di tutti i pacchetti sul canale. Grazie a queste informazioni e a quelle precedentemente memorizzate, è possibile determinare se il messaggio è corrotto o se è ancora accettabile e, in secondo caso, il pacchetto viene inviato al modulo di decisione — *decider*.

Il modulo Decider Il comportamento del modulo di decisione è un'estensione della classe `BasicSnrDecider` e ne eredita quindi la caratteristica fondamentale che è quella di poter essere raggiunto da un messaggio solo se quest'ultimo proviene dal livello inferiore mentre tutti quelli provenienti dal livello superiore (in questo caso, il MAC layer) sono automaticamente re-indirizzati al livello sottostante, cioè al PHY layer. Lo scopo del *decider* è, appunto, decidere se i messaggi provenienti dal livello inferiore possono essere passati al livello superiore o se devono essere distrutti. La struttura del PHY layer così come descritta nello standard, si presenta quindi suddivisa in due pozioni distinte, privilegiando una separazione funzionale dei moduli. Il modulo del paragrafo precedente, gestisce i servizi previsti per il layer e controlla se il pacchetto può essere ricevuto da un punto di vista "formale", mentre questo modulo effettua un secondo passaggio decisionale basato, questa volta, su considerazioni legate al rapporto segnale rumore.

Il valore numerico del rapporto segnale rumore è calcolato dalla funzione `addNewSnr()` nel modulo di livello PHY. Ogni volta che, durante la ricezione, il livello di rumore al ricevitore cambia, questa funzione aggiunge il valore numerico alla lista degli SNR e passa

questa lista al modulo di decisione, assieme al messaggio considerato. All'arrivo del messaggio, avvertito dalla funzione `handleLowerMsg()`, la lista con i valori degli SNR rilevati viene elaborata, confrontando il valore minimo di quella lista con un valore predefinito, denominato *snrThreshold*: se l'SNR minimo è inferiore alla soglia, il messaggio viene distrutto mentre, in caso contrario, viene inviato al MAC layer.

Il modulo MAC A livello MAC è necessario ricordare che il protocollo IEEE 802.15.4 specifica due tipi di dispositivi, illustrati nella Sezione 1.2, ovvero i dispositivi a piene funzionalità (FFD) e i dispositivi a funzionalità ridotte (RDF), a cui corrispondono rispettivamente le due descrizioni comportamentali `MacFfd802154` e `MacRfd802154`. L'implementazione dei moduli RFD, presente nel Mobiliy Framework, supporta tutte le primitive richieste per il completamento di questo lavoro, mentre sono mancanti alcune definizioni relative a funzionalità aggiuntive come la sicurezza e la disassociazione da una rete PAN.

La principale funzione da osservare è quella che si occupa dell'algoritmo CSMA-CA. Ogni messaggio proveniente dal livello network viene gestito tramite la chiamata della funzione `handleUpperMsg()` che realizza una serie di controlli verificando, ad esempio, che il frame non sia vuoto, che sia specificato il destinatario, etc. Se tutti i controlli vengono superati, il messaggio ricevuto viene incapsulato in un'unità che prende il nome di MPDU; con il termine "incapsulazione" in OMNeT++ si intende l'inserimento del messaggio in un altro contenitore che raccoglie il messaggio più ulteriori informazioni che simulano la lettura e la scrittura degli header. La stessa funzione `handleUpperMsg()` termina con la chiamata alla funzione `nextMacAction()`, una funzione ausiliaria nata con lo scopo di semplificare il codice. Viene sostanzialmente realizzata una macchina a stati finiti che tiene traccia del flusso di esecuzione interno evitando di dover controllare ad ogni passaggio lo stato del modulo; per far avanzare lo stato di esecuzione è sufficiente chiamare il metodo `nextMacAction()`. Il codice che realizza questa funzione è molto lungo e presenta una gran numero di casi da gestire. Per esempio, dopo l'esecuzione della funzione `handleUpperMsg()`, questo metodo controlla che il buffer di trasmissione non sia pieno ed eventualmente sceglie quale algoritmico del CSMA-CA utilizzare. Anche se questa implementazione prevede l'esistenza di entrambi gli algoritmi di CSMA-CA descritti nella Sezione 1.4.1, solo quello unsoltted è stato utilizzato e sarà in seguito descritto. Innanzitutto, la funzione `unslottedCsma()` inizializza le variabili necessarie impostando il numero di backoff (NB) a zero e l'esponente di backoff (BE) al minimo valore concesso dallo standard, provvedendo all'invio di un self-message che modella il tempo di attesa casuale necessario. Allo scadere di questo timer viene richiamata la funzione `cca()` che invia al livello fisico un frame di controllo con la richiesta di utilizzare il servizio CCA. La funzione `handleLowerControl()` si occupa di raccogliere la risposta ricevuta, chiamando la funzione `ccaConfirm()` con argomento pari allo stato rilevato.

Se il canale è occupato, il flusso di esecuzione riprende dalla funzione `unslottedCsma()` che, dopo aver incrementato opportunamente le variabili NB e BE, verifica se può rieseguire un attesa di backoff oppure se il numero di backoff disponibili è finito. Se il canale è libero, il flusso di esecuzione ritorna alla funzione `nextMacAction()` che chiama a sua volta la funzione di invio dei messaggi `sendDownFrame()`. Questa, si occupa di passare al livello fisico il messaggio più vecchio presente nel buffer di trasmissione, delegandogli lo scopo di effettuare la trasmissione.

La ricezione di un messaggio proveniente dal livello inferiore attiva la funzione `handleLowerMsg()` che, dopo aver controllato il tipo di frame ricevuto, richiama la funzione più adatta alla sua gestione — ad esempio, alla ricezione di un frame di comando, la funzione chiamata sarà `handleCommandFrame()`. Nel caso in cui il frame ricevuto sia un frame di dati e si rilevi la richiesta di una conferma di avvenuta ricezione, è compito della funzione `handleDataFrame()` occuparsi della creazione e dell'invio degli Acknowledgement (ACK) necessari. Successivamente sarà la stessa funzione a passare il messaggio al livello network.

Il modulo NET Come osservabile nella Figura 2.2, il più alto livello dello stack OSI/ISO implementato nel nodo è il livello network che agisce da generatore di traffico e da gestore dei livelli inferiori. Lo scopo di una WSN è formare una rete in cui tutti i nodi RFD siano connessi con almeno un nodo FFD che agisce da coordinatore e costituisce una PAN, secondo le topologie previste. Per soddisfare questi requisiti, sono state realizzate due tipologie di MAC, una per i dispositivi RFD e una per i dispositivi FFD. Sono quindi necessari due tipi di NET layer: uno per i dispositivi preposti alla generazione di pacchetti, chiamato `Send_802154`, ed uno per i dispositivi preposti alla ricezioni, chiamato invece `Receive_802154`.

In particolare la classe `Receive_802154` tiene traccia del numero di nodi ricevuti e del numero di messaggi duplicati. É possibile specificare un gran numero di parametri per questo layer, decidendo il comportamento del nodo. Nella fase di inizializzazione, il modulo legge i parametri definiti e può inviare al livello MAC il comando di sincronizzarsi alla rete oppure di iniziare la creazione di una nuova rete, a seconda dei valori letti. Sono quindi necessarie solo due funzioni in questo modulo: la funzione `handleLowerMsg()` e la funzione `handleLowerControl()`. La prima è chiamata quando si rileva l'arrivo di un nuovo pacchetto e si limita a operare una serie di controlli oltre che l'aggiornamento di alcune variabili. La seconda invece ha il solo scopo di verificare le informazioni di controllo che il MAC invia, come ad esempio, avvisi sull'impossibilità di sincronizzarsi alla rete.

La classe `Send_802154` è un'estensione della precedente perché, oltre a fornire la possibilità di riceve pacchetti, prevede anche la possibilità di inviarli. Vengono quindi aggiunti altri parametri all'implementazione, oltre che nuove funzioni. La funzione

`handleSelfMessage()`, utilizzata per la gestione delle temporizzazioni, consente di impostare l'invio periodico di pacchetti ad una distanza temporale prefissata o aleatoria, tramite la variabile `interArrivalTime`. La variabile `packetCount`, invece, imposta il massimo numero di pacchetti da inviare, in modo da fermare il processo al raggiungimento della soglia.

2.4.1 Gli host

L'insieme delle classi descritte individuano vari tipi di dispositivi, che vengono chiamati *host*. Quelli in grado solamente di ricevere pacchetti dal canale sono *host di ricezione* e vengono identificati nelle simulazioni con il nome `RecvHost802154`; a livello MAC utilizzano l'implementazione di nodi RFD e a livello NET viene utilizzata la classe `Receive_802154`. Un secondo tipo di dispositivi è composto dagli *host generatori di traffico*, che nelle simulazioni sono associati al nome `SendHost802154`; questi utilizzano lo stesso MAC layer dei precedenti (RFD) ma un livello NET basato sulla classe `Send_802154`. É anche possibile trovare nodi responsabili di coordinare la rete che, come già menzionato, devono essere sempre presenti in una rete PAN; questi *host coordinatori* prendono il nome `CoordHost802154` in fase di simulazione ed hanno necessariamente un'implementazione MAC di tipo FFD mentre per il livello NET sono utilizzabili entrambe le alternative, secondo gli usi.

2.5 Il modulo *Interference*

La possibilità di valutare l'azione congiunta delle due reti è dovuta alla presenza del modulo che prende il nome di *Interference*, realizzato dall'ing. Tramarin nel corso della sua tesi [1]. Il simulatore non prevede l'analisi congiunta di più reti e non è in grado di valutare i fenomeni dovuti alla coesistenza, proprio poiché i messaggi vengono scambiati tra nodi di reti che non comunicano tra loro in alcun modo. Il punto di contatto tra le due reti è appunto il modulo di interferenza: utilizzando le informazioni raccolte nella fase di setup della rete, all'interno del modulo viene registrato un elenco tutti i NIC messaggi inviati nel mezzo: nel caso in cui rilevi la presenza simultanea di due o più trasmissioni da parte di reti diverse, questo modulo rende conto dell'effetto di interferenze reciproca distruggendo quei messaggi che, a causa dell'interferenza, risultano corrotti e inutilizzabili. Dal punto di vista della simulazione, quando un nodo riceve un messaggio, il suo strato fisico chiede al modulo di interferenza se tale messaggio può essere considerato valido o meno; il modulo di interferenza, utilizzando le informazioni sulla potenza del frame e valutando la presenza di altre trasmissioni nello stesso istante, decide se il messaggio deve essere scartato o meno, comunicandolo al PhyLayer. Il ruolo del modulo di interferenza può essere schematizzato come in Figura 2.3.

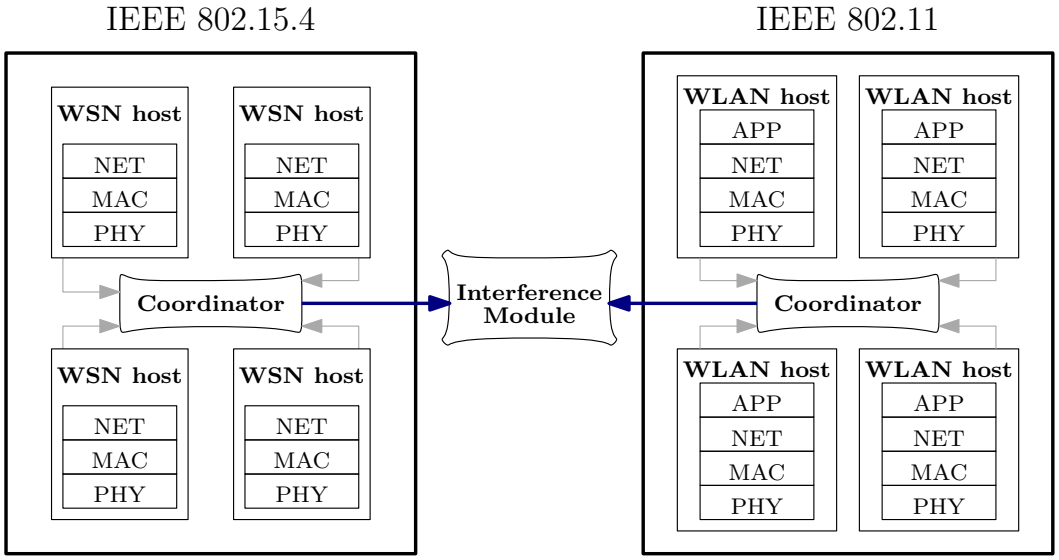


Figura 2.3: Immagine che mostra il ruolo del modulo di interferenza.

Capitolo 3

Modifiche protocollari

Introduzione

Nei sistemi industriali, è sempre più diffusa la necessità di realizzare reti di sensori per monitorare le varie fasi di un processo produttivo o controllare alcuni parametri di interesse con una attenzione sempre maggiore all'affidabilità di queste comunicazioni.

Il documento [2] pone l'attenzione sul problema dell'interferenza dovuta alla coesistenza delle reti di sensori con reti Wireless Local Area Network (WLAN), con particolare riguardo alle WSN che utilizzano il protocollo IEEE 802.15.4 nella banda ISM da 2.4 GHz. Nel documento si indaga quanto e come la tecnica di ritrasmissione utilizzata, possa migliorare le prestazioni di sistema, passando in rassegna varie strategie. Tutte queste considerazioni vengono indagate nelle reti con topologia denominata *polling* e costituiscono un insieme di modifiche al protocollo.

3.1 Il polling

Lo schema che usualmente viene utilizzato per realizzare reti di sensori a scansione progressiva è denominato polling; questo consiste nella visita ciclica di tutti i nodi, eseguita da parte di un'entità di rete gerarchicamente superiore che ha il compito di contattare e interrogare le varie stazioni. In generale si indica con il nome di *master* il nodo incaricato di gestire la comunicazione nel suo complesso, e con il nome di *slave* ciascuno dei dispositivi che vengono interrogati, fornendo informazioni: uno schema strutturale della rete è osservabile in Figura 3.1. L'interrogazione su polling si differenzia in modo sostanziale dalla comunicazione spontanea poiché la facoltà di iniziare una trasmissione dati è delegata completamente al nodo master mentre i nodi slave hanno la possibilità di trasferire informazioni solo nel momento in cui avvertono una richiesta da parte del master.

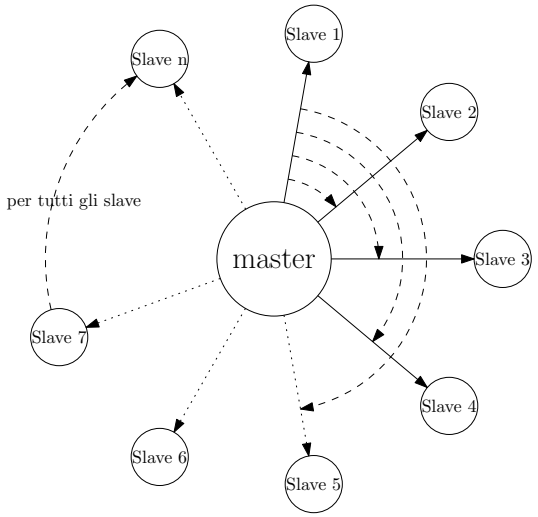


Figura 3.1: Esempio di topologia di polling: il nodo master occupa la posizione centrale della rete e scansiona tutti gli slave.

Questo modello è largamente impiegato, specie a livello industriale, poiché introduce una serie di vantaggi tipici del controllo di qualità. Notiamo che è presente una temporizzazione più rigida, se paragonata alla comunicazione spontanea: è, infatti, il master a decidere se e quando stabilire una comunicazione con uno slave, delimitando i tempi di accesso al mezzo e riducendo il rischio di collisioni. La forte temporizzazione colloca questa tecnica nell'ambito del controllo *realtime*, dove, con questo termine si intende la capacità di caratterizzare la durata temporale di un'azione entro dei limiti temporali predefiniti. Il controllo realtime consente, inoltre, di avere una visione più ampia di quello che sta succedendo permettendo la verifica costante dello stato di collegamento, specie nelle applicazioni di maggior criticità.

La scansione ciclica è determinata da ritmi temporali predefiniti: ciascun ciclo, denominato *Cycle Period*, ha una durata prefissata ed il suo inizio individua il momento in cui le informazioni all'interno del nodo master vengono re-inizializzate. In particolare, nelle tecniche illustrate in sezione 3.3, l'inizio del Cycle Period determina il caricamento degli indirizzi dei nodi slave all'interno del nodo master, in modo da poter ricominciare la scansione. Il periodo si presenta suddiviso in due porzioni temporali: la prima porzione, denominata *Periodic Window*, è la regione in cui la comunicazione viene gestita dal master, mentre la seconda porzione, denominata *Aperiodic/Idle Window* può essere utilizzata per lo scambio di messaggi che non seguono una precisa temporizzazione. In questo caso, l'unica sezione di interesse è la Periodic Window: la dimensione di questa finestra temporale deve essere scelta in modo che tutti i nodi possano essere interrogati almeno una volta, considerando anche un tempo per la gestione delle ritrasmissioni. Questa finestra può essere idealmente suddivisa in slot temporali entro i quali si svolge una sequenza di polling, cioè l'insieme di operazioni che comprendono la richiesta di

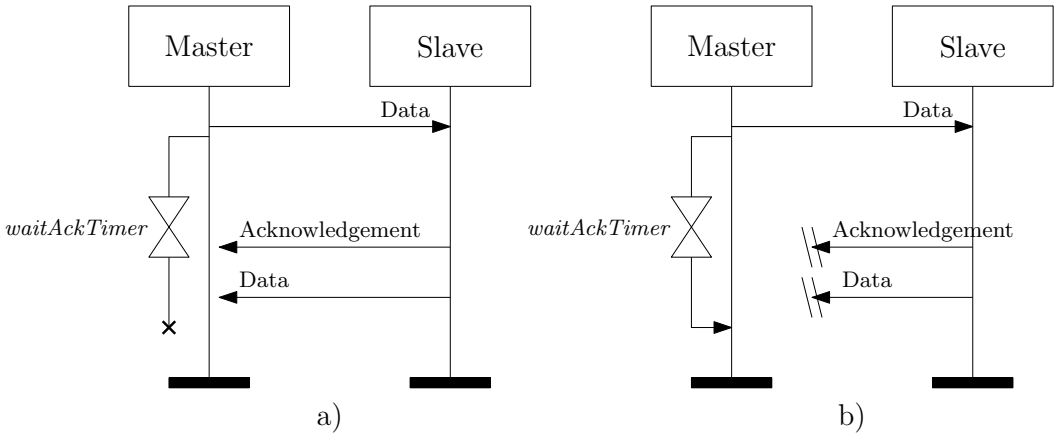


Figura 3.2: Schema temporale di una sequenze di polling. a) Polling eseguito con successo. b) Polling fallito a causa dell'interferenza

dati da parte dal master e il trasferimento del dato informativo.

La Figura 3.2 rappresenta uno schema della comunicazione tra master e slave: come si nota, il nodo master attende un tempo ben determinato prima di dichiarare fallita la trasmissione, tempo che può essere monitorato internamente tramite un timer apposito. Lo slave che riceve la richiesta genera un frame breve di tipo ACK per informare il master che la trasmissione è avvenuta con successo, seguito dal pacchetto informativo vero e proprio. A volte si utilizza inserire il frame ACK direttamente all'interno del frame di informazione: questa tecnica, denominata PiggyBack, consente di ridurre il numero di trasmissioni necessarie e di ottimizzare i tempi a scapito di una maggiore lunghezza del pacchetto. La trasmissione ha successo se si riescono a completare tutte le operazioni necessarie e il dato viene raccolto dal master; in alternativa, la trasmissione fallisce se, al termine del periodo di tempo prefissato, il master non riceve il dato che ha richiesto.

3.2 Polling e Interferenza

Come osservato nell'introduzione, nei sistemi wireless, più che in ogni altro sistema di comunicazione, il rischio di interferenza è un fattore certamente non trascurabile. I fenomeni che possono ostacolare il corretto funzionamento della rete possono essere dovuti:

- all'interazione interna, cioè quella le cui cause sono riconducibili al funzionamento della rete stessa. Ne sono un esempio le collisioni che si verificano quando più nodi della stessa rete desiderano accedere al mezzo nello stesso momento, oppure quando si verifica il fenomeno del *multiple path fading* - la distorsione introdotta dall'ambiente in cui avviene la trasmissione.

- all'interazione esterna, dovuta cioè a fenomeni di interferenza con altre fonti di onde elettromagnetiche presenti nell'ambiente di interesse.

Il risultato della presenza di collisioni è generalmente un degrado delle prestazioni complessive e dell'affidabilità della comunicazione. L'organizzazione di una rete secondo uno schema di polling ha, come osservato, il vantaggio di introdurre una forte temporizzazione: se da un lato questo può ridurre i fenomeni di interazione interna (organizzando la comunicazione) dall'altro la stessa temporizzazione rende il sistema più vulnerabile all'interferenza esterna. Può accadere, infatti, che nel momento in cui viene richiesto un dato ad un nodo sensore, questo, a causa dell'interferenza, non sia in grado di fornirlo prontamente ed è possibile che, nel momento in cui il dato arrivi a destinazione, abbia perso di utilità. Nel caso in cui il pacchetto vada perduto nel corso della trasmissione, si procede alla sua *ritrasmissione*: questa operazione fa parte dei metodi di controllo degli errori ed ha lo scopo di rendere affidabile la trasmissione a fronte di un canale non affidabile. In generale, quando il mittente invia un messaggio, aziona un timer e si mette in attesa di una risposta da parte del nodo ricevente: se allo scadere del tempo il nodo ricevente non ha fornito alcuna risposta, il pacchetto viene considerato perso e il mittente può provvedere alla sua ritrasmissione con modalità che variano a seconda della strategia adottata.

3.3 Tecniche di ritrasmissione

La ritrasmissione è l'insieme di operazioni che porta al rinvio di un pacchetto a seguito di una trasmissione senza successo. Le modalità con cui il nodo master, o più precisamente il suo MAC layer, decide di effettuare l'operazione prede il nome di *tecnica di ritrasmissione*: essa determina i tempi, la quantità e i criteri con cui il nodo master effettua le trasmissioni successive verso gli slave per richiedere le informazioni. Sono di seguito osservate principalmente quattro tecniche di ritrasmissione che saranno successivamente implementate su simulatore e confrontate tramite un'analisi delle prestazioni.

3.3.1 BIR

La prima tecnica osservata è denominata Boundled Immediate Retransmission, ed è comunemente impiegata al livello MAC del protocollo IEEE 802.15.4: in caso di insuccesso della sequenza di polling, il protocollo prevede l'immediato rinvio del pacchetto, ricominciando di fatto una nuova sequenza di polling. Ogni ritrasmissione compiuta provoca l'aumento di una variabile interna che si arresta al raggiungimento del valore definito dalla variabile *macMaxFrameRetries*: raggiunto questo valore il pacchetto viene considerato perso e lo slave non viene più contattato fino al prossimo *Cycle Period*.
Come alternativa a questa strategia, ne sono state proposte altre che si prefiggono lo

scopo di migliorare le caratteristiche della trasmissione tramite una gestione più attenta dell'ordine di scansione dei nodi.

3.3.2 UIR

La prima tecnica di ritrasmissione alternativa analizzata è la Unboundled Immediate Retransmission, che si presenta come una variante della precedente Boundled Immediate Retransmission (BIR). A seguito di una trasmissione inefficace da parte del nodo master, questo fa seguire una serie di ritrasmissioni che terminano solo quando si riesce a portare a compimento la comunicazione oppure alla fine del tempo concesso dal *Cycle Period*. Questa tecnica viene utilizzata per osservare soprattutto quanto possa influire la scelta del limite per le ritrasmissioni nella BIR sull'efficacia della strategia.

3.3.3 QR

Una seconda strategia è la Queued Retransmission (QR), che si presenta su una struttura completamente diversa rispetto a quella delle due tecniche precedenti. Essa, infatti, basa il suo funzionamento su una lista, memorizzata internamente al nodo master, che viene riempita all'inizio di ogni *Cycle Period* con tutti gli indirizzi MAC dei nodi della rete. Successivamente viene eseguita una scansione degli slave, seguendo l'ordine dettato dalla lista: se la trasmissione con uno slave avviene correttamente, il suo indirizzo è rimosso dalla lista mentre, se ciò non accade, l'indirizzo viene rimosso dalla lista e aggiunto in coda. Lo scorrimento della lista avviene continuamente, fino all'esaurimento degli indirizzi disponibili o fino alla fine del *Cycle Period*.
Questa tecnica, rispetto alla precedente, presenta il vantaggio di fornire sempre una scansione di tutti i nodi, evitando che qualche nodo possa non venir mai contattato; inoltre i tentativi di comunicazione con uno slave avvengono ad una distanza temporale maggiore rispetto al caso precedente: considerato il carattere temporale delle interferenze esterne, questo consente di ottenere una minor probabilità che due trasmissioni consecutive presso lo stesso nodo siano ugualmente fallimentari.

3.3.4 AQR

La terza strategia osservata è la Adaptive Queued Retransmission (AQR), che può essere vista come un'estensione del caso precedente. Anche questa tecnica, infatti, basa il suo funzionamento sullo scorrimento di una lista interna, con la differenza che in questo caso gli indirizzi dei nodi sono ordinati secondo un parametro statistico $\bar{s}_n$ che tiene conto della storia dello slave associato, a fronte di un numero di n tentativi. Ad ogni tentativo di comunicazione il master aggiorna il valore del parametro statistico, aumentandolo o diminuendolo a seconda del successo o dell'insuccesso trasmissione: il parametro è

quindi una media pesata esponenzialmente, descritto dalla relazione:

$$\bar{s}_{n+1} = \alpha \bar{s}_n + (1-\alpha) s_{n+1}$$

in cui s_{n+1} è una variabile indicatrice dell'ultimo risultato ottenuto che assume valore unitario nel caso in cui la comunicazione sia avvenuta con successo, altrimenti assume valore zero. Nella formula, appare il parametro α che, variando tra zero e uno, pesa l'espressione spostando il peso sulla storia o sull'ultimo risultato ottenuto: un valore pari a 0.9, come quello scelto per realizzare la simulazione, sposta il peso del parametro a favore della sua storia.

All'inizio di ogni *Cycle Period*, quindi, la lista verrà inizializzata ordinando gli indirizzi secondo il valore $\bar{s}_n$ e, come per la strategia QR, le lista verrà scorsa eliminando gli indirizzi dei nodi che hanno risposto al polling con successo e spostando in coda gli indirizzi dei nodi verso i quali la comunicazione è fallita. Analogamente alla tecnica precedente, la gestione temporale risulta più efficiente poiché viene sempre effettuato un tentativo di comunicazione verso tutti i nodi: in via aggiuntiva, l'ordinamento secondo il parametro statistico consente di contattare per primi quei nodi che hanno mostrato in passato di essere più affidabili, permettendo al sistema di stabilizzarsi e di ottenere sempre le prestazioni migliori.

In Figura 3.3 sono riportati gli schemi concettuali che rappresentano gli algoritmi per le tecniche di trasmissione illustrate nelle sezioni precedenti. Queste tecniche di ritrasmissione, costituiscono di fatto una modifica al protocollo analizzato: quello che si desidera ottenere è definire un riferimento protocollare che contempli più tecniche e che ne consenta la valutazione e la comparazione.

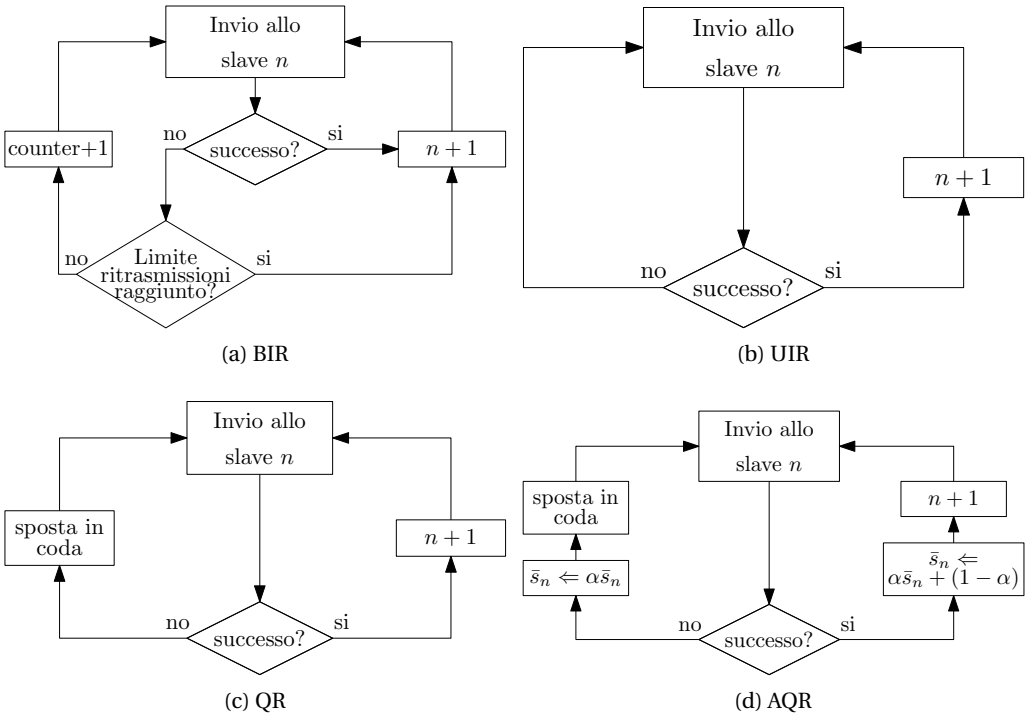


Figura 3.3: Schemi concettuali rappresentanti le operazioni compiute dal MAC layer per eseguire le varie ritrasmissioni.

Capitolo 4

Implementazione su Simulatore

Introduzione

In questo capitolo sarà discussa l'implementazione delle tecniche di ritrasmissione, illustrate nel capitolo precedente, all'interno dell'ambiente di simulazione OMNeT++. Da una prima analisi della struttura di rete che si vuole simulare, notiamo che è necessaria la presenza di almeno due tipi di nodi, rappresentanti il nodo master e i nodi slave, che saranno caratterizzati da comportamenti diversi. La struttura topologica dei due nodi è la stessa, ereditata da quella del nodo base della rete IEEE 802.15.4, illustrata nel Capitolo 2. Sarà di seguito fornita una descrizione complessiva del funzionamento della rete, in cui verrà indicato il ruolo svolto dai vari moduli di rete, per poi passare ad una descrizione dettagliata delle implementazioni dei nodi master e slave.

4.1 Descrizione complessiva

In questa sezione verrà fornita una descrizione complessiva del funzionamento della rete, focalizzando l'attenzione sulle implementazioni dei vari componenti solo nelle sezioni successive. Come richiesto dall'architettura adottata, la rete dispone di due tipi di dispositivi: un nodo della rete agisce da master, richiedendo informazioni, mentre altri nodi agiscono da slave soddisfacendo le richieste di dati con dei messaggi di risposta. Dal punto di vista implementativo, il master comunica agli slave le necessità di ricevere dei dati da essi, utilizzando dei frame di comando (*command frame*): questa scelta tuttavia non è formalmente corretta, poiché l'applicazione non rientra negli ambiti di utilizzo di questo tipo di frame. A questo proposito è da osservare che lo scopo della simulazione non è quello di ricalcare in tutto e per tutto l'oggetto di analisi, ma al contrario, un uso consapevole dello strumento e dell'ambiente di simulazione consente di ridurre la complessità della simulazione stessa con il vantaggio di raggiungere gli obiettivi in tempi più rapidi senza compromettere i risultati di simulazione. La scelta di utilizzare dei

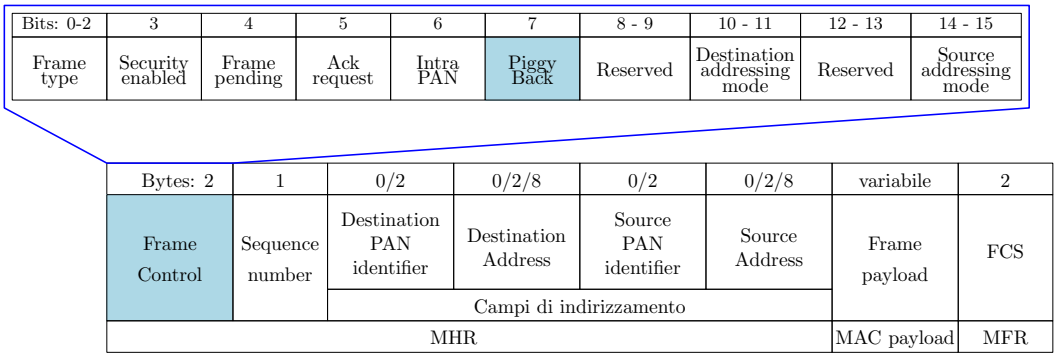


Figura 4.1: Modifica alla struttura del data frame per accogliere il bit di controllo relativo al PiggyBack.

command frame non ha alcuna ripercussione sui risultati dell'analisi poiché i fenomeni interferenziali non hanno correlazione con il tipo di frame utilizzato, ma consente invece di agire in modo più selettivo sul codice, velocizzando il processo di implementazione.

Nel momento in cui gli slave ricevono il command frame, tramite l'interpretazione delle informazioni contenute nell'header, comprendono a chi è rivolto il messaggio, che tipo di risposta si aspetta il master e se sia richiesta una risposta di tipo ACK assieme o prima del dato.

4.1.1 Il PiggyBacking

Dopo aver letto le informazioni, lo slave destinatario si preoccupa di inviare le informazioni richieste: in queste simulazioni, tutti i nodi dispongono della modalità denominata *PiggyBacking* [10] che consiste nell'incapsulamento dell'ACK frame direttamente nel frame dati in modo da risparmiare tempo in fase di trasmissione limitandosi ad inviare un solo messaggio.

Se lo slave, leggendo il MAC header del messaggio ricevuto, osserva che il campo *ackRequest* è settato ad uno, invierà in risposta un solo pacchetto di dati contenente nel payload il contenuto informativo necessario e avente *sequence number* pari a quello dell'ACK frame non inviato. La presenza di un bit di controllo, denominato appunto PiggyBack, settato ad uno, farà capire al master che il frame non è un semplice data frame, ma contiene in sè l'ACK richiesto. Dal punto di vista implementativo, si è rivelato necessario aggiungere un bit di controllo alla struttura del data frame, come mostrato in Figura 4.1.

4.1.2 Il coordinatore di rete

Un'altra figura che partecipa alla simulazione è il coordinatore della rete di sensori IEEE 802.15.4. Questo nodo è necessario all'interno di ogni rete e si occupa della gestione

Tabella 4.1: Tabella dei parametri del coordinatore

Parametro	Tipo di dato	Descrizione
`aBaseSlotDuration`	unsigned int	Partecipa a determinare la distanza temporale tra due beacon
`startPAN`	boolean	Definisce se il nodo avvia l'inizializzazione della rete
`panCoord`	boolean	Indica se il nodo agisce da coordinatore della rete

temporale della rete distribuendo i pacchetti di sincronizzazione (denominati *beacon*) che consentono ai nodi di avere lo stesso riferimento temporale. Nella simulazione in esame, il ruolo del coordinatore è stato affidato ad un nodo che rimane estraneo alla topologia di polling: questa scelta è stata fatta per scindere le funzionalità di coordinatore di rete da quelle di gestore dell'informazione (master) anche se, all'atto pratico, si preferisce convogliare le due figure in un unico dispositivo. In ogni caso, l'ambiente di simulazione consente di passare facilmente da una scelta all'altra semplicemente impostando come coordinatore il nodo master, dotandolo delle funzionalità necessarie per essere un nodo FFD. In questo modo, tutti i nodi della rete parteciperebbero, secondo i ruoli, alla trasmissione di dati vera e propria. Per configurare un nodo FFD come coordinatore della rete è sufficiente agire a livello NET impostando i parametri di simulazione `startPAN` e `panCoord` al valore booleano `true`, mentre tutti gli altri parametri concordano con quelli impostati per gli altri nodi della rete. Nelle simulazioni oggetto della tesi, il coordinatore occupa un ruolo decisamente secondario; esso partecipa alla fase di inizializzazione della rete, inviando i pacchetti di beacon a tutti i nodi della PAN per poi rimanere quiescente per tutta la durata della simulazione. Per non rendere necessario l'invio di ulteriori pacchetti di sincronizzazione, il tempo tra due beacon consecutivi è stato volutamente allungato rispetto a quello da standard: questa variazione è stata fatta a livello MAC, modificando il valore della costante *aBaseSlotDuration* da 60 a 600, andando ad agire direttamente sul calcolo del tempo di beacon, come accennato nel Capitolo 1 alla Sezione 1.4.1.

4.1.3 Fattori di interferenza

Come osservato nell'introduzione, lo scopo della simulazione è osservare il comportamento della rete WSN quando è sottoposta ad interferenza da parte di sorgenti nella stessa banda, in particolare si valuta la coesistenza con una rete basata sul protocollo IEEE 802.11. Dal punto di vista dell'implementazione, la presenza di queste sorgenti di interferenza è ottenuta introducendo tre moduli, ovvero due dispositivi della rete interfe-

rente e un modulo per la valutazione della coesistenza. I moduli della rete interferente sono realizzati con una struttura analoga a quella delle rete 802.15.4, in Figura 2.2. Essi sono composti di una serie di sotto-moduli rappresentanti i layer dello stack protocollare (secondo le disposizioni necessarie) accompagnati dai moduli per la gestione della mobilità (*mobility*) e per la condivisione di informazioni di stato (*blackboard*). A differenza dei nodi della WSN, ogni nodo di questa rete dispone di un modulo per il livello applicazione dello stack ISO/Open Systems Interconnection (OSI) e di un modulo per la gestione interna degli indirizzi, che nel caso specifico, non è di particolare interesse. Sono, quindi, due i moduli della rete interferente presenti nell'ambiente di simulazione: di questi, uno agisce da trasmettitore (un ruolo che lo standard definisce con il termine Access Point) inviando pacchetti all'altro, che a sua volta svolge il ruolo di stazione ricevente. La definizione dello stack protocollare di questi nodi è molto articolata e permette una simulazione altamente personalizzabile tramite la definizione di svariati parametri; tuttavia, in questa trattazione, si può limitare l'osservazione ai soli parametri di interesse per gli obiettivi proposti ovvero quelli che vanno ad influenzare direttamente l'entità dell'interferenza prodotta. Ad esempio, attraverso i parametri di livello fisico è possibile agire sulle potenze in trasmissione, sulle soglie di rumore, sulle frequenze di trasmissione, mentre con i parametri di livello applicazione è possibile controllare la ripetizione con cui vengono trasmessi i pacchetti di interferenza o la loro durata. La Tabella 4.2 mostra un elenco di alcuni dei parametri che possono essere regolati per definire il funzionamento della rete ed adattarla alle esigenze della simulazione.

Il modulo per la valutazione dell'interferenza è una parte fondamentale per la simulazione; il suo funzionamento, descritto in dettaglio nella Sezione 2.5, consente la valutazione dei fenomeni dell'interferenza nel corso della simulazione. L'implementazione del modulo consente la regolazione di alcuni parametri, come ad esempio la definizione del modello di propagazione utilizzato o il valore di alcune costanti utilizzate nell'algoritmo decisionale. Tuttavia, in questo ambito di applicazione, questi parametri svolgono un ruolo secondario e non verranno presi in considerazione.

4.2 Implementazione degli slave

L'implementazione dei nodi slave ha richiesto l'apporto di piccole modifiche rispetto all'implementazione standard dei nodi della rete: il ruolo dei nodi slave è, infatti, quello di rispondere correttamente alle richieste di dati ricevute dal master, producendo e trasmettendo un pacchetto con le informazioni necessarie. Dal punto di vista strutturale e topologico, ciascuno slave presenta ancora la struttura mostrato in Figura 2.2. Per il raggiungimento degli scopi prefissati, non è stato necessario apportare modifiche al layer PHY poiché questo livello di funzionamento non è coinvolto in modo diretto nelle ritrasmissioni: il suo scopo è fare da tramite, consentendo ai livelli superiori di realizzare la

Tabella 4.2: Tabella dei parametri dei moduli della rete 802.11

sim.host[*].appl		
headerLength	integer	Lunghezza del PHY header (in bits)
propModel	numeric	Modello per la propagazione
receiver	numeric	Indica il numero di ricevitori
sender	boolean	Indica se tale nodo è un trasmettitore
sendAck	boolean	Indica se è richiesto ACK frame di risposta
waitForAck	numeric	Tempo di attesa per riceve un ACK frame
waitBeforeSend	numeric	Tempo di attesa tra due invii consecutivi
waitBeforeFirstSend	numeric	Attesa iniziale prima di iniziare gli invii
sim.host[*].net		
headerLength	integer	Lunghezza del NET header (in bits)
sim.host[*].nic.mac		
headerLength	integer	Lunghezza del MAC header (in bits)
queueLength	integer	Lunghezza della coda
bitrate	numeric	Velocità di trasmissione
defaultChannel	numeric	Canale selezionato
sim.host[*].nic.snrEval		
headerLength	integer	Lunghezza del PHY header (in bits)
transmitterPower	numeric	Potenza di trasmissione [mW]
carrierFrequency	numeric	Frequenza portante
thermalNoise	numeric	Rumore termico

comunicazione. Il sublayer MAC, nella sua implementazione su modulo `MacRfd_802154`, ha invece subito alcune modifiche proprio poiché coinvolto direttamente del sistema delle ritrasmissioni. Inoltre, il suo ruolo di generatore di informazioni è compatibile con l'implementazione da standard quindi non è stato necessario agire pesantemente sul codice.

Comportamento in ricezione

Il cambiamento più importante è stato l'introduzione di un sistema di gestione dei command frame da parte del master: quando uno slave riceve un messaggio, la funzione `handleLowerMsg` ne determina il tipo di frame e, con una struttura *switch-case*, avvia la corrispondente funzione di gestione. Nel caso di un command frame, la funzione di gestione utilizzata è `handleCommandFrame`, che realizza una serie di operazioni in successione: come prima cosa controlla che lo stato interno del livello MAC sia compatibile con la ricezione di un command frame, poi legge, dal campo ID del frame stesso, il tipo di frame ricevuto e, in base a questo, una struttura switch-case decide che azione avviare. A questo punto, se il command frame è di tipo `DATA_REQUEST`, viene chiamata la funzione `handleDataRequestFrame`: è questa funzione che si occupa praticamente di gestire la ricezione del comando dal master. Qui viene osservato se il command frame include la richiesta di un ACK di conferma e, in caso affermativo, avvia la generazione e l'invio del frame stesso; contestualmente si considera se utilizzare la modalità di PiggyBack, a seconda delle impostazioni dello slave. Infine viene inviata al *livello NET* la richiesta di generazione di messaggi utilizzando un comando di tipo `MCPS.Data-Request.control`.

A livello NET, l'unica modifica apportata al codice è stata la gestione dei comandi provenienti dal layer inferiore. Per raggiungere questo scopo sono stati utilizzate le funzionalità delle unità `MCPS` e `MLME`, descritte nella Sezione 1.4, in particolare sfruttando l'esistenza di un collegamento tra i layer per inoltrare la richiesta della creazione del dato. Questo passaggio di informazioni è stato realizzato andando ad agire sulla funzione `handleLowerControl()` ed impostando l'invio del pacchetto alla ricezione di un comando di controllo di tipo `MCPS_DATA_REQUEST`.

Comportamento in trasmissione

L'arrivo del comando di controllo al livello NET avvia la sequenza di trasmissione del messaggio, che viene generato dalla funzione `sendPacket()`: questa imposta come destinatario il master della rete e, tramite la funzione `sendDown()`, passa il contenuto informativo al livello MAC. L'arrivo del messaggio a questo livello aziona la funzione `handleUpperMsg()` che si occupa semplicemente di incapsulare il contenuto informativo del livello superiore e di mettere il *mac-Frame* così ottenuto nella coda di trasmissione, denominata `macTxBuffer`. A questo punto è la macchina a stati finiti, interna al MAC

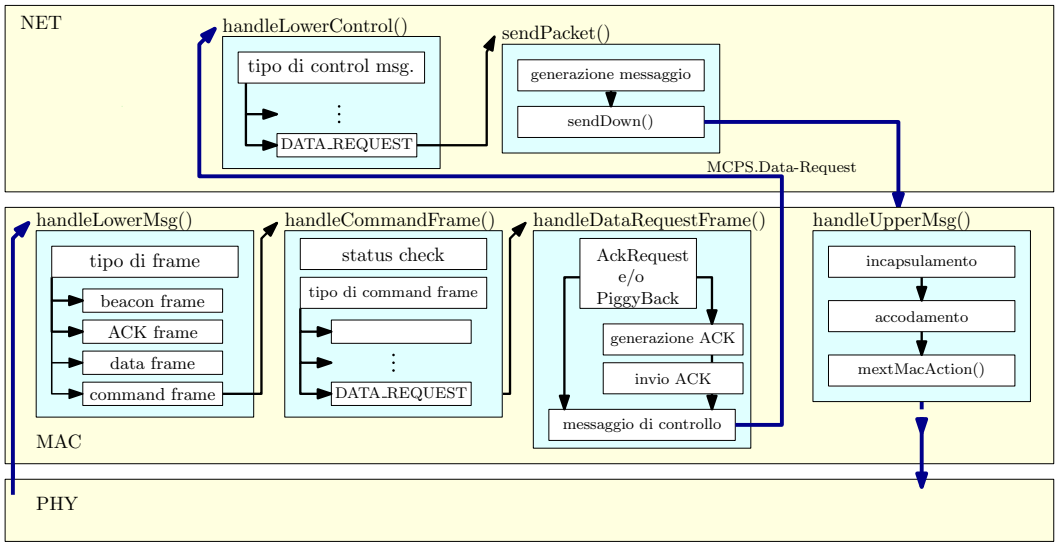


Figura 4.2: Schema delle funzioni svolte da uno slave nella ricezione e risposta ad una richiesta di dati.

layer, che si occupa di trasmettere il pacchetto al MAC Layer del nodo master, utilizzando i servizi di trasmissione del livello fisico sottostante.

4.3 Implementazione del nodo master

Per realizzare l'implementazione di un nodo master è stata utilizzata la stessa struttura degli altri nodi della rete, comprensiva quindi di tutti i moduli illustratati in Figura 2.2. L'implementazione del livello NET è molto diversa da quella dei nodi salve perché deve prevedere la realizzazione di tutte le funzionalità tipiche di un nodo master, descritte in seguito; per l'implementazione del sublayer MAC si è utilizzato il modulo base del protocollo e, sfruttando l'ereditarietà, la si è estesa aggiungendo le funzioni necessarie per la gestione delle ritrasmissioni. Lo strato fisico dello stack protocollare invece è rimasto del tutto invariato, ed è stato utilizzato ancora una volta come fornitore di servizi, in particolare fornendo il servizio di ricetrasmissione dei pacchetti.

4.3.1 La SlaveList

Tutto il sistema per la gestione della comunicazione e per la scansione degli slave, si basa sul funzionamento di una struttura dati, denominata *SlaveList*, che raccoglie le informazioni su tutti i nodi che devono essere contattati in ogni ciclo temporale. Dal punto di vista implementativo, la SlaveList è stata realizzata a partire da una lista, il cui elemento fondamentale (*entry*) è il tipo di dato - struttura che prende il nome di `slaveListEntry_t`. Una rappresentazione schematica della struttura di una entry e

della SlaveList si può osservare in Figura 4.3 mentre un estratto della sua definizione appare nel Listing 4.1. Si nota che ciascuna entry è una collezione di sette campi:

slaveAddress É il campo principale della entry e contiene l'indirizzo MAC dello slave a cui la entry è associata. É utilizzato sia in fase di invio dei pacchetti, per determinare il destinatario, sia nella riorganizzazione della lista alla fine di ciascun Period Cycle.

checkedFlag É un campo booleano utilizzato durante lo scorrimento della lista. Consente di distinguere tra gli indirizzi dei nodi che sono già stati processati, nella corrente finestra temporale, e quelli che devono ancora esserlo. La presenza di questo campo si rivela necessaria per non dover rimuovere le entry dalla lista: un elemento della SlaveList con `checkedFlag` `true` verrà ignorato ai fini dello scorrimento della lista senza bisogno di essere realmente eliminato. In questo modo si può risparmiare tempo alla fine del PeriodCycle evitando di dover caricare in memoria la lista ma semplicemente scorrendola e cambiando il campo checkedFlag a `false`.

bufferFrame É un campo di tipo `macTxBufferEntry_t` che contiene quindi al suo interno questa struttura. Questa inclusione, sebbene non strettamente necessaria per le dinamiche di funzionamento, consente di agevolare le operazioni di invio dei messaggi. Come osservato nel Capitolo 2 l'invio dei messaggi avviene tramite la creazione di una struttura che incapsula i MAC frame e li mette in una coda di trasmissione: osservato che in questo frangente non si è interessati a trasmettere un determinato contenuto informativo, è possibile utilizzare lo stesso MAC frame, già incapsulato, in modo da velocizzare la procedura di accodamento. Per questo motivo una copia del frame incapsulato viene conservata all'interno della `slaveListEntry_t`.

totalFrameSent, successFrameSent, failedFrameSent Sono tre campi di tipo intero che contengono informazioni sul numero di trasmissioni totali, con successo e fallite, verso lo slave associato alla entry. L'uso di questi campi si rivela particolarmente importante in fase di debug e di analisi dei dati, per osservare nel dettaglio il comportamento del master.

statParameter è un campo numerico, di tipo `double`, che memorizza il valore statistico definito dalla formula in Sezione 3.3.4 e rappresenta il fattore discriminante nel caso in cui la tecnica di ritrasmissione sia AQR.

4.3.2 Inizializzazione della SlaveList

Fin dalla fase di inizializzazione, il master deve provvedere alla creazione della SlaveList, includendo al suo interno tutti le informazioni relative ai nodi ad essa associati: questa

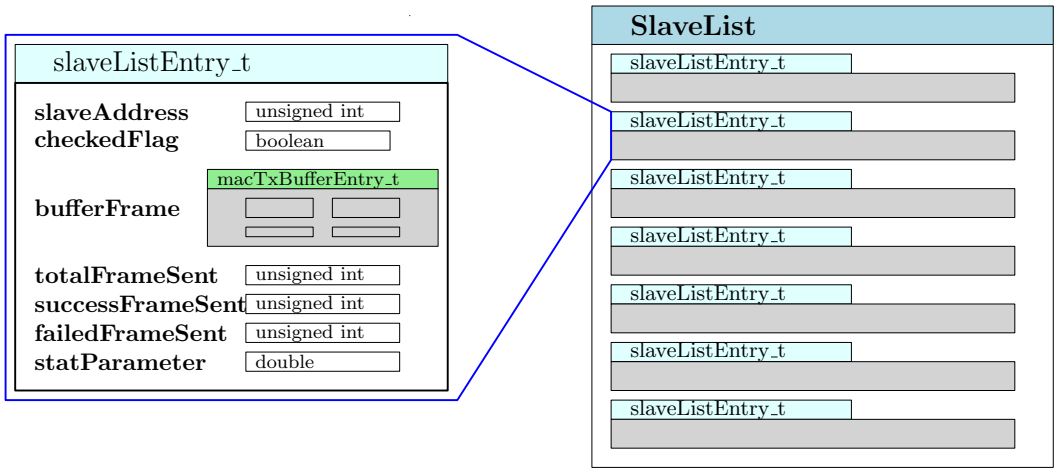


Figura 4.3: Rappresentazione schematica del contenuto della SlaveList ed di ciascuna delle sue entry.

Listing 4.1: Definizionde della SlaveList

```
typedef struct {
        /** Mac address of the associated slave*/
        unsigned int slaveAddress; // default : 0xffff
        /** Indicate if the Slave has been polled during this cycle*/
        bool checkedFlag;                                   // default:false
        /* Entry that contains the bufferFrame that has to be moved into
         * the bufferTx frame */
        macTxBufferEntry_t bufferFrame;
        /** Total number of frame_request generated from the NET layer,
         * directed to this slave*/
        unsigned int totalFrameSent;
        /** Number of frame_request with failed transmission to this slave*/
        unsigned int failedFrameSent;
        /** Number of frame_request with success transmission to this slave*/
        unsigned int successFrameSent;
        /** Statistic parameter that shows the reliability of the slave*/
        double statParameter;
} slaveListEntry_t;
/** @brief The main data structure that contains all the information
  * needed to perform the retransmission strategy used */
std::list<slaveListEntry_t> slaveList;
```

Listing 4.2: Creazione di un messaggio

```
1        ...
2        macCycleTimer= new cMessage("CycleTimer");
3        ...
```

funzionalità è offerta all'interno del metodo `handleLowerMsg()` mentre gli indirizzi da collezionare sono trasmessi dal livello superiore. La procedura di inizializzazione parte dal livello NET: qui sono memorizzati gli indirizzi di tutti i nodi della rete e, uno alla volta, vengono generati tanti messaggi quanti sono gli slave della rete, indirizzandoli di volta in volta a un nodo diverso. Nel caso in esame, si è scelto di assegnare degli indirizzi MAC incrementali, in modo da poter realizzare questa fase con l'uso di un semplice contatore. I pacchetti di livello NET vengono passati al livello sottostante inserendoli nella lista `sendBuffer` che si occupa a sua volta dell'invio. I messaggi generati dal NET layer vengono raccolti dal sublayer MAC e vengono processati dalla funzione `handleUpperMsg()`: qui il pacchetto ricevuto viene letto e dal suo header viene estratto l'indirizzo di destinazione del pacchetto. Successivamente viene creata una struttura del tipo `SlaveListEntry_t` e si controlla se nella SlaveList è già presente una entry associata a questo indirizzo: in caso affermativo la entry viene rimpiazzata con quella nuova, altrimenti la nuova entry viene semplicemente aggiunta. Al termine di questa operazione è compito del sublayer MAC avvisare il livello superiore che l'operazione si è conclusa, tramite l'invio di un ulteriore pacchetto di tipo `MCPS-DATA.confrim`, utilizzandolo in modo relativamente improprio. La ricezione da parte del NET layer di un messaggio di tipo `MCPS-DATA.confirm` viene letta come una conferma del riuscito inserimento del MAC address all'interno della SlaveList, consentendo l'invio del successivo messaggio in coda presso il NET. La fase di inizializzazione della lista occupa un zona molto limitata dell'estensione temporale a disposizione, concludendosi rapidamente dopo l'inizializzazione dei moduli.

4.3.3 Gestione temporale

Il modulo `MacRfd_802154_poll` deve gestire tutte le fasi della comunicazione con gli slave, compresa la gestione temporale del periodo. Per fare questo, si avvale dell'uso di temporizzatori interni azionati in corrispondenza di determinati eventi e la cui scadenza determina l'insorgere di altri eventi.

In OMNeT++, ogni modulo gestisce i propri timer con lo stesso sistema con cui vengono inviati i messaggi, ovvero spedendo a se stesso un *self-message*: la creazione di un self message avviene tipicamente nella fase di inizializzazione del modulo, con l'uso della sintassi illustrata dal Listing 4.2.

Il modulo considerato crea, in fase di inizializzazione, due timer utilizzati per la gestione temporale, il timer `macCycleTimer` e il timer `macPeriodicWindowTimer`. Il primo definisce la durata di un intero ciclo, all'inizio del quale avviene la re-inizializzazione delle informazioni sui nodi, ricominciando la scansione degli slave; il secondo timer definisce una ragione temporale di contenimento all'interno della quale il master ha la facoltà di iniziare una richiesta di dati verso un determinato slave. Questo secondo timer viene utilizzato per delimitare la porzione temporale del ciclo in cui è ammessa la trasmissione: prima di ogni trasmissione, infatti, il nodo master controlla che il timer `macPeriodicWindowTimer` non sia scaduto, utilizzando la sua presenza come indicatore della possibilità di trasmettere. Questo consente di evitare la situazione in cui, a cavallo di due cicli diversi vi sia una trasmissione in corso. Lo scadere del timer `macCycleTimer`, invece, avvia la funzione di reinizializzazione della lista, separando due set di trasmissioni adiacenti.

4.3.4 Reinizializzazione della SlaveList

All'inizio di ogni ciclo temporale, la lista viene reinizializzata per effetto dello scadere del timer `macCycleTimer`; la funzione che svolge questa operazione è denominata `initSlaveList()`, ed assume comportamenti diversi al variare della tecnica di ritrasmissione selezionata. Lo scopo finale di questo metodo è rendere la lista pronta per essere utilizzata nel ciclo successivo e iniziare la prima trasmissione del ciclo: risulta quindi necessario *riordinare* la lista ed *eliminare* le informazioni temporanee in essa contenute. Come si apprende dalla lettura della sezione 3.3, le tecniche di ritrasmissione BIR, Unboundled Immediate Retransmission (UIR) e QR richiedono che la lista sia ordinata all'inizio di ogni ciclo, secondo un'ordine arbitrario ma prefissato e invariante. La tecnica AQR invece richiede che l'ordinamento avvenga in modo da privilegiare le entry a cui sia associato un parametro statistico prossimo all'unità, operando di fatto un ordinamento decrescente. Per realizzare queste due modalità in modo semplice e flessibile, si è utilizzata la funzione `sort()` associata alla SlaveList: questa funzione accetta come parametro un *oggetto funzione* (o *funtore*) le cui caratteristiche determinano il tipo di ordinamento. In particolare l'oggetto funzione deve disporre dell'operatore ``()'' che restituisca un valore booleano che indichi se i due oggetti sono stati passati in ordine strettamente decrescente; l'espressione può anche essere scritta in una sola riga, ottimizzando l'algoritmo di ricerca. In questo caso sono stati realizzati due oggetti-funzione riportati nel Listing 4.3, che a seconda della tecnica di ritrasmissione scelta, realizzavano l'ordinamento necessario. Questo approccio fornisce flessibilità al codice consentendo l'aggiunta di altre tecniche di ordinamento in modo semplice e modulare.
La seconda operazione necessaria in questa fase è la rimozione delle informazioni temporanee contenute nella SlaveList: in particolare è necessario porre a zero il parametro

Listing 4.3: Definizione degli oggetti funzione nei due casi osservati estratto dal file MacRfd_802154_poll.h

```
struct compare_entry_by_address {
        bool operator()(slaveListEntry_t first, slaveListEntry_t second)
        {return ( (unsigned int) first.slaveAddress < (unsigned int) second.slaveAddress);}
};
struct compare_entry_by_statistic {
        bool operator()(slaveListEntry_t first, slaveListEntry_t second)
        {return ( (double) first.statParameter > (double) second.statParameter);}
};
```

che conta il numero di ritrasmissioni eseguite nel ciclo e marcare con `false` il campo checkedFlag, riabilitando tutte le entry. A seguito di questa fase, il primo nodo della SlaveList viene trasmesso, dando via alla serie di trasmissioni e ritrasmissioni.

4.3.5 Trasmissione e Ritrasmissione

Il principio di funzionamento di tutto il sistema di ritrasmissione si basa su un criterio implicito che può essere denominato *criterio del capolista*: questo implica che, in ogni momento, l'indirizzo che si trova al primo posto della SlaveList, appartiene al prossimo nodo a cui deve essere inviata la richiesta di dati. Questo permette una gestione più semplificata ed intuitiva della procedura di trasmissione e consente una successiva aggiunta di altre tecniche di ritrasmissione a quelle già esistenti. All'inizio di ogni ciclo, la prima operazione eseguita dopo l'inizializzazione della lista, è la trasmissione del capolista, se presente. In generale è possibile immaginare che la trasmissione di dati dal MAC layer allo PHY layer, avvenga ponendo il frame da trasmettere all'interno del buffer di trasmissione `macTxBufferFrame`: l'implementazione di tutte le operazioni compiute dal layer per realizzare la fase di arbitraggio e l'accesso al mezzo viene importato dalle precedenti versioni del progetto con cui si è lavorato, come illustaro nella Sezione 1.4.1. Contestualmente alla trasmissione del frame, viene avviato il timer `macWaitAckTimer` che, come suggerisce il nome, tiene traccia del tempo di attesa della risposta da parte dello slave contattato. La funzione fondamentale che si occupa della trasmissione o ritrasmissione dei frame prende il nome di `transmitFrame()` che viene invocata allo scadere di ogni SlotTimer, con un parametro che dipende da ciò che è avvenuto nel corso dello slot precedente.

Infatti, se nella sezione precedente il master riceve un *data frame* da parte dello slave contattato, la funzione `transmitFrame()` sarà invocata con parametro `true`, mentre se si è verificato lo scadere del timer associato all'ACK frame, il parametro passato alla funzione sarà `false`. In altre parole, questa seconda situazione indica che la trasmissione precedente è fallita a causa dell'interferenza e che è necessario provvedere all'applicazione delle tecniche di ritrasmissione.

La funzione `transmitFrame()` presenta una struttura molto schematica ed è composta da: una fase di aggiornamento delle variabili interne e di emissione dei segnali utili alla simulazione; una fase di gestione della ritrasmissione, tipica della tecnica adottata, per definire il nuovo capolista; un fase finale di invio del pacchetto.

BIR

La prima tecnica di ritrasmissione implementata è la BIR, descritta nella sezione 3.3.1: una volta all'interno della funzione `transmitFrame`, il master controlla il valore del campo `retries` all'interno del `bufferFrame` del capolista, confrontandolo con il valore contenuto nella variabile `aMaxFrameRetries`, che delimita il valore massimo di ritrasmissioni verso uno stesso nodo per ogni ciclo. Se il numero di ritrasmissioni eccede questo valore massimo impostato, il campo checkedFlag della lista è posto a `true` e la entry viene posta in coda alla lista; viceversa, se il numero di ritrasmissioni non ha superato la soglia indicata, la lista non viene modificata in alcun modo, lasciando la entry in testa alla lista in modo che sarà ritrasmessa successivamente.

UIR

Questa seconda tecnica si presta particolarmente a questo tipo di implementazione data la sua semplicità: nel caso in cui si sia verificata un collisione nel precedente invio (e il parametro di funzione è quindi `false`), non sono necessarie operazioni poiché la entry deve essere mantenuta a capo della lista. In alternativa, se la trasmissione precedente è avvenuta con successo, il campo `checkedFlag` della lista è posto a `true` e la entry viene messa in coda alla lista, proprio come nel caso della tecnica BIR.

QR

Questa tecnica di ritrasmissione, descritta in sezione 3.3.3, viene implementata in modo leggermente diverso rispetto alle precedenti: si parte con l'immaginare la SlaveList suddivisa in due sezioni, una composta da entry già processate (*polled*) ed una, in testa, composta da entry ancora da processare (*unpolled*). Ogni volta che la funzione `transmitFrame()` viene invocata con parametro `false`, il capolista viene posto in coda alla parte della lista non processata, in modo da renderlo disponibile per una successiva ritrasmissione; al contrario, se il parametro di funzione è `true`, viene attivato il campo `checkedFlag` e la entry è posta in coda alla parte di lista già processata. In questo modo, non appena nello scorrimento della lista si incontra una entry con il campo checkedFlag pari a `true`, risulta chiaro che tutti i nodi della lista sono stati processati: sarà la funzione di reinizializzazione ad occuparsi di ripristinare il corretto ordine delle entry.

AQR

Quest'ultima tecnica di ritrasmissione si presenta come una variante della precedente ed è stata realizzata, quindi, un'implementazione del tutto analoga. Come nel caso precedente, la SlaveList è suddivisa nelle due sezioni *polled* e *unpolled*, e la entry capolista viene spostata in coda alla sezione *unpolled* in caso di collisione o all'interno, ordinata, della sezione *polled* in caso di trasmissione efficace. La differenza principale con la strategia precedente sta nel metodo reinizializzazione della lista, che viene ordinata utilizzando un diverso criterio di ordinamento, basato sul parametro statistico.

Capitolo 5

Analisi dei dati

Introduzione

In questo capitolo verranno presentati e discussi i risultati delle simulazioni eseguite, indicando nel dettaglio quali prove sono state effettuate e le impostazioni utilizzate per la loro realizzazione. Tutte le simulazione condotte hanno lo scopo di osservare quanto e come l'implementazione realizzata sia aderente al modello reale e, per questo motivo, le prove eseguite ricalcano l'ambiente di osservazione presentato nell'articolo [2] e in [11]. Verranno di seguito descritte le procedure per la raccolta dei dati, le impostazioni di simulazione che di volta in volta si sono utilizzate e le modalità di elaborazione delle informazioni; verranno inoltre presentati alcuni grafici riassuntivi particolarmente significativi e, infine, si potranno comparare i risultati ottenuti con quelli attesi traendo delle conclusioni sulla bontà dell'implementazione realizzata, evidenziandone pregi e possibili migliorie.

5.1 Il meccanismo dei segnali

Lo scopo principale di eseguire delle simulazioni è sicuramente acquisire una serie di informazioni che verranno analizzate ed utilizzate nelle fasi di post-elaborazione. Come illustrato nel manuale [12], OMNeT++ offre varie strategie che consentono di raccogliere dati nel corso della simulazione per poi utilizzarli in un secondo momento a seconda degli scopi; in particolare è possibile distinguere tra risultati di tipo *scalare*, costituiti da un singolo valore per ciascuna simulazione, oppure di tipo *vettoriale*, composti da una successione temporale di dati per ogni simulazione eseguita. In generale è preferibile utilizzare dati scalari quando si desidera ottenere una descrizione riassuntiva dell'esito della simulazione - sia questo un risultato di tipo *numerico*, come il numero totale di pacchetti persi, o un dato *statistico*, come il tempo medio di attesa per un determinato evento, o la sua deviazione standard. Al contrario è consigliabile l'impiego di grandezze

vettoriali quando si è interessati ad osservare l'evoluzione di una grandezza nel corso della simulazione - come può essere la lunghezza di una lista d'attesa, i tempi di ritardo nelle trasmissioni dei vari pacchetti, lo stato interno dei moduli, etc. Il meccanismo di raccolta dati può avvenire sia in modo *esplicito*, indicando nel codice sorgente (comportamentale) di ciascun modulo le operazione da eseguire, avvalendosi di tutte le funzioni messe a disposizione dalle librerie C++, sia ricorrendo all'impiego delle funzioni di simulazione, tra cui compaiono, dalla versione 4.1 di OMNeT++, anche i *segnali di simulazione*.

Quest'ultimo approccio è in molti casi da preferire a quello tradizionale poiché consente di raccogliere dati nella forma che si preferisce, passando facilmente da dati scalari a dati vettoriali, il tutto senza richiedere di volta in volta, pesanti manipolazioni del codice sorgente. Il meccanismo dei segnali è un concetto molto versatile che si presta ad usi diversi e funzionano nel modo seguente: i segnali, in generale, sono emessi dai componenti e si propagano all'interno della struttura gerarchica dal modulo più interno fino al modulo principale, nell'ottica di utilizzare moduli composti (*compound modules*). A ciascun livello, il segnale può attivare delle porzioni di codice denominate *listeners* il cui compito è proprio registrare i segnali ricevuti ed utilizzarli per gli scopi prefissati. Un segnale attiva un determinato listener solo se questo riconosce il codice identificativo del segnale ricevuto ed è stata precedentemente fatta una procedura di registrazione di tale codice presso il listener; in alcuni casi è possibile associare a più segnali uno stesso codice identificativo in modo che questi possano attivare gli stessi listeners. Per queste ragioni il meccanismo dei segnali può essere utilizzato per l'implementazione di comunicazioni basate sulla pubblicazione di informazioni presso moduli sottoscrittori, oltre che nella raccolta dati.

La particolarità dei segnali consente, infatti, di scegliere di volta in volta quale tipo di informazioni si desidera memorizzare, andando semplicemente ad agire sul file di configurazione: è possibile ottenere un livello di analisi regolabile e piacere, determinando se e quali dati raccogliere, oppure si può passare velocemente da un output vettoriale ad uno scalare, come può essere passare da una collezione dei tempi di attesa al solo valor medio, o ancora è possibile eseguire delle semplici elaborazioni sui dati prima che questi siano raccolti, come delle valutazioni di soglia o la creazione di istogrammi.
Il meccanismo dei segnali può essere utilizzato in combinazione con le proprietà del linguaggio NED per ottenere un utile disaccoppiamento tra la generazione dei dati e la loro memorizzazione con lo scopo di aumentare la flessibilità complessiva. La dichiarazione di segnali statistici avviene a livello di file NED del modulo generatore, utilizzando una precisa sintassi, come in Listing 5.1; la parola chiave `@statistic` permette di definire il tipo di dato che si desidera raccogliere, indicando il nome del segnale, un eventuale titolo aggiuntivo e il tipo di dato che si vuole ottenere. Questo ultimo campo è quello che determina il dato ottenuto (che può essere vector, count, last, sum, mean, min, max, timeavg, stats, histogram, etc.) e può essere modificato anche dal file di configurazione,

Listing 5.1: Creazione di segnali per uso statistico

```
parameters:
...
        @statistic[Master_success](title="Successful_poll"; record=vector);
        @statistic[Master_fail](title="Failed_poll"; record=vector);
        @statistic[Master_cycle](title="EndCycle_poll"; record=vector);
...
```

rendendo la modifica del parametro ancor più agevole. L'emissione dei segnali avviene a livello di codice sorgente, utilizzando la funzione `emit()`; questa consente di inviare ogni tipo di dato numerico o, più in generale, qualsiasi tipo di oggetto al meccanismo dei segnali, in modo che questo venga passato al listener in ascolto. Nel caso di segnali ad uso statistico, non è necessario implementare alcun algoritmo di ascolto poiché l'uso della parola chiave `@statistic` indica già l'impiego del segnale; sarà quindi cura del simulatore stesso gestire questo tipo di informazioni. Solitamente è buona norma che i segnali siano creati in fase di inizializzazione, utilizzando la funzione `registerSignal()` e che venga loro assegnato un *titolo* che ne permetta l'identificazione nel momento dell'analisi dei risultati.

5.1.1 I segnali utilizzati

Nelle simulazioni correnti sono stati previsti quattro tipi di segnali di simulazione per la raccolta di dati statistici: tre implementati a livello di nodo master e uno presente nelle implementazioni degli slave. Posto che lo scopo delle simulazioni era la raccolta di dati circa i tempi di servizio dei nodi e le probabilità di successo, si è deciso di creare dei segnali che tenessero traccia dei successi e degli insuccessi delle trasmissioni compiute dal nodo master. Questi segnali vengono generati in corrispondenza del rilevamento di una trasmissione, producendo un segnale di tipo `safeArrivalSignal`, registrato con il nome di *Master_Success* in caso affermativo oppure un segnale `unArrivalSignal`, denominato *Master_Fail*, in caso di trasmissione corrotta. Ciascuno di questi segnali emette un valore numerico, pari all'indirizzo MAC dell'ultimo nodo destinatario, associato al relativo *timestamp* che riporta l'istante in cui il segnale viene emesso. Alla fine della simulazione si otterranno due vettori di coppie di valori. Il terzo segnale, localizzabile a livello di nodo master, prende il nome di *Master_Cycle* e il suo scopo è identificare la conclusione di ciascun ciclo di polling, inviando un segnale contenente il numero di nodi correttamente serviti dall'inizio della simulazione al momento dell'emissione. Infine, una quarta tipologia di segnale viene emessa dagli slave ogni volta che vengono contattati dal nodo master; questo segnale non è associabile ad una corretta trasmissione completa tra il master e lo slave poiché il trasferimento di dati potrebbe essere corrotto nella sua seconda fase, quella dallo slave al master.

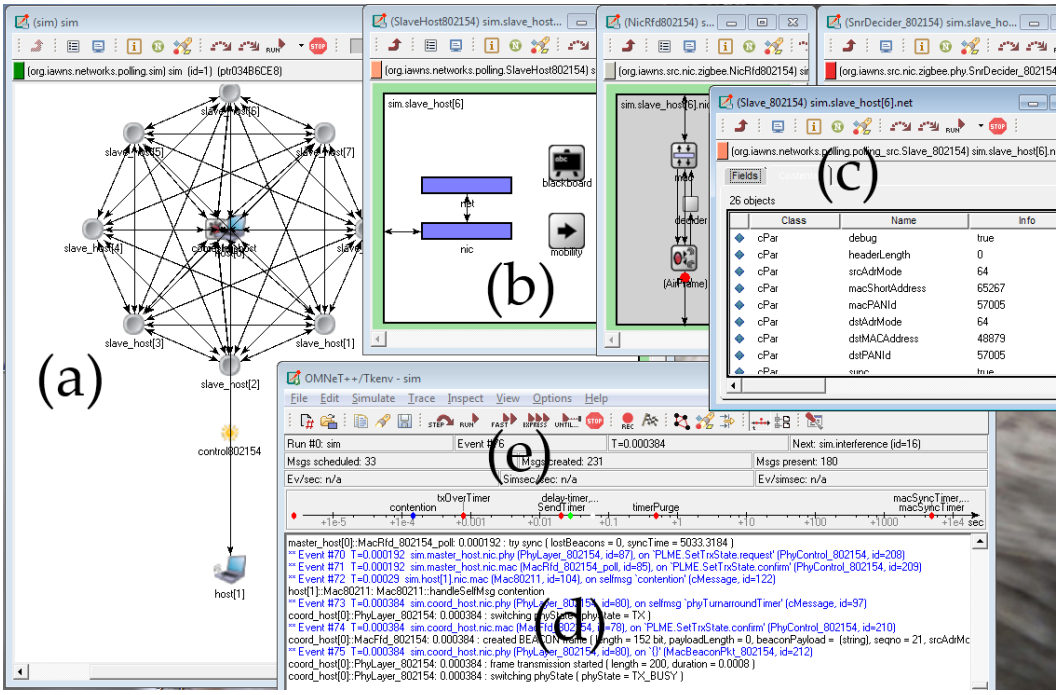


Figura 5.1: Una schermata di simulazione.
In figura sono osservabili: a) la finestra principale della rete, b) dettaglio sui moduli interni ad un nodo, c) e sulle sue variabili, d) console dei messaggi di simulazione con e) controlli di simulazione e dati riepilogativi.

5.2 La raccolta di dati

Una volta impostati tutti i parametri necessari, è possibile avviare la simulazione sia in forma grafica, sia da riga di comando. Nel primo caso, verrà avviato l'ambiente grafico TkEnv che, attraverso un'interfaccia utente semplificata, consente di mostrare in che modo avvengono le transizioni di informazioni tra i moduli; ogni modulo, infatti, è rappresentato con un'icona e le connessioni presenti con gli altri dispositivi sono indicate con delle linee continue su cui si possono osservare delle animazioni rappresentanti i messaggi scambiati. Uno screenshot esemplificativo è osservabile in Figura 5.1 in cui sono riportate i principali strumenti di osservazione e controllo della simulazione.

Questo ambiente si rivela particolarmente utile nella fase di programmazione e debugging perché consente un controllo completo sul funzionamento del simulatore, ma, al contrario, si rivela poco adatto alla raccolta di grandi quantità di dati. In molti casi, perciò, è necessario realizzare diverse simulazioni successive che si differenziano tra loro per i parametri usati e memorizzare i vettori e gli scalari risultanti su files diversi; per questo scopo è sicuramente preferibile l'uso di un simulatore a riga di comando che, oltre a eseguire le medesime operazioni in tempi decisamente più ridotti, si presta facilmente all'esecuzione ripetitiva, tramite la creazione di script di esecuzione. Anche in questo

Listing 5.2: Variabili di simulazione

```
1 ...
2 sim.host[*].appl.waitBeforeSend = ${gap = 10e-3, 20e-3, 30e-3, 50e-3}
3 sim.master_host.nic.mac.retStrategy = ${strategy= 1,2,3,4}
4 ...
```

OMNeT viene incontro alle esigenze dell'utente, eseguendo tutte queste operazioni in modo automatizzato grazie a una opportuna sintassi che, utilizzata nel file di configurazione (`*.ini`), indica al software quali parametri far variare tra le diverse esecuzioni. Un esempio di sintassi è osservabile nel Listing 5.2 in cui si vogliono eseguire sedici diverse simulazioni, con tutte le possibili combinazioni dei parametri `waitBeforeSend` e `retStrategy`. Il file di configurazione, infatti, ha lo scopo di raccogliere tutte le impostazioni necessarie per l'esecuzione della simulazione, che viene letto in fase di esecuzione importando i valori in esso contenuti.

Ciascuna simulazione porta alla produzione di 3 files di testo: il primo, con estensione `.sca` contiene tutti i dati di natura scalare; il secondo, con estensione `.vec` contiene tutti i dati di natura vettoriale; il terzo, se presente, con estensione `.vci` raccoglie gli eventuali istogrammi generati nel corso della simulazione. OMNeT++ dispone di un software dedicato, in grado di elaborare i dati prodotti dalle simulazioni e di manipolarli per ottenere gli output richiesti; questo applicativo, integrato nell'ambiente di sviluppo, consente di eseguire filtraggi sui dati isolando quelli di interesse, e di ottenere grafici e istogrammi secondo le necessità. Tuttavia, in questa sede, si è preferito utilizzare MATLAB per l'analisi e l'elaborazione dei dati, utilizzando l'applicativo integrato solo per esportare i dati.

5.3 Realizzazione delle simulazioni

A seguito della realizzazione dei moduli, descritta nel Capitolo 4, si è potuti procedere alla simulazione di alcune reti di interesse. Come già osservato, una verifica della bontà dell'esecuzione si può ottenere da un confronto tra risultati ottenuti con prove sperimentali — su dispositivi reali — e i risultati offerti dal simulatore. Ripercorrendo quanto presentato nell'articolo [2] si è cercato di mostrare dei risultati paragonabili, partendo quindi, da un insieme di impostazioni che potessero rappresentare le medesime condizioni iniziali.

5.3.1 Setup di simulazione

Si è scelto di realizzare una rete ad 8 slave ed un nodo master, utilizzando una topologia fissa con gli slave disposti su un anello di raggio pari a 10 metri e il master posizionato centralmente, in modo che i percorsi fisici di collegamento fossero indipendenti dallo slave considerato, come visibile in Figura 5.2.

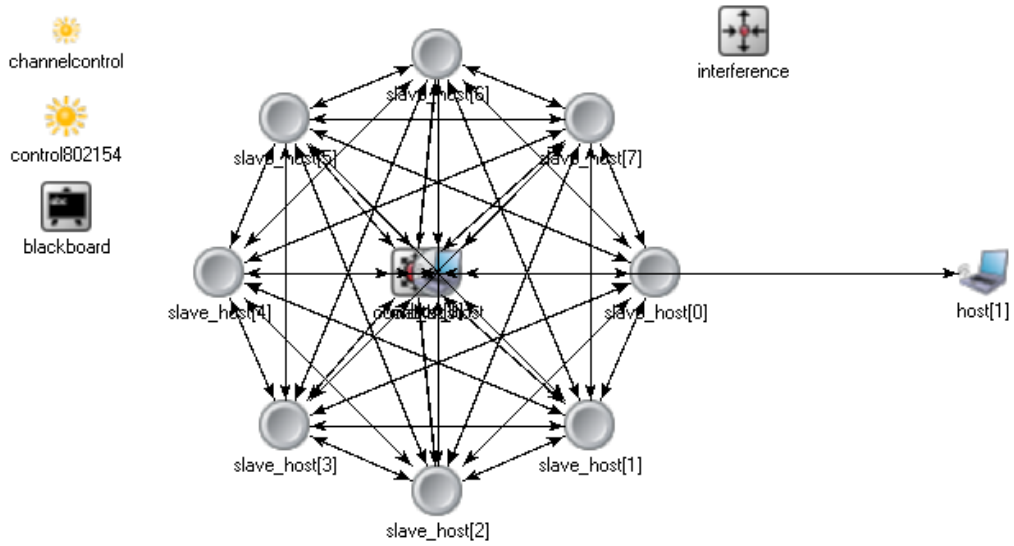


Figura 5.2: Screenshot di simulazione: è visibile la disposizione dei nodi della rete di test.

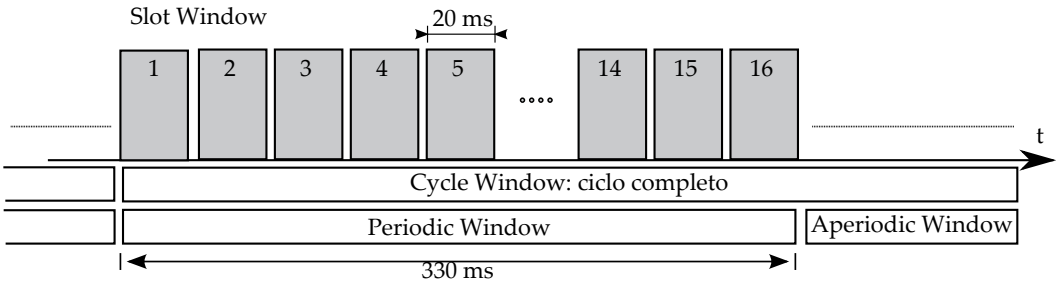


Figura 5.3: Schema della ripartizione temporale di ciascun ciclo.

La rete interferente è composta di due stazioni: una trasmittente, posizionata al centro della rete di sensori, e una ricevente posta esternamente. Il segnale interferente utilizzato è un'insieme di pacchetti di una rete 802.11, con una durata aleatoria uniforme compresa tra 1 ms e 10 ms. Il tempo che intercorre tra due pacchetti dell'interferente, invece, è modellabile come una variabile aleatoria esponenziale con media regolabile tramite parametro di simulazione, ma generalmente posta a 10 ms. La potenza della trasmittente interferente è settata ad un valore costante di -10 dBm, mentre sui nodi è osservabile una potenza dovuta all'azione del master di circa $0.676\mu W$ a fronte di una sensitività di $10pW$. Sono inoltre costanti tutte le dimensioni delle finestre temporali utilizzate dalla rete di polling, con tempo di ciclo pari a 400 ms e finestra attiva di durata 330 ms, come pure il numero di slot temporali assegnati, 16, ciascuno di durata pari a 20 ms. Un riepilogo delle principali impostazioni si possono trovare raccolte nella tabella 5.1, mentre una rappresentazione delle temporizzazioni si può osservare in Figura 5.3.

Tabella 5.1: Riepilogo dei parametri per la simulazione

Parametro	Valore
Tempo simulato	2050 s
Numero di nodi della rete	8
Distanza slave-master	10 m
Durata della Cycle Window	400 ms
Durata della Periodic Window	330 ms
Numero di Slot Window	16
Durata di ciascun Slot Window	20 ms
Numero di ritrasmissioni concesse (BIR)	2
Valore del parametro α (AQR)	0.9
Durata dell'interferenza	$\mathbb{U}[1,10]$ ms
Tempo di interarrivo	$\{10,20,30,50\}$ ms

5.3.2 Esperimenti eseguiti

Dopo le impostazioni iniziali, si è provveduto alla realizzazione di alcuni esperimenti significativi, con lo scopo congiunto di analizzare il comportamento della rete e valutare il funzionamento del simulatore.

Analisi dei tempi di servizio

Come prima cosa, siamo interessati ad osservare la distribuzione di probabilità del tempo di servizio di un nodo al variare della tecnica di ritrasmissione scelta; per ogni nodo viene osservato il tempo che intercorre tra due trasmissioni con successo e se ne determina l'andamento della distribuzione di probabilità sulla frequenza relativa dei tempi misurati. Questa informazione viene prelevata dall'osservazione del vettore *Master_success*, il vettore che contiene i tempi di polling (con successo) di tutti i nodi della rete. Selezionando di volta in volta, solo i tempi di un singolo nodo, è stato possibile estrarre tutti i tempi di accesso allo slave e, eseguendo differenze successive, tutte le durate dei tempi di accesso. Utilizzando le funzionalità di MATLAB si è poi potuto realizzare un istogramma con la frequenza dei vari risultati che, normalizzato, fornisce una stima della distribuzione di probabilità del tempo che intercorre tra due accessi ad un medesimo nodo. I grafici ottenuti sono osservabili in Figura 5.4: in particolare, nella colonna di destra sono riportati i grafici ottenuti per le varie strategie di ritrasmissione scelte, mentre nella colonna di sinistra sono mostrati i grafici di riferimento, per un successivo confronto.

Analisi del tempo di servizio medio

Da questi grafici abbiamo potuto ricavare l'andamento medio per ogni singola tecnica di trasmissione, considerando congiuntamente il comportamento di tutti i nodi. L'andamento risultante, identificabile con il nome di *andamento medio* per la tecnica in esame, si presta ad una comparazione con le altre tecniche di trasmissione, al fine di osservarne le caratteristiche; tale grafico è osservabile in Figura 5.5.

Analisi della probabilità di fallimento

Un grafico di diversa tipologia ma comunque interessante è quello relativo alla probabilità che un nodo della rete venga servito correttamente in un ciclo; quest'indagine è particolarmente interessante perché permette di osservare più da vicino eventuali sbilanciamenti a favore (o a sfavore) di un determinato nodo. L'informazione necessaria può essere nuovamente prelevata dal vettore *Master_success*, selezionando e contando le occorrenze di ogni singolo nodo all'interno del vettore, che corrispondono al numero di successi. Tale valore, normalizzato al numero totale di cicli eseguiti, permetterà di determinare la percentuale di perdita del nodo, stimandone la probabilità di insuccesso in un ciclo. Il grafico ricavato da questa analisi si può trovare in Figura 5.6.

Analisi del numero medio di nodi non serviti

L'ultimo insieme di prove eseguite aveva lo scopo di indagare il numero di medio di nodi che non viene contattato con successo in un ciclo; questa prova, che offre in forma riassuntiva gli stessi risultati della prova precedente, è stata eseguita più volte variando, oltre che la tecnica di ritrasmissione, anche la distanza media tra l'emissione di due segnali interferenti consecutivi. Lo scopo della prova è verificare in che misura questo parametro influisce sulle prestazioni medie della rete, senza prestare attenzione a eventuali sbilanciamenti, come quelli osservabili nelle prove individuali. I risultati di questo esperimento sono riportati in Figura 5.7.

5.4 Valutazione dei risultati

Il modo migliore per valutare la qualità del simulatore ed avere una visione d'insieme sui risultati raggiunti è confrontare i grafici ottenuti con quelli attesi.

5.4.1 Analisi dei tempi di servizio

Come è possibile notare, tutti i grafici ottenuti rispecchiano quanto si voleva ottenere: l'andamento delle curve di densità di probabilità (in Figura 5.4) segue fedelmente quello

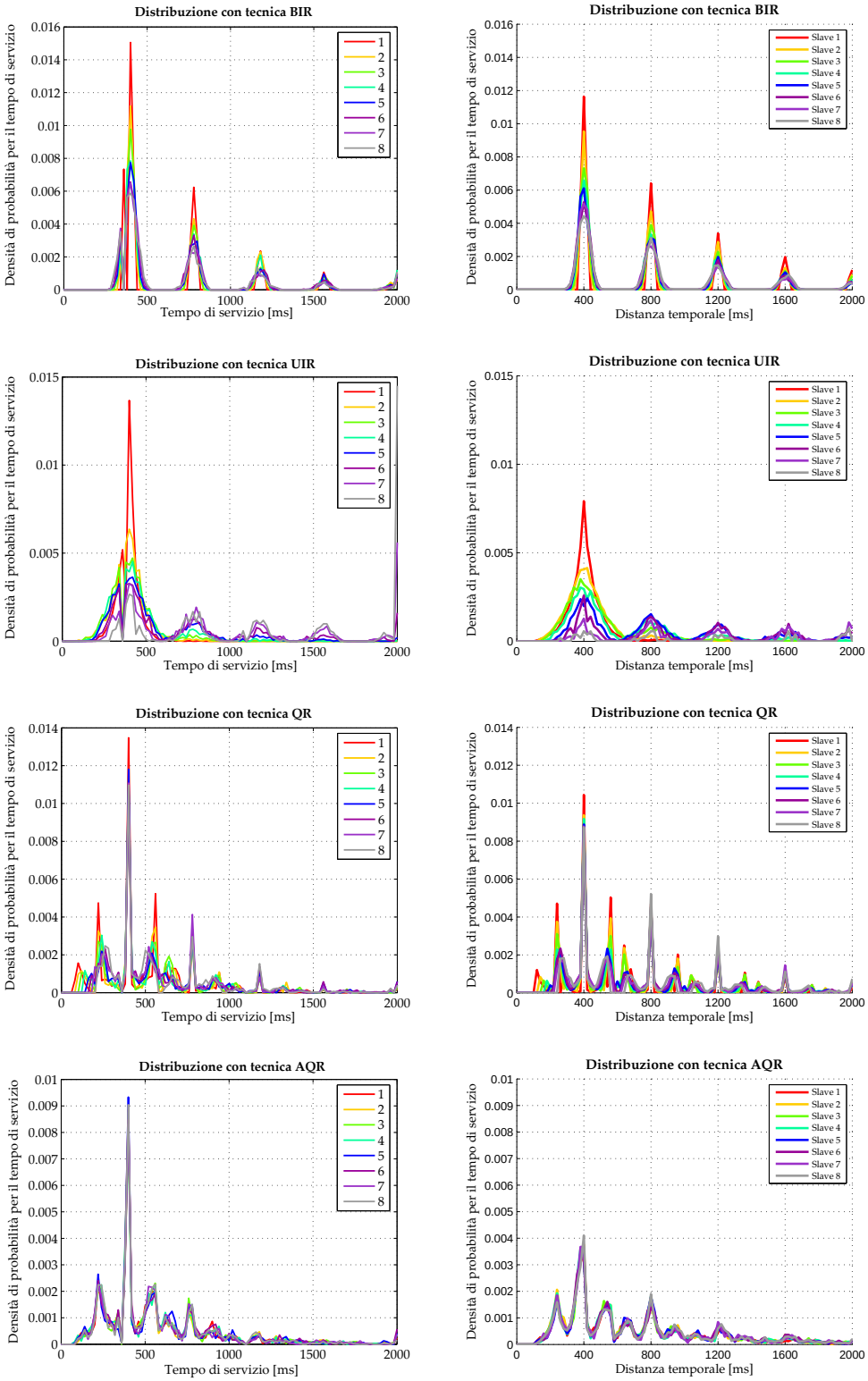


Figura 5.4: Grafici delle densità di probabilità per il tempo di servizio.
A sinistra: grafici di riferimento. A destra: grafici ottenuti con il simulatore in esame.

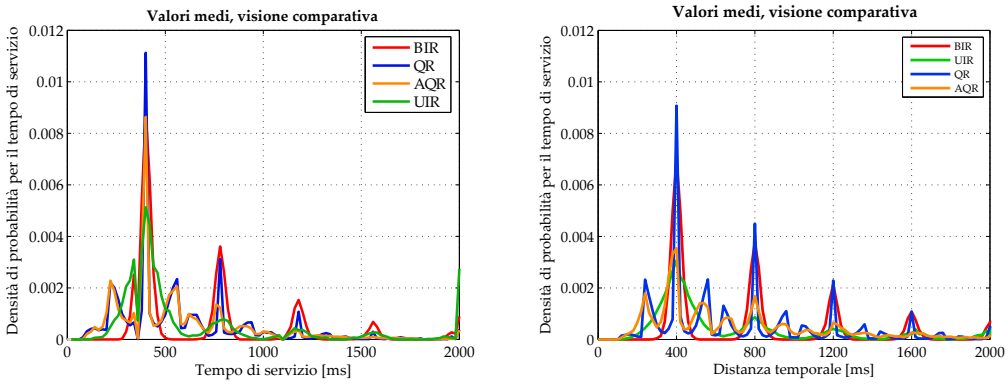


Figura 5.5: Grafici delle densità di probabilità medie vs. tecniche di ritrasmissione.
A sinistra: grafico di riferimento. A destra: grafico ottenuto con il simulatore in esame.

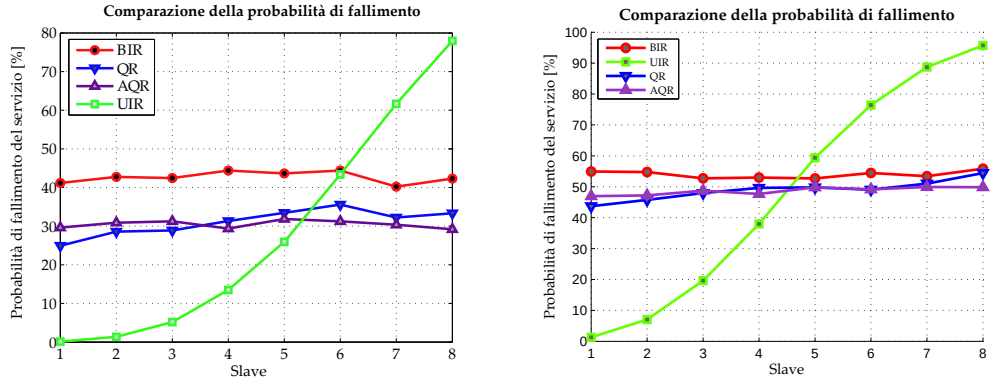


Figura 5.6: Grafico della percentuale di cicli persi - senza successo - per ogni nodo.
A sinistra: grafico di riferimento. A destra: grafico ottenuto con il simulatore in esame.

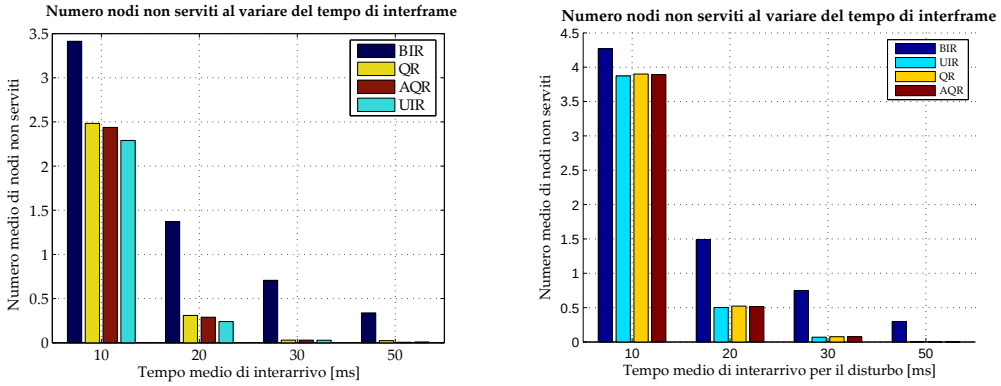


Figura 5.7: Grafico del numero medio di nodi serviti.
Vengono fatti variare sia la tecnica di ritrasmissione sia il tempo di inter-arrivo del segnale interferente.
A sinistra: grafico di riferimento. A destra: grafico ottenuto con il simulatore in esame.

desiderato, riportando dei valori numerici che si possono considerare confrontabili con quelli richiesti.

I grafici della tecnica BIR riportano la distribuzione corretta, evidenziando delle zone con probabilità elevata in corrispondenza dei tempi multipli della durata di un ciclo (400 ms) intervallate da zone a probabilità nulla. In entrambi i grafici è possibile osservare che lo Slave 1 ha una densità di probabilità dalla caratteristica più acuta, concentrata su intervalli più brevi e con probabilità maggiori, e che questa ripidezza decade leggermente all'aumentare del numero dello slave. É comunque possibile individuare una differenza sull'altezza del primo picco, a sfavore della versione simulata, che si fa meno marcata nei picchi di ordine superiore.

Per i grafici della tecnica UIR è facilmente possibile individuare le stesse creste, meno ripide che nel caso precedente, che riproducono correttamente il grafico di riferimento con una differenza marcata, come nel caso precedente, per il valore di probabilità del primo picco, in cui è possibile stimare una differenza superiore al 30%, sempre a sfavore della versione simulata.

Un discorso analogo vale anche per i grafici relativi alla tecnica QR, dove l'andamento generale è ampiamente rispettato, con la presenza di picchi di probabilità a intervalli multipli del tempo di ciclo, ma è osservabile una riduzione delle ampiezze per i primi picchi rispetto a quanto atteso.

Per la tecnica AQR la differenza con il grafico di riferimento è ancor più marcata, soprattutto a causa della mancanza del picco principale, uguale per tutti gli slave, in corrispondenza della prima distanza di ciclo.

5.4.2 Analisi del tempo medio di servizio

Per quanto riguarda il grafico in Figura 5.5 si può affermare che i risultati trovati sono soddisfacenti. La funzione di media, eseguita a partire dai tempi di servizio singoli riduce le disuguaglianze osservate nei grafici precedenti, permettendo di ottenere risultati aderenti al modello. Ancora una volta l'ampiezza del primo picco si presenta attenuato per tutte le curve disegnate, ma l'entità dell'effetto è decisamente ridotta rispetto ai casi precedenti.

5.4.3 Analisi della probabilità di fallimento

La Figura 5.6, che riporta le probabilità di fallimento dei vari slave, mostra un andamento corrispondente a quello previsto: il grafico della tecnica BIR assume sempre valori superiori a quelli delle curve per le tecniche QR e AQR, che tra loro mostrano valori confrontabili, mentre il grafico della tecnica UIR mostra lo stesso andamento crescente che si può osservare anche nei grafici di riferimento. Quello che cambia, tuttavia, è sempre il fattore di scala, che rende i grafici ottenuti dalla simulazione, peggiorativi rispetto a quelli

attesi, mostrando per ogni tecnica e per ogni slave, valori di probabilità di fallimento superiori.

5.4.4 Analisi del numero medio di nodi non serviti

Il grafico del numero medio di nodi non serviti, in Figura 5.7, presenta le caratteristiche osservabili nel grafico di riferimento. In tutte e quattro le situazioni di osservazione, in cui è stato fatto variare il tempo medio di interarrivo dell'interferenza, la strategia BIR presenta sempre il maggior numero medio di nodi non serviti, seguita a distanza dalle altre tecniche, che offrono risultati paragonabili. Ancora una volta, la stima sui valori è peggiorativa rispetto a quanto realmente misurato: in particolare, nel primo caso, con tempo di interarrivo medio pari a 10 ms si possono osservare errori nell'ordine del 30% per la BIR e nell'ordine del 70% per le altre tecniche. Tuttavia, la situazione si normalizza considerevolmente nel caso in cui si utilizzino tempi di interarrivo più prolungati, arrivando a valori assolutamente accettabili.

5.4.5 Considerazioni

Da un'analisi più approfondita è stato possibile osservare degli scostamenti tra i valori numerici attesi e quelli calcolati; queste differenze possono essere ricondotte a uno o più fattori, che verranno ora brevemente illustrati. Un primo fattore che può influire sulla correttezza numerica dei valori riscontrati è sicuramente il regime di non idealità in cui l'esperimento reale è stato condotto: questo, infatti, è stato eseguito in un ambiente reale, con dei limiti fisici che producono una variazione del comportamento in trasmissione e in ricezione rispetto a quello simulato. Il più rilevante tra questi è certamente il riverbero prodotto dai percorsi multipli compiuti dal segnale per transitare dal master allo slave (e viceversa), un fattore che degrada le prestazioni e che in questa sede non è stato considerato.

Un secondo fattore può essere ricercato nella modellazione del modulo *interference*, sede centrale per l'analisi congiunta delle due reti, che identifica l'interferenza come sovrapposizione temporale tra due invii, limitandosi alla distruzione dei pacchetti che presentino coesistenza temporale. Questo modello, sebbene sufficiente in un'analisi preliminare, necessita di essere ulteriormente raffinato per aumentare la precisione dei risultati ottenuti.

Un terzo fattore può essere nascosto nella caratterizzazione delle funzioni utilizzate per il calcolo del Packet Error Ratio (PER) a partire dal rapporto segnale/interferente. Come illustrato in [1] e in [13], la valutazione del PER partendo dai livelli di Signal to Interference Ratio (SIR) avviene con l'uso di una formula:

$$BER = \frac{8}{15}\frac{1}{16}\sum_{k=2}^{16}(-1)^k\binom{16}{k}e^{20\,\mathrm{SIR}(\frac{1}{k}-1)}$$

che, a causa della sua complessità, viene solitamente eseguita in forma approssimata per ridurre il costo computazionale necessario, e diventa quindi una funzione nella forma:

$$f(t) = \frac{1}{1 + e^{a(t-b)}}$$

Come si può osservare, questa funzione è a sua volta legata a due parametri, `a` e `b`, che vengono determinati su base sperimentale a partire dalle prestazioni dei nodi della rete. In questo caso particolare sono stati utilizzati:

$$a = 5.59 \qquad b = 3.16$$

ma, alla luce di ulteriori confronti potrebbe essere necessario aggiustare questi valori per ottenere un comportamento più aderente a quello previsto.

In generale è possibile affermare che se, a parità di condizioni di rete, i dati simulativi sono peggiori di quelli misurati, la curva che modella il PER è "*pessimistica*" rispetto a quanto succede in dispositivi reali e quella che si ottiene è una stima cautelativa. Dall'osservazione dei grafici e dalle considerazioni illustrate si può facilmente notare che i dati simulati sono peggiori di quelli reali ovvero, a parità di condizioni, indicano una probabilità di successo inferiore a quella sperimentalmente rilevata: siamo quindi in presenza di una condizione di sicurezza che consente di fare stime sempre attendibili poiché sovrastimate.

5.5 Possibili migliorie

Le imperfezioni notate offrono lo spunto per l'introduzione di migliorie al lavoro svolto. Come osservato, il modello utilizzato per valutare l'entità dell'interferenza è attualmente basato sulla sola coesistenza temporale dei due pacchetti e potrebbe essere ulteriormente affinato; ad esempio si potrebbe considerare la possibilità di trasmettere comunque i pacchetti che hanno subito danneggiamenti parziali, demandando alle fasi successive di elaborazione dell'informazione il compito di determinare se il pacchetto può essere ricostruito o meno. Un'ulteriore miglioria apportabile potrebbe essere l'implementazione di altre tecniche di ritrasmissione, operazione semplificata dalla modularità con cui sono state implementate le funzioni di trasmissione, ritrasmissione e reinizializzazione alla fine di ogni ciclo. Si potrebbe, inoltre, implementare la gestione dei *beacon frame* che, all'attuale stato di realizzazione non sono stati considerati. In ogni caso, per quanto si tenti di avvicinare la simulazione alla sua realizzazione pratica, esistono sempre delle imperfezioni che rendono la simulazione un valido strumento di indagine limitatamente nel caso in cui si cerchino soluzioni approssimate.

Conclusioni

In questa tesi è stato trattato il problema dell'interferenza tra reti di sensori senza fili, basate sul protocollo IEEE 802.15.4 e reti wireless locali che facciano riferimento alla famiglia di protocolli IEEE 802.11. In particolare si è voluto esaminare il comportamento di queste WSN in ambienti industriali, in cui lo schema utilizzato è il *polling*, con un nodo centrale — master — che richiede e raccoglie le informazioni dai nodi sensori — slave. Con riferimento all'articolo "*Retransmission strategies for cyclic polling over wireless channels in the presence of interference*" [2], si è voluto implementare un simulatore in grado di mostrare il comportamento di questa topologia al variare della tecnica di ritrasmissione utilizzata. Per realizzare queste osservazioni si è utilizzato il simulatore ad eventi OMNeT++, su cui era già disponibile un'implementazione base dei due protocolli necessari, oltre che un modulo per la valutazione congiunta degli effetti di interferenza. Da questo punto di partenza, sono stati realizzati dei nuovi moduli che permettono l'implementazione delle varie tecniche di ritrasmissione di interesse, secondo le indicazioni dell'articolo, ma che lasciano spazio anche a future implementazioni di tecniche alternative. Infine, per valutare il lavoro compiuto, si è scelto di riprodurre al simulatore alcuni dei grafici presenti nell'articolo di riferimento. In particolare si sono voluti osservare quelli riguardanti le prestazioni della rete dal punto di vista della gestione temporale — distribuzione di probabilità dei tempi di servizio dei nodi — e dal punto di vista dell'affidabilità complessiva — numero medio di cicli falliti e numero medio di nodi non serviti. Dal confronto dei risultati è stato possibile trarre una serie di considerazione e di valutazioni del lavoro svolto. É stato possibile constatare un'ottima corrispondenza tra i grafici attesi e quelli ottenuti, il che sta ad indicare che l'implementazione realizzata è conforme a quanto richiesto. Tuttavia sono emerse anche alcune incongruenze, in particolar modo relative ai fattori di scala ottenuti. Quello che si è osservato è che, a parità di condizioni, i grafici ottenuti forniscono una visione peggiorativa di quella reale, indicando ad esempio, probabilità di accesso più basse o sovrastimando il numero medio di nodi non serviti.

In ogni caso, proprio poiché queste stime sono "peggiorative", i risultati sono accettabili e identificano il caso peggiore, su cui eventuali dimensionamenti, forniscono sempre un valore maggiorato e cautelativo.

Acronimi

ACK Acknowledgement
Tipo di frame di risposta usato per indicare una ricezione corretta.

AP Access Point
Dispositivo che fornisce agli altri nodi della rete, l'accesso ai servizi di distribuzione.

AQR Adaptive Queued Retransmission
Tecnica di ritrasmissione basta su una coda prioritaria.

BIR Boundled Immediate Retransmission
Tecnica di ritrasmissione che prevede un numero limitato di tentativi immediati.

CCA Clear Channel Assessment
Servizio che determina l'occupazione del canale.

CSMA-CA Carrier Sense Multiple Access with Collision Avoidance
Algoritmo che consente l'accesso al mezzo da più dispositivi evitando la collisione di pacchetti.

ED Energy Detection
Servizio di rilevamento dell'energia sul canale.

FFD Full-function Device
Dispositivo della rete con insieme di funzioni completo.

ISM Industrial Scientific and Medical
Banda delle frequenze non riservate ed utilizzabili liberamente per scopi non commerciali, medici, industriali o scientifici.

LQI Link Quality Indication
Servizio per determinare la qualità di collegamento.

MAC Medium Access Control
Sottosezione del penultimo layer dello stack OSI.

NED Network Description Language
Linguaggio utilizzato per la descrizione topologica dei nodi.

NIC Network Interface Controller
Hardware che implementa i livelli PHY e MAC dello stack ISO/OSI.

O-QPSK Offset - Quadrature Phase-Shift Keying
Modulazione di fase con le componenti in quadratura e ritardo tra toro.

OSI Open Systems Interconnection
Modello standard per le reti di calcolatori.

PAN Personal Area Network
Rete che permette la comunicazione tra dispositivi vicini ad un singolo utente, con raggio d'azione di pochi metri.

PER Packet Error Ratio
Incidenza del numero di pacchetti corrotti sul numero di pacchetti ricevuti totali

PHY Physical (riferito a Physical Layer)
Il layer più basso dello stack OSI.

PIB PAN Information Base
Locazione in cui sono contenute le informazioni sulle impostazioni di un nodo.

PPDU Packet Protocol Data Unit
Pacchetto finale trasmesso e ricevuto a livello fisico.

QR Queued Retransmission
Tecnica di ritrasmissione che organizza i nodi in una coda.

RFD Reduced-function Device
Dispositivo della rete a ridotto insieme di funzioni.

SAP Service Access Point
Punto del modulo tramite cui viene fornito un servizio al livello adiacente.

SIR Signal to Interference Ratio
Rapporto di potenze tra il segnale utile e il segnale interferente

UIR Unboundled Immediate Retransmission
Tecnica di ritrasmissione che prevede un numero illimitato di tentativi.

WLAN Wireless Local Area Network
Rete locale senza fili.

WSN Wireless Sensor Network
Reti di sensori senza fili.

Bibliografia

[1] F. Tramarin, "Cross-layer analysis of interfering wireless sensors networks," Tesi di Laurea Magistrale in Ingegneria Elettronica, Università degli Studi di Padova, Dipartimento di Ingegneria dell'Informazione, 2008.

[2] G. Gamba, F. Tramarin, and A. Willig, "Retransmission strategies for cyclic polling over wireless channels in the presence of interference," *Industrial Informatics, IEEE Transactions on*, vol. 6, pp. 405 –415, ago 2010.

[3] C. Reinisch, W. Kastner, G. Neugschwandtner, and W. Granzer, "Wireless technologies in home and building automation," in *Proc. 5th IEEE Int Industrial Informatics Conf*, vol. 1, pp. 93–98, 2007.

[4] M. A. B. Sarijari, R. A. Rashid, M. R. A. Rahim, and N. H. Mahalin, "Wireless home security and automation system utilizing zigbee based multi-hop communication," in *Proc. Telecommunication Technologies 2008 and 2008 2nd Malaysia Conf. Photonics. NCTT-MCP 2008. 6th National Conf*, pp. 242–245, 2008.

[5] S. Nourizadeh, C. Deroussent, Y. Q. Song, and J. P. Thomesse, "Medical and home automation sensor networks for senior citizens telehomecare," in *Proc. IEEE Int. Conf. Communications Workshops ICC Workshops 2009*, pp. 1–5, 2009.

[6] G. Gamba, L. Seno, and S. Vitturi, "Theoretical and experimental evaluation of polling times for wireless industrial networks using commercially available components," in *Proc. IEEE Conf. Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2010.

[7] M. Bertocco, G. Gamba, A. Sona, and F. Tramarin, "Investigating wireless networks coexistence issues through an interference aware simulator," in *Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation ETFA 2008*, pp. 1153–1156, 2008.

[8] M. Bertocco, A. Sona, and F. Tramarin, "Design of experiments for the assessment of coexistence between wireless networks," in *Proc. IEEE Instrumentation and Measurement Technology Conf. (I2MTC)*, pp. 928–932, 2010.

[9] "Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). IEEE Std 802.15.4-2006," 2006.

[10] H. S. Sang and J. C. Soon, "Method for transmitting wireless data using piggyback," 2007.

[11] G. Gamba, F. Tramarin, and A. Willig, "Retransmission strategies for cyclic polling over wireless channels in the presence of interference," in *Proc. IEEE Conf. Emerging Technologies & Factory Automation ETFA 2009*, pp. 1–8, 2009.

[12] A. Varga and OpenSim, *Omnet Maual, 4.1*.

[13] IEEE, "Ieee recommended practice for information technology telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 15.2: coexistence of wireless personal area networks with other wireless devices operating in unlicensed frequency band," 2003.