Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Reti di sensori e attuatori per progetti M2M per l'efficientamento dei processi industriali

G. Campobello, N. Donato, R. Guida, A. Segreto, M.A. Segreto, S. Serrano

Report RdS/PAR2015/073

RETI DI SENSORI E ATTUATORI PER PROGETTI M2M PER L'EFFICENTAMENTO DI PROCESSI INDUSTRIALI

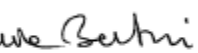
G. Campobello[1], N. Donato[1], R. Guida[2], A. Segreto[1], M.A. Segreto[2], S. Serrano[1]
[1]Dipartimento di Ingegneria, Università degli Studi di Messina
[2]Centro Ricerche ENEA di Bologna

Settembre 2016

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2015
Area: "Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici"
Progetto: "D.3 Efficienza energetica nel settore industria"
Obiettivo: e.1 "Rete di sensori e attuatori per progetti M2M"

Responsabile del Progetto: Ing. Ilaria Bertini, ENEA



Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "*Reti di sensori e attuatori per progetti M2M"*
Responsabile scientifico ENEA: Ing. Maria-Anna Segreto, ENEA

Responsabile scientifico Dipartimento di Ingegneria: Prof. Ing. Giuseppe Campobello, UniME

Ringraziamenti.
Si ringraziano gli studenti, i dottorandi e i borsisti che hanno collaborato alla realizzazione del progetto e la ditta WEVA per la collaborazione fornita nello sviluppo del software.

Indice

Sommario

I temi sviluppati nel presente documento illustrano le attività svolte nell'ambito dell'accordo di collaborazione tra ENEA e il Dipartimento di Ingegneria dell'Università di Messina avente come obiettivo la progettazione di una rete di sensori e attuatori wireless basata su protocolli machine-to-machine (M2M) e finalizzata all'efficientamento energetico del settore industria.

Nell'ambito del progetto, al fine di definire le specifiche del sistema, sono stati dapprima analizzati gli scenari applicativi di maggiore interesse per il progetto (smart metering, smart grid e smart industries). Successivamente è stato analizzato lo stato dell'arte delle reti di sensori wireless con l'obiettivo di definire una architettura di riferimento per le applicazioni suddette individuando l'hardware, i protocolli di comunicazione e i software di sviluppo maggiormente idonei per gli scopi del progetto. In particolare sono stati analizzati in dettaglio gli aspetti inerenti i consumi energetici e la sicurezza nelle reti di sensori ed è stata definita una architettura interamente basata su protocolli aperti (COAP, IPv6, 6LowPAN) e software open source (TinyOS, MySQL). È stato poi sviluppato un modello analitico in grado di guidare le scelte progettuali ed è stato realizzato un testbed mediante il quale sono stati validati i risultati previsti dal modello. Le misure sperimentali hanno confermato la possibilità di applicare le reti di sensori in ambienti industriali e la loro efficacia in applicazioni di smart metering. Infine è stato realizzato un software che semplifica il setup e la gestione della rete di sensori, permette la memorizzazione dei dati acquisiti su database e fornisce un front-end per la visualizzazione e l'elaborazione.

1 Introduzione

Una rete di sensori wireless (Wireless Sensor Network, WSN) è composta da una moltitudine di dispositivi, detti *nodi sensori*, capaci di interagire con l'ambiente circostante e di comunicare tra loro al fine ultimo di monitorare e/o controllare un fenomeno fisico [1].

In genere i dati acquisiti dai nodi sensori sono inviati ad un punto di raccolta, detto *Gateway* (o anche *Base Station* o *Sink)*, il quale può interfacciarsi ad Internet e rendere così i dati facilmente accessibili agli utenti come illustrato in Figura 1. Il Sink, oltre a ricevere le informazioni provenienti dai nodi sensori, fornisce anche servizi per la memorizzazione e visualizzazione dei dati, per la gestione della rete e per la configurazione/programmazione dei nodi (tali servizi sono in genere fruibili solo dagli amministratori e/o ad utenti autorizzati).

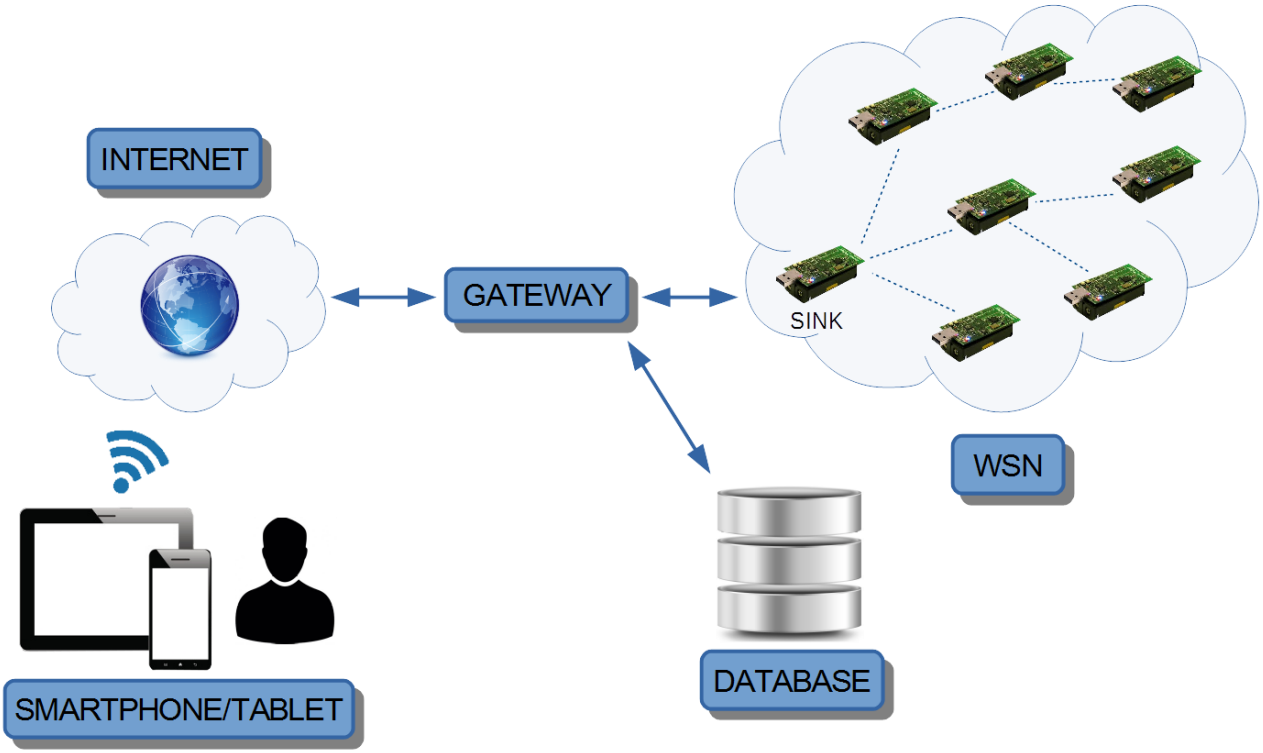


Figura 1 Architettura di una WSN.

Le tecnologie oggi disponibili per la realizzazione di reti di sensori sono ampiamente mature e diffuse e diverse piattaforme sono disponibili sul mercato ed accessibili a prezzi contenuti. In particolare esiste un'ampia scelta di sensori, dai più semplici e a basso costo come i sensori di temperatura, umidità e luminosità, fino a sensori estremamente sofisticati che includono GPS o reagenti chimici in grado di rilevare la presenza di specifici inquinanti o agenti biologici. Ciò permette l'utilizzo delle WSN in svariati scenari applicativi come ad esempio:

- monitoraggio ambientale;
- smart metering;
- controlli industriali;
- agricoltura di precisione;
- monitoraggio strutturale;
- dispositivi elettromedicali;
- sistemi di sorveglianza;
- applicazioni militari.

L'elevato numero di contesti applicativi deriva altresì dai numerosi vantaggi delle WSN rispetto ai sistemi convenzionali di monitoraggio e controllo via cavo e in particolare:

- bassi costi di installazione e manutenzione (le WSN permettono infatti di coprire aree anche molto estese con costi di installazione praticamente nulli);
- elevata flessibilità (derivante dalla facilità con cui è possibile aggiungere e/o riposizionare i nodi).

Nell'ambito dell'efficientamento industriale, in particolare, le WSN possono rivestire un ruolo molto importante attraverso la realizzazione di reti di *smart metering* quale strumento per la misurazione dei risparmi conseguibili a seguito di interventi di efficientamento, consentendo la valutazione dei consumi e delle dispersioni di energia di un singolo impianto o di un intero compendio immobiliare, sia prima dell'intervento di riqualificazione che durante la sua realizzazione, permettendo altresì la misurazione in tempo-reale dei risparmi conseguiti e il controllo in tele-gestione dei consumi post intervento.
Inoltre il monitoraggio in tempo reale dei consumi di luce, gas e acqua con l'ausilio di WSN consentirebbe di intervenire sugli impianti regolando lo scambio sia di energia sia di informazioni sul funzionamento, offrendo anche la possibilità di intervenire tempestivamente e da remoto in caso di problematiche o guasti, senza dover ricorrere all'intervento sul posto.
È quindi possibile ed auspicabile un largo impiego delle WSN ad ogni livello della rete di distribuzione (dalla centrale, alla rete intelligente, fino alla singola unità) al fine ultimo di valutare i consumi energetici e, alla luce dei risultati riscontrati, programmare interventi di efficientamento e verificarne l'efficacia.
Altro aspetto importante riguarda la possibilità di utilizzare le WSN per il controllo degli impianti di riscaldamento, ventilazione e illuminazione al fine di una gestione che riesca a trovare il giusto compromesso tra servizio fornito, comfort, costi e consumi energetici.
Mediante le WSN, e più in generale il nuovo paradigma dell'*Internet of Things* (*IoT*) [2], è quindi possibile arrivare a definire un ambiente completamente automatizzato in cui le decisioni mirate all'efficientamento energetico siano prese direttamente dai dispositivi. Ciò è oggi possibile grazie al fatto che le reti di sensori sono in grado di interagire non solo con gli utenti ma anche tra di loro attraverso opportuni protocolli, noti come protocolli Machine-to-Machine (M2M o D2D, Device-to-Device). Ovviamente in tale scenario risulta indispensabile affrontare e risolvere diversi problemi, legati alle limitate risorse disponibili sui dispositivi e alla loro eterogeneità, al fine ultimo di garantire prestazioni adeguate (in termini di velocità di trasferimento dati, ritardi, consumi, sicurezza ecc.), l'interoperabilità con altre reti (in particolare con reti WiFi e Internet) oltre che semplicità di utilizzo.
La risoluzione di tali problematiche non è semplice, infatti a fronte dello scenario suddetto occorre rilevare che ancora oggi gran parte dei sensori in ambiente industriale è ad ispezione visiva e i sistemi di controllo sono spesso basati su una configurazione manuale che non sempre risulta essere ottimale (si pensi al controllo dell'illuminazione e/o del riscaldamento negli edifici industriali).
A ciò si aggiunge il fatto che gli studi presenti in letteratura spesso mirano a risolvere una singola problematica (sicurezza/prestazioni/interoperabilità) e non affrontano il problema dell'integrazione delle WSN in ambito industriale nella sua globalità, non essendo così in grado di fornire una soluzione che includa tutti i livelli architetturali, partendo dall'hardware e finendo dall'interfaccia utente; viceversa le soluzioni commerciali esistenti, pur fornendo una soluzione, sono spesso basate su hardware e software proprietari e pertanto non sono interoperabili.
Il presente elaborato si inquadra in tale contesto con l'obiettivo di definire una architettura di riferimento per le applicazioni suddette, interamente basata su protocolli aperti e software open source, individuando l'hardware, i protocolli di comunicazione e i software di sviluppo maggiormente idonei e analizzando in dettaglio i parametri che ne determinano le prestazioni.
Il documento è articolato come segue.
Nel Capitolo 2 sono analizzati i principali scenari applicativi di interesse per il progetto (smart metering, smart grid e smart industries) al fine della **definizione delle specifiche** architetturali e prestazionali.
Nel Capitolo 3 è analizzato lo stato dell'arte dei **protocolli M2M** al fine di individuare i protocolli M2M che maggiormente si adattano alle limitate risorse computazionali dei nodi sensori nelle WSN.
Nel Capitolo 4 sono analizzate le **piattaforme per WSN** presenti sul mercato al fine di selezionare quelle più adatte ad applicazioni di monitoraggio ambientale e controllo remoto finalizzate al risparmio energetico ed all'efficientamento di processi industriali, oltre che per il miglioramento del comfort abitativo. In particolare

nel capitolo sono riportati diversi confronti tra le piattaforme commerciali esistenti in termini di costi, funzionalità e prestazioni.
Nel Capitolo 5 verranno esaminati i **protocolli di comunicazione** specifici per WSN e sarà illustrato lo stack protocollare dell'architettura proposta.
Nel Capitolo 6 si analizzerà lo stato dell'arte dei **sistemi operativi** per WSN, ne verranno esaminate le principali funzionalità con riferimento alla gestione delle risorse ed alla minimizzazione dei consumi energetici.
Nel Capitolo 7 si illustreranno gli aspetti legati alla **sicurezza** delle WSN e le possibili soluzioni implementative specifiche per WSN.
Nel Capitolo 8 verranno illustrate le principali **tecniche di minimizzazione dei consumi energetici** (duty cycling, data aggregation, ecc.) atte a massimizzare il tempo di operatività dei nodi sensori.
Nel Capitolo 9 sarà fornito un **modello analitico** in grado di guidare le scelte progettuali e che permette di valutare l'effetto dei parametri di configurazione sulle prestazioni della rete in termini di efficienza energetica, throughput, probabilità di perdita e ritardi.
Nel Capitolo 10 sarà presentata l'architettura delineata e il **testbed** realizzato mediante il quale sono state effettuate diverse misure sperimentali che hanno permesso di validare il modello analitico sviluppato, oltre che confermare la possibilità di applicare le reti di sensori in ambienti industriali e la loro efficacia in applicazioni di smart metering. È inoltre presentato un **software di gestione** appositamente sviluppato per semplificare il setup e la configurazione delle rete di sensori e per permettere la memorizzazione automatica dei dati acquisiti su database fornendo un front-end per la visualizzazione e l'elaborazione dei dati.
Nel Capitolo 11 saranno esposte le conclusioni e riassunti i risultati raggiunti oltre che identificate alcune possibili attività future.

2 Analisi dei requisiti e definizione delle specifiche

La progettazione di un sistema può essere vista come un processo di trasformazione che parte dalla ideazione del sistema per arrivare alla sua realizzazione. Diverse sono le metodologie di progettazione definite dall'Ingegneria del Software [3] e dagli standard internazionali per la realizzazione di software di qualità (es. DoD STD2167A [4]) che identificano le fasi per attuare tale processo (si veda ad esempio la Figura 2) ma per tutte l'analisi dei requisiti e la definizione delle specifiche è il primo e forse il più importante passo della progettazione. Tale fase del processo di progettazione deve essere indipendentemente dalle metodologie o dai tool che verranno adottati per lo sviluppo del sistema stesso e deve prendere in considerazione gli scenari applicativi ai quali il sistema è rivolto.

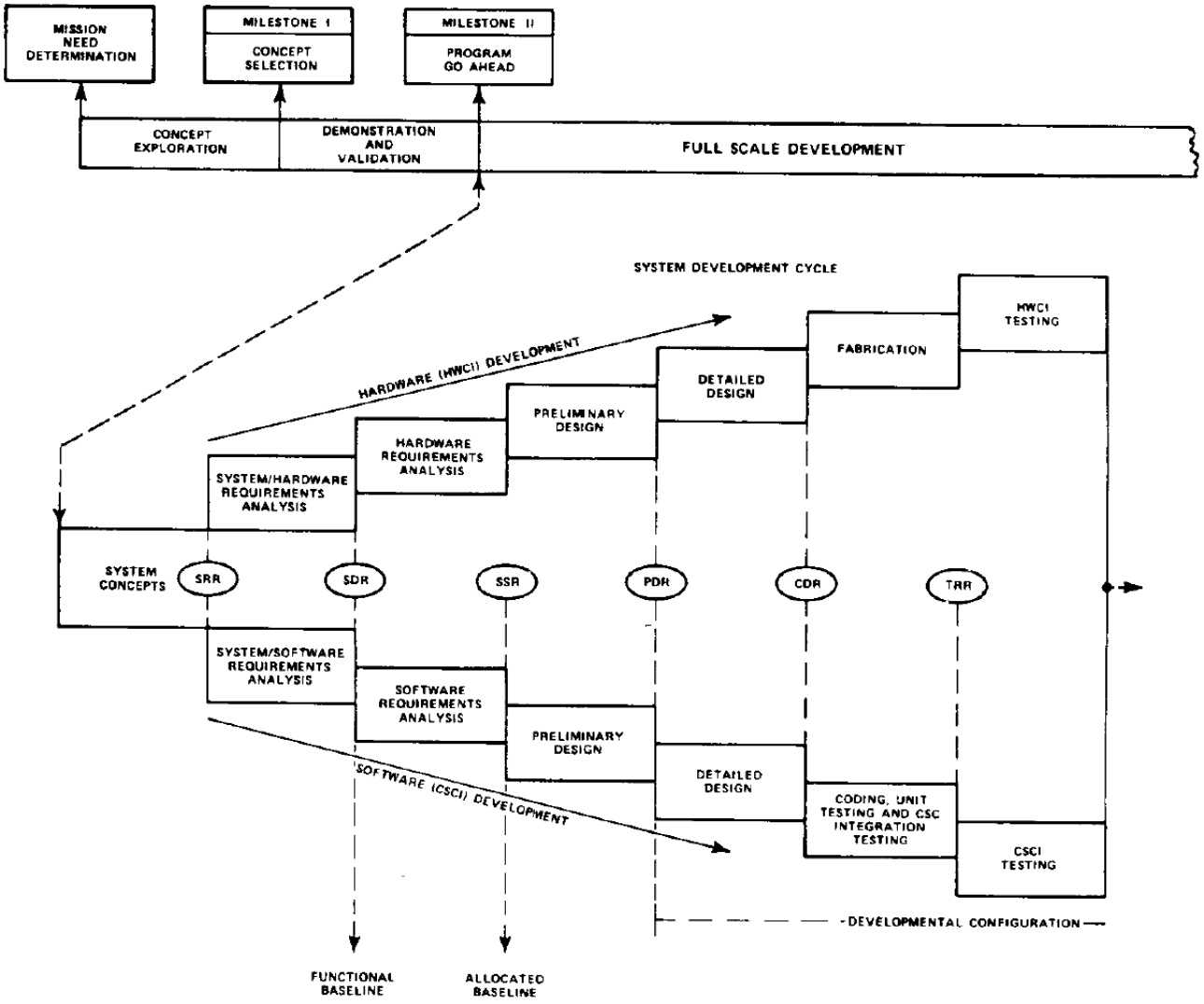


Figura 2 Fasi di progettazione [4].

Nell'ambito dell'attività al fine di definire le specifiche sono stati quindi analizzati i principali scenari applicativi di interesse per il progetto MiSE-ENEA e in particolare:

- i sistemi di automazione industriale (smart industries);
- le reti di distribuzione dell'energia (smart grid);
- i sistemi di misurazione intelligenti (smart meters).

Di seguito è riportata una sintesi dell'analisi condotta e che ha guidato le attività di ricerca e sviluppo realizzate nell'ambito del progetto MiSE-ENEA.

2.1 Requisiti per l'Automazione Industriale

Obiettivo principale dell'automazione industriale è quello di accrescere il numero di risorse interconnesse (sensori, attuatori, basi di dati, ecc.) di modo che sistemi distribuiti e macchine intelligenti possano migliorare la visibilità e quindi la gestione del processo produttivo [5].

Oggi ciò è reso possibile grazie alla riduzione dei costi di CPU e memorie e più in generale dei sistemi embedded che permettono la realizzazione di macchine intelligenti a basso costo; inoltre la riduzione delle dimensioni e dei consumi energetici dei sistemi suddetti permette di dislocare un maggior numero di dispositivi per unità di area.

Per definire tale rivoluzione sono stati coniati i termini Industrial Internet of Things (IIoT) e Industry 4.0, nomi alternativi dati al paradigma dell'Internet of Things [6] quando specificatamente rivolti ai processi industriali. In particolare riduzione dei costi, riduzione del time-to-market, tracciabilità di merci e prodotti, gestione in tempo reale dei processi, integrazione di nuove tecnologie (pur mantenendo la compatibilità con le tecnologie e i sistemi esistenti), aumento della produttività e della qualità dei prodotti, maggiore tolleranza ai guasti, riduzione dell'impatto ambientale (in termini di consumi energetici e di scarti di produzione) e un maggiore livello di sicurezza sono alcuni dei risultati attesi dall'introduzione di questo paradigma nel processo produttivo.

Tutto ciò richiede però la disponibilità di dispositivi e macchine più efficienti e flessibili e soprattutto interconnesse oltre che nuovi sistemi di gestione distribuiti facilmente fruibili dagli operatori.

È possibile quindi identificare tre "soggetti" coinvolti nei processi di automazione:

- le macchine;
- gli operatori;
- il sistema di supervisione.

Nel nuovo paradigma le macchine non devono più essere dei semplici utensili ma devono avere un certo livello di "intelligenza" ovvero devono essere in grado di monitorare i propri componenti e l'ambiente circostante, devono essere consapevoli delle proprie capacità e devono renderle disponibili, mediante appositi servizi, agli utenti/operatori al fine di permettere a questi ultimi di monitorare e configurare i parametri di funzionamento delle macchine stesse. Tutto ciò è oggi reso possibile grazie all'utilizzo dei protocolli **Machine-to-Machine** (M2M).

Un operatore deve essere in grado di comunicare con la macchina al fine di specificarne i parametri che ne regolano il funzionamento in termini di consumi energetici e tempi dei cicli produttivi. In particolare gli operatori richiedono che i dispositivi siano **plug-and-play** in modo di semplificare e ridurre i tempi per le operazioni di installazione e set-up.

In tale contesto le nuove tecnologie di interconnessione sono fondamentali per fornire un accesso rapido a risorse e dati remoti; in particolare le **tecnologie wireless** permettono l'accesso in tempo reale anche in condizioni di mobilità, oltre che di distribuire l'intelligenza del sistema.

Il sistema di supervisione deve essere distribuito poiché l'instradamento di tutti i dati ad un'unica unità di supervisione e controllo comporta un aumento dei ritardi di trasmissione e limita fortemente la scalabilità del sistema (ovvero il massimo numero di dispositivi gestibili).

Le reti di sensori e attuatori wireless, anche grazie ai bassi costi di installazione e manutenzione, rappresentano lo strumento ideale per conferire al sistema quell'intelligenza, oltre che l'infrastruttura, necessaria per distribuire e automatizzare i processi decisionali.

Anche la scalabilità delle risorse di storage è necessaria in tali contesti e per tale motivo risulta importante la possibilità di avvalersi di tecnologie cloud.

A tal fine occorre affiancare alle WSN l'uso di tecnologie e **protocolli Internet-oriented** possibilmente basati su standard aperti.

Le WSN ed i protocolli Internet (in particolare IPv6) risultano quindi gli elementi fondanti di tale rivoluzione.

In accordo con tale scenario i principali requisiti di sistema nell'ambito dell'automazione industriale sono stati identificati nel progetto europeo FP7 IoT@Work [7], che ha visto fra l'altro la collaborazione di FIAT, Microsoft e Siemens, e sono di seguito elencati:

Flessibilità, scalabilità e riconfigurabilità della rete: deve essere possibile aggiungere dispositivi o moduli software per la creazione di nuove funzionalità e servizi senza limiti sul numero e possibilmente senza alcuna degradazione apprezzabile della qualità del servizio*;* in particolare variazioni topologiche dovute ad aggiunta o rimozione dei dispositivi o a link failure devono essere gestite automaticamente; ciò è possibile mediante appositi algoritmi di routing in grado di gestire topologie di rete di tipo mesh;

Inizializzazione e riconfigurabilità dei nodi: i nodi devono poter essere riprogrammati (almeno offline) e riconfigurati (anche da remoto); in particolare deve essere possibile inizializzare lo stato dei dispositivi. Tali operazioni devono poter essere svolte solo da parte del personale autorizzato*;*

Sicurezza: il sistema deve garantire l'autenticità, l'integrità, la confidenzialità e la validità dei messaggi e permettere di gestire diversi livelli di accesso per le diverse tipologie di utenti (tecnici, operatori, amministratori di rete, contabilità, ecc.). L'accesso remoto deve essere possibile solo agli utenti autorizzati. Un dispositivo deve poter entrare a far parte della rete solo se preventivamente autenticato. I meccanismi di sicurezza dovrebbero essere robusti ma anche leggeri (per non impattare negativamente sulle prestazioni della rete);

Tracciabilità: le operazioni effettuate sul sistema dai vari utenti devono essere tracciate e memorizzate in appositi file di log sia per motivi di sicurezza che di gestione (oltre che per eventuali operazioni di fatturazione);

Affidabilità (reliability): deve essere garantita l'affidabilità del sistema e del trasporto dell'informazione. A tal fine possono essere utilizzate tecniche di controllo di errore (ARQ, FEC) in combinazione con tecniche di routing dinamico;

Disponibilità di tool di gestione: occorre la disponibilità di tool di gestione grafici che semplifichino il set-up e la gestione del sistema; e permettano di configurare dispositivi da remoto, di analizzarne lo stato (es. batteria, qualità delle comunicazioni, ecc.) e/o identificare situazioni di guasto;

Identificazione dei dispositivi: i dispositivi devono avere indirizzi numerici univoci, in particolare si auspica l'utilizzo di indirizzi IPv6 che renderebbero i dispositivi immediatamente accessibili alle tecnologie e ai servizi Internet-based. Rispetto all'utilizzo di indirizzi IPv4 l'uso di indirizzi IPv6 permette una maggiore scalabilità e una più semplice integrazione dei meccanismi di sicurezza. Si ritiene in fine necessaria la possibilità di identificare le risorse anche con indirizzi logici (tipo URI) anziché numerici al fine di semplificare la gestione del sistema da parte degli operatori;

Meccanismi di discovery: Appositi meccanismi devono poter permettere di rilevare un nodo e ottenere informazioni sui servizi da esso offerti*;*

Duty Cycling: il sistema deve prevedere tecniche di duty-cycling per la gestione di dispositivi in modalità stand-by al fine di ridurre i consumi energetici.

Definizione della Qualità del Servizio (QoS): deve essere prevista la possibilità di gestire e/o garantire la qualità del servizio in dipendenza del tipo di applicazione. In particolare devono essere monitorati ritardo e perdita dei pacchetti (o più in generale latenza e affidabilità) e per le applicazioni real-time anche la variazione del ritardo (jitter);

Nell'analisi della Qualità di Servizio di una rete diverse sono le metriche che è possibile prendere in considerazione; in genere se una metrica sia o meno più importante di un'altra dipende dallo scenario applicativo. Di seguito quindi ci si limita ad elencarle (in ordine sparso e non per importanza) e a chiarirne il significato.

Efficienza energetica: l'efficienza energetica è il parametro di progettazione principale nelle WSN per almeno tre motivi: 1) per massimizzare il tempo di operatività (lifetime) dei dispositivi dato che molti dispositivi sono alimentati a batterie; 2) per ridurre i costi energetici e di gestione (ricaricare una batteria ha infatti un costo sia in termini energetici che di tempo); 3) per ridurre l'impatto ambientale, già oggi le industrie ICT sono infatti responsabili dell'10% dei consumi mondiali [8] e tale percentuale è destinata a crescere. In italia, nel 2011, la Telecom Italia [9] stimò che il consumo energetico relativo alle sue attività rappresenta 1% del totale dei consumi in Italia [8].

Vista l'importanza dell'argomento le tecniche per ridurre i consumi energetici saranno analizzate nel Capitolo 8.

Throughput: il throughput identifica la quantità di dati trasferita nell'unità di tempo; questo dipende da diversi fattori e in particolare, nel caso di trasmissioni wireless, dal numero di utenti/dispositivi che condividono il canale. In genere su canale wireless sono le collisioni (ovvero la possibilità di trasmissioni contemporanee da parte di più dispositivi) a limitare il throughput. Minimizzare il numero di collisioni è spesso l'obiettivo principale dei protocolli di accesso al mezzo (MAC)

Latenza: per le applicazioni di controllo e automazione industriale è fondamentale minimizzare il tempo trascorso fra la generazione di un comando (in generale di un evento) e la sua attuazione (ovvero la risposta del sistema all'evento stesso); buona parte di tale intervallo temporale è legato alla latenza introdotta dalla rete a causa delle collisioni e dei tempi di elaborazione e memorizzazione (la propagazione del segnale avviene in genere a velocità prossime a quelle della luce ed è quindi trascurabile per distanze anche dell'ordine dei chilometri). Tale latenza aumenta con l'aumentare del numero di dispositivi ma meccanismi di gestione della priorità possono ridurre tale problema.

Affidabilità/Reliability: l'affidabilità intesa come probabilità che le informazioni raggiungano correttamente la destinazione indipendentemente dalle condizioni del canale e dalla possibilità di guasti nella rete, è un altro aspetto importante delle WSN. Soprattutto nei sistemi di controllo e automazione la perdita di pacchetti può non essere tollerabile (specie se lo scambio di tali pacchetti è funzionale alla gestione di condizioni di allarme). Anche in scenari non intrinsecamente real-time, come ad esempio le applicazioni di monitoraggio ambientale, una perdita di pacchetti eccessiva può costituire un serio problema potendo compromettere la stessa funzionalità primaria della rete (ovvero la capacità di osservare un fenomeno o una grandezza fisica).

Nel presente documento sono analizzati i parametri che maggiormente influenzano le metriche suddette; in particolare un modello analitico in grado di determinare throughput, latenza e affidabilità della rete in funzione dei parametri di configurazione è proposto nel Capitolo 10.

Interoperabilità: le reti WSN e i protocolli M2M utilizzano molteplici standard di comunicazione (ZigBee, BLE, WiFi,ecc.) spesso non interoperabili ma che operano sostanzialmente nella stessa porzione di spettro. Una corretta condivisione dello spettro unita a tecniche per ridurre le interferenze risultano quindi fondamentali per evitare la rapida degradazione delle prestazioni. Una analisi dei protocolli per WSN verrà affrontata nel Capitolo 5 mentre i protocolli specifici per applicazioni M2M saranno illustrati nel Capitolo 3.

In fine un altro parametro non meno importante da tenere in considerazione è quello del *costo*. Una WSN con molte se non tutte le caratteristiche suddette ma la cui implementazione comporta dispositivi costosi e ingombranti potrebbe non essere mai adottata. L'analisi dei costi e delle prestazioni dei nodi commerciali per reti WSN è sviluppata nel Capitolo 4.

L'analisi dei trade-off fra tutti i fattori suddetti risulta quindi importante oltre che complessa.

L'importanza (ranking) dei requisiti che determinano la QoS di una WSN è stato oggetto di studio di un survey inerente l'impiego di reti di sensori wireless nell'ambito industriale prodotto alla fine del 2014 dalla On World in collaborazione con l'International Society of Automation (ISA) [10] e a cui ha partecipato anche l'unità di ricerca. I risultati di tale survey, basati su interviste effettuate ad oltre 220 esperti di automazione fra ricercatori, ingegneri delle principali aziende manifatturiere e compagnie del settore oil & gas, oltre che rivenditori di sistemi di automazione industriale, permettono di affermare quanto segue:

- i fattori ritenuti più importanti per una WSN rivolta ad applicazioni industriali sono l'affidabilità dei dati (per il 96% degli intervistati), la sicurezza della rete (per l'87%) e la facilità di utilizzo e di accesso ai dati (per il 69%), seguiti della disponibilità di dispositivi a basso costo (59%) e dall'autonomia delle batterie (56%);
- i fattori principali che limitano l'impiego delle WSN in ambito industriale sono la sicurezza (59%), il costo (49%), la limitata autonomia delle batterie (44%) e la complessità (41%);
- almeno due intervistati su tre indica che i fattori su cui rileva un minor grado di soddisfazione e che quindi andrebbero migliorati sono: i costi, la non semplice integrazione con altre reti, l'autonomia delle batterie e l'assenza di tool che ne semplifichino l'utilizzo;
- sebbene per la realizzazione di WSN in ambito industriale gran parte degli intervistati si affidi a standard proprietari quali WirelessHART (42%), ZigBee (13%) e ISA100.11[a] (19%), oltre il 75% degli stessi ritiene importante l'utilizzo di standard aperti, in particolare il 50% degli intervistati auspica che i sensori siano indirizzabili mediante protocolli IP e accessibili da dispositivi mobili (smartphone);
- l'80% degli intervistati ritiene fondamentale l'impiego di reti mesh per poter accrescere arbitrariamente il numero di nodi e l'estensione della rete.

2.2 Requisiti per la reti di distribuzione dell'energia (Smart Grid)

Figura 3 Schema di una rete elettrica.

Una rete elettrica può essere schematizzata come in Figura 3 e prevede essenzialmente tre segmenti (o fasi):

1. il segmento di Generazione/Produzione costituito dalle centrali elettriche e dagli altri sistemi di generazione e accumulo;
2. il segmento di Trasmissione, costituito da sottostazioni, tralicci, trasformatori e linee elettriche di alta tensione (132-220-400 kV);
3. il segmento di Distribuzione essenzialmente costituito dalle linee elettriche a media (15-20-23kV) e bassa (230-400V) tensione che alimentano gli utenti finali e dai sistemi di telemetria.

Per quanto si tratti di un sistema complesso e di notevole estensione geografica, fatta eccezione per l'introduzione dei sistemi di telemetria atti a rendere più efficiente il sistema di fatturazione, e dei sistemi di controllo centralizzati (SCADA) presenti nelle centrali e nei siti di generazione, gran parte della rete elettrica è rimasta sostanzialmente invariata per oltre cento anni. Solo a seguito dei blackout del settembre 2003 (in Italia) e del novembre del 2006 quest'ultimo ha coinvolto più di 15 milioni di utenze in Europa, si è compresa la necessità di fornire alla rete elettrica un sufficiente grado di "intelligenza" [11]. Ad oggi, sebbene esista un sistema di monitoraggio e controllo per la sezione di trasmissione, manca un sistema di

supervisione distribuito in grado di monitorare tutti i segmenti in maniera puntuale e in tempo reale oltre che di telecontrollare a distanza tutti i sotto-sistemi. Tale sistema di telecontrollo è oggi necessario al fine di aumentare l'affidabilità e l'efficienza della rete elettrica in considerazione anche dell'aumento della complessità della rete, legato alla diversificazione dei sistemi di produzione dell'energia (basati sempre più su energie rinnovabili e in particolare su sistemi fotovoltaici dislocati anziché su centrali ben localizzate), oltre che per predisporre i meccanismi di sicurezza necessari per reagire tempestivamente alle possibili calamità naturali oltre che ad eventuali attacchi terroristici.

In tale contesto il paradigma delle Smart Grid [12] si propone di rivoluzionare le reti di distribuzione dell'energia elettrica migliorandole in termini di efficienza, affidabilità e flessibilità.

In particolare tale paradigma prevede l'impego di reti di sensori e attuatori wireless per realizzare comunicazioni bidirezionali che permetterebbero da un lato di monitorare temperatura, tensione e vibrazioni dei sotto-sistemi e dall'altro di inviare segnali di controllo per attivare sistemi di ventilazione in caso di sovra-temperature oltre che segnalazioni di allarme nel caso ad esempio di sovra-tensioni legate a problemi elettrici o di vibrazioni tali da identificare rischi strutturali, furti e/o manomissioni. Va de se inoltre che un monitoraggio dei consumi su tutta la rete e non solo nell'utenza finale permetterebbe di ridurre gli sprechi e anticipare situazioni di guasto.

Al fine di promuovere lo sviluppo delle Smart Grid in Europa e avere una visione condivisa dai diversi paesi dell'Unione e da tutti gli attori del sistema elettrico la Commissione Europea diede mandato nel 2011 ad alcuni dei principali organismi di standardizzazione europea (CEN, CENELEC ed ETSI) che costituirono lo Smart Grid Coordination Group (SGCG).

Il mandato della commissione europea (M/490) evidenziava le seguenti esigenze: ottenere celerità nelle azioni; permettere il coinvolgimento di un enorme numero di soggetti interessati e la necessità di lavorare in un contesto dove molte delle attività hanno un'estensione internazionale.

Il primo passo verso una visione condivisa è stato la definizione di un linguaggio comune. A questo scopo il Gruppo si è impegnato a promuovere l'interoperabilità e a sviluppare la convergenza degli standard europei e nazionali. Lo SGCG ha terminato il suo mandato alla fine del 2014 con i seguenti documenti (richiesti dalla commissione europea) prodotti da 4 diverse commissioni di lavoro:

- *Extended Set of Standards support Smart Grids deployment* che individua l'insieme di standard per supportare lo sviluppo di smart grid;
- *Overview Methodology (and its annexes)*: modello di sviluppo del mercato generale *(General Market Model Development)*, manuale utente per l'architettura dei modelli di smart grid *(Smart Grid Architecture Model User Manual)* e gestione della flessibilità *(Flexibility Management)*;
- *Smart Grid Interoperability and its tool:* Interoperabilità delle smart grid e suoi strumenti*;*
- *Smart Grid Information Security:* Sicurezza delle informazioni nelle smart grid*.*

La partecipazione italiana allo SGCG è stata coordinata dal CT 313 del CEI ed è stata garantita da esperti di RSE ed ENEL Distribuzione, che hanno contribuito attivamente alle quattro commissioni di lavoro.

Dai documenti suddetti oltre che da altri studi di organismi internazionali, ad esempio [12], possono essere individuate le seguenti parole chiave che costituiscono i principali requisiti di sistema nell'ambito delle Smart Grid:

Scalabilità: considerata la notevole estensione della rete (migliaia di chilometri di linee dislocate sul territorio nazionale) e l'elevato numero di elementi che potrebbe essere necessario monitorare (solo i trasformatori sono diverse decine di migliaia) si ritiene che la scalabilità sia il requisito fondamentale in ambito Smart Grid. Gli elementi di sensing, devono quindi avere un costo ridotto e il loro numero non deve essere limitato a priori; deve inoltre essere possibile aggiungere o rimuovere un elemento di sensing con facilità (occorre quindi gestire variazioni topologiche); a differenza di un sistema per l'automazione industriale la rete è decisamente più estesa occorre quindi prevedere trasmissioni multi-hop. Tale requisito

può essere garantito solo mediante opportuni **algoritmi di instradamento** in grado di gestire topologie di rete di tipo mesh.

Autonomia: paradossalmente, nonostante la presenza della rete elettrica, i sistemi di sensing e attuazione devono essere autoalimentati; infatti mettere dei trasformatori nelle linee di alta e media tensione per ottenere i 3V necessari per alimentare i nodi sensori non è proponibile sia per motivi di costo che di ingombri. Si ritiene che attualmente, fra le varie fonti energetiche disponibili per l'alimentazione dei nodi, l'alimentazione possa essere solo a batterie visto che le tecniche di recupero dell'energia dall'ambiente (harvesting) oltre ad incidere sui costi, aumenterebbero gli ingombri (vedi sistemi fotovoltaici e eolici) o sarebbero poco efficienti (come nel caso di vibrazioni e campi elettromagnetici). Ciò non di meno uno studio in tal senso sarebbe auspicabile. Al fine di evitare continui interventi di manutenzione l'autonomia deve essere elevata e quindi occorre minimizzare i consumi mediante apposite **tecniche di ottimizzazione dei consumi energetici** (duty-cycling, compressione dati, ecc.).

Sicurezza: in accordo con diversi studi [12] e organismi internazionali (EPRI) un requisito fondamentale delle smart grid risulta essere quello della **cyber-security**. È di fondamentale importanza che lo scambio di informazioni relative alle operazioni di sensing e di telecontrollo siano cifrate con appositi algoritmi al fine di limitare possibili frodi o attacchi terroristici. Occorre però tenere presente che le limitate risorse computazionali dei nodi sensori non permettono l'uso di meccanismi di cifratura sofisticati; inoltre le tecniche di crittografia incidono negativamente sull'autonomia oltre che sulle prestazioni (throughput e ritardi) della rete. In accordo con diversi organismi di standardizzazione quali il NIST e il FIPS, si ritiene che allo stato attuale la soluzione preferibile per garantire un adeguato livello di sicurezza limitando la degradazione delle prestazioni si basi sull'utilizzo dell'algoritmo di cifratura AES (Advanced Encryption Standard). Nell'ambito dell'attività di ricerca sono stati quindi analizzati i meccanismi di sicurezza disponibili per le reti di sensori.

Robustezza: a differenza di altri contesti le smart grid prevedono scenari notevolmente differenti in termini di canale di propagazione, traffico e topologie che ne rendono la progettazione particolarmente complessa. In particolare lo scenario di propagazione può essere particolarmente ostile per le comunicazioni wireless: infatti occorre considerare la presenza di ostacoli e in particolare di strutture metalliche che per tramite di riflessioni possono portare a fenomeni di multi-path fading (ovvero di affievolimento del segnale per effetto dei percorsi multipli) con conseguente riduzione delle prestazioni; è poi necessario garantire una adeguata immunità ai campi elettromagnetici provenienti da trasformatori e linee di trasmissione e prevedere l'interoperabilità con diversi sistemi di comunicazione (WiFi, Bluetooth, sistemi cellulari 3G/4G ecc.) che, operando nello stesso range di frequenza, possono portare a interferenze. Nell'ambito dell'attività sono state condotte alcune misure sperimentali, riportate nel Capitolo 4, atte a determinare il livello di robustezza delle WSN in vari scenari di propagazione.

Latenza: in generale il controllo di un sistema o di un processo in tempo reale pone dei limiti stringenti sui tempi di reazione e di elaborazione. Tali limiti sono in genere dipendenti dalla applicazione. Sulla base del "Codice di Rete" redatto da Terna [13], azienda titolare delle attività di trasmissione e dispacciamento della rete nazionale dell'energia elettrica, ed approvato dalla AEEG, il controllo in tempo reale dei sottosistemi della rete elettrica richiede il trasferimento di dati e comandi con un ritardo massimo compreso in genere fra 2 e 4 secondi ma che può ridursi ad 1 secondo se si tratta di eventi o allarmi di particolare importanza e anche 0.2 secondi per segnalazioni o comandi relativi agli impianti appartenenti ai piani di difesa. In seguito ci si riferirà a tali tempistiche quando si parlerà di sistemi real-time. Nell'ambito dell'attività di ricerca è stata vagliata la capacità delle WSN di rispettare tali ritardi sia con modelli analitici che con misure sperimentali.

2.3 Requisiti per sistemi di misurazione intelligenti (Smart Meter)

Un sistema di misurazione intelligente (smart meter) può essere definito come un *"sistema elettronico capace di misurare i consumi energetici, fornendo più informazioni rispetto ad un misuratore convenzionale, oltre che di trasmettere e ricevere dati mediante una forma di comunicazione elettronica"* [14]. In ambito

europeo diverse sono le Direttive che riguardano tali sistemi. In particolare la direttiva 2012/27/EU prevede l'impiego di smart meter negli Stati membri dell'Unione Europea in considerazione dei diversi vantaggi che questi comporterebbero fra cui una maggiore consapevolezza dei propri consumi energetici da parte dei consumatori (energy footprint) ed una possibile partecipazione attiva degli stessi al mercato energetico in grado di migliorarne l'efficienza complessiva.

Di contro diverse sono le problematiche tecniche relative ad esempio all'installazione ed all'interoperabilità di contatori intelligenti per servizi differenti (elettricità, acqua, gas e calore) anche in considerazione dei differenti assetti e posizioni degli Stati membri in merito. Al fine di analizzare tali problematiche la Commissione europea diede mandato nel 2009 ad alcuni dei principali organismi di standardizzazione europea (CEN, CENELEC ed ETSI) di sviluppare una architettura aperta per sistemi di smart metering. A fronte di tale mandato (numero M/441) venne fondato il Gruppo di Coordinamento per gli Smart Meter (SM-CG) che tra il 2009 e il 2016 ha prodotto diversi report inerenti i sistemi di misurazione intelligente.

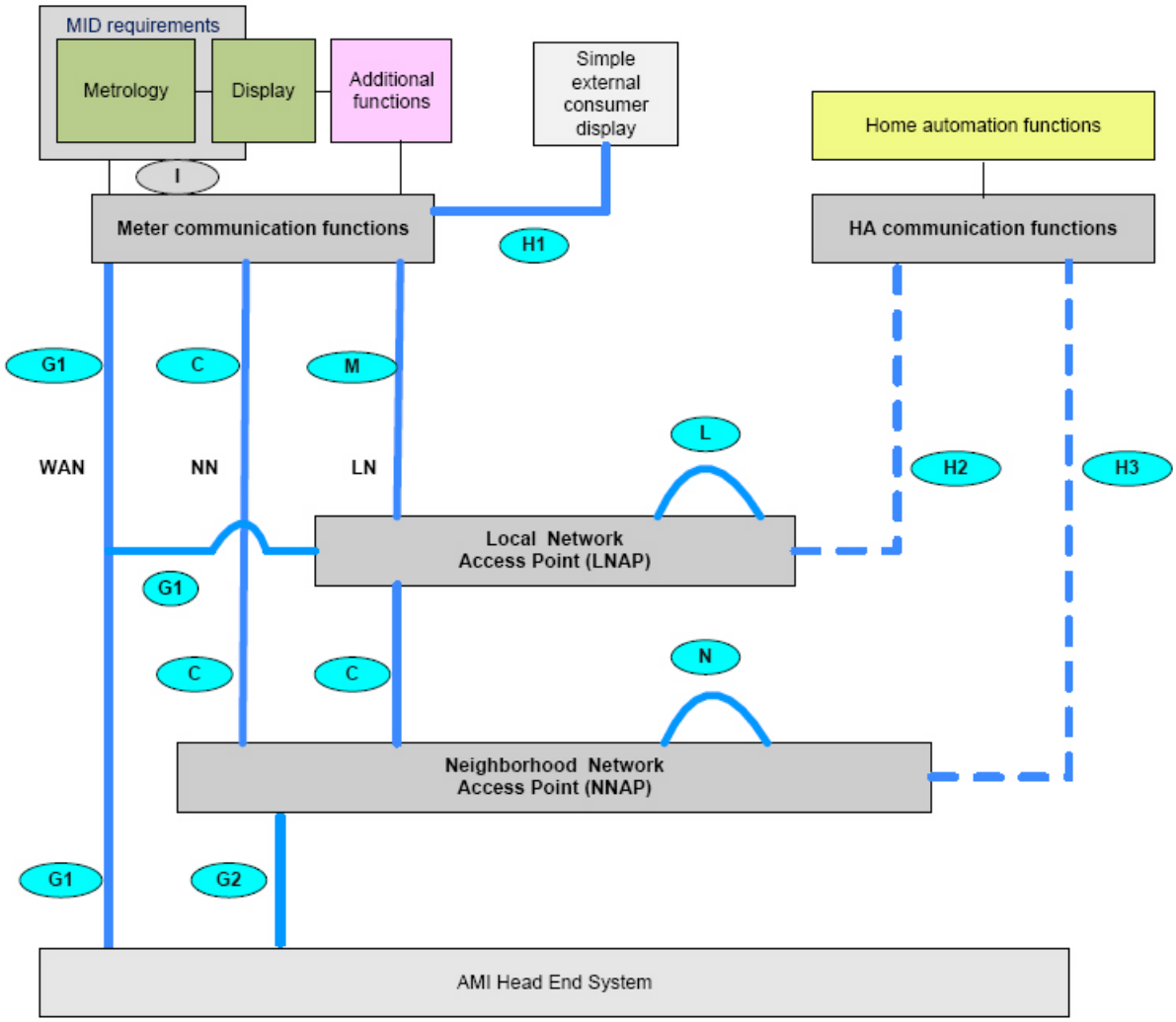


Figura 4 Schema dell'architettura di riferimento per le comunicazioni di smart metering [15].

In particolare nel report tecnico TR 50572 [15] del 2011 è definita l'architettura in Figura 4 e sono elencate le seguenti funzionalità necessarie per identificare un contatore come "intelligente" (ovvero le funzionalità necessarie oltre quelle di misura e visualizzazione già presenti nei comuni contatori):

- **Lettura remota** dei registri metrologici e loro invio alle organizzazioni di mercato designate.
- **Comunicazione bidirezionale** tra i sistemi di misura e le organizzazioni di mercato designate.

- **Supporto di sistemi tariffari** e di pagamento avanzati.
- **Controllo remoto** della fornitura (abilitazione/disabilitazione) e limitazione di flusso/potenza.
- **Comunicazioni sicure** al fine di esportare i dati metrologici a sistemi di visualizzazione e analisi dell'utente o di terzi designati dall'utente.
- Poter mettere a disposizione le informazioni metrologiche tramite **portale/gateway ad un sistema di visualizzazione** all'interno della casa/edificio o di dispositivi ausiliari.

Alle suddette si è aggiunta, come ulteriore funzionalità, quella di

- Permettere la **comunicazione** dei sistemi di misurazione **con dispositivi o gateway per la fornitura di servizi per l'efficienza energetica** e per la gestione della domanda.

Dal rapporto tecnico scaturisce la Raccomandazione 2012/148/UE che specifica dieci requisiti minimi per gli smart meter:

- comunicazione della lettura direttamente all'utente o ad un terzo da questi designato;
- aggiornamento dei dati di lettura con sufficiente frequenza (almeno 15 minuti);
- telelettura del contatore;
- comunicazione bidirezionale verso reti esterne per manutenzione e controllo;
- lettura sufficientemente frequente per la pianificazione della rete;
- supporto a regimi tariffari avanzati;
- telecomando on/off dell'erogazione e/o del flusso o limitazione della potenza;
- sicurezza delle comunicazioni;
- prevenzione e accertamento delle frodi;
- possibilità di importazione/esportazione (ovvero prelievo e immissione) e misurazione reattiva.

In seguito si considererà quindi un intervallo di 15 minuti come tempo di rilevazione delle misure per applicazioni di smart metering.

Si noti che il primo punto della Raccomandazione 2012/148/UE è disatteso dagli attuali sistemi di smart metering che prevedono l'invio dei dati di consumo prima al gestore e poi all'utente.

Nell'ambito dell'attività si è quindi investigata la possibilità di utilizzare le WSN come interfaccia fra l'utente e gli smart meter (sia in ambito industriale che per applicazioni di domotica) definendo una architettura, illustrata nel Capitolo 10, che può essere efficacemente utilizzata per la realizzazione di Local Network Access Point (si veda la Figura 4) garantendo all'utente l'accesso diretto ai dati e permettendogli di monitorare i consumi da un semplice terminale mobile (smartphone o tablet).

La Direttiva 27/2012/UE è stata recepita dall'Italia con il DL 102/2014 che ha dato mandato all'Autorità per l'energia elettrica, il gas ed il sistema idrico (AEEGSI) di predisporre le specifiche per la seconda generazione dei misuratori intelligenti. Quest'ultima nel documento 416/2015/R/EEL indica i seguenti criteri generali per smart meter "a prova di futuro" (*future-proof design*):

- minimizzazione delle esigenze di riprogrammazione;
- massima indipendenza da componenti hardware aggiuntivi;
- separazione delle risorse tra telegestione e messa a disposizione dei dati;
- interoperabilità con dispositivi di terze parti per la messa a disposizione dei dati ai clienti o a terze parti designate;
- intercambiabilità tra sistemi di imprese distributrici diverse;
- immunità in ambienti elettromagnetici perturbati;

- multicanalità per la comunicazione e la messa a disposizione dei dati;
- sicurezza informatica avanzata (cybersecurity);
- progressiva integrazione con i sistemi intelligenti di distribuzione;
- minimizzazione dei vincoli di retro compatibilità.

Si noti che i nodi sensori di una WSN se opportunamente progettati e configurati possono facilmente implementare le funzionalità suddette.

L'Italia e la Svizzera sono stati fra i primi paesi a dotarsi di sistemi di misurazione intelligente, ben prima della direttiva Europea. In particolare in Italia ENEL ha installato i primi contatori intelligenti per l'energia elettrica già nel 2001 (progetto Telegestore). Come rilevato dalla AEEGSI nel documento 416/2015/R/EEL tali contatori hanno però diversi limiti dovuti principalmente all'impego di protocolli proprietari e all'utilizzo di tecnologie ad onde convogliate (Power Line Carrier, PLC). Infatti:

1) la comunicazione tramite onde convogliate richiede che la linea elettrica sia in tensione (in assenza di tensione ogni comunicazione è interdetta, non vi è quindi la possibilità di inviare allarmi). La presenza di un canale di comunicazione diverso (a radiofrequenza) permetterebbe di conoscere lo stato di funzionamento anche in assenza di tensione, con potenziali benefici di miglioramento della continuità di servizio.
2) Il rumore derivante da apparecchiature elettroniche del consumatore e le emissioni elettromagnetiche degli inverter degli impianti fotovoltaici risultano ostacolare notevolmente la comunicazione PLC su linea elettrica.

A ciò si aggiunge il fatto che la velocità di trasmissione possibile con PLC è piuttosto limitata (la banda a disposizione è inferiore ai 100kHz e la bit-rate attuale è di appena 2.4kbps)

Il documento 416/2015/R/EEL auspica quindi la multimodalità, e in particolare l'uso di comunicazioni a radiofrequenza, come strumento per superare gli svantaggi della prima generazione. In particolare sempre nel documento viene asserito che per assicurare interoperabilità è necessario che il protocollo di comunicazione sia aperto e unico a livello nazionale e che occorre concordare una pila protocollare completa.

In tale contesto affiancare alle PLC l'utilizzo di WSN basate su protocolli IP (ovvero i protocolli Internet, aperti e ampiamente collaudati) potrebbe rappresentare una soluzione efficace ed efficiente per permettere di trasmettere informazioni sui consumi energetici all'utente finale direttamente e in tempo reale, oltre che per fornire un secondo canale di comunicazione con il gestore utile per la segnalazione e rilevazione di guasti alla linea elettrica.

Analoghe considerazioni possono farsi per gli smart meter relativi ai servizi di fornitura di gas e acqua dove allo stato attuale le soluzione proposte per la comunicazione si basano sull'utilizzo del Wireless M-Bus (standard UNI CEI EN 13757-4) [16] o di comunicazioni cellulari. Entrambe le soluzioni richiedono ingenti investimenti. Nel primo caso occorre infatti la realizzazione di una nuova infrastruttura di rete basata su uno standard attualmente poco sperimentato, nel secondo caso occorre considerare i costi relativi alle SIM e al traffico dati. Va osservato inoltre che in uno scenario in cui i diversi smart meter per le diverse utility presentano soluzioni tecnologiche differenti, l'utente difficilmente potrebbe avere accesso ai dati relativi a tutte le proprie utenze in tempo reale e tramite un'unica piattaforma.

L'architettura proposta nel presente elaborato fornisce una soluzione in tal senso con un approccio diametralmente opposto in cui le informazioni prevenienti dagli smart meter sono accessibili direttamente all'utente per tramite di una rete WSN permettendo così all'utente di monitorare i consumi da un semplice terminale mobile (smartphone o tablet); la soluzione risulta a basso costo poiché il gestore, grazie alle WSN, potrebbe avere accesso remoto agli smart meter già per tramite di Internet e del router ADSL dell'utente stesso e semplici apparati WiFi garantirebbero la possibilità di interrogare direttamente i nodi sensori da parte degli operatori autorizzati al fine di rilevare guasti e/o frodi.

2.4 Conclusioni

In questo Capitolo sono stati analizzati i principali requisiti degli scenari applicativi di interesse per il progetto (smart industries, smart grid e smart metering). La Tabella 1 riporta una sintesi dei requisiti individuati per i diversi scenari e che hanno guidato le scelte progettuali nell'ambito dell'attività.

A fronte dell'analisi suddetta sono stati individuati i punti attenzionati nell'analisi dello stato dell'arte delle WSN e che hanno portato alla definizione dell'architettura e alla realizzazione dei firmware e del software sviluppati per il progetto. Più precisamente:

- *costi:* una panoramica sulle piattaforme commerciali per la realizzazione di WSN è fornita nel Capitolo 4;
- *protocolli:* i protocolli specifici per WSN e i protocolli M2M sono analizzati rispettivamente nel Capitolo 3 e 5 con particolare enfasi ai protocolli aperti che permettono l'interoperatività delle WSN con reti IP;
- *sicurezza:* gli aspetti legati alla sicurezza nelle WSN sono analizzati nel Capitolo 7;
- *consumi/autonomia:* le tecniche per la riduzione dei consumi energetici sono analizzate nel Capitolo 8;
- *prestazioni/QoS* (throughput, ritardi, affidabilità ed efficienza energetica): un modello in grado di determinare le prestazioni della rete a partire dai principali parametri di configurazione è fornito nel Capitolo 9;
- *semplicità di utilizzo:* un software di gestione in grado di semplificare la configurazione di reti WSN e di fornire strumenti per la visualizzazione e l'elaborazione dei dati è illustrato nel Capitolo 10.

Tabella 1 Requisiti e specifiche di sistema.

Automazione industriale (smart industries)
• *Flessibilità, scalabilità e riconfigurabilità della rete* • *Inizializzazione e riconfigurazione dei nodi* • *Costi* • *Sicurezza* • *Interoperabilità* • *Semplicità di utilizzo e disponibilità di tool di gestione* • *Identificazione dei dispositivi e meccanismi di discovery* • *Tecniche di risparmio energetico (Duty Cycling)* • *Definizione della Qualità del Servizio (QoS)* • *Affidabilità (reliability)* • *Efficienza energetica* • *Throughput* • *Latenza*
Rete di distribuzione dell'energia (smart grid)
• *Scalabilità (algoritmi di instradamento dinamici per reti mesh)* • *Autonomia (minimizzazione dei consumi energetici)* • *Sicurezza* • *Robustezza* • *Latenza (ritardo massimo 4 secondi)*
Sistemi di misurazione intelligenti (smart meter)
• *Lettura remota* • *Comunicazione bidirezionale* • *Supporto di sistemi tariffari* • *Controllo remoto* • *Comunicazioni sicure*

- *Messa a disposizione delle informazioni metrologiche tramite portale/gateway di visualizzazione all'interno della casa/edificio o di dispositivi ausiliari*

3 Protocolli Machine-to-Machine (M2M)

I protocolli Machine-to-Machine (M2M) hanno come obiettivo quello di permettere lo scambio di informazioni fra dispositivi in maniera totalmente autonoma, ovvero senza un coinvolgimento esplicito da parte dell'uomo (se non limitatamente ad operazioni di supervisione); in tale contesto l'accezione "macchina" va quindi intesa in antitesi alla parola "uomo", per riferirsi in genere a dispositivi (sensori, tag RFID, attuatori, ecc.) o anche parti di software (agenti). Questi protocolli costituiscono l'elemento fondante del paradigma dell'Internet of Things (IoT) in base al quale in un prossimo futuro saranno connessi ad Internet anche oggetti comuni come documenti cartacei e contenitori per il cibo [17] prevedendo che già nel 2022 le comunicazioni M2M costituiranno il 45% del traffico Internet [18].

Le comunicazioni M2M sono altresì alla base dei Cyber Physical Systems (CPS) ovvero di quei sistemi che vedono l'integrazione di sistemi di comunicazione, elaborazione e di controllo [19].

Machine-to-Machine (M2M), Internet of Things (IoT), Cyber Physical Systems (CPS), Industry 4.0, Industrial Internet, Internet of Everything, Big Data, e TSensors sono tutti nomi di paradigmi (fra loro sovrapposti e interconnessi e di cui spesso si perdono i confini) che riflettono diversi sforzi e tecnologie mirate alla comunicazione fra il mondo dell'informazione (fatto da risorse informatiche come database e software), il mondo cybernetico (fatto da macchine e dispositivi) e il mondo reale (fatto da uomini e oggetti di uso comune da parte dell'uomo stesso).

È opinione comune che tali paradigmi rivoluzioneranno a breve il mondo delle telecomunicazioni e non solo. Ampio è in fatti il numero di scenari applicativi: dall'automazione di processi industriali [20], alla domotica [21], ai sistemi di trasporto intelligenti [22], alla telemedicina [23], ai sistemi di smart metering [24] e alle smart grid [25] e smart city [26].

Nonostante i diversi possibili campi applicativi e le relative opportunità di business, notevoli sono i problemi ancora da affrontare in tale contesto legati principalmente a due fattori:

1. l'eterogeneità delle "macchine": i dispositivi M2M sono in genere dispositivi low-cost spesso basati su semplici microcontrollori e con notevoli differenze e limiti in termini di memoria, capacità di elaborazione, risorse energetiche, ecc.; in altri casi possono però essere dispositivi decisamente evoluti come smartphone e tablet di ultima generazione; non vi è quindi uniformità negli oggetti in termini di risorse hardware e software e ciò complica notevolmente la realizzazione delle applicazioni. Tutto ciò ha portato ad una notevole frammentazione del mercato;
2. la notevole differenza fra il traffico generato dall'uomo (human-based) e quello generato dalle macchine (machine-based): le comunicazioni human-based (fra uomini e macchine) sono in genere di tipo request-response, hanno cioè origine da una richiesta da parte dell'uomo, e sono spesso finalizzate al trasferimento di medie e grosse quantità di dati per singola richiesta (si pensi ad esempio al download di un file video) per cui i tempi di risposta possono essere ragionevolmente dell'ordine dei secondi; viceversa nelle comunicazioni M2M le comunicazioni scaturiscono da un evento e non da una richiesta, i tempi di risposta richiesti sono in genere dell'ordine dei milli-secondi (specie per applicazioni rivolte all'automazione) e le trasmissioni riguardano spesso piccole quantità di dati trasmessi in maniera sporadica.

In particolare quest'ultima differenza permette di comprendere perché i protocolli comunemente utilizzati in Internet (TCP e HTTP) non siano idonei per le comunicazioni M2M. Da qui la necessità di nuovi protocolli e più in generale di nuovi paradigmi e modelli di comunicazioni specifici per M2M.

Gran parte delle soluzioni M2M oggi esistenti derivano dalle reti di sensori (WSN) e dai sistemi di automazione industriale (SCADA). In tali contesti la comunicazione diretta M2M è però più una opzione che non la tipologia di comunicazione più rilevante; infatti, la comunicazione fra dispositivi in tali contesti esiste, ma si limita spesso al semplice inoltro dell'informazione al fine ultimo di estendere la copertura della rete. In ambito SCADA e WSN, inoltre, tutte le comunicazioni sono dirette verso un unico punto di raccolta (il

sink nelle WSN ovvero l'unità master in ambito SCADA) dove le informazioni sono elaborate per essere poi rese accessibili e/o per prendere decisioni centralizzate finalizzate all'attuazione.

Nell'accezione pura del M2M invece tutte le operazioni, comprese quelle di controllo e gestione, dovrebbero essere distribuite e lasciate alle stesse macchine, senza la necessità di un nodo centrale con funzioni di supervisione.

Sebbene diversi siano stati gli sforzi in passato in tale direzione, molti dei sistemi M2M sviluppati sono "verticali", rivolti cioè ad applicazioni specifiche e spesso basati su protocolli proprietari e/o su architetture monolitiche e poco flessibili [6].

Per facilitare lo sviluppo di applicazioni M2M risulta quindi di fondamentale importanza la disponibilità di standard in grado di garantire interoperabilità fra i dispositivi e i servizi M2M.

Nel corso degli anni diversi sono stati gli organismi interessati e le attività di standardizzazione riferite al M2M. Fra le più rilevanti:

- ETSI: L'European Telecommunications Standards Institute (ETSI) ha fondato nel 2009 un comitato tecnico (TC) specifico per l'M2M definendo una infrastruttura di riferimento standard per le applicazioni M2M [27] (la prima release è del 2011 e una seconda del 2013). In particolare alcuni scenari applicativi M2M nell'ambito degli smart metering sono descritti nel documento TR 102 898 [28];
- 3GPP: sotto la sigla 3GPP (3rd Generation Partnership Project) sono raggruppati diversi organismi di standardizzazione per le telecomunicazioni (fra cui l'ETSI in Europa, ARIB/TTC in Giappone, ATIS negli Stati Uniti e CCSA in Cina); a tale partnership si devono i principali standard della telefonia cellulare (GSM, UMTS e LTE) oltre che una serie di specifiche tecniche (TS 22.X e TS 23.X) finalizzate all'integrazione delle comunicazioni M2M nelle reti cellulari. In particolare con riferimento alla Release 13 e al documento TS 22.368 [29] sono analizzati i requisiti inerenti congestione di rete, consumi energetici, indirizzamento, identificazione e sicurezza;
- TIA: Telecommunications Industry Association (TIA) è l'equivalente negli Stati Uniti dell'ETSI in Europa. Il comitato tecnico TR-50 ha identificato i requisiti delle comunicazioni M2M e sviluppato un protocollo per la comunicazione di eventi indipendente dal mezzo fisico di comunicazione [30];
- IEEE: L'IEEE (Institute of Electrical and Electronic Engineers) nell'ambito dello standard IEEE 802.16 (commercialmente noto come WiMax) ha pubblicato un emendamento (802.16p) che definisce una interfaccia radio a larga banda per comunicazioni M2M tramite WiMax. I principali aspetti presi in considerazione sono relativi ai consumi energetici, alla scalabilità e all'autenticazione dei dispositivi.

L'elenco non è esaustivo e diversi sono stati i comitati tecnici sorti in diverse parti del Mondo specificatamente per applicazioni M2M (ad esempio il TC10 del CCSA in Cina) e/o legati indirettamente al M2M per tramite dell'IoT (ad esempio GISFI in India, ITU-T SG13 negli USA).

Tale frammentazione e dispersione di sforzi e di investimenti ha termine nel 2012 a seguito della nascita del progetto *oneM2M* ad opera dei principali organismi di standardizzazione (SDO) nel campo delle telecomunicazioni (ETSI (Europe), ARIB (Japan), ATIS (U.S.), CCSA (China), TIA (U.S.), TTA (Korea), and TTC (Japan)), col fine ultimo di definire uno standard globale per la comunicazione di dispositivi e servizi per applicazioni M2M.

La prima definizione dello standard è stata pubblicata nel febbraio 2015 (una seconda release è prevista per la fine di quest'anno, 2016) e ad oggi aderiscono al progetto oneM2M più di 260 membri (fra cui IEEE, Cisco, Telecom Italia, Intel e Huawei).

Poiché l'architettura scaturita dal progetto oneM2M ha già in se diversi elementi importanti per il progetto MiSE-ENEA si ritiene utile e importante presentarla brevemente rimandando alla specifiche e ai report tecnici (accessibili gratuitamente dal sito dell'ETSI, www.etsi.org) per approfondimenti.

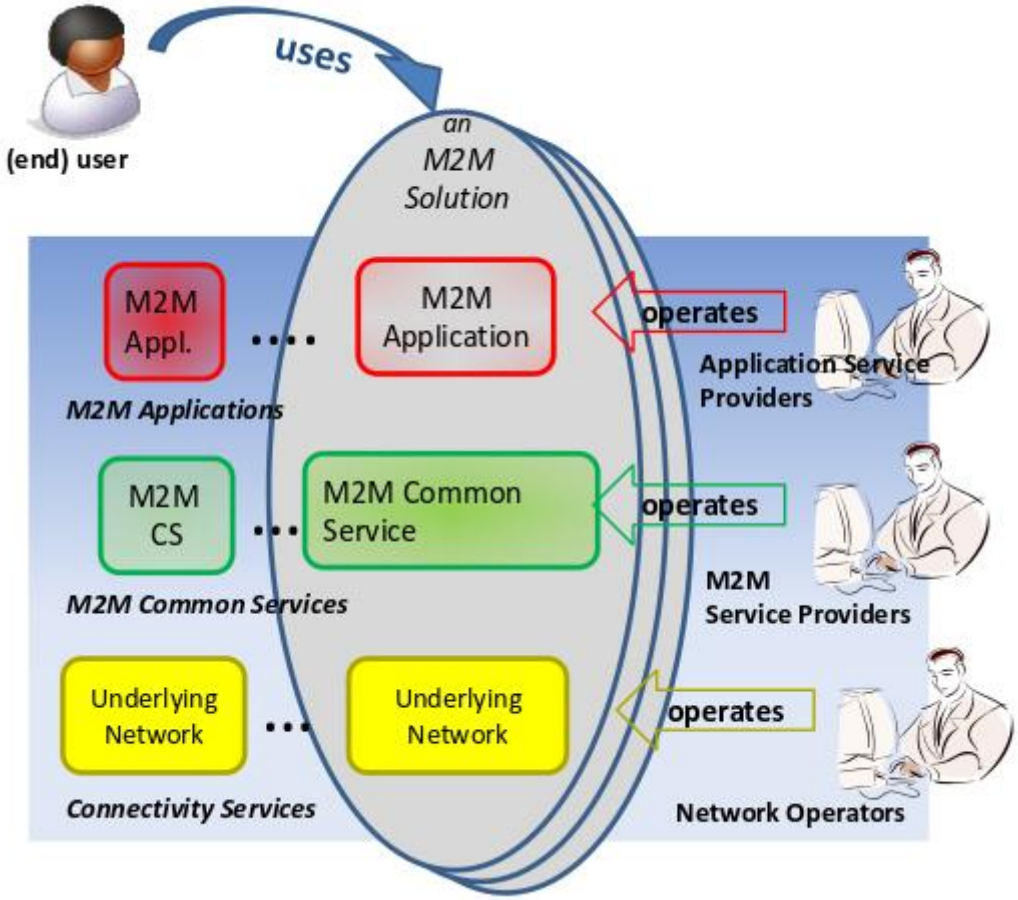


Figura 5 Ecosistema oneM2M [31].

3.1 *L'architettura oneM2M*

L'architettura oneM2M [31] è una architettura di tipo "orizzontale" (non rivolta ad una applicazione specifica). Essa si basa sulla definizione di un ecosistema (si veda la Figura 5) atto a descrivere le interazioni fra le diverse tipologie di utenti che possono intervenire in uno scenario M2M, ovvero: i fruitori delle applicazioni (End User, ovvero singoli individui o anche Compagnie), i fornitori di servizi applicativi (Application Service Provider), i fornitori di servizi M2M (M2M Service Provider) ed infine gli operatori di rete (Network Operator) atti a fornire servizi di connettività.

Partendo da tale ecosistema lo standard oneM2M definisce una architettura a tre strati (layer) denominati rispettivamente Application Layer, Common Services Layer e Network Layer. Per ogni strato possono essere presenti una o più entità rispettivamente dette Application Entity (AE), Common Services Entity (CSE) e Network Services Entity (NSE) che contengono le funzionalità necessarie per lo sviluppo di servizi e applicazioni M2M.

Un ruolo fondamentale nell'architettura oneM2M è svolto dalle entità CSE che costituiscono sostanzialmente un middleware in grado di semplificare lo sviluppo delle applicazioni e renderle interoperabili fornendo una o più delle seguenti funzionalità (meglio note come Common Service Functions, CSF):

- *Registration:* permette ad una AE (o ad una CSE remota) di registrarsi per utilizzare i servizi offerti dalla CSE del nodo.
- *Discovery:* è responsabile della ricerca delle informazioni inerenti risorse e loro attributi (in genere risponde a richieste provenienti da altri CSE o AE).

- *Security:* fornisce funzioni per la protezione di dati sensibili (credenziali, chiavi, ecc.), per il controllo degli accessi e autenticazione e per la protezione dell'identità (es. fornisce identità virtuali non riconducibili a quella originaria).
- *Group Management:* permette la definizione di gruppi per eventuali trasmissioni multicast o per semplificare la gestione dei diritti di accesso (così come avviene nei sistemi operativi).
- *Data Management & Repository:* definisce le funzioni per memorizzare, elaborare e aggregare dati ed eventualmente convertirli di formato.
- *Subscription & Notification:* per tramite di queste funzionalità una AE (o un'altra CSE remota) può richiedere la sottoscrizione ad una risorsa di una CSE. A seguito degli appositi controlli sui diritti di accesso la sottoscrizione viene confermata. Inoltre in caso di variazioni della risorsa (cancellazione, variazione di parametri ecc.) le AE (o la CSE remota) riceve una notifica. I vantaggi di tale meccanismo (noto come publish-subscribe) saranno chiariti in seguito.
- *Device Management:* per tramite di queste funzioni le AE possono gestire i dispositivi senza conoscerne i dettagli tecnologici (anche se dettagli specifici potrebbero essere necessari per implementare ad esempio le funzioni di diagnostica).
- *Application & Service Management:* definisce le funzioni per configurare e aggiornare CSE e AE.
- *Communication Management & Delivery Handling:* è responsabile della gestione delle comunicazioni (ad esempio decide quando e come instaurarle).
- *Network Service Exposure, Execution and Triggering:* permette di accedere ai servizi di rete astraendo dal particolare mezzo trasmissivo o dalle tecnologie di comunicazione supportate dalla rete.
- *Location:* permette di ottenere informazioni di geo-localizzazione sui nodi utili per i servizi di localizzazione.
- *Service Charging & Accounting:* gestisce i servizi (sia offline che online) soggetti a pagamento/fatturazione, cattura eventi correlati e genera record per la fatturazione, comunica con i nodi infrastrutturali collegati a tali servizi.

Nell'architettura oneM2M un nodo altro non è che un insieme di più entità, anche appartenenti a strati diversi.

In particolare è possibile distinguere fra nodi CSE-Capable (che presentano almeno una entità CSE) e nodi Non-CSE-Capable (che non hanno entità CSE).

Altre tipologie di nodi previste dallo standard sono:

- Application Service Node (contiene una CSE e almeno una AE), tipicamente sono dispositivi M2M con risorse tali da poter fornire servizi anche ad altri nodi;
- Application Dedicated Node (non contiene CSE ma contiene almeno una AE), tipicamente sono dispositivi "resource-constrained" (es. i mote nelle WSN);
- Middle Node (contiene una CSE ma non necessariamente contiene AE), tipicamente sono nodi gateway atti a fornire servizi localizzati (elaborazione, aggregazione, accesso ad Internet o reti WAN). Nell'ambito M2M la presenza di gateway è da ritenersi necessaria per poter sopperire alle limitate risorse dei dispositivi M2M "resource-constrained" (ovvero i nodi ADN) e conferire flessibilità al sistema.
- Non-oneM2M Node (sono nodi che non contengono entità oneM2M), un esempio di tale tipo di nodi è dato dai tag RFID ovvero da tutti quei dispositivi che non hanno capacità di elaborazione.

L'insieme dei nodi suddetti costituisce il Field Domain (tradotto letteralmente "dominio di campo").

Vi è, Infine, un'altra tipologia di nodi prevista dallo standard:

- Infrastructure Node (contiene una CSE, non necessariamente contiene AE), è previsto un solo IN per ogni Service Provider atto a fornire servizi di tipo subscription, security and charging;

L'insieme degli IN costituiscono l'Infrastructure Domain (tradotto letteralmente "dominio delle infrastrutture") pensato per piattaforme cloud-based.

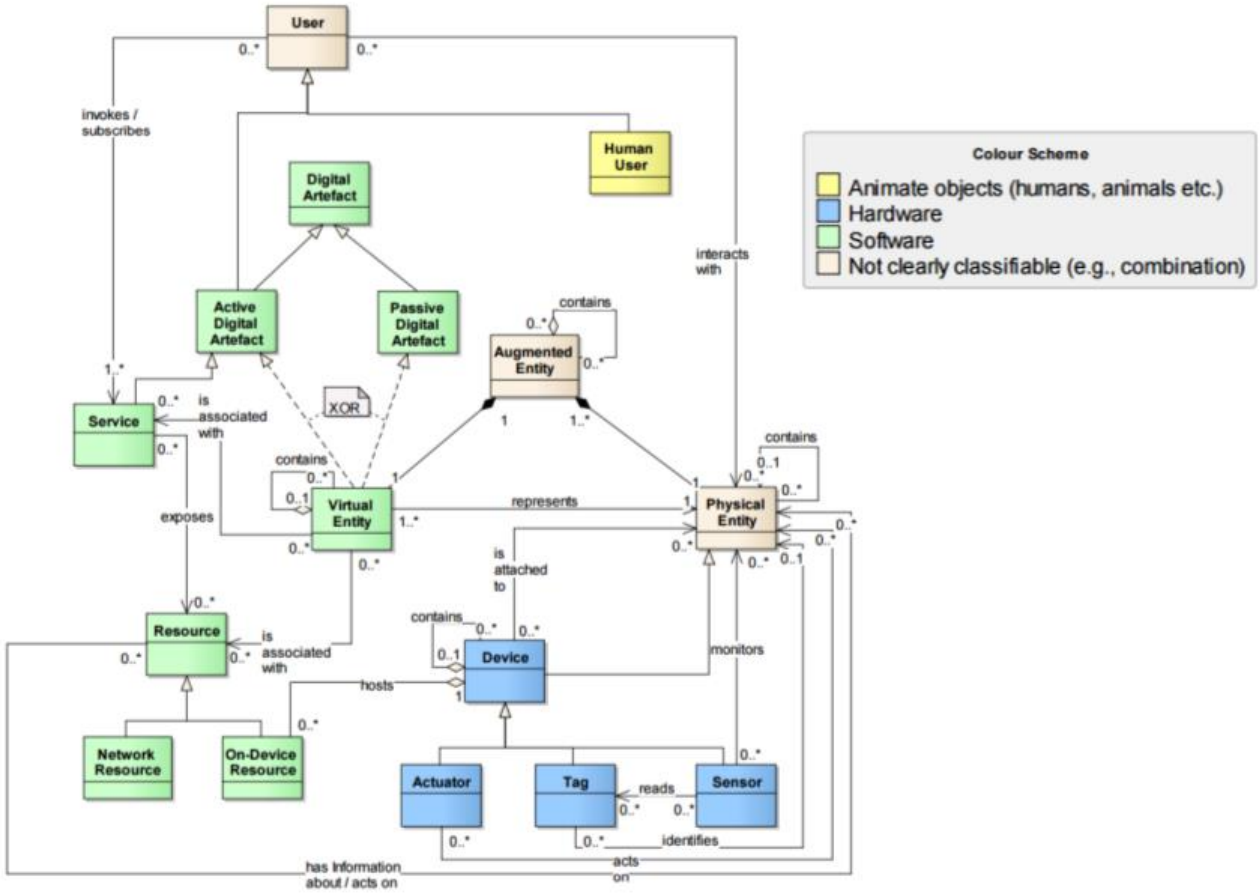
Prima del oneM2M diverse altre architetture per l'IoT sono state definite in numerosi progetti Europei [32].

Fra i più noti:

- CASAGRAS [33], conclusosi nel 2009 e rivolto principalmente alla tecnologia RFID;
- IoT-A [34], progetto EU FP7 rivolto alla definizione di un modello architetturale, mostrato in Figura 6, capace di tener conto delle diverse interazioni fra i vari componenti (Utenti, Servizi, Risorse e Dispositivi) e che deriva da una attenta analisi di requisiti provenienti da diversi stakeholder e scenari applicativi accessibile al link

  http://www.iot-a.eu/public/requirements/copy_of_requirements

- CERP-IoT [35] e iCore [36], entrambi progetti EU FP7, rispettivamente del 2010 e del 2011, che hanno adottato il modello proposto in IoT-A;
- AIM [37], progetto del 2010, che, anche se non rivolto specificatamente all'IoT, risulta particolarmente interessante per il progetto MiSE-ENEA poiché finalizzato ad una architettura per la virtualizzazione, modellizzazione e gestione dei consumi energetici degli elettrodomestici.



Source: http://www.iot-a.eu/public/public-documents/d1.5/view

Figura 6 Architettura IoT-A.

Ad oggi le architetture proposte da tali progetti sono più dettagliate che non l'architettura oneM2M, ancora in fase di definizione. D'altra parte non possono essere considerati standard internazionali.

Si ritiene quindi che l'architettura oneM2M debba essere il riferimento anche per il progetto MiSE-ENEA pur prendendo spunto dai progetti e dagli standard precedenti per poter arrivare ad una architettura di dettaglio.

Questa visione ha portato al prototipo di WSN descritto nel Capitolo 10 capace di integrare i paradigmi e protocolli per M2M di seguito descritti.

3.2 Paradigmi di comunicazione M2M

Due sono i principali meccanismi di comunicazione M2M [38]:

- il primo meccanismo noto come *request-based,* basato sul paradigma *request-response*, prevede che sia un nodo o una applicazione (client) ad interrogare un altro nodo (server) il quale risponde alla richiesta ad esempio restituendo il valore di una grandezza fisica; l'interrogazione avviene in genere ciclicamente, con un meccanismo noto come polling o in alternativa mediante meccanismi simili alle query dei database.;
- il secondo meccanismo noto come *event-based*, basato sul paradigma *publish-subscribe*, prevede che la comunicazione abbia origine da un evento (ad esempio la variazione di una grandezza fisica) a seguito del quale un nodo informa gli altri nodi interessati.

Il paradigma *request-response* è alla base delle architetture software denominate RESTful (ovvero REST-conformi). L'architettura REST (Representational State Transfer) fu ideata nel 2000 da Roy T. Fielding [39], a cui si deve anche il protocollo HTTP1.1, e prevede che una risorsa web su un server, identificata univocamente da un Uniform Resource Identifiers (URI) (e.s. http://myhost.com/…), possa essere richiamata da un client mediante appositi metodi identificabili con operazioni verbali (GET per ottenere informazioni su una risorsa, PUT per modificarla, POST per crearla e DELETE per eliminarla). Le richieste sono stateless ovvero senza stato per cui una singola richiesta deve contenere tutte le informazioni necessarie al server per eseguirla. Una risorsa può poi essere collegata ad un'altra tramite link permettendo così l'accesso ad altre risorse distribuite nel World Wide Web.

Tale meccanismo, sebbene semplice, flessibile e scalabile, risulta essere inefficiente per le comunicazioni M2M a causa dell'elevato numero di informazioni che devono essere scambiate fra client e server per singola richiesta, oltre che per la necessità di ripetere ciclicamente le richieste per sapere ad esempio quando una variabile assume un valore diverso. Ciò comporta un notevole spreco di risorse sia in termini di banda che di consumi energetici.

Una alternativa alle architetture RESTful per la realizzazione di servizi web è il SOAP [40]. Mediante l'uso del linguaggio XML il SOAP permette la definizione di servizi basati su una semantica più generale (ma anche più complessa). Tali servizi, come nel caso dell'architettura RESTful, risultano indipendenti dalla piattaforma e possono essere richiamati da nodi remoti indipendentemente dal sistema operativo o dal linguaggio di programmazione usati. Il vantaggio del linguaggio XML è quello di generare messaggi maggiormente comprensibili all'uomo (se confrontati con sequenze di bit) ma a scapito di un elevato overhead, sia in termini di memoria che di risorse di elaborazione, per cui tale linguaggio non risulta idoneo per comunicazioni M2M.

Alcuni sforzi per ridurre tale problema hanno portato ad una versione binaria e compressa dell'XML nota come EXI [41].

Un meccanismo che permette di ridurre notevolmente il numero di messaggi scambiati è il meccanismo event-based basato sul paradigma *publish-subscribe.* Tale paradigma prevede che i client (detti subscriber) manifestino il loro interesse ad una determinata informazione/risorsa comunicandolo ad un particolare nodo detto broker; questa operazione di registrazione (detta subscription) avviene in genere una sola volta.

Quando un dispositivo o un server (più propriamente detto publisher) ha un evento/informazione da comunicare (ovvero da pubblicare, da cui il nome del paradigma) la comunica al broker il quale a sua volta avviserà i subscriber interessati trasmettendo l'informazione in multicast.

Il meccanismo event-based ha quindi diversi vantaggi:

1. i subscriber sono immediatamente avvertiti dell'evento non appena questo è stato prodotto riducendo al minimo il tempo per la disseminazione del dato;
2. viene eliminata la fase di richiesta, riducendo così i messaggi trasmessi e quindi i consumi energetici;
3. non occorre coordinare un polling da parte di più entità permettendo così una maggiore scalabilità del sistema;
4. il meccanismo risulta indipendente dalla topologia della rete, ciò consente di riposizionare i nodi e di gestire la loro eventuale mobilità più facilmente senza che siano necessarie informazioni di localizzazione.

A fronte di tali vantaggi si ritiene che tale meccanismo debba essere preferito in ambienti M2M.
I meccanismi suddetti possono essere meglio compresi analizzando i principali protocolli specifici per le comunicazioni M2M e in particolare i protocolli COAP e MQTT di seguito descritti (COAP e MQTT sono fra l'altro i protocolli attualmente previsti dallo standard oneM2M).

Per un confronto più dettagliato fra i protocolli suddetti si rimanda a [42].

3.3 COAP

COAP (COnstrained Application Protocol) è un protocollo sviluppato dall'IETF, descritto nel documento RFC7252 [43] e pensato per estendere l'architettura REST alle comunicazioni M2M. Come per il protocollo HTTP l'identificazione delle risorse avviene tramite URI (del tipo coap://…) inoltre, come nel caso dell'HTTP, l'accesso alle risorse avviene tramite un meccanismo request-response basato su quattro tipi di richieste (GET per ottenere informazioni su una risorsa, PUT per modificarla, POST per crearla e DELETE per eliminarla).

In particolare COAP fornisce un meccanismo di discovery che può essere utilizzato da altri nodi per conoscere le risorse di un particolare nodo server semplicemente con una richiesta del tipo

```
get coap://SERVERADDR/.well-known/core
```

(NOTA: nell'esempio suddetto il nodo server è identificato come SERVERADDR, in pratica trattasi di un indirizzo IP).

A seguito della richiesta suddetta il nodo server risponderà con un messaggio del tipo

```
2.05 "Content"
</sensors/temperature>;rt="TemperatureC";if="sensor",
</sensors/light>;rt="LightLux";if="sensor"
```

La risposta consiste quindi in una serie di link che identificano le risorse messe a disposizione del nodo (nell'esempio suddetto il nodo dispone di due sensori rispettivamente di luminosità e temperatura).

Un nodo client può accedere alle risorse a seguito della fase di discovery con una seconda richiesta di tipo GET. Ad esempio con

```
get /sensors/temperature
```

il client può richiedere il valore della temperatura monitorata dal nodo server ottenendo una risposta del tipo

```
2.05 "Content"
```

```
24.5C
```

contenente il valore della temperatura acquisita (nell'esempio pari a 24.5 gradi Celsius).

A differenza del protocollo HTTP però il COAP è pensato per dispositivi con scarse risorse computazionali ed aggiunge servizi specifici per M2M.

Una analisi dettagliata dei vantaggi del COAP rispetto al protocollo HTTP è riportata in [44].

Oltre all'interoperabilità con architetture web, un altro vantaggio del COAP è il fatto di utilizzare il protocollo UDP al posto del TCP. Il protocollo UDP ha un overhead e una complessità inferiore rispetto al protocollo TCP (basti pensare alle differenti dimensioni degli header dei due protocolli, rispettivamente pari a 8 byte e 20 byte) e per questo motivo il protocollo UDP maggiormente si sposa con i dispositivi M2M; di contro però il protocollo UDP non garantisce la consegna dei messaggi a destinazione (ha quindi una minore reliability). A ciò però supplisce il COAP il quale prevede un meccanismo di ritrasmissione automatica (ARQ) di tipo stop&wait a back-off esponenziale (utile per il controllo delle congestioni) basato su quattro tipi di messaggi: CON, NON, ACK e RST.

In particolare i messaggi di tipo CON (Confirmable) vengono ritrasmessi fin quando non viene ricevuto un messaggio ACK di conferma prodotto dalla destinazione (si veda la Figura 7).

Per trasmettere informazioni che non richiedono conferma è possibile usare messaggi di tipo NON (Non-Confimable), utili anche per trasmissioni di tipo multicast; infine messaggi di tipo RST (Reset) possono essere utilizzati per reinizializzare una comunicazione a seguito, ad esempio, di situazioni di errore.

L'header introdotta dal COAP è in genere di soli 4 byte che anche se aggiunti agli 8 byte dell'UDP porta ad un overhead complessivo di 12 byte, decisamente inferiore a quello del TCP (pari a 20byte).

Altro notevole vantaggio del COAP è legato al fatto che le risposte possono essere asincrone.

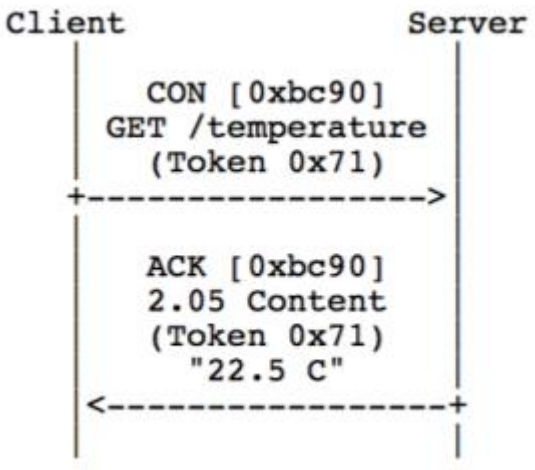


Figura 7 Esempio di GET con risposta sincrona.

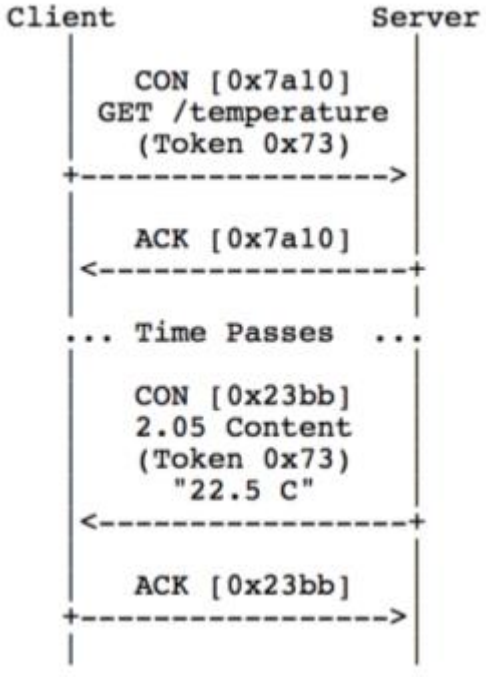


Figura 8 Esempio di GET con risposta asincrona.

Un nodo client può richiedere con un messaggio CON il valore di una temperatura; a tale messaggio il nodo server può rispondere direttamente con un ACK "vuoto" (si veda la Figura 8). In questo secondo caso l'ACK permette al server di informare il client che ha ricevuto il messaggio ma che processerà la richiesta successivamente. In particolare, sempre con riferimento alla Figura 8 il valore di temperatura verrà

trasmesso con un secondo messaggio CON questa volta originato dal server. Il campo Token presente nell'header COAP permette di identificare univocamente la richiesta anche se processata in più fasi.

Inoltre specificando l'opzione "Observe", viene attivato il meccanismo di publish-subscribe per cui il nodo client sarà informato ogni qual volta il valore monitorato subisce una variazione (è previsto l'uso di trasmissioni multicast direttamente dal nodo server, che funge quindi anche da broker).

Altro aspetto importante del COAP è la possibilità di utilizzarlo con il Datagram Transport Layer Security (DTLS) che permette di garantire integrità e confidenzialità nello scambio di messaggi. Va però osservato che l'utilizzo del DTLS aumenta l'overhead sia in termini di risorse computazionali che di risorse energetiche. Inoltre trasmissioni multicast con DTLS non sono possibili (studi in tale direzione sono in corso da parte del gruppo DICE dell'IETF [45]).

È stata infine investigata la trasmissione di messaggi COAP su reti cellulari mediante SMS [46] e via cavo mediante standard seriali RS232/422/485 [47] [48] [49].

Diversi altri aspetti sono stati e/o sono attualmente oggetto di attività di ricerca ad esempio per migliorare l'efficienza del COAP nelle applicazioni real-time e nella trasmissione di file di grosse dimensioni; tali opzioni, sebbene meno usuali, risultano utili in diverse applicazioni come ad esempio per l'aggiornamento dei firmware dei dispositivi, possibile in maniera efficiente per tramite dell'introduzione di un meccanismo di frammentazione a blocchi presenti nel COAP.

A fronte dei notevoli vantaggi si ritiene che il protocollo COAP sia ad oggi il protocollo più idoneo per applicazioni di Advanced Metering Infrastuctures e in particolare per gli scopi del progetto MiSE-ENEA.

3.4 DDS

Il Data Distribution Service (DDS) definisce una architettura middleware standardizzata dalla Object Management Group (OMG) e basata sul paradigma publish-subscribe (di tipo brokerless ovvero peer-to-peer) che combina la definizione di un protocollo per comunicazioni real-time (RTPS) e di una Application Program Interface (API) nota come DCPS.

Pensato per comunicazioni peer-to-peer direttamente fra dispositivi può utilizzare diversi protocolli di trasporto (UDP, TCP oltre che link seriali e architetture shared-memory) ed è disponibile per diversi linguaggi (C/C++, Java, Ada, Ruby ecc.). Integra un meccanismo di discovery ed è in corso la definizione di un framework per la sicurezza a diversi livelli (autenticazione, controllo degli accessi, cifratura, gestione delle chiavi, ecc.). Prevede inoltre oltre venti livelli di QoS capaci di specificare reliability, latenza ecc.

Nonostante i diversi vantaggi del DDS rispetto al COAP in termini di scalabilità, affidabilità, interoperabilità e prestazioni (specie se si considerano applicazioni real-time e big-data) si ritiene che la complessità del DDS sia giustificabile solo per applicazioni real-time e business-critical come il controllo del traffico aereo, servizi di trading finanziario.

3.5 MQTT

MQTT, acronimo di Message Queueing Telemetry Transport, è un protocollo basato sul paradigma publish-subscribe. Originariamente sviluppato dalla IBM nel 1999 per reti e dispositivi constrained. Rispetto al COAP è un protocollo più complesso (in genere per essere supportato occorre che i dispositivi abbiano 64kB di RAM contro i 10kB richiesti dal COAP), presenta infatti 14 tipi di messaggio e tre livelli di Qualità del Servizio (QoS):

- 0 il messaggio viene inviato una sola volta (similmente ai messaggi NON del COAP);
- 1 il messaggio può essere ritrasmesso più volte (similmente ai messaggi CON del COAP);
- 2 il messaggio viene trasmesso esattamente una volta garantendone la consegna (una modalità analoga non è presente nel COAP).

Quest'ultimo livello permette di ottenere una reliability più elevata rispetto al COAP ma a scapito di una maggiore latenza e di un maggiore overhead.

MQTT usa TCP e/o TLS come protocollo di trasporto (il TLS permette di garantire integrità e confidenzialità nel trasporto di messaggi). MQTT prevede, inoltre, un meccanismo di autenticazione in fase di instaurazione di una connessione (per accedere alle risorse vengono richiesti user name e password).

Sebbene MQTT supera COAP in termini di reliability e sicurezza, la sua complessità e il fatto di basarsi sul TCP lo rende meno idoneo per applicazioni M2M basate su nodi di tipo constrained.

Per risolvere in parte tali problemi è stata sviluppata una versione di MQTT per dispositivi resource constrained nota come MQTT-SN [50] che prevede essenzialmente l'utilizzo di un gateway interposto tra i client (nodi sensori) e il broker. Tale soluzione richiede però un compromesso tra complessità e scalabilità. In [51] un confronto fra COAP e MQTT-SN mediante simulazioni in Omnet++ mostra che a parità di condizioni COAP ha un throughput maggiore. Considerato anche che MQTT non prevede un meccanismo di discovery si ritiene che la scelta di COAP sia più indicata per il progetto MiSE-ENEA.

3.6 Conclusioni

In questo Capitolo sono stati analizzati i principali standard e protocolli per comunicazioni M2M. A seguito dell'analisi dello stato dell'arte è stato quindi individuato il COAP come protocollo idoneo per la realizzazione applicazioni M2M per reti di sensori.

4 Stato dell'arte delle reti di sensori

Una rete di sensori wireless (WSN) è essenzialmente composta da una moltitudine di dispositivi, detti *nodi sensori*, capaci di interagire con l'ambiente circostante e di comunicare tra loro al fine ultimo di monitorare e/o controllare un fenomeno fisico [1].

A tal fine tre sono le funzionalità principali che devono essere assolte dai nodi sensori:

- *Sensing:* ovvero la misura di grandezze fisiche (temperatura, umidità, ecc.);
- *Processing:* l'elaborazione delle misure acquisite;
- *Communication:* la comunicazione con altri nodi, tipicamente attraverso interfacce a Radio Frequenza (RF).

La Figura 9 mostra l'architettura tipica di un nodo sensore ovvero le unità che lo costituiscono e le loro interazioni rappresentate mediante uno schema a blocchi; in particolare è possibile notare le unità che implementano le funzionalità suddette ovvero: una unità di elaborazione (*Processing Unit*), tipicamente costituita da una CPU o da un microcontrollore e dalle memorie per dati e programmi; l'unità di comunicazione (*Transceiver*), ovvero un dispositivo di trasmissione radio; l'unità di acquisizione (*Sensing Unit*), costituita da uno o più convertitori analogico digitali (ADC) e dai sensori necessari per misurare le grandezze fisiche da monitorare e che può essere affiancata da una unità di attuazione (non illustrata in figura) in grado di pilotare relè o fornire tensioni di controllo; si noti poi l'unità di alimentazione (*Power Unit*) atta a fornire l'energia necessaria al funzionamento del nodo sensore. Altre unità mostrate nell'architettura di Figura 9 ma non sempre presenti sono il *Mobilizer* e il *Location finding system*: il *Mobilizer* è necessario quando i nodi sensori devono potersi muovere autonomamente; il *Location finding system* serve nel caso in cui la posizione del nodo sensore debba essere rilevata con precisione e può coincidere, ad esempio, con una unità GPS.

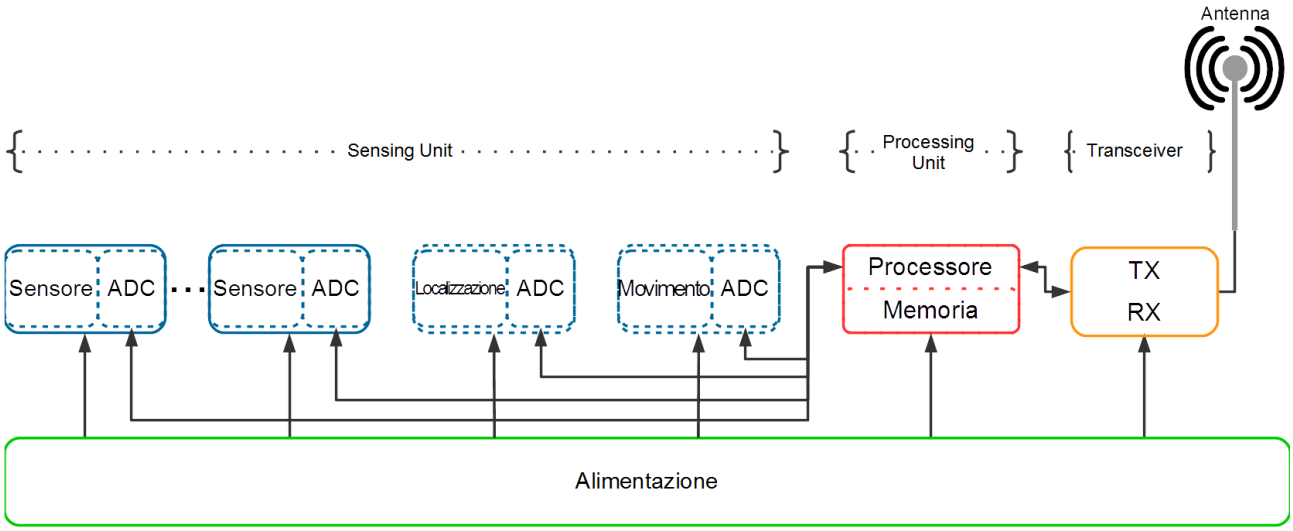


Figura 9 Schema a blocchi nodo sensore.

L'architettura suddetta è realizzata in genere mediante due schede elettroniche: la prima detta *Mote,* si occupa dell'elaborazione e della trasmissione dei dati, ed è pertanto costituita da un microcontrollore e da un transceiver (possono essere due circuiti integrati distinti o, di recente, anche un unico System-on-chip); la seconda scheda, detta *sensor board*, fornisce le interfacce elettriche verso sensori e attuatori.

Alcuni nodi sensori potrebbero però non avere sensor board e svolgere solo funzioni di comunicazione ed elaborazione (si tratta di mote tipicamente utilizzati da router o da relayer per estendere il campo di copertura della rete o per elaborare e aggregare dati di altri nodi).
Al fine di definire la piattaforma più adatta ad applicazioni di acquisizione di dati ambientali per il miglioramento del comfort, il risparmio energetico e l'efficientamento di processi industriali è stato analizzato lo stato dell'arte delle reti di sensori wireless. Un sintesi è riportata nelle sezioni seguenti.

4.1 Nodi sensori

Di seguito sono illustrate le principali piattaforme WSN esistenti in commercio analizzandone le caratteristiche funzionali e gli indici prestazionali. La valutazione delle prestazioni verrà effettuata tenendo conto delle risorse computazionali, quali ad esempio la frequenza dei microcontrollori utilizzati, la quantità di memoria disponibile, i consumi di potenza, potenze di trasmissione e il range di trasmissione radio. Nei prossomi capitoli saranno, invece, analizzati i protocolli e i sistemi operativi per WSN.

Una prima distinzione si può fare tra le piattaforme WSN low-end (anche note come "resource constrained") e high-end [1]. Le piattaforme low-end sono caratterizzate da risorse limitate in termini di capacità di elaborazione, memoria e tipi di interfacce di comunicazione. Queste piattaforme sono adatte a svolgere compiti di rilevamento di parametri fisici oltre a fornire una infrastruttura di interconnessione ed hanno il vantaggio di un basso costo e bassi consumi, sono pertanto le piattaforme da preferire in ambienti industriali dove costi e consumi sono quasi sempre le principali metriche tenute in considerazione nella progettazione.
In alcuni contesti applicativi occorre però una più elevata potenza di elaborazione ed una gestione differente della rete, per esempio quando è necessario connettere la rete di sensori a reti cellulari 4G o quando occorrono funzionalità particolarmente onerose da un punto di vista computazionale come ad esempio la gestione di stream video. Per soddisfare questi requisiti, sono state sviluppate le piattaforme WSN di tipo high-end.
Le piattaforme hardware low-end esaminate nell'ambito del progetto sono le seguenti:
MicaX [52]: sono stati sviluppati inizialmente dalla Berkeley University e sono oggi commercializzati dalla MEMSIC inc. (precedentemente nota come Crossbow). L'unità di elaborazione è un microcontrollore Atmel a 8-bit con una frequenza di clock che varia tra 4 e 16 MHz ed una memoria flash di 128-256KB. La memoria programma è limitata a 4KB mentre la memoria dati è di 512KB. Le piattaforme MicaX differiscono principalmente per il transceiver. In particolare Mica e Mica2 hanno un transceiver CC1000 che trasmette nelle bande ISM a 433MHz o a 916MHz con un data-rate di 40kbps; diversamente la piattaforma MicaZ ha un transceiver CC2420 che implementa lo standard IEEE 802.15.4 (descritto nel Capitolo 5) e trasmette nella banda ISM alla frequenza di 2,4GHz e con un data-rate di 250kbps. I mote delle piattaforme suddette dispongono di un connettore di espansione a 51-pin al quale possono essere collegate facilmente una vasta gamma di sensor board con vari tipi di sensori (di temperatura, umidità, luminosità, pressione, acustici, accelerometri, GPS, ecc.). Tale connettore rende inoltre accessibili ingressi analogici e digitali del microcontrollore oltre che le interfacce seriali I2C, SPI e UART.
Cricket [52]: I mote Cricket sono simili ai Mica2 ma in più includono un transceiver ad ultrasuoni utile per applicazioni di telerilevamento.
IRIS [52]: I mote IRIS prodotti da Memsic inc. sono una evoluzione dei MicaZ. La CPU è un microcontrollore Atmel a 8-bit (ATMega128l) con una frequenza di clock di 16MHz ed una memoria programma di 8KB. Il transceiver è un Atmel RF230 che implementa lo standard IEEE 802.15.4 e trasmette nella banda ISM alla frequenza di 2,4GHz e con un data-rate di 250kbps. Microcontrollore e transceiver si trovano in un unico dispositivo ad alta integrazione e questo permette ai mote Iris di avere dimensioni e consumi ridotti rispetto ad altri mote. Gli IRIS hanno una potenza massima di trasmissione maggiore rispetto a quella dei MicaZ che si riflette in un range trasmissivo maggiore (50m indoor e 300m outdoor dell'IRIS contro i 30m indoor e 100m outdoor dei MicaZ). Anche i mote IRIS hanno il connettore a 51-pin, dunque tutte le sensor-board sviluppate per i MicaZ sono compatibili.

Figura 10 Mote IRIS.

LOTUS [52]: La piattaforma Lotus, sempre prodotta da Memsic inc., ha un microcontrollore Cortex M3 con architettura ARM a 32-bit con frequenza di clock da 10 a 100MHz, una memoria programma di 512KB e una RAM di 64KB. Il transceiver è l'Atmel RF231, lo stesso presente negli IRIS. Come l'IRIS ha un connettore di espansione a 51-pin.

TelosB [52]**/ Tmote Sky** [53]**/ CM5000** [54]: I mote TelosB e Tmote Sky prodotti rispettivamente da Memsic inc. e Sentilla sono simili. Rispetto ai mote MicaZ hanno lo stesso transceiver, ma una RAM più grande (10KB) e sono basati su un microcontrollore della Texas Instruments a 8MHz (TI MSP430). In entrambe le piattaforme i Mote hanno svariati sensori già integrati (IR, luminosità, umidità e temperatura) e due connettori rispettivamente da 6 e 10 pin per interfacciare altri sensori. È inoltre presente una interfaccia USB che ne permette una facile connessione a sistemi di elaborazione convenzionale (PC, notebook, tablet ecc.). I mote CM5000 sviluppati da Advanticsys sono completamente compatibili con la piattaforma TelosB (stessa famiglia di microcontrollore MSP430, stesso transceiver CC2420 e stessi sensori); la memoria programma è però di 48KB e prevede la possibilità di aggiungere una memoria flash esterna di 1MB.

Figura 11 Tmote Sky.

Figura 12 TelosB mote.

EYES [55]: I mote della famiglia EYES sono il risultato di un progetto europeo omonimo della durata di tre anni e sono simili alla famiglia TelosB/Tmote Sky. I mote hanno un microcontrollore a 16-bit con 60kB di memoria programma e 2kB di memoria dati [56]. I mote non hanno sensor board separate, ma alcuni sensori sono integrati sul mote stesso (accelerometro, sensore di temperatura, umidità, luminosità, di pressione). Il transceiver è il TR1001 che lavora nella banda ISM a 868MHz e supporta un rate di trasmissione di 115,2 kbps. L'interfaccia di programmazione è seriale RS232.

SHIMMER [57]: La piattaforma SHIMMER (v3) utilizza un microcontrollore della Texas Instruments MSP430 a 24MHz. Trasmette ad una frequenza di 2,4GHz con protocollo Bluetooth 2.1+ e bit rate massima 3Mbps. È una piattaforma con diversi sensori sviluppata prevalentemente per il monitoraggio dei parametri biomedicali e delle prestazioni sportive ma può essere utilizzata anche per monitoraggio ambientale. In particolare ha un accelerometro, sensori che rilevano i parametri biofisici e sensori ambientali.

TinyNode+ [58]: La piattaforma TinyNode ha un microcontrollore MSP430 della Texas Instruments. Il transceiver è Xemics (XE1205 o SX1211) e trasmette alle frequenze 868 o 915 MHz [59]. La piattaforma è sviluppata specificatamente per applicazioni "parking-related", come ad esempio il controllo degli ingressi

nei parcheggi. La gamma di sensori a disposizione è orientata al solo campo applicativo appena menzionato e la piattaforma non ha flessibilità per essere usata in altre applicazioni. Non si ritiene, dunque, adatta al progetto in oggetto.

Figura 13 Mote BTnode.

BTnode [60]: La BTnode è una piattaforma sviluppata dal *Computer and Engeneering and Networks Laboratory* e dal *Research Group for Distributed Systems* di Zurigo per scopi di ricerca nell'ambito delle reti MANET. La piattaforma è basata su un microcontrollore Atmel ATMega128 a 8-bit e 8 MHz di frequenza di clock. Ha 64 KB di RAM, 128 KB di memoria FLASH ROM e 4 KB di EEPROM. Integra un transceiver Chipcon CC1000 che trasmette nella banda ISM 433-915 MHz e un transceiver Zeevo ZV4002 (bluetooth v.1.2) che può semplificare la comunicazione con smartphone. La BTnode è stata utilizzata in progetti di ricerca come NCCR [61] e Smart-Its [62]. Ha un connettore di espansione a 40-pin che rende disponibili UART, SPI, I2C, GPIO, ADC, Timer e Led. Si ritiene che la soluzione del doppio transceiver possa essere importante per rendere la piattaforma più versatile aggiungendo la possibilità di collegarsi con altri dispositivi, come ad esempio gli smartphone. Tuttavia, la versione del bluetooth utilizzata in questo dispositivo (v2.1) non è low-energy, dunque al vantaggio di una maggiore versatilità si affianca lo svantaggio ben più rilevante dell'eccessivo consumo di potenza.

EZ430-RF2480/RF2500T [63]**:** Le piattaforme EZ430-RF2480 e RF2500T sono prodotte dalla Texas Instruments ed hanno un microcontrollore TI MSP430 a 16-bit e 16MHz di frequenza di clock. Presentano una RAM di 1KB, la più piccola rispetto a tutte le piattaforme presentate fino ad ora, ed una memoria Flash di 32KB. Il transceiver è il Chipcon CC2500 che trasmette nella banda ISM a 2,4GHz ad una velocità di 250kbps.

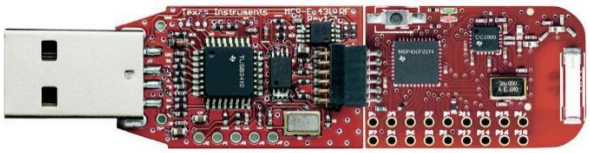


Figura 14 Mote Texas Instruments EZ430.

Waspmote [64]**:** Una piattaforma WSN di recente sviluppo è la Waspmote prodotta nel 2013 dalla Libelium. La Waspmote ha un microcontrollore ATMega128 a 8-bit e 16MHz di frequenza di clock, lo stesso microcontrollore della piattaforma Iris. Oltre ad un transceiver Xbee-802.15.4 che trasmette nella banda ISM a 2,4GHz con velocità 250kbps supporta però differenti tecnologie radio come: 3G/GPRS, WiFi, RFID/NFC/Bluetoothe anche una interfaccia ModBus utile per power metering.

Figura 15 Waspmote.

NI WSN3202 [65]**:** la piattaforma NI WSN3202 è prodotta dalla National Instruments ed è programmabile con l'utilizzo di LabView. Si basa su un microcontrollore MSP430 e un transceiver IEEE 802.15.4 ed ha una memoria Flash di 248KB. Il principale svantaggio di tale mote è l'elevato costo in confronto agli altri mote disponibili a cui si aggiungono i costi della licenza del software Labview.

OpenMote [66]: È una piattaforma sviluppata dalla OpenMote Technologies. I mote OpenMote-CC2538 sono basati su un SoC che integra un microcontrollore Cortex-M3 a 32-bit e 32MHz di frequenza di clock e un transceiver simile al CC2520 (compatibile quindi con lo standard IEEE 802.15.4-2006). Il mote ha 32KB di RAM e 512 di Flash ed interfacce di tipo GPIO, Timer e ADC. Associato al modulo OpenBattery (che fornisce l'alloggiamento per le batterie di alimentazione), dispone di un sensore di temperatura e umidità, un sensore di luminosità e un accelerometro (tutti interfacciati con la I2C). Infine abbinato alla OpenBase permette la realizzazione di BaseStation (con funzionalità per la programmazione e la possibilità di interfacciare i mote a PC oltre che direttamente a reti Ethernet a 10/100Mbps).

Figura 16 OpenMote-CC2538, OpenBattery e OpenBase.

Z1 [67]**:** La piattaforma Z1 sviluppata da Zolertia è stata progettata con l'intento di mantenere la compatibilità con la famiglia di mote Tmote, migliorandone però le prestazioni e la flessibilità in termini di sensor board disponibili. È basata su microcontrollore MSP430 ed il transceiver è il CC2420 (trasmette alla frequenza di 2,4GHz ed ha una velocità di trasmissione di 250kbps). Ha una memoria RAM di 8KB ed una Flash di 92KB. Integra inoltre un sensore di temperatura, un accelerometro a 3-assi e un connettore a 52-pin di espansione per altri sensori che rende disponibili UART, SPI, I2C, GPIO, ADC, DAC, USB, Timer. Supporta diversi Sistemi Operativi come TinyOS, Contiki, RIOT, e OpenWSN.

Per un più rapido confronto le principali caratteristiche delle piattaforme su menzionate sono riportate in 1. In Figura 10 è riportata inoltre una timeline con l'anno di introduzione sul mercato delle varie piattaforme appena analizzate.

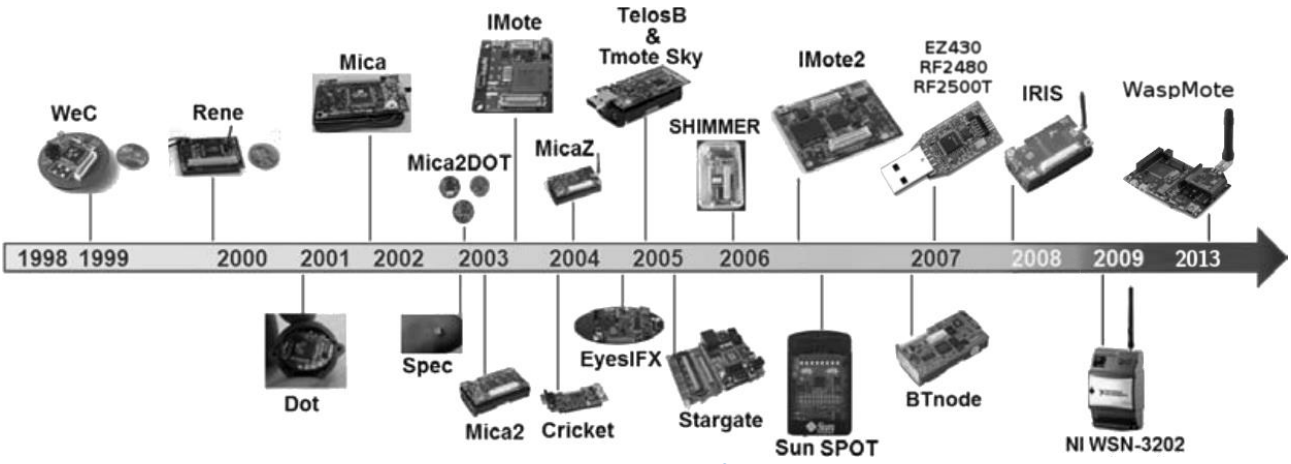


Figura 17 Principali piattaforme WSN.

Si precisa che nell'ambito dell'attività di ricerca sono state analizzate diverse altre piattaforme low-end non riportate nella Tabella 2 e non discusse nel presente report perché simili a quelle appena analizzate e/o con prestazioni inferiori.

Tabella 2 Piattaforme Mote [1] [68].

Tipo di Mote	Frequenza CPU (MHz)	Memoria Programma (KB)	RAM (KB)	Frequenza Radio	Velocità di trasmissione (kbps)
Mica	6	128	4	868 MHz	40
Mica2	16	128	4	433/868/916 MHz	76,8
MicaZ	16	128	4	2,4GHz	250
IRIS	16	128	8	2,4GHz	250
Cricket	16	128	4	433 MHz	76,8
LOTUS	10-100	512	64	2,4GHz	250
EYES	8	60	2	868 MHz	115
SHIMMER3	24	256	16	2,4GHz/BT[1]	921
TelosB/Tmote/ CM5000	16	48	10	2,4GHz	250
Sun SPOT	133-400	Up to 4096	Up to 1024	2,4GHz	250
BTnode	8	128	64	433-915/BT[1]	76,8/721
NI WSN-3202	8	128	188/248	0.9/2,4GHz	250
Imote	12	512	64	2,4GHz (BT)	100
Imote2	13-416	32000	256	2,4GHz	250
TI EZ430	16	32	1	2,4GHz	250
Waspmote	16	128	8	2,4GHz	250
Z1	16	92	8	2,4GHz	250
OpenMote	32	512	32	2,4GHz	250

[1] *BTnode e SHIMMER hanno un transceiver bluetooth*

Per quanto concerne lo stato dell'arte delle piattaforme high-end, anche note come Multimedia Wireless Sensor Network (MWSN) [69], si differenziano dalle precedenti poiché hanno risorse computazionali nettamente superiori per permetterne l'utilizzo nelle in scenari applicativi che necessitano l'elaborazione di segnali audio e video. Le principali piattaforme high-end presenti sul mercato sono di seguito brevemente descritte.

Stargate [52]: ha un microprocessore PXA-255 Xscale a 400MHz con 64MB di RAM e 32 di Flash. Il processore è in grado di effettuare algoritmi avanzati di elaborazione delle immagini. La piattaforma supporta un S.O. Linux e due differenti moduli radio: IEEE 802.11 (WiFi) e IEEE 802.15.4.

Imote2 [70]*:* è una piattaforma progettata in modo specifico per applicazioni WSN che richiedono alte prestazioni della CPU/DSP. Il processore è un Intel PXA271 Xscale (supporta frequenze di clock da 13MHz a 624MHz) e dispone di una memoria SRAM di 256KB, di una memoria Flash da 32MB e di una SDRAM da 32MB. Il transceiver è un CC2420 (IEEE 802.15.4) con antenna integrata.

CMUCam3 [71]: è un computer embedded con processore ARM7TDMI. Ha inoltre un processore dedicato Philips LPC2106 collegato ad una videocamera CMOS Omni-Vision. È anche dotata di librerie per la compressione JPEG e per l'elaborazione delle immagini basate sul linguaggio LUA, un linguaggio di scripting leggero che ne rende più facile la programmazione.

SunSPOT [72]: La Sun SPOT, il cui nome per esteso è Sun Small Programmable Object Technology, originariamente prodotta dalla Sun Microsystems (successivamente acquisita da Oracle). Ha una scheda madre, eSPOT, con processore ARM9 (Atmel AT91SAM9G20) a 32-bit e con frequenza di clock massima di 400MHz. Il sistema operativo supportato è Squawk VM, essenzialmente una Java Virtual Machine. Il transceiver è il CC2420. Alla scheda eSPOT vengono collegate le sensor board, ribattezzate dalla Sun come "application-board". La Sun SPOT è dotata di batterie ricaricabili attraverso interfaccia USB. Ha una memoria ROM interna di 64KB e due moduli SRAM da 16KB.

MeshEye [73]: è una piattaforma adatta ad applicazioni di videosorveglianza ed è in grado di rilevare, attraverso un sistema video stereo e a bassa risoluzione, la posizione degli oggetti anche in movimento, oltre che la loro dimensione. Ha un processore ARM7TDMI a 32-bit e 47,92 MHz; 64MB di SRAM e 256KB di Flash; il transceiver è il CC2420.

WiCa [74]: è un nodo con alte prestazioni dotato di un processore video dedicato IC3D a 80MHz che si interfaccia con una videocamera VGS a 24-bit. Supporta una SRAM on-chip di 10MB (4 frame VGA).

Cyclops [75]: è un piccolo dispositivo con videocamera sviluppato per WSN e compatibile con i nodi MicaZ e Mica2. Ha una videocamera Agilent ADCM-1700 CMOS, una FPGA Xilinx e un microcontrollore ATMega128.

La maggiore potenza di elaborazione delle piattaforme high-end ha, come conseguenza immediata, un maggior costo dei dispositivi e maggiori consumi di energia. Spesso tali dispositivi per garantire una autonomia superiore alle 24 ore necessitano di una alimentazione di rete. Risultano quindi meno flessibili dal punto di vista dell'installazione e del riposizionamento dei nodi rispetto alle piattaforme low-end. Si ritiene pertanto che tali dispositivi debbano essere utilizzati al più come gateway o dove sia di principale interesse l'acquisizione di segnali video. Diverse piattaforme high-end sono inoltre recentemente uscite fuori produzione (es. SunSPOT). Tenuto conto anche degli scenari applicativi specifici del progetto, ovvero della necessità di minimizzare i consumi energetici, si ritiene che l'utilizzo di processori con frequenze di clock superiori ai 100MHz non solo non sia necessario ma sia anche controproducente per gli scopi del progetto. Pertanto si è deciso di approfondire l'analisi ai soli mote low-end.

4.2 Analisi comparativa piattaforme low-end

Nei mote low-end le risorse sono limitate e ciò aumenta notevolmente l'effort necessario per garantire le prestazioni desiderate soprattutto in termini di affidabilità e robustezza. Il termine "constrained" è spesso utilizzato in tali contesti in alternativa al termine "low-end" in riferimento ad uno o più parametri che devono essere tenuti in considerazione in tutte le fasi di progettazione e i cui vincoli sono prioritari per garantire che la rete WSN abbia le prestazioni desiderate. In particolare, ci si riferisce a WSN energy-constrained quando il parametro che pone il vincolo principale è il consumo di potenza, oppure WSN cost-constrained quando l'applicazione per cui è progettata la rete di sensori necessita di un cospicuo numero di nodi ed il costo diventa un parametro da ottimizzare.
Al fine di ottenere il giusto compromesso tra consumi, costi e prestazioni occorre quindi una accurata scelta dell'hardware: un microcontrollore o una CPU che abbia una frequenza di clock più alta garantirà una

elevata capacità computazionale ma avrà un maggiore consumo di potenza di un altro microcontrollore che funzioni a frequenza più bassa; d'altra parte con una minore velocità di elaborazione occorrerà più tempo per eseguire un determinato algoritmo e questo può comportare dei seri limiti alle applicazioni real-time (necessarie nei sistemi di controllo industriale). Inoltre, la quantità di memoria programma disponibile (spesso legata anche alle prestazioni della CPU) pone un vincolo allo sviluppo delle applicazioni (in termini di complessità); infine la quantità di memoria dati disponibile limita la quantità di parametri che sarà possibile memorizzare sul dispositivo. A ciò si aggiunge la necessità di valutare attentamente la scelta dei moduli di trasmissione radio poiché da questi dipende il consumo di potenza oltre che l'interoperabilità con altre reti.
Per i motivi suddetti si ritiene importante riportare un confronto delle piattaforme low-end illustrate al Paragrafo 4.1. I confronti sono effettuati sulla base dei seguenti parametri prestazionali (KPI, Key Performance Indicators):

- capacità computazionale;
- costi e dimensioni;
- moduli di trasmissione.

4.3 Capacità computazionale

La Tabella 3 riassume in modo sintetico le caratteristiche dei microprocessori alla base dei mote low-end precedentemente analizzati permettendone un confronto in termini di capacità di elaborazione e quantità di memoria disponibile.
È possibile osservare che tre sono le CPU ricorrenti nei mote low-end: ATMEGA128, MSP430 e Cortex M3. I mote basati sull'ATMEGA128 (es. Waspmote, Iris) hanno oltre che lo stesso microcontrollore anche la stessa memoria programma (128kB) e dunque possono considerarsi equivalenti in termini di capacità di elaborazione. Il secondo gruppo di piattaforme, basate su microcontrollori MSP430 (es. TelosB, Z1, EZ430), hanno prestazioni simili alle precedenti in termini di frequenza di clock ma differiscono notevolmente in termini di memoria disponibile (in genere hanno una memoria flash di capacità inferiore ma una memoria dati superiore). Infine le piattaforme basate su processori Cortex M3 (OpenMote e Lotus) sono quelle che hanno una maggiore frequenza di clock e maggiori risorse di memorizzazione.

Tabella 3 Piattaforme WSN - capacità computazionale.

Tipo di mote	Costruttore	Microcontrollore	Frequenza	Memoria Programma	Memoria Dati (RAM)
EYES	*Infineon/University of Twente*	*TI MSP430*	*8MHz*	60K	2k
BTnode	Btnode	ATMEGA128L	*8MHz*	128K	64+180K
MicaZ	Memsic inc.	ATMEGA128L	*16MHz*	128K	4K
TelosB/Tmote Sky/ CM5000	Memsic inc./Moteiv/Advanticsys	*TI MSP430F1611*	*8MHz*	48K	10K
IRIS	Memsic inc.	ATMEGA1281	*16MHz*	128K	8K
Cricket	Memsic inc.	ATMEGA128L	*8MHz*	128K	4K
LOTUS	Memsic inc.	NXPLPC1758 (Cortex-M3)	*10-100MHz*	512K	64K
NI WSN3202	National Instriments	*TI MSP430*	*8MHz*	120K	188K/248K

EZ430-F2500T	Texas Instruments	*TI MSP430F2274*	*16MHz*	32K	1K
Waspnode	Libelium	ATMEGA128	*16MHz*	128K	8K
Z1	Zolertia	*TI MSP430F2617*	*16MHz*	92K	8K
Shimmer	Shimmer	*TI MSP430*	*24MHz*	256K	16K
OpenMote	OpenMote Tecnologies	Cortex-M3	*32MHz*	512K	32K

4.4 Costi e dimensioni dei mote

In Tabella 4 sono riportati peso, dimensioni e prezzo dei mote low-end precedentemente illustrati [76].
Per quanto concerne le dimensioni le varie piattaforme possono essere considerate equivalenti, fatta eccezione per i WSN3202 le cui dimensioni sono decisamente maggiori poiché già dotati di involucro (il che giustifica in parte il costo maggiore).
Per quanto concerne i costi, gran parte dei mote si trovano nella fascia di prezzo 80-150€. Al di sotto di tale fascia si trova solo la piattaforma EZ430 della Texas Instruments, mentre all'altro estremo si trovano i mote Shimmer e NI WSN3202 (questi ultimi con un costo di circa 900€, superiore quasi di un ordine di grandezza rispetto ai precedenti).

Tabella 4 Piattaforme WSN- dimensioni, peso e cost.

Piattaforma WSN	Dimensioni (mm)	Peso (g)	Costo (Euro)
EZ430	80,5*29,5*11	31	52
MicaZ	58*32*7	18	88
TelosB/TmoteSky	65*31*6	23	85
CM5000	81*32*55	17,5	90
Z1	56,8*34.5*7	N/A	95
OpenMote	60*32*7	N/A	100
BTnode	58*32*7	18	98
IRIS	58*32*7	18	103
Waspnodes	73,66*51*13	20	155
Cricket	58*32*7	18	200
LOTUS	76*34*7	18	268
Shimmer	51*34*14	23,6	359
NI WSN3202(*)	235*84*38	256	924

(*) il peso e le dimensioni della WSN 3202 sono riferiti al nodo comprensivo di involucro

4.5 Moduli radio

Le piattaforme WSN sviluppate negli ultimi decenni sono caratterizzate da una varietà di transceiver radio. Tale diversificazione rende necessario un confronto in termini di caratteristiche trasmissive come: banda, bit-rate, potenza massima in trasmissione, sensitività, range di copertura.

In Tabella 5, vengono riportati range di frequenza di trasmissione e data rate dei dispositivi radio utilizzati dai principali mote low-end.

Come è possibile osservare molti dei mote sono basati sullo standard IEEE 802.15.4 (che sarà discusso nel Capitolo 5) e che prevede essenzialmente tre bande di utilizzo: 868MHz, 915MHz e 2.4GHz. Solo alcuni mote (es. Cricket) lavorano con frequenze inferiori (433MHz).

Le bande suddette sono dette bande ISM (acronimo di Industrial, Scientifical and Medical, poiché pensate per applicazioni in tali ambiti) e s differenza di altre bande non necessitano di costi aggiuntivi legati all'acquisto di licenze per il loro utilizzo. Da qui la preferenza di queste bande nelle WSN (così come per altre tecnologie wireless, es. WiFi).

La scelta della banda ISM più congeniale per una data applicazione deriva in genere da un compromesso fra range di copertura, banda (e quindi massima data-rate), dimensioni dei dispositivi e immunità alle interferenze.

Come regola generale, nell'ipotesi ti utilizzare antenne isotrope e a parità di potenza in trasmissione, per ottenere un maggiore range di copertura sono preferibili frequenze più basse (ovvero lunghezze d'onda più alte, essendo $\lambda = \frac{c}{f}$).

Tale risultato può essere facilmente giustificato dalla formula del collegamento radio (formula di Friis) che permette di calcolare la potenza in ricezione (P_r) in condizioni di spazio libero e di campo lontano data la potenza in trasmissione (P_t), la distanza fra trasmettitore e ricevitore (d) e la lunghezza d'onda (λ) del segnale trasmesso:

$$P_r = \frac{P_t \cdot \lambda^2}{(4\pi \cdot d)^2} G_t G_r$$

Di contro la banda che può essere allocata su frequenze più alte è in genere maggiore permettendo così velocità di trasmissione più elevate. Inoltre la disponibilità di una banda maggiore permette l'utilizzo di tecniche di spread spectrum utili a ridurre l'effetto di interferenze. Infine frequenze più alte permettono di ridurre le dimensioni delle antenne. Per tutti questi motivi la banda a 2.4GHz è in genere quella preferita (potendo sopperire all'unico inconveniente di un ridotto range di trasmissione con tecniche multihop).

La banda ISM nell'intorno dei 2.4GHz ha inoltre l'ulteriore vantaggio di essere l'unica disponibile a livello mondiale (viceversa la banda a 868MHz non è disponibile in USA e la banda a 915MHz non è disponibile in Europa).

Alla luce di tali valutazioni la banda ISM a 2.4GHz è da ritenersi preferibile per gli scopi del progetto. Va però detto che in tale banda sono presenti diverse tecnologie di comunicazione wireless (WiFi, Bluetooth, ecc.) che possono generare interferenze tali da compromettere il funzionamento delle WSN. In particolare mentre può essere prevista la co-presenza di reti WSN IEEE802.15.4 e di reti WLAN IEEE802.11g (ovvero le reti WiFi attualmente più comuni, a 54Mbps) non altrettanto può dirsi per le reti WLAN IEEE802.11n il cui spettro si sovrappone completamente con quello delle WSN. La tipologia di reti WLAN presenti va quindi analizzata prima della installazione di una rete WSN.

Tabella 5 Confronto tipi di chip radio e frequenze di trasmissione.

Tipo di mote	Chip Radio	Range di Frequenze	Bit Rate (Kbps)
BTnode	CC1000	433-915 MHz	76,8
EYES	TR1001	868 MHz	115
MicaZ	CC2420	2.4 GHz	250
TelosB/Tmote Sky/CM5000	CC2420	2.4 GHz	250
IRIS	RF230	2.4 GHz	250
Cricket	CC2420	433 MHz	76,8
LOTUS	RF231	2.4 GHz	250
EZ430-F2500T	CC2420	2.4 GHz	250
Waspmote	Xbee-802.15.4	2.4 GHz	250
Z1	CC2420	2.4 GHz	250
OpenMote	CC2520	2.4 GHz	250

Nella Tabella 6 Moduli radio supportati dai mote e loro caratteristiche tecniche.sono elencati i dispositivi radio che trasmettono nella banda a 2,4GHz e implementano lo standard IEEE 802.15.4. Come è possibile osservare il più diffuso ed utilizzato è il CC2420. Va però detto che la recente introduzione di dispositivi SoC (System on Chip) ha permesso di integrare in un unico chip microcontrollore e transceiver riducendo così le dimensioni e i consumi di potenza. È il caso del dispositivo AT RF230 prodotto da Atmel e supportato dalla piattaforma Iris, che rispetto al più comune CC2420 ha prestazioni migliori, come è possibile osservare dalla Tabella 6 dove sono riassunti i dati dei transceiver wireless CC2420, RF230 e di altri transceiver nella banda ISM a 2.4GHz utilizzati dai principali mote.

Dai dati riportati dalla Tabella 6 è evidente che i mote IRIS sono da preferire in termini di consumi energetici (in particolare hanno un assorbimento di corrente in modalità sleep/idle/power down nettamente inferiore rispetto agli altri dispositivi, permettendo così di ridurre i consumi mediante tecniche di duty-cycling, si veda il Capitolo 8). Risultano invece decisamente elevati i consumi dei Waspmote in ricezione e trasmissione (circa il triplo degli altri nodi).

Nell'ultima riga della Tabella 6 è anche riportato il massimo range di copertura dei mote, ovvero la massima distanza fra due nodi dotati di antenne isotrope, ricavato a partire dai dati di sensitività mediante le seguenti relazioni:

$$P_{TX}(dBm) - S(dBm) = L_p(dB) + M(dB)$$

$$L_p(dB) = 20 \cdot \log_{10}\left(\frac{4 \cdot \pi \cdot d_0}{\lambda}\right) + 10 \cdot \alpha \cdot \log_{10}\left(\frac{d}{d_0}\right)$$

Nelle espressioni suddette P_{TX} è la potenza in trasmissione (espressa in dBm), S è la sensitività del ricevitore (in pratica la minima potenza apprezzabile dal ricevitore, espressa in dBm), M (espresso in dB) detto margine di fading è un margine introdotto per garantire una certa qualità del segnale in ricezione, (in pratica S+M coincide con la potenza P_{RX} desiderata in ricezione), L_p sono le perdite di percorso (espresse in dB), d_0 è una distanza di riferimento (tipicamente pari a 1m per trasmissioni indoor e 8m per trasmissioni outdoor) mentre α è un coefficiente detto esponente delle perdite di percorso e tipicamente compreso tra 1,5 e 6 (pari a 2 in spazio libero).

Il modello suddetto mostra come a parità di potenza trasmessa una minore sensitività permetta di ottenere un range di copertura maggiore. In particolare in spazio libero il range di copertura raddoppia ogni 6dB di sensitività.

Sulla base di tali considerazioni e dei dati riportati dalla Tabella 6 è possibile asserire che i mote IRIS e EZ430, avendo una sensitività migliore, sono da preferire in termini di range di copertura.

In particolare nelle ultime due righe della Tabella 6 sono riportati i range di copertura dei diversi mote ricavati assumendo M=15dB e due set di parametri corrispondenti alle condizioni di propagazione in spazio libero (d_0=1m e α=2) e ad un modello più realistico (d_0=8m e α=3,3) definito dallo standard IEEE802.15.4 [77] (che, come si vedrà nel prossimo Capitolo, rappresenta uno degli standard più comuni per l'implementazione di WSN). Si precisa che come potenza trasmessa si è assunto 0dBm per tutti i mote tranne che per gli IRIS per i quali si è considerata un potenza di 3dBm per fare un confronto a parità di consumi con altri mote.

Tabella 6 Moduli radio supportati dai mote e loro caratteristiche tecniche.

	TelosB/Tmote Sky/CM5000/ MicaZ	IRIS	EZ430-F2500T	Waspmote	EZ430-F2480	OpenMote
Radio Chip	CC2420	RF230	CC2500	Xbee-802.15.4	CC2480	CC2520
Data Rate (kbps)	250	250	250	250	250	250
Banda ISM	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz
Corrente in ricezione (mA)	18,8	15,5	16,6	50	26,7	20
Corrente in trasmissione (mA)	17,4 (0dBm)	17,5 (3dBm)	21,2 (0dBm)	45 (0dBm)	26,9 (0dBm)	24 (0dBm)
Consumo di Potenza in Ricezione (mW)	56,4	46,5	49,8	165	80,1	66
Consumo di Potenza in Trasmissione (mW)	52,2	52,5	63,6	148,5	80,7	79,2
Consumo di Potenza in Sleep/Idle/powe-down (mW)	1,28	60nW	1,2	33pW	1,5	1,23mW
Modulazione	O-QPSK	O-QPSK	OOK, 2_FSK, GFSK	PWM	O-QPSK	O-QPSK
Sensitività (dBm)	-95	-101	-108	-92	-92	-97
Range 802.15.4 (m)	36	68	90	30	30	42
Range ideale (m)	97	275	435	69	69	123

Di seguito è riportato un semplice foglio di calcolo che permette di determinare il range di copertura di un mote sulla base del modello suddetto.

	A	B	C	D
1	**Parametro**	**Simbolo**	**Valore**	**Unità di misura**
2	Potenza in trasmissione	Ptx	0	dBm
3	Sensitività	S	-95	dBm
4	Margine di fading	M	15	dB
5	Perdite di percorso ammissibili	Lp	80	dB
6	Lunghezza d'onda	lambda	0,12244898	m
7	Distanza di riferimento	d0	8	m
8	Perdite di percorso alla distanza di riferimento	L(d0)	58,28248912	dB
9	Coeffiente delle perdite di percorso	alfa	3,3	
10	Distanza massima = C7*10^((C5-C8)/(10*C9))	d	36,40796268	m

Figura 18 Scenario di propagazione outdoor relativo alle misure di Tabella 7.

A verifica del modello suddetto sono state realizzate delle misure sperimentali con mote Iris e TelosB inviando a varie distanze 1000 pacchetti e misurando il numero di pacchetti persi. La Tabella 7 riporta la percentuale di pacchetti persi mediata su 10 esperimenti al variare della distanza ottenuta per lo scenario di propagazione illustrato in Figura 18.

Tabella 7 Misure sperimentali.

Distanza (m)	% Pacchetti persi	
	IRIS	TelosB
36m	0.0%	0.0%
50m	0.0%	0.1%
65m	0.1%	91.4%

Come è possibile osservare le misure sperimentali sono in accordo con il modello teorico adottato.

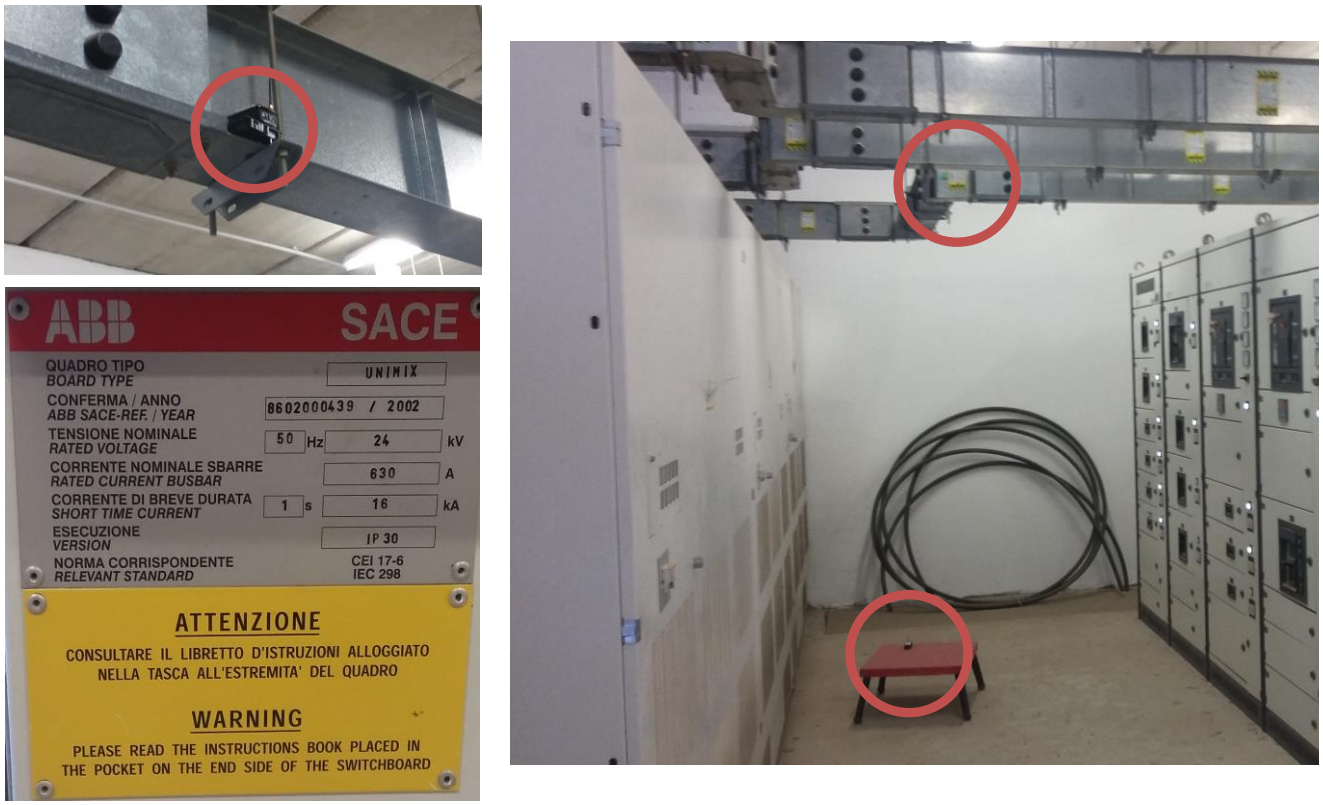


Figura 19 Cabina di media tensione (24kV) c/o Dipartimento di Ingegneria.

Nell'ambito dell'attività sono state effettuate molteplici misure in differenti scenari di propagazione. In particolare, è stata validata la possibilità di utilizzare le WSN anche in ambienti industriali, tipicamente "ostili" alle comunicazioni wireless, ovvero in presenza di strutture metalliche oltre che di campi elettromagnetici prodotti ad esempio da trasformatori a media ed alta tensione (si veda la Figura 19) e da motori elettrici (si veda la Figura 20).

Le campagne di misura hanno avuto la durata minima di un'ora e sono state realizzate con un rate di trasmissione di un pacchetto al secondo al fine di collezionare almeno 3600 pacchetti per ogni esperimento. Anche negli scenari suddetti la perdita di pacchetti è risultata sempre inferiore allo 0.1%.

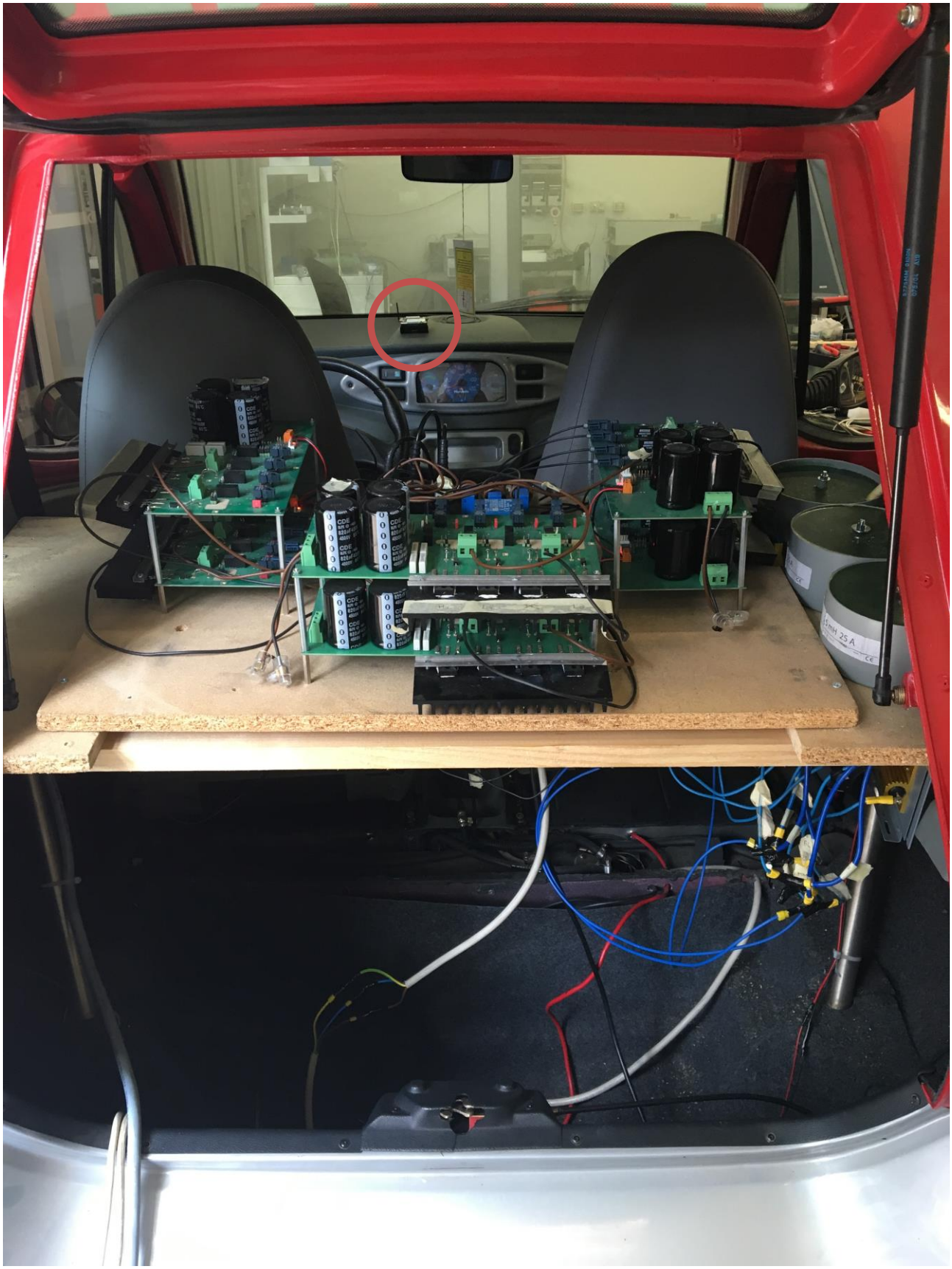


Figura 20 Macchina elettrica presso laboratorio di Elettronica di Potenza del Dipartimento di Ingegneria (Motore Magnetic BLQ 65 P 47 sincrono a magneti permanenti 150/4700RPM e inverter sperimentale a 10kHz).

4.6 Conclusioni

Non essendoci mote rivelatisi superiori per tutte le metriche analizzate, si è ritenuto di dover legare la scelta della piattaforma WSN principalmente all'applicazione da sviluppare. Si precisa inoltre che le considerazioni di seguito riportate in questo report e in particolare le valutazioni sulle piattaforme WSN esaminate non hanno la finalità di escludere piattaforme perché non efficienti ma solo l'obiettivo di selezionare quelle ritenute più adatte al campo applicativo di interesse per il progetto.

Tali considerazioni saranno accompagnate nei capitoli successivi da altre valutazioni inerenti altri elementi architetturali non ancora esaminati (in particolare i protocolli di rete supportati e i sistemi operativi). Solo con uno sguardo attento a tutti gli aspetti architetturali è stato possibile individuare la piattaforma più idonea per il progetto.

Partendo dall'analisi dei costi dei mote (Tabella 4), sebbene la piattaforma TI EZ430 risulti economicamente la più conveniente, non si ritiene idonea per il progetto a causa delle limitate risorse in termini di memoria (solo 1KB di RAM) totalmente insufficienti per gli scopi del progetto che richiede fra l'altro l'utilizzo di protocolli M2M difficilmente implementabili con meno di 10kB di memoria. Si ritiene invece proibitivo il costo dei mote NI WSN3202 e Shimmer.

Per quanto concerne i transceiver (Tabella 6) come si è detto è preferibile l'utilizzo della banda ISM a 2.4GHz il che porta ad escludere i mote EYES, Cricket e BTNode. Sempre con riferimento alla Tabella 6 considerato la necessità di eliminare gli EZ430 per motivi legati alla memoria disponibile, gli IRIS risultano ad oggi la piattaforma più conveniente sia in termini di consumi che di range di copertura. Va però osservato che gli IRIS non hanno sensori integrati (occorre aggiungere una sensor board) e non hanno un connettore USB o Ethernet (per la programmazione e per l'interoperabilità con PC e Tablet occorre quindi una terza scheda) inoltre hanno una memoria RAM di soli 8KB che per alcune applicazioni potrebbe essere insufficiente.

Per quanto riguarda le risorse computazionali (Tabella 3) e in particolare per la memoria RAM la piattaforma OpenMote, con i suoi 32KB, potrebbe risultare la scelta ideale ma come l'IRIS occorrono altre schede per sensori e programmatore (inoltre nell'OpenMote anche l'alloggiamento per l'alimentazione è in una scheda separata). L'unica alternativa agli IRIS rimane quindi la piattaforma TelosB che oltre ad integrare in un'unica scheda mote, sensor board e programmatore, sono prodotti dallo stesso produttore degli IRIS (la MEMSIC) e sono entrambi supportati dai principali sistemi operativi per WSN (TinyOS e Contiky) il che garantisce l'interoperabilità hardware e semplifica la realizzazione del software.

A fronte delle considerazioni suddette si è deciso quindi di optare per l'utilizzo di due tipi di mote per il progetto, precisamente IRIS e TelosB, sfruttando i primi per ottimizzare consumi e range di copertura e i secondi per applicazioni che richiedono una maggiore flessibilità e/o memoria.

5 Protocolli di comunicazione

Per descrivere l'architettura di una WSN, si può ricorrere al modello OSI (Open Systems Interconnection), un modello definito dall'ISO nello standard 7498 e che divide le funzionalità di rete in sette livelli (detti anche strati o layer) secondo lo schema riportato in Figura 21.

Ad ogni livello vi è in genere un protocollo che ha lo scopo di fornire funzionalità ai livelli superiori nascondendone i dettagli implementativi. L'insieme dei protocolli viene detto stack o pila protocollare.

Modello OSI			
Strato		**Unità Dati**	**Funzionalità**
7.	Applicazione	Dati	Applicazioni (API) e Servizi
6.	Presentazione		Rappresentazione dei dati e crittografia
5.	Sessione		Gestione delle sessioni
4.	Trasporto	Segmenti / Datagrammi	Trasmissioni affidabili end-to-end, segmentazione e multiplazione
3.	Rete	Pacchetti	Indirizzamento e instradamento
2.	Collegamento Dati	Trame	Controllo di flusso e degli errori; accesso al mezzo (MAC)
1.	Fisico	Bit	Trasmissione del segnale attraverso il mezzo fisico (modulazioni, interfacce elettriche, ecc.)

Figura 21 Modello OSI.

I livelli che costituiscono il cuore di una WSN sono i primi tre (fisico, collegamento dati e rete). Va detto per completezza che la separazione di questi layer in una WSN non è però così netta, in particolare il livello 1 e il livello 2 sono spesso unificati per creare dei protocolli di comunicazione efficienti, leggeri e capaci di interfacciarsi con l'hardware per supportare funzionalità avanzate finalizzate ad un maggiore risparmio energetico.

I layer superiori al quarto assumono invece un'importanza minore fatta eccezione per il livello applicazione che fornisce l'interfaccia verso l'utente.

5.1 Protocolli di comunicazione per WSN

Di seguito verranno descritti i principali protocolli di comunicazione di interesse specifico per le WSN (IEEE 802.15.4, ZigBee, 6LowPAN, WirelessHART, ISA 100, WiMi, SimpliciTI, Z-Wave, EnOcean, KNX-RF, THREAD e Bluetooth Low Energy).

IEEE 802.15.4 [78]**,** per quanto concerne i primi due livelli (livelli fisico e collegamento dati) gran parte delle WSN si basano sullo standard IEEE 802.15.4, appositamente sviluppato per comunicazioni wireless a bassa potenza.

Lo standard **IEEE 802.15.4** regola il livello fisico ed il livello MAC (Media Access Control) delle reti Low-Rate WPAN (LR-WPAN) ovvero reti wireless orientate a tutti quelli scenari applicativi che richiedano bassi consumi ed un basso data rate, come ad esempio l'automazione industriale, la domotica e il monitoraggio ambientale.

Al livello fisico lo standard (nella versione 2011) prevede ben 15 modalità che definiscono le bande di frequenza e le tecniche di modulazione utilizzate.

Come già chiarito nel Paragrafo 4.5 la banda di interesse per l'applicazione in oggetto è la banda ISM a 2,4GHz (da 2400 a 2483.5MHz) poiché disponibile in tutto il mondo e perché permette l'utilizzo di tecniche di spread spectrum (SS) utili per ridurre le interferenze.

Per la trasmissione su tale banda lo standard IEEE 802.15.4 prevede una modulazione di tipo Offset-Quadrature Phase Shift Keying (O-QPSK) e una velocità di trasmissione pari a 250kbps. Due sono le tecniche di spread spectrum previste, DSSS (Direct-Sequence SS) e CSS (Chirp SS), di queste la prima è decisamente la più utilizzata nei transceiver commerciali.

La sensitività minima del ricevitore è di -85dBm che consente una distanza massima fra trasmettitore e ricevitore di 200m in spazio libero.

A livello MAC il protocollo di accesso al mezzo prevede due modalità di funzionamento, slotted e unslotted, entrambe basate su un meccanismo noto come Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). La Figura 22 illustra il diagramma a blocchi del meccanismo CSMA-CA sia nel caso slottato che nel caso non slottato.

In generale il meccanismo CSMA/CA prevede che, prima di trasmettere, un nodo attenda per un tempo casuale (detto periodo di backoff) a seguito del quale viene testato il canale. Se il canale non risulta occupato il nodo trasmette immediatamente altrimenti deferisce la trasmissione aumentando il periodo di back-off (con un algoritmo noto come back-off esponenziale). Superato un certo numero di tentativi di trasmissione il pacchetto viene scartato ed è quindi considerato perso (ovviamente sono possibili ritrasmissioni da parte dei protocolli di livello superiore).

Nella versione slottata di CSMA-CA, i limiti del periodo di backoff di ogni dispositivo sono allineati: l'inizio del primo periodo di backoff di ogni dispositivo è allineato con l'inizio della trasmissione del beacon da parte di un nodo coordinatore. Il sottostrato MAC deve assicurare che lo strato fisico cominci tutte le sue trasmissioni ai confini di un periodo di backoff.

Nella versione non slottata di CSMA-CA, i periodi di backoff di un dispositivo non sono relazionati ai tempi dei periodi di backoff degli altri dispositivi della PAN.

In entrambi i casi ogni dispositivo deve mantenere tre variabili per ogni tentativo di trasmissione di un frame: *NB, CW,* e *BE*.

- *NB* è il numero di volte che all'algoritmo CSMA-CA è stato richiesto di effettuare il back off mentre tentava la trasmissione; questo valore è inizializzato a 0 prima di tentare una nuova trasmissione.
- *CW* è la lunghezza della finestra di contesa che definisce il numero di periodi di backoff per i quali è necessario trovare il canale libero da attività prima che la trasmissione possa cominciare. La variabile *CW* è utilizzata solo nel caso della versione slottata.
- *BE* è l'esponente di backoff, che è legato a quanti periodi di backoff un dispositivo deve attendere prima di tentare di accedere al canale.

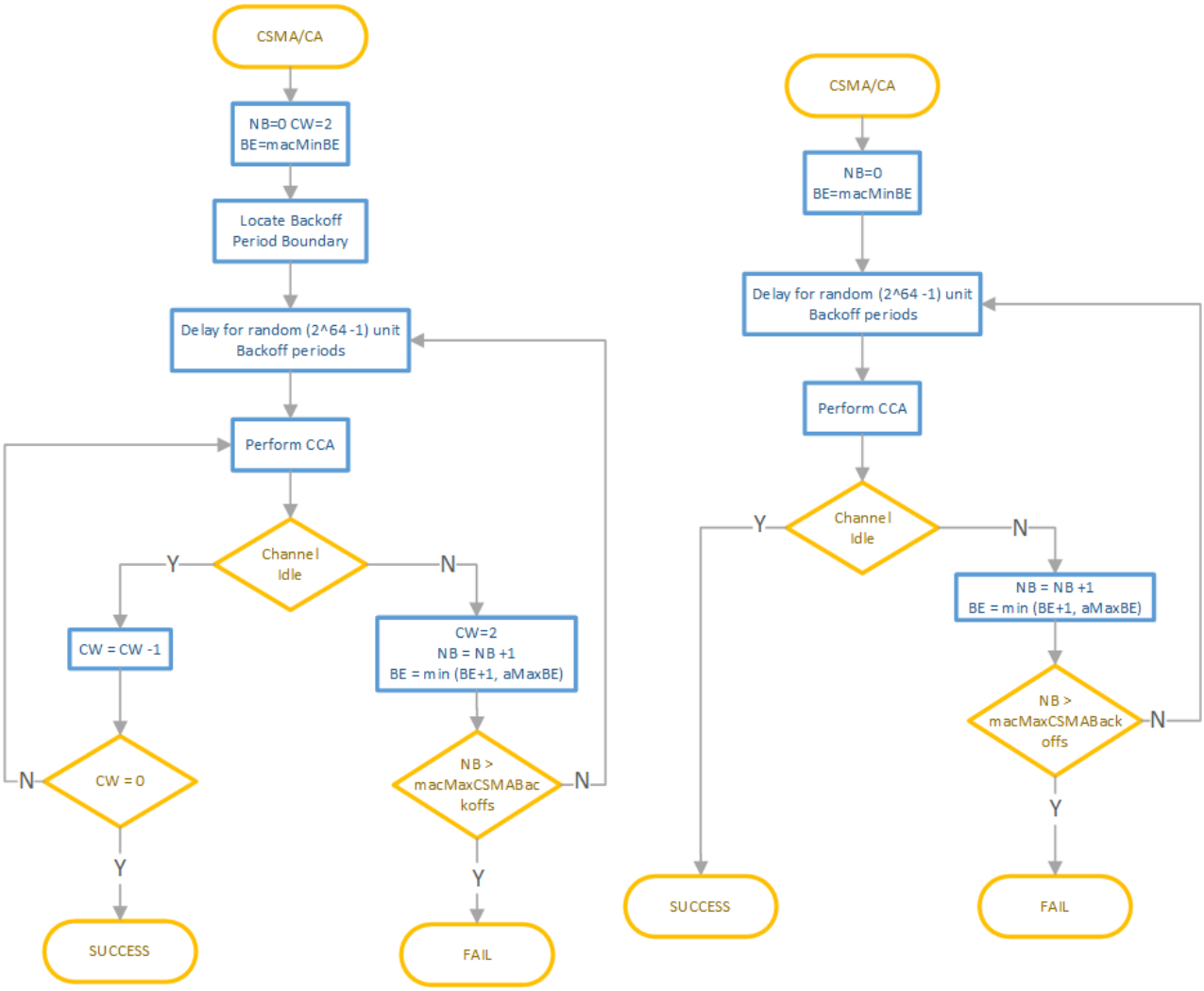


Figura 22 Algoritmo di back-off slotted (a sinistra) e unslotted (a destra).

Per ridurre i consumi energetici si utilizzano tecniche di duty-cycling, approfondite nel Paragrafo 8.1, ovvero si fa in modo che il trasmettitore e il ricevitore si attivino periodicamente per un breve periodo di tempo. In particolare lo standard permette di operare con un duty-cycle (rapporto fra il tempo in cui il dispositivo è attivo e il tempo di ciclo che intercorre fra due attivazioni successive) minimo del 1%.

La struttura della trama IEEE 802.15.4 è illustrata in Figura 23.

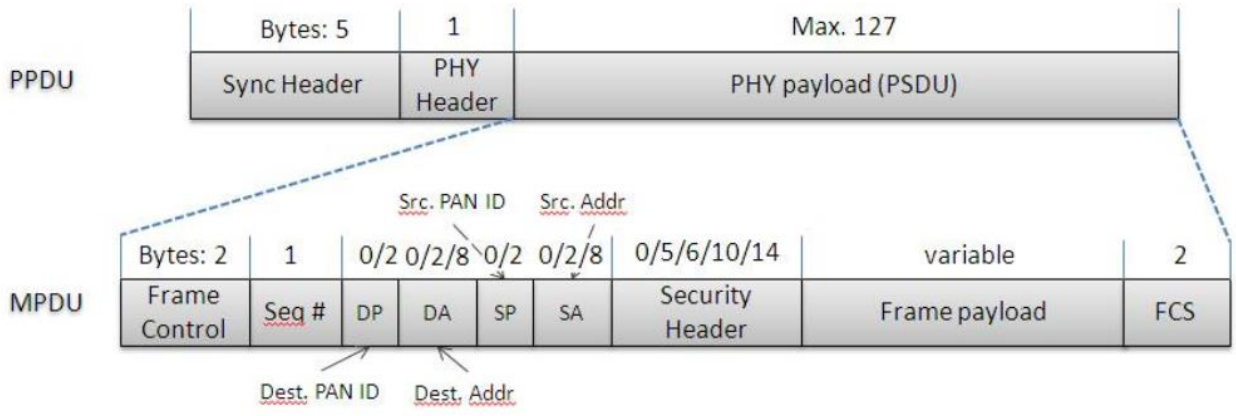


Figura 23 Formato trama IEEE802.15.4.

La lunghezza massima del payload a livello fisico (PSDU) è di 127 byte.

Al livello MAC sono previsti diversi campi: 2 byte per il campo Frame Control (il tipo di frame), 1 byte che specifica un numero di sequenza (necessario per identificare le ritrasmissioni), da 0 a 20 byte per l'indirizzamento, un campo da 0 a 14 byte con opzioni per la sicurezza, e un Frame Check Sequence (FCS) da 2 byte che permette di controllare l'integrità della trama. Lo standard prevede due diverse modalità di indirizzamento a 16 bit (short) o a 64 bit (long). Nel caso di indirizzamento di tipo short (2 byte per l'indirizzo di destinazione e 2 byte per l'indirizzo sorgente) e supponendo di non dover utilizzare opzioni per la sicurezza, la dimensione massima del payload a livello MAC risulta essere pari a 127-9=118 byte.

Lo standard IEEE802.15.4, a seguito della sua prima pubblicazione del 2003, ha subito diverse revisioni, la più recente è quella del 2011 a seguito della quale sono stati approvati diversi emendamenti con lo scopo di aggiungere nuove funzionalità e/o migliorarne il funzionamento. Fra queste si segnalano

L'emendamento **IEEE 802.15.4e** [77] il quale introduce un meccanismo inteso per applicazioni che possono negoziare la latenza a favore dell'efficienza energetica. Permette a un nodo di operare con un bassissimo duty cycle (ad esempio 1% o inferiore) mentre continua ad apparire sempre attivo ai livelli superiori. Inoltre definisce cinque nuovi modi operativi del MAC fra cui il Time Slotted Channel Hopping (TSCH) progettato per l'automazione industriale e il controllo di processo, fornisce supporto per comunicazioni multi-hop e multi-canale attraverso una multiplazione TDMA.

L'emendamento **IEEE 802.15.4g** definisce nuove modalità di funzionamento dello strato fisico e modifica il livello MAC con l'obiettivo di implementare reti wireless per smart metering utility (SUN). In particolare prevede l'utilizzo della banda a 169MHz e supporta diversi data rate:

- Multi-rate e multi-regional frequency shift keying (MR-FSK);
- Multi-rate e multi-regional orthogonal frequency division multiplexing (MR-OFDM);
- Multi-rate e multi-regional offset quadrature phase-shift keying (MR-O-QPSK).

Sullo standard IEEE802.15.4 sono basati diversi protocolli di livello superiore che differiscono fra loro per costi, prestazioni e scenari applicativi e che completano lo stack. Il più noto fra questi è ZigBee.

ZigBee [79] è il nome di un insieme di protocolli di comunicazione definiti dalla "ZigBee Alliance", una associazione di oltre 300 membri tra cui Silicon Labs, Philips, Texas Instruments, Motorola e Analog Device.

A livello fisico e MAC Zigbee si appoggia allo standard IEEE802.15.4. La caratteristica che rende ZigBee efficiente dal punto di vista energetico è l'implementazione di tecniche di duty-cycling (anche dello 0,15%) che permettono ad un nodo di restare in modalità sleep per la maggior parte del tempo. Il numero massimo di nodi in una rete è di 2^{16} con un range di copertura massimo di un singolo nodo di 200m. Altra caratteristica di interesse dello standard ZigBee è che può implementare l'algoritmo di cifratura AES a 128 bit, per rendere più sicura la trasmissione.

Il livello di rete specifica i meccanismi affinché i nodi possano associarsi alla rete, si occupa della sicurezza dei pacchetti, dell'instradamento e della individuazione dei nodi vicini. In particolare supporta il protocollo di routing AODV. A questo livello è possibile distinguere fra tre tipologie di reti ZigBee: ZigBee RF4CE (la più semplice, pensata per comunicazioni dirette fra due nodi), ZigBee PRO (la più comune, permette la gestione di reti mesh con 2^{16} nodi; esiste anche una estensione nota come Green Power che supporta dispositivi alimentati con tecniche di harvesting), ZigBee IP (pensata per permettere l'interoperabilità di reti ZigBee con reti IPv6).

Il livello trasporto non è definito da ZigBee.

Il livello applicativo è costituito da diversi blocchi: il sotto-strato "support" (APS), che si occupa di mantenere le tabelle di connessione dei dispositivi e di inoltrare i messaggi fra i dispositivi connessi; il "device objects" (ZDO), che definisce i ruoli dei dispositivi all'interno della rete (come ad esempio quello di coordinatore o di dispositivo terminale), avvia e/o accetta le richieste di connessione, stabilisce connessioni

sicure fra i dispositivi di rete, individua i dispositivi nella rete e determina quali servizi applicativi forniscono; ed infine l'"application framework" (AF) che permette l'esecuzione dei diversi "profili" [80]. I "profili" sono alla base del funzionamento di Zigbee, pensati appositamente per garantire l'interoperabilità fra prodotti e applicazioni sviluppati da produttori diversi. I profili possono essere pubblici (definiti dalla stessa Zigbee Alliance) o privati (vendor-specific). In entrambi i casi sono definiti da codici a 16 bit (Profile ID).

Fra i profili che possono essere di interesse per il progetto vi sono l'Industrial Plant Monitoring (Profile ID 0x0101), Building Automation (0x0105) e Smart Energy (0x0109).

L'uso di ZigBee per scopi commerciali richiede l'adesione alla ZigBee Alliance (un fee da pagare annualmente), questo contrasta in parte con gli scopi del progetto rivolto all'utilizzo di protocolli open source.

6LowPAN è l'acronimo di "IPv6 over Low-power Wireless Personal Area Networks", un protocollo finalizzato alla trasmissione di pacchetti IPv6 su reti IEEE 802.15.4 le cui specifiche sono definite nel documento RFC4919 [81] emanato dell'IETF (Internet Engineering Task Force). 6LowPAN prevede l'assegnazione di un indirizzo IPv6 univoco globale a qualunque dispositivo permettendo quindi la connessione dei nodi direttamente ad Internet, senza la necessità di introdurre gateway o proxy. IPv6 utilizza un indirizzamento a 128 bit e quindi possono essere indirizzati univocamente fino a $3.4 \cdot 10^{38}$ dispositivi. Più specificatamente lo standard definisce le regole per lo scambio di informazioni fra il protocollo IPv6 (al livello rete) e lo strato MAC dell'IEEE 802.15.4 definendo uno strato di adattamento che permette di comprimere l'header IPv6 di 40 byte in un header più compatto di soli 3 byte (a seconda del metodo di compressione scelto anche l'header dei protocolli superiori UDP/TCP/ICMP può essere compresso, richiedendo complessivamente 7 byte) e di frammentare i pacchetti IPv6 (la cui lunghezza può essere di 1280 byte) in frame di lunghezza massima di 127 byte, conformi quindi al livello MAC IEEE 802.15.4. Il protocollo si occupa, inoltre, della gestione e dell'assegnazione automatica degli indirizzi ai dispositivi (una delle caratteristiche peculiari di 6LowPAN è quella di poter assegnare dinamicamente ai dispositivi indirizzi "short" a 16 bit e di implementare un routing gerarchico in reti mesh); in particolare è possibile l'auto riconfigurazione sfruttando la versione 3 del protocollo Simple Network Management Protocol (SNMPv3) [82].

Si noti che senza il 6LowPAN la massima dimensione del payload per le applicazioni nel caso di utilizzo di IPv6 e UDP risulterebbe essere 127-9-40-8 = 70 byte (o anche 54byte se fossero necessarie le opzioni per la sicurezza); mentre con 6LowPAN tale valore risulta essere 127-9-7=111 byte (quasi il 60% in più).

Uno dei vantaggi di 6LowPAN nei confronti di ZigBee consiste nella intrinseca interoperabilità con IPv6. I dispositivi 6LowPAN sono infatti in grado di comunicare direttamente con altri dispositivi IPv6 mentre i dispositivi ZigBee necessitano di un gateway 802.15.4/IP per interagire con una rete IP. La decisione sulla scelta dello standard dipende dall'obiettivo dell'applicazione. Per applicazioni nelle quali non è necessario interfacciare direttamente i nodi sensori con reti IP o per le quali la dimensione delle unità informative è piccola le prestazioni di ZigBee possono essere migliori [83]; in termini però di scalabilità ed efficienza energetica è però il 6LowPAN ad essere la scelta migliore [84]. Inoltre 6LowPAN non richiede costi di licenza e diversi sono i sistemi operativi per reti di sensori che forniscono già una implementazione del 6LowPAN [81]. In particolare è pienamente supportato dalle piattaforme iris e TelosB scelte per il progetto.

WirelessHART [85] è un protocollo pensato appositamente per applicazioni di automazione dei processi industriali che deriva, aggiungendogli connettività wireless, dal più noto protocollo HART (Highway Addressable Remote Transducer) sviluppato alla fine degli anni '80 dalla "HART Foundation" [86] e utilizzato nei sistemi di controllo cablati.

In particolare i dispositivi basati su WirelessHART mantengono la compatibilità con i dispositivi HART esistenti, in termini di comandi e servizi. Il protocollo è basato sullo standard IEEE 802.15.4-2006 ed utilizza quindi la banda a 2,4GHz. Rispetto a ZigBee e 6LowPan la principale peculiarità del WirelessHART è l'utilizzo di un diverso protocollo di livello data-link, il Time Synchronized Mesh Protocol (TSMP), originariamente sviluppato da Dust Networks. Il TSMP si basa sulla tecnica TDMA per gestire l'accesso dei nodi al mezzo trasmissivo ed evitare le collissioni. In pratica il tempo viene diviso in slot di 10ms assegnati ai diversi

dispositivi. Ciò rende più efficienti le comunicazioni (riducendo e rendendo quasi deterministici i tempi di trasmissione) ma a scapito della scalabilità. Il WirelessHART prevede anche dei meccanismi di channel hopping e channel blacklisting a livello rete che permettono di rendere la trasmissione maggiormente immune ai disturbi (interferenze e rumore). Più precisamente il channel hopping si occupa di spostare la trasmissione dati a differenti frequenze in periodi di tempo differenti; il channel blacklisting si occupa di escludere, o mettere in una "lista nera", i canali che presentano maggiore interferenza con il segnale. Il WirelessHART supporta inoltre un routing con ridondanza per aumentare l'affidabilità della rete.

La sicurezza è implementata sia a livello data-link che a livello rete. A livello data-link, o MAC, è previsto un algoritmo di crittografia AES a 128 bit e a livello rete è assicurata l'integrità dei dati end-to-end. L'applicazione che si occupa della gestione della sicurezza (Security Manager) è implementata nel gateway, che gestisce tutti i servizi per la sicurezza. In particolare sono previste chiavi distinte sia per le comunicazioni dispositivo-dispositivo che per le comunicazioni dispositivo-gateway.

Nonostante il WirelessHART sia considerato uno standard robusto, affidabile ed efficiente in termini di consumi energetici, esso ha diversi limiti [87], in particolare quello di non essere compatibile con reti IP. Si tratta inoltre di uno standard proprietario (anche se adottato da diverse aziende come ABB ed Emerson).

ISA-100.11a [88], definito dall'ISA (International Society of Automation) nel 2009, è un altro degli standard per le reti wireless basato sull'IEEE 802.15.4-2006.

Come il WirelessHART utilizza tecniche TDMA e di channel hopping per ridurre le interferenze. A differenza del WirelessHART il timeslot non è fissato e sono quindi possibili differenti tecniche di channel hopping (slow, fast e mixed). In paricolare riducendo il timeslot è possibile ottenere comunicazioni con bassa latenza, tipicamente pari a 100ms [87].

A livello rete, la compatibilità con il protocollo IPv6 e 6LowPAN permette una maggiore interoperabilità con altre reti rispetto a WirelessHART. Inoltre, ISA-100 prevede interfacce per la coesistenza con WirelessHART.

Il protocollo ISA-100 è simile a WirelessHART anche per la gestione della sicurezza. Esiste, infatti, un gestore della sicurezza (Security Manager) e un gateway (System Manager) che cooperano per la generazione, memorizzazione e distribuzione delle chiavi di crittografia e si occupano, inoltre, della gestione delle procedure di autenticazione. L'algoritmo di crittografia utilizzato è l'AES a 128bit. In tale protocollo, differentemente da WirelessHART, le funzionalità per la sicurezza sono però opzionali: se da un lato tale scelta fornisce maggiore flessibilità (le opzioni per la sicurezza possono ad esempio essere eliminate a vantaggio del tempo di vita dei nodi) dall'altro permette la commercializzazione di prodotti ISA100 che non dispongono di tali opzioni.

MiWi [89] è uno stack protocollare *proprietario* della "Microchip" leggero e ottimizzato per l'invio di brevi messaggi. Nella sua versione base supporta topologie di rete a stella, ad albero e a maglia. Esiste poi una variante del protocollo, denominata **MiWi P2P**, che supporta sia topologie peer-to-peer che topologie a stella. Il numero massimo di dispositivi previsto è di 1024 con range di copertura di 125m indoor e 550m outdoor. Può utilizzare crittografia a 32, 64, 128 bit con algoritmo AES. Un'altra variante del protocollo è **MiWi Pro** che supporta topologie di reti mesh con 8000 nodi ed un massimo di 64 hops. Il limite principale di tale protocollo è che può essere utilizzato solo su dispositivi microchip.

WISA [90] è l'abbreviazione di Wireless Interface for Sensor and Actuators ed è un protocollo *proprietario* sviluppato da ABB nel 2003. È basato sul protocollo IEEE 802.15.1 (Bluetooth) e trasmette alla frequenza di 2,4GHz con una velocità di trasmissione di 1Mbps usando tecniche di frequency hopping. Ha due sistemi che possono essere accoppiati: Wisa_COM, che si occupa solo della comunicazione wireless e WISA_POWER, che si occupa della comunicazione e alimentazione attraverso una interfaccia RF simile a quella degli RFID.

ANT [91] è un protocollo wireless ed al tempo stesso una tecnologia dedicata ad applicazioni che riguardano lo sport e la salute, ma anche l'automazione industriale. Questo significa che, nonostante sia un protocollo *proprietario*, è sotto certi aspetti mantenuto "aperto". Trasmette a 2,4GHz e il massimo numero di dispositivi è limitato a 65533.

Il protocollo ANT implementa gli strati protocollari fisico, rete e trasporto. Inoltre, include funzionalità di crittografia con chiavi di sicurezza a basso livello. La configurazione tipica consiste in un host/MCU che comunica con uno o più nodi che implementano il protocollo ANT. È l'host che si occupa di stabilire la connessione con i moduli ANT e mantenerla, attraverso semplici messaggi bidirezionali definiti dal protocollo.

ANT è un protocollo che gestisce topologie di rete peer-to-peer, a stella, ad albero e mesh.

Il protocollo supporta la possibilità di collegare numerose reti, pubbliche o private. Una particolare rete può specificare un set di regole operative per tutti i nodi in essa inclusi. Due nodi ANT, per poter comunicare, devono appartenere alla stessa rete. Ciò dà la possibilità di definire reti che possono essere pubbliche, appositamente condivise tra differenti compagnie, dando la possibilità di avere un sistema aperto di dispositivi che comunicano tra di loro. Una rete "gestita" definisce regole e comportamenti specifici che ne regolano l'uso. Un esempio di una rete *gestita* è la rete ANT+. Le aziende che hanno accettato la promessa di interoperabilità ANT+ fanno parte della ANT+ Alliance, un gruppo di interesse speciale che promuove il valore del marchio ottimizzata e collaborazioni con altri prodotti di livello superiore. Il vantaggio principale di questa rete è l'interoperabilità tra i dispositivi che permette la comunicazione senza fili con altri prodotti ANT+. Le applicazioni target includono, appunto, qualsiasi monitoraggio di sensori wireless per attività sportive o per il benessere.

SimpliciTI [92] è un protocollo *proprietario* della Texas Instruments sviluppato appositamente per la realizzazione di semplici e piccole reti wireless sfruttando i SoC RF della TI CC25XX e CC430 o, in alternativa, un transceiver della serie CC1XXX con un microcontrollore MSP430. Trasmette alle frequenze: 433MHz, 868MHz e 915MHz. L'unica piattaforma WSN, tra quelle illustrate nel Capitolo 4, che è compatibile con questo protocollo è la Z1 di Zolertia.

Z-Wave [93] è un protocollo *proprietario*, inizialmente sviluppato da Zen-Sys, adesso distribuito da Sigma Designs e promosso dalla Z-Wave Alliance. È un protocollo progettato appositamente per applicazioni di controllo remoto e domotica. Trasmette alla frequenza di 868MHz con un range di 30m e bit-rate massima di 40kbps. Può collegare al massimo 232 nodi all'interno della stessa rete. Z-Wave, a differenza di altri protocolli (es. ZigBee e 6LowPAN) non è un protocollo di livello 3 che si basa sul protocollo 802.15.4, ma un protocollo monolitico in cui, oltre ai livelli MAC e trasporto, sono implementati i livelli routing e applicazione.

EnOcean [94] è un protocollo *proprietario* sviluppato in Germania dalla compagnia omonima EnOcean. Trasmette nella banda ISM a 868MHz in Europa e alla frequenza di 315MHz in Nord America. Il range di trasmissione è di 30 metri indoor e 300m outdoor. La caratteristica che lo distingue dai protocolli precedentemente analizzati è che i moduli wireless basati su questo protocollo non hanno bisogno di alimentazione a batterie ma possono recuperare energia dall'ambiente mediante tecniche di energy harvesting, ovvero sfruttando moduli fotovoltaici o l'effetto piezoelettrico. Non sono previste in tale protocollo tecniche di crittografia.

KNX [95] è definito dallo standard internazionale ISO/IEC14543-3, oltre che dagli standard europei ENELEC EN50090, CEN EN 13321-1 e 13321-2, e dagli standard cinesi GB/Z 20965 e americani ANSI/ASHRAE 135. Trasmette alla frequenza di 868MHz con una modulazione FSK e raggiunge velocità di trasmissione di circa 16kbps. Lo standard KNX è *proprietario*.

Bluetooth Low Energy (BLE) [96]: Un'alternativa recente alla combinazione IEEE802.15.4-ZigBee è data dal Bluetooth Low Energy (LE) e in particolare la versione 4.2 [97].

Uno dei vantaggi di questo standard è quello di essere già presente in dispositivi palmari (smartphone e tablet di ultima generazione). L'utilizzo del BLE al posto dello standard IEEE802.15.4 permetterebbe quindi di utilizzare un comune dispositivo palmare come sink, oltre che consentire l'accesso ai dati della WSN da parte di un qualsiasi dispositivo con connessione BLE. Occorre però tenere presente che esistono diverse versioni di BLE (4.0, 4.1 e 4.2) notevolmente differenti fra loro.

Va osservato infatti che fino alla versione 4.0 il Bluetooth aveva come obiettivo primario la connessione attraverso un collegamento radio a basso consumo tra una coppia di dispositivi (master e slave), come ad esempio un televisore e il suo telecomando, uno smart-watch e uno smartphone o delle cuffie e un riproduttore musicale, o al più la creazione di piccole reti (dette piconet, composte da al più otto dispositivi, un master e sette slave).

La versione 4.1 ha rappresentato un primo significativo cambiamento a questa architettura introducendo il concetto di topologia Scatternet in cui un nodo master può essere contemporaneamente slave di un altro nodo master. Questo permette la realizzazione di topologie multi-hop ma asimmetriche (basate intrinsecamente sulle piconet, e che non permettono ai nodi di comunicare ininterrottamente senza stabilire una connessione master-slave).

Solo il Bluetooth 4.2, le cui specifiche sono state rilasciate nel dicembre 2014, può essere promosso come un protocollo per IoT, in quanto, oltre ad avere consumi paragonabili con i dispositivi 802.15.4 (la corrente di picco è di soli 15mA) consente la connessione di dispositivi con indirizzamento IPv6 (anche se per la realizzazione di reti mesh utilizza lo stesso meccanismo delle scatternet della versione 4.1 o soluzioni proprietarie). Grazie all'indirizzamento IPv6 il sink della WSN può essere un qualunque computer, uno smartphone, o addirittura il cloud senza che sia necessario del software aggiuntivo o un gateway (ma semplicemente mediante un comune web browser). Un altro parametro che rende preferibile la versione 4.2 alle versioni precedenti è la dimensione massima del pacchetto (o meglio del payload). In generale un payload più grande permette un throughput maggiore, necessario per supportare funzionalità come l'aggiornamento del firmware da remoto, oltre che l'utilizzo di protocolli per la sicurezza, ma soprattutto per un più veloce trasferimento dei dati provenienti dai sensori [98].

Nelle versioni del BLE 4.0 e 4.1 la dimensione massima del payload è di soli 27 byte, decisamente inferiore a quella prevista dallo standard 802.15.4 (pari a 127 byte), mentre nella versione 4.2 del BLE la dimensione del payload è di 251 byte.

Recentemente (giugno 2016) è stato annunciato il rilascio per la fine del 2016 o inizio del 2017 anche della versione 5 di Bluetooth LE con la promessa di quadruplicare il range di copertura, raddoppiare la velocità di trasmissione, incrementare la capacità di trasmissione in broadcast dell'800% e anche di garantire il supporto di topologie mesh. Al momento però non possono essere presi in considerazione tali benefici visto che non risultano trasnceiver commerciali per il BLE 5.

A fronte di tali considerazioni si ritiene che solo la versione 4.2 possa essere ritenuta idonea agli scopi del progetto. D'altra parte però ad oggi solo pochi smartphone di fascia alta integrano la versione 4.2 del BLE (es. iphone 6/6s e Samsung S7) e anche per i mote presenti in commercio è in genere presente al più la versione 4.1. Ciò può essere ricondotto al fatto che i Controller BLE di versioni differenti (es. 4.x e 3.0) sono incompatibili e sebbene esistano chip dual-mode (es. CC2650) questi hanno un costo superiore. Attualmente per la realizzazione di WSN lo standard 802.15.4 è quindi da considerare una tecnologia più matura e consolidata.

Di seguito sono riportati ulteriori dettagli sull'architettura protocollare del BLE utili per un confronto più puntuale fra i due standard.

L'architettura di un nodo BLE prevede una parte Controller ed una parte Host. Nel Controller sono implementati il livello fisico e il livello collegamento dati [99].

Il Controller è solitamente implementato in un SoC con radio integrata. L'Host è invece implementato in un processore convenzionale e comprende i livelli superiori, come ad esempio il Link Control, l'Adaptation Protocol (L2CAP), l'Attribute Protocol (ATT), il Generic Attribute Profile (GATT), il Security Manager Protocol (SMP) e il Generic Access Profile (GAP), di seguito discussi. La comunicazione tra Controller e Host è definita dall'interfaccia standard Host Controller Interface (HCI).

Dal punto di vista dello strato fisico i dispositivi Bluetooth LE operano nella banda non licenziata ISM a 2.4 Ghz. Più precisamente si utilizzano 40 canali RF le cui frequenze centrali sono date dalla relazione (2402 + k · 2) MHz dove k = 0, ..., 39. Rispetto all'802.15.4 si hanno quindi più canali e con una banda maggiore. Se da

un lato la maggiore banda permette di avere una maggiore la velocità di trasmissione dall'altro comporta in genere un rapporto segnale-rumore minore e un maggior livello di interferenza. È però previsto un meccanismo di frequency hopping ("salto di frequenza") per alleviare le interferenze e il fading.

Il livello di potenza di trasmissione del segnale può variare da un minimo di -20dBm ad un massimo di 10dBm. L'addendum 5 del 15 dicembre 2015 [97] ha incrementato la potenza massima di trasmissione portandola a 20dBm prevedendo 4 classi di potenza dei dispositivi: 1 (potenza minima 10dBm, potenza massima 20dBm), 1.5 (potenza minima -20dBm, potenza massima 10dBm), 2 (potenza minima -20dBm, potenza massima 4dBm), 3 (potenza minima -20dBm, potenza massima 0dBm). Il range di copertura di dispositivi BLE in classe 1 e 1.5 è di 30m paragonabile quindi a quella di alcuni dispositivi 802.15.4 mentre per le altre classi è decisamente inferiore (sotto i 10m).

La modulazione è di tipo Gaussian Frequency Shift Keying (GFSK) con un prodotto larghezza di banda per periodo di bit BT=0.5. I dati trasmessi hanno una symbol rate di 1 milione di simboli al secondo a cui corrisponde un throughput massimo di circa 300kbps per la version 4.1 e di 780bps per la versione 4.2, in entrambi i casi superiore al massimo throughput previsto dall'802.15.4 (140kbps).

Le operazioni del Link Layer possono essere descritte attraverso una macchina a stati finiti a 5 stati: "Standby State" (il dispositivo non trasmette e non può ricever alcun frame); "Advertising State" (il dispositivo trasmette frame nello "advertising channel" e possibilmente ascolta, riceve ed eventualmente risponde a questi frame); "Scanning State" (il dispositivo ascolta lo "advertising channel" per verificare la presenza di frame inviati da dispositivi che sono nello stato "advertising"); "Initiating State" (il dispositivo ascolta lo "advertising channel" per verificare la presenza di frame da uno specifico dispositivo e rispondere a questi frame per iniziare una connessione); "Connection State" (il dispositivo può assumere il ruolo di master o slave per comunicare con un altro dispositivo).

Ai dispositivi è assegnato un indirizzo di 48 bit che può essere univoco e pubblico o random (static o private).

I 40 canali disponibili dell'interfaccia aerea sono suddivisi in due categorie: "advertising channel" (3 canali) e "data channel" (37 canali). Gli "advertising channel" sono usati per scoprire i dispositivi, inizializzare una connessione e trasmettere dati in broadcasting. I "data channel" sono usati per le comunicazioni fra i dispositivi connessi. Ad ogni canale è associato un indirizzo univoco. Due dispositivi che intendono comunicare devono condividere lo stesso canale. Per fare questo i due transceiver dei dispositivi devono essere sintonizzati sulla stessa frequenza. Dato che il numero di canali è limitato e che molti dispositivi possono operare indipendentemente all'interno della stessa area ci sarà una elevata probabilità che due diverse coppie abbiano i transceiver sintonizzati alla stessa frequenza, provocando quindi delle collisioni nei frame trasmessi. Per cercare di mitigare questo problema viene utilizzato il meccanismo del frequency hopping il cui pattern è specificato nel campo "Access Address".

Ogni frame può avere una dimensione minima di 10byte e una dimensione massima di 265 byte a seconda della lunghezza del payload (minimo 2 byte, massimo 257 byte). Oltre al campo "Access Address" di 4 byte saranno presenti un preambolo di 1byte e un campo CRC di 3byte.

Il livello data link definisce due ruoli: master e slave. Per ottenere un minore consumo di potenza, i dispositivi slave sono in stato di sleep e si attivano periodicamente per controllare se il master ha inviato un pacchetto. Il master determina gli istanti in cui gli slave devono mettersi in ascolto e così coordina l'accesso al mezzo utilizzando la tecnica TDMA (Time Division Multiple Access). È il master che fornisce le informazioni necessarie per il frequency hopping e che ha il controllo della connessione.

Ai livelli superiori, l'Internet Protocol Support Profile (IPSP), rilasciato con la versione Bluetooth 4.2, fornisce funzionalità di discovery permettendo a due dispositivi Bluetooth abilitati al protocollo IPv6 (uno centrale e l'altro periferico) di scoprirsi l'un l'altro e stabilire una connessione a livello data-link.

I "Bluetooth LE Generic Attribute Profiles" (GATT) permettono di scoprire se IPSP è supportato mentre il "Bluetooth LE L2CAP Credit Based Flow Control Mode" permette lo scambio dei dati. Il GATT definisce il trasferimento dati bidirezionale tra host e Controller, o nodi sensori e Gateway per utilizzare la

terminologia più adatta alle WSN. Il GATT è utilizzato solo dopo che la connessione è già stata stabilita tra i dispositivi. Ogni dispositivo della rete BLE è individuato da un ID unico (UUID).

Al di sopra del GATT, l'"IP Support Service" (IPSS) è utilizzato per stabilire il ruolo del nodo. Il working draft IETF del 2015 di J. Nieminen, et al. Dal titolo "IPv6 over Bluetooth Low Energy" [100] discute la modalità standardizzate per trasmettere pacchetti IPv6 attraverso Bluetooth LE. Il draft espone le tecniche per utilizzare 6LoWPAN e il collegamento fra 6LoWPAN e Bluetooth LE: in particolare viene adottato il meccanismo di compressione dell'header 6LoWPAN (per la frammentazione è raccomandato l'uso del meccanismo di frammentazione del Bluetooth L2CAP e non del 6LowPAN). Per permettere l'uso di tablet o smartphone come gateway verso la WSN, questi devono quindi implementare il protocollo 6LoWPAN. Diverse implementazioni di 6LoWPAN e Bluetooth LE esistono già per i nodi sensori: ad esempio il sistema operativo Contiki supporta sia 6LoWPAN che Bluetooth LE. I servizi Bluetooth GATT HTTP Proxy Service (HPS) e RESTful AAPIs ulteriormente aiuteranno la connettività fra dispositivi Bluetooth LE e Internet. Nel dominio IoT, il Constrained Application Protocol (CoAP), essendo più leggero del protocollo HTTP, sarà da preferire a quest'ultimo.

Per quanto concerne gli aspetti legati alla sicurezza, la versione 4.2 del BLE definisce dei livelli di sicurezza paragonabili a quelli dello standard Bluetooth generico (questo non accadeva con le versioni precedenti 4.0 e 4.1).

Per la gestione delle chiavi viene utilizzato l'algoritmo asimmetrico "Elliptic Curve Cryptography" (ECC) con l'uso di curve ellittiche raccomandate dalla Federal Information Processing Standards (FIPS). L'algoritmo approvato AES-CCM (anch'esso approvato dal FIPS) è utilizzato per crittografare i messaggi. La versione Bluetooth 4.2 fornisce un sistema di comunicazione sicuro fra due dispositivi vicini. Lo strato collegamento dati può proteggere il collegamento radio contro intercettazioni passive e, in alcuni casi, contro attacchi Man-in-the-Middle (MITM) in funzione della disponibilità di display o tastiere collegati ai dispositivi.

Nell'ambito della IoT, sarebbe possibile anche implementare una sicurezza End-to-End utilizzando congiuntamente IPSS/IPSP e quindi abilitando protocolli computazionalmente leggeri di sicurezza di Internet come "Internet Protocol security/ Internet Key Exchange" (IPsec/IKE) o Datagram Transport Layer Security (DTLS).

Dal punto di vista della "privacy" già la versione 4.0 di Bluetooth LE prevede frequenti cambiamenti dell'indirizzo del dispositivo (chiamato "indirizzo privato") per limitare la capacità di tracciare il dispositivo stesso per lunghi periodi di tempo. Un dispositivo Bluetooth LE che vuole stabilire una connessione con un altro dispositivo deve essere in grado di risolvere gli indirizzi privati che sono generati utilizzando la "Resolving Identify Key" (IRK) condivisa durante la fase di connessione. Bluetooth 4.2 aggiunge a questo la risoluzione dell'indirizzo privato nel Controller. Inoltre è supportata una "white list" di indirizzi privati nel Controller: questo permette di generare indirizzi privati senza coinvolgere l'Host e accettare o rifiutare richieste consultando la "white list". In questo modo è possibile ridurre la frequenza con cui l'Host viene risvegliato e quindi ridurre sensibilmente il consumo di energia.

THREAD [101] è una architettura basata su protocolli aperti sviluppata dal Thread Group (Nest Labs/Google, Samsung, Intel, Microsoft, HP, Qualcomm, Silicon Labs, OSRAM, Atmel, Dell, Huawei, LG, Philips, STM, Texas Instruments, Whirlpool sono solo alcuni degli otre 240 partner afferenti) pensata per comunicazioni wireless M2M (Machine-to-Machine) affidabili, a basso consumo di potenza e a basso costo. È stata progettata in modo specifico per applicazioni di domotica (Connected Home) basate su connessioni di rete compatibili con il protocollo IP, ciò al fine di rendere i dispositivi più facilmente accessibili ed interfacciabili con altre reti già presenti. Una delle caratteristiche è la facilità di installazione, avvio e gestione della rete. Thread, infatti, include protocolli che permettono ai dispositivi di auto-configurarsi e risolvere i problemi di instradamento quando questi si presentano.

L'architettura è piuttosto recente (Giugno 2015) e attualmente la documentazione è accessibile solo ai partner (l'affiliazione richiede il pagamento di un canone annuale).

Alcune informazioni sono però già reperibili in rete e sono di seguito riportate.

In particolare in Figura 24 è mostrata la struttura dello stack protocollare implementato da Thread. Esso si basa su IEEE 802.15.4 per gli strati fisico e collegamento dati e sul 6LowPAN per lo strato di rete, oltre che sul protocollo UDP a livello trasporto.

L'utilizzo di l'IPv6 (6LowPAN) fa sì che i dispositivi possano comunicare tra di loro accedendo a servizi cloud e interagendo con utenti che utilizzano le applicazione Thread per dispositivi mobili.

Dal punto di vista della sicurezza, Thread prevede che nuovi dispositivi possano unirsi alla rete solo se autorizzati, inoltre la comunicazione tra i nodi è crittografata.

In particolare ci sono tre fasi che un dispositivo deve superare per poter essere aggiunto alla rete: *Discovery*, *Commissioning* e *Attaching*. La fase di discovery prevede che il dispositivo che vuole unirsi alla rete esegue la scansione di tutti canali, invii un messaggio di richiesta in ognuno di essi e attenda il messaggio di risposta. Il messaggio, detto beacon, contiene nel payload il Service Set Identifier (SSID), che permette al nuovo dispositivo di capire se la rete ha accettato un nuovo membro. Alla fine della fase di discovering, la rete Thread utilizza un messaggio MLE (Mesh Link Establishment) per stabilire la tabella di instradamento attraverso cui il nuovo dispositivo può fare la procedura di commissioning. La procedura di commissioning serve ad inviare al nuovo dispositivo le informazioni necessarie per collegarsi alla rete Thread. Successivamente il dispositivo che ha già le informazioni di commissioning, si mette in contatto con il Parent Router e può così collegarsi alla rete (attaching), scambiando messaggi di tipo BLE (Bluetooth Low Energy).

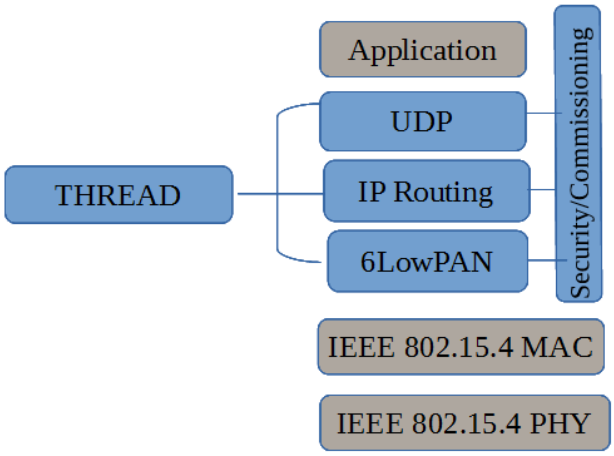


Figura 24 Strati protocollari Thread.

5.2 Confronto tra i protocolli di comunicazione

Appare chiaro dalla descrizione precedente come allo stato attuale sia disponibile un vasto numero di protocolli di comunicazione wireless con differenti caratteristiche e prestazioni. È dunque importante individuare quelli che maggiormente rispondono alle specifiche del progetto ed ove possibile confrontarne le prestazioni in termini dei principali indici importanti per applicazioni industriali e in particolare:

- consumi
- robustezza (immunità ai disturbi e tolleranza ai guasti)
- sicurezza
- costi

Un interessante confronto dei protocolli di comunicazione è stato realizzato da Gravogl et Alii in [102]; in tale confronto sono valutate le prestazioni dei diversi protocolli in termini di consumi dei dispositivi e di robustezza ai disturbi prodotti da altre reti wireless.

Per quanto concerne i consumi, valutati negli stati di sleep, trasmissione e ricezione, come è possibile osservare dalla Tabella 8, i protocolli basati sullo standard IEEE 802.15.4 (ZigBee, 6LowPAN, WirelessHART, ISA-100), hanno in genere consumi di potenza minori soprattutto in modalità sleep (la modalità più rilevante per applicazioni low-power basate su tecniche di duty-cycling).

Tabella 8 Consumi di potenza negli stati sleep, ricezione e trasmissione nei protocolli di comunicazione wireless .

Protocolli	**Consumo di Potenza**		
	Sleep	**Trasmissione**	**Ricezione**
IEEE 802.15.4	0,06 µW	36,9 mW	34,8 mW
ANT	1,8 µW	39 mW	33,9 mW
ONE-NET	0,3 µW	63 mW	57,9 mW
EnOcean (*)	0,6 µW	99 mW	72 mW
Z-Wave (*)	8,25 µW	75,9 mW	120 mW
Bluetooth 4.0	330 µW	215 mW	215 mW

(*) Per EnOcean e Z-Wave sono stati considerati valori di un generico microcontrollore a bassa potenza

A fronte di tali confronti si ritiene che ad oggi lo standard IEEE802.15.4 risulta essere il più indicato per il progetto ai fini della realizzazione dei livelli protocollari fisico e collegamento-dati. Si ritiene però che tale scelta possa e debba essere rivalutata in un recente futuro se il BLE v4.2 (o il BLE v5, annunciato a giugno 2016), attualmente presente solo in smartphone di fascia alta (es. Galaxy S6/S7), sarà integrato in smartphone, tablet e mote di largo consumo.

Per quanto riguarda la robustezza ai disturbi, fattore ugualmente importante nelle applicazioni industriali, risultano essere particolarmente vantaggiosi i protocolli che implementano tecniche di spread spectrum. In particolare possono essere considerati idonei per applicazioni industriali e per il progetto in esame tutti i protocolli basati su 802.15.4 (il quale prevede una spread spectrum di tipo DSSS oltre che una bit-rate di 250kbps che rappresenta un buon compromesso tra velocità di trasferimento dati e consumi di potenza).

Un confronto approfondito in tale direzione tra WirelessHART e ZigBee è riportato in [103]. Dai risultati di tale confronto si evince che WirelessHART risulta più affidabile in ambienti industriali e fornisce una maggiore immunità e robustezza ai disturbi. Ciò è legato al fatto che in ZigBee la gestione dei canali è statica, ovvero, non si applica nessuna tecnica di frequency hopping o di riallocazione delle frequenze e l'accesso al canale è statistico (basato su tecniche CSMA). D'altra parte, però, il WirelessHART non è diffuso come ZigBee e non è implementato in nessuna delle piattaforme WSN esaminate precedentemente. Considerazioni simili possono essere fatte per un confronto fra 6LowPAN e WirelessHART va però detto che 6LowPAN (al contrario di ZigBee e WirelessHART) è uno standard aperto e non richiede il pagamento di canoni annuali.

Un confronto fra ZigBee e 6LowPAN nelle reti smart grid è riportato in [104]. Dai risultati di tale confronto è possibile asserire che il 6LowPAN risulta più robusto rispetto a ZigBee, grazie alla compatibilità con IP di

6LowPAN che permette di utilizzare algoritmi di routing più evoluti e con una maggiore tolleranza ai guasti di rete (network breakdown).

Infine, sempre in termini di robustezza, WirelessHART risulta superiore all'ISA100.11a. Nel protocollo ISA-100.11a, infatti, solo il router si occupa di effettuare le operazioni di instradamento quindi se quest'ultimo perde la connessione tutti i nodi della rete non potranno più comunicare tra di loro. Al contrario, in WirelessHART (così come con 6LowPAN) ogni nodo può effettuare le operazioni di instradamento o routing.

Per quanto concerne la sicurezza, 6LowPAN, ZigBee, WirelessHART e ISA-100 implementano a livello collegamento dati lo stesso algoritmo di crittografia (AES a 128bit), dunque dal punti di vista del livello di sicurezza possono essere considerati equivalenti. Le modalità di gestione della sicurezza risultano però essere differenti. In WirelessHART e ISA-100 esiste un gestore della sicurezza che si occupa dell'assegnazione delle chiavi di accesso alla rete e dell'autenticazione dei nodi, funzionalità invece assente in ZigBee. La differenza sostanziale tra WirelessHART e ISA-100 è che in quest'ultimo protocollo la sicurezza è resa opzionale per rendere la rete più flessibile. Come già osservato se da un lato tale scelta fornisce maggiore flessibilità (le opzioni per la sicurezza possono ad esempio essere eliminate a vantaggio del tempo di vita dei nodi) dall'altro permette la commercializzazione di prodotti ISA-100 che non dispongono di tali opzioni. 6LowPAN non ha una gestione della sicurezza ma, compatibilmente con le risorse dei dispositivi, possono essere adottati i meccanismi di sicurezza delle reti IP (in particolare il TLS e la sua versione per nodi resource-constrained DTLS).

Protocollo	Robustezza	Sicurezza	Costi/Licenze
Zig-Bee	☹	☺	☺
WirelessHART	☺	☺	☹
ISA-100.11a	☺	☺	☹
6LowPAN+IPv6+ RPL+DTLS	☺	☺	☺

5.3 Protocolli di routing per WSN

Un *protocollo di routing* ha lo scopo di individuare i possibili percorsi tra due nodi e scegliere quello ottimale secondo una opportuna metrica.

I protocolli di routing possono dividersi in protocolli *proattivi* e protocolli *reattivi*.

Nei protocolli *proattivi* (o table-driven) un nodo memorizza i possibili percorsi e le relative metriche in apposite tabelle dette *tabelle di routing*. Tali tabelle sono aggiornate dinamicamente mediante lo scambio di appositi pacchetti.

Esempi di protocolli proattivi per reti di sensori sono

- *DSDV* (Destination-Sequenced Distance Vector) [105],
- *RPL (*Routing Protocol for Low-power and Lossy networks) definito dal gruppo di lavoro ROLL dalla IETF [106] e specificatamente ideato per reti mesh basate su IPv6/6LowPAN
- *CTP* (Collection Tree Protocol) [107] pensato per reti di sensori con topologia ad albero e specificamente ideato per collezionare i dati provenienti dai nodi sensori ed inviarli ad un sink.

Il principale svantaggio di tali algoritmi consiste nei tempi di reazione in caso di riconfigurazione di una rete a seguito dell'aggiunta o eliminazione di un nodo e della necessità di inviare periodicamente pacchetti per aggiornare le tabelle di routing (impattando così sui consumi energetici).

Nei protocolli *reattivi* (anche detti on-demand) i percorsi vengono determinati su richiesta quando necessario con un processo di discovery basato sull'invio di pacchetti di Route Request. Esempi di protocolli reattivi per WSN sono:

- Ad hoc On-Demand Distance Vector Routing (AODV) [108];
- Dynamic Source Routing (DSR) [109].

Il principale svantaggio degli algoritmi reattivi consiste nella latenza necessaria per determinare di volta in volta il percorso ottimale.

Diversi sono i lavori in letteratura che riportano utili confronti degli algoritmi suddetti, ad esempio [110] e [111].

Sulla base dell'analisi dei lavori suddetti e di altri lavori in letteratura si è scelto il protocollo RPL quale protocollo di routing per il progetto MiSE-ENEA. Tale scelta è motivata dal fatto che, come dimostrato nei lavori suddetti, a parità di condizioni RPL ha un minore overhead, rispetto ad altri protocolli è infatti necessario scambiare un minor numero di pacchetti in corrispondenza di variazioni topologiche e ciò si riflette in minori consumi energetici [112]. Inoltre presenta una minore latenza ovvero una maggiore velocità di risposta alle variazioni topologiche oltre che una maggiore reliability in presenza di ostacoli (condizione normale in applicazioni di smart metering) [113]. L'utilizzo del protocollo RPL permette di soddisfare i requisiti individuato nel Capitolo 2 inerenti la riconfigurabilità della rete e il supporto di reti mesh. In particolare grazie al RPL a seguito di spostamento o aggiunta di un nodo la topologia della rete viene automaticamente riconfigurata in tempi dell'ordine di qualche decina di secondi.

5.4 Conclusioni

Sulla base dei confronti e delle considerazioni suddette si ritiene che la scelta ottimale in termini di protocolli per il progetto MiSE-ENEA sia quella di utilizzare IEEE802.15.4 per quel che concerne il livello fisico e data-link, 6LowPAN/IPv6 per i livelli superiori e RPL quale protocollo di routing.

Si noti che l'utilizzo del protocollo IPv6 soddisfa il requisito di interoperabilitàidentificato nel Capitolo 2, mentre il protocollo RPL, scelto come protocollo di routing, garantisce il supporto di reti mesh e permette la riconfigurazione automatica della rete in caso di spostamento aggiunta e/o eliminazione di nodi.

Architetture simili sono state proposte in diversi progetti europei legati all'IoT (ad esempio SENSEI [114], uno dei più grandi progetti FP7 sulle WSN) oltre che in diversi prodotti commerciali inerenti il M2M come ad esempio Thread [101], InterDigital [115] e ciò avvalora ulteriormente la nostra scelta.

Va però osservato che non tutti i problemi sono risolti da tali protocolli e che molte delle piattaforme IoT basate su IPv6 sono ancora in fase di definizione (es. Thread) o alle prime sperimentazioni (es. oneM2M)

Per tale motivo l'attività di ricerca ha avuto come obiettivo la realizzazione di un testbed in grado di verificare le prestazioni di tali protocolli e determinare il set di parametri in grado di massimizzarne le prestazioni.

6 Sistemi Operativi

6.1 Introduzione

Nelle WSN low-end le risorse computazionali sono limitate al fine di minimizzare i consumi e quindi massimizzare il tempo di vita dei nodi. In tale contesto, spesso riferito come "*resource constrained",* la disponibilità di un sistema operativo (S.O.) diventa essenziale per poter gestire le risorse computazionali e di sensing in maniera efficiente. In generale un S.O. permette di schedulare le operazioni necessarie per l'esecuzione di uno o più task in modo da condividere le risorse di elaborazione (CPU, memoria, ecc.) tra i differenti task garantendo che questi siano eseguiti nell'intervallo di tempo desiderato [1]. Esempi di S.O. ampiamente noti per sistemi di elaborazione general-purpose sono Windows, Linux e MacOS. Tali S.O. non possono però essere utilizzati nelle WSN (se non per la realizzazione di gateway), basti pensare al fatto che un S.O. Windows richiede diversi GB per l'installazione contro le poche decine di kB disponibili nei mote. Nonostante tale limitazione i S.O. per WSN devono comunque gestire molte più periferiche di un S.O. convenzionale. Per esempio un mote deve essere in grado di collezionare dati da un sensore e processarli in tempo reale oltre che poter ricevere ed eseguire comandi dal Sink, operazioni non richieste ad un S.O. convenzionale. Da tali considerazioni si comprende come nella definizione di una WSN la scelta del S.O. sia fondamentale, al pari (se non più) della scelta delle caratteristiche hardware dei mote.

In particolare un S.O. per WSN ideale dovrebbe avere i seguenti requisiti:

- supportare un elevato numero di piattaforme/mote
- gestione efficiente (possibilmente dinamica) della memoria
- gestione efficiente di eventi, task e thread (overhead, context-switching, gestione priorità, politiche di scheduling, ecc.)
- richiedere per l'installazione risorse di storage compatibili con quelle dei mote
- supporto di librerie (API/componenti) per la gestione dei dispositivi
- meccanismi per l'ottimizzazione dei consumi energetici
- supporto di protocolli di rete (in particolare IP, algoritmi di routing e possibilmente protocolli M2M)

Di seguito viene riportata una sintesi dello stato dell'arte dei S.O. per WSN con particolare riferimento ai requisiti suddetti.

6.2 Stato dell'arte

Nel contesto dei sistemi embedded (ovvero sistemi di elaborazione il cui scopo è noto prima del loro sviluppo) sono stati sviluppati nel corso degli anni svariati sistemi operativi; i più noti sono riportati in Tabella 9. Di seguito verranno descritti i S.O. embedded più diffusi nell'ambito delle piattaforme per WSN: TinyOS, Contiki, Mantis, Nano-RK e LiteOS.

Tabella 9 Progetti di ricerca di S.O. per WSN [116].

OS	Web URL
TinyOS	http://www.tinyos.net
SOS	http://nesl.ee.ucla.edu/projects/sos
Contiki	http://www.sics.se/~adam/contiki
t-Kernel	http://www.cs.virginia.edu/~lg6e/t-kernel
MantisOS	http://mantis.cs.colorado.edu
EYES	http://www.eyes.eu.org/
PEEROS	http://www.eyes.eu.org/
VMSTAR	http://senses.ucdavis.edu
OSSTAR	http://senses.ucdavis.edu
MagnetOS	http://www.cs.cornell.edu/People/egs/magnetos/
Nano-QPlus	http://www.etri.re.kr
kOS	http://www.ee.ucl.ac.uk/secoas/
CORMOS	http://www.ics.forth.gr/carv/scalable/cormos.html
SenOS	http://redwood.snu.ac.kr/
DCOS	http://www.eyes.eu.org/

TinyOS

TinyOS è stato sviluppato nell'ambito del progetto *smart dust* da un consorzio guidato dall'Università della California, Berkeley in collaborazione con Intel [117]. La versione corrente di TinyOS è la 2.1.2 rilasciata il 20 Agosto 2012 anche se esistono diversi fork (a volte identificati come versione 3).

L'architettura di TinyOS è *monolitica*, questo significa che il S.O. e le applicazioni costituiscono una unica immagine fisica. Questo tipo di architettura si contrappone a quella *modulare* in cui le applicazioni e il S.O. sono compilati come un insieme di moduli distinti anche se cooperanti. La scelta fra architetture monolitiche e modulari dipende da un compromesso tra prestazioni e flessibilità. Un kernel monolitico ha il vantaggio di non introdurre un overhead necessario per l'allocazione dinamica dei moduli e richiede pertanto minori risorse di memorizzazione. D'altra parte una architettura monolitica non permette la riprogrammazione parziale dell'applicazione, ovvero per modificare anche un solo modulo si dovrà ricompilare tutta l'immagine del sistema.

Grazie alla sua architettura monolitica TinyOS può essere installato occupando soli 400 Byte tra memoria di programma e dati [118].

Alla base di TinyOS vi è il linguaggio NesC, un linguaggio definito come "un dialetto del C". La realizzazione di applicazioni in NesC è basata su delle entità computazionali indipendenti, dette *componenti,* gestite secondo un modello di tipo *event-driven.*

In pratica una applicazione TinyOS consiste nella definizione di componenti (mediante moduli) e nella loro interconnessione (mediante configurazioni). In particolare il top-level di una applicazione TinyOS è in genere una configurazione (atta ad interconnettere più componenti) mentre l'implementazione dei singoli componenti è realizzata tramite moduli.

A livello di codice, un componente è simile ad una classe di un qualunque linguaggio di programmazione orientato agli oggetti (ad esempio C++ o Java), e permette di definire metodi e variabili (o meglio *interfacce* nella nomenclatura TinyOS).

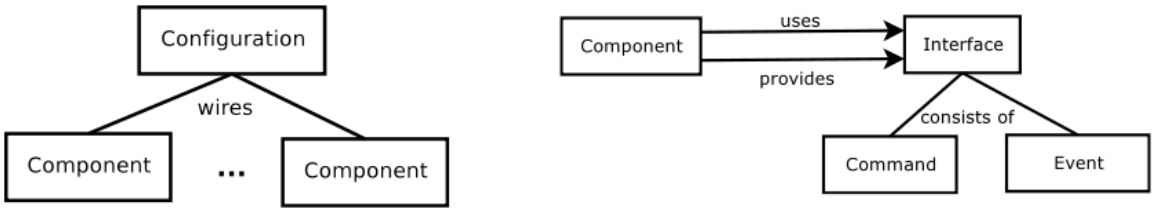


Figura 25 Principali astrazioni funzionali in TinyOS.

Un componente definisce le proprie interfacce e usa le interfacce di altri componenti.

Le interfacce sono implementate dai moduli e consistono di *comandi* ed *eventi*.

- I *comandi* permettono di richiamare i servizi offerti da altri componenti mediante la keyword *call (*possono essere considerati un'astrazione analoga ai metodi pubblici in C++).
- Gli *eventi* permettono di inviare segnali e notifiche da un componente ad un altro al completamento dell'esecuzione di un dato blocco di codice. In questo modo un componente non è costretto a bloccare il sistema per ottenere la funzione richiesta.

Un comando può eseguire più *task*. Un task è usato per operazioni interne di un componente che richiedono molte istruzioni; si richiama con la keyword *post* e permette di suddividere, così, il carico computazionale. Può essere richiamato ricorsivamente poiché la sua esecuzione è gestita da uno scheduler.

Nelle prime versioni di TinyOS i task erano gestiti dallo scheduler con un algoritmo FIFO ed erano non-preemptive (non erano previste interruzioni prima del completamento), nelle ultime versioni è stato introdotto un differente algoritmo detto EDF (Erliest Deadline First) per facilitare le applicazioni real-time.

TinyOS permette inoltre la gestione del multi-threading mediante le librerie TinyThread [119] e, a partire dalla versione 2.1, con TOS Threads [120]. TOS Threads ha un approccio di tipo coarse-grained; in pratica un thread può cedere volontariamente la sua priorità di esecuzione quando è costretto ad attendere qualche evento.

TinyOS gestisce la memoria in modo totalmente statico, non esistono quindi l'equivalente di malloc e calloc presenti nel linguaggio C; nelle ultime versioni è stato però aggiunto il supporto alla protezione della memoria mediante un compilatore chiamato Deputy che trasforma il sorgente prodotto dall'utente in un codice contenente dei controlli da eseguirsi in fase di compilazione e/o in esecuzione.

TinyOS 2.x supporta il protocollo 6LowPan che permette di fornire un indirizzo IPv6 ai mote [121], caratteristica questa che si ritiene fondamentale per gli scopi del progetto al fine ultimo di semplificare l'interazione fra gli utenti e la WSN. Altri protocolli di comunicazioni supportati sono: TYMO (una versione del protocollo DYMO per TinyOS); Dissemination e DIP (che creano delle variabili condivise da tutta la rete e permettono al Sink di inviare messaggi in broadcast), CTP (Collection Tree Protocol) che permette al Sink di collezionare i dati provenienti dai mote e RPL (Routing Protocol for Low Power and Lossy Networks) che permette di creare reti multi-hop di tipo mesh ottimizzando l'instradamento. Nelle versioni più recenti è disponibile anche il COAP, uno dei più noti protocolli M2M.

Una caratteristica importante per i S.O. sviluppati per WSN è la disponibilità di metodologie di risparmio energetico (queste verranno approfondite nel Capitolo 8). Questo è uno degli aspetti che rende TinyOS particolarmente interessante grazie al fatto di avere una gestione automatica delle modalità di consumo energetico basata su un modello di tipo split-phase e event-driven [122]. In pratica quando non ci sono task in coda da eseguire, lo scheduler di TinyOS mette automaticamente in stato di sleep il microcontrollore in modo da ridurre i consumi di potenza. Analoghe tecniche esistono anche per le unità radio per mettere in stato di idle il transceiver quando non è necessaria la trasmissione o ricezione di dati.

Contiki

Il S.O. Contiki è stato sviluppato inizialmente da Adam Dunkels al Networked Embedded Systems group dell'istituto svedese di informatica e nasce in un progetto finalizzato all'I*nternet of Things* [123]. Come TinyOS è un sistema operativo event-driven, ma a differenza di TinyOS ha un'architettura di tipo modulare. Al posto dei componenti, Contiki utilizza i *servizi* alla cui implementazione si può accedere mediante apposite interfacce. Il linguaggio di programmazione utilizzato è il C (linguaggio decisamente più conosciuto del nesC e questo ha fatto sì che di recente Contiki sia diventato il S.O. per WSN preferito da molti sviluppatori).

Anche Contiky offre il supporto multi-threading, mediante proto-thread [124], ovvero una versione dei thread per sistemi con risorse limitate, che riduce l'overhead necessario alla gestione dei thread a soli 2 bytes. Poiché la loro implementazione è sita nel livello più alto del kernel questa funzionalità risulta indipendente dalla piattaforma. Un timer del microcontrollore gestisce l'eventuale interruzione di un

thread a favore di un altro (con un meccanismo di tipo preemptive); all'overflow del timer lo stato è salvato nell'apposita area dello stack per poter essere ripreso in seguito.

Contiki offre due categorie di eventi: gli eventi sincroni, che vengono smistati immediatamente e che di fatto sono quelli a più alta priorità (possono interrompere un thread), e gli eventi asincroni, i quali vengono messi in coda e gestiti in un secondo momento dallo scheduler con un algoritmo di polling (così da gestire anche eventuali race-conditions). Gli eventi sono generalmente eseguiti fino al loro completamento, non è quindi implementato un sistema di sincronizzazione (le eventuali interruzioni sono gestite internamente dall'evento specifico).

La gestione della memoria è dinamica e prevede meccanismi per ridurre il problema della frammentazione. La memoria è suddivisa in blocchi di dimensioni fisse, delle macro sono fornite per l'allocazione (memb_alloc), e la deallocazione (memb_free). Contiki non fornisce però alcun meccanismo per la protezione della memoria tra varie applicazioni.

Contiki mette a disposizione un file system che facilita l'uso delle memorie non volatili (FLASH, EEPROM), chiamato Coffee, esso richiede 5Kb nella ROM e 0.5Kb di RAM in esecuzione. Questo file system divide la memoria in blocchi la cui dimensione generalmente coincide con quella dei blocchi della memoria flash.

Contiki supporta una vasta gamma di protocolli di comunicazione, in particolare il µIP, una versione di protocollo IP con un set di funzionalità ridotte. µIP è compatibile sia con IPv4 che con IPv6, supporta inoltre i protocolli TCP, UDP ed ICMP, ma ne può utilizzare solo uno per volta. Esiste inoltre un altro protocollo detto Rime leggero e molto flessibile. Contiki offre inoltre un'implementazione di 6LowPan e del RPL (Routing Protocol for Low Power and Lossy Networks) chiamata ContikiRPL. Contiki implementa anche una versione del COAP.

Mantis

Mantis è l'acronimo di MultimodAl system for NeTworks of In-situ wireless Sensors ed è un progetto sviluppato da un gruppo di ricerca dell'Università del Colorado, Boulder [125]. Pensato inizialmente per piattaforme PDA o PC, successivamente è stato implementato anche sui nodi sensori. L'architettura è stratificata, ogni livello funge da macchina virtuale per il livello superiore. Anche questo S.O., come Contiki, è sviluppato in linguaggio C.

Il sistema operativo Mantis non è event-driven, ma thread-driven. Supporta nativamente il multitasking pre-emptive; il numero massimo di task che può gestire contemporaneamente è però fissato, impostato di default a 12. Un meccanismo di context-switching suddivide il tempo per i thread attivi assegnando dei time-slots che di default sono di 10ms ciascuno. Le race-conditions sono gestite da un sistema a semaforo; ogni thread può avere 5 livelli di priorità, l'algoritmo di scheduling, di tipo round-robin, esegue prima gli elementi a priorità più alta, terminati i quali, esegue quelli a priorità più bassa. Questo sistema di priorità permette di realizzare applicazioni con funzionalità real-time, anche se Mantis non rientra tra i sistemi operativi real-time, perché l'algoritmo di scheduling non prevede alcun controllo delle deadline di un thread. Dal punto di vista della quantità di memoria allocata per ogni thread, la gestione dei thread introdotta in Mantis risulta meno efficiente rispetto ai S.O. precedentemente trattati. Si ritiene che l'overhead introdotto sia significativo per sistemi resource-constrained come le WSN.

La memoria in Mantis è usata per lo più staticamente, è permesso l'utilizzo di malloc e calloc ma è disincentivato dagli stessi produttori, inoltre non è previsto alcun meccanismo di protezione per la memoria.

Per quanto riguarda i protocolli di comunicazione; lo stack di Mantis è suddiviso in due parti. La prima, quella più in alto nella pila ISO-OSI, risiede nello stesso layer riservato all'utente; questo da un lato aumenta la flessibilità, permettendo ad esempio all'utente di implementare il proprio algoritmo di routing, dall'altro aumenta l'overhead dal momento che non è possibile usare direttamente le funzioni offerte dal S.O. ma è necessario utilizzare le API di Mantis per accedere ad ogni funzione del sotto-sistema di comunicazione. La seconda parte dello stack è implementata nel layer COMM che unisce tutte le funzionalità di comunicazione a basso livello, accomunando interfaccia seriale, USB e radio. Non è però disponibile alcuna API per le applicazioni multicast.

Nano-RK

È un progetto della Carnegie Mellon University [125], attualmente è però limitato all'utilizzo di due soli modelli di mote, il MicaZ e fireFly. È un sistema operativo con una architettura monoltica, molto leggero (kernel, scheduler e stack di rete occupano insieme appena 500 bytes); è orientato al multithreading ed inoltre è tra i pochi sistemi operativi per le WSN che supporta delle API per applicazioni real-time. Nano-RK è sviluppato in linguaggio C.

L'approccio scelto per la gestione dei task è di tipo statico, ovvero in fase di compilazione devono essere già note e fissate le deadline e la suddivisione delle risorse, sebbene sia possibile cambiare questi parametri anche in tempo reale; questa soluzione è però sconsigliata dagli stessi autori. Anche la gestione della memoria è interamente statica, inoltre non è previsto alcun meccanismo per il controllo della memoria.

A differenza di sistemi operativi orientati event-driven, Nano-RK ricorre al paradigma di programmazione multi-tasking, che essendo molto più noto ai programmatori ne permette una maggiore velocità di apprendimento. Questa facilitazione, però, ha un prezzo: i thread sono di tipo full pre-emptive, quindi è permessa l'interruzione di un thread a favore di un altro a priorità maggiore in qualunque momento, quest'interruzione richiede il salvataggio dello stato del processore in memoria. Questa operazione, detta context-switch, è eseguita da un Task Control Block che si occupa di salvare non solo i dati presenti nel task, ma anche i parametri relativi alla priorità, deadline, banda e CPU; se eseguita frequentemente causa uno spreco di memoria e di energia che si ripercuote negativamente sulle prestazioni del sistema.

Lo scheduler mette a disposizione due tipi di algoritmo a semafori (livelli di priorità) per la gestione dei task: il Rate Monotonic, che è utilizzato per i task periodici real-time (in tal caso priorità e periodo vanno impostati staticamente da codice) ed il Rate Harmonized, un algoritmo che permette di ottimizzare l'allocazione temporale dei task in modo da ridurre al minimo i periodi di idle del microcontrollore. È inoltre implementato un algoritmo detto Priority Ceiling per prevenire il problema ricorrente dell'inversione di priorità.

Nano-RK offre un servizio di comunicazione che mette a disposizione degli oggetti simili a socket con indirizzamento a 16 bit (gestiti direttamente dall'applicazione specifica e non dal S.O.). A livello data-link è stato implementato un protocollo detto RT-Link basato su un algoritmo di duty-cycling sincrono (per coerenza al supporto real-time) che prevede che un dato periodo temporale sia diviso in due parti dette CS (Contention Slots) e SS (Scheduled Slots); durante la prima parte i nodi accedono casualmente al canale mediante un protocollo a contesa asincrono per prenotare uno dei time slot contenuti nella seconda parte. Inoltre sono previste della API che a seconda della priorità di un'applicazione possono aumentare o diminuire la banda riservata.

LiteOS

LiteOS è un sistema operativo sviluppato dall'Università dell'Illinois Urbana-Campaign [126] ed ha un'architettura modulare, simile a Unix, dal quale eredita un file-system gerarchico e il supporto per la programmazione ad oggetti mediante un linguaggio ad alto livello chiamato Lite C++. È orientato al multi-threading ma supporta anche la programmazione ad eventi.

L'architettura è suddivisa in tre parti:

- **Satellite:** è un ambiente a riga di comando che permette di interagire con i nodi della rete e che risiede direttamente sulla base-station (il che riduce l'occupazione di ROM). Il comando è interpretato dalla base-station e inviato al mote via radio; il nodo così può rispondere con dei parametri anziché con una query in formato testuale, il che riduce la quantità di dati trasmessi (e di conseguenza i consumi, come si vedrà nel Capitolo 8).
- **LiteFS**: è il file-system che mette a disposizione una gerarchia con file e cartelle del tutto simile a quella di Unix, permettendo di montare dispositivi radio e memoria dati nel file system gestendoli come file, offrendo un accesso semplice e unificato in lettura e scrittura a tali componenti.
- **Kernel**: il cuore del sistema offre sia funzioni di callback per l'uso del paradigma ad eventi che uno

scheduler per il multithreading; in particolare lo scheduler è di tipo round robin basato sulle priorità. Per prevenire il problema della corsa critica è disponibile una direttiva che rende un blocco di istruzioni atomiche (ovvero eseguite senza possibilità di interruzione). L'algoritmo scelto per lo scheduler permette l'esecuzione di un task fino al complemento, rendendo questo sistema inadatto alle applicazioni di tipo real-time.

La memoria è di tipo heap, è gestita dinamicamente e cresce in direzione opposta rispetto allo stack di sistema. A differenza di altri S.O., il kernel LiteOS non si compila assieme all'applicazione; questo permette di separare lo spazio di indirizzamento di sistema da quello dell'applicazione, offrendo nativamente un sistema di protezione della memoria.

Attualmente LiteOS è supportato solo da piattaforme MicaZ ma potrebbe essere facilmente modificato per un suo utilizzo anche per piattaforme simili quali Iris o Tmote.

Squawk VM

Squawk [127] non è un vero S.O. ma una Virtual Machine (VM) sviluppata in Java che può essere eseguita senza S.O. in una WSN. Le funzionalità di Squawk includono la gestione degli interrupt, dei protocolli di rete e delle risorse computazionali. Squawk gestisce, inoltre, i thread con un sistema a semaforo che blocca i thread con minore priorità durante l'esecuzione dei thread a più alta priorità. Dato che Squawk non viene eseguito su un S.O. lo spazio di memoria occupato dal sistema si riduce, risultando pari a 80KB per la VM e 250KB per le librerie (CLDC, Connected Limited Device Configuration) [128], che suppliscono alla mancanza di un S.O. su cui viene eseguita la VM. Lo spazio richiesto per Squawk è inferiore quindi ad un comune Linux embedded [129], ma comunque decisamente superiore rispetto a TinyOS e agli altri S.O. trattati in precedenza. Tale differenza è principalmente legata al fatto che Squawk e i relativi driver, compreso quelli inerenti il protocollo 802.15.4, sono scritto in Java e non in C o NesC.

La gestione dei blocchi di codice viene fatta traducendo le classi standard in blocchi di codice pre-linkati e memorizzati in una ROM [128]. Squawk implementa il meccanismo di isolamento dell'applicazione, che consiste nella rappresentazione delle applicazioni in termini di oggetti. Tale rappresentazione rende possibile l'esecuzione contemporanea di diverse applicazioni.

Nonostante la scelta di sviluppare Squawk come una VM sia stata fatta con l'intento di rendere la piattaforma software portabile in tutti i sistemi embedded, attualmente l'unica piattaforma WSN che include Squawk è Sun SPOT, prodotta dalla stessa Sun Microsystems. Non si ritiene quindi sia semplice fare un porting su piattaforme differenti.

RIOT

RIOT [130] è un S.O. sviluppato da INRIA per applicazioni IoT. Ha una architettura monolitica e supporta il multi-threading. Le caratteristiche che lo rendono adatto all'IoT sono il ridotto spazio occupato dal S.O., l'efficienza dal punto di vista del consumo energetico, le funzionalità real-time e la gestione modulare e configurabile dello stack. RIOT implementa un microkernel che si occupa della gestione del thread, ha uno scheduler degli eventi con gestione delle priorità e una API per la comunicazione tra i differenti processi (IPC, Inter Process Communication). Il microkernel ha, inoltre, delle librerie per la crittografia dei dati, supporta la programmazione con strutture dati come hash table e query con priorità e fornisce anche una shell di comandi. Un importante requisito per garantire che l'esecuzione runtime abbia una durata costante è che l'allocazione della memoria nel kernel sia statica e gestita da una lista circolare e pre-lincata di thread a dimensione fissa. L'allocazione della memoria per le applicazioni è invece dinamica [131]. Per ottimizzare le prestazioni per il consumo di potenza, il S.O. entra in uno stato di idle ogni volta che non ha thread in sospeso da eseguire. RIOT è compatibile con varie piattaforme, come TelosB, STM32, Cortex-M3 e, in genere, tutte le piattaforme basate su microcontrollore MSP430 [132]. Le applicazioni per RIOT sono scritte in C/C++ e questo rende più agevole la curva di apprendimento. Implementa diversi protocolli fra cui 6LowPAN, COAP, RPL e l'UDP. Gli ambienti di sviluppo disponibili per RIOT sono gcc (GNU Compiler Collection, un compilatore Linux), valgrind e gdb (GNU Debugger).

OpenWSN

OpenWSN [133] è stato sviluppato nel 2010 dalla Berkeley University of California all'interno del progetto Berkeley OpenWSN Project. Si tratta di un S.O. open source compatibile con una vasta varietà di piattaforme hardware e orientato all'IoT. Come RIOT, TinyOS e Contiki, anche OpenWSN supporta i protocolli 6LowPAN, RPL e Coap [134]. Il linguaggio utilizzato per sviluppare le applicazioni è il C. L'architettura è monolitica e supporta il multi-threading.

OpenWSN utilizza due livelli di astrazione: il Berkekey Socket Abstraction (BSA) e l'Hardware Abstraction (HA). Il primo livello di astrazione (BSA) fa sì che due applicazioni differenti comunichino attraverso socket, univocamente definiti da indirizzi IP e numeri di porta. Grazie a tale livello lo sviluppo di un'applicazione OpenWSN è simile ad un applicazione di rete sviluppata su un normale PC. Il secondo livello di astrazione (HA) si occupa di raggruppare tutte le funzioni che comunicano con l'hardware (come ad esempio le funzioni che scrivono nei registri) in un insieme di file chiamati Board Support Package (BSP). In pratica esiste un BSP per ogni piattaforma supportata e questo permette alla maggior parte del codice di essere condiviso tra tutte le piattaforme.

Il kernel di OpenWSN include uno scheduler, OpenOS, che si occupa di gestire le priorità di esecuzione dei task. Anche OpenWSN, come RIOT, entra in modalità di deep-sleep quando non ci sono task (o thread) da eseguire, per rendere più efficiente il consumo di potenza. Lo scheduler è non-preemptive, che significa che un task che è in esecuzione non viene interrotto da altri task, che vengono messi in coda. Una funzionalità interessante è che lo scheduler OpenOS non è legato al kernel OpenWSN, di conseguenza lo stack dello scheduler può essere eseguito come una parte di un differente S.O..

6.3 Confronto S.O.

In questa sezione è riportata una sintesi di alcuni lavori presenti in letteratura inerenti il confronto dei S.O. precedentemente analizzati

Un confronto particolarmente interessante è proposto in [116] nel quale gli autori analizzano i S.O. TinyOS, Contiki, Mantis, LiteOS e Nano-RK confrontandoli in termini di scalabilità, riprogrammazione dinamica, consumi energetici e possibilità di gestire processi real-time in diversi scenari applicativi (monitoraggio ambientale indoor/outdoor, telemedicina, applicazioni industriali e militari).

Al termine del confronto gli autori concludono l'analisi osservando che per applicazioni di monitoraggio ambientale indoor tutti i sistemi operativi appena elencati risultano equivalenti mentre per applicazioni outdoor una preferenza si ha per Contiki, Mantis e Nano-RK. Per applicazioni industriali il S.O. più adatto risulta Nano-RK (per la capacità di gestire processi real-time). D'altra parte Nano-RK non è supportato dalle piattaforme disponibili per WSN come si evince dalla Tabella 10 che riporta una griglia con le piattaforme WSN precedentemente analizzate e i sistemi operativi da esse supportati.

Tabella 10 Sistemi Operativi supportati e piattaforme di sviluppo.

	TelosB/TmoteSky	CM5000	MicaZ/ Mica2	IRIS	SunSPOT	EZ430-Fxxxx	Waspmote	Z1	OpenMote
Contiki	■	■		■				■	■
TinyOS	■	■	■	■		■		■	■
Mantis OS	■		■						
Mote Runner				■					
Sqwawk VM					■				
Nano-RK									
Lite OS			■						
RIOT								■	■
OpenWSN								■	■

Dall'analisi della Tabella 10 si osserva che TinyOS è il sistema operativo che ad oggi supporta il maggior numero di piattaforme (in pratica tutte quelle elencate con eccezione dei Waspmote e SunSPOT). Ad eccezione di TinyOS, Contiky e OpenWSN tutti gli altri S.O. si limitano al supporto di una o due piattaforme. Ciò può essere in parte giustificabile per S.O. recenti (ad esempio OpenWSN) ma diversamente si ritiene che un S.O. con diversi anni di sviluppo e poche piattaforme supportate possa riflettere una difficoltà intrinseca di adattamento nel contesto resource-constrained delle WSN.

In [122] è riportato un altro confronto tra i sistemi operativi TinyOS, Contiki e LiteOS valutando la capacità di utilizzo di risorse limitate, di eseguire operazioni concorrenti, la flessibilità e i consumi energetici. Da tale confronto appare chiaro che:

- LiteOS è il S.O che richiede minori risorse seguito da TinyOS;
- TinyOS è l'unico S.O. monolitico. Questo rende TinyOS meno flessibile ma più efficiente rispetto all'utilizzo della memoria. Un S.O. monolitico ha infatti il vantaggio di non introdurre un overhead necessario per l'allocazione dinamica dei moduli e richiede pertanto minori risorse di memorizzazione.
- Dal punto di vista della gestione concorrente tutti e tre i S.O. hanno un kernel di tipo event-driven, ma Contiki offre nativamente anche librerie multi-threading. Recentemente però librerie simili sono state proposte anche per TinyOS [119].
- Tutti e tre i S.O. sono flessibili e in grado di gestire differenti tipi di applicazioni. Tuttavia, quando è necessario aggiornare il firmware di una applicazione, Contiki e LiteOS possono sostituire dinamicamente una o più parti dell'applicazione, mentre con TinyOS deve essere sostituito tutto il firmware a causa dell'architettura monolitica. Ci sono tuttavia lavori in letteratura che mostrano come sia possibile superare tale limite di TinyOS.

È inoltre importante sottolineare che non sempre una allocazione della memoria dinamica è vantaggiosa, poiché deve essere accompagnata da metodologie di protezione della memoria necessarie per evitare bug difficilmente rilevabili. Con una memoria allocata in modo statico è semplice evitare che lo spazio di memoria di una applicazione non venga scritto da altre con semplici controlli già in fase di compilazione (esistenti in TinyOS). Quando l'allocazione della memoria è invece dinamica (come in Mantis e Contiki) servono tecniche più complesse per effettuare la protezione della memoria non sempre implementate. In Mantis, per esempio, la gestione della memoria è dinamica ma senza alcuna protezione tanto che l'utilizzo dell'allocazione dinamica è di fatti scoraggiata [116]. Contiki ha una gestione della memoria dinamica con alcune metodologie di protezione (`memb_alloc()` e `memb_free()`), ma non implementa metodologie

di protezione della memoria tra una applicazione e l'altra. Nano-RK ha una gestione della memoria statica ma, al contrario di TinyOS, senza meccanismi di protezione. Gli sviluppatori devono tener conto nell'allocazione della memoria di questa mancanza.

A ciò si aggiunge il fatto che da un confronto effettuato fra LiteOS e TinyOS riportato in [126], nonostante LiteOS abbia una gestione della memoria dinamica non sembra presentare particolari vantaggi rispetto a TinyOS (ad eccezione della minore dimensione del kernel). Dai benchmark utilizzati per il confronto dei due S.O. LiteOS risulta infatti avere prestazioni comparabili ma non migliori. Si ritiene quindi di poter concludere che la gestione dinamica della memoria non sia essenziale per lo sviluppo delle applicazioni inerenti WSN.

In [124] è poi riportato un confronto tra Contiki e TinyOS in termini di consumi energetici da cui risulta vincente TinyOS. In particolare gli autori osservano che i S.O. event-driven (es. TinyOS) sono preferibili ai S.O. thread-driven in termini di gestione dei consumi energetici. La gestione thread-driven inoltre introduce un overhead proporzionale al numero di thread che porta ad un maggior consumo di memoria da parte del S.O. a scapito della memoria disponibile per l'applicazione. Questo è uno dei motivi per cui TinyOS, nonostante abbia una gestione della memoria statica e quindi meno flessibile, risulti più efficiente dal punto di vista della gestione della memoria e dei consumi di potenza. Inoltre, sempre per quanto riguarda le modalità di gestione e ottimizzazione dei consumi energetici, TinyOS ha meccanismi già integrati che non necessitano quindi della scrittura di codice ad-ok, mentre Contiki richiede che tale aspetto sia affrontato da parte di chi progetta l'applicazione.

Alla luce di tali considerazioni e del fatto che per il progetto MiSE-ENEA la minimizzazione dei consumi energetici e la disponibilità di piattaforme sono requisiti prioritari rispetto alla possibilità di una riprogrammazione dinamica, si ritiene che TinyOS debba essere il S.O. da preferire per il progetto. TinyOS presenta però come unico svantaggio la necessità di sviluppare in un linguaggio, nesC, meno conosciuto rispetto ad altri (C, C++, Java) e ciò potrebbe comportare tempi di sviluppo maggiori.

Sistema Operativo	Architettura	Modello di programmazione	Linguaggio di programmazione	Piattaforme supportate
TinyOS	Monolitico	Event-driven con priorità + TOS Threads	NesC	TelosB/Tmote, CM5000, MicaZ, IRIS, EZ430, Z1, OpenMote
Contiki	Modulare	Protithreads e eventi	C	TelosB/Tmote Sky, CM5000, IRIS, EZ430, Z1, OpenMote
Mantis OS	A livelli	Thread	C	MicaZ, TelosB/Tmote
Nano-RK	Monolitico	Thread	C	N/A
LiteOS	Modulare	Thread ed eventi	LiteOS++	MicaZ
RIOT	Monolitico	Multi-threading	C/C++	Z1, OpenMote
OpenWSN	Monolitico	Multi-threading	C	Z1, OpenMote

6.4 Conclusioni

In questo Capitolo sono stati analizzati i principali sistemi operativi per reti di sensori. A fronte dell'analisi dello stato dell'arte è stato individuato in TinyOS il sistema operativo più idoneo per lo sviluppo di applicazioni M2M sia per le basse risorse computazionali, sia perché fornisce meccanismi per la riduzione dei consumi energetici. Inoltre è supportato da un elevato numero di piattaforme e implementa nativamente i protocolli IPv6 e COAP.

7 Sicurezza

7.1 Introduzione: gestione della sicurezza (security), della riservatezza (privacy) e della affidabilità (trust) in IoT

Lo sviluppo delle comunicazioni automatiche fra gli oggetti potrebbe rappresentare un pericolo per il nostro futuro [135]. Inoltre, senza accorgercene, tag RFID inseriti nei nostri dispositivi, nei nostri vestiti o generi alimentari possono a nostra insaputa essere attivati per trasmettere il loro identificativo o altre informazioni. In questo modo si realizza potenzialmente un meccanismo di sorveglianza in grado di pervadere una buona parte delle nostre vite.
Lo sviluppo di IoT deve quindi includere funzioni relative alla gestione della affidabilità, riservatezza e sicurezza di tutti i dati che vengono scambiati. Queste funzioni potrebbero essere concentrate in uno specifico strato della pila protocollare o essere distribuite sull'intera pila, dall'oggetto di astrazione alla composizione del servizio, in una maniera tale da non degradare le prestazioni del sistema o introdurre un eccessivo sovraccarico.

IoT è estremamente vulnerabile agli attacchi di **sicurezza (security)** per diverse ragioni: i dispositivi sono spesso incustoditi e quindi si prestano facilmente ad attacchi fisici; la maggior parte delle comunicazioni sono wireless e quindi le intercettazioni sono estremamente semplici; la maggior parte dei dispositivi hanno scarsa capacità di calcolo e di energia e quindi non possono implementare meccanismi complessi di sicurezza. Nello specifico la maggior parte dei problemi relativi alla sicurezza riguarda l'autenticazione e l'integrità dei dati. L'autenticazione risulta complicata perché necessita di infrastrutture appropriate che realizzino l'obiettivo attraverso lo scambio di appropriati messaggi con gli altri nodi. In IoT tale approccio non è possibile per il fatto che diversi dispositivi, come i tag RFID o i nodi sensore, non possono scambiare molti messaggi con i server di autenticazione perché esaurirebbero velocemente la loro energia.

Uno degli attacchi più difficili da risolvere è quello conosciuto con il nome di "man-in-the-middle attack" (letteralmente attacco con l'uomo in mezzo). Si consideri il caso in cui un nodo è utilizzato per identificare qualcosa o qualcuno e, di conseguenza, fornisce accesso ad un certo servizio o ad una certa area (ad esempio si consideri un passaporto elettronico o alcune chiavi elettroniche basate su RFID). Il seguente attacco potrebbe essere condotto con successo: si consideri il caso in cui un nodo A vuole autenticare altri elementi del sistema attraverso un meccanismo RF e che un "aggressore" voglia rubare l'identità di un elemento B. L'aggressore potrà posizionare due transceiver: il primo vicino al nodo A, che chiamiamo B0, e il secondo vicino al nodo B, che chiamiamo A0. L'idea di base dell'aggressore sarà quella di fare in modo che il nodo A consideri che il nodo B0 sia il nodo B e che il nodo B consideri che A0 sia il nodo A. Con questo obiettivo, il nodo B0 trasmetterà il segnale di richiesta ricevuto dal nodo autenticante A al transceiver A0. Il transceiver A0 trasmetterà tale segnale così che il nodo B possa riceverlo. Il segnale trasmesso da A0 è una replica esatta del segnale trasmesso da A. Di conseguenza sarà impossibile per il nodo B capire che il segnale non sia stato trasmesso da A. Analogamente sarà impossibile per il nodo B capire che il segnale non è stato trasmesso da A e, quindi, risponderà con i suoi dati di identificazione. Il nodo A0 riceverà tale risposta e la trasmetterà al nodo B0 che la trasmetterà al nodo A. Il nodo A non potrà distinguere che tale risposta non sia stata trasmessa dal nodo B e, quindi, identificherà il transceiver B0 come il dispositivo B e gli fornirà l'accesso. Si osservi che questo meccanismo può essere implementato indipendentemente dal fatto che i segnali trasmessi siano crittografati o meno.

Le soluzioni per l'integrità dei dati dovrebbero garantire che un aggressore non possa modificare i dati da trasmettere senza che il sistema ne rilevi il cambiamento. I dati possono essere modificati dagli aggressori sia mentre sono memorizzati nei nodi sia mentre vengono trasferiti. Per proteggere i dati contro il primo tipo di attacco si utilizzano memorie protette (ad esempio i tag *EPCglobal Class-1 Generation-2* e *ISO/IEC 18000–3* proteggono sia le operazioni di lettura sia quelle di scrittura attraverso una password). Per proteggere i dati contro il secondo tipo di attacco, i messaggi possono essere protetti utilizzando degli schemi di codifica come il *Keyed-Hash Message Authentication Code (HMAC)*. È un sistema basato sulla

condivisione di una chiava comune segreta fra il tag e la destinazione del messaggio utilizzato in combinazione con una funzione hash per fornire l'autenticazione. Si osservi che una soluzione come la precedente può avere diverse problematiche. Infatti la lunghezza della password supportata dalla maggior parte delle tecnologie tag è troppo corta per permettere robusti sistemi di protezione. Inoltre, anche se lunghe password sono supportate, la loro gestione risulta un compito arduo specialmente quando si utilizzano dispositivi di diversi produttori, come nel caso di IoT. Infine si osservi che tutte le soluzioni proposte per supportare la sicurezza utilizzano alcune metodologie di crittografia. Gli algoritmi di crittografia consumano una parte considerevole delle risorse sia in termini di energia che di larghezza di banda alla sorgente e alla destinazione. Tali soluzioni devono essere opportunamente adattate per girare su dispositivi (come tag RFID e nodi sensori) che sono particolarmente vincolati in termini di capacità energetiche, di comunicazione e computazionali.

Il concetto di **riservatezza (privacy)** è profondamente radicato nella nostra civiltà ed è riconosciuto in tutte le legislazioni dei paesi civilizzati. Come già detto le preoccupazioni relative alla protezione della riservatezza sono una notevole barriera contro la diffusione delle tecnologie coinvolte in IoT. Tali preoccupazioni sono ben giustificate in quanto i modi con i quali i dati vengono collezionati, estratti e consegnati in IoT sarà completamente diverso da quello con il quale siamo abituati e ci sarà un elevatissimo numero di occasioni per collezionare dati personali. Per i singoli individui potrebbe diventare impossibile controllare e scoprire se e dove vengono conservate informazioni private. Inoltre, la continua diminuzione dei costi per la memorizzazione delle informazioni potrebbe portare con molta probabilità al fatto che queste vengono conservate infinitamente con un significativo impatto sull'oblio digitale nella prospettiva delle persone. Ne deriva che IoT rappresenta un ambiente nel quale la riservatezza degli individui è seriamente minacciata in diversi modi. Inoltre, mentre i problemi di riservatezza di Internet riguardano principalmente utenti che utilizzano la rete (che hanno un ruolo attivo), nello scenario di IoT i problemi di riservatezza riguarderanno anche persone che non utilizzano per niente i servizi di IoT. La riservatezza dovrebbe essere garantita in maniera tale che ciascun individuo possa controllare quali dati personali siano collezionati, chi colleziona i dati e quando questo accade. Inoltre i dati personali collezionati dovrebbero essere utilizzati solamente con lo scopo per il quale è stato autorizzato il servizio e, infine, i dati dovrebbero essere memorizzati solamente per il tempo strettamente necessario.

Per esempio si consideri uno scenario applicativo di una rete di sensori che renda più confortevole gli ambienti di una casa o di un ufficio e, in particolare, si consideri il caso di un palazzo dove sono concentrati diversi uffici. Alcuni sensori saranno posizionati nell'ambiente per tracciare la posizione delle persone in maniera da controllare l'illuminazione e il riscaldamento. Se il sistema è installato solo per incrementare la comodità degli ambienti e ridurre il consumo energetico dovranno essere applicate regole di riservatezza che garantiscano: 1) che il sistema di tracciatura della posizione non collezioni informazioni che riguardino i singoli individui ma consideri solo gruppi (in maniera tale che la posizione e il movimento degli individui non possa essere associata all'identità di una determinata persona); 2) le persone siano informate delle motivazioni e delle modalità con le quali i loro movimenti sono tracciati dal sistema (la maggior parte delle legislazioni richiedono che le persone siano informate ogni qual volta la loro riservatezza possa essere violata); 3) i dati collezionati dal sistema dovrebbero essere elaborati esclusivamente per le motivazioni di illuminazione e riscaldamento e cancellati dalla memoria del sistema appena non più necessari.

Per gestire in maniera appropriata il processo di collezione dei dati è necessario che in tutti i sottosistemi di IoT che interagiscono con delle persone venga presa in considerazione la riservatezza.

Nel contesto di Internet tradizionale il gruppo W3C ha definito la "Piattaforma per le preferenze di riservatezza" - "Platform for Privacy Preferences" (P3P) che fornisce un linguaggio per la descrizione delle politiche e delle impostazioni di riservatezza e quindi permette una gestione automatica dei parametri relativi alla riservatezza in funzione dei requisiti impostati dall'utente. Sempre nel contesto dei servizi tradizionali di Internet, attraverso l'appropriata impostazione delle applicazioni che girano sui terminali degli utenti, i momenti nei quali le informazioni personali vengono rilasciate può essere facilmente rilevato e l'entità che li sta acquisendo può essere identificata attraverso ben definite procedure di autenticazione.

Il problema diventa di impossibile soluzione nel caso di una rete di sensori. Infatti, le persone che entrano in un'area dove è installata una rete di sensori non possono controllare le informazioni che saranno collezionate e, in particolare, che riguardano la loro riservatezza. Ad esempio, si consideri una rete di telecamere installate in una determinata area. Il solo modo che gli individui hanno per evitare che le telecamere riprendano la loro immagine sarà quello di non entrare nell'area. In questo contesto una possibile soluzione è quella di ridurre la capacità della rete di collezionare dati ad un livello di dettaglio tale che possa compromettere la riservatezza. Per esempio, una rete di sensori potrebbe rendere anonimi i dati rilevando solo in maniera approssimata la posizione delle persone trovando un compromesso con il livello di dettaglio richiesto dall'applicazione. Un altro esempio relativo alla rete di sensori composta da telecamere per videosorveglianza è quello di poter sfocare le immagini delle persone per proteggere la loro riservatezza. Se si dovessero verificare alcuni eventi che lo richiedono le immagini relative ad alcune persone potrebbero essere ricostruite (non sfocate) con l'autorizzazione della magistratura.

Nel caso di sistemi a RFID il problema della riservatezza diviene duplice. Infatti se da un lato spesso i tag RFID sono passivi e rispondono esclusivamente alle interrogazioni dei lettori indipendentemente dalla volontà del proprietario, dall'altro lato un aggressore può intercettare la risposta da un tag e trasmetterla a un altro lettore. Le soluzioni proposte richiedono che i tag siano in grado di eseguire procedure di autenticazione. Questo comporta più elevati costi dell'infrastruttura di autenticazione che non può essere sviluppata in sistemi complessi come quelli degli scenari di IoT. Sono state proposte delle soluzioni che usano parametri impostati dagli utenti che permettono di negoziare la propria riservatezza. Per evitare le intercettazioni si potrebbe proteggere la trasmissione dei dati utilizzando sistemi di crittografia. Comunque questo tipo di soluzione permette ancora a lettori non autorizzati di rilevare la presenza del tag RFID. Per risolvere questo problema si possono utilizzare delle soluzioni nelle quali il segnale che viene trasmesso dal lettore ha la forma di un pseudo-rumore (in pratica non si riesce a rilevare l'incremento di potenza dovuto alla trasmissione del segnale in maniera tale che l'attivazione della trasmissione non possa essere rilevata da lettori non autorizzati).

Per assicurare che i dati collezionati siano utilizzati solo per i servizi e dai fornitori autorizzati sono state proposte delle soluzioni che fanno affidamento a sistemi chiamati "mediatori di riservatezza" (privacy broker). Un proxy interagisce con l'utente da una parte e con i servizi dall'altra garantendo che il fornitore ottenga esclusivamente l'informazione dell'utente che è strettamente necessaria. L'utente potrà impostare le preferenze del proxy ma non potrà impostare o controllare le politiche utilizzate dallo stesso. Inoltre tale soluzione non sarà affatto scalabile. Infine per ciò che riguarda il problema dell'oblio digitale, appare complicato adottare nell'ambito IoT alcune tecniche che sono state implementate in Internet (si consideri ad esempio la possibilità che viene data agli utenti che condividono immagini o altri tipi di file nei social network di fare in modo che questi scadano dopo un certo periodo di tempo e vengono automaticamente cancellati dalle memorie).

7.2 Meccanismi di sicurezza (Security Mechanism)

La sicurezza è un importante campo delle WSN e, come detto nel precedente Paragrafo, i suoi meccanismi devono differire rispetto a quelli tradizionali in particolare per due diverse ragioni [136]: 1) vincoli sulle capacità energetiche, computazionali e di comunicazione dei dispositivi; 2) rischi legati ad attacchi di natura fisica come la cattura e la manomissione dei nodi.

Si possono distinguere due tipi di meccanismi: a basso livello e ad alto livello.

Meccanismi a basso livello (low level mechanism)

- *Costituzione delle chiavi e inizializzazione della affidabilità (Key establishment and trust setup)*

  Il primo requisito per poter inizializzare una rete di sensori sicura è la costituzione delle chiavi di crittografia. In genere i sensori hanno una scarsa capacità computazionale e gli algoritmi di crittografia a chiave pubblica sono troppo complessi per essere eseguiti. Le tecniche di costituzione della chiave devono inoltre essere scalabili per reti con centinaia o migliaia di nodi. I sensori

possono avere la necessità di inizializzare chiavi con i nodi vicini e con i nodi di aggregazione dei dati. Lo svantaggio di questo approccio è che aggressori in grado di ascoltare per lunghi periodo di tempo e distribuiti sul territorio della rete potrebbero anche ricostruire l'insieme completo delle chiavi e interrompere il funzionamento del sistema.

- *Segretezza e autenticazione (Secrecy and authentication)*

  La maggior parte delle applicazioni per reti di sensori richiedono protezione contro le intercettazioni, l'intrusione e la modifica dei pacchetti. Il modo standard per difendersi è dato dalla crittografia. Si deve scendere comunque a notevoli compromessi quando si introduce la crittografia in una rete di sensori. Per le comunicazioni punto-punto la crittografia end-to-end permette di ottenere un elevato livello di sicurezza ma richiede che le chiavi siano impostate fra tutti gli end-point e questo può essere non compatibile in presenza di dispositivi passivi e dispositivi che operano in modalità diffusiva (broadcast). La crittografia a livello data-link con una chiave condivisa per tutta la rete semplifica il processo di inizializzazione della chiave e supporta la partecipazione di dispositivi passivi e di dispositivi che operano in modalità broadcast, ma nodi aggressori intermedi potrebbero intercettare o alterare i messaggi.

- *Riservatezza (Privacy)*

  Come tutte le reti di comunicazione e come discusso nell'introduzione le reti di sensori hanno grosse problematiche legate alla riservatezza.

- *Robustezza verso la negazione del servizio di comunicazione (Robustness to communication denial of service)*

  Un avversario potrebbe tentare di disturbare le operazioni della rete attraverso la diffusione di un segnale ad alta energia. Se questa trasmissione è sufficientemente potente l'intero sistema di comunicazione potrebbe bloccarsi.

- *Instradamento sicuro (Secure routing)*

  L'instradamento e l'inoltro dei dati è un servizio cruciale per permettere le comunicazioni in una rete di sensori. Sfortunatamente i protocolli di instradamento sono molto vulnerabili alla sicurezza. Per esempio un aggressore potrebbe lanciare un attacco di negazione del servizio verso il protocollo di instradamento, impedendo la comunicazione. Il più semplice attacco prevede l'inserimento di informazioni di instradamento contraffatte nella rete che hanno come conseguenza l'inconsistenza delle regole di instradamento stesse nella rete.

- *Robustezza alla cattura fisica dei nodi (Resilience to node capture)*

  Una dei problemi più impegnativi nelle reti di sensori è la sicurezza contro aggressori che possono catturare uno o più nodi. In molte applicazioni i nodi sensori sono posizionati in posti facilmente accessibili agli eventuali aggressori. Tale esposizione permette che un aggressore possa fisicamente catturare i nodi sensore, estrarre le chiavi di crittografia, modificare il firmware del sensore o sostituirlo con uno che sia sotto il suo controllo. Custodie resistenti alla manomissione potrebbero essere una possibile difesa ma sono tipicamente molto costose. Sono quindi preferibili soluzioni di tipo software.

Meccanismi ad alto livello (High-Level Mechanism)

- *Gestione sicura dei gruppi (Secure group management)*

  Operazioni di analisi e di aggregazioni dei dati sono spesso eseguite da gruppi di nodi. Conseguentemente sono necessari protocolli di sicurezza per la gestione dei gruppi, per l'ammissione sicura di nuovi membri nel gruppo e per supportare una corretta comunicazione del gruppo. L'uscita dell'elaborazione del gruppo è tipicamente trasmessa ad una stazione base. Tale uscita dovrà essere autenticata per assicurare che provenga da un gruppo valido.

- *Aggregazione sicura dei dati (Secure data aggregation)*

  I valori misurati vengono spesso aggregati per evitare di trasferire elevate quantità di informazione (ridondante) alla stazione base. Per esempio il sistema può calcolare la temperatura media di una regione geografica combinando le misure provenienti da diversi sensori o aggregare diversi valori acquisiti per evitare falsi allarmi. In relazione all'architettura della rete l'aggregazione può avvenire in diversi posti della stessa. Tutte le operazioni di aggregazione dovrebbero essere comunque rese sicure.

- *Rilevamento delle intrusioni (Intrusion detection)*

  Le reti di sensori wireless sono suscettibili a diverse forme di intrusione. Le applicazioni di rilevamento delle intrusioni fanno spesso riferimento al monitoraggio delle zone perimetrali dell'area da rendere sicura.

7.3 Classificazione dei meccanismi di sicurezza (Classification of security)

I meccanismi di sicurezza si possono suddividere in tre grandi categorie [136]: 1) ostacoli alla sicurezza della rete di sensori; 2) requisiti per una rete di sensori sicura; 3) aggressioni.

Ostacoli alla sicurezza della rete di sensori (Obstacles of Sensor Security)

1. *Risorse molto limitate (Very Limited Resources)*

   Tutti i meccanismi di sicurezza richiedono la disponibilità di una certa quantità di risorse per l'implementazione e, in particolare, spazio nella memoria dati e nella memoria di programma, energia per fare girare gli algoritmi e trasmettere i messaggi legati alla sicurezza. Tipicamente queste risorse sono molto limitate nei dispositivi sensori.

   1. *Memoria limitata (Limited Memory and Storage Space)*

      Un sensore è tipicamente un piccolo dispositivo con una limitatissima memoria sia per i dati sia per il codice. Per realizzare un efficace meccanismo di sicurezza è necessario limitare la dimensione del suo codice.

   2. *Energia limitata (Power Limitation)*

      L'energia è la risorsa più vincolante di una WSN. Tipicamente un nodo sensore non può essere facilmente sostituito o ricaricato. Quindi la carica della batteria deve essere conservata per estendere la durata di vita di ogni nodo sensore e quindi la durata di vita dell'intera rete. Aggiungendo meccanismi di sicurezza ai nodi sensore bisogna considerare l'impatto sulla durata di vita dello stesso. Il consumo di energia aggiuntivo sarà dovuto alle elaborazioni richieste per le funzioni di sicurezza (crittografia, firma digitale), alle trasmissioni dei messaggi legati alla sicurezza o degli eventuali overhead nei messaggi informativi e alla memorizzazione dei dati legati alla gestione della sicurezza.

2. *Comunicazioni inaffidabili (Unreliable Communication)*

   Le comunicazioni inaffidabili sono un altro aspetto della sicurezza. La sicurezza della rete si basa pesantemente su un protocollo definito che a sua volta dipende dalle comunicazioni.

3. *Trasferimenti inaffidabili (Unreliable Transfer)*

   Tipicamente le tecniche di instradamento dei pacchetti di una WSN sono connectionless e quindi intrinsecamente non affidabili. I pacchetti potrebbero essere danneggiati a causa di errori nel canale o scartati da nodi particolarmente congestionati. Questo comporta la perdita o il danneggiamento di alcuni pacchetti. L'elevato tasso di errore del canale obbliga gli sviluppatori dei protocolli a dedicare molte risorse alla gestione degli errori.

4. *Conflitti (Conflicts)*

   Anche se il canale fosse affidabile la comunicazione potrebbe non esserlo. A causa della natura diffusiva delle trasmissioni in una WSN conflitti nel trasferimento di pacchetti provenienti da diverse sorgenti si possono verificare e il trasferimento può fallire. In una WSN con elevata densità dei nodi questo potrebbe divenire il problema più importante.

5. *Latenza (Latency)*

   In un instradamento multi-hop la congestione della rete e l'elaborazione dei nodi può portare a un elevata latenza nella rete e rendere quindi difficile la sincronizzazione tra i diversi nodi sensore.

6. *Operazioni incustodite (Unattended Operation)*

   In relazione alle funzioni di una particolare rete di sensori i nodi possono rimanere incustoditi per lunghi periodi di tempo. Questo comporta tre diverse problematiche:

   1. *Esposizione ad attacchi materiali (Exposure to Physical Attacks)*

      Il sensore potrebbe essere posizionato in un ambiente aperto ad avversità, cattive condizioni meteorologiche e così via. La probabilità che un nodo sensore debba tollerare un attacco materiale è quindi molto più alta di quella di un tipico PC o dispositivo palmare che è spesso posizionato in un posto sicuro e deve solo fronteggiare attacchi provenienti dalla rete.

   2. *Gestione remota (Managed Remotely)*

      La gestione remota dei nodi sensori rende virtualmente impossibile rilevare manomissioni fisiche e problematiche legate alla manutenzione fisica del dispositivo. Forse l'esempio più estremo di questa problematica è quello legato all'ambito militare di un nodo sensore per missioni di ricognizione dietro le linee nemiche. In questo caso il nodo dopo essere stato posizionato potrebbe non avere più alcun contatto con le forze amiche.

   3. *Assenza di un nodo centralizzato per la gestione (No Central Management Point)*

      Una rete di sensori dovrebbe essere una rete distribuita senza un punto di gestione centralizzato. Se non progettata correttamente questo però può comportare una organizzazione difficile, inefficiente e fragile.

Requisiti per un rete di sensori sicura (Security Requirements)

I requisiti comprendono quelli di una tipica rete di comunicazione ma devono estendersi alle peculiarità di una WSN.

- *Riservatezza dei dati (Data Confidentiality)*

  La riservatezza dei dati è la problematica principale per la sicurezza di una rete. Ogni rete con un obiettivo di sicurezza deve tipicamente affrontare per primo questo problema. In una rete di sensori la riservatezza dei dati è legata ai seguenti problemi: una rete di sensori non dovrebbe far trapelare i valori letti dai sensori ad eventuali reti vicine; specialmente in applicazioni militari, i dati memorizzati nei nodi potrebbero essere particolarmente sensibili; in diverse applicazioni i nodi comunicano dati particolarmente sensibili, ad esempio distribuzioni delle chiavi quindi è estremamente importante realizzare dei canali di comunicazione sicuri; le informazioni pubbliche come l'identità dei sensori e le chiavi pubbliche dovrebbero essere crittografate per proteggersi contro gli attacchi di analisi del traffico.

- *Integrità dei dati (Data Integrity)*

  Con l'implementazione della riservatezza, un avversario potrebbe non essere in grado di rubare l'informazione. Comunque questo non significa che i dati siano sicuri. L'avversario può cambiare i dati o portare scompiglio nella rete. Ad esempio un nodo contraffatto può aggiungere alcuni frammenti o manipolare i dati di un pacchetto. Il pacchetto modificato può essere spedito al ricevitore originale. La perdita dei dati o il loro danneggiamento può avvenire perfino senza la

presenza di un nodo contraffatto a causa delle pessime condizioni dell'ambiente di comunicazione. L'integrità dei dati dovrebbe comunque assicurare che i dati ricevuti non siano stati alterati (volontariamente o involontariamente) durante il transito nella rete.

- *"Freschezza" dei dati (Data Freshness)*

  Oltre alla riservatezza e all'integrità dei dati è anche necessario che venga assicurata la "freschezza" dei messaggi. In maniera informale questo implica che i dati siano recenti e assicura che nessun vecchio messaggio venga rimesso in gioco. Questo requisito è particolarmente importante quando nel progetto sono prese in considerazione strategie a chiave condivisa. Tipicamente le chiavi condivise non necessitano di essere cambiate nel tempo. Comunque, se trascorre troppo tempo per la propagazione delle nuove chiavi condivise lungo i nodi della rete diviene semplice per un avversario utilizzare un attacco di tipo duplicazione. Inoltre è facile disturbare il normale lavoro dei sensori, se il sensore non è consapevole del nuovo cambiamento della chiave. Per risolvere questo problema possono essere aggiunti dei contatori ai pacchetti per assicurare la "freschezza" dei dati.

- *Disponibilità (Availability)*

  Alcuni approcci scelgono di modificare il codice per utilizzare un numero più elevato di codici. Per ottenere lo stesso obiettivo altri approcci utilizzano comunicazioni aggiuntive. Alcuni approcci impongono limitazioni rigorose all'accesso dei dati o propongono schemi non disponibili (come schemi centralizzati) per semplificare l'algoritmo. Tutte queste tecniche indeboliscono la disponibilità di un sensore e della rete per i seguenti motivi: l'elaborazione aggiuntiva consuma più energia, quando le batterie si esauriscono i dati non saranno più disponibili. Anche le comunicazioni consumano più energia. Inoltre, l'incremento dei messaggi trasmessi incrementa il numero dei possibili conflitti. Utilizzando uno schema centralizzato si introdurrebbe un singolo punto di vulnerabilità e questo minaccerebbe enormemente la disponibilità della rete.

- *Auto-Organizzazione (Self-Organization)*

  Una rete di sensori è tipicamente una rete ad hoc che richiede che ogni sensore sia indipendente e flessibile al punto di auto-organizzarsi e auto-ripararsi in relazione a diverse situazioni.

- *Sincronizzazione temporale (Time Synchronization)*

  La maggior parte delle applicazioni per reti di sensori confidano in alcune forme di sincronizzazione temporale. Con lo scopo di diminuire il consumo di energia, la radio di ogni sensore può essere spenta periodicamente. Inoltre i sensori possono desiderare di calcolare il ritardo end-to-end di un pacchetto. Una rete di sensori "collaborativa" può richiedere la sincronizzazione di un gruppo di sensori per applicazioni di tracciamento.

- *Localizzazione sicura (Secure Localization)*

  Spesso l'utilità di una rete di sensori fa affidamento nella sua capacità di localizzare accuratamente e automaticamente ogni sensore della rete. Una rete di sensori progettata per localizzare guasti necessiterà di accurate informazioni di localizzazione per indicare con esattezza la posizione del guasto. Sfortunatamente un aggressore può facilmente manipolare le informazioni di localizzazione non sicura inviando falsi segnali di guasti o inviando falsi segnali di risposta.

- *Autenticazione (Authentication)*

  Un avversario non si limita a modificare i pacchetti dati. Può cambiare l'intero flusso informativo iniettando pacchetti aggiuntivi. Il ricevitore deve essere in grado di verificare che i dati utilizzati in qualunque processo di decisione siano originati dalla sorgente corretta. D'altro canto quando si costruisce la rete di sensori, l'autenticazione è necessaria per diversi compiti amministrativi, l'autenticazione dei dati permette al ricevitore di verificare che i dati siano realmente spediti da chi si sta proclamando come mittente. Nel caso di una comunicazione fra due parti, l'autenticazione può essere ottenuta attraverso dei meccanismi puramente simmetrici: il mittente e il ricevitore

condividono una chiave segreta per elaborare il codice di autenticazione del messaggio (MAC) di tutti i dati comunicati.

Aggressioni (Attacks)

Le WSN sono particolarmente vulnerabili a diversi tipi di aggressioni. Tale vulnerabilità è dovuta essenzialmente alla natura diffusiva del mezzo trasmissivo. Le aggressioni verso le WSN possono essere considerate da due diversi punti di vista: quelle contro i meccanismi di sicurezza e quelle contro i meccanismi di base di funzionamento della rete (come il meccanismo di instradamento).

- *Aggressioni passive (Passive Attacks)*

  Il monitoraggio e l'ascolto del canale di comunicazione da aggressori non autorizzati è noto come aggressione passiva.

  - *Aggressioni verso la riservatezza (Attacks against Privacy)*

    Il principale problema è legato al fatto che le WSN intensificano il problema della riservatezza in quanto rendono facilmente disponibile, attraverso accesso remoto, una enorme quantità di informazione. Alcune delle aggressioni più comuni verso la riservatezza sono:

    - *Monitoraggio e intercettazione (Monitor and Eavesdropping):*

      Si tratta del più comune attacco alla riservatezza. Curiosando nei dati l'avversario potrebbe facilmente scoprire i contenuti della comunicazione. Quando il traffico trasporta informazioni di controllo sulla configurazione della rete di sensori che potenzialmente contengono informazioni dettagliate rispetto a quelle accessibili in un server di localizzazione, l'intercettazione può operare effettivamente contro la violazione della riservatezza.

    - *Analisi del traffico (Traffic Analysis):*

      Anche se i messaggi sono trasferiti in maniera crittografata è ancora altamente possibile un analisi dei pattern di comunicazione. Le attività dei sensori possono potenzialmente rilevare perfino informazioni che permettono all'avversario di causare danni alla rete di sensori.

    - *Avversari camuffati (Camouflage Adversaries):*

      Si può inserire un proprio nodo o compromettere i nodi nascondendosi nella rete di sensori. All'inizio questi nodi si comportano come nodi normali della rete per attirare la trasmissione di pacchetti, quindi analizzano tali pacchetti per individuare informazioni riservate.

- *Aggressioni attive (Active Attacks)*

  In questo caso gli aggressori non autorizzati monitorano, ascoltano e modificano il flusso di dati nel canale di comunicazione.

  - *Aggressioni verso l'instradamento (Routing Attacks in Sensor Networks)*

    Le aggressioni che operano a livello di rete sono note come aggressioni verso l'instradamento.

    - *Informazioni di instradamento false, alterate e duplicate (Spoofed, altered and replayed routing information)*

      Un instradamento ad hoc non protetto è vulnerabile a questi tipi di aggressioni, dato che ogni nodo opera da router e può quindi direttamente instradare l'informazione. Alcune modalità di aggressione prevedono di creare percorsi di instradamento ad anello,

l'allungamento o l'accorciamento dei percorsi, la generazione di falsi messaggi di errore, l'incremento della latenza end-to-end.

- *Inoltro selettivo (Selective Forwarding)*

  Un nodo contraffatto può scartare selettivamente solo un certo tipo di pacchetti. Particolarmente efficace se combinato con un'aggressione che raccoglie il traffico attraverso il nodo. Nelle reti di sensori generalmente si assume che i nodi inoltrino fedelmente i messaggi ricevuti. Alcuni nodi compromessi si potrebbero rifiutare di inoltrare i pacchetti, comunque i nodi vicini potrebbero cominciare ad utilizzare altre rotte.

- *Aggressione della Sibilla (Sybil Attack)*

  In molti casi i sensori di una WSN potrebbero avere la necessità di lavorare insieme per svolgere un determinato compito, quindi possono utilizzare distribuzione di sottocompiti e ridondanza di informazione. In questa situazione un nodo può dichiarare di non essere un solo nodo utilizzando le identità legittime degli altri nodi. Questo tipo di aggressione dove un nodo falsifica l'identità di più di un nodo è chiamata "aggressione della Sibilla". La Sibilla cerca di degradare l'integrità dei dati, la sicurezza e l'utilizzazione delle risorse che l'algoritmo distribuito cerca di ottenere. L'aggressione della Sibilla può essere effettuata per aggredire un sistema di memorizzazione distribuito, un meccanismo di instradamento, di aggregazione dei dati, di elezione, un'equa allocazione delle risorse e di rilevamento di un malfunzionamento.

- *Aggressione del buco nero o della dolina (Black hole/Sinkhole Attack)*

  In questa aggressione un nodo contraffatto agisce come un buco nero attraendo tutto il traffico della rete di sensori. Specialmente se si utilizza un protocollo basato sul flooding, l'aggressore ascolta le richieste per le rotte e risponde ai nodi indicando che lui contiene il percorso migliore o più breve verso la stazione base. Una volta che il nodo contraffatto è stato in grado di inserirsi fra i nodi di comunicazione, sarà in grado di fare qualunque cosa con i pacchetti che lo attraversano. Questo attacco può anche influenzare nodi che sono considerevolmente lontani dalla stazione base.

- *Aggressione ad inondazione di pacchetti HELLO (HELLO Flood Attack)*

  I pacchetti HELLO sono utilizzati come un arma per imbrogliare i sensori di una WSN. Un aggressore con un terminale radio ad elevata potenza (ad esempio un notebook) e una elevata capacità di elaborazione invia pacchetti HELLO a un numero di nodi sensore che sono distribuiti in un'ampia area all'interno della WSN. I sensori, convinti che l'avversario sia un proprio vicino, per spedire le informazioni alla stazione base cercano di utilizzare il nodo aggressore come router e quindi vengono imbrogliati.

- *Aggressione del buco del verme*

  Il nodo aggressore cattura i pacchetti (o i bit) in una posizione della rete e li inoltra in un'altra posizione. La ritrasmissione dei bit può essere fatta in maniera selettiva. Questo tipo di attacco è una minaccia significativa per una WSN perché non richiede che sia compromesso alcun sensore e potrebbe essere attuata perfino nella fase iniziale della vita della rete quando i sensori cominciano a scoprire le informazioni sui propri vicini. Quando un nodo B (per esempio la base station o ogni altro sensore) diffonde il pacchetto per la richiesta delle rotte, l'aggressore riceve il pacchetto e lo replica nelle sue vicinanze. Ogni nodo che riceve questo pacchetto considererà il nodo B come suo vicino e lo marcherà come suo genitore. Quindi perfino se i nodi vittime sono distanti diversi hop dal nodo B, l'aggressore in questo caso li convince che B è solo ad un hop di distanza da loro e quindi crea il cosiddetto buco del verme.

- *Negazione del servizio (Denial of Service - DoS)*

  La più semplice aggressione di DoS cerca di esaurire le risorse disponibili nei nodi vittime spedendo pacchetti non necessari e quindi impedendo agli utenti legittimi della rete di accedere ai servizi o alle risorse. Per aggressione DoS non si intende solo il tentativo dell'avversario di sovvertire, interrompere o distruggere una rete, ma anche ogni evento che diminuisce le capacità della rete di fornire un servizio. Nelle WSN diversi tipi di attacchi DoS in diversi strati possono essere eseguiti. I meccanismi per prevenire tali attacchi comportano l'incremento delle risorse di rete per permettere il respingimento, meccanismi robusti di autenticazione e di identificazione del traffico.

- *Sovversione di un nodo (Node Subversion)*

  La cattura di un nodo può rilevare le sue informazioni fra le quali le chiavi di crittografia e quindi compromettere il funzionamento dell'intera rete.

- *Malfunzionamento di un nodo (Node Malfunction)*

  Un nodo mal funzionante può generare dati sbagliati che potrebbero esporre l'integrità della rete di sensori specialmente se si tratta di un nodo di aggregazione o di un nodo leader di n cluster.

- *Interruzione del funzionamento di un nodo (Node Outage)*

  Se si tratta di un nodo leader di un cluster, il protocollo della WSN dovrebbe essere abbastanza robusto e mitigare l'effetto dell'interruzione fornendo rotte alternative.

- *Attacchi fisici (Physical Attacks)*

  Le reti di sensori operano tipicamente in ambienti esterni e ostili. In questi ambienti le piccole dimensioni dei sensori insieme alla natura distribuita e non sorvegliata del loro posizionamento li rende altamente suscettibili ad aggressioni di natura fisica, cioè legati alla distruzione fisica del nodo. Differentemente da molte delle altre aggressioni già elencate, quelle fisiche distruggono permanentemente il nodo e quindi le perdite sono irreversibili. Ad esempio, gli aggressori possono estrarre le chiavi di crittografia, manomettere i circuiti elettronici, modificare il firmware del sensore o introdurre del firmware contraffatto sotto il loro controllo.

- *Corruzione dei messaggi (Message Corruption)*

  Qualunque modifica al contenuto dei messaggi da un aggressore ne compromette l'integrità.

- *Falso nodo (False Node)*

  Un nodo falso comporta l'aggiunta di un nodo alla rete da parte di un aggressore e provoca l'inserimento di dati contraffatti. L'intruso potrebbe aggiungere un nodo al sistema che inserisce dati falsi o si oppone al passaggio dei dati corretti. L'inserimento in una WSN di un nodo contraffatto è una delle aggressioni più pericolose che si possa verificare. Codici contraffatti inseriti nella rete potrebbero diffondersi a tutti i nodi, potenzialmente distruggere l'intera rete o, ancora peggio, consegnare l'intera rete nelle mani dell'avversario.

- *Aggressioni con duplicazione di un nodo (Node Replication Attacks)*

  Concettualmente, l'aggressione con la duplicazione di un nodo è abbastanza semplice; un aggressore cerca di aggungere un nodo a una rete di sensori copiando l'ID di un nodo presente nella rete. Un nodo duplicato in questo modo può compromettere gravemente le prestazioni della rete di sensori. I pacchetti possono essere corrotti o perfino instradati in maniera errata. Questo può comportare la disconnessione di parte della rete, false letture dei sensori, ecc. Se un aggressore può guadagnare l'accesso materiale all'intera rete potrà anche copiare le chiavi di crittografia e utilizzarle nei nodi sensori duplicati. Inserendo i nodi duplicati in specifici punti della rete, l'aggressore potrebbe facilmente manipolare uno specifico segmento della rete attraverso la disconnessione di tale segmento dall'intera rete.

- *Raccolta passiva delle informazioni (Passive Information Gathering)*

  Un avversario con risorse elevate può collezionare informazioni dalla rete di sensori se queste non sono crittografate. Un intruso con un potente ricevitore e una ben progettata antenna può facilmente prelevare il flusso di dati. L'intercettazione di messaggi contenenti la posizione fisica dei nodi sensore permette a un aggressore di localizzarli e poi distruggerli. Oltre alla posizione dei sensori, un avversario può osservare il contenuto dei messaggi e identificare gli ID, i marcatori temporali e altri specifici campi. Per minimizzare la minaccia della raccolta di informazioni in maniera passiva è necessario utilizzare robuste tecniche di crittografia.

- *Aggressione sulle informazioni in transito (Attacks on Information in transit)*

  In una rete di sensori i nodi monitorano i cambiamenti di specifici parametri o valori e li riportano al sink in funzione dei requisiti. Mentre viene trasferita, l'informazione in transito può essere alterata, contraffatta, duplicata o cancellata. Dato che le comunicazioni wireless sono vulnerabili alle intercettazioni, qualunque aggressore può monitorare il flusso di traffico e agire per interromperlo, intercettarlo, modificarlo o costruire pacchetti che forniscono informazioni errate alla stazione base o al sink.

- *Inondazione di informazioni (Information Flooding)*

  Un meccanismo di instradamento casuale o un meccanismo di generazione di traffico fantasma possono essere utilizzati per mascherare il traffico reale in modo da rendere complicato a un avversario la tracciabilità della sorgente analizzando il traffico in rete.

  - *Implementazione di base del meccanismo di inondazione (Baseline Flooding)*

    Nell'implementazione di base ogni nodo della rete inoltra un messaggio alla volta e nessun nodo ritrasmette un messaggio che è stato precedentemente trasmesso. Quando un messaggio raggiunge un nodo intermedio, il nodo verifica se quel messaggio è stato già ricevuto e inoltrato. Se è la prima volta che viene ricevuto allora il nodo diffonderà il messaggio a tutti i suoi nodi vicini. Altrimenti il messaggio verrà semplicemente scartato.

  - *Inondazione probabilistica (Probabilistic Flooding)*

    Solo un insieme di nodi della rete parteciperanno all'inoltro dei dati mentre gli altri si limiteranno a scartarli. Un possibile punto debole di questo approccio è che alcuni messaggi potrebbero perdersi nella rete e quindi si compromette la connettività dell'intera rete.

  - *Inondazione con messaggi contraffatti (Flooding with Fake Messages)*

    Le precedenti strategie di inondazione possono diminuire le possibilità di violazione della riservatezza. Questa osservazione suggerisce che un approccio per diminuire il rischio di perdita della riservatezza della posizione delle sorgenti consiste nell'aumentare il traffico dei protocolli di inondazione introducendo più sorgenti che inseriscano messaggi contraffatti nella rete. Facendo così l'aggressore non avrà idea di quali pacchetti siano veri e quali contraffatti.

  - *Inondazione fantasma (Phantom Flooding)*

    L'inondazione fantasma condivide lo stesso principio di quella probabilistica nel senso che entrambi gli approcci cercano di inviare messaggi verso diverse posizioni della rete così che l'avversario non può ricevere un flusso continuo di messaggi per localizzare le sorgenti. L'inondazione probabilistica è poco efficiente nell'ottenere questo obiettivo perché i percorsi più brevi sono più probabili per trasferire i messaggi. Un possibile tentativo è quello di confondere l'aggressore nascondendo la sorgente vera a favore di una sorgente contraffatta chiamata sorgente fantasma. Nell'inondazione fantasma ogni messaggio sperimenta due fasi: 1) una fase di cammino che può essere casuale o guidato e 2) una seguente fase di inondazione destinata a consegnare il messaggio al sink. Quando la

sorgente spedisce un messaggio questo sarà di tipo unidirezionale in modalità casuale per i primi “hwalk” passi (fase della camminata casuale). Dopo gli “hwalk” passi il messaggio è inondato utilizzando la tecnica di inondazione di base (fase dell’inondazione).

7.4 Gestione della affidabilità (Trust management)

Gli schemi di gestione della affidabilità possono essere classificati in tre categorie [136]: centralizzati, distribuiti e ibridi.

Gli schemi **centralizzati** di gestione della affidabilità **(Centralized trust management – CTM)** hanno un singolo server globale affidabile che determina i livelli di affidabilità di ogni nodo della rete. Questo permette di avere un più basso carico computazionale ai nodi sensori perché la maggior parte delle elaborazioni sono effettuate dal server centralizzato che non ha vincoli di potenza di calcolo e memoria. Questo approccio comunque ha lo svantaggio di avere un singolo punto di vulnerabilità che lo rende poco affidabile. Inoltre non permette ai diversi nodi di avere diversi valori di affidabilità rispetto a un determinato nodo. Per le reti di sensori di elevata estensione lo schema centralizzato non è applicabile a causa dell’elevato costo legato allo scambio dei valori di affidabilità di ogni nodo sensore periferico con la stazione base.

Anche gli schemi **distribuiti (Distributed trust management - DTM)** non lavorano bene nelle WSN di elevata estensione. Questi schemi prevedono infatti che ogni nodo calcoli il livello di affidabilità verso tutti gli altri nodi della rete e quindi si incrementa il carico computazionale dei singoli nodi, lo spazio di memoria occupata perché ogni nodo deve mantenere tali valori per l’intera rete all’interno di una tabella e la dimensione della tabella sarà direttamente proporzionale a quella della rete. Ogni nodo però è a conoscenza del proprio livello di affidabilità verso tutti gli altri nodi e quindi si elimina il traffico necessario per lo scambio di tali informazioni verso il server dello schema centralizzato.

L’approccio distribuito è più affidabile di quello centralizzato perché non ha un singolo punto di vulnerabilità. Un’ulteriore possibilità è quella di utilizzare uno schema distribuito ristretto i cui i nodi sensori mantengono i livelli di affidabilità solo per i nodi vicini **(localized DTM)**. Questo approccio ha però il problema che si introducono ritardi e dipendenze nei calcoli dei livelli di affidabilità per i nodi lontani.

Gli schemi **ibridi (Hybrid trust management – HTM)** hanno le proprietà sia di quelli centralizzati sia di quelli distribuiti. L’obiettivo è ridurre i costi associati con la valutazione dei livelli di affidabilità rispetto a un approccio puramente distribuito. In questi schemi si utilizza una suddivisione della rete in cluster nelle quale un nodo “capo” si comporta da server centralizzato per l’intero cluster. Rispetto allo schema distribuito si introduce una maggior quantità di traffico di rete.

I precedenti schemi di gestione della affidabilità possono essere classificati anche in altre due categorie: gestione della affidabilità basata su certificati e gestione della affidabilità basata sui comportamenti.

Nella **gestione della affidabilità basata su certificati (certificate-based trust management)** la affidabilità si basa sulla fornitura di un valido certificato assegnato ad ogni nodo da un’autorità di certificazione centralizzata od un altro emittente di affidabilità.

Nell’approccio della **gestione della affidabilità basata sui comportamenti (behavior-based trust management)** un’entità clacola i livelli di affidabilità attraverso il monitoraggio diretto o indiretto degli altri nodi.

7.5 Gestione della sicurezza nelle reti 6LowPAN

Le WSN per loro natura sono aperte ad aggressioni che possono origliare e inserire pacchetti che vengono scambiati, possibilmente utilizzando tecnologie con antenne avanzate che permettono di effettuare le aggressioni al di fuori del range dei dispositivi IEEE802.15.4. È necessario quindi prendere in considerazioni gli aspetti legati alla sicurezza e alla riservatezza in modo appropriato.

Si dovrebbero ottenere i seguenti obiettivi [137]:

- **Riservatezza:** non dovrebbe essere possibile origliare i dati da parte di ascoltatori non autorizzati, cioè i dati dovrebbero rimanere segreti al di fuori dei dispositivi autorizzati a partecipare alla comunicazione. È possibile ottenere la riservatezza rendendo l'informazione non intelligibile attraverso la cifratura crittografica.
- **Integrità:** non dovrebbe essere possibile a terzi non autorizzati di alterare i dati cioè i dati che saranno ricevuti da un dispositivo autorizzato devono essere identici a quelli trasmessi dalla controparte autorizzata. Nel mondo digitale, la manomissione con un messaggio può non lasciare alcuna traccia rilevabile, così l'integrità è spesso attenuta aggiungendo controlli di integrità crittografici ai messaggi. Un check di integrità di un messaggio consiste spesso in un controllo di autenticazione cioè nella verifica che il messaggio sia stato effettivamente generato dalla sorgente che si sta dichiarando.
- **Disponibilità:** i dati dovrebbero essere disponibili ai dispositivi in modo da rispettare i tempi nei quali gli servono. In altre parole il sistema non deve essere soggetto ad attacchi di negazione del servizio.

Una volta che gli obiettivi di sicurezza sono stati definiti in funzione delle specifiche richieste dall'applicazione è necessario capire i modelli delle possibili minacce: 1) cosa un aggressore sarà in grado di fare per poter lavorare contro gli obiettivi di sicurezza; 2) il beneficio che un aggressore può ricavare rompendo le barriere di sicurezza predisposte.

Per i sistemi wireless come 6LoWPAN il modello delle minaccie che può essere adottato è quello legato alla sicurezza di Internet: si assume che l'aggressore ha praticamente il controllo completo del canale di comunicazione (può leggere qualunque messaggio, rimuovere o cambiare messaggi, inserire nuovi messaggi). Nel caso di Internet si assume che i sistemi finali non siano compromessi principalmente perché sarebbe difficile mantenere una sicurezza completa se questa assunzione non viene fatta. Nel caso delle LoWPAN la piccola dimensione e la natura distribuita dei nodi aumenta significativamente la minaccia in quanto risulta relativamente facile ottenere fisicamente il controllo di almeno un nodo della rete: i requisiti legati ai bassi costi dei nodi limitano le possibilità di rendere i nodi resistenti alle manomissioni. Risulta di fondamentale importanza che la protezione dell'intera rete non dipenda dall'integrità di ogni singolo nodo della rete. Il problema può essere di difficile soluzione ed è necessario trovare il giusto compromesso fra i possibili danni e i costi per fornire e mantenere la sicurezza.

La gestione della sicurezza nelle reti IPv6

La sicurezza può essere gestita attraverso i protocolli TLS (Transport Layer Security) o DTLS (Datagram TLS). IPSecurity (IPSec) può essere utilizzato a livello di rete ma utilizza molte risorse. I vincoli nelle reti 6LoWPAN impediscono l'uso dell'intera suite protocollare IPSec. In questo tipo di reti è necessario gestire le chiavi di crittografia utilizzando il minimo payload e limitare i messaggi scambiati fra i nodi. LSEND (Lightweight Secure Neighbour Discovery) è una estensione del protocollo SEND che permette di rendere sicuro il meccanismo di scoperta dei nodi vicini implementabile in 6LoWPAN. L'Advanced Encryption Standard (AES) permette di abilitare la sicurezza a livello data-link. Gli strati di rete 6LoWPAN che sfruttano la crittografia AES a 128 bit dello standard IEEE 802.15.4 forniscono una certa protezione. L'autenticazione può essere fatta utilizzando il modello dell'anatroccolo che risuscita (resurrecting duckling model). Pesanti protocolli di sicurezza devono essere evitati il più possibile a causa della intrinseca carenza di energia e di risorse di calcolo dei nodi di una 6LoWPAN. La crittografia di tipo simmetrico è preferita. L'implementazioni di meccanismi di sicurezza a livello di trasporto ("socket") o l'uso di sofisticati firewall in ogni nodo non sono possibili. Comunque il protocollo DTLS può essere usato e fornisce un adeguato livello di sicurezza end-to-end fra i client web e i server web. La crittografia simmetrica AES può essere utilizzata per la riservatezza e l'integrità dei dati. AES/CCM (counter with cipher block chaining message authentication code CBC-MAC) è particolarmente efficiente e un algoritmo molto sicuro purché la stessa "nonce" non si verifichi due volte

con la stessa chiave. Per questo motivo molte applicazioni richiedono che le chiavi vengano rigenerate più volte durante la vita di una rete 6LoWPAN.

La gestione della sicurezza nello strato data-link di IEEE802.15.4

Lo standard IEEE802.15.4 fornisce servizi per la gestione della sicurezza a livello data-link. Si possono utilizzare tre modalità operative: 1) modalità non sicura; 2) modalità con Access Control List (ACL); 3) modalità sicura.

Nella **modalità non sicura (unsecured mode)**, come dice lo stesso nome, non è fornito alcun servizio di sicurezza.

Nella **modalità ACL (ACL mode)** i dispositivi mantengono una lista degli altri dispositivi con i quali possono comunicare. Ogni comunicazione proveniente da un dispositivo non presente nella lista viene ignorata. Comunque è da notare che questa modalità non offre meccanismi di crittografia e quindi è molto semplice clonare l'indirizzo del nodo sorgente del messaggio.

La **modalità sicura (secured mode)** è progettata per usare facilmente chiavi di crittografia simmetriche con lo scopo di fornire riservatezza e autenticità dei dati e protezione. È possibile utilizzare una specifica chiave per ogni coppia di dispositivi (link key) o una chiave comune per un gruppo di dispositivi. Comunque i meccanismi utilizzati per sincronizzare e scambiare le chiavi non sono definiti nello standard e lasciati alle applicazioni.

Il grado di protezione può essere modificato per ogni frame. Inoltre i frame sicuri possono essere instradati da dispositivi che non supportano la sicurezza.

Per proteggere la riservatezza e/o l'autenticità dei messaggi lo standard IEEE802.15.4-2011 utilizza la modalità di cifratura **counter with cipher block chaining mode* (CCM*)**, una estensione della modalità CCM definita dallo standard ANSI X9.63.2001. La modalità CCM* è compatibile con la modalità CCM e allo stesso tempo permette di superarne alcune limitazioni. La modalità CCM opera con cifrari a blocchi di 128 bit e include una particolare combinazione del modo di operare della modalità a contatore **(Counter (CTR) mode)** e del modo di operare della modalità dei cifrari a blocchi concatenati **(Cipher-Block Chaining (CBC))**, utilizzando una singola chiave. La modalità CCM permette tag di autenticazione di lunghezza variabile (da 32 a 128 bit) quindi permette di variare il grado di protezione contro modifiche non autorizzate. La modalità CCM permette inoltre una efficiente implementazione per il fatto che necessita solo dell'implementazione di una trasformazione di cifratura del sottostante cifrario a blocchi (e non necessita di una trasformazione di decifratura) e per il fatto che utilizza una chiave singola piuttosto che chiavi multiple. Fra gli svantaggi della CCM ricordiamo:

- non permette di occuparsi esclusivamente della riservatezza (non tutte le implementazioni richiedono l'autenticazione dei dati ad esempio perché fornita da meccanismi esterni)
- è vulnerabile ad alcune aggressioni specifiche se viene utilizzata con tag di autenticazione a lunghezza variabile.

La modalità CCM* estende le funzionalità della modalità CCM fornendo la possibilità di implementare esclusivamente la riservatezza e permettendo l'uso sicuro di tag di autenticazione di lunghezza variabile. Allo stesso tempo le specifiche della modalità CCM* sono compatibili con quelle della modalità CCM.

CCM* prende come ingresso una stringa "a" che deve essere autenticata utilizzando un codice hash e una stringa "m" che deve essere crittografata e fornisce in uscita un testo cifrato comprentende sia la versione crittografata di "m" sia codice CBC di autenticazione del messaggio (CBC MAC) di "a". Nello standard IEEE802.15.4 il "nonce" è un campo di 13 byte: 8 byte contenenti l'indirizzo del dispositivo che ha originato il frame, 4 byte relativi al contatore di frame e 1 byte è il codice del livello di sicurezza.

I parametri necessari per la sicurezza saranno contenuti nel campo "auxiliary control" dell'header. Questo è composto da un campo di controllo (1 byte), dal contatore di frame (4 byte) che permette la protezione contro gli attacchi di tipo duplicazione e un campo di identificazione della chiave (di 0,1,5 o 9 byte). I primi 3 bit del campo di controllo della sicurezza indicheranno la modalità di sicurezza per il frame, tale modalità

determinerà la dimensione di M nell'algoritmo CCM* (0,4,8 o 16 byte) e i campi dati inclusi nelle stringhe "a" e "n" utilizzate per il calcolo del testo finale cifrato (attributi di sicurezza). I successivi 2 bit indicano la modalità di identificazione della chiave e i rimanenti bit sono riservati. Le seguenti tabelle illustrano il significato rispettivamente dei primi 3 bit e dei successivi 2 bit del campo di controllo della sicurezza.

Security control field	Security attributes	Data confidentiality (data in "m" string)	Data authenticity (data in "a" string)
000	None	OFF	NO
001	Mic-32		MHR, Auxiliary security header, non-payload fields, Unsecured payload fields
010	Mic-64		
011	Mic-128		
100	Encrypted fields	Unsecured payload fields	NO
101	Enc. fields + Mic-32		MHR, Auxiliary security header, non-payload fields
110	Enc. fields + Mic-64		
111	Enc. fields + Mic-128		

Key identifier mode	Description	Key identifier field length
00	Implicitly from the originator and the recipient of the frame	0
01	from the 1-byte Key-index subfield of the Key identifier field, using the MAC layer default Key source	1
10	Explicitly from the 4-bytes Key source subfield, and the 1-byte Key index subfield of the Key identifier field (part of the auxiliary security header)	5
11	Explicitly from the 8-bytes Key source subfield, and the 1-byte Key index subfield of the Key identifier field (part of the auxiliary security header)	9

Si noti che il protocollo IEEE802.15.4 è molto vincolante: non fornisce meccanismi per la frammentazione e il riassemblaggio, di conseguenza dato che la dimensione massima del pacchetto è di 127 byte e l'header MAC e il campo FCS occupano rispettivamente 6 e 19 byte, le applicazioni devono fare attenzione all'invio di pacchetti non sicuri più grandi di 108 byte. Le applicazioni che implementano meccanismi di sicurezza introducono header di sicurezza costituiti da un numero di byte compreso tra 7 e 15 e l'autenticazione dei messaggi richiede un numero di byte compreso tra 0 e 16. Nel peggiore dei casi solo 77 byte rimangono per l'applicazione.

IEEE802.15.4 non gestisce la distribuzione delle chiavi. L'interfaccia fra lo strato MAC e la memorizzazione della chiave è una funzione di lookup della chiave che fornusce un parametro stringa di lookup che è utilizzato come indice per recuperare la chiave appropriata.

Con l'identificazione implicita della chiave (KeyIdMode = "00") l'informazione di lookup è basata sull'indirizzo IEEE802.15.4. La progettazione implica che il trasmettitore indicizza la sua chiava in riferimento alle destinazioni e il ricevitore indicizza le sue chiavi in riferimento alle sorgenti. Con l'identificazione esplicita delle chiavi, il dato di lookup è costituito da un identificatore della sorgente della chiave e un indice di chiave. La progettazione implica che la memorizzazione della chiava sia organizzata in diversi gruppi chiamati sorgenti (uno dei quali è il macDefaultKeySource). Ogni sorgente di chiave comprende diverse chiavi identificate da un indice. Lo standard CCM specifica che una data chiave non può essere utilizzata per cifrare più di 261 blocchi, quindi le applicazioni che utilizzano lo standard IEEE802.15.4 non dovrebbero solo assegnare le chiavi ma anche cambiarle periodicamente.

Ci sono alcuni problemi con le modalità di sicurezza dello standard IEEE802.15.4 e con la fattibilità di supportare diversi modelli di distribuzione delle chiavi ma queste limitazioni possono essere superate utilizzando i livelli più alti dello stack. Questi ultimi dovrebbero però garantire dei modelli di distribuzione delle chiavi che non consumino molta energia. Un'altro problema deriva dal fatto che lo standard non propone uno schema esplicito per la gestione delle chiavi. Infine un'importante vulnerabilità è legata al fatto che i frame di ACK non hanno meccanismi di sicurezza e questo rende molte misure di sicurezza implementate a livello MAC inutili. In particolare, dato che l'integrità dei frame di ACK non è protetta si apre la porta a nodi contraffatti che possono impedire a un dispositivo autorizzato di ricevere un particolare frame, e questo è possibile falsificando un ACK utilizzando il non cifrato numero di sequenza del frame dati e spedendolo alla sorgente mentre si genera la necessaria interferenza per non fare arrivare il frame dati al destinatario. Sono quindi possibili delle aggressioni DoS attraverso la contraffazione degli ACK che potrebbero essere prevenute rendendo gli ACK sicuri.

La gestione della sicurezza nello strato di rete di 6LoWPAN

Perfino se venisse utilizzato il miglior meccanismo di gestione della sicurezza a livello data-link, le informazioni non sarebbero più protette non appena lasciano il singolo collegamento. Questo rende le informazioni vulnerabili in ogni punto che è responsabile dell'inoltro al livello di rete o su ogni collegamento che ha meno stringenti requisiti di sicurezza. Inoltre, un aggressore posizionato al livello di rete potrebbe essere in grado di direzionare i pacchetti su percorsi che contengono ulteriori nodi di instradamento da lui controllati.

Quindi l'implementazione di meccanismi di sicurezza end-to-end che proteggono le comunicazioni lungo l'intero percorso fra i nodi che stanno comunicando è un elemento fondamentale per ciascun sistema di sicurezza robusto.

IPSec è una piattaforma a livello di rete che fornisce riservatezza e integrità dei dati al livello IP. Originariamente sviluppato per IPv6 è stato applicato anche per operare con IPv4.

I due componenti principali sono i formati del pacchetto e le relative specifiche che definiscono i meccanismi di integrità e riservatezza per i dati e uno schema di gestione della chiave chiamato Internet Key Exchange (IKE) [138], di cui oggi ne esiste una seconda versione (IKEv2). IPSec/IKE può essere implementato in una modalità di trasporto host-to-host o in modalità tunnel nella rete e supporta una varietà di algoritmi di crittografia (HMAC-SHA1, tripleDES-CBC, and AES-CBC).

Anche se IPv6 raccomando l'uso di IPSec, la possibilità di utilizzarlo su dispositivi a bassa potenza è oggetto di dibattito principalmente a causa dell'incremento dei requisiti di potenza e di memoria imposti dall'implementazione degli algoritmi di crittografia. A causa del relativamente complesso insieme di protocolli che costituisce IKE, IPSec è generalmente considerato scarsamente adattabile ai requisiti di 6LoWPAN. In ogni caso IPSec può essere implementato con successo su dispositivi intelligenti se gli algoritmi e le dimensioni della chiave sono scelti con attenzione.

Definisce due formati di pacchetti per proteggere i dati crittografandoli: l'IP **Authentication Header (AH)** [139] che fornisce esclusivamente protezione verso l'integrità dei dati e l'autenticazione e l'IP **Encapsulating Security Payload (ESP)** che combina queste funzionalità con quelle relative alla protezione della riservatezza attraverso la codifica.

A differenza delle altre estensioni dell'header di IPv6 che semplicemente precedono il loro payload, ESP lo ingloba. Una parte del pacchetto sarà crittografata (protezione della riservatezza) mentre la parte più rilevante è sottoposta alla protezione dell'integrità. Le chiavi e altre informazioni, come gli algoritmi di crittografia, sono memorizzati nei nodi. Il pacchetto in dettaglio, illustrato nella figura seguente, sarà costituito da un campo *security parameter index (SPI)*, schematizzato in Figura 26, che identifica i parametri specifici per la sicurezza come il materiale per codificare utilizzato per questo verso della comunicazione.

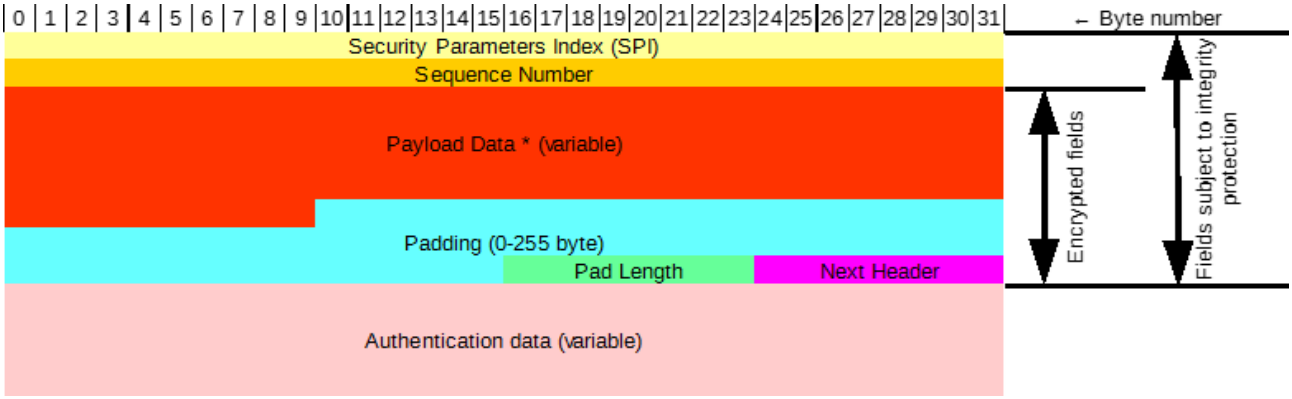


Figura 26 Formato del pacchetto SPI.

Uno specifico insieme di chiavi chiamato *Security Association (SA)*. Ogni pacchetto apparterrà a uno specifico SA. Diversi SA possono usare diversi algoritmi di crittografia. Per inizializzare un SA IPSec usa il protocollo IKE. Per i pacchetti unidirezionali il campo SPI è un identificatore con significato locale per il ricevitore, cioè è assegnato da ricevitore in maniera tale che faciliti l'elaborazione locale dei pacchetti entranti ESP.

IPsec fa uso sia di meccanismi di cifratura simmetrici sia di meccanismi di cifratura asimmetrici. Per dispositivi di bassa potenza i meccanismi simmetrici sono da preferire perché hanno una più bassa complessità computazionale. Nel dettaglio questi dispositivi sono spesso equipaggiati con hardware in grado di effettuare i calcoli dell'algoritmo AES in maniera veloce e quindi si può fare in modo che IPSec si avvantaggi del supporto di tale hardware. Utilizzando l'accelerazione hardware di AES come parte di IPSec SA, il dispositivo ottiene un livello di prestazione nella cifratura che è impossibile avere via software. Il *sequence number* è un numero senza segno a 32 bit che si incrementa di uno per ogni pacchetto spedito sulla connessione sicura; può anche essere la parte meno significativa di una sequenza a 64 bit mantenuta nella connessione sicura. Il campo *payload data* insieme con i campi che seguono *padding, pad length* e *next header* costituiscono la parte del pacchetto che è crittografata.

Il campo *payload data* specifica come i dati decifrati devono essere interpretati dal ricevitore. Permette di identificare se il payload è un pacchetto IP (si sta operando in modalità tunnel) o se si tratta di un pacchetto UDP (si tratta quindi di un header UDP seguito dal relativo payload e si sta operando in modalità trasporto). La modalità tunnel è particolarmente utile per la sicurezza dei gateway (gateway di Virtual Private Network – VPN); la modalità trasporto è una modalità più compatta per ottenere la sicurezza end-to-end perché non è necessario spedire un secondo header IP.

Il campo *padding* (la cui lunghezza è scritta nel campo *pad length*) può essere aggiunto per arrotondare la lunghezza del payload e del relativo trailer (campi *pad length* e *next* header) a un numero multiplo di 4 byte.

Infine il campo per il controllo dell'integrità dei dati (*integrity check value - ICV*) è utilizzato insieme alle altre informazioni dell'header ESP, del payload, del trailer per verificare l'integrità del pacchetto. La lunghezza di questo campo è definita durante le operazioni di associazione della sicurezza.

I nodi 6LoWPAN sono tipicamente dotati di hardware in grado di effettuare le elaborazioni di cifratura, decifratura e controllo dell'integrità dell'algoritmo AES/CCM. Questo hardware è tipicamente disponibile in maniera tale che il software dei livelli al di sopra del data-link possano utilizzarlo. In questi casi l'algoritmo AES/CCM è il candidato ovvio per la piattaforma di crittografia da utilizzare end-to-end per ottenere riservatezza e implementare i meccanismi di integrità e autenticazione.

In questi casi il payload crittografato comincia con la trasmissione in chiaro di un vettore di inizializzazione *(initialization vector - IV)* di otto byte. Questo è preceduto da 3 byte nel campo *SPI (SA)* utilizzati per produrre un "nonce" di 8 byte da inserire nell'algoritmo CCM. Seguirà il payload crittografato. Si noti che al payload sarà stato aggiunto un trailer prima della cifratura per rendere le sue dimensioni multiple di 4 in byte. Infine sarà aggiunto il campo ICV di 8 byte o opzionalmente di 12 byte (quale di questi due valori utilizzare sarà deciso durante le operazioni di associazione della sicurezza). ESP con AES/CCM può essere anche meno pesante per la crittografia end-to-end e il controllo dell'integrità fra nodi 6LoWPAN. La maggior parte dei chip che implementano il protocollo IEEE802.15.4 rendono le operazioni dell'overhead il più possibile veloci. L'overhead dei singoli pacchetti può essere reso più piccolo utilizzando da 1 a 4 byte per il padding (incluso il campo *pad length)* e 8 byte per la trasmissione esplicita dell'*initialization vector (IV)*. Se sono necessarie prestazioni più efficienti una versione speciale di AES/CCM potrebbe essere definita in modo da creare overhead ancora meno lunghi o utilizzare forme di speciali compressioni dell'header derivando il vettore di inizializzazione da altre informazioni presenti nella parte destinata al livello MAC del pacchetto.

La gestione della sicurezza nello strato di trasporto di 6LoWPAN

Il **transport layer security (TLS)** fornisce un canale end-to-end sicuro fra due nodi terminali della rete. Fornisce meccanismi per la riservatezza, l'integrità e l'autenticazione. TLS è stato originariamente sviluppato con il nome di *Secure Sockets Layer (SSL)* dalla *Netscape Corporation* per il loro web browser, ma è stato successivamente standardizzato dall'IETF come TLS.

TLS è costituito da diversi strati e protocolli. Nello strato più basso è utilizzato un algoritmo di crittografia simmetrico per fornire riservatezza e integrità. Per stabilire una chiave da utilizzare con l'algoritmo di crittografia simmetrica TLS prima esegue un'autenticazione combinata con un protocollo di scambio sicuro della chiave. L'autenticazione può essere sia unilatere (viene autenticato solo un nodo terminale della connessione) o bilatere (entrambi i nodi terminali sono autenticati).

Prima di cominciare la fase di autenticazione, il nodo terminale TLS inizia una fase di negoziazione del protocollo. Durante questa fase decide quale protocollo di crittografia utilizzare durante la connessione. Il TLS supporta diversi protocolli di crittografia. Nella fase di autenticazione fa un uso estensivo di algoritmi di crittografia asimmetrici. Per i dispositivi a bassa potenza questi meccanismi risultano non appropriati. Quindi, sono stati fatti diversi sforzi per rendere più leggeri gli algoritmi di crittografia per ottenere sicurezza end-to-end con microprocessori di non elevata potenza di calcolo. TLS può essere utilizzato fra le comunicazioni end-to-end che coinvolgono il sink e i rimanenti nodi della rete ma non è conveniente utilizzarlo per le comunicazioni che coinvolgono qualunque altra coppia di nodi. In particolare perché TLS richiede un meccanismo di trasporto affidabile (TCP) e spesso le WSN e 6LoWPAN non supportano TCP.

Il **Datagram Transport Layer Security (DTLS)** [140] è un protocollo utilizzato per il traffico a datagrammi sicuro per applicazioni client/server. DTLS è costituito da un *Record Protocol* che coinvolge altri protocolli che permettono operazioni di Handshake (stretta di mano), di Alert (allarmi) e di scambio di dati applicativi. Durante la fase iniziale di Handshake il server e opzionalmente il client si autenticano utilizzando una **Public Key Infrastructure (PKI)**. Le connessioni più leggere utilizzano il **Constrained Application protocol (CoAP)** specificatamente progettato per le *Low-Power and Lossy Networks (LLNs)* ma che facilmente si adatta al protocollo HTTP. In seno a CoAP è stato proposto l'uso di DTLS come protocollo di sicurezza per la gestione automatica della chiave, la crittografia dei dati e l'autenticazione. Il protcollo CoAP con il protocollo DTLS è noto come **secure-CoAP (CoAPs)**.

La Figura 27 mostra uno scenario di rete IoT che include una rete 6LoWPAN con dei nodi che utilizzano il protocollo CoAPs e un *6LoWPAN Border Router (6LBR)*.

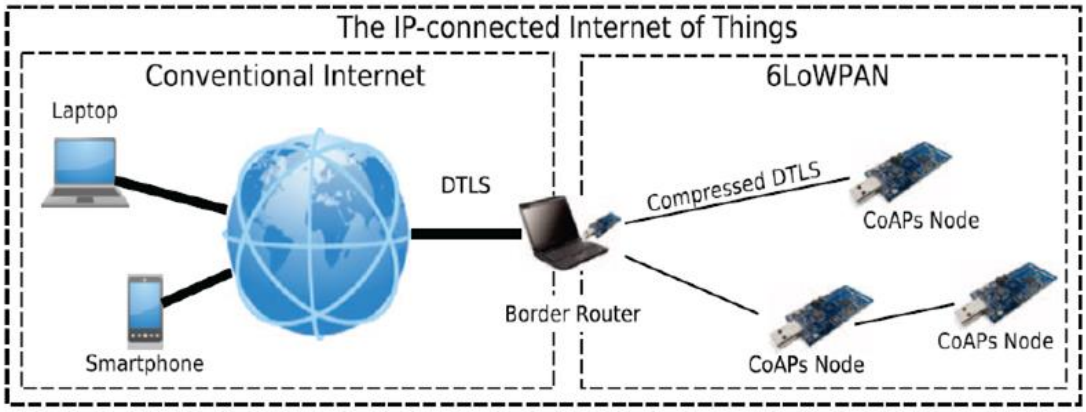


Figura 27 Scenario IoT - 6LoWPAN e CoAP.

I messaggi interni a 6LoWPAN viaggiano in formato compresso. Il 6LBR agisce da ponte fra la rete 6LoWPAN e Internet convenzionale e quindi permette l'accesso ai dispositiviCoAP/6LoWPAN da qualunque parte di Internet. Questo implica che 6LBR è un dispositivo autenticato della rete che ha ottenuto tale autenticazione prima del meccanismo di bootstrap. I dispositivi CoAPs possono comunicare con connessioni sicure con gli host di Internet come laptop e smartphone.

I dispositivi LLN avendo dei vincoli di risorse sono propensi ad aggressioni di tipo flooding con l'obiettivo di esaurire le risorse particolarmente limitate. Un aggressore è facilmente un host di Internet che conosce l'indirizzo del dispositivo da aggredire e può tentare di iniziare ripetutamente le operazioni di handshake con il dispositivo LLN con l'obiettivo di disturbarlo.

Per come è costituito il meccanismo di handshake di DTLS permette a tutti di iniziare la procedura, quindi permette al dispositivo di creare ripetutamente un nuovo contesto DTLS quando viene ricevuta una richiesta di handshake. Di conseguenza un aggressore può esaurire le risorse di un dispositivo LLN alimentato a batteria. Per prevenire questo tipo di attacchi nel protocollo DTLS è stato introdotto un meccanismo senza stato a cookie. Comunque dato che l'energia si consuma anche quando i messaggi vengono semplicemente inoltrati qualunque traffico di rete non desiderato e contraffatto potrebbe inondare la rete e utilizzare non solo le risorse del dispositivo di destinazione ma anche i nodi che effettuano le operazioni di routing e tutti gli altri nodi vicini che riceveranno il messaggio a livello fisico.

I messaggi CoAP contengono spesso comandi ai nodi per eseguire dei compiti specifici come "accendi le luci" o "invia un messaggio di aggiornamento". Le riproduzione contraffatta di questi messaggi può alterare il comportamento dei nodi e potrebbe portare a un fallimento completo del sistema. Senza il nodo 6LBR tali pacchetti contraffatti possono avere effetti sulle prestazioni paragonabili a quelli delle aggressioni di inondazione. Nel protocollo DTLS sono definiti dei meccanismi per rilevare i messaggi riprodotti in maniera contraffatta. Comunque accade che prima che un nodo rilevi che il messaggio è riprodotto in maniera contraffatta lo avrà già ricevuto, elaborato e possibilmente inoltrato. Tutte queste operazioni consumano energia con lo stesso effetto delle aggressioni di inondazione. Infine è necessario implementare dei meccanismi per identificare le aggressioni di tipo "amplificazione". I pacchetti "amplificati" sono pacchetti di Internet di dimensioni enormi e quindi comportano una frammentazione nel nodo 6LBR e questo comporta un elevato consumo di energia nello stesso nodo per effettuare le operazioni di frammentazione e de-frammentazione e anche nei nodi del percorso verso la destinazione che dovranno inoltrare una elevata quantità di pacchetti nati dalla frammentazione.

Essendo stato progettato per Internet, DTLS risulta comunque essere un protocollo molto pesante per i dispositivi di IoT. Tali dispositivi possono utilizzare il protocollo 6LoWPAN per comprimere i lunghi header del livello IP. Sfruttando le caratteristiche di compressione di 6LoWPAN è possibile comprimere i messaggi e gli header del protocollo DTLS.

Lo standard 6LoWPAN definisce la **compressione dell'header IP (IP Header Compression - IPHC)** per l'header IP e la **successiva compressione dell'header (Next Header Compression - NHC)** per i successivi

campi di estensione dell'header e l'header UDP. Lo standard non fornisce i modi con cui comprimere il carico UDP e degli eventuali strati superiori. L'IETF RFC 7400 "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)" propone un plug-in per 6LoWPAN che può essere utilizzato per comprimere il payload UDP.

DTLS come parte del payload UDP può essere compresso utilizzando la tecnica 6LoWPAN-GHC. I bit ID di NHC per UDP-GHC sono utilizzati per differenziare NHC per UDP da NHC per UDP-GHC. La sequenza "11010" in NHC per UDP-GHC indica che il payload UDP è compresso con 6LoWPAN-GHC. Una volta che UDP-GHC è definito, è possibile comprimere il protocollo DTLS fornendo GHC per i messeggi DTLS.

Quando si termina 6LoWPAN-GHC per DTLS si usa un codice di STOP (la sequenza "10010000") per indicare che la compressione dell'header DTLS è completata. Sono stati definiti 6LoWPAN-GHC per l'header Record, l'header di Handshake, i messaggi Hello del client e del server. Le stesse tecniche di compressione si possono utilizzare per altri messaggi di Handshake. Con queste tecniche di compressione si possono ottenere significative riduzioni dello spazio occupato.

7.6 Gestione della sicurezza utilizzando il sistema operativo TinyOS

In TinyOS la sicurezza può essere implementata a livello data-link utilizzando lo standard **AES** e quindi fornendo la sicurezza per una connessione a singolo hop [137]. Un semplice esempio può essere provato in accordo all'applicazione *CoapBlip* inclusa nell'ambiente TinyOS ($TOSROOT/apps/CoapBlip).

Sono necessari due motes uno configurato come server e uno come router che sfrutta il protocollo Point-to-Point (PPP) ($TOSROOT/apps/PppRouter).

L'utente può modificare l'elenco delle risorse disponibili nel mote server attraverso il Makefile utilizzando le variabili DCOAP_RESOURCE_TEMP (temperatura), DCOAP_RESOURCE_HUM (umidità), DCOAP_RESOURCE_VOLT (tensione) e DCOAP_RESOURCE_LED (led). È possibile anche cambiare il prefisso IPv6 attraverso la variabile DIN6_PREFIX.

Quando viene utilizzato il chip CC2420 è possibile utilizzare l'accelerazione hardware per effettuare le operazioni di cifrature dell'algoritmo AES. Per il loro utilizzo è necessario aggiungere il flag CC2420_HW_SECURITY nel Makefile. Un esempio è già incluso nell'ambiente TinyOS nella cartella

$TOSROOT/apps/tests/cc2420/TestSecurity.

TinySec supporta due modalità operative: solo autenticazione **(TinySec-Auth)** e autenticazione e crittografia **(TinySec-AE)**. Il payload è crittografato utilizzando un cifrario a blocchi di Skipjack. L'autenticazione viene effettuata eseguendo un CBC-MAC sul payload crittografato e l'header del pacchetto. In TinySec-Auth l'intero pacchetto è autenticato ma i dati non sono crittografati. TinySec esegue la crittografia solo via software e quindi non si può avvantaggiare di alcuna accelerazione hardware. L'intera implementazione di TinySec richiede 256 byte di RAM e 8152 byte di ROM.

Lo sviluppo delle curve di crittografie ellittiche **(Elliptic Curve Cryptography - ECC)** ha dimostrato che nelle WSN possono essere utilizzate chiave di crittografia pubbliche **(Public Key Cryptography – PKC).** ECC è basato sulle strutture algebriche delle curve ellittiche su campi finiti. Più difficile è risolvere un problema matematico più sicuro sarà l'algoritmo. Ad esempio l'algoritmo **Rivest Shamir Adleman (RSA)** si basa su un problema di fattorizzazione di interi che ha una soluzione sub esponenziale. ECC è basato su un problema di curve ellittiche logaritmico discreto (*Elliptic Curve Discrete Logarithmic Problem - ECDLP)* con una soluzione totalmente esponenziale. Di conseguenza ECC può ottenere lo stesso livello di sicurezza si RSA ma con delle chiavi sensibilmente più piccole. Una chiave a 160 bit di ECC fornisce lo stesso livello di sicurezza di una a 1024 bit di RSA e una chiave a 224 bit di ECC è equivalente a una a 2048 bit di RSA. Chiavi più piccole permettono di effettuare calcoli più veloci, di occupare meno memoria, consumare meno energia e banda di trasmissione. L'uso di ECC permette approssimativamente di diminuire di 1.5 volte il consumo di memoria e ottenere una velocità nelle operazioni di crittografia fino a 5 volte maggiore rispetto a quella di RSA.

TinyECC è una libreria portatile ed efficiente sviluppata da A Liu et al. dell'Università del Nord Carolina. Fornisce un meccanismo di firma digitale chiamato *Elliptic Curve Digital Signature Algorithm (ECDSA)*, un protocollo per lo scambio delle chiavi chiamato *Elliptic Curve Diffie Hellman (ECDH)* e un meccanismo di crittografia a chiave pubblica chiamato *(ECIES)*. L'ultima versione di TinyECC (la 2.0) supporta diverse piattaforme come MICA2/MICAz, TelosB/Tmote Sky, BSNV3 e Imote2. Per poterla utilizzare si deve installare la versione di TinyOS 2.1.1 o una versione più recente.

TinyECC fornisce le seguenti interfacce:

- NN.nc definisce l'interfaccia NN implementata nel file NNM.nc e relativa alle operazioni con i grandi numeri interi.
- ECC.nc definisce l'interfaccia ECC implementata nel file ECCM.nc e fornisce le operazioni di base e avanzate sulle curve ellittiche.
- ECDSA.nc definisce l'interfaccia ECDSA implementata nel file ECDSAM.nc e fornisce la verifica e la generazione delle firme ECDSA.
- ECIES.nc definisce l'interfaccia ECIES implementata nel file ECIESM.nc e fornisce le operazioni di crittografia e decrittografia ECIES.
- ECDH.nc definisce l'interfaccia ECDH implementata nel file ECDHM.nc e calcola le chiavi ECDH stabilite.
- SHA1.nc definisce l'interfaccia SHA1 implementata nel file SHA1M.nc e fornisce le funzioni SHA-1.
- CurveParam.nc definisce l'interfaccia CurveParam che permette di ottenere i parametri delle curve ellittiche e ottimizzare le moltiplicazioni. Insieme ai file omega. secp128*.nc, secp160*.nc, secp192*.nc implementa questa interfaccia per fornire i parametri per definire le curve ellittiche dello *Standards for Efficient Cryptography Group (SECG)*. I nomi delle curve devono essere definite nel makefile per selezionare i parametri delle curve ellittiche.

TinyECC è scritto in NesC. È pronto per l'uso ed è un software che abilita le operazioni di crittografia a chiave pubblica ECC ed include l'ottimizzazione di tali operazioni. TinyECC supporta i meccanismi ECDSA, ECDH e ECIES definiti negli standard per la crittografia efficiente (Standards for Efficient Cryptography) e i parametri delle curve ellittiche come secp160k1, secp160r1 e secp160r2. ECDH (rispettivamente ECDSA) è unna variante del protocollo di accordo sulle chiavi di Hellman (rispettivamente Digital Signature Algorithm). ECIES, noto anche come *Elliptic Curve Augmented Encryption Scheme ECAES*, supporta la sicurezza semantica. Le ottimizzazioni di ECC includono la Barrett Reduction, le Hybrid Multiplication, la Hybrid Squaring, il Projective Coordinate System, lo Sliding Window Method, le Shamir's Trick e le Curve-Specific. TinyECC supporta i parametri ECC a 128 bit, 160 bit e 192 bit. Ricordiamo che i parametri ECC a 160 bit hanno lo stesso livello di sicurezza di RSA a 1024 bit.

7.7 Gestione della sicurezza utilizzando il sistema operativo Contiki

Nel 2009 è stato introdotto un livello di rete sicuro per le WSN basate sul sistema operativo Contiki denominato **ContikiSec** [137]. Fornisce tre modalità operative: riservatezza **(ContikiSec-Enc)**, autenticazione **(ContikiSec-Auth)** e contemporaneamente autenticazione e crittografia **ContikiSec-AE)**.

AES-CBC-CS (Cipher Block Chaining-Ciphertext Stealing) è stata selezionata come unica modalità di cifratura. Il cifrario a blocchi simmetrico richiede un **vettore di inizializzazione (initialization vector - IV)** costituito da un blocco di dati casuali. Contiki-Auth utilizza l'**algoritmo CMAC (Cipher-based Message Authentication Code)** per la generazione di un campo MAC di 4 byte. Contiki-AE che ottiene il più alto livello di sicurezza fornendo contemporaneamente riservatezza, autenticazione e integrità dei dati utilizza l'**Offset Codebook Mode (OCB)** che è basato su AES.

Uno dei maggiori svantaggi di ContikiSek è l'uso di un meccanismo semplice di gestione della chiave per il quale tutti i nodi della rete presentano la stessa chiave a 128 bit. L'intera rete sarà molto più vulnerabile a diverse aggressioni. Infatti se un nodo è compromesso l'intera rete diverrà insicura.

Una versione di IPSec per una 6LoWPAN basata sul sistema operativo Contiki è stata rilasciata da Shahid Raza dello Swedish Institute of Computer Science (SICS) ed è disponibile online al seguente link https://svn.code.sf.net/p/contikiprojects/code/sics.se/ipsec/. Questa soluzione di IPSec utilizza una chiave precondivisa per stabilire le SA. In altre parole lo schema di gestione delle chiavi della piattaforma IPSec basato sul protocollo IKEv2 non è stato implementato. È stata implementata la versione compressa di IPSec. Per l'autenticazione AH la modalità HMAC-SHA1-96 necessita di 24 byte (ridotti a 16 byte per la versione compressa). In confronto il protocollo IEEE802.15.4 in modalità AES-CBC-MAC-96 usa 12 byte. Per la crittografia ESP, AES-CBC necessita di 18 Bytes (ridotti a 12 byte per la versione compressa). In confronto lo standard IEEE802.15.4 in modalità AES-CTR usa 5 byte. Per la crittografia e l'autenticazione ESP la modalità AES-CBC e HMAC-SHA1-96 necessitano di 30 byte (ridotti a 24 byte nella versione compressa). In confronto lo standard IEEE802.15.4 in modalità AES-CCM-128 usa 21 byte.

V. Perelman ha proposto una implementazione di **TLS/DTLS** per il sistema operativo Contiki basato su un cifrario a chiave precondivisa **(TLS_PSK_WITH_AES_128_CCM_8)**. Il codice sorgente della libreria è disponibile pubblicamente con la BSD 2-Clause Licence e può essere scaricato dal link cnds.eecs.jacobs-university.de/software.

O. Bergmann ha progettato una specifica libreria che permette di creare un server con il supporto di **DTLS**. La versione attuale è la 0.8.2 (February 2015), il sorgente è disponibile al link https://sourceforge.net/projects/tinydtls/files/.

7.8 Recenti attività di ricerca sulla sicurezza nelle WSN

Recentemente è stata effettuata in [141] una classificazione delle possibili aggressioni che possono essere effettuate a una WSN in tre categorie: obiettivi, prestazioni e attività degli strati.

- Classificazione delle aggressioni verso lo scopo di funzionamento della rete: possiamo distinguere aggressioni passive e attive.
  - Aggressioni Passive [142] [143] [144]: Queste aggressioni sono principalmente contro la riservatezza dei dati. Un aggressore monitora il traffico non crittografato e cerca informazioni sensibili che possono essere usate per altri tipi di aggressioni. Le aggressioni passive includono l'analisi del traffico, il monitoraggio delle comunicazioni, la decifratura di traffico crittografato in maniera poco robusta e la cattura di informazioni di autenticazione. L'intercettazione passiva delle operazioni di rete permette all'avversario di prevedere le azioni che verranno intraprese. Queste aggressioni permettono che vengano scoperte informazioni senza il consenso o la conoscenza degli utenti della rete.
  - Aggressioni Attive: in questo caso l'aggressore esegue delle operazioni per ottenere il controllo della rete. Alcuni esempi sono DoS, modifica dei dati, buco nero, duplicazioni, dolina, contraffazioni, inondazioni, generazione di interferenze, generazione di segnali ad elevata potenza, buco del verme, falsificazioni, inondazione con pacchetti HELLO, contraffazione di nodi, distruzione della cooperazione fra i nodi, aggressione "man-in-middle", inoltro selettivo e nodi contraffatti.
- Classificazione delle aggressioni verso le prestazioni di funzionamento della rete [145] [146] [147]: possiamo distinguere in aggressioni esterne e interne.
  - Aggressioni esterne: possono causare una intercettazione passiva sui dati trasmessi così come possono estendere l'inserimento di dati falsi nella rete per consumare le risorse di rete a portare ad aggressioni di tipo DoS.

- Aggressioni interne: possono danneggiare la rete furtivamente impedendo autenticazioni e autorizzazioni in quanto si inseriscono come nodi legittimi della rete, hanno accesso alle informazioni e non è facile prevedere le mosse delle loro aggressioni. Gli aggressori interni possono lanciare diversi tipi di attacchi come modifiche, intercettazioni, errati inoltri o scarto di pacchetti. Quest'ultimo tipo di attacco è difficile da scoprire perché non si può facilmente distinguere se il pacchetto si è perso per un'aggressione, per una collisione o per il rumore. Inoltre non permette all'informazione di raggiungere la base station e degrada significativamente le prestazioni della rete come il tasso di consegna dei pacchetti che si abbassa notevolmente a causa degli scarti ripetuti.

- Classificazione delle aggressioni in funzione dello strato di rete coinvolto.
  - Livello fisico [148] [149] [150]: tale tipo di aggressione spazia dalla cattura materiale del nodo ai disturbi sul canale radio. L'assenza di un controllo materiale sui singoli nodi rende questo tipo di aggressione difficile da prevenire rispetto alle aggressioni di tipo software. Il disturbo del canale radio è una delle aggressioni più importanti a livello fisico perché interferisce con le normali operazioni della rete. Un aggressore può trasmettere continuamente un segnale radio sul canale wireless, può spedire segnali ad elevata energia per bloccare il mezzo wireless e impedire ai nodi la comunicazione. Questo può portare a un aggressione DoS per questo livello.
  - Livello Data Link [147] [150] [151]: gli aggressori possono deliberatamente violare il comportamento predefinito dei protocolli a livello data link. Per esempio può indurre collisioni distruggendo un pacchetto e quindi causando un elevato consumo di energia del nodo a causa delle ripetute ritrasmissioni. Inoltre può intercettare ed esaminare i messaggi con lo scopo di dedurre informazioni dal loro andamento. Questo può essere effettuato anche se i messaggi sono cifrati. Infine potrebbe anche provocare la non equità fra i nodi ingannando il meccanismo di priorità dello strato MAC.
  - Livello di rete [145] [149] [152]: in questo strato sono tipiche le aggressioni di tipo DoS che mirano a distruggere le informazioni di instradamento e quindi le operazioni di tutta la rete ad-hoc. L'aggressione di tipo dolina come già detto cerca di realizzare un'esca per tutto il traffico verso il nodo contraffatto creando una metaforica dolina con il nodo compromesso al centro. Anche se l'aggressore riesce a conquistare un singolo nodo questo può essere sufficiente per ottenere il controllo dell'intera rete. I nodi aggressori possono rifiutarsi di inoltrare certi messaggi o scartarli. L'inserimento di informazioni di instradamento falsificate, alterate o duplicate è la forma più diretta di aggressione contro i protocolli di instradamento in ogni rete.
  - Livello di trasporto [146] [152]: un aggressore può ripetutamente fare richieste di nuove connessioni finché non vengono esaurite le risorse dei nodi coinvolti nelle connessioni.
  - Livello applicazione [151] [153]: diversi tipi di aggressioni possono essere condotte in questo livello come sopraffazione, sconfessione, correzione dei dati ed esecuzione di codice contraffatto. Nell'aggressione di sopraffazione, ad esempio, l'aggressore cerca di inoltrare enormi volumi di traffico verso la base station occupando tutta la larghezza di banda della rete ed esaurendo le risorse energetiche dei nodi.

Recentemente una enorme attività di ricerca si è focalizzata nel campo della sicurezza delle WSN. Molti ricercatori hanno fornito soluzioni utilizzando chiavi di crittografia simmetriche. La sicurezza degli algoritmi di crittografia asimmetrica è legata alla complessità del problema matematico che utilizza, questo tipicamente comporta un considerevole dispendio di energia rispetto agli algoritmi con chiave simmetrica che sono realizzati applicando iterativamente semplici operazioni di crittografia. Nonostante i sistemi di crittografia a chiave pubblica sono considerati troppo pesanti per essere utilizzati nelle WSN, recentemente, diversi lavori di ricerca hanno dimostrato la possibilità di implementazione.

- Algoritmi di crittografia simmetrica nelle WSN

L'idea di base è quella di caricare le informazioni di segretezza nei nodi sensori prima che questi vengano installati nella rete. Le informazioni di segretezza possono essere le stesse chiavi o informazioni ausiliarie che permettano ai nodi di ricavare le vere chiavi. Con la chiave segreta i nodi possono comunicare in maniera sicura [154]. Lo svantaggio principale di questa modalità è che compromettendo un nodo (accedendo alla chiave in esso caricata) si può compromettere l'intera rete. Per superare questa limitazione sono stati proposti diversi meccanismi che permettono di stabilire coppie di chiavi piuttosto che un'unica chiave globale. Perring et al. hanno proposto **SPINS**, un protocollo di gestione delle chiavi che è basato su una base station di fiducia in grado di distribuire le chiavi. SPINS è costituito da due parti: SNEP (Secure Network Encryption Protocol) e µTESLA (micro time efficient streaming loss tolerant authentication). Questo protocollo offre molte proprietà di sicurezza come quella semantica, di autenticazione dei dati, protezione contro i duplicati, "freschezza" dei dati ed è ottimizzato per comunicazioni wireless [155] [156]. **LEAP** (Localized Encryption and Authentication Protocol) [157] è un protocollo di gestione della chiave per supportare diversi profili di comunicazione. Ogni nodo memorizza quattro tipi di chiavi: individuale, di coppia, di cluster, di gruppo. La chiave individuale è condivisa da un nodo e dalla stazione base. La chiave di coppia è condivisa fra un nodo e ognuno dei sui vicini (una per ogni vicino). La chiave di cluster è condivisa da un nodo con tutti i suoi vicini (la stessa per tutto il cluster). Una chiave di gruppo è una chiave comune per l'intera rete. Le chiavi individuali sono precaricate nei nodi. Dopo l'installazione, i nodi vicini stabiliscono le chiavi di coppia. Essi si autenticano utilizzando la chiave individuale precaricata la quale viene cancellata non appena le chiavi di coppia sono stabilite. Per stabilire le chiavi di cluster e la chiave di gruppo i nodi utilizzano messaggi in broadcast e inoltrati. Il protocollo utilizza µTesla [155] per autenticare le trasmissioni in broadcast.

In **BROSK** (BROadcast Session Key negotiation protocol) [158] ogni nodo invia in bradcast un messaggio che contiene il suo "nonce". In questo modo ogni due nodi vicini che si sentono vicendevolmente possono calcolare una chiave comune che è funzione dei lore due "nonce". I nodi vicini si autenticano con una chiave precaricata che si suppone sia non leggibile nel caso in cui il nodo venga catturato da un avversario.

In [159] **Blom** descrive una sistema di generazione di una classe ottima di chiavi simmetriche. In questa soluzione alcune delle possibili chiavi di collegamento di una rete di N nodi sono rappresentati come una matrice di chiavi di dimensione $(\lambda+1) \times N$. Il meccanismo memorizza piccole quantità di informazione in ogni nodo sensore in modo che alcune coppie di nodi possano calcolare i relativi campi della matrice e usarli come chiave del collegamento. Questa soluzione assicura una sicurezza di livello λ, che significa che le chiavi sono sicure se non più di λ nodi sono compromessi. Un'altra soluzione di livello λ è presentata in [160] ed è chiamata **Polynomial-based key pre distribution scheme**. Questo meccanismo distribuisce un polinomio condiviso da ogni sensore. Ogni sensore memorizza un polinomio con $(\lambda+1)$ coefficienti e ogni coppia di nodi può stabilire una chiave utilizzando le proprietà di simmetria dei polinomi. La soluzione è di livello λ in quanto un gruppo di nodi minore di $\lambda+1$ non può risalire alle coppie di chiavi. **LBKs** (location-based keys) [161] si basa sulla localizzazione dell'informazione per ottenere la gestione della chiave. Le chiavi sono stabilite in accordo al posizionamento geografico del sensore. **Eschenauer and Gligor** [162] hanno proposto un meccanismo basato su una preventiva distribuzione casuale della chiave. Ogni sensore casualmente preleva un insieme di chiavi e il loro identificatore da un gruppo prima dell'installazione. Quindi una fase di scoperta della chiave condivisa viene eseguita. Durante questa fase due vicini si scambiano e confrontano una lista di identificazione delle chiavi nelle loro catene di chiave. Praticamente ogni nodo sensore invia in broadcast un messaggio e riceve un messaggio da ogni nodo in visibilità radio che trasporta le liste con l'ID di chiave. In questo modo ogni nodo ha una certa probabilità di condividere almeno una chiave comune. La complicazione di questo meccanismo è quella di trovare un buon compromesso fra la dimensione del gruppo di chiavi e il numero di chiavi memorizzate in ogni nodo per trovare la migliore probabilità. Lo svantaggio

principale è che se il numero di nodi compromessi aumenta, la parte di collegamenti affetti incrementa proporzionalmente. **Key Infection** [163] è un protocollo di sicurezza leggero utilizzabile per prodotti che non hanno particolari esigenze di sicurezza dove un aggressore può monitorare solo una percentuale fissa di canali di comunicazione.

- Algoritmi di crittografia simmetrica nelle WSN

  Gura et al. [164] hanno dimostrato che entrambe le crittografie RSA e basate su curve ellittiche sono implementabili su piccoli dispositivi senza accelerazione hardware. Utilizzando una CPU a 8 bit CPUs, ECC ha dei vantaggi prestazionali rispetto a RSA. Un ulteriore vantaggio è che le chiavi a 160 bit di ECC permettono di trasmettere messaggi più corti rispetto alle chiavi a 1024 bit di RSA. In particolare hanno dimostrato che la moltiplicazione ECC è confrontabile alle operazioni a chiave pubblica RSA e un ordine di grandezza più veloci delle operazioni a chiave privata RSA

  In [165] **Watro** et al. hanno mostrato che parte del sistema di crittografia RSA può esere successivamente implementato in un sensore wireless. Il sistema **TinyPK** [165] è progettato per permettere l'autenticazione e la condivisione della chiave fra sensori a risorse limitate. Il protocollo è utilizzato insieme a servizi di crittografia simmetrica per nodi di rete come TinySec. In particolare sono state implementate le operazioni pubbliche RSA sui sensori e quelle private su altri dispositivi come laptop. In [166] **Malan** et al. hanno dimostrato l'implementazione del metodo di crittografia di Diffie-Hellman basato sul Elliptic Curve Discrete Logarithm Problem. Inoltre hanno mostrato che una chiave pubblica può essere generata in 34 secondi e che i parametri per la sicurezza possono essere distribuiti fra i nodi di una stessa rete basata su MICA2 utilizzando solo 1 kbyte di RAM e 34 kbyte di ROM.

  Wang et al. nel lavoro [167] hanno proposto un meccanismo a chiave pubblica per WSN. Hanno realizzato un accesso basato su ECC che permette di stabilire una coppia di chiavi e il controllo d'accesso sia locale sia remoto. Hanno eseguito dei test di confronto dell'implementazione fra le tecniche a chiave simmetrica e chiave pubblica utilizzando un nodo MICAz e un HP iPAQ. Hanno dimostrato che in questo caso un meccanismo a chiave pubblica è più vantaggioso di un meccanismo a chiave simmetrica in termini di uso della memoria, complessità del messaggio ed elasticità nella sicurezza. In [168] è presentata una implementazione di ECC molto efficiente chiamata **WM ECC** basata sulle operazioni sui numeri primi.

  TinyPEDS [169], sviluppato per WSN asincrone, permette la memorizzazione distribuita efficiente e riservata di dati sensibili su dispositivi con limitate risorse. **TinyPBC** [170] è una implementazione efficiente delle funzioni dei meccanismi di crittografia basati sulla *Pairing-based Cryptography (PBC)* in un processore a 8 bit in grado di elaborare le coppie di chiavi in solo 5.45s utilizzando un ATmega128L.

7.9 La sicurezza nelle comunicazioni machine-to-machine

Le comunicazioni Machine-to-Machine (M2M) chiamate anche Machine-Type-Communication (MTC) dal Third Generation Partnership Project (3GPP) sono relative a comunicazioni fra piccoli e poco costosi dispositivi dove non è prevista la partecipazione di operatori umani.

Un importante ostacolo che ha reso la crescita di M2M a macchia di leopardo e ostacola il massiccio uso di applicazioni M2M è la sicurezza. Inoltre la natura dell'ambiente di implementazione di M2M e la sensitività dei dati da scambiare rendono le reti M2M vulnerabili a numerose aggressioni di tipo hardware, software e di rete. I dispositivi M2M sono più pervasivi dei dispositivi di comunicazione personali quindi possono essere collezionate grandi quantità di dati personali. Se on vengono prese appropriate misure di sicurezza si può andare incontro a diverse problematiche legate alla riservatezza delle persone e dei dati [171].

La specifica ETSI TS 102 690 [27] fornisce un'architettura M2M con un insieme generico di caratteristiche. Introduce un Service Capability Layer (SCL) progettato per lavorare con uno stile di architettura compatibile

con il Representational State Transfer (REST). SCL ha un ruolo significativo nell'architettura ETSI M2M in quanto fornisce funzioni che devono essere condivise fra le diverse applicazioni e permette di esporle attraverso un insieme di interfacce aperte. Fra i diversi SCL c'è un servizio chiamato SEC (per la sicurezza) che gestisce i servizi di bootstrap M2M e permette di stabilire la chiave. Generic Communication (xGC) è un altro servizio responsabile dello scambio di dati sicuri e della distribuzione delle applicazioni. Si noti che la "x" può essere "D", "G" o N in riferimento rispettivamente a Device, Gateway e Network. Per permettere la comunicazione fra le diverse entità di M2M sono state definite 4 interfacce: mIa, dIa, mId [172] e mIm [27]. **dIa** collega l'applicazione con l'SCL localizzato nel dispositivo. Si possono avere tre scenari: device application (DA) e service capability layer (DSCL) nello stesso dispositivo; nessun servizio implementato nel dispositivo in quanto risiede nel gateway (GSCL); dispositivo non conforme con lo standard ETSI e allora la DA si connette a un servizio di rete (NSCL). **mId** è l'interfaccia che collega due servizi uno che risiede nel dominio di rete (NSCL) e l'altro nel dominio del dispositivo (DSCL/GSCL). **mIa** è l'interfaccia fra l'NSCL e l'applicazione che risiede nel dominio di rete (NA). **mIm** è un'interfaccia inter-dominio che permette a due differenti NSCL in due diverse reti di comunicare.

Insieme con l'architettura M2M e le relative interfacce, lo standard ETSI propone anche una gerarchia di chiavi in cui ogni insieme di dispositivi/gateway M2M mantiene una chiave radice (K_{mr}). La chiave radice può essere consegnata al dispositivo o al gateway M2M in tre diversi modi. Il primo consiste nel fornirla all'interno di un ambiente sicuro (Secure Environment) durante la fase di realizzazione o posizionamento del dispositivo. Nel secondo modo la chiave radice viene derivata dalle credenziali d'accesso alla rete in base a un Authentication and Key Agreement (AKA) utilizzando delle procedure basate su Generic Bootstrapping Architecture (GBA) o Extensible Authentication Protocol (EAP). L'ultimo modo la chiave radice M2M è fornita in una procedura d'accesso indipendente alla rete attraverso l'uso di EAP over PANA [173] o TLS over TCP. A partire dalla chiave radice viene ricavata una chiave di connessione M2M (K_{mc}). La chiave K_{mc} è utilizzata per rendere sicura l'interfaccia mId che collega il DSCL/GSCL e il NSCL. Una nuova chiave di connessione viene calcolata per ogni procedura di servizio di connessione M2M.

Come per le WSN i dispositivi M2M sono generalmente installati in posizioni raggiungibili e ci si aspetta che operino per lunghi periodi di tempo.

Quindi diversi tipo di aggressioni materiali possono essere condotte contro i dispositivi [174]. Inoltre dato che i dati misurati sono inoltrati a server remoti attraverso una rete wireless o wired, M2M può essere soggetta ad aggressioni verso le informazioni inoltrate. Possono essere eseguite anche aggressioni verso il corretto funzionamento del sistema. Si possono identificare alcune delle principali minacce verso le comunicazioni M2M e classificarle in tre categorie in funzione del fatto che l'obiettivo sia lo stesso dispositivo M2M, il corretto funzionamento del sistema o i dati che vengono scambiati. In quest'ultimo caso possiamo ulteriormente distinguere aggressioni di tipo fisico, logico o sui dati.

- *Physical Attacks:* queste aggressioni hanno come obiettivo lo strato di rete fisico [175] ma anche l'hardware o il software dei dispositivi M2M.
    - *Side Channel Attacks* [176]: i dispositivi M2M sono generalmente installati in posizioni raggiungibili dove gli avversari possono facilmente accedervi ed eseguire aggressioni al canale. Queste aggressioni potrebbero essere sia basate sul consumo di energia, sulle informazioni di temporizzazione, interruzione del canale e capacità di recuperare le chiavi di segretezza utilizzate.
    - *Software Modification e Malwares:* modifiche del software possono essere effettuate da un avversario o da un utente ingannevole per alterare le corrette operazioni del dispositivo M2M. L'utente ingannevole può fare in modo di ridurre la tariffa che deve pagare. Per esempio alcuni comportamenti di questo tipo possono riguardare i misuratori intelligenti o le applicazioni di pagamento elettronico. Queste aggressioni possono essere effettuate anche senza l'accesso materiale al dispositivo M2M dato che spesso è permesso l'aggiornamento del software in modalità Over The Air (OTA). L'impatto di queste minacce è ancora peggiore se si considerano applicazioni di salute o dei trasporti in cui l'avversario può prendere il controllo del dispositivo.

- *Distruzione o furto del dispositivo M2M o della Universal Integrated Circuit Card (UICC) in esso inclusa*: essendo installati in posizioni raggiungibili i dispositivi M2M o le loro UICC possono essere facilmente rubate. Comunque il furto delle UICC non sarà più possibile in quanto si sta lavorando a una UICC integrata nota come eUICC [177] che sarà saldata con il dispositivo M2M.

- *Logical Attacks:* queste aggressioni hanno come obiettivo il corretto funzionamento del sistema senza effettuare alcun cambiamento del software dei dispositivi.
  - *Imitazione*: quando si ha a che fare con comunicazioni M2M un aggressore può spiare l'identità di un server interno, un dispositivo M2M, un gateway e così via. Queste aggressioni possono portare a importanti perdite finanziarie e umane. Per esempio un avversario che riesce a spiare l'identità di un misuratore intelligente può caricare i propri pagamenti nel conto di un altro. È ancora peggio se riesce a impersonare il server in quanto sarà in grado di spedire comandi di controllo a tutti i dispositivi M2M della rete. In alcune applicazioni questi attacchi possono mettere in pericolo le vite umane.
  - *Negazione del servizio (Denial of Service - DoS)* [178]*:* dato che la maggior parte dei dispositivi M2M è alimentata a batteria, la diffusione costante di pacchetti non significativi può prematuramente consumare la carica della batteria e provocare il fallimento dell'applicazione. Questi attacchi possono essere eseguiti, per esempio, in riferimento ad applicazioni di telesorveglianza per evitare che l'applicazione sia in grado di rilevare l'intrusione. Inoltre aggressioni DoS possono essere generate per provocare l'indisponibilità delle risorse che può portate ad enormi perdite finanziarie in molte applicazioni e perfino provocare dei blackout quando si ha a che fare con reti intelligenti di trasporto dell'energia. Si noti che ogni tipo di entità può essere soggetta ad aggressioni DOS come i dispositivi, i gateway, l'infrastruttura sottostante e perfino il server remoto.
  - *Aggressioni di ritrasmissione (Relay Attacks)* : un avversario può condurre un'aggressione di ritrasmissione per convincere un'entità che è nelle vicinanze del trasmettitore o del ricevitore. Anche questa aggressione può avere come obiettivo i dispositivi, i gateway o l'intero dominio di rete. Per esempio può essere condotta con un'applicazione di pagamento elettronico per inoltrare la richiesta di pagamento di una unità a bordo di un veicolo *(On Board Unit - OBU)* per fare in modo che il proprietario paghi in più per viaggi che non ha mai fatto.

- *Aggressione verso i dati:* in questi casi l'obiettivo è l'informazione scambiata.
  - *Aggressioni alla riservatezza* [179]: a causa della pervasività dei dispositivi M2M attraverso le intercettazioni dei pacchetti scambiati, aggressori possono invadere la riservatezza degli utenti collegandosi ai dispositivi M2M o alle informazioni in transito verso gli individui e quindi ricavare abitudini degli utenti, condizioni di salute e così via. Per esempio, dato che l'indirizzo MAC del dispositivo è statico e indica che si tratta di un dispositivo di monitoraggio del cuore, quindi dal semplice riconoscimento di tale indirizzo l'avversario saprà che il proprietario soffre di problemi al cuore. Alcune applicazioni come quelle legate a pagamenti di assicurazione o elettronici in cui le informazioni di localizzazione sono spedite al fornitore del servizio, comportano seri rischi di violazione della riservatezza in quanto abilitano un potenziale avversario a monitorare i viaggi degli utenti.
  - *Modifica dei dati e inserimento di informazioni false:* i dati possono essere compromessi durante la loro trasmissione così come su un dispositivo o una applicazione server. Se si considera il caso di applicazione di salute elettronica o chiamata elettronica, la modifica dei valori misurati e delle informazioni di localizzazione può causare la morte delle persone. D'altro canto, in alcune applicazioni, l'inserimento di dati falsi [180] può causare gravi perdite finanziarie. Nel caso di applicazioni di chiamata elettronica, per esempio, i servizi d'emergenza potrebbero essere chiamati e arrivare sul posto senza la reale necessità.

- *Intercettazione e inoltro selettivo:* un avversario può intercettare e ritardare o scartare alcuni pacchetti ricevuti. L'impatto di una tale minaccia dipende dal contenuto dei pacchetti scartati. Se le informazioni scartate originano da un'applicazione sensibile come quella di una chiamata elettronica per esempio, l'impatto di tale minaccia può essere alquanto importante. Generalmente tali attacchi sono condotti contro le infrastrutture sottostanti come il dominio di rete ma potrebbero anche essere effettuati contro gateway M2M.

Attualmente ci sono state diverse proposte che tentano di mettere in sicurezza le comunicazioni M2M sviluppate non solo in ambito accademico e dalle industrie manifatturiere ma anche da organizzazioni di standardizzazione come l'iniziativa globale di *OneM2M* e l'*ETSI*.

Possiamo distinguere 5 diversi tipi di problematiche da risolvere: gestione delle chiavi *(Key Management)*, autenticazione *(Authentication)*, riservatezza della vita delle persone *(Privacy)*, riservatezza dei dati *(Confidentiality)*, integrità dei dati *(Integrity).*

- *Key Management*

  La soluzione della problematica della gestione delle chiavi è alla base del funzionamento di tutte le altre problematiche di sicurezza. A causa delle limitate risorse dei dispositivi M2M non è posssible pensare a sistemi di crittografia a chiave pubblica *(Public Key Cryptography - PKC)* e alla necessità di una infrastruttura di chiave pubblica *(Public Key Infrastructure - PKI)* o a un generatore di chiave privata *(Private Key Generator – PKG).* Sebbene molti dei sistemi di gestione della chiave proposti fanno riferimento a *PKC* quasi tutte le soluzioni propongano l'uso di una chiave simmetrica fra i dispositivi M2M e un gateway o un server remoto.

  - Nel meccanismo proposto da **Ben Saied** et al. [181] un nodo M2M che non può gestire pesanti operazioni asimmetriche affida a vicini più potenti di effettuare tali operazioni per suo conto. Inoltre questi proxie intraprendono la consegna di un codice casuale dal nodo in questione al server remoto e viceversa. In base allo scambio di questi due codici segreti entrambe le entità calcoleranno la chiave simmetrica da utilizzare per le successive comunicazioni sicure. Per evitare che i codici casuali siano conosciuti vengono suddivisi in diversi scambi e un solo scambio è possibile per ogni proxy cifrato con la coppia di chiavi precondivise. Una volta ricevuto ogni proxy utilizzerà la coppia pubblica/privata di chiavi, fornite da una terza parte di affidabilità *(Trusted Third Party - TTP)*, per inoltrare in maniera sicura la sua condivisione al server remoto.
  - Il meccanismo proposto da **Hussan** et al. di gestione della chiave è progettato per dispositivi che utilizzano IPv6 su reti 6LoWPAN [81]. In questa soluzione i dispositivi sono classificati in tre categorie in funzione delle loro risorse e del loro ruolo: dispositivi terminali *(End Device - 6ED)* che rappresentano i nodi con le risorse più stringenti, *Router (6LR)* che hanno vincoli meno stringenti e sono intermediari tra i nodi e i *Border Router (6LBR)* che sono autorizzati a gestire l'autenticazione di altri dispositivi. Lo schema consiste in tre fasi: una fase di autenticazione in cui i 6LBR autenticano sia i nodi 6LR che 6ED. Una volta autenticati, i 6LBR forniscono in maniera sicura con una crittorgrafia ECC una coppia pubblica/privata di chiavi che sarà usata durante la successiva fase. Una seconda fase di generazione di chiave dove la chiave simmetrica sarà comunicata in modalità sicura fra il 6ED e il server remoto. Questa chiave è generata utilizzando il meccanismo ECDH. A causa dei vincoli di risorse dei nodi 6ED, il 6LR gestisce il processo di costituzione delle chiavi per suo conto e quindi le inoltra in modalità sicura al 6ED crittografandole con la coppia di chiavi precondivise. Questo meccanismo, come il precedente, hanno due grossi problemi: richiedono che i nodi M2M memorizzino e gestiscano un grande numero di chiavi; entrambi fanno riferimento sull'assunzione di affidabilità dei 6LR e dei 6LBR. Inoltre in [181] se i proxy collaborano con gli aggressori potrebbero essere in grado di stabilire la chiave simmetrica mentre in [182] sia il 6LR che il 6LBR possono decifrare qualunque

messaggio scambiato fra il nodo 6ED e il server remoto dato che possono calcolare la chiave segreta stabilita.

- **Nicanfar** et al. [183] fanno riferimento all'uso della crittografia basata sull'identità *(Identity Based Cryptography - IBC)*. Differentemente dai meccanismi classici di IBC dove una funzione univoca è applicata all'ID dell'entità per ottenere la chiave pubblica, la proposta fa uso di una funzione F_i e di un codice segreto entrambi dinamici. La chiave pubblica $PubK_i(ID) = F_i(ID)$ mentre la chiave privata è calcolata come $PrvK_i(ID) = s_i F_i(ID)$. Inoltre il PKG periodicamente invia in broadcast una funione f_i che è applicata all'attuale funzione F_i per ottenere la nuova funzione $F_{i+1} = f_i(F_i)$. La chiave segreta s è sostituita da due valori s_i e $\tilde{s}_i$ dove $s_{i+1} = f_{i+1}(s_i)$ e $\tilde{s}_{i+1} = f_{i+1}(s_i) = (a \cdot \tilde{s}_i + b) mod\ q$. Per aggiornare le diverse chiavi propongono tre temporizzatori di breve, medio e lungo periodo. Allo scadere del breve periodo si aggiorna la funzione F_i, nel medio periodo si aggiornano valori di *a* e di *b* mentre nel lungo periodo tutti i valori di segretezza vengono aggiornati. Utilizzando un approccio simile viene fornito un meccanismo di gestione della chiave multidirezionale nel quale la *Sender Multicast Key (SMK) e* la *Receiver Multicast Key (RMK)* sono calcolati grazie al loro nuovo schema IBC come se esse fossero rispettivamente una regolare chiave pubblica e privata. Lo schema quindi propone un meccanismo efficiente sia per gli scambi unidirezionali che multidirezionali. Tuttavia se uno dei pacchetti PKG inviati in broadcast viene perso si creerà un problema di mancata sincronizzazione e nessuna entità potrebbe essere in grado di decifrare i pacchetti.
- Il meccanismo proposto da **Y. Li** [184] è basato su ECC e assume l'esistenza di un *Certificate Agent (CA)* che gestisca la consegna di una coppia di chiave pubblica/privata sia al gateway che ai diversi sensori e dispositivi. Queste chiavi sono usate per abilitare l'autenticazione delle entità in comunicazione e stabilire in modalità sicura una chiave simmetrica fra un nodo o un gruppo di nodi e il gateway. Nello schema q_v e $Q_v=q_vP$ sono rispettivamente la chiave pubblica e privata del gateway *V*. Nella fase iniziale il dispositivo M2M genera casualmente due interi K_u e *r* di *k* e *(160-k)* bit. Quindi calcola $D_u = H(K_u \parallel r)$, $D_u = d_u P$ e $R = d_u Q_v$.

  Una volta calcolati invia $D_u, T = (K_u \parallel r) \oplus R.x\ e\ Sig_u T$ al gateway dove .‖. rappresenta l'operatore di concatenazione. Una volta ricevuti i dati il gateway calcola $R = q_v D_u$ e verifica se il D_u ricevuto eguaglia $H(K_u \parallel r)$. In questo caso genererà un intero a *k* bit C_v e spedirà *(V, C_v, r)* al nodo M2M crittografate con la chiave K_u. Quindi entrambe le parti possono calcolare la chiave della sessione come $KDF(K_u \parallel C_v \parallel U \parallel V) = K_{uv} \parallel session\ K$, dove K_{uv} corrisponde alla chiave MAC per il nodo *U* e il gateway *V.* Lo schema non risulta comunque scalabile in quanto richiede la gestione di un enorme numero di chiavi e certificati.
- Il meccanismo proposto da **Kamto** et al. [185] fa riferimento all'uso dell'algoritmo di scambio delle chiavi *Diffie Hellman (DH).* Il dispositivo M2M e il gateway rispettivamente scelgono un valore casuale x_i e spediscono $x_i = g^{x_i}$ all'altro in maniera tale che la controparte possa calcolare la chiave simmetrica $K_{GW_i} = X_i^{GW} = X_{GW}^{X_i}$. Inoltre propongono un meccanismo di gestione della chiave di gruppo dobe la chiave di gruppo è il prodotto di tutte le chiavi di coppia $K_G = K_{GW_1} \cdot K_{GW_2} \cdot \ldots \cdot K_{GW_n}$. Quest'ultima è calcolata dal gateway e quindi inviata singolarmente a ogni dispositivo in maniera crittografata con la chiave di coppia condivisa. Quando la chiave di gruppo deve essere aggiornata il gateway la calcola nuovamente e la spedisce singolarmente a ciascun nodo. Anche questo schema non è scalabile in quanto il processo con cui viene stabilita e aggiornata la chiave di gruppo richiede l'invio della stessa in modalità unidirezionale ad ogni singolo dispositivo, inoltre è attaccabile con l'aggressione Man-In-The-Middle e sfrutta intensamente le risorse dei dispositivi a seguito dell'implementazione dell'algoritmo DH.
- **I. Doh** et al [186] hanno proposto un meccanismo che permette lo scambio di una chiave simmetrica tra dispositivi M2M che vogliono comunicare direttamento quando sono vicini. I due dispositivi possono essere collegati alla stessa base station *(eNodeB)* o a diversi *eNodeB* ma

devno essere connessi allo stesso *Serving Gateway (SGW).* Nel primo caso è l'*eNodeB* che genera la chiave e la fornisce in maniera sicura a entrambi i nodi crittografata con la chiave che viene utilizzata per le comunicazioni mobili. Nel secondo caso la chiave è generata dal *SGW* e spedita a entrambi i *eNodeBs* che la inoltrano ai relativi dispositivi M2M. Dato che la consegna della chiave condivisa può impiegare molto tempo, specialmente quando i dispositvi sono connessi a diversi *eNodeB,* i dispositivi M2M cominciano a scambiarsi pacchetti crittografati con una chiave temporanea. Appena ricevono la chiave condivisa le chiavi temporanee vengono sostituite da quella condivisa. Anche in questo caso viene affrontato il problema delle comunicazioni di gruppo e fornita la gestione di un meccanismo di chiave di gruppo basato sul *Predistribution and local Collaboration based Group Rekeying (PCGR)* [187]. Nel PCGR, l'*entità di gestione della mobilità (Mobility Managament Entity - MME)* calcola la chiave di gruppo sulla base di condivisioni fornite da diversi dispositivi. Il processo di aggiornamento della chiave provoca un elevato overhead che è stato reso meno pesante modificando lo schema facendo in modo che solo un *eNodeB*that contribuisca a stabilire la chiave. Anche in questo caso il meccanismo non risulta scalabile perché l'aggiornamento della chiave di gruppo richiede lo scambio di un numero elevato di pacchetti. Inoltre può essere implementato solo con dispositivi dotati di *SIM* con chiave associata e non c'è un modo sicuro di scambiare la chiave e prevenire che intercettatori possano scoprirla.

- *Authentication*

  L'autenticazione può riguardare sia l'entità *(entity authentication)* che l'origine dei dati *(data-origin authentication)*. L'*entity authentication* permette alle parti in comunicazione di verificare che l'altro partecipante sia veramente chi si sta dichiarando mentre la *data-origin authentication* assicura che il messaggio è stato generato da una data entità.

  - *Data-origin authentication*

    Uno dei pochi lavori che si occupa di questa problematica è quello di **Bartoli** [188] che cerca di assicurare la *data-origin authentication* attraverso l'aggiunta di un preambolo di autenticazione *(Authentication Preamble - AP)* allo strato fisico. Quindi quando ricevono un pacchetto sia i nodi intermedi che la destinazione finale possono verificare l'origine dello stesso e decidere immediatamente se rigettarlo o meno. L' *AP* è costituito da 32 bit dell'uscita di un messaggio di autenticazione codificato con una funzione hash *(Hash-based Message Authenticated Code - HMAC).* Il primo *AP* è generato grazie a una funzione *HMAC* che ha come ingresso le identità dei due nodi che stanno comunicando in combinazione a una coppia di chiavi condivise. La chiave è derivata usando un *HMAC* da una chiave precondivisa master *MK* inizialmente fornita a tutti i nodi. Si osservi che ogni *AP* è usato una sola volta e i seguenti *AP* sono generati grazie a un *HMAC* avente come ingresso i precedenti *AP.* In questo modo possono essere evitate le aggressioni di tipo *DoS.* Comunque questo metodo ha due difetti dovuti al fatto che *l'AP* non dipende dal contenuto del pacchetto. Quindi un avversario può in maniera legittima scartare un pacchetto e costruirne un altro di sua scelta appendendo l'*AP* catturato. Inoltre richiede che sia distribuita la stessa chiave a tutti i nodi prima che vengano installati. Infine una soluzione del genere può essere usata solo quando si utilizzano nodi stazionari i cui vicini non cambiano nel tempo. Infatti la master key *MK* deve essere cancellata non appena sono calcolate le chiavi necessarie per prevenire che un nodo compromesso possa generare tutte le coppie di chiavi. Quindi i nuovi nodi vicini non possono generare in maniera sicura una chiave condivisa.

  - *Entity Authentication*

    - Nel meccanismo di **Nicanfar** et al. [189] i certificati di autorizzazione *(Certificate Authority)* sono affidati a un server di associazione della sicurezza *(Security Associate - SA)* che calcola la chiave privata del dispositivo applicando una specifica funzione a un valore segreto e a un contatore. Quando un nuovo dispositivo si vuole unire alla rete seleziona uno dei dispositivi già autenticati nominandolo come suo agente di autenticazione (*Authentication*

Agent - AG). L' *AG* selezionato agirà da intermediario fra il nuovo dispositvo e *l'SA.* Inoltre tutti gli scambi relativi al processo di autenticazione sono spediti dal dispositivo all'*AG* crittografati con la sua chiave pubblica. Completata la loro ricezione, l'*AG* le decifra e li ricodifica con la chiave pubblica del *SA* prima di inoltrarli verso l'*SA*. L'*AG* confronta anche i numeri seriali del dispositivo per verificare se quello che ha ricevuto dell'*SA* è consistente con quello spedito al dispositivo. Una volta autenticato, il dispositivo preleva la funzione per essere abilitato a calcolare la chiave pubblica del *SA*. Quindi potrà spedire messaggi cifrati verso l'*SA* e richiedere una chiave privata. Si osservi che con questo schema tutte le chiavi pubbliche e private sono facilmente aggiornate attraverso il semplice broadcast di una nuova funzione come nel lavoro [183]. Il fatto che ogni dispositivo può agire come un *AG* rappresenta il maggior difetto di questo meccanismo. Infatti un dispositivo compromesso può agire da *AG* e può facilmente falsificare l'identità di qualunque altro dispositivo senza essere notato.

- Nel meccanismo proposto da **Chen** et al. [190] vengono fatte le seguenti assunzioni: 1) il fornitore dei servizi M2M *(M2M Service Provider - MSP)* e i dispositivi mobili sono forniti di un insieme di algoritmi di crittografia e 2) l' *MSP*, i dispositivi M2M e i sensori condividono uno spazio iniziale della chiave e alcuni algoritmi di crittografia computazionalmente leggeri. Questi algoritmi sono utilizzati per generare una chiave di crittografia usa e getta *(One-Time Password* - OTP *encryption key)*. Per essere autenticato il dispositivo mobile codifica una richiesta di autenticazione utilizzando una chiave casuale e uno degli algoritmi forniti all'inizio. La richiesta codificata è prima inviata a tutti i nodi sensore vicini che la inoltrano verso l'*MSP*. L'indice dell'algoritmo utilizzato e della chiave di crittografia sono aggiunti alla richiesta. Non appena riceve questo messaggio l'*MSP* risponderà con un messaggio codificato e con una chiave generata in base allo spazio della chiave precondiviso. Appena il dispositivo mobile avrà i parametri necessari spediti dall'*MSP* calcola la chiave simmetrica e completa il processo di autenticazione. Comunque la sicurezza dell'intero sistema può essere messa a repentaglio se uno dei nodi viene compromesso in quanto il suo funzionamento si basa sulla sicurezza dello spazio della chiave precondiviso.
- Il meccanismo proposto da **Nabeel** et al. [191] si basa sulle funzioni materialmente non clonabili *(Physically Unclonable Function - PUF)* che non sono altro che l'equivalente hardware di funzioni non invertibili sia per l'autenticazione sia per la generazione della chiave. Dato che una *PUF* non può essere clonata, nella fase iniziale, il fornitore del servizio dovrebbe avere accesso al dispositivo M2M per recuperare e memorizzare sia il valore hash dell'uscita della *PUF* s_i sia il *Pedersen commitment* [192] del dispositivo Com_i. Il primo, s_i , sarà utilizzato in seguito per autenticare il dispositivo utilizzando il *Schnorr's Zero-knowledge proof of knowledge (ZKPK)* [193]. Una volta autenticato, sia il fornitore del servizio M2M che il dispositivo calcolano una chiave simmetrica come $K=H(s_i, l_i)$ dove l_i è un valore casuale. La chiave *K* sarà utilizzata sia per la riservatezza che per l'integrità dei dati delle successive comunicazioni e può essere facilmente aggiornata attraverso la modifica del valore casuale l_i. È ovvio che l'uso di PUF previene attacchi di falsificazione dell'identità in quanto nessuna entità può calcolare le risposte corrispondenti. Comunque il meccanismo richiede le funzionalità per avere accesso al dispositivo sia prima della sua installazione sia quando il fornitore del servizio eventualmente cambia.

Dato l'elevato numero di dispositivi M2M in una rete se ciascuno effettuasse individualmente l'autenticazione si potrebbe avere un sovraccarico della rete a causa dell'elevato numero di pacchetti di segnalazione. Per evitare che questo accada alcuni lavori di ricerca hanno provato l'implementazione di meccanismi di autenticazione di gruppo. Tali meccanismi comunque fanno riferimento sempre a dispositivi connessi attraverso la rete *Long-Term Evolution (LTE)*.

- **Cao** et al. [194] e **LGTH** [195] cercano di ridurre l'overhead dell'autenticazione fra i dispositivi M2M e i *eNodeB/MME*. Per ottenere questo obiettivo entrambi gli schemi fanno

riferimento all'organizzazione di dispositivi in gruppi e alla selezione di un leader per ogni gruppo. Infatti quest'ultimo riceve richieste di autenticazione firmate dagli altri dispositivi, le aggrega e le inoltra all'*MME.* Dopo la sua ricezione, l'*MME* verifica le firme prima di spedire una risposta firmata al leader del gruppo che la invia in broadcast ai membri del gruppo in modo che ogni nodo possa calcolare la coppia di chiavi stabilita con l'*MME*. I due schemi permettono anche ai dispositivi di autenticare l'*MME.* Il meccanismo di Cao fa uso di una crittografia a chiave simmetrica e richiede un *PKG* per fornire sia ai dispositivi che all'*MME* le loro rispettive chiavi private mentre la soluzione di C. Lai et al è basata su operazioni a chiave simmetrica che possono essere effettuate direttamente sui dispositivi.

- **SE-AKA** [196] cerca di ridurre l'overhead fra l'*MME* e l'*Home Subscriber Server (HSS)* dovuto all'autenticazione di rete. Il primo dispositivo da autenticare compie una procedura di autenticazione completa con l'aiuto del HSS mentre i rimanenti dispositivi sono autenticati localmente dall'*MME* senza che avvenga alcuno scambio di informazioni con l'*HSS.* A questo scopo durante l'autenticazione del primo dispositivo che è basata su *Message Authentication Code (MAC)*, l'*HSS* fornisce all'*MME* tutti i parametri necessari e l'elenco dei dispositivi del gruppo selezionato abilitandolo così alle successive autenticazioni degli altri dispositivi del gruppo. Ottenuta con successo l'autenticazione una coppia di chiavi simmetriche calcolata con il protocollo ECDH è condivisa fra il dispositivo e l'*MME*. Questa chiave sarà usata per i futuri scambi in sicurezza. Comunque questo meccanismo minimizza solo l'overhead fra l'*HSS* e l'*MME* ma non permette a un insieme di dispositivi di essere autenticati simultaneamente.
- ***SEGR*** [197] affronta la riduzione dell'overhead di autenticazione nel caso specifico di dispositvi che effettuano il roaming da una rete *3GPP* a una rete *Worldwide Interoperability for Microwave Access (WiMAX)* e viceversa. Lo schema proposto opera nel seguente modo: primo il centro di generazione della chiave *(Key Generation Center - KGC)* seleziona una chiave privata master *s* e inoltra i parametri di sistema sia al *MME* che al *Wimax Access Service Network Gateway (ASN-Gw)* e fornisce a ogni dispositivo M2M una chiave parziale. Anche tutti i dispositivi selezionano un valore segreto e calcolano le loro chiavi pubbliche. Si noti che i dispositivi sono organizzati in gruppi e per ogni gruppo viene selezionato un nodo leader. Questo nodo gerstirà la consegna di un aggregato di firme non certificate che permetteranno l'autenticazione di tutti i membri del gruppo. Ottenuta l'autenticaziome, una chiave simmetrica ECDH viene stabilita fra i dispositivi M2M e l'*MME.* Anche se questo schema permette la riduzione dell'overhead richiede comunque la gestione di un elevato numero di certificati in quanto è basta sulla crittografia asimmetrica.

- *Privacy*

  Diverse applicazioni M2M gestiscono informazioni sensibili relative a dati personali o di localizzazione. Risulta quindi di primaria importanza preservare le informazioni private dell'utente se si vuole comunque ottenere un massivo impiego delle applicazioni M2M. Inoltre un bilanciamento fra i benefici forniti dalle applicazioni M2M e gli obblighi di riservatezza degli utenti deve essere considerato.

  - **Nicanfar** et al. [198] propongono un sistema che si basa sull'uso di IBC e di pseudonimi in relazione ai sistemi di ricarica degli autoveicoli elettrici. Nel loro schema ogni veicolo elettrico *(Electric Vehicle - EV)* è identificato da un *Smart Grid Server (SGS)* attraverso la sua identità permanente mentre una stazione di ricarica conosce solamente uno pseudonimo dell'*EV* che è aggiornato ogni qualvolta il veicolo si collega a una stazione di ricarica. Per essere ricaricato, l'*EV* inoltra una richiesta firmata grazie alla sua chiave privata momentanea. Questa firma permetterà al *SGS* di verificare l'account corrispondente e informare la stazione di ricarica riguardo alla possibilità di accettare o rifiutare la ricarica. Dato che la stazione di ricarica non può risalire dallo pseudonimo all'identificatore reale dell'*EV* questo meccanismo preserva la

riservatezza dell'utente e permette che la ricarica sia effettuata correttamente. In ogni caso questo meccanismo non protegge la riservatezza dell'utente nei confronti del *SGS*. Inoltre genera un traffico addizionale fra l'*EV* e l'*SGS* per fornire all'*EV* la chiave privata corrispondente al nuovo pseudonimo dato che solo *l'SGS* la può calcolare.

- Il meccanismo proposto da ***Au*** et al. [199] si basa sull'uso delle firme *BBS+* [200], sul *Pedersen's* commitment, e sulla *Zero Knowledge Proof of Knowledge (ZKPK)* [201]. Ogni utente che vuole beneficiare del servizio di ricarica del veicolo elettrico, che è identificato dalla propria *On Board Unit (OBU)* si deve registrare selezionando casualmente alcuni valori per i quali si impegna attraverso il meccanismo di Pedersen. Quindi deposita una quantità di denaro *D* nel suo account e mette una firma *BBS+ σ* sul vettore *(I, D, s, t)* dove *I* identifica univocamente l'*OBU*, *s* è scelto casualmente e *t* permette l'identificazione dell'utente. Invece la firma *BBS+* insieme ai valori *s* e *D* forma un gettone di valore *D$*. Quando il veicolo necessita di una ricarica, l'*OBU* fornisce alla stazione un impegno *(I, D, t)* insieme a un *ZKPK* calcolato correttamente. Dopo aver verificato che sono corretti e che il gettone non è mai stato utilizzato la stazione fornisce all'*OBU* una nuova firma *BBS+* sul vettore *(I, D', s, t)* dove è il bilancio rimanente dell'OBU. Dato che l'*OBU* non rivela direttamenti i valori di *I* e di *t* ma solo un impegno per essi, questi rimangono segreti per la stazione. In questo modo la riservatezza dell'utente è garantita. Inoltre lo schema permette il supporto della tracciabilità quindi abilita la localizzazione dell'utente quando necessario. È importante che questo sarà possibile solo quando l'utente lo autorizza rilevando il valore segreto di *t*. Il maggiore limite di questo meccanismo consiste nell'uso di un appaiamento bilineare che non può essere supportato da dispositivi con risorse vincolate.
- **Popa** et al. [202] affronta il problema della riservatezza della posizione per applicazioni automotive di *Pay As You Drive (PAYD)* e *electronic toll (eToll)*. Il meccanismo invia continuamente le informazioni di localizzazione verso il server. Per assicurare l'anonimato dell'utente si utilizzano identificatori casuali per i quali l'utente produce degli impegni per prevenire successivamente che possa negare che un identificatore gli appartiene.
- Anche **Chen** et al. [203] affrontano il problema precedente ma facendo uso di una firma di gruppo che non solo fornisce l'anonimato al firmatario fra i membri del gruppo ma anche permette l'identificazione del firmatario quando richiesto. Il metodo richiede una computazione molto pesante quando devono essere identificati dei firmatari bari perché l'autorità deve decodificare tutti i vettori di localizzazione e calcolare i conti di ogni utente. In questo modo identifica chi sta barando ma mantiene segreti i vettori di localizzazione. Se si volesse individuare quale delle due entità sta truffando (l'utente o il gestore del servizio) si devono fornire all'autorità l'insieme dei vettori di localizzazione per i quali i pagamenti dell'utente sono più bassi rispetto a quanto richiesto dal gestore del servizio. In questo caso l'autorità verrà a conoscenza della posizione degli utenti (almeno di quelli considerati bari).
- La soluzione di **Troncoso** et al. [204] è basata sulla conversione dei dati raccolti (numero di kilometri percorsi, orario e tipologia di strada) in informazioni di pagamento all'interno di un OBU installato nel veicolo ed utilizzata per la polizza di assicurazione. Nel meccanismo si fa riferimento all'uso di identificativi della polizza e di codice *(ID_{policy}* e *ID_{code})* per rappresentare rispettivamente i tassi della polizza (ad esempio il costo per km percorso) e la versione del software utilizzato. Queste informazioni sono spedite insieme alle informazioni di pagamento e a un identificativo temporale *(timestamp TS)* per evitare aggressioni con messaggi di duplicazione. Si noti che l'*OBU* firma tutti i dati scambiati per assicurare la loro integrità. Per essere sicuri che l'*OBU* non stia agendo in modo malizioso i dati sono memorizzati in maniera crittografata e firmata in una memoria USB. La chiave simmetrica utilizzata per codificare è generata dall'*OBU* e fornita all'assicurato in due diverse condivisioni *($K=K_{S1}+K_{S2}$):* una scritta sulla memoria USB e l'altra consegnata con la ricevuta. In questo modo un utente può verificare quando vuole l'elenco delle informazioni raccolte che sono state utilizzate per

calcolare i dati aggregati. Comunque questa soluzione fa riferimento all'uso di crittografia con chiave pubblica. Quindi richiede la gestione di un elevato numero di chiavi e certificati. Inoltre, finché l'*OBU* non è certificato, la soluzione proposta non previene che un utente malizioso possa modificare i tassi e ingannare il server.

- Alcune soluzioni cercano di proteggere la riservatezza degli utenti di smart meter. **Li** et al. [184] propongono un meccanismo che sfrutta un'architettura di comunicazione ad anello *(Ring Communication Architecture - RCA)* che consiste di diversi smart meter e un aggregatore. Il primo smart meter aggiunge le sue misurazioni a una chiave segreta condivisa con il servizio pubblico, quindi moltiplica il risultato utilizzando il suo *Walsh Code,* effettua uno scorrimento a destra del risultato di un numero casuale prima di spedirlo insieme con l'indice del numero casuale al prossimo nodo. Dopo aver ricevuto l'informazione, il nodo destinazione recupera i valori delle informazioni mascherate attraverso uno scorrimento a sinistra dei dati ricevuti. Quindi aggiunge le proprie misurazioni e fa scorrere a destra tutti i dati utilizzando un ulteriore numero casuale. Il processo continua finché si raggiunge il primo nodo che fa scoorere a destra la metà di destra dell'informazione ricevuta, aggiunge un valore hash al risultato e lo spedisce al nodo aggregatore. Dato che l'aggregatore non conosce il valore segreto condiviso dallo smart meter e dal servizio pubblico e nemmeno il numero di bit per il quale la metà destra è stata fatta scorrere non può risalire alla informazione della misurazione reale. Prima di spedire l'informazione ricevuta al servizio pubblico, alcuni scorrimenti avanti/indietro vengono applicati. In questo modo il servizio pubblico non può identificare a quale smart meter ogni misura fa riferimento. La maggiore limitazione di questo meccanismo consiste nel fatto che è richiesta la distribuzione preventiva di diverse chiavi e due tabelle di indici. Inoltre si può applicare solo nelle applicazioni dove i dispositivi sono statici e vicini. Infine ogni smart meter può modificare i valori delle misurazioni di uno o più smart meter quindi il metodo si può applicare solo se tutti gli smart meter assicurano di essere onesti.
- I meccanismi proposti da **Kalogridis** et al. [205] e da **Varodayan** et al. [206] si basano sulla offuscazione dei dati di consumo attraverso l'aggiunta di batterie al sistema. Infatti in un dato istante e in relazione alle politiche usate la batteria può essere attiva o inattiva. Nel primo caso conserva energia comportandosi come un carico o fornisce energia comportandosi come un generatore. Quando è inattiva non conserva né fornisce energia. In quest'ultimo caso il sistema si comporta come se non c'è alcuna batteria collegata. Dato che lo stato della batteria non è predicibile, i dati di consumo dell'utente possono essere nascosti e gli avversari non possono risalire agli andamenti dei consumi. Questo schema comunque richiede l'uso di una batteria addizionale.

- *Confidentiality*

  Il modo migliore per assicurare la riservatezza dei dati è attraverso la crittografia utilizzando tecniche sia simmetriche che asimmetriche. Mentre la crittografia simmetrica è meno pesante dal punto di vista computazionale e incontra i requisiti legati ai vincoli delle risorse, richiede che entrambe le parti condividano una chiave simmetrica e questo pone il problema di come distribuire tali chiavi. D'altro canto la PKC consuma molte risorse e richiede o il collegamento PKI della chiave pubblica alla corrispondente entità o un PKG che implementi una terza parte garantita in grado di fornire a ogni dispositivo la sua chiave privata. Quindi PKC non è appropriata per essere usata con i dispositivi M2M. Ad ogni modo alcuni studi di ricercatori hanno provato a studiarne la possibile implementazione.

  - **Adiga** et al. [207] hanno verificato la disponibilità di IBC in un ambiente M2M. Per questo motivo hanno effettuato degli esperimenti relativi all'uso di *Identity Based Encryption (IBE)* basati sul *Tate pairing* [208] per mostrare che è possibile utilizzare questo metodo in applicazioni M2M. Comunque gli esperimenti sono stati effettuati su macchine che sono

sensibilmente più potenti di ordinari dispositivi M2M. Inoltre hanno considerato un insieme di 1000 dispositivi mentre il numero atteso di dispositivi M2M è dell'ordine di bilioni.

- **Shih** et al. [209] hanno valutato la disponibilita di utilizzare PKC implementato in hardware per garantire la sicurezza di dispositivi M2M attraverso un *Application Specific Integrated Circuit (ASIC)* che implementa due sistemi di crittografia a chiave pubblica post quantici *(post-quantum public key cryptosystems)* denominati **NTRUEncrypt** [210], un sistema di crittografia a lattice e **TTS** [211], un sistema di crittografia multivariato. Inoltre attraverso diverse implementazioni e analisi delle prestazioni hanno fornito il progetto di un hardware efficiente che supporta le operazioni algebriche richieste dal PKC mantenendo buone prestazioni rispetto ai consumi energetici. Ad ogni modo la disponibilità di hardware in grado di effettuare PKC per assicurare tutte le richieste dei servizi di sicurezza operando con applicazioni M2M deve ancora essere provata.
- **Marin** et al. [212] utilizzano una famiglia ottima di numeri primi, chiamati *shifting primes*, che permettono al chip MSP430 che non possiede un moltiplicatore hardware di eseguire veloci moltiplicazioni. Utilizzando questo metodo il microprocessore MSP430 può d'ora in poi effettuare velocemente moltiplicazioni scalari attraverso operazioni di somma e di scorrimento. Comunque anche se questa soluzione è più efficiente di altre e disponibile per il chip MSP430 richiede 4058 cicli e quindi 817ms per calcolare una moltiplicazione scalare. È necessario quindi avere altri miglioramenti se si vuole utilizzare tale meccanismo con applicazioni che devono soddisfare stringenti requisiti sul ritardo.

- *Integrity*

  Nelle comunicazioni M2M devono essere assicurati sia l'integrità dei dati che dei dispositivi. Inoltre, dato che i dispositivi sono installati in posizioni accessibili, gli avversari o gli utenti maliziosi potrebbero accedervi per eseguire qualunque forma di attacco materiale.

  - **Lu** et al. [213] propongono un meccanismo che previene dalle minacce esterne abilitando il riconoscimento dei nodi compromessi. I vari nodi M2M sono organizzati in coppie che si monitorano l'uno con l'altro in base a messaggi faro *(beacon messages)*. Quando un nodo non riceve più *beacon frames* dal suo vicino per un determinato periodo di tempo lo considera compromesso. Quindi l'aggressione verso un nodo compromesso può essere rilevata velocemente. Inoltre il meccanismo fornisce anche un metodo cooperativo di autenticazione dei dati che sfrutta l'uso di *Bandwidth-Efficient Cooperative Authentication (BECAN)*. Infatti quando un nodo vuole spedire dati ricorre ai suoi vicini per generare un valido MAC. In questo modo l'inserimento di dati e l'alterazione di dati può essere filtrata e i nodi compromessi possono essere rilevati. Comunque questo meccanismo può essere applicato solo nel caso in cui molti nodi sono installati in maniera statica e forniscono lo stesso servizio.
  - Il meccanismo proposto da **Ren** et al. [214] riguarda l'integrità dei dati scambiati e propone uno schema che tenta di garantire la riservatezza e l'integrità mentre si focalizza sull'affidabilità dei dati. Per fornire tale affidabilità fa uso di 4 algoritmi: *Choose-Median, Choose-Most, Choose-Nearest* e *Trust-based enhancement*. Inoltre questi algoritmi abilitano l'identificazione del valore realmente misurato all'interno di un insieme di differenti valori. Comunque nelle comunicazioni M2M i diversi dispositivi non effettuano il monitoraggio dello stesso parametro o evento, quindi questa soluzione non è praticamente utilizzabile.

Riassumendo si può concludere che sfortunatamente, a parte il lavoro [183], tutti gli schemi proposti di gestione della chiave non sono scalabili. Inoltre quasi tutte le soluzioni, ad eccezione di [181], non contemplano dispositivi eterogenei. L'autenticazione dell'origine dei dati è possibile nei dispositivi con risorse vincolate e per le applicazioni sensibili ai ritardi. Comunque può essere usata solo quando si utilizzano dispositivi statici. Le soluzioni riguardanti l'autenticazione delle entità sono nella maggior parte dei casi non scalabili. Inoltre il processo di autenticazione richiede computazioni pesanti anche da parte dei nodi e quindi non si adattano alle risorse vincolate dei dispositivi M2M. In relazione alla riservatezza il

meccanismo in [202] è l'unico che garantisce sia l'efficienza che la possibilità di implementazione in dispositivi con risorse vincolate. Sfortunatamente tutti gli altri schemi che tentano di preservare la riservatezza non riescono a trovare un buon compromesso fra robustezza, ritardo e vincoli delle risorse. La stessa cosa per le proposte relative all'integrità dei dati che non sono appropriati per gli ambienti M2M dove, differentemente dalle WSN, un dato compito è generalmente gestito da un singolo dispositivo.

7.10 Conclusioni

Diversi sono i requisiti di sicurezza per applicazioni M2M. Ciò non di meno occorre tenere in considerazione le limitate risorse computazionali dei mote che non permettono l'implementazione di meccanismi sofisticati. Allo stato attuale dell'attività, in considerazione del compromesso esistente fra prestazioni e sicurezza, si è ritenuto sufficiente l'implementazione di meccanismi di crittografia basati sull'algoritmo AES in modo da rendere sicure le comunicazioni fra i nodi ed evitare l'ascolto da parte di utenti non autorizzati, lasciando a meccanismi di protezione convenzionali (firewall e WPA) il controllo dell'accesso alla rete.

8 Tecniche di ottimizzazione dei consumi energetici nelle WSN

I nodi sensori sono in genere alimentati da batterie la cui sostituzione o ricarica comporta un aggravio dei costi di gestione e una, seppur momentanea, interruzione dell'operatività dei nodi. Inoltre in alcuni contesti la sostituzione delle batterie può essere una operazione non praticabile (si pensi ad esempio al monitoraggio ambientale in zone impervie come pendii o vulcani o a aree industriali con accesso limitato per motivi di sicurezza).

Al fine di massimizzare il tempo di operatività di una rete (lifetime) è quindi di fondamentale importanza minimizzarne i consumi energetici.

In questo Capitolo saranno quindi analizzate le principali tecniche di ottimizzazione dei consumi energetici per WSN note in letteratura.

Per comprendere l'importanza di tali tecniche nelle reti di sensori procediamo con una stima del tempo di vita di un mote in assenza di tecniche di ottimizzazione dei consumi, trascurando l'energia eventualmente assorbita dalle sensorboard (ipotesi lecita per molti tipi di sensori, come ad esempio sensori di temperatura e umidità, ma non per sensorboard contenenti GPS).

Tabella 11 Assorbimenti di corrente e potenza media dissipata da un mote TelosB al 100% di operatività (e tensione di alimentazione pari a 3V).

Unità	**Stato**	**Assorbimento** (mA)	**Potenza** (mW)	**%Tempo**	
CPU (MSP430)	Sleep	0,0051	0,0150	0	
	Attivo	1,8	5,4	100	P_{CPU}=5,4 (mW)
Transceiver (CC2420)	Idle	0,0021	0,0063	0	
	RX	23	69	50	
	TX (0dBm)	17	51	50	P_{TRX}=60 (mW)
					P_{mote}=65,4 (mW)

In generale il tempo di vita di un nodo (T_{life}), ovvero il tempo di operatività del nodo prima che sia necessario sostituire o ricaricare le batterie, può essere stimato uguagliando l'energia disponibile fornita dalle batterie (data dal prodotto fra il numero di batterie e la capacità della singola batteria, $E_{batt} = N_{batt} \cdot C_{batt}$) all'energia necessaria per il funzionamento del nodo (data dal prodotto fra la potenza media dissipata dal nodo e il tempo di operativià del nodo, $E_{mote} = P_{mote} \cdot T_{life}$).

Per cui:

$$T_{life} = \frac{N_{batt} \cdot C_{batt}}{P_{mote}}$$

La potenza media dissipata da un nodo (P_{mote}) dipende da molteplici fattori e in particolare, dalla tensione di alimentazione del nodo, dalle correnti assorbite dalle varie unità che lo compongono (CPU, transceiver, sensori ecc.) e dagli stati di funzionamento (ON/ACTIVE, OFF/SLEEP/IDLE). A titolo di esempio nella Tabella 11 sono riportate le correnti assorbite dal microcontrollore e dal transceiver di un nodo sensore commerciale (TelosB, precedentemente descritto nel Capitolo 4) nelle varie modalità di funzionamento, da cui è stata ricavata la potenza medie assorbita dal nodo (P_{mote}) supponendo che il microcontrollore e il transceiver siano sempre attivi e che il transceiver sia impiegato per il 50% del tempo in trasmissione e per il 50% del tempo in ricezione. Sotto tali ipotesi la potenza media P_{mote} necessaria per il funzionamento del nodo risulta essere pari a circa 65.4mW.

Supponendo quindi il nodo alimentato da due batterie alcaline formato AA da 1,5 Volt della capacità di 2000mAh (ovvero circa 10000J ciascuna) il tempo di vita del nodo risulterebbe essere $T_{life} = N_{batt} \cdot$

$C_{batt}/P_{mote} = 306000$ secondi ovvero circa 85 ore (risultati simili si avrebbero anche con batterie AA ricaricabili NiMH da 1,25V). È evidente quindi che in assenza di apposite tecniche di ottimizzazione dei consumi energetici sarebbe necessaria la sostituzione delle batterie con una cadenza almeno settimanale, intervallo da ritenersi di fatto troppo breve per qualsiasi applicazione industriale.

Esistono in letteratura svariati approcci e diverse tecniche di risparmio energetico nelle WSN. Un interessante survey è proposto in [215] che, come illustrato in Figura 1, permette di classificare le tecniche di risparmio energetico essenzialmente in tre categorie:

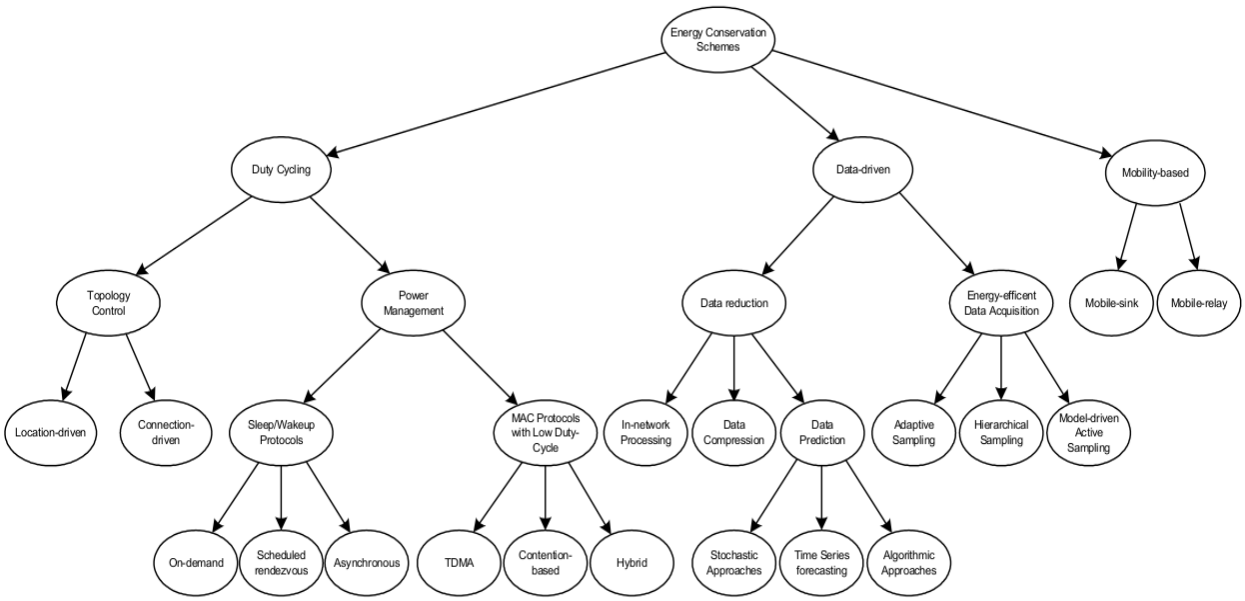


Figura 28 Classificazione tecniche per il risparmio energetico [215].

- **Duty-cycling:** tali tecniche si basano sulla possibilità di mettere i mote in stato di stand-by (o di sleep); in tale stato il consumo di energia dei mote si riduce notevolmente ma occorre sincronizzare i nodi in modo da garantire l'operatività della rete.
- **Data-driven:** tali tecniche sono mirate a ridurre la quantità di dati da trasmettere (es. tecniche di compressione e/o aggregazione dei dati). Tale approccio è limitato dalla capacità computazionale dei mote che non permette l'utilizzo di tecniche di compressione convenzionali (es. zip, rar ecc.).
- **Mobility-based:** tali tecniche si basano sulla possibilità di adottare dei sink mobili per la raccolta dei dati. L'idea di base è quella di far sì che un mote trasferisca i dati al sink solo quando quest'ultimo risulti in prossimità potendo così impiegare delle potenze minori per la trasmissione.

Delle tecniche suddette si ritiene che l'ultima non sia attualmente idonea per applicazioni industriali poiché comporta in genere l'utilizzo di unità complesse e costose (es. droni) oltre che una latenza elevata (legata alla necessità di attendere il passaggio di un sink per acquisire i dati la cui velocità deve essere limitata per motivi fisici e di sicurezza), per cui nell'attività di ricerca ci si è concentrati esclusivamente sulle tecniche di duty-cycling e data-driven.

È importante osservare che per ridurre i consumi energetici è in genere necessario agire sull'hardware o sui protocolli di comunicazione main questa fase e visti gli scopi decisamente più generali del progetto stesso, si è esclusa la possibilità di realizzare hardware dedicato o di procedere con la realizzazione di nuovi protocolli. Per cui l'attività di ricerca si è concentrata sull'utilizzo di piattaforme hardware e di protocolli esistenti determinandone il set di parametri che maggiormente incidono sui consumi energetici e le configurazioni atte a ottimizzare i consumi stessi.

8.1 Duty-cycling

In genere nei mote è il transceiver l'unità con maggiori consumi di potenza. Con riferimento alla Tabella 12 è possibile infatti osservare che la corrente assorbita dal transciever in un TelosB è di un ordine di grandezza superiore rispetto alla corrente assorbita dal microcontrollore; per alcuni mote tale rapporto è addirittura superiore. Pertanto, una notevole riduzione dei consumi energetici può avvenire agendo sul sistema radio. In particolare, agendo sulla programmazione dei nodi e sui parametri dei protocolli di livello MAC (Medium Access Control) è possibile far sì che il transceiver di un nodo si "spenga" (o più correttamente si porti in modalità sleep/stand-by) ogni qual volta non sia necessario né trasmettere né ricevere riducendo così i consumi energetici. È evidente però la necessità di risvegliare il nodo (wake-up) per ricevere e trasmettere i dati.

La percentuale di tempo in cui un transceiver rimane attivo rispetto al tempo totale di funzionamento è detta *duty-cycle* e le tecniche basate su tale approccio prendono il nome di tecniche di *duty-cycling*.

Più precisamente il duty-cycle è definito come

$$\delta = \frac{T_{on}}{T_{on} + T_{off}}$$

dove T_{on} e T_{off} sono gli intervalli di tempo in cui il transceiver è rispettivamente attivo o spento. La somma dei tempi T_{on} e T_{off} è detta *periodo* del duty-cycle.

Ovviamente tanto più è basso il valore del duty-cycle, tanto più grande sarà il tempo di vita di un nodo.

Tabella 12 Assorbimenti di corrente e potenza media dissipata da un mote TelosB con duty-cycle del 1% .

Unità	**Stato**	**Assorbimento** (mA)	**Potenza** (mW)	**%Tempo**	
CPU (MSP430)	Sleep	0,0051	0,0150	99	
	Attivo	1,8	5,4	1	P_{CPU}=0,069 (mW)
Transceiver (CC2420)	Idle	0.0021	0.0063	99	
	RX	23	69	0.05	
	TX (0dBm)	17	51	0.05	P_{TRX}=0,606 (mW)
					P_{mote}=0,675 (mW)

Per fare un esempio concreto, supponendo di utilizzare un mote TelosB con un duty-cycle pari al 1% (ovvero considerando 10ms di attività del mote per ogni secondo trascorso) il tempo di vita passerebbe dalle 85 ore precedentemente stimate (si veda la Tabella 11) ad oltre un anno, come mostrato dai consumi in tabella Tabella 12.

In tale contesto è però di fondamentale importanza comprendere che in una rete di sensori un nodo fa spesso da "ripetitore" (relayer) per altri nodi. Per cui un singolo nodo "spento" può compromettere la trasmissione dei dati da parte di diversi altri nodi. Quindi se da un lato valori di duty-cylcling piccoli sono preferibili per ridurre i consumi energetici dall'altro comportano un aumento della perdita e del ritardo dei pacchetti il che si traduce in una riduzione della capacità di trasmissione (throughput). Tale legame verrà chiarito nel prossimo capitolo.

È evidente quindi la necessità di tecniche e protocolli mirati alla corretta sincronizzazione dei nodi al fine di non compromettere l'operatività dell'intera rete e garantire adeguate prestazioni in termini di ritardo, perdita di pacchetti e capacità di trasmissione.

Le tecniche di duty-cicling sono implementate dai protocolli di livello MAC che possono essere distinti in:

- *Contention-free* (senza contesa)*:* i protocolli contention-free mirano a limitare la possibilità che due

o più nodi trasmettano contemporaneamente (tale circostanza è detta collisione) e si basano su tecniche di accesso multiplo deterministiche a divisione di tempo (TDMA), di frequenza (FDMA) e di codice (CDMA) [216]. Sebbene le tecniche FDMA e CDMA permettano di ottenere migliori prestazioni in termini di latenza e throughput rispetto alle tecniche TDMA-based, queste ultime sono in genere preferite e maggiormente usate nelle WSN per motivi di costo. Esempi di protocolli MAC in questa categoria sono: TRAMA [217], FLAMA [218], LMAC [219], FlexiTP [220], TDMA-ASAP [221]; WirelessHART e ISA100.11a (già discussi nel Capitolo 5) sono esempi di protocolli commerciali che emulano tecniche TDMA per cercare di garantire ritardi limitati (spesso condizione necessaria per le funzioni di attuazione). Essenzialmente negli algoritmi TDMA-based il tempo è diviso in intervalli, detti time-slot, assegnati ai vari nodi. Solo durante il time-slot assegnatogli un nodo esce dalla modalità sleep e può ricevere o trasmettere. Il duty-cycle dipende quindi dalla durata del time-slot. I protocolli MAC basati su TDMA offrono da un lato il vantaggio di ridurre i consumi e permettere di avere ritardi predicibili (nelle ipotesi non sempre realistiche di conoscere la topologia della rete, il numero di nodi e il traffico da essi generato) ma dall'altro hanno anche diversi svantaggi in termini di flessibilità, scalabilità, overhead e complessità: necessitano infatti che tutta la rete condivida lo stessa "riferimento temporale" il che implica la realizzazione di opportuni algoritmi di sincronizzazione; in genere il sink deve assegnare i time-slot ai nodi e tale procedura deve essere ripetuta ad ogni variazione topologica; inoltre l'assegnazione rigida dei time-slot limita le prestazioni della rete tanto che in molti scenari, soprattutto per bassa intensità di traffico, le tecniche a contesa (di seguito illustrate) risultano avere prestazioni superiori anche in termini di ritardo e throughput.

- *Contention-based* (con contesa)*:* i protocolli a contesa sono basati sulla tecnica di accesso CSMA (Carrier Sense Multiple Access) e possono essere distinti in sincroni e asincroni. I protocolli di tipo *sincrono* mirano a sincronizzare i nodi in modo da garantire che, nonostante l'impiego di tecniche di duty-cylcing, trasmettitore e ricevitore siano accesi contemporaneamente. Esempi di protocolli per WSN in tale categoria sono SMAC, TMAC, DMAC, ZMAC, WiseMAC, SyncWUF. In tali protocolli i nodi si scambiano periodicamente pacchetti per la sincronizzazione (SYNC) al fine di negoziare la schedulazione delle attività. Tali pacchetti però consumano risorse in termini di energia, banda e capacità di elaborazione. Inoltre una perfetta sincronizzazione è difficilmente ottenibile a causa del drift (deriva/disallineamento) del clock dei dispositivi spesso trascurato nelle simulazioni e che porta ad una notevole differenza tra la teoria e la pratica nelle prestazioni di tali algoritmi. Nei protocolli *asincroni* un nodo si accende periodicamente e indipendentemente dagli altri nodi per ascoltare il canale e rilevare la presenza di pacchetti a lui destinati. La maggior parte dei protocolli *asincroni* prevedono che un nodo che voglia iniziare una trasmissione mandi un preambolo e rimanga in attesa di una risposta da parte del nodo ricevente. Esempi di tali protocolli sono BMAC [222], XMAC, BoxMAC [223] (versione 1 e 2, utilizzato da TinyOS), ContikiMAC (utilizzato dal S.O. Contiki) e MXMAC. Esistono però altri approcci basati ad esempio sull'utilizzo di un canale distinto per indicare ai nodi quando svegliarsi (es. STEM); rispetto all'utilizzo di un preambolo tale approccio ha lo svantaggio di aumentare i costi del transceiver. Altri algoritmi sfruttano la possibilità che sia il ricevitore a segnalare l'inizio di una comunicazione (es. RI-MAC). Tale approccio permette di ridurre i consumi energetici dovuti alla trasmissione del preambolo ma rende poco predicibili i ritardi. Per un recente survey sui protocolli MAC asincroni si rimanda a [224]

Le tecniche con contesa asincrone sono in genere più semplici, non richiedendo funzioni di sincronizzazione e/o informazioni sulla topologia della rete, per cui possono essere facilmente implementate in mote low-end; sono inoltre altamente scalabili (si adattano automaticamente al traffico di rete e alla topologia senza dover limitare a priori il numero di nodi). D'altra parte però tali algoritmi hanno come principale limite l'impossibilità di garantire un ritardo massimo a causa delle possibili (ed imprevedibili) collisioni fra pacchetti. È proprio questo il principale scoglio da superare per estendere le reti WSN ai sistemi di attuazione e più in generale ai sistemi di controllo real-time.

Nell'ambito dell'attività di ricerca si sono quindi confrontati due protocolli che rappresentato lo stato dell'arte nelle rispettive tipologie di protocolli contention-based, specificatamente:

- l'IEEE802.15.4/ZigBee [80] (in modalità beacon-enabled) in rappresentanza dei protocolli a contesa sincroni;
- BoXMAC [223] in rappresentanza dei protocolli a contesa asincroni.

Tale confronto ha portato alla conclusione che l'utilizzo di BoXMAC (e più in generale dei protocolli asincroni) sia da ritenersi più congeniale per gli ambiti di applicazione identificati dal progetto MiSE-ENEA per i seguenti motivi:

1) La gestione del duty-cycle da parte dell'IEEE802.15.4 prevede la presenza di nodi sempre attivi (detti FFD, Fully Functional Devices) allo scopo di trasmettere appositi segnali (beacon) necessari per sincronizzare la rete. L'uso di FFD ha i seguenti svantaggi:
   - aumento dei consumi energetici: gli FFD, dovendo essere sempre attivi, non possono utilizzare tecniche di duty-cycling; per garantire un adeguato tempo di operatività della WSN occorre quindi connettere tali dispositivi alla rete di alimentazione;
   - aumento dei costi: per fornire agli FFD l'alimentazione di rete occorre la posa di appositi cavi di alimentazione oltre che l'utilizzo di trasformatori;
   - degradazione della robustezza: nel caso di problemi alla rete di alimentazione tutti i nodi della WSN risulterebbero non raggiungibili.

   Solo nel caso di reti di piccola estensione, considerando una rete con topologia a stella, il numero di FFD potrebbe essere limitato ad uno (rendendo marginali gli svantaggi suddetti) ma ciò contraddice uno dei requisiti del progetto ovvero la scalabilità. Per estendere le dimensioni della rete si ricorre quindi a topologie di tipo cluster-tree. In tali topologie i nodi sono divisi in gruppi (detti cluster) ciascuno facente capo ad un FFD, gli FFD sono connessi al sink con una topologia ad albero (si veda la Figura 29 Topologie WSN.

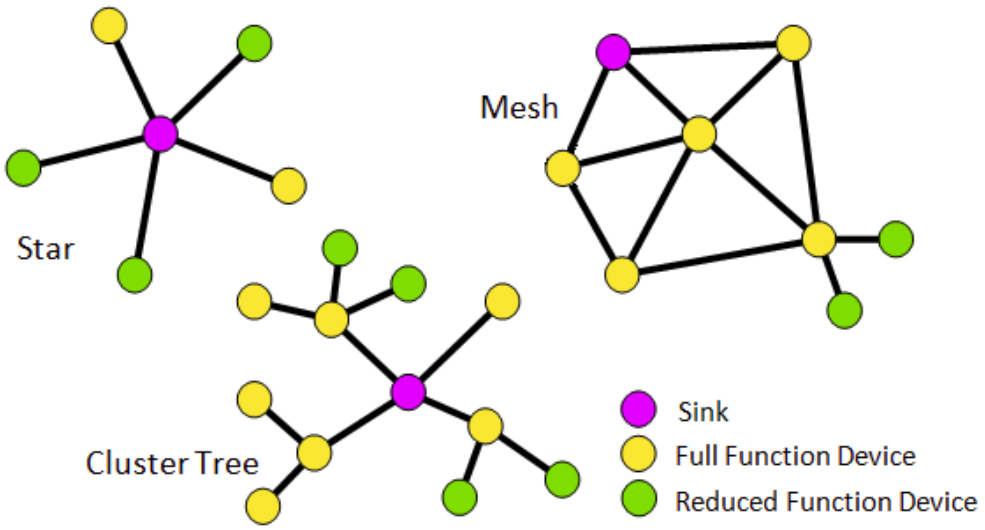


Figura 29 Topologie WSN

2) In topologie cluster-tree gli FFD devono essere coordinati; in particolare per ogni ramo la somma dei duty-cycle degli FFD deve essere inferiore ad 1 onde evitare la sovrapposizione dei segnali di beacon. Tale condizione può essere ottenuta specificando opportunamente in fase di inizializzazione della rete i parametri di funzionamento del protocollo 802.15.4, in particolare il Beacon Order (BO) e il Superframe

Order (SO) da cui dipende il duty-cycle secondo la relazione $\delta = 2^{(SO-BO)}$, ma tale vincolo rende problematica l'estensione e la riconfigurazione della rete (e costituisce un limite all'utilizzo dei protocolli basati sul MAC 802.15.4 quali ZigBee, WirelessHART e ISA100.11a).

Le problematiche suddette possono essere superate con l'impiego di topologie mesh e protocolli asincroni. L'attività di ricerca si è quindi concentrata sui protocolli asincroni analizzandone lo stato dell'arte e cercando di determinarne i limiti prestazionali sia con modelli analitici che con misure sperimentali. In particolare l'attività ha riguardato la scelta del duty-cycle ottimale al fine di minimizzare i consumi energetici garantendo ritardi e reliability.

Di seguito si riporta una breve descrizione dei principali protocolli asincroni per WSN noti in letteratura al fine di evidenziarne le caratteristiche salienti; seguono le considerazioni che hanno portato alla scelta del protocollo MAC ritenuto più idoneo per il progetto.

- B-MAC [222] (Berkley-MAC): è uno dei primi e dei più semplici protocolli asincroni. Tale protocollo prevede che in tutti i nodi il transceiver si risvegli periodicamente con periodo T; il trasmettitore prima di inviare il pacchetto testa il canale, distinguendo due casi: 1) il canale è libero, in tal caso invia un preambolo poco più lungo del periodo T; ciò garantisce che il preambolo sia rilevato dal ricevitore indipendentemente dall'istante in cui quest'ultimo si risveglia; 2) il canale è occupato, in tal caso la trasmissione viene ripetuta dopo un intervallo casuale (algoritmo di back-off), così da evitare che i nodi trasmettano contemporaneamente, garantendo una buona protezione dalle collisioni.
- X-MAC [225]: in questo protocollo si sostituisce il lungo preambolo del BMAC con un treno di preamboli più brevi (in pratica l'header del pacchetto) così da ridurre i consumi. Il ricevitore che rileva uno di questi preamboli risponde subito con un ACK al trasmettitore che passerà ad inviare direttamente il pacchetto, questo accorgimento permette di ridurre l'occupazione del canale. Inoltre i nodi che rilevano il preambolo ma non sono destinatari del pacchetto possono immediatamente ritornare in stato di sleep (evitando l'overhearing).
- BoX-MAC-v2 [223] (utilizzato dal S.O. TinyOS e noto anche come LPL): in questo protocollo, i preamboli brevi di XMAC sono costituiti dall'intero pacchetto questo permette di ridurre i tempi di trasmissione (anche del 50%) visto che un nodo riceve il pacchetto non appena si sveglia; inoltre in ricezione per rilevare la presenza dei premabolo si misura l'energia del segnale ricevuto (non è necessario decodificare il pacchetto) e ciò riduce notevolmente il tempo in cui i nodi rimangono in ascolto (e di conseguenza i consumi energetici, fino al 50% rispetto a XMAC).
- ContikiMAC [226] (utilizzato dal S.O. Contiki): rispetto a BoX-MAC si ha un l'aggiunta di un meccanismo di sincronizzazione (phase-locking) tramite il quale un nodo trasmittente può "apprendere" le temporizzazioni del nodo ricevente osservando gli istanti di trasmissione degli ACK. Sapendo quando il nodo ricevente si risveglierà è possibile ridurre il numero di preamboli trasmessi e quindi ridurre i consumi energetici e l'occupazione del canale.

Sebbene fra i protocolli suddetti ContikiMAC risulti essere il più recente e anche il più efficiente in quei contesti in cui si hanno solo attività di sensing omogenee, esso richiede che tutti i nodi abbiano lo stesso periodo di trasmissione (rete isocrona) o, in alternativa, che il trasmettitore venga pre-programmato con una tabella contenente il periodo di trasmissione di tutti i possibili ricevitori. L'ipotesi di rete isocrona non è in genere verificata nelle reti oggetto del progetto che prevedono una elevata estensione oltre che sensori e attuatori di varia natura; inoltre ricorrere a tabelle pre-programmate, oltre che essere una soluzione poco flessibile, comporterebbe un minore spazio di memoria disponibile per le applicazioni. Si è ritenuto quindi di dover escludere l'utilizzo di ContikiMAC e quindi di scegliere Box-MAC quale protocollo di livello MAC per l'implementazione della WSN oggetto del progetto. Si è in particolare optato per Box-MAC-v2 perché rispetto alla v1 risulta immune ad attacchi di tipo Denial of Sleep (con BoxMac-v1 un hacker potrebbe far esaurire la batterie dei nodi sensori semplicemente sfruttando una trasmissione continua da parte di un nodo).

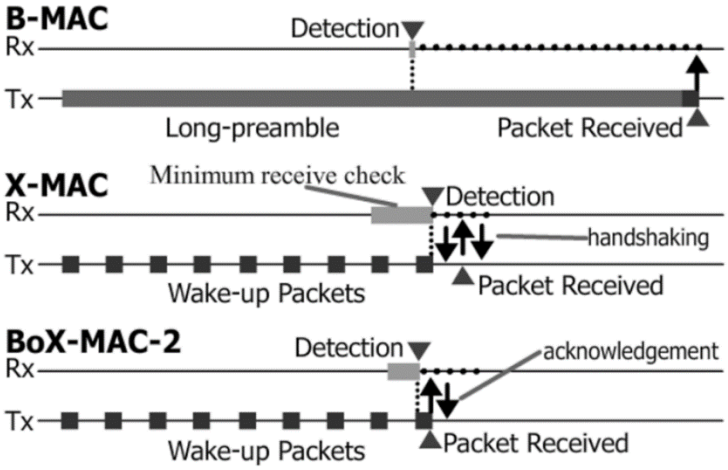


Figura 30 Confronto tra i protocolli X-MAC, Box-Mac2 e B-MAC.

In BoX-MAC-2 i principali parametri che determinano il duty-cycle sono i seguenti:

- *LPL_SLEEP_INTERVAL* (espresso in millisecondi): definisce il periodo del duty-cycle (in seguito indicato con T_{LPL}) e di default è fissato a 512ms. Può essere impostato mediante direttive di compilazione (-D) e/o appositi flag nei file di make (CGLAG). Come sarà meglio chiarito nel Capitolo 9 aumentare il valore di questo parametro permette di ridurre i consumi energetici ma a scapito del ritardo di trasmissione.
- *DELAY_AFTER_RECEIVE* (espresso in ms): tale parametro indica per quanto tempo il ricevitore rimane attivo dopo la ricezione di un pacchetto. Può essere impostato mediante direttive di compilazione (-D) e/o appositi flag nei file di make (CGLAG). Il valore di default è di 100ms (definito nel file di header tos/types/Lpl.h). Il fatto che il ricevitore rimanga attivo dopo la ricezione permette di trasmettere più pacchetti consecutivamente (burst) senza dover attendere un ciclo per ogni pacchetto. Come sarà meglio chiarito nel Capitolo 9 aumentare il valore di questo parametro permette di aumentare il throughput ma a scapito di un maggiore consumo energetico. In genere il tempo di elaborazione di un pacchetto è trascurabile rispetto a questo valore per cui il parametro *DELAY_AFTER_RECEIVE* determina il valore di T_{on} a seguito della ricezione di un pacchetto.
- *MAX_LPL_CCA_CHECKS* (espreso in jiffies ovvero periodi di clock): per i dispositivi basati sul transceiver CC2420 (es. telosb e micaz) determina il tempo (in seguito indicato con T_{CCA}) per cui viene testato il canale al fine di sapere se è libero o se vi sono pacchetti. In pratica tale valore determina il minimo valore di T_{on}. Più precisamente il parametro specifica quante volte viene testato il pin CCA del transceiver che segnala l'assenza o la presenza di segnale sul canale e di default è fissato a 400 (il valore è definito nel file di header tos/chips/cc2420/lpl/DefaultLpl.h).

In condizioni di basso traffico di rete, ovvero se la ricezione di pacchetti avviene con un intervallo decisamente superiore al tempo specificato dal parametro LPL_SLEEP_INTERVAL, situazione tipica nel caso di smart metering, il duty-cycle è quindi dato da

$$\delta = T_{CCA}/T_{LPL}$$

Sulla base di quanto detto sulle tecniche di duty-cycling è possibile quindi affermare che ridurre il valore di T_{CCA} (e quindi di MAX_LPL_CCA_CHECKS) permette di ridurre i consumi energetici.

Il parametro MAX_LPL_CCA_CHECKS non influenza però solo i consumi ma anche la probabilità di perdita di pacchetti.

Vista l'importanza del parametro *MAX_LPL_CCA_CHECKS* si è ritenuto quindi di dover approfondire, sia con misure sperimentali sia con analisi teoriche, al fine di determinarne il valore ottimale.

In teoria al valore di default 400 dovrebbe corrispondere un tempo T_{CCA} pari a 12.2ms. Questo perché il controllo del pin CCA (almeno nel TelosB) è cadenzato da un oscillatore a 32768Hz per cui 400 periodi di

clock (o jiffies) corrispondono a 400/32768=12.2ms. Il tempo indicato dalla variabile DUTY_ON_TIME nel file DefaultLpl.h è però di 11ms e ciò costituisce una discrepanza con il valore teorico atteso.

Si è quindi proceduto con misure sperimentali.

Nell'ambito dell'attività è stato realizzato un testbed utile per verificare i parametri suddetti oltre che per determinare i consumi energetici di un nodo nelle varie fasi del suo funzionamento.

In Figura 31 è riportato il setup di misura realizzato.

Ai fini della misura della corrente istantanea è stata posta in serie al pacco batterie del mote una resistenza di sensing (R_s) del valore nominale di 10 Ohm (scelto in modo da poter considerare trascurabile la caduta di tensione sulla resistenza in relazione alla tensione di alimentazione). Il valore esatto della resistenza, pari a 10.12 Ohm, è stato misurato con un multimetro Keysight 34461A. Successivamente per tramite di un oscilloscopio Keysight MSOX3012 è stata misurata la tensione ai capi della resistenza di sensing (V_s) e la tensione ai capi delle batterie (V_{batt}). Le misure possono essere trasferite ad un PC in tempo reale ed importate in Matlab [227] per una successiva elaborazione.

In particolare a partire dai valori suddetti possono essere determinate la corrente istantanea assorbita dal mote e la potenza istantanea per tramite delle relazioni

$$I_{mote} = \frac{V_s}{R_s} \quad \text{e} \quad P_{mote} = V_{batt} \cdot \frac{V_s}{R_s}$$

Lo stesso setup di misura ha permesso di misurare il valore reale di T_{CCA} al variare del parametro *MAX_LPL_CCA_CHECKS.*

La Figura 32 mostra l'andamento temporale della corrente assorbita dal mote al variare del parametro *MAX_LPL_CCA_CHECKS*.

Come è possibile osservare in tutte le misure vi è un transitorio iniziale della durata di circa 3ms, successivamente la corrente rimane stabile per un tempo che dipende dal parametro *MAX_LPL_CCA_CHECKS* a seguito del quale si ha un transitorio di discesa della durata di circa 1.1ms. I transitori fanno sì che la misura del T_{CCA} non sia univoca ma dipenda dalla soglia selezionata per identificare l'inizio o la fine dei transitori.

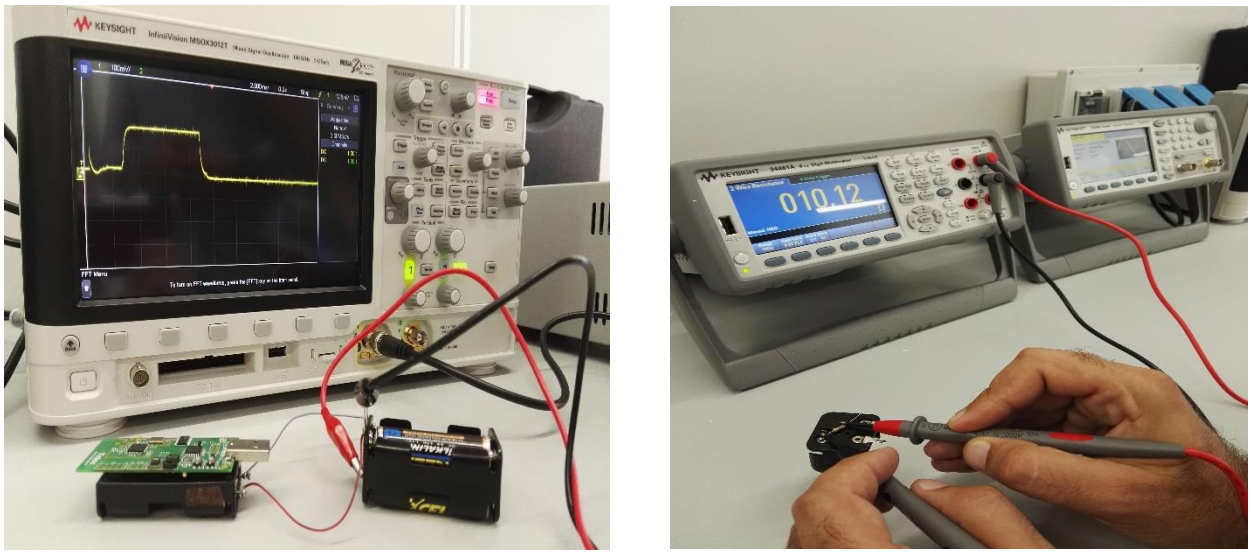


Figura 31 Set-up di misura (a sinistra) e preventiva misura della resistenza di sensing (a destra).

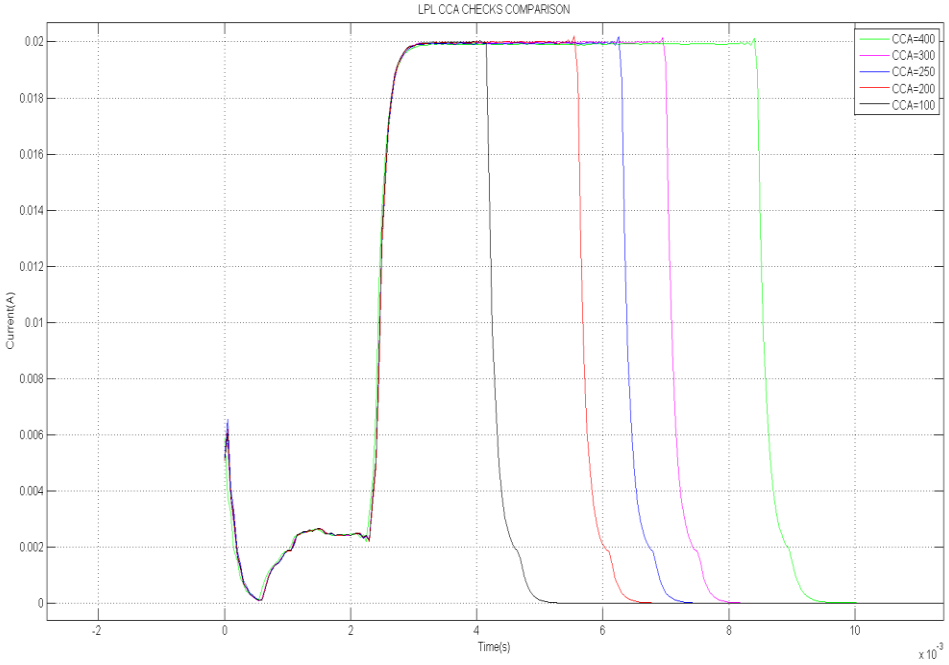


Figura 32 assorbimento di corrente di un telosB al variare del parametro MAX_LPL_CCA_CHECKS.

La Tabella 13 riporta i valori di E_{CCA} misurati al variare del parametro *MAX_LPL_CCA_CHECKS* per due soglie di riferimento (pari al 50% del valore massimo e al 95% del valore massimo) oltre che l'energia complessiva dissipata dal mote per l'ascolto del canale (E_{CCA}), determinata moltiplicando i valori della potenza istantanea misurata nell'intervallo T_{CCA} per il tempo di campionamento.

Tabella 13 valori di T_{CCA} e di E_{CCA} al variare di MAX_LPL_CCA_CHECKS.

MAX_LPL_CCA_CHECKS	T_{CCA} (ms) (soglia @50%, 10mA)	T_{CCA} (ms) (soglia @95%, 19mA)	E_{CCA} (uJ)
400	6,08	5,30	377
300	4,63	3,85	290
200	3,22	2,47	205
100	1,79	1.05	119

Dalla tabella è possibile osservare che l'energia dissipata per impulso è proporzionale al valore del parametro MAX_LPL_CCA_CHECKS. In particolare se espressa in µJ l'energia dissipata per il solo ascolto del canale è circa pari al valore del parametro MAX_LPL_CCA_CHECKS. Pur trattandosi di una coincidenza tale risultato fornisce una semplice formula in grado di determinare un limite superiore al tempo di vita di un nodo:

$$T_{life} < \frac{E_{batt}}{E_{CCA}} \cdot T_{LPL} \approx 5555 \cdot \frac{\text{LPL\_SLEEP\_INTERVAL [ms]}}{\text{MAX\_LPL\_CCA\_CHECKS}} [\text{ ore }]$$

A titolo di esempio con un LPL_SLEEP_INETERVAL di 512 millisecondi e MAX_LPL_CCA_CHECKS pari a 400 il tempo di vita massimo di un nodo non potrà essere superiore alle 7111 ore (ovvero circa 300 giorni). Un modello più accurato per la stima del tempo di vita sarà presentato nel Capitolo 9.

Il parametro MAX_LPL_CCA_CHECKS non influenza solo i consumi ma anche la probabilità di perdita di pacchetti.

Sulla base di [223] per garantire la corretta rilevazione dei pacchetti il valore di T_{CCA} deve essere superiore alla somma del tempo necessario per la trasmissione dell'ACK e del tempo di backoff.

Il tempo necessario per trasmettere l'ACK sulla base dello standard 802.15.4 è pari a 544us (192us tempo di turnaround fra la fine di un pacchetto e l'inizio di un altro e 352us per trasmettere gli 11 byte dell'ACK comprensivi dell'header di livello fisico).

Per quanto concerne invece il tempo di backoff, secondo lo standard 802.15.4 è un valore casuale scelto fra un minimo di 320us (aUnitBackoff pari alla durate di 20 simboli, ovvero 10byte) e un valore massimo che varia nel tempo in funzione del numero di ritrasmissioni effettuate dal nodo; sempre secondo lo standard 802.15.4 il valore massimo alla prima trasmissione è 2.24ms (BE=3) mentre il massimo assoluto è 9.92ms (BE=5).

Analizzando l'implementazione di TinyOS del LPL, e in particolare il codice tos/chips/cc2420/lpl/PowerCycleP.nc, il valore massimo per il backoff è invece di 50tick (circa 1.53ms).

In realtà ai tempi suddetti occorre anche aggiungere i ritardi legati al microcontrollore (per l'esecuzione del codice necessario per testare il segnale CCA) ed al transitorio del VCO del transceiver, che in genere sono complessivamente al di sotto di 1ms.

Sulla base dei valori suddetti il valore di T_{CCA} non può essere al di sotto di 3ms (544µs+1.53ms+1ms) da cui la necessità di imporre un valore di MAX_LPL_CCA_CHECKS superiore a 250.

Tale ipotesi è stata oggetto di verifica sperimentale.

La Tabella 14 riporta la probabilità di consegna dei pacchetti (Packet Delivery Ratio, PDR) al variare del parametro MAX_LPL_CCA_CHECKS ottenuta sperimentalmente considerando la trasmissione di 10000 pacchetti fra due TelosB posti alla distanza di 3m.

Tabella 14 Packet Delivery Ratio, PDR) al variare del parametro MAX_LPL_CCA_CHECKS.

MAX_LPL_CCA_CHECKS	PDR
400	100%
300	100%
250	92%
200	88%

I valori ottenuti sono in accordo con i risultati teorici.

Sulla base dello studio suddetto il valore di MAX_LPL_CCA_CHECKS può essere ridotto a 300, con una conseguente riduzione dei consumi energetici del 30% rispetto all'utilizzo del valore di default pur garantendo una adeguata reliability.

Il testbed realizzato ha quindi permesso di determinare sperimentalmente la migliore configurazione del set di parametri inerenti il duty-cycle.

8.2 Data-reduction

I consumi energetici dipendono principalmente dal tempo in cui il transceiver di un mote rimane nello stato attivo per trasmettere o ricevere pacchetti. Occorre infatti tenere presente che la trasmissione di un singolo bit richiede la stessa energia di 3000 istruzioni [228]. È evidente quindi che è possibile ridurre i consumi energetici riducendo la quantità di dati da trasmettere. Su tale considerazione si basano le tecniche di data-reduction che si rilevano particolarmente efficaci quando il fenomeno da monitorare è lentamente variabile o più in generale quando i dati da trasmettere sono correlati temporalmente e/o spazialmente.

Per ridurre la quantità di dati da trasmettere è possibile adottare diverse tecniche [215] che possono essere classificate come segue:

- Tecniche di acquisizione: mirate ad ottimizzare la frequenza di campionamento al fine di ridurre il numero di dati da inviare nell'unità di tempo.
- Tecniche di compressione locali: permettono di ridurre la quantità di dati da trasmettere sfruttando la correlazione temporale del fenomeno fisico da monitorare.
- Tecniche di compressione distribuite: permettono di ridurre la quantità di dati da trasmettere sfruttando la correlazione spaziale del fenomeno fisico da monitorare.
- Tecniche di aggregazione (data aggregation e in-network processing).

8.2.1 Tecniche di acquisizione

La quantità di dati da acquisire e trasferire per unità di tempo dipende dalla rapidità di variazione della grandezza fisica che si sta monitorando. In particolare il teorema di Nyquist [229] fissa un limite inferiore specificando che per non avere perdita di informazione a seguito del campionamento di un segnale con frequenza massima f_m, la frequenza di campionamento deve essere almeno pari a $2f_m$.

È evidente che la frequenza di campionamento può differire di diversi ordini di grandezza a seconda della grandezza fisica da monitorare: se per grandezze fisiche ambientali come la temperatura, l'umidità o la pressione atmosferica può essere ragionevole anche un campione ogni trenta secondi, nel caso del monitoraggio real-time di sistemi vibranti potrebbe essere necessaria una frequenza dell'ordine delle decine di campioni al secondo.

È quindi indispensabile poter fissare tale parametro in accordo alle grandezze misurate. Sebbene esistano delle tecniche in grado di adattare automaticamente la frequenza di campionamento alla grandezza misurata (tecniche di adaptative-sampling [215]) tali tecniche si basano sull'esistenza di un modello (non sempre disponibile e/o computazionalmente compatibile con le risorse dei mote) o sullo scambio di informazioni fra i nodi con conseguente aumento del traffico di rete. Nell'ambito dell'attività si è quindi ritenuto sufficiente permettere una configurazione manuale delle frequenza di campionamento mediante apposite interfacce grafiche. In particolare il firmware sviluppato permette di configurare tale parametro da remoto mediante l'invio di appositi pacchetti di inizializzazione (set-up).

8.2.2 Tecniche di compressione locale

Le tecniche di compressione locale sfruttano la correlazione temporale dei dati per ridurre il numero di bit necessario per rappresentare le singole misure. La correlazione temporale, osservabile in molti fenomeni fisici, implica che i valori delle differenze fra campioni consecutivi ($r_i = x_i - x_{i-1}$, detti residui) sono più piccoli rispetto ai valori dei campioni originari e possono quindi essere codificati con un numero inferiore di bit.

L'efficacia di una tecnica di compressione può essere misurata per tramite del rapporto di compressione r_c definito come

$$r_c = 1 - \frac{B_c}{N \cdot w}$$

Dove *N* è il numero di misure/campioni da comprimere, *w* è il numero di bit per campione (prima della compressione) e B_c è il numero di bit necessari per rappresentare le *N* misure dopo la compressione. Ovviamente una tecnica di compressione è tanto più efficace quanto maggiore è il rapporto di compressione.

In generale le tecniche di compressione possono essere classificate in tecniche di compressione con perdita (*lossy*) e tecniche di compressione senza perdita (*lossless*). Sebbene le prime permettano di raggiungere

rapporti di compressione maggiori, non sono state considerate idonee per il progetto in esame per i seguenti motivi:

1. le tecniche di compressione con perdita non permettono di ricostruire fedelmente l'informazione originaria. Per molte applicazioni di telemetria (smart-meter) o di telecontrollo (smart-building) una non perfetta ricostruzione dell'informazione potrebbe compromettere irreparabilmente le operazioni di controllo e di fatturazione;
2. gli algoritmi lossy sono computazionalmente onerosi e difficilmente potrebbero essere eseguiti su dispositivi a microcontrollore;
3. gli algoritmi di compressione con perdita devono essere necessariamente studiati, implementati e tarati per la specifica tipologia di segnale fisico da comprimere del quale devono essere a priori note le caratteristiche statistiche (ad esempio una tecnica di compressione lossy per un segnale audio non si adatta alla compressione di un segnale video e viceversa).

Queste considerazioni, insieme all'intenzione di realizzare un nodo sensore che sia il più possibile general-purpose (in grado cioè di acquisire e trasmettere qualunque tipo di segnale fisico) hanno spinto la nostra attività di ricerca a considerare esclusivamente tecniche di compressione senza perdita.

Si ritiene opportuno a questo punto evidenziare che sebbene esistano diversi algoritmi e formati di compressione dati senza perdita (zip e rar, giusto per citarne alcuni) tali algoritmi non sono implementabili nelle reti di sensori a causa delle limitate risorse computazionali dei mote. È quindi necessario rivolgere l'attenzione a tecniche specifiche per reti di sensori che tengano conto espressamente in considerazione la capacità computazionale a disposizione nei mote.

Un recente survey sulle tecniche di compressione senza perdita per reti di sensori è stato proposto dagli stessi autori in [230].

Tabella 15 Dizionari utilizzati da alcuni algoritmi di compressione.

r_i	c_i (ND-Encoding)	c_i (SHuffman)
0	00	00
+1,-1	01+index (1bit)	010+index (1bit)
+3,+2,-2,-3	10+index (2bit)	011+index (2bit)
+5,+4,-4-,-5	110+index (2bit)	100+index (3bit)
+7,+6,-6,-7	1110+index (2bit)	100+index (3bit)
others	1111+valore orginario (wbit)	variable prefix+ variable index

Le tecniche di compressione locali senza perdita prevedono due approcci:

- *dictionary-based*: basate su un dizionario utilizzato per codificare efficacemente i residui r_i con parole di codice c_i. Le parole di codice sono in genere realizzate dalla concatenazione di un prefisso s_i e di un indice $a_i (c_i = [s_i | a_i])$, e sono scelte con l'obiettivo di assegnare a valori più probabili codici di lunghezza inferiore. In Tabella 15 sono illustrati i dizionari dell'algoritmo ND-encoding [231] e dell'algoritmo Huffman [232]. Tali tecniche hanno il vantaggio di essere computazionalmente semplici ma permettono di ottenere buone prestazioni in termini di rapporto di compressione solo se i residui sono piccoli ovvero per lo più concentrati nell'intervallo [-7,7].
- *predicition-based*: mediante l'utilizzo di tecniche di predizione lineare determinano il dizionario in maniera adattativa. Un esempio è dato dalla tecnica TwoModal proposta in [233]. Lo svantaggio di tali algoritmi risiede nel fatto che l'algoritmo di predizione, a causa delle limitate risorse computazionali dei mote, può essere eseguito solo dal sink; il dizionario deve quindi essere

inoltrato ai nodi sensori con conseguente aumento del traffico di rete. Occorre poi dimensionare opportunamente il periodo di aggiornamento del dizionario (ovvero stabilire ogni quanto eseguire la fase di training).

In [230] è stato proposto un nuovo algoritmo denominato MinDiff che combina i due approcci e adotta dinamicamente il dizionario al range dei residui per tramite di una semplice predizione locale da parte dei nodi, evitando quindi l'intervento del sink e l'overhead necessario per la trasmissione del dizionario da parte di quest'ultimo.

Lo pseudo-codice dell'algoritmo MinDiff è il seguente:

INPUT: set di N campioni $X=\{x_i : i=1..N\}$

OUTPUT: codifica degli N campioni, $C=\{c_i : i=1..N\}$ + parametri K, μ ,FLAG

ALGORITMO MinDiff:

Fase I (Calcoli preliminari)

- residui $r_i=x_i-x_{i-1}$
- minimo dei residui, $m=\min\{r_i\}$
- range dei residui, $R=\max\{r_i\}-\min\{r_i\}$
- parametro $K=\text{ceil}\{\log_2(R+1)\}$
- parametro S_3 = numero di residui il cui modulo è inferiore o uguale a 3
- parametro FLAG = risultato confronto $S_3 \cdot (K-1) > 2N$

  (FLAG=1 se il confronto è positivo, altrimenti FLAG=0)

Fase II (Codifica):

Per $i=1$, $c_i = x_i$ (codificato con w bit)

Per $i>1$, codicica dipendente dal valore del parametro FLAG

Caso A) FLAG==1 => residuo codificato con il valore $c_i = r_i - \mu$ (unsigned rappresentato con K bit)

Caso B) FLAG==0 => residuo codificato secondo il dizionario seguente

r_i	c_i (MinDiff*)
0	00
+1,-1	01+index (1bit)
+3,+2,-2,-3	10+index (2bit)
Altri valori	11+valore differenza $r_i - \mu$ (Kbit)

Si noti che per la decodifica delle informazioni occorre che all'interno del pacchetto oltre alle codifiche c_i siano specificati anche i valori dei parametri μ, *K e FLAG*. Pertanto il rapporto di compressione dell'algoritmo MinDiff nel caso peggiore (caso A) risulta essere

$$r_c = 1 - \frac{w + log_2(w) + 1 + w + (N-1)\cdot K}{N \cdot w} \approx 1 - \frac{K}{w} - \frac{2}{N}$$

Risultati analitici e sperimentali mostrano che per residui con distribuzione gaussiana e valori di N compresi fra 32 e 128 il rapporto di compressione dell'algoritmo MinDiff risulta differire da quello di un algoritmo ottimo (entropico) al più del 7%.

L'algoritmo suddetto è stato implementato tramite appositi firmware nell'ambito dell'attività di ricerca e sono stati condotti dei test su dati reali provenienti da misure sperimentali di pressione e temperatura. I test hanno mostrato che il rapporto di compressione medio dell'algortimo MinDiff raggiunge il 75% per misure indoor e il 25% per misure outdoor.

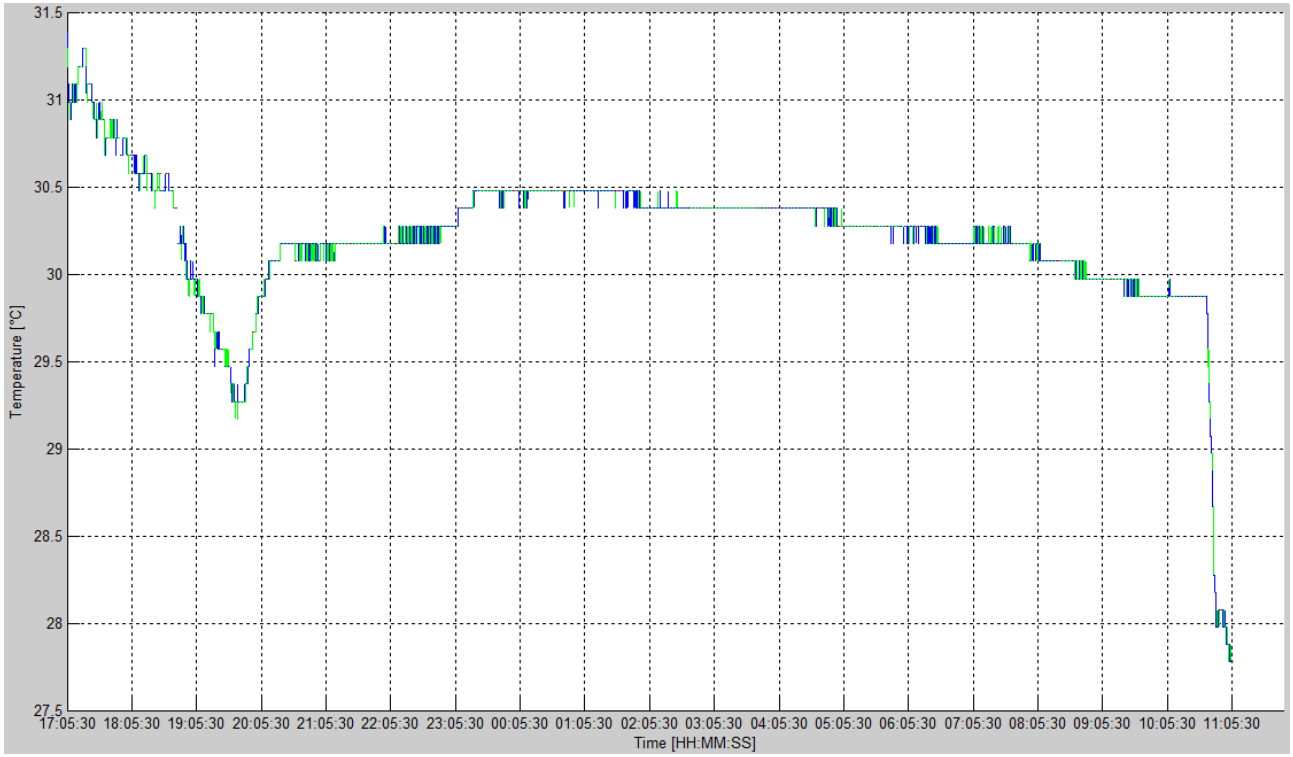


Figura 33 Andamento delle temperature rilevate sperimentalmente.

A titolo di esempio in Figura 33 è riportato l'andamento della temperatura acquisite tra il 25 luglio 2016 (5:05pm) e il 26 luglio 2016 (11:05am) presso il laboratorio di Comunicazioni Wireless del Dipartimento di Ingegneria dell'Università di Messina. Come è possibile osservare la temperatura misurata risulta compresa fra 27 e 31 °C (l'orario è stato scelto di modo che intervenissero i sistemi di condizionamento e si verificassero i transitori legati all'accensione e allo spegnimento di questi ultimi in modo da analizzare anche il caso peggiore legato ad ampie escursioni di temperatura in brevi periodi). Le misure sono state acquisite con un tempo di campionamento di 1 secondo e l'algoritmo è stato applicato a set di N=100 campioni consecutivi. I dati compressi sono stati effettivamente inviati ad un sink distante 2 metri al fine di misurare anche la probabilità di perdita dei pacchetti. Per il set di misure in esame l'algoritmo MinDiff ha ottenuto un rapporto di compressione medio pari a 76.9%, un rapporto di compressione massimo pari a 82.7% e la percentuale di pacchetti persi è stata del 5%.

8.2.3 Tecniche di compressione distribuite

Gli algoritmi di compressione distribuiti sfruttano la correlazione spaziale ovvero il fatto che nodi in prossimità acquisiscono spesso informazioni simili e altamente correlate. Alla base di tali algoritmi vi sono diversi risultati della teoria dell'informazione quali ad esempio il teorema di Slepian-Wolf [234] (alla base delle tecniche di Distributed Source Coding, DSC) e i recenti paradigmi del Compressive Sensing (CS) [235] e del Network Coding (NC) [236]. Abbinate a tecniche di routing tali algoritmi permettono di aggregare l'informazione di più pacchetti provenienti da nodi differenti in un unico pacchetto di dimensioni inferiori rispetto alla somma delle dimensioni dei pacchetti aggregati riducendo così il traffico di rete e aumentando il tempo di vita dei nodi. Un recente survey sulle tecniche di compressione distribuite per reti di sensori è stato proposto dagli stessi autori in [237]. Molte di tali tecniche risultano particolarmente complesse tanto che gran parte degli studi in letteratura sono basati esclusivamente su simulazioni e/o modelli. La complessità di tali tecniche è inoltre giustificabile solo per reti particolarmente dense (dell'ordine di un

nodo per metro quadrato) oltre che estese (dell'ordine delle centinaia di nodi). La verifica sperimentale dell'efficacia di tali tecniche richiederebbe quindi l'implementazione di un testbed con almeno diverse decine di nod. Per tali motivi si è deciso di non implementare tali tecniche nell'ambito dell'attività. Ciò non di meno, si ritiene auspicabile uno studio in tal senso come attività futura, anche in considerazione della possibilità intrinseca di tali tecniche non solo di ridurre i consumi energetici ma anche di migliorare l'affidabilità e il throughput della rete.

8.2.4 Tecniche di aggregazione

Un pacchetto può essere scomposto in due parti:

- il payload: che contiene l'insieme di tutti i dati relativi a misure provienenti da sensori e/o comandi per gli attuatori.
- l'header: che contiene tutte le informazioni di controllo e indirizzamento necessarie al corretto funzionamento dei vari livelli protocollari.

Mentre gli algoritmi di compressione precedentemente discussi, tendono alla riduzione della lunghezza del payload, le tecniche di aggregazione possono essere utilizzate per ridurre l'overhead dovuto all'header dei protocolli.

Come analizzato nel Capitolo 5 inerente i protocolli per WSN, nel caso di pacchetti UDP/IPv6 l'header è di 57byte, circa il 45% dei 127byte disponibili a livello fisico; si comprende quindi come ridurre le dimensioni dell'header possa essere importante quanto la compressione del payload.

Mediante le tecniche di compressione previste dal protocollo 6LowPAN la percentuale suddetta (sempre nell'ipotesi di un pacchetto di dimensione massima) scende a circa il 13%, percentuale che si ritiene accettabile. Nel caso però di un payload di soli 2 byte (contenente ad esempio una singola misura rappresentata con 16bit), l'overhead dovuto all'header anche in presenza di compressione 6LowPAN risulterebbe del 90%. È evidente quindi che l'utilizzo di tecniche di aggregazione può essere importante soprattutto per pacchetti con payload di piccole dimensioni (contenenti una singola misura o un singolo comando). Una soluzione semplice consiste nel far sì che un nodo relayer aggreghi più pacchetti ricevuti in un unico pacchetto. In questo modo, il peso dell'header viene ridotto di un fattore pari al numero di pacchetti aggregati.

Nell'ambito dell'attività non sono state prese in considerazione le tecniche di data aggregation poiché a fronte dei vantaggi suddetti introducono diversi svantaggi ed in particolare:

- riducono l'affidabilità di consegna (un singolo pacchetto perso implica la perdita di diverse informazioni)
- aumentano i ritardi (occorre attendere che il nodo relayer riceva un numero opportuno di pacchetti prima di effettuare l'aggregazione, ciò comporta un aumento dei ritardi di accodamento)
- riducono le risorse di memorizzazione disponibili per le applicazioni (occorre riservare spazio di memoria per poter memorizzare temporaneamente i pacchetti da aggregare)

Si ritiene quindi che l'implementazione delle tecniche di data aggregation non possa prescindere da uno studio combinato delle tecniche in grado di aumentare l'affidabilità di consegna (reliability), quali ad esempio le tecniche di correzione di errore (FEC) e/o di network coding. Pur ritenendo tale studio importante, al pari delle tecniche di minimizzazione dei consumi energetici, non si è proceduto in tal senso si propone di affrontare tale studio nelle attività prossime future.

Ciò non di meno, è stato effettuato uno studio preliminare utile a determinare il grado ottimo di aggregazione (ovvero il numero di pacchetti da aggregare) ai fini dei consumi energetici; tale studio, riportato nel Capitolo 9 permette di determinare analiticamente, in considerazione della reliability e dell'overhead dei protocolli, la dimensione ottima di un pacchetto ai fini della minimizzazione dei consumi energetici.

8.3 Conclusioni

In questo Capitolo sono state analizzate le principali tecniche di riduzione dei consumi energetici nelle reti di sensori individuando nelle tecniche di duty cycling e negli algoritmi di compressione quelle più efficaci. Sono stati inoltre individuati i parametri di configurazione di TinyOS su cui agire per ridurre i consumi energetici misurandone l'impatto sui consumi energetici anche attraverso misure sperimentali. Inoltre è stato implementato l'algoritmo di compressione MinDiff validandone sperimentalmente l'efficacia.

9 Modello analitico

In questo Capitolo verranno introdotte le principali metriche che definiscono le prestazioni di una rete WSN e sarà descritto un semplice modello analitico in grado di fornire i legami fra le metriche suddette e i diversi parametri di progettazione, in particolare il duty-cycle (δ), il traffico offerto per nodo (R_s, espresso in pacchetti per unità di tempo) e il numero massimo di ritrasmissioni (K_{max}).

Per definire le prestazioni ovvero la Qualità di Servizio (QoS) offerta da una rete WSN sono state considerate le seguenti metriche:

- **Tempo di vita** (T_{life}): il tempo di vita di una WSN è definito come il tempo che intercorre fra quando viene attivata la rete e quando il primo nodo esaurisce le sue riserve energetiche; un maggior tempo di vita indica quindi che una WSN ha minori consumi energetici.
- **Affidabilità di consegna:** l'affidabilità di consegna (nota come reliability) può essere misurata in termini del **Packet Delivery Ratio** (PDR) ovvero il rapporto tra il numero di pacchetti ricevuti a destinazione e il numero totale di pacchetti trasmessi; tale metrica tiene in considerazione la possibilità che per errori di trasmissione e/o collisioni i pacchetti non raggiungano la destinazione.
- **Latenza di rete** o **ritardo end-to-end** (D_{net}): la latenza di rete è il tempo che intercorre fra la trasmissione di un messaggio e la sua ricezione; tale metrica è particolarmente importante per applicazioni inerenti reti di attuatori e sistemi di controllo in tempo reale.
- **Throughput** (Thr)**:** massima velocità di trasmissione che un nodo può sostenere.

Di seguito si mostrerà come il duty-cycle abbia un impatto diretto su tutte le metriche suddette mentre il traffico offerto, se mantenuto entro i limiti tali da non congestionare la rete, influenza principalmente il tempo di vita; infine il numero massimo di ritrasmissioni comporta un compromesso fra PDR e latenza.

Parametro/Metrica	T_{life}	PDR	D_{net}	Thr
δ	X	X	X	X
R_s	X			X
K_{max}	X	X	X	

Tabella 16 Importanza dei parametri di progettazione di una rete sulle metriche di QoS.

9.1 Affidabilità di consegna di una WSN

L'affidabilità di una WSN può essere misurata sulla base della Packet Delivery Ratio (PDR) ovvero della probabilità che un pacchetto raggiunga la destinazione.

Tale probabilità dipende dalle condizioni del canale, dal numero di hop tra sorgente e destinazione, dal traffico di rete e dal protocollo di comunicazione.

In prima approssimazione la PDR può essere stimata come segue.

Lo standard 802.15.4 [77] prevede che la probabilità di errore attesa nella trasmissione di un singolo bit, nota come Bit Error Rate (BER), possa essere calcolata a partire dal rapporto segnale-rumore-interferenza (SINR) come

$$BER = \frac{8}{15} \cdot \frac{1}{16} \sum (-1)^k \binom{16}{k} e^{20*SINR*\left(\frac{1}{k}-1\right)}$$

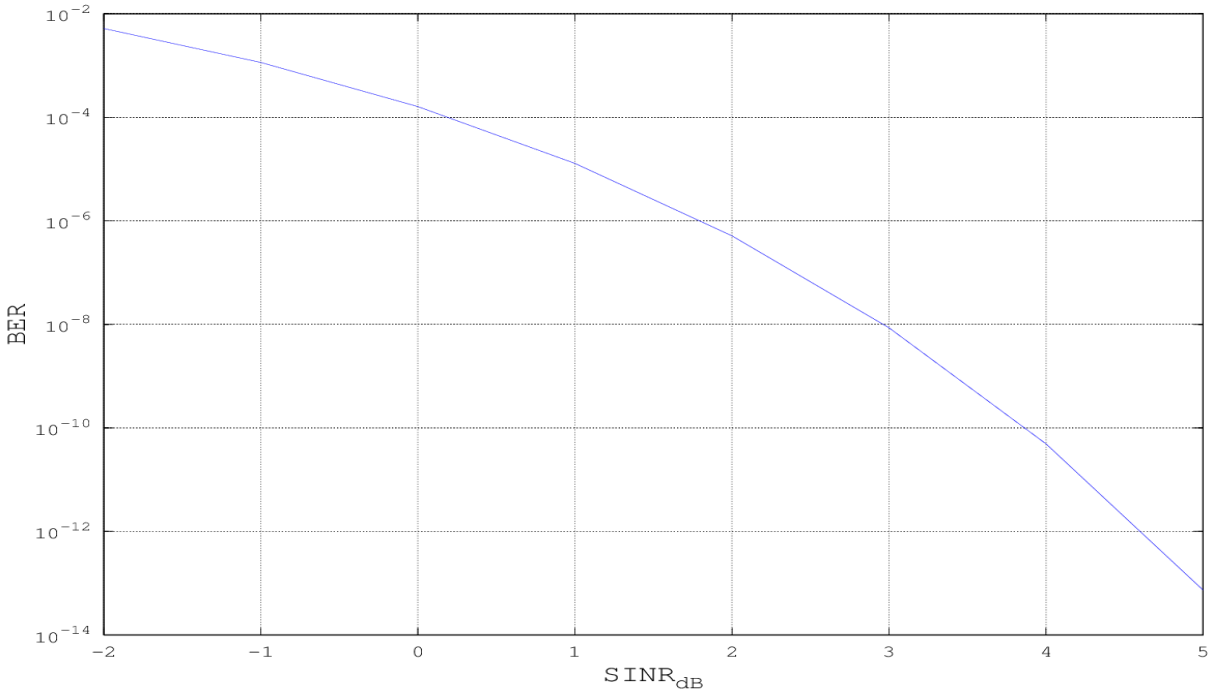


Figura 34 BER 802.15.4 al variare del SINR.

Nota la BER e considerando un pacchetto di lunghezza *L* (al più pari a 127 byte nello standard 802.15.4) è poi possibile determinare la probabilità che un pacchetto sia corrotto, indicata come Packet Error Rate (PER), per tramite della relazione:

$$PER = 1 - (1 - BER)^L$$

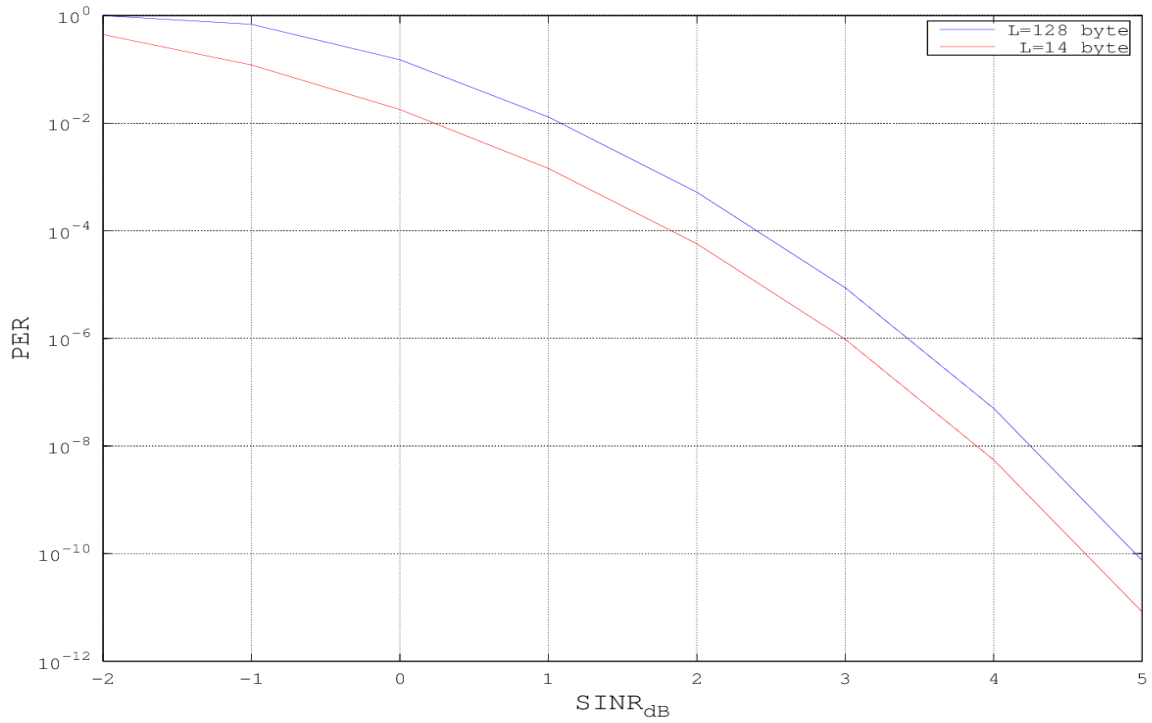


Figura 35 PER 802.15.4 al variare del SINR per differenti valori di L (128 byte, 14 byte).

In particolare per *SINR*=0dB ed *L*=127 byte si ha una *PER* pari a 0.15 mentre per un *SINR* di 1dB si ha *PER*=0.013. In seguito si assumeranno questi due valori come valori limite della PER per singolo hop riferendosi ad essi come “worst case environment” e “best case environment”.

Oltre che per effetto del canale un pacchetto potrebbe non essere ricevuto anche a causa delle possibili collisioni. Indicando con PCR (Packet Collision Rate) la probabilità che un pacchetto non sia ricevuto a causa delle collisioni, la probabilità totale che un pacchetto non sia ricevuto a distanza di un hop può esprimersi come

$$PER_{tot} = 1 - (1 - PER)(1 - PCR) \approx PER + PCR$$

La probabilità di collisione può essere stimata sulla base di un risultato noto nella teoria delle probabilità come "paradosso del compleanno" [238] e che permette di determinare la probabilità *p(n)* che almeno due persone su *n* siano nate nello stesso giorno dell'anno:

$$p(n) = 1 - \frac{365!}{365^n \cdot (365 - n)!}$$

Tale risultato può essere immediatamente esteso per determinare la probabilità che in un intervallo di backoff di B slots almeno due nodi su *n* scelgano lo stesso slot per trasmettere (ovvero che vi siano collisioni):

$$P_c = 1 - \frac{B!}{B^n \cdot (B - n)!} \approx 1 - e^{-\frac{n(n-1)}{2B}}$$

dove l'approssimazione è valida per $n \ll B$.

Supponendo che un pacchetto sia perso a seguito di C_{max} collisioni consecutive la probabilità di perdita per collisione diviene

$$PCR = P_c^{C_{max}}$$

È possibile dimostrare che fissato $C_{max} = 4$ e per un numero di nodi

$$n \leq 0.5 \cdot \sqrt{2B}$$

la PCR è dell'ordine di 0.001 per cui trascurabile rispetto alla PER sia per il caso worst che best-environment; può quindi assumersi $PER_{tot} = PER$.

Si noti che l'espressione precedente determina la relazione fra il massimo numero di nodi e l'intervallo di backoff per poter considerare trascurabile l'effetto delle collisioni.

Consideriamo adesso che il pacchetto debba fare *H* hop per raggiungere la destinazione (ovvero che vi siano altri *H-1* nodi fra sorgente e destinazione), la probabilità che il pacchetto non arrivi a destinazione (anche detta probabilità di perdita di percorso) è pari a

$$PER(H) = 1 - (1 - PER_{tot})^H$$

Infine indicato con K_{max} il massimo numero di ritrasmissioni previste dal protocollo di comunicazione end-to-end, la probabilità che, considerate le ritrasmissioni, un pacchetto raggiunga la destinazione è pari a

$$PDR = 1 - PER(H)^{K_{max}}$$

È evidente quindi che all'aumentare di K_{max} aumenta la probabilità suddetta e quindi migliora la realibility.

Nella tabella seguente sono riportati i valori di K_{max} necessari per ottenere una determinata PDR al variare del numero di hop *H* rispettivamente per i casi best/worst.

PDR	**H=1**		**H=4**		**H=10**	
	best	worst	best	worst	best	worst
0.95	1	2	1	4	2	14
0.99	2	3	2	7	3	21
0.999	2	4	3	10	4	32

Si noti che per avere una elevata PDR, a parità di altre condizioni, occorre aumentare il valore di K_{max} o limitare il numero massimo di hop. Occorre però tenere presente che, come verrà dimostrato nelle prossime sezioni, aumentare il valore di K_{max} ha come effetto quello di ridurre il tempo di vita e di aumentare la latenza della rete.

9.2 Tempo di vita di una WSN

Fissata l'energia totale *E* a disposizione di un nodo è possibile in prima approssimazione ricavarne il tempo di vita dalla relazione

$$T_{life} = \frac{E}{R_s \cdot e \cdot N_p}$$

dove R_s è il traffico offerto da un nodo (ovvero il numero di pacchetti al secondo generati da un nodo), *e* è l'energia necessaria per la trasmissione di un pacchetto e N_p è il numero di pacchetti trasmessi da un nodo tenuto conto anche delle ritrasmissioni necessarie per garantire che il pacchetto raggiunga la destinazione nonostante la probabilità di perdita introdotta dal canale.

La relazione suddetta può essere giustificata considerando che se *e* è l'energia necessaria per trasmettere un singolo pacchetto, l'energia totale necessaria per la trasmissione di N_p pacchetti sarà $E = N_p \cdot e$

In una rete ideale, supponendo che un nodo emetta un pacchetto ogni T_s secondi, *N* pacchetti sarebbero trasmessi in un tempo pari a $T_{ideale} = T_s \cdot N = (1/R\_s) \cdot (E/e)$ secondi; per cui in una rete ideale $T_{ideale} = E/(R_s \cdot e)$ sarebbe anche il tempo di vita del nodo ovvero il tempo necessario ad esaurire l'energia disponibile, *E*. In una rete reale, ovvero in presenza di errori di trasmissione e perdita di pacchetti, lo stesso pacchetto deve essere trasmesso più volte per garantire che esso raggiunga la destinazione, per cui a fronte di un pacchetto generato devono essere trasmessi $N_p = 1 + N_r$ pacchetti con N_r numero di ritrasmissioni; in presenza di ritrasmissioni quindi il tempo di vita si riduce, quindi, di un fattore N_p da cui l'espressione del tempo di vita

$$T_{life} = \frac{E}{R_s \cdot e \cdot (N_r + 1)}$$

L'espressione suddetta mostra che fissata l'energia a disposizione per l'alimentazione di un nodo, per aumentare il tempo di vita della rete è possibile agire in tre modi:

1. riducendo il traffico offerto (R_s);
2. riducendo il numero di ritrasmissioni (N_r);
3. riducendo l'energia per pacchetto (*e*).

Il valore di R_s dipende in genere dall'applicazione. Analizziamo quindi gli altri due fattori.

Il numero atteso di ritrasmissioni N_r può essere calcolato a partire dalla probabilità di perdita del percorso (*PER(H)*) e dal numero massimo di ritrasmissioni K_{max} come:

$$N_r = \sum i\, PER(H)^i \big(1 - PER(H)\big) + K_{max} PER(H)^{K_{max}} = \frac{PER(H) - PER(H)^{K_{max}+1}}{1 - PER(H)}$$

Considerando ad esempio K_{max}= 4 e *H*=3 si ha N_r=0.61 nel caso di worst-case (*PER*=0.15) e N_r=0.04 nel caso di best-case (*PER*=0.013).

In generale per K_{max} sufficientemente grande si ha $N_r \cong PER(H)/(1 - PER(H))$ e $N_p \cong 1/(1 - PER(H))$.

Si noti che il numero atteso di ritrasmissioni aumenta col numero di hop ma si mantiene inferiore a 1 finché *PER(H)* è inferiore a 0.5. Per avere un basso numero di ritrasmissioni occorre quindi, a parità di condizioni, limitare il numero massimo di hop, *H*. Come si vedrà nella prossima sezione ridurre il numero di hop è importante anche per ridurre la latenza di rete.

L'energia per pacchetto *e* è data dalla somma di diversi contributi, in particolare:

- energia spesa per l'*elaborazione* da parte del microcontrollore, e_{MCU};
- energia spesa dal *transceiver* (necessaria non solo per trasmettere ma anche per ascoltare il canale), e_{TRX};
- energia spesa dai *sensori,* e_{Sens};

per cui

$$e = e_{MCU} + e_{TRX} + e_{Sens}$$

Analizziamo i singoli contributi.

L'energia spesa dal microcontrollore può essere valutata come

$$\mathrm{e_{MCU}} = \mathrm{T_{elab}} \cdot \mathrm{I_{MCU}} \cdot \mathrm{V_{batt}} + \mathrm{T_{sleep}} \cdot \mathrm{I_{MCUsleep}} \cdot \mathrm{V_{batt}}$$

dove I_{MCU} e $I_{MCUsleep}$ sono le correnti assorbite dal micro rispettivamente durante l'elaborazione e durante la fase di inattività, V_{batt} è la tensione delle batterie, T_{elab} è il tempo necessario per l'elaborazione di un pacchetto che può essere valutato come e $T_{elab} = \frac{\mathrm{N_{istr}}}{\mathrm{f_{ck}}}$ dove N_{istr} è il numero di istruzioni (espresse in periodi di clock) necessarie per elaborare un pacchetto e f_{ck} è la frequenza di clock del microcontrollore, mentre $T_{sleep} = T_s - T_{elab}$ è il tempo trascorso dal microcontrollore in stato di sleep.

Ad esempio considerando 10000 istruzioni, una frequenza di clock di 4MHz, una corrente assorbita di 1.8mA e una tensione di alimentazione di 3V, e trasurando la corrente in stadio di sleep, si ha $e_{MCU} = 13.5\ \mu J$.

L'energia spesa dai sensori può essere determinata come

$$e_{Sens} = N_{sens} \cdot T_{sens} \cdot \frac{V_{batt}^2}{R}$$

dove *R* è la resistenza equivalente del sensore, T_{sens} il tempo di ogni lettura e N_{sens} è il numero di letture. Ad esempio considerando una resistenza equivalente del sensore di 10kW e un tempo di 10ms per ogni misura, occorre una energia di 9μJ per ogni misura.

Tipicamente, al fine di limitare l'utilizzo della memoria, il numero di misure N_{sens} è fissato in modo che sia possibile trasmettere le misure acquisite in un unico pacchetto; poiché nello standard 802.15.4 la dimensione del pacchetto è limitata a 127 byte, assumendo di rappresentare ogni misura con 2 byte, il valore di N_{sens} non può superare 63. Ne consegue che e_{sens} è in genere inferiore ai 570mJ.

L'energia spesa dal transceiver è data dalla somma dei seguenti contributi:

- energia per la *trasmissione* ovvero l'energia spesa nello stato attivo per la trasmissione dei pacchetti (dipende dalla distanza *d* fra trasmettitore e ricevitore e dalla lunghezza *L* del pacchetto);
- energia per la *ricezione* ovvero l'energia spesa nello stato attivo per la ricezione dei pacchetti (dipende dalla lunghezza del pacchetto);
- energia spesa per l'*ascolto del canale* (dipende dal duty-cycle e dal traffico ovvero dalle collisioni);
- energia spesa nello stato di *sleep* (in genere può essere trascurata).

I contributi suddetti possono essere determinati come segue:

Supponendo che per ogni bit trasmesso o ricevuto un nodo impieghi una energia pari a e_b allora la trasmissione o ricezione di un pacchetto di lunghezza *L* può essere calcolata come $e_b \cdot L$.

Il valore di e_b può essere determinato a partire dai dati sui consumi del transceiver come segue

$$e_b = V_{batt} \cdot I_{TRX} \cdot T_b$$

dove V_{batt} è la tensione delle batterie, I_{TRX} è la corrente assorbita dal transceiver e T_b il tempo di trasmissione di un bit (pari a 1/250000=4ms nel caso di 802.15.4).

Considerata una corrente di 20mA e una tensione di alimentazione di 3V si ha e_b=240nJ/bit. Nel caso di un pacchetto di dimensione massima (127 byte) occorre quindi una energia paria a $e_b \cdot L$=2.44mJ.

Indicato con $T_s = 1/R_s$ il tempo che intercorre fra la generazione di due pacchetti consecutivi, con T_p il tempo di trasmissione e/o ricezione di un pacchetto di lunghezza L e con N_p il numero di pacchetti trasmessi/ricevuti nell'intervallo T_s, il tempo in cui un nodo rimane in attesa è $T_w = T_s - N_p \cdot T_p$. In questo intervallo il nodo alterna periodicamente periodi di attività (per ascoltare il canale) a periodi di sleep. Indicando con T_{CCA} il tempo di permanenza nello stato attivo e con T_{SL} il tempo di permanenza nello stato di sleep, nell'intervallo di tempo T_w un nodo controllerà il canale circa $[T_w/(T_{CCA} + T_{SL})]$ volte per cui il tempo passato attivamente in fase di ascolto sarà $T_A = [T_w/(T_{CCA} + T_{SL})] \cdot T_{CCA}$ e in tale intervallo il nodo consumerà una energia pari a

$$e_{TA} = V_{batt} \cdot I_{listen} \cdot T_A \approx V_{batt} \cdot I_{listen} \cdot T_s \cdot \delta$$

L'approssimazione deriva dal fatto che anche considerando la dimensione massima del pacchetto (127byte), T_p è al più 4.2ms per cui $N_p \cdot T_p$ è in genere trascurabile rispetto a T_s (T_s è tipicamente dell'ordine dei secondi o minuti per attività di sensing e non inferiore ai 100ms anche per molte applicazioni real-time).

È importante sottolineare che è necessario fissare i valori di T_{CCA} e T_{SL} opportunamente in funzione di T_s al fine di rendere e_{TA} trascurabile rispetto ad altri contributi.

Come è possibile osservare dalla tabella seguente infatti il valore di e_{TA} varia fortemente in funzione dei parametri suddetti (in particolare come si vede dal confronto dei Casi 1 e 2 in tabella lasciare i valori di default può comportare un notevole spreco di energia, confrontando i casi 3 e 4 è inoltre evidente l'importanza di ridurre il duty-cycle agendo su T_{CCA}).

Tabella 17 Energia nelle fasi di ascolto al variare dei parametri di duty-cycling.

Caso	T_S	T_{CCA}	$T_{CCA} + T_{SL}$	δ (%)	e_{TA} (mJ)
1	2	12.5	512	2.44	2.93
2	15	12.5	512	2.44	21.96
3	15	12.5	4200	0.30	2.70
4	15	6.0	4200	0.14	1.26

Va osservato che molti modelli analitici e simulatori trascurano il contributo di e_{TA} che invece, come mostrato negli esempi suddetti, può essere rilevante (superiore anche di un ordine di grandezza rispetto all'energia spesa per la trasmissione di un pacchetto, come si vede dal caso 2) soprattutto se non si dimensionano correttamente i parametri inerenti il duty-cycle. Si noti che affinchè e_{TA} sia inferiore a $e_b \cdot L$ occorre che

$$\delta \cdot T_s < L \cdot T_b \cdot \frac{I_{TRX}}{I_{listen}}$$

Considerando L=127 byte e $I_{TRX} = I_{listen}$ la condizione suddetta diviene

$$\delta \cdot T_s < 4 \cdot 10^{-3}$$

Si noti che tale condizione è verificata solo per il caso 4 ovvero dove si è agito opportunamente sui valori dei parametri che determinano il duty-cycle.

Indicando con e_0 la somma dell'energia spesa dal microcontrollore, dai sensori e dal transceiver nello stato di ascolto ($e_0 = e_{MCU} + e_{sens} + e_{TA}$) è possibile esprime l'energia necessaria per la trasmissione di ogni pacchetto generato (tenendo conto anche delle ritrasmissioni) come

$$e = e_0 + e_b \cdot L \cdot N_p$$

Per cui un modello più accurato per la stima del tempo di vita risulta essere dato dalla seguente espressione

$$T_{life} = \frac{E}{R_s \cdot [e_0 + e_b \cdot L \cdot (N_r + 1)]}$$

L'espressione precedente si basa sull'ipotesi che un nodo debba trasmettere solo pacchetti generati da se stesso; in realtà in una WSN un nodo fa da relayer per altri nodi per cui oltre ai pacchetti generati dovrà trasmettere i pacchetti ricevuti da altri nodi. Indicando con γ il rapporto fra il numero di pacchetti ricevuti e quelli generati è possibile determinare il tempo di vita di un nodo relayer come segue

$$T_{life} = \frac{E}{R_s \cdot [e_0 + e_b \cdot L \cdot (1 + \gamma) \cdot (N_r + 1)]}$$

Dalla relazioni precedenti è evidente che è possibile aumentare il tempo di vita agendo sulla dimensione dei pacchetti (*L*). Si noti che *L* influenza il tempo di vita anche per tramite di N_r (in particolare riducendo *L* si riduce la *PER* e di conseguenza N_r).

Da qui si comprende l'importanza delle tecniche di compressione nelle WSN.

9.3 Efficienza energetica e dimensione ottima del pacchetto

È possibile definire l'efficienza energetica E_{eff} di una WSN come rapporto fra il numero bit di informazione utile trasmessi e l'energia spesa per ogni pacchetto.

Sulla base del modello del tempo di vita appena ricavato per un pacchetto di dimensione *L* occorre un'energia $e = e_0 + e_b \cdot L \cdot N_p$. In generale però a causa dell'overhead introdotto dai protocolli di comunicazione, a fronte di L bit trasmessi solo L-O saranno bit di informazione utile (altrimenti detto Payload), mentre bit saranno necessari per heder del pacchetto. L'efficienza energetica può quindi essere espressa come

$$E_{eff} = \frac{L - O}{e_0 + e_b \cdot L \cdot N_p}$$

Si noti che N_p è funzione di *L* in particolare si ha

$$N_p \approx \frac{1}{1 - PER(H)} \approx \frac{1}{(1 - H \cdot BER)^L}$$

dove la seconda approssimazione è lecita se BER<<1/H (ipotesi tipicamente verificata).

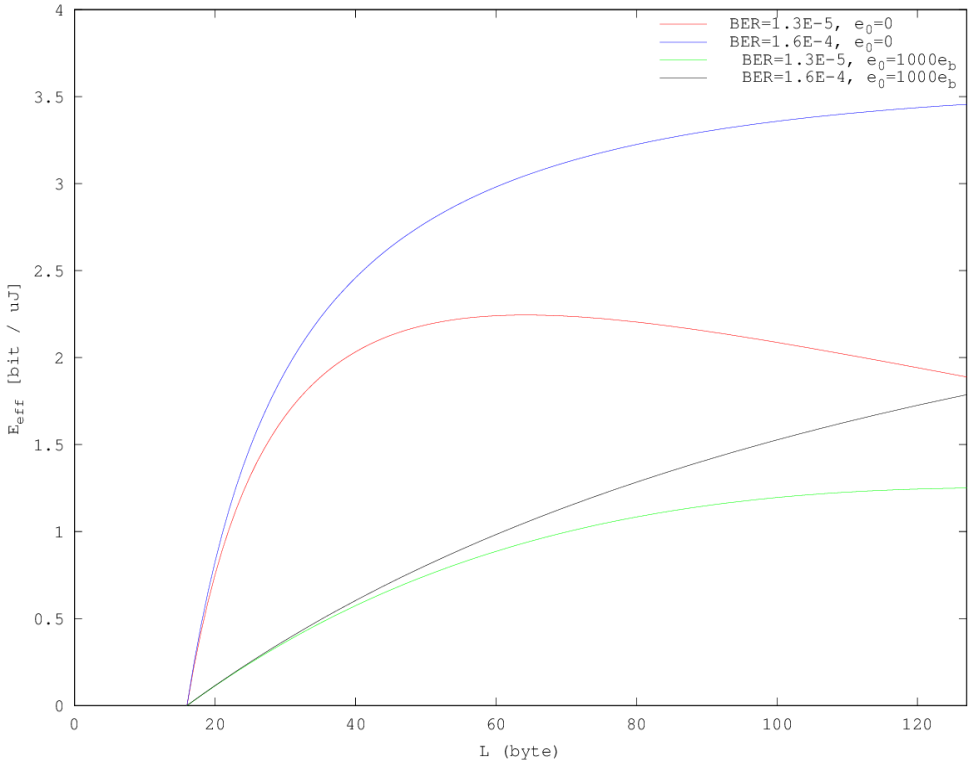


Figura 36 Efficienza enegetica per diversi valori di BER e di e0 (O=16 byte, H=4, eb=240nJ/bit).

In Figura 36 è illustrato l'andamento dell'efficienza energetica al variare di L per diversi valori di $H \cdot BER$ per un overhead O fissato pari a 16byte (overhead tipico dei protocolli 802.15.4 e 6LowPAN).

Dai grafici è possibile osservare che esiste un valore di L tale da massimizzare l'efficienza energetica e che all'aumentare di H×BER tale valore diminuisce.

È possibile dimostare che se e_0 è trascurabile rispetto a $e_b \cdot L \cdot N_p$ il valore ottimo di L è pari a

$$L_{opt} \approx \frac{O}{2} + \sqrt{\frac{O}{-ln(1 - H \cdot BER)}}$$

Viceversa se e_0 è grande rispetto a $e_b \cdot L \cdot N_p$ allora l'efficienza energetica è una funzione crescente e conviene far sì che i pacchetti abbiano la dimensione massima possibile (127 byte).

9.4 Latenza di rete

Il ritardo atteso nella trasmissione di un pacchetto è anch'esso legato ai parametri che determinano il duty cycle. Per determinare tale ritardo è possibile procedere come segue.

Consideriamo dapprima due nodi A e B in visibilità radio (ovvero l'uno nel range di copertura dell'altro). Senza perdita di generalità supponiamo che il nodo B sia attivo nell'intervallo (0, T_{CCA}) e vada in stato di sleep nell'intervallo $(T_{CCA}, T_{CCA} + T_{SL})$ e che il nodo A scelga l'istante di inizio della trasmissione t con distribuzione uniforme nell'intervallo $(0, T_{CCA} + T_{SL})$. Il nodo A potrà trasmettere a B direttamente (con ritardo praticamente nullo) se B è nello stato attivo, ovvero se l'istante scelto per la trasmissione appartiene all'intervallo $(0, T_{CCA})$, altrimenti, prima di trasmettere, dovrà attendere che B si risvegli (in tal caso A dovrà attendere il successivo periodo di attività di B che si verificherà al tempo $T_{CCA} + T_{SL} - t$). Sulla base di tali ipotesi è possibile determinare il ritardo atteso E(D) tramite la seguente espressione

$$E(D) = \int D\, f_D(D) dD = \frac{1}{T_{CCA} + T_{SL}} \left[\int 0\, dD + \int (T_{CCA} + T_{SL} - D)\, dD \right] = \frac{T_{SL}^2}{2(T_{CCA} + T_{SL})}$$

Si noti che essendo in genere T_{SL} grande rispetto a T_{CCA} in prima approssimazione si ha

$$\mathrm{E(D)} \approx \frac{T_{SL}}{2}$$

Al fine di stimare il ritardo massimo è utile determinare anche la varianza del ritardo:

$$\sigma_D^2 = \int \big(D - E(D)\big)^2 f_D(D) dD = \int D^2 f_D(D) dD - [E(D)]^2$$

$$\frac{1}{T_{CCA} + T_{SL}} \left[\int 0^2 \, dD + \int (T_{CCA} + T_{SL} - D)^2 \, dD \right] - [E(D)]^2 = \frac{T_{SL}^3}{T_{CCA} + T_{SL}} \left[\frac{1}{3} - \frac{T_{SL}}{4(T_{SL} + T_{CCA})} \right]$$

Sempre nell'ipotesi che T_{SL} sia grande rispetto a T_{CCA} l'espressione precedente può essere approssimata come

$$\sigma_D^2 \approx \frac{T_{SL}^2}{12}$$

Nell'ipotesi di distribuzione gaussiana dei ritardi la probabilità che un campione si allontani dal valore atteso per più di $3\sigma_D$ é dell'ordine dell'uno per mille. Nell'ipotesi di poter considerare trascurabile tale probabilità è possibile stimare il ritardo massimo della comunicazione fra due nodi in visibilità radio come

$$D_{max} = E(D) + 3\sigma_D \approx \big(1 + \sqrt{3}\big) \frac{T_{SL}}{2}$$

Nel caso A e B non siano in visibilità radio, supponendo che A e B distino H hop, è possibile assumere che media e varianza aumenteranno entrambe di un fattore H. Ne consegue che il ritardo massimo in una rete di H hop può essere stimato come

$$D_{net}(H) = H \cdot E(D) + 3\sqrt{H}\sigma_D \approx \big(H + \sqrt{3 \cdot H}\big) \frac{T_{SL}}{2}$$

Nella tabella sottostante è riportato il rapporto D_{net}/T_{SL} per diversi valori di *H*.

H	D_{net}/T_{SL}
1	1.366
2	2.225
3	3.000
4	3.732
5	4.436
10	7.738

A verifica del modello suddetto, per tramite del testbed realizzato sono state condotte delle misure con una rete costituita da 5 hop al variare del tempo di sleep.

In Tabella 18 sono riportati i risultati ottenuti trasmettendo 2000 pacchetti (tutti i ritardi sono espressi in millisecondi).

Tabella 18 Confronto tra i dati sperimentali e il modello proposto.

T_{SL} (ms)	Ritardo medio	Ritardo medio	Ritardo massimo	Ritardo massimo	Outliers

	(misurato)	(modello)	(misurato)	(modello)	(%)
212	504	530	1137	940	15
512	1219	1280	3688	2271	10
1012	2347	2530	8109	4490	12

Come è possibile osservare i valori sperimentali e teorici della media sono molto prossimi; per quanto riguarda il valore massimo del ritardo stimato dal modello è inferiore (questo perché la distribuzione reale non è gaussiana) ma la percentuale di pacchetti che superano il valore atteso (outliers, riportati nell'ultima colonna) è dell'ordine del 10%.

Figura 37 Scenario dell'esperimento relativo alla Tabella 17.

Il modello suddetto non tiene conto della possibile perdita di pacchetti e delle conseguenti ritrasmissioni. Ciò non di meno può essere considerato valido per una rete a basso traffico dove il numero di collisioni è trascurabile e/o il numero di ritrasmissioni atteso è trascurabile (come nel caso di best-case environment).

In presenza di ritrasmissioni il ritardo introdotto dalla rete dipenderà anche dal meccanismo che gestisce le ritrasmissioni e in particolare dalla configurazione dei timer associati. Più precisamente ogni volta che viene trasmesso un pacchetto un nodo avvia un timer e attende un messaggio di conferma (ACK) per un tempo massimo pari a $T_{ACKWAIT}$. Se il messaggio di conferma non viene ricevuto entro lo scadere del timer allora il nodo ritrasmetterà il pacchetto per cui ogni pacchetto perso determina un aumento del ritardo di un tempo pari a $T_{ACKWAIT}$.

Indicato con Nr il numero atteso di ritrasmissioni, il tempo necessario per la trasmissione di un pacchetto può essere stimato come

$$T_{TX} = T_p + D_{net}(H) + N_r \cdot T_{ACKWAIT}$$

Si noti che se da un lato ridurre $T_{ACKWAIT}$ permetterebbe di ridurre T_{TX} dall'altro un valore eccessivamente basso porterebbe alla ritrasmissione anche di pacchetti ricevuti correttamente con conseguente aumento del traffico generato, aumento delle collisioni e riduzione del throughput della rete.

Idealmente $T_{ACKWAIT}$ deve essere pari al massimo round-trip-time (RTT) ovvero al tempo massimo che intercorre fra la trasmissione di un pacchetto e la ricezione dell'ACK:

$$RTT = 2 \cdot D_{net}(H) + T_{elab}$$

dove T_{elab} è il tempo necessario ad un nodo per elaborare un pacchetto e generare un ACK e D_{net} è la latenza introdotta dalla rete.

In pratica il valore di RTT non è costante nel tempo di conseguenza $T_{ACKWAIT}$ deve essere adattato alle condizioni di rete mediante appositi algoritmi.

Il protocollo M2M COAP ad esempio prevede che il $T_{ACKWAIT}$ (noto come RTO nella terminologia COAP) cresca esponenzialmente col numero di ritrasmissioni.

In particolare inizialmente il valore del timer è impostato ad un valore casuale T_0 scelto nell'intervallo (ACK_TIMEOUT,ACK_TIMEOUT×ACK_RANDOM_FACTOR), di default compreso fra 2 e 3 secondi. Ad ogni ritrasmissione il valore del RTO viene raddoppiato per cui se K_{max} è il massimo numero di ritrasmissioni (indicato come MAX_RETRANSMIT e di default pari a 4) allora il massimo tempo fra la prima trasmissione di un pacchetto e la sua ultima ritrasmissione sarà pari a $(2^{Kmax} - 1) \cdot T_0$ (con i valori di default è compreso fra 30 e 45 secondi) e il massimo valore di $T_{ACKWAIT}$ è a $(2^{Kmax} - 1) \cdot T_0$ (di default compreso fra 62 e 93 secondi). I tempi suddetti possono essere ridotti agendo sul parametro ACK_TIMEOUT.

9.5 Throughput di una WSN

Il throughput di una rete rappresenta il massimo numero di bit che possono essere inviati nell'unità di tempo.

Esso può essere determinato a partire dal tempo totale medio di trasmissione di un pacchetto (T_{tot}) e dal numero di bit inviati per pacchetto al netto dell'overhead protocollare (Payload=L-O) mediante la relazione

$$Throughput = Payload/T_{tot}$$

In questa sezione si intende determinare il massimo throughput di una rete WSN basata sul livello fisico dello standard 802.15.4 e sul livello MAC BoxMACv2 di TinyOS. Si considereranno tre casi:

1. una rete a singolo hop senza perdita e senza duty-cycle;
2. una rete a singolo hop con perdita e senza duty-cycle;
3. una rete multi-hop con duty-cycle.

Caso 1: Rete a singolo hop senza perdita

Nel caso di singolo hop senza perdita i contributi che determinano il ritardo di trasmissione di un pacchetto sono:

- il tempo di accesso al canale (legato all'algoritmo di backoff), T_{bo}: l'algoritmo BoxMACv2 prevede che prima della trasmissione un nodo resti in attesa un tempo casuale che inizialmente (Initial Backoff) risulta pari al massimo a 50 jiffies (1.525ms);
- il tempo di trasmissione del pacchetto, T_p: considerando la bit-rate di 250kbps e supponendo la dimensione massima del pacchetto (127 byte) si ha T_p=(127+5+1)/250000=4.256ms (il calcolo considera anche la dimensone dell'header del livello fisico);
- il tempo di commutazione TX/RX (ACKTURNAROUND), T_{ta}: lo standard 802.15.4 prevede che tale tempo sia di 192us ma in TinyOS risulta essere pari a 7 jiffies (214us);
- il tempo di trasmissione dell'ack, T_{ack}: un ack ha una dimensione di 5 byte a livello MAC a cui occorre aggiungere i 6 byte dell'header di livello fisico per cui si ha T_{ack}=(6+5)/250000=0.352ms.

Il tempo totale risulta quindi

$$T_{tot} = T_{bo} + T_p + T_a + T_{ack} = 6.347ms$$

Infine considerando un payload a livello MAC di 114 byte (sui 127 disponibili) il throughput risulta essere

$$Throughput = Payload/T_{tot} = \mathbf{143690\ bps}$$

Si noti che il Throughput, anche nelle condizioni ideali, risulta essere notevolmente più basso della massima bitrate disponibile prevista dallo standard (250kbps).

Caso 2: rete a singolo hop con perdita

Calcoliamo adesso il throughput nel caso di perdita di pacchetti.

Assumiamo a titolo di esempio una perdita pari al 15% (ovvero pari alla PER del worst-case).

Per l'85% dei pacchetti (ovvero per i pacchetti che non richiedono ritrasmissione) il tempo di tramissione sarà quello stimato in precedenza (indicato con T_{totnr}).

Per il restante 15% invece possiamo assumere che al più vi sia una ritrasmissione (N_r=1) per cui il tempo totale sarà

$$T_{totr} = N_r \cdot \left(T_{bo} + T_p + T_{ACKWAIT}\right) + T_{bo} + T_p + T_{ta} + T_{ack} = 19.928\ ms$$

dove con Tackwait si è indicato il tempo di attesa dell'ack, che di default in TinyOS è pari a 256 jiffies (7.8ms)

Il tempo medio di trasmissione risulterà quindi

$$T_{tot} = 0.85 \cdot T_{totnr} + 0.15 \cdot T_{totr} = 8.384\ ms$$

a cui corrisponde un throughput pari a

$$Throughput = Payload/T_{tot} = \mathbf{109250\ bps}$$

Nello scenario suddetto, a causa delle ritrasmissioni, si ha quindi una riduzione del throughput di circa il 25% rispetto al caso precedente.

Caso 3: rete multi-hop con duty-cycle

Gli scenari suddetti non tengono conto del duty-cycle. In presenza di duty-cycle il tempo di trasmissione può essere determinato sulla base del modello della latenza di rete precedentemente sviluppato.

In particolare supponendo N_r ritrasmissioni e H hop il tempo per la trasmissione di un pacchetto risulterà essere

$$T_{totr} = N_r \cdot \left(T_{bo} + T_p + T_{ACKWAIT}\right) + T_{bo} + T_p + H \cdot E(D) + T_{ta} + T_{ack}$$

Supponendo $T_{ackwait}$ dimensionato in modo da essere pari al RTT massimo ovvero

$$T_{ACKWAIT} = RTT = 2 \cdot D_{net}(H) + T_{elab}$$

si ha

$$\begin{aligned} T_{tot} &= N_r \cdot \left(T_{bo} + T_p + 2 \cdot D_{net}(H) + T_{elab}\right) + T_{bo} + T_p + 2 \cdot H \cdot E(D) + T_{ta} + T_{bo} + T_{ack} \\ &\approx 2\left(N_r \cdot D_{net}(H) + H \cdot E(D)\right) = \left[N_r \cdot \left(H + \sqrt{3 \cdot H}\right) + H\right] \cdot T_{SL} \end{aligned}$$

dove l'ultima approssimazione è lecita se i tempi T_{SL}, T_{ack}, T_p, T_{elab} e T_{bo} sono trascurabili rispetto a E(D) e D_{net} (ovvero piccoli rispetto a T_{SL}).

Supponendo ad esempio T_{SL} =512ms e H=4 il tempo atteso rispettivamente in assenza di ritrasmissioni e nel caso di una sola ritrasmissione risulta essere T_{totnr} =2.048s e T_{tottr} =8.887s.

Ipotizzando come nel caso precedente una condizione di worst-case in cui il 15% dei pacchetti richiede una ritrasmissione e l'85% non richieda ritrasmissioni, il tempo medio di tramissione risulterà essere

$$T_{tot} = 0.85 \cdot T_{totnr} + 0.15 \cdot T_{totr} = 3.074s$$

a cui, sempre considerando un payload di 114 byte come nei casi di studio precedenti, corrisponde un throughput pari a

$$Throughput = Payload/T_{tot} = 297\ bps$$

Il throughput appena calcolato, sebbene compatibile con applicazioni di sensing, in cui tipicamente occorre trasferire pochi byte al secondo o addirittura al minuto, è decisamente più basso dei valori precedentemente ricavati per i casi di studio 1 e 2 e non accettabile per applicazioni di attuazione. Potrebbe quindi sembrare che questo sia il prezzo da pagare per ridurre i consumi energetici mediante tecniche di duty-cycling. In pratica non è così.

Il throughput di un singolo nodo può essere aumentato facendo sì che i nodi non entrino immediatamente in modalità di sleep al termine della trasmissione o ricezione di un pacchetto ma rimangano in modalità di ascolto per un opportuno intervallo di tempo (in TinyOS tale intervallo è definito dalla variabile *DELAY_AFTER_RECEIVE* e di default è pari a 100ms*)*. In tal modo sono possibili trasmissioni a burst ovvero più pacchetti possono essere inviati consecutivamente. Nel caso di trasmissioni a burst il ritardo $H \cdot E(D)$ va considerato solo per il primo dei pacchetti trasmessi in un burst.

Per comprendere meglio quanto appena detto e cosa ciò comporti facciamo un esempio.

Supponiamo che un nodo trasmetta dieci pacchetti consecutivamente e che trovando il canale libero ciò avvenga in un tempo $10 \cdot (T_{bo} + T_p) = 57.8ms$. Il burst raggiungerà il nodo di destinazione dopo un tempo pari a $H \cdot E(D) = 1.024$s; supponendo che il nodo di destinazione invii immediatamente gli ack la trasmissione dei pacchetti avrà richiesto un tempo pari a

$$T_{tot} = 10 \cdot (T_{bo} + T_p) + H \cdot E(D) + T_a + 10 \cdot T_{ack} = 1.086s$$

Si noti che il numero di pacchetti trasmessi in tale tempo è pari a 10 per cui il throughput risulta essere

$$Throughput = 10 \cdot Payload/T_{tot} = \mathbf{10502\ bps}$$

decisamente superiore al caso precedente e compatibile con le velocità di trasmissione tipiche degli standard per smart metering (ad esempio M-Bus la cui velocità tipica è di 9600bps).

Occorre poi osservare che tale throughput è per singolo nodo e poiché nell'intervallo di sleep di un nodo altri nodi possono trasmettere, il throughput complessivo della rete risulta decisamente più alto potendo in teoria raggiungere i 140kbps visti per il caso di studio 1.

Tabella 19 Risultati sperimentali.

Tempo ping (ms)	Traffico totale (bps)	Media ritardo (ms)	Deviazione standard ritardo (ms)	Pacchetti Ritrasmessi (%)	Pacchetti persi (%)
1000	6446	70	5,8	0	0
500	10560	63	101,68	0	0
250	18752	109,58	417,64	4	0
100	43328	830,05	2449,6	32	0

Nell'ambito dell'attività sono state effettuate diverse misure sperimentali per avvalorare l'analisi suddetta. Ad esempio, in Tabella 19 sono mostrati i risultati sperimentali relativi allo scenario di Figura 35Figura 38 Scenario dell'esperimento in Tabella 18. nel quale avvengono cinque trasmissioni contemporanee tutte dirette verso lo stesso sink: una trasmissione è generata da un nodo TelosB che trasmette pacchetti COAP con un rate di trasmissione di due pacchetti al secondo e altre quattro da parte di altrettanti nodi Iris che generano pacchetti ICMPv6 (ping). Il periodo di trasmissione dei pacchetti ping è stato configurato in modo da variare il traffico complessivo offerto.

Dalla tabella è evidente che finché il traffico offerto totale è inferiore a 10560bps il numero di ritrasmissioni risulta nullo. Viceversa il numero di ritrasmissioni diviene considerevole (superiore al 30%) se il rate di trasmissione totale supera i 40kbps. Si noti però che il meccanismo di ritrasmissione del COAP permette di avere in entrambi i casi una perdita di pacchetti nulla, a scapito di un aumento della latenza di rete che cresce all'aumentare del traffico offerto.

Figura 38 Scenario dell'esperimento in Tabella 18.

9.6 Risultati e considerazioni

Il modello analitico suddetto è stato implementato per tramite di fogli di calcolo che facilitano lo studio dell'effetto dei diversi parametri sulle metriche suddette.

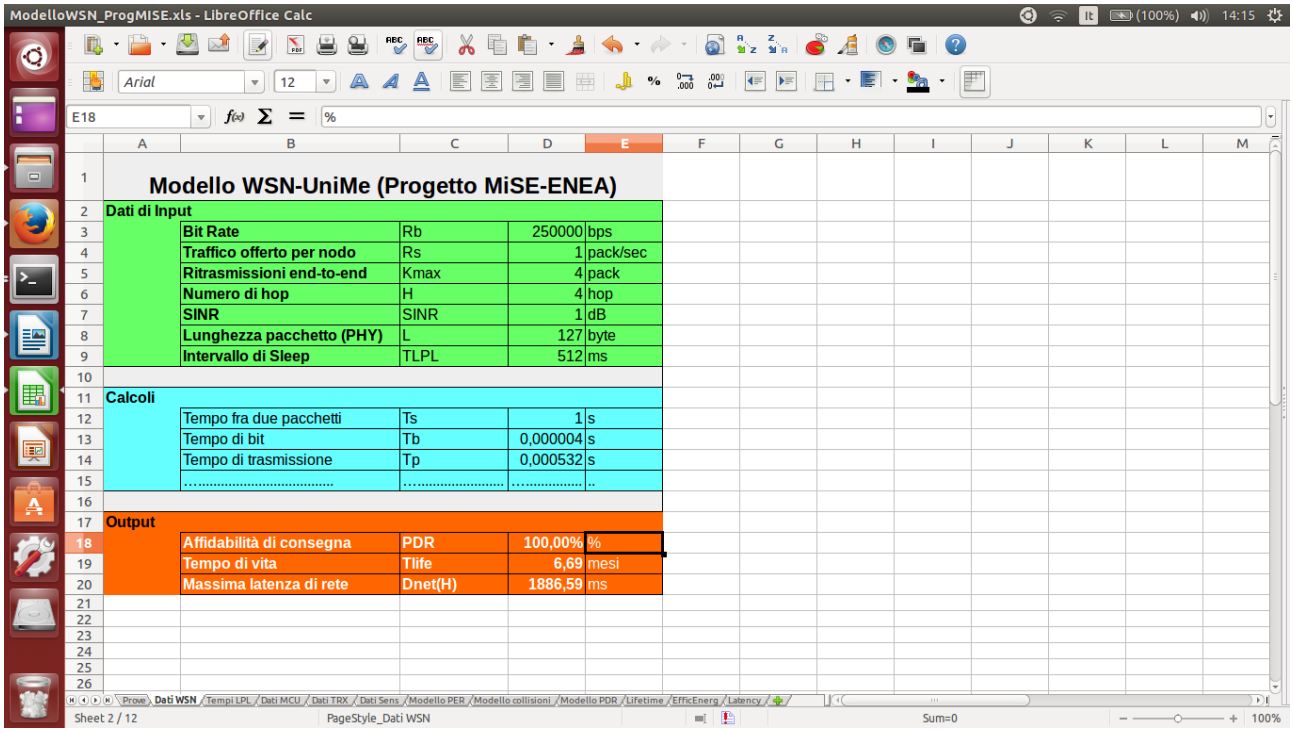


Figura 39 Modello WSN-UniMe, fogli di calcolo.

Inserendo i dati relativi ai mote TelosB sono stati quindi ricavati i risultati riportati in Tabella 20 relativi a diversi casi di studio di seguito analizzati e di interesse per il progetto.

In tutti i casi di studio si è considerato uno scenario di worst-case (SINR=0dB, PER=0.15), il numero di ritrasmissioni K_{max} è stato fissato a 4 e si è considerata una rete con H=4 hop, capace quindi di coprire una distanza massima di 200m.

Tabella 20 Risultati del modello per diversi casi di studio.

Parametri e Metriche	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
Bit Rate (Rb,bps)	250000	250000	250000	250000	250000
Traffico offerto (Rs, pack/sec)	0,00111	0,1	1	1	5
Ritrasmissioni end-to-end (Kmax)	4	4	4	4	4
Numero di hop (H)	4	4	4	4	1
SINR (dB)	0	0	0	0	0
Lunghezza pacchetto (L,byte)	127	127	127	40	40
Intervallo di sleep (TLPL,s)	17	1	0,512	0,512	0,1
Affidabilità di consegna (PDR, %)	**94,52%**	**94,52%**	**94,52%**	**99,87%**	**99,999%**
Tempo di vita (Tlife, mesi)	**184**	**16**	**5**	**7**	**1,5**
Massima latenza di rete (Dnet(H), s)	**63,4**	**3,7**	**1,9**	**1,9**	**0,13**

Caso 1) Smart metering

Il primo caso di studio analizzato (seconda colonna della Tabella 19) ha avuto come obiettivo quello di verificare la possibilità per uno smart metering basato su WSN di raggiungere un tempo di vita di 15 anni (ovvero 180 mesi) nell'ipotesi di trasmettere un pacchetto ogni 15 minuti.

Come è possibile osservare dalla seconda colonna della Tabella 20 tale tempo di vita è pienamente raggiungibile fissando un periodo di duty-cycle di 17 secondi, ottenendo una affidabilità di consegna anche in condizione di worst-case del 94.5% e un ritardo massimo di trasmissione di circa un minuto.

Caso 2) Sensori per applicazioni industriali

Il secondo caso di studio analizzato (terza colonna della Tabella 20) ha permesso di determinare il tempo di vita di un nodo sensore per applicazioni industriali nell'ipotesi di dover acquisire una misura al secondo e di trasmettere un pacchetto ogni 10 secondi (contenente quindi 10 misure), garantendo un ritardo massimo di trasmissione di 4 secondi. Come è possibile osservare le specifiche suddette sono raggiungibili impostando un periodo di duty-cycle di 1 secondo. Con la configurazione suddetta il tempo di vita risulta essere di 16 mesi.

Caso 3) Attuatori per applicazioni industriali

Il terzo caso di studio analizzato (quarta colonna della Tabella 20) ha permesso di determinare il tempo di vita di un nodo attuatore per applicazioni industriali nell'ipotesi di dover trasmettere un comando di attuazione al secondo, garantendo un ritardo massimo di trasmissione di 2 secondi. Come è possibile osservare le specifiche suddette sono raggiungibili impostando un periodo di duty-cycle di 512ms. Con la configurazione suddetta il tempo di vita risulta essere di 5 mesi.

Caso 4) Effetti della compressione

Il quarto caso di studio (quinta colonna della Tabella 20) mostra come la riduzione delle dimensioni del pacchetto (possibile ad esempio con tecniche di compressione dei dati) abbia un effetto positivo sia sul tempo di vita che sull'affidabilità di consegna.

A parità di periodo di duty-cycling del caso precedente, infatti, ridurre la dimensione del pacchetto da 127byte a 40byte permette di aumentare il tempo di vita del 40% (passando da 5 mesi a 7 mesi) oltre che di avere una affidabilità di consegna del 99.9% (al posto del precedente 94.5%).

Caso 5) Segnalazione allarmi

L'ultimo caso di studio analizzato (ultima colonna della Tabella 20) a differenza dei casi precedenti considera una rete a singolo hop (H=1) al fine di ottenere bassi ritardi di trasmissione, condizione necessaria nel caso di segnalazione di allarmi o di controllo di attuatori critici. La tabella mostra che fissando un periodo di duty-cycling di 100ms è possibile garantire la trasmissione di 5 pacchetti al secondo con una trascurabile probabilità di perdita (la PDR risulta del 99.999%) e un ritardo massimo atteso di circa

129ms. Ovviamente ciò a scapito del tempo di vita che in tali condizioni critiche si riduce ad appena un mese e mezzo.

10 Prototipo di una WSN per applicazione M2M

In questo Capitolo è descritto il prototipo di una WSN per applicazioni M2M realizzato nell'ambito dell'attività e utilizzato per i test sperimentali. In particolare saranno descritti tutti i componenti del prototipo dall'hardware, ai protocolli, al firmware, fino al software di interfaccia.

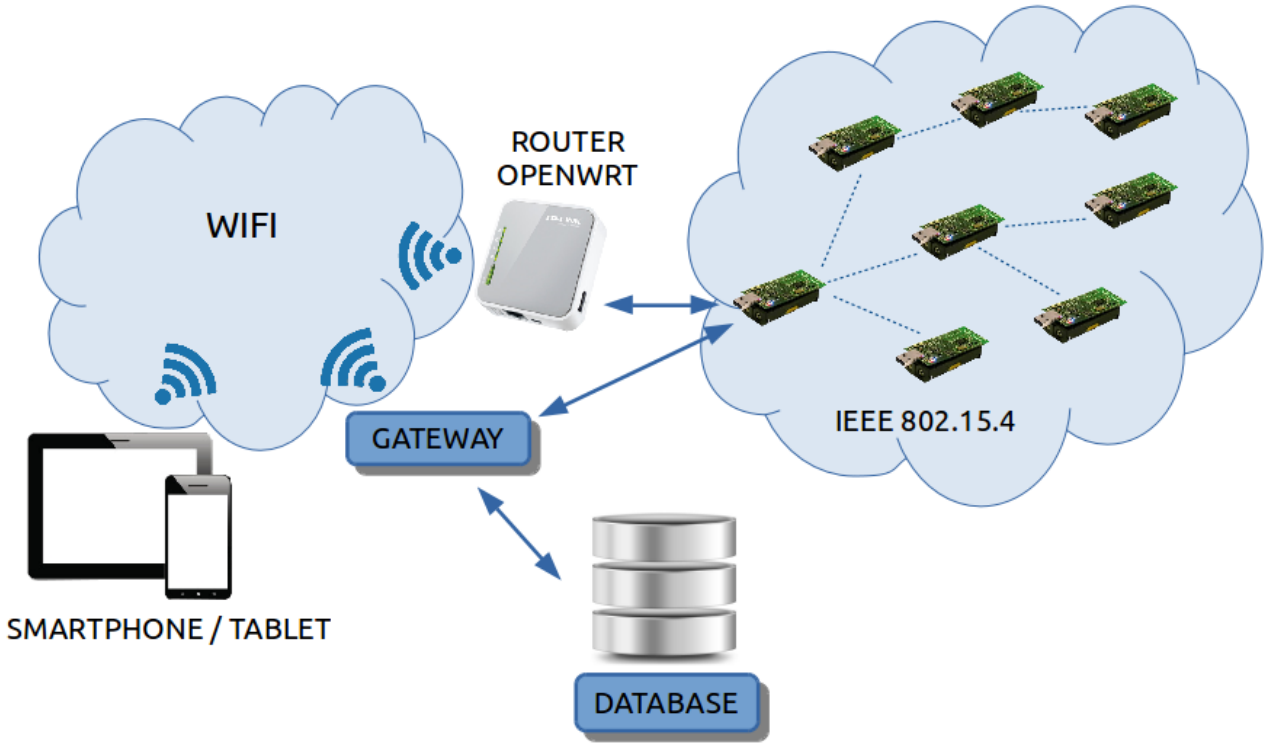


Figura 40 Architettura di WSN per applicazioni M2M.

L'architettura del prototipo realizzato è illustrata in Figura 40 ed è composta da

- n. 8 nodi sensori (n.5 Iris e n. 3 TelosB);
- un router WiFi (TP-Link modello TL-MR3020 su cui è stato installato OpenWRT 15.05);
- un gateway costituito da un comune PC con S.O. Linux Ubuntu 14.04 su cui è stato installato TinyOS 3;
- un database MySQL per l'archiviazione dei dati;
- il software di gestione denominato EasyWSN.

Di seguito sono forniti i necessari dettagli implementativi sui componenti suddetti.

10.1 Nodi Sensori

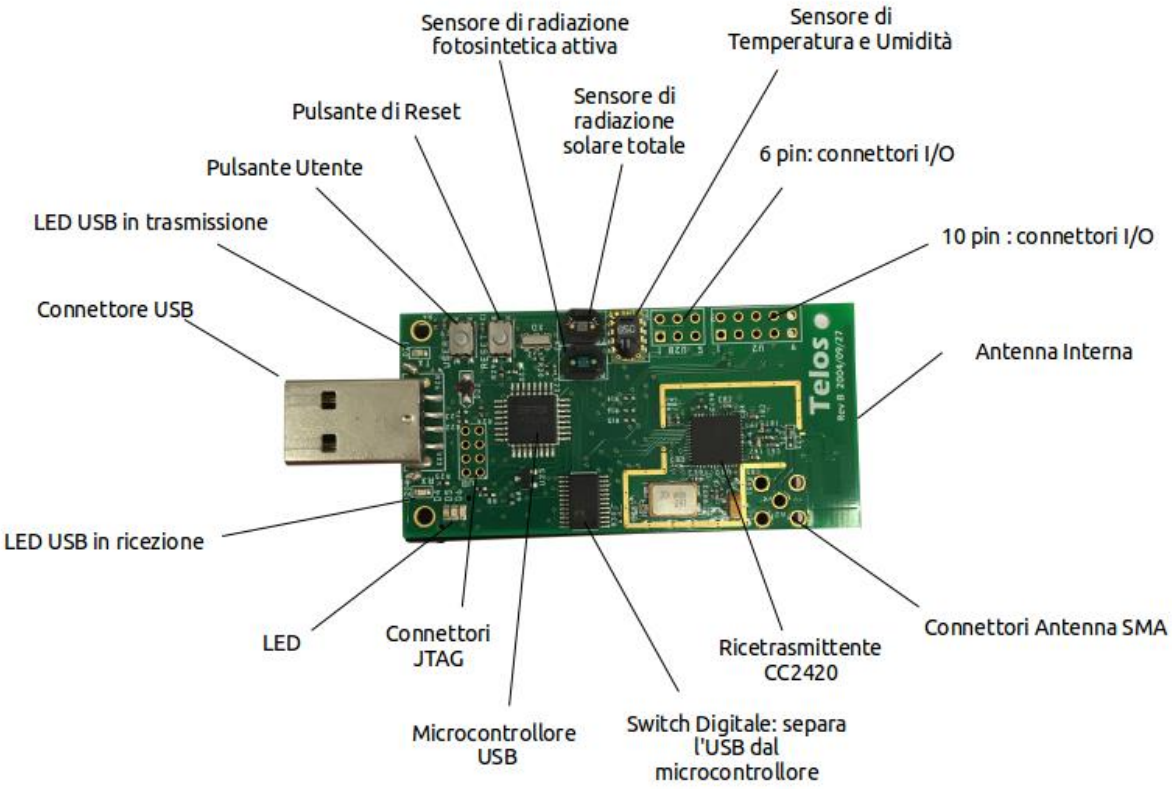


Figura 41 TelosB: sensori, attuatori e interfacce.

Sulla base dell'analisi condotta nel Capitolo 4 sono stati scelti come nodi sensori per la realizzazione del testbed i mote Iris e TelosB, entrambi prodotti dalla MEMSIC.

I nodi sono equipaggiati con i seguenti sensori e attuatori:

- sensori e attuatori presenti sui TelosB (Figura 41)
  - n.2 sensori di luminosità: Hamamatsu S1087 (320nm-730nm) e Hamamatsu S1087-01 (320nm-1100nm);
  - n.1 sensore di umidità e temperatura: Sensirion SHT11
    - range umidità relativa RH 0-100% con risoluzione 0.03% RH;
    - range di temperatura -40°C-124°C con risoluzione 0.01°C;

Sono inoltre presenti dei connettori con le seguenti interfacce di espansione:

- n.6 ADC a 12-bit (ingressi analogici utili per interfacciare ulteriori sensori);
- n.4 I/O digitali (utili per pilotare relè e altri dispositivi ON/OFF e/o per generare segnali PWM);
- n.2 DAC a 12-bit (utili per generare segnali di riferimento e/o di controllo nel range 0-3V);
- n.1 interfaccia I2C per la comunicazione con altri sensori/attuatori dotati di interfaccia seriale.

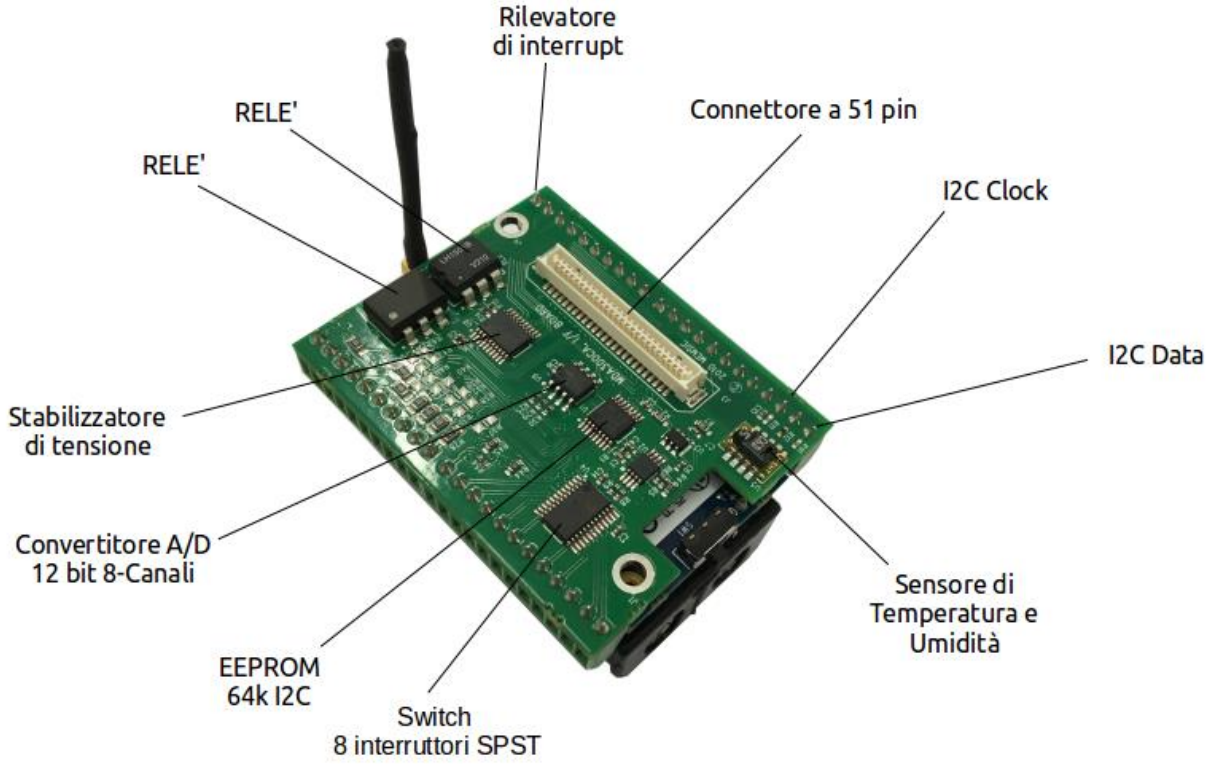


Figura 42 Dettaglio Scheda MDA300 connessa a mote IRIS.

Sensori e attuatori su IRIS (sensor-board MDA300, Figura 42)

- n.1 sensore di temperatura e umidità SHT11
  - range umidità relativa RH 0-100% con risoluzione 0.03% RH;
  - range di temperatura -40°C-124°C con risoluzione 0.01°C;
- n.2 canali relè (uno normalmente aperto e l'altro normalmente chiuso, 100V 150mA);
- n.11 ADC a 12-bit (configurabili come single-ended o differenziali);
- n.6 canali I/O digitali;
- n.1 rilevatore di interrupt;
- n.1 EEPROM da 64K utile per la calibrazione dei sensori;
- n.1 interfaccia I2C per la comunicazione con altri sensori o attuatori dotati di interfaccia seriale;

I sensori e gli attuatori suddetti sono stati scelti al fine di permettere il più ampio spettro possibile di applicazioni finalizzate all'efficientamento energetico in ambito industriale, dal monitoraggio ambientale fino al controllo di sistemi di condizionamento (HVAC) e di illuminazione.

Nell'ambito dell'attività sono stati inoltre identificati altri sensori, attuatori e circuiti integrati interfacciabili con i mote suddetti per tramite degli ADC e dell'interfaccia I2C già presente nelle sensor-board selezionate.

In particolare si evidenzia il circuito integrato ADE7753, in grado di misurare potenza elettrica nelle sue diverse componenti (attiva, reattiva e apparente).

Lo stack protocollare utilizzato dai nodi sensori è illustrato in Figura 43.

COAP	APP
UDP	TRASPORTO
RPL	
IPv6	RETE
6LowPAN	
BoxMAC2	MAC
IEEE 802.15.4	PHY

Figura 43 Stack protocollare utilizzato dai mote.

I protocolli suddetti sono stati già illustrati nei Capitoli 3 e 5.

La programmazione dei nodi avviene collegando il mote al Gateway per tramite della porta USB (nel caso dei TelosB direttamente, mentre nel caso degli Iris, non disponendo questi di un connettore USB, occorre la scheda di programmazione MIB520) e compilando il codice corrispondente al firmware del nodo.

I firmware principali realizzati sono tre:

1. *PppRouter* usato per la programmazione del sink (da collegare al gateway e/o al router WiFi);
2. *NewCoap* usato per la programmazione dei nodi Telosb;
3. *NewUDP* usato per la programmazione dei nodi Iris.

I relativi codici sorgente sono stati predisposti secondo le convenzioni previste da TinyOS. In particolare ogni firmware prevede un Makefile contenente i FLAG necessari per la configurazione dei parametri di funzionamento.

A titolo di esempio di seguito è mostrato il Makefile relativo al NewCoap dove sono evidenti i FLAG per specificare i parametri di configurazione di tutto lo stack protocollare, dai parametri del transceiver, alle URI del COAP desiderate, passando dai parametri dell'algoritmo di routing (RPL), del protocollo IPv6 e delle tecniche di duty-cycling (LPL).

```
COMPONENT=NewCoapC

#Configurazione Canale, Potenza in TX (0 dBm)
CFLAGS += -DCC2420_DEF_CHANNEL=26
CFLAGS += -DCC2420_DEF_RFPOWER=31

#Configurazione RPL
PFLAGS += -DRPL_ROUTING -DRPL_STORING_MODE -I$(TINYOS_OS_DIR)/lib/net/rpl
PFLAGS += -DRPL_OF_0=1
PFLAGS += -DRPL_OF_MRHOF=0

#Configurazione LPL
#CFLAGS += -DLOW_POWER_LISTENING
#CFLAGS += -DLPL_SLEEP_INTERVAL=512

#Assegnamento prefisso IPv6 (statico)
PFLAGS += -DIN6_PREFIX=\"fec0::\"

#Configurazione CoAP Server
CFLAGS += -DCOAP_SERVER_ENABLED
CFLAGS += -DCOAP_SERVER_PORT=5683L
CFLAGS += -DMAX_URI_LENGTH=14

#Configurazione CoAP Client
#CFLAGS += -DCOAP_CLIENT_ENABLED
#CFLAGS += -DCOAP_CLIENT_PORT=61617L
#Assegnamento indirizzo IPv6 Client
CFLAGS += -DCOAP_CLIENT_DEST=\"fec0::100\"

#Set delle possibili risorse attivabili
#CFLAGS += -DCOAP_RESOURCE_TEMP
#CFLAGS += -DCOAP_RESOURCE_HUM
#CFLAGS += -DCOAP_RESOURCE_VOLT
#CFLAGS += -DCOAP_RESOURCE_ALL
#CFLAGS += -DCOAP_RESOURCE_KEY
#CFLAGS += -DCOAP_RESOURCE_SLEEP_LPL
#CFLAGS += -DCOAP_RESOURCE_OBS_TIME
#CFLAGS += -DCOAP_RESOURCE_PID
#CFLAGS += -DCOAP_RESOURCE_RELE
#CFLAGS += -DCOAP_RESOURCE_ROUTE
#CFLAGS += -DCOAP_RESOURCE_CK
CFLAGS += -DCOAP_RESOURCE_LED

#Opzioni CoAP (Block-Wise, Well-Known, Observe, Debug Payload)
#CFLAGS += -DWITHOUT_BLOCK
#CFLAGS += -DWITHOUT_WELLKNOWN
#CFLAGS += -DCOAP_RESOURCE_ETSI_IOT_OBSERVE
CFLAGS += -DWITHOUT_OBSERVE
CFLAGS += -DSHORT_ERROR_RESPONSE

#PDU Size
CFLAGS += -DCOAP_MAX_PDU_SIZE=127
#Content type
CFLAGS += -DCOAP_CONTENT_TYPE_PLAIN
#PreACK (ms)
CFLAGS += -DCOAP_PREACK_TIMEOUT=1000

GOALS = blip coap
CFLAGS += -I.
TINYOS_ROOT_DIR?=../..
include $(TINYOS_ROOT_DIR)/Makefile.include
```

Editati i FLAG, la compilazione del firmware avviene per tramite del comando

make <PLATFORM> install,<ID_MOTE> bsl,/dev/tty<USBPort>

dove <PLATFORM> identifica il tipo di nodo che si vuole programmare (iris o telosb), <ID_MOTE> deve essere sostituito da un numero intero che identificherà l'indirizzo IPv6 assegnato al nodo (il cui prefisso sarà fec0::), mentre <USBPort> è il numero di porta USB a cui è stato connesso il mote o il programmatore.

Ad esempio, il comando

make telosb install,3 bls,/dev/ttyUSB0

il cui output è illustrato in Figura 44, permette di programmare tramite la porta seriale ttyUSB0 un mote TelosB assegnandogli l'identificativo 3 (a cui corrisponderà l'indirizzo IPv6 fec0::3).

```
[INFO] script
           45122 bytes in ROM
            6422 bytes in RAM
[INFO] size (toolchain):
   text    data     bss     dec     hex filename
  45316      76    6348   51740    ca1c build/telosb/main.exe
[INFO] generating symbol table
[INFO] generating listing
[INFO] creating ihex file
[INFO] writing TOS image
[INFO] writing TOS buildinfo
[INFO] running the wiring check
[INFO] setting the node id to 3
[INFO] found mote on /dev/ttyUSB0 ("using bsl,/dev/ttyUSB0")
[INFO] installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out-3
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
45392 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out-3 build/telosb/main.ihex.out-3
```

Figura 44 Output visualizzato dal terminale.

Nella Tabella 21 sono infine riportate le principali URI proposte per il progetto MiSE-ENEA.

Tabella 21 principali URI previste per il progetto MiSE-ENEA.

#	FLAG	URI	DESCRIZIONE
1	COAP_RESOURCE_TEMP	/st	Sensore di Temperatura
2	COAP_RESOURCE_HUM	/sh	Sensore di Umidità
3	COAP_RESOURCE_VOL	/sv	Sensore di Tensione (ADC)
4	COAP_RESOURCE_ALL	/r	Sensori di Temperatura, Umidità e di Tensione
5	COAP_RESOURCE_ROUTE	/rt	Tabella di Routing
6	COAP_RESOURCE_LED	/l	LED
7	COAP_RESOURCE_ETSI_IOT_OBSERVE	/obs	Rende una risorsa osservabile
8	COAP_RESOURCE_RELE	/rele	Controllo Relè
9	COAP_RESOURCE_PID	/pid	Parametri controllore PID
10	COAP_RESOURCE_SLEEP_LPL	/sleeplpl	Periodo duty-cycle
11	COAP_RESOURCE_CK	/ck	Chiave di cifratura AES
12	COAP_RESOURCE_OBS_TIME	/obstime	Periodo di campionamento

Mentre le URI 1-7 della Tabella 21 sono già presenti nell'implementazione COAP di TinyOS, le URI 8-12 sono state realizzate da UniMe nell'ambito dell'attività al fine di tener conto delle diverse esigenze delle comunicazioni M2M per applicazioni di controllo industriale e di smart metering.

In particolare:

- la URI *sleeplpl* permette di definire il tempo in cui un nodo rimane in stato di inattività e quindi di agire sul duty-cycle che, come visto nel Capitolo 9, risulta essere il principale parametro di progettazione per ottenere il miglior compromesso fra latenza e consumi energetici;
- la URI *pid* può essere utilizzata nel caso di sistemi di controllo retroazionati basati su controllori PID (Proportional-Integral-Derivative controller). In particolare la URI permette di definire le tre costanti (KP, KI, KD) da cui dipendono le prestazioni di un sistema di controllo retroazionato;
- la URI *obstime* permette di modificare il tempo di campionamento con cui vengono acquisiti i dati;
- la URI *rele* permette di comandare relè o altri azionamenti ON/OFF interfacciati ai nodi di attuazione;
- la URI *ck* permette di impostare e modificare la chiave di cifratura AES.

Altre URI utili in ambito industriale sono state definite ma non ancora implementate al momento della stesura del presente lavoro.

Si ritiene importante evidenziare che il numero di URI che possono essere presenti contemporaneamente sul dispositivo dipende strettamente dalla memoria ROM disponibile nel nodo sensore. In particolare le limitate risorse dei nodi TelosB permettono di attivare al più due URI contemporaneamente dato che la gestione dei protocolli IPv6 e RPL richiede circa 36kB sui 48kB disponibili.

In Tabella 22 è riportata la quantità di memoria necessaria per ogni URI.

Tabella 22 Occupazione in memoria delle diverse URI.

URI	ROM (bytes)
/st	4944
/sh	5022
/sv	7860
/r	10352
/rt	3758
/l	3644
/obs	1858

10.2 Router

Figura 45 Router WiFi TP-Link M3020 .

Nell'ambito dell'architettura proposta si è deciso di utilizzare un router WiFi opportunamente configurato per permettere la comunicazione fra i nodi sensori e un qualsiasi dispositivo con interfaccia WiFi IEEE802.11g/n. In particolare il router permette di interfacciare la rete di sensori con smartphone e tablet. Tale componente si è reso necessario poiché i dispositivi portatili attuali non supportano nativamente il protocollo 802.15.4 e pertanto per comunicare con i nodi sensori sarebbero necessari appositi dongle USB 802.15.4. Ciò implicherebbe un costo addizionale per ogni dispositivo portatile; inoltre tali dongle non sono sempre supportati dai S.O. mobile (i driver dei dongle sono spesso rivolti al solo S.O. Windows).

Viceversa, nell'architettura proposta, un unico router WiFi (nello specifico il TP-Link MR3020 dal costo di circa 20€), funge da "ponte" fra le reti 802.15.4 e 802.11 permettendo l'accesso alla rete di sensori come ad una qualsiasi altra rete WiFi. Il router è stato predisposto in modo da autoconfigurare un qualsiasi dispositivo portatile purché questo abbia le necessarie credenziali di accesso alla rete.

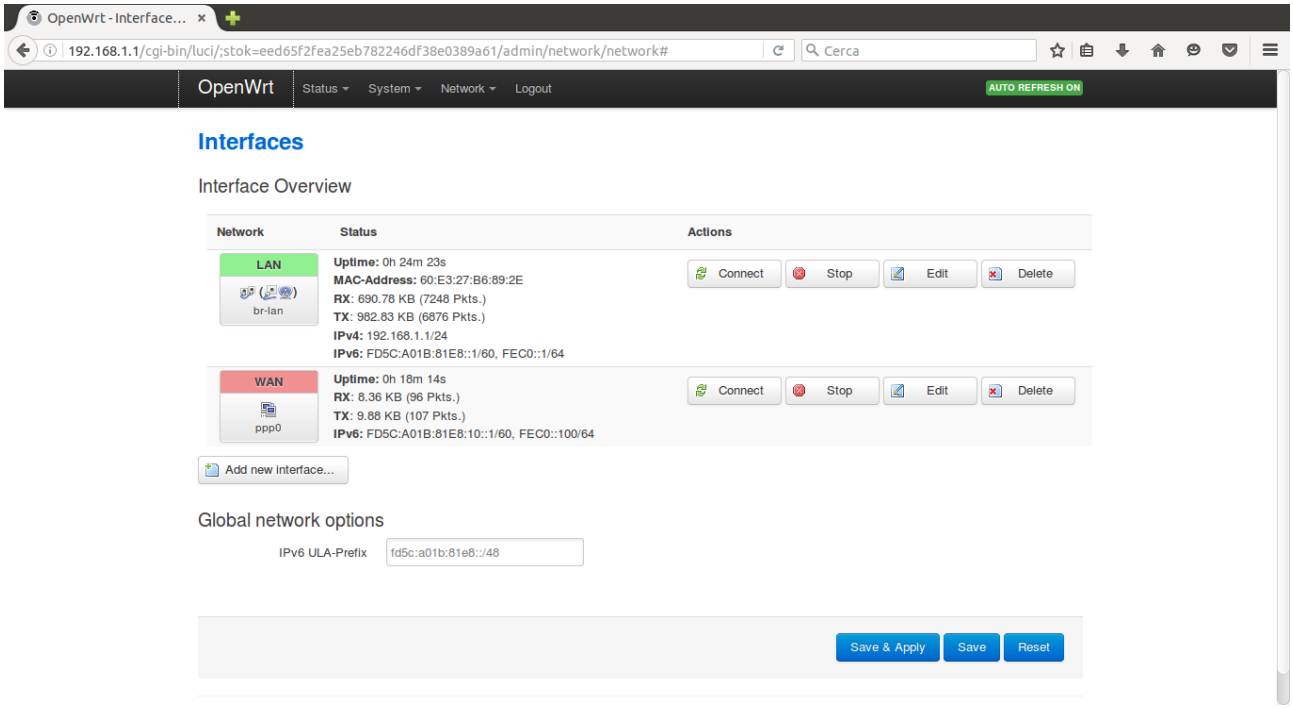


Figura 46 Web GUI di configurazione del router.

Il router può essere configurato mediante l'interfaccia grafica messa a disposizione da OpenWRT (LuCI, mostrata in Figura 46). In particolare è possibile abilitare meccanismi di sicurezza tipici delle reti WiFi (es. WPA2) e autorizzare o meno l'accesso da parte di altre reti mediante firewall. Per tramite del Router i dispositivi autorizzati possono visualizzare lo stato della rete, ottenere informazioni sulle grandezze fisiche misurate dai nodi sensori e fornire comandi di attuazione senza che sia necessario l'installazione di alcun software specifico ma semplicemente utilizzando un comune browser.

10.3 Test comunicazioni M2M

L'architettura suddetta permette comunicazioni fra utenti e dispositivi oltre che fra due dispositivi direttamente mediante lo scambio dei comandi previsti dal protocollo COAP (in particolare GET e PUT).

Ai fini di verificare il corretto funzionamento del testbed realizzato si è utilizzato il browser Mozilla-Firefox che dispone di un plugin denominato Copper [239] in grado di abilitare un qualsiasi PC con S.O. Windows o Linux all'uso del protocollo M2M COAP (si veda la Figura 47).

Per tramite di un browser e del protocollo COAP un utente può ottenere informazioni sullo stato dei sensori e/o comandare attuatori navigando attraverso i link come farebbe con una comune pagina web simulando così una comunicazione M2M.

A titolo di esempio in Figura 47 è possibile vedere il plugin Copper in azione a seguito dell'interrogazione tramite browser di un nodo sensore. Più precisamente l'utente, dopo aver digitato l'indirizzo IPv6 del nodo di interesse (nell'esempio fec0::2) e aver premuto il pulsante Discover, riceve l'elenco delle risorse presenti sul dispositivo remoto, ovvero un elenco di link (URI) corrispondenti a sensori/attuatori e parametri di configurazione del dispositivo.

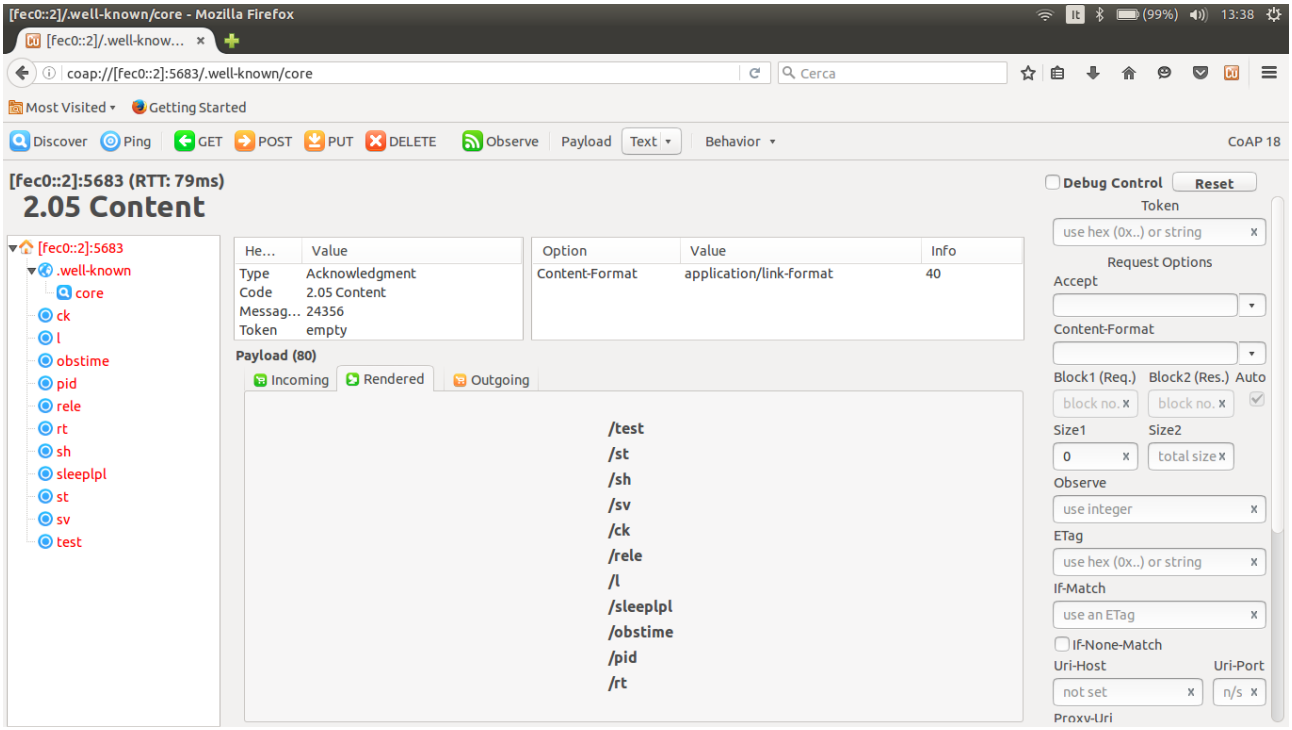


Figura 47 Plug-in Copper per Mozilla-Firefox.

In alternativa al Copper, su dispositivi con S.O. Android, è possibile utilizzare l'app Aneska [240] (la cui interfaccia è riportata in Figura 48).

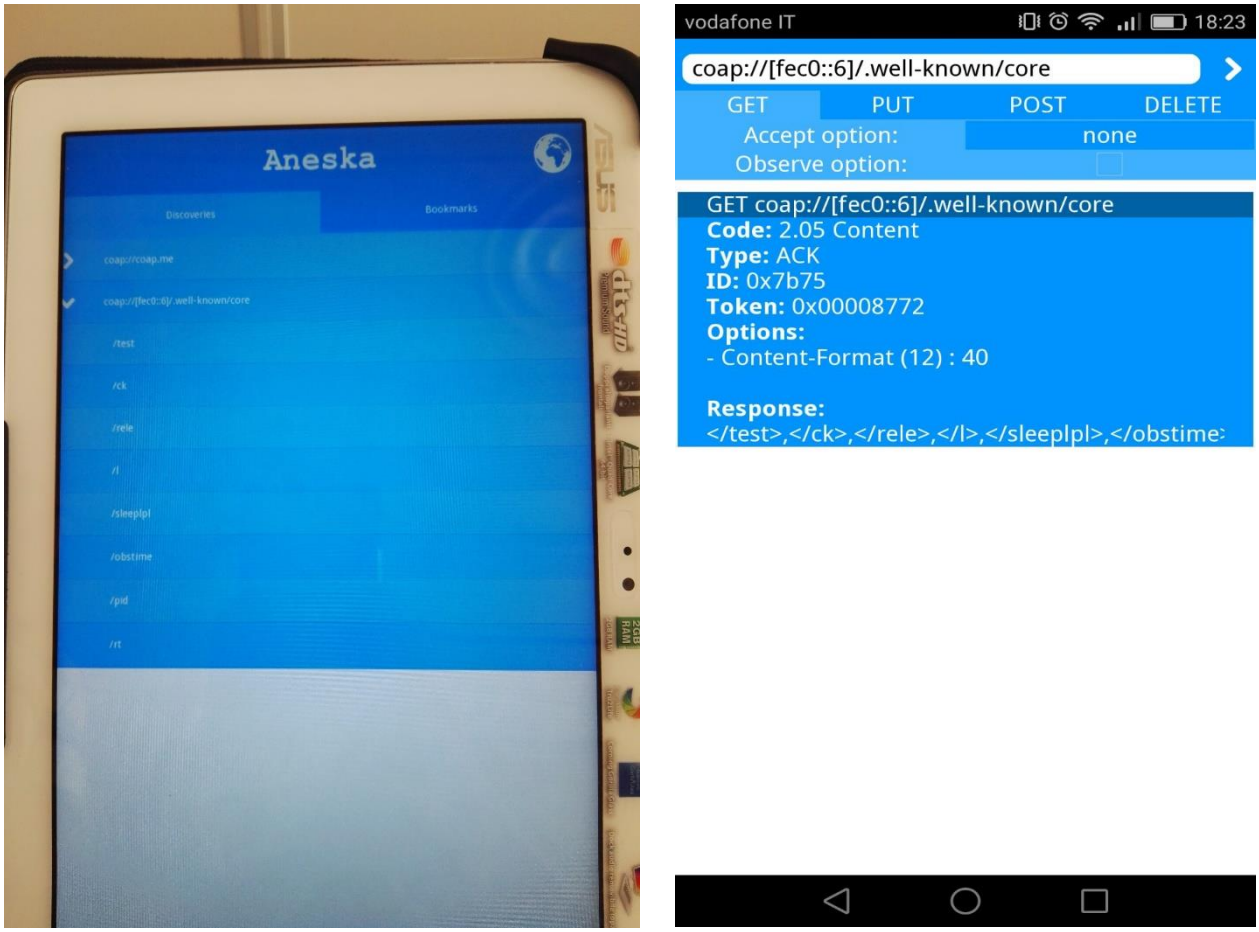


Figura 48 L'app Aneska per dispositivi mobile con S.O. Android.

In entrambe le applicazioni (Copper o Aneska), l'utente può vedere lo stato di un sensore selezionando l'URI desiderata e premendo il pulsante GET; inoltre è possibile inviare un valore di attuazione premendo il pulsante PUT.

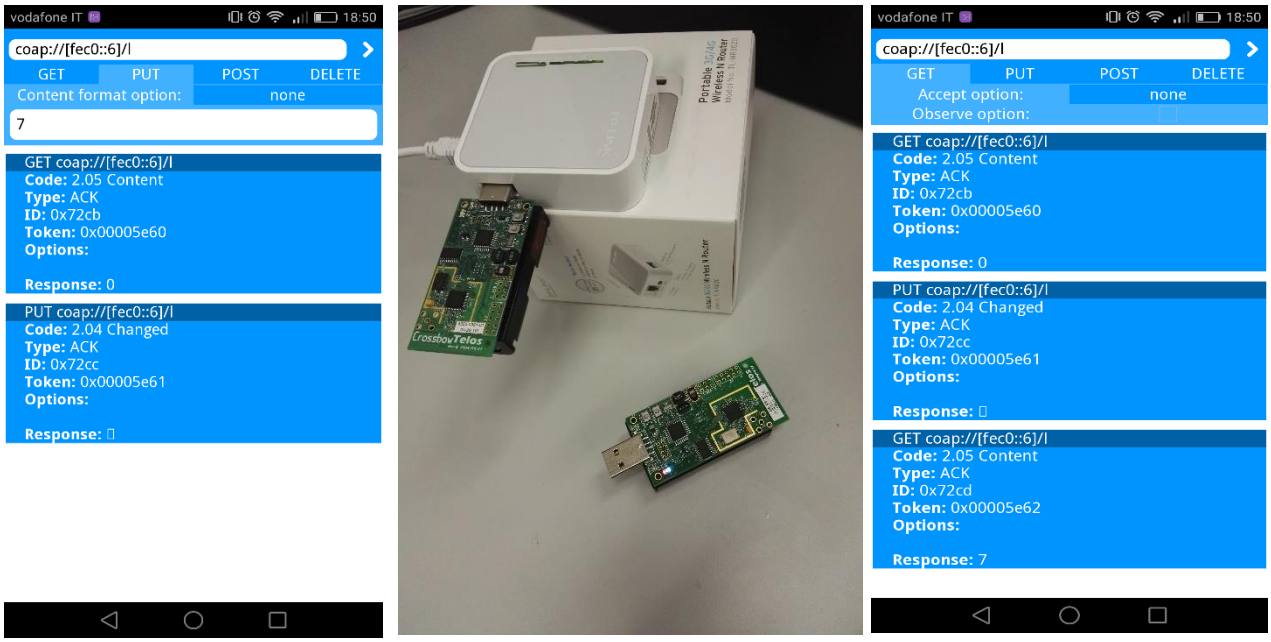


Figura 49 Esecuzione di GET e PUT mediante Aneska.

In Figura 49 è riportato un esempio dove per tramite di Aneska si eseguono tre operazioni, rispettivamente GET, PUT, GET, sulla risorsa /l relativa ai tre led presenti sui nodi TelosB.

Più precisamente, nell'esempio di Figura 49 l'utente esegue dapprima una GET ottenendo il valore 0 (ad indicare che tutti i led sono spenti), successivamente digitando il valore 7 e premendo il pulsante PUT i led vengono accesi, in fine rieseguendo una GET l'utente ottiene conferma dello stato dei nodi (il valore di ritorno della seconda GET è infatti 7).

10.4 Gateway e DataBase

Il Gateway è costituito da un comune PC con S.O. Linux e permette la programmazione dei nodi sensori (come descritto nella sezione Nodi Sensori del presente Capitolo) oltre che l'esecuzione del software di gestione della rete (descritto nella prossima Sezione) al fine della configurazione della rete. Il Gateway può interfacciarsi alla rete direttamente o per tramite del router WiFi. Inoltre altre interfacce di comunicazione, ad esempio verso reti cellulari 3G/4G, possono essere facilmente aggiunte.
Nell'ambito del testbed realizzato il Gateway è stato usato anche per ospitare un database MySQL atto alla memorizzazione dei dati provenienti dalla rete di sensori ma il sistema può essere configurato per operare con un database remoto (su Cloud).
L'architettura del database è stata ideata per essere la più semplice possibile in modo da permetterne l'interrogazione anche da altri software oltre quello appositamente sviluppato per il progetto MiSE-ENEA.
Di seguito è fornita una descrizione della struttura del DataBase e delle tabelle che permettono la configurazione della rete e la memorizzazione dei dati acquisiti dai nodi.
La tabella principale del DataBase è denominata *ConfigurazioneWSN* e contiene tutti i dati di configurazione dei nodi della rete. Ogni entry, ovvero ogni riga della tabella, contiene le informazioni su un nodo appartenente alla rete. Il numero di entry della tabella coincide con il numero di nodi. Le entry hanno la seguente struttura:

- *id*: è l'identificativo del nodo e coincide con l'ultima cifra dell'indirizzo IPv6 assegnato al nodo;
- *nome logico*: rappresenta una descrizione che si può associare al nodo, in modo da identificare quest'ultimo più facilmente (Esempio: nodo stanza laminazione; nodo sala riunioni; ecc.);
- *tipo*: costituisce il tipo di nodo e/o la corrispondente scheda di acquisizione, se presente;

- *periodo di campionamento*: è l'intervallo di tempo tra due valori consecutivi acquisiti dal sensore;
- *periodo di sleep*: rappresenta il periodo del duty-cycle;
- *coordinate*: rappresentano le coordinate x e y relative al posizionamento del nodo su specifica planimetria.

I dati acquisiti dai nodi sono memorizzati in una serie di tabelle identificate dall'id del nodo stesso e dal mese e anno dell'acquisizione (i nomi delle tabelle sono del tipo DatiNodo_id_mese_anno).
Si è deciso di non riportare tutti i dati in un'unica tabella al fine di ridurre i tempi necessari per l'accesso ai dati (query di scrittura/lettura/ricerca).
Tali tabelle sono create automaticamente a partire dalla ricezione del primo pacchetto (la suddivisione in mesi è trasparente all'utente).
Oltre ai dati provenienti dai sensori ogni tabella contiene:

- *id*: è l'identificativo del nodo che ha trasmesso il pacchetto e coincide con l'id inserito nella tabella ConfigurazioneWSN;
- *packettype*: è un identificativo che serve per distinguere il tipo di pacchetto e come esso debba essere interpretato;
- *periodo di campionamento*: il periodo di campionamento del sensore;
- *campioni*: il numero di campioni contenuti in ogni pacchetto;
- *timestamp*: istante in cui un pacchetto viene ricevuto, espresso in coordinate UTC. Dal timestamp e dal periodo di campionamento è possibile ricavare il tempo di ricezione dei singoli campioni necessario per creare i grafici temporali delle grandezze fisiche acquisite;
- *per*: la packet-error-rate, ovvero la percentuale di pacchetti persi;
- *batteria*: il livello di tensione della batteria al momento dell'acquisizione;
- *rssi*: il livello di potenza del segnale ricevuto dal nodo (RSSI).

Un'altra tabella del database denominata *LeggiAttuazione* contiene tutte le informazioni necessarie per definire le leggi di controllo eseguite dai nodi attuatori.
Ogni campo della tabella contiene:

- *id*: l'identificativo del nodo;
- *tipoAttuatore*: consente di determinare il tipo di attuazione da implementare e la legge di attuazione ad essa associata;
- *canale*: ogni nodo può gestire due canali differenti per l'attuazione con due differenti leggi di attuazione;
- *descrizione*: può identificare l'area in cui è collocato l'attuatore e/o specificare il tipo di attuazione;
- *nodi*: contiene il (o gli) id dei nodi che intervengono nella legge di attuazione ;
- *valoreRif*: è un valore di riferimento per la grandezza fisica da controllare a cui la legge di attuazione deve convergere (se si tratta di un controllo a catena chiusa);
- altri parametri di configurazione della legge di attuazione.

La tabella GestioneUtenti contiene informazioni sugli utenti, come:

- *Nome*: nome dell'utente;
- *Cognome*: cognome dell'utente;
- *user_name*: nome identificativo assegnato all'utente. Viene utilizzato dall'utente per effettuare il login;
- *email*: indirizzo e-mail utilizzato per la messaggistica d'errore. Il sistema prevede che quando un nodo non invia pacchetti per un intervallo di tempo predefinito, venga inviata un e-mail all'indirizzo specificato, contenente l'id del nodo e il messaggio d'errore per avvertire dell'anomalia;
- *Gruppo*: specifica i diritti dell'utente (Amministratore, Operatore, Utente standard, Ospite)

Le password non sono salvate in chiaro sul database per motivi di sicurezza ma su un file opportunamente cifrato. Più precisamente le singole password sono memorizzate in md5 e l'intero file può essere cifrato con AES. La chiave di cifratura è nota solo all'Amministratore che è l'unico a poter aggiungere utenti al sistema.

10.5 Software di gestione: interfaccia grafica e front-end utente

I dati presenti sul database possono essere visualizzati ed elaborati mediante un software di gestione, denominato EasyWSN, sviluppato dall'Università di Messina in collaborazione con la ditta WEVA [WEVA]. Lo stesso software permette anche di configurare la rete di sensori e di monitorarne lo stato e/o fornire comandi di attuazione mediante il protocollo M2M COAP. Inoltre, il software permette all'utente di definire le leggi di attuazione necessarie per comandare gli attuatori della rete.
Di seguito vengono illustrate le funzionalità del software. c0076à-lllòfdr
In Figura 50 è illustrata la finestra principale del software che include una toolbar (in alto), un'area di disegno (al centro) e una barra di stato (in basso). La toolbar del software EasyWSN permette tramite pulsanti di richiamare tutte le azioni necessarie: dalla configurazione della rete alla visualizzazione ed elaborazione dei dati; l'area di disegno permette la visualizzazione di informazioni e l'interazione con i nodi; la barra di stato contiene due icone che forniscono informazioni sullo stato della connessione al router WiFi e al sink della WSN.

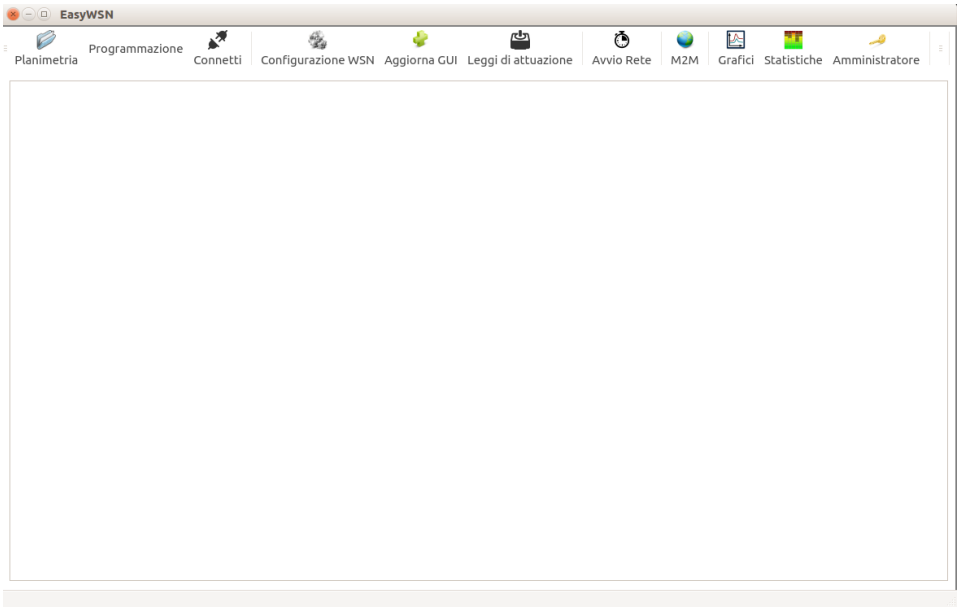


Figura 50 Finestra principale EasyWSN.

I pulsanti della toolbar, il cui particolare è mostrato in Figura 51 sono ordinati da sinistra verso destra in accordo con il flusso di utilizzo del software che si articola nelle seguenti fasi.

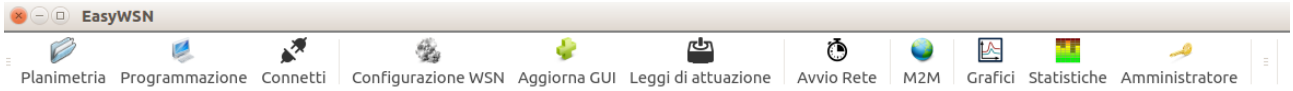


Figura 51 ToolBar principale del software EasyWSN.

Fase I – Set-Up della rete:
Definizione dello scenario: il pulsante "*Planimetria*" della toolbar principale mostrata in Figura 51 permette di importare una immagine da utilizzare come sfondo per rappresentante l'area in cui i nodi sono stati realmente installati (es. zone di produzione industriale, uffici, magazzini ecc.). Il formato attualmente supportato è .svg (un formato vettoriale che permette quindi alle immagini di essere ingrandite o rimpicciolite), ma sono possibili altri formati (es. jpg). In Figura 52 è mostrata la finestra per la selezione dell'immagine da impostare come sfondo dell'area di disegno.

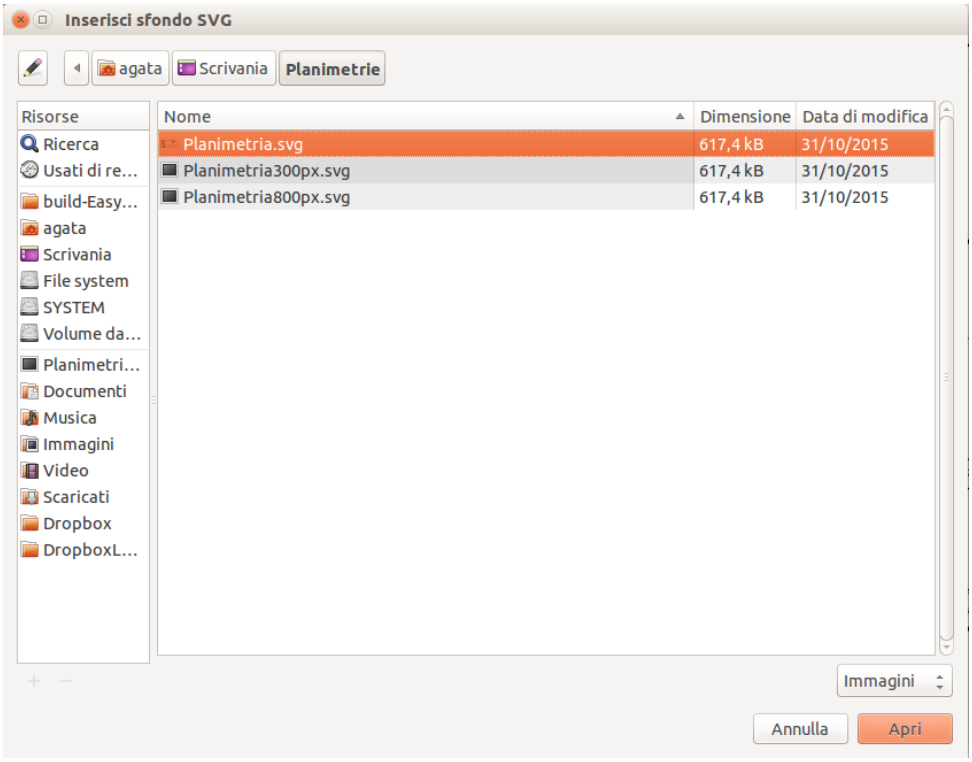


Figura 52 Planimetria – Selezione file.

Dopo la selezione della planimetria l'interfaccia grafica appare come in Figura 53.

L'inserimento della planimetria permette il posizionamento dei nodi dopo la loro configurazione. In tale fase non compare ancora nessun nodo nell'interfaccia poiché l'individuazione dei nodi connessi alla rete M2M avviene in modo automatico, così come descritto nella *Fase II.*

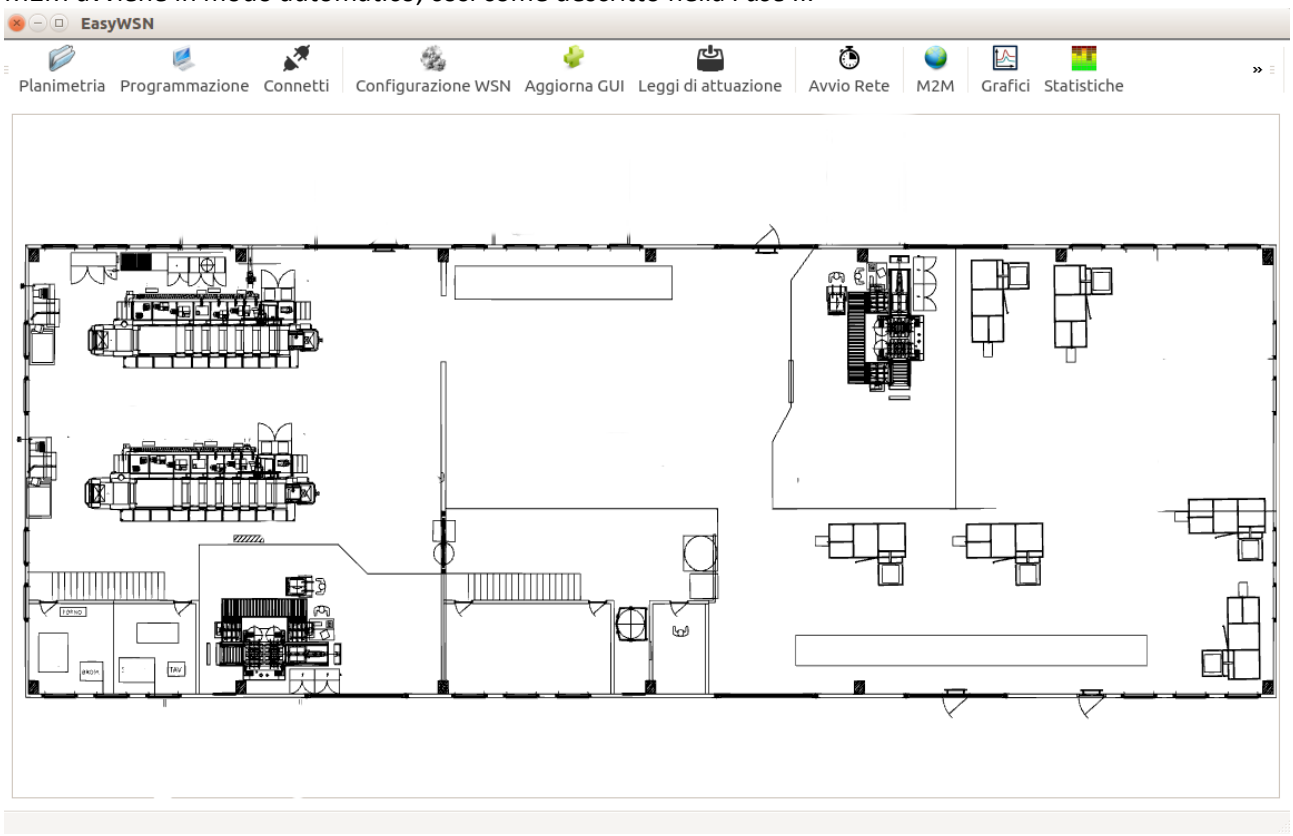


Figura 53 Finestra principale – Fase I.

Programmazione: il pulsante *Programmazione* apre una shell del sistema operativo che permette la programmazione dei dispositivi e il download del firmware sulla base di quanto descritto nella sezione Nodi Sensori del presente Capitolo.
Posizionamento e Cablaggi: a questo punto i dispositivi programmati possono essere fisicamente posizionati nell'ambiente reale. In genere i nodi sono alimentati a batterie e non richiedono cablaggi se non per collegare gli attuatori ai sistemi di controllo (l'esempio tipico è quello del collegamento di un relè del mote ad un sistema di azionamento presente nell'ambiente industriale). Per alcuni nodi però potrebbe essere preferibile l'alimentazione da rete (tramite trasformatore). Tale circostanza si potrebbe verificare qualora fossero presenti attuatori o sensori particolarmente power-hungry o nodi concentratori per i quali fosse necessario garantire una connettività h24.
Alimentazione e Accensione: i dispositivi vengono accesi (collegando le batterie e/o mediante pulsante di accensione presente sul mote) si ha quindi una fase di bootstrap della durata di pochi minuti in cui i nodi avviano il loro sistema operativo, mandano messaggi per farsi riconoscere dagli altri nodi e creano automaticamente la rete; a questo punto i nodi rimangono in attesa di messaggi di configurazione.
Connessione: premendo il pulsante "*Connetti*" (si veda la toolbar in Figura 51) il software di gestione rileva la presenza dei dispositivi mediante l'invio automatico di pacchetti ICMPv6 e restituisce un messaggio nella barra di stato sull'esito della connessione e sul numero di nodi rilevati; in tale fase potrebbe essere necessario o utile riposizionare i nodi per migliorare la qualità di ricezione del segnale dei singoli nodi.
In particolare per determinare la qualità del segnale ricevuto o altre informazioni di un nodo può essere utilizzato il COAP (pulsante M2M della toolbar descritto in seguito).

Fase II – Configurazione della rete e avvio acquisizione
Configurazione sensori: premendo il pulsante "*Configurazione WSN*" l'amministratore può inserire una descrizione simbolica del nodo per facilitarne l'identificazione (es. "Controllo aerazione sala laminazione"). Per ogni nodo è possibile poi specificarne il Tipo e i principali parametri di funzionamento (vedi Figura 54). Il Tipo è una etichetta che identifica il firmware associato al nodo; attualmente è possibile selezionare solo TelosB o Iris-MDA300. In futuro si prevede di estendere la selezione ad altri firmware, al fine di gestire altri mote e sensor-board, e di far sì che tale campo sia automaticamente individuato dal software anziché specificato dall'utente.

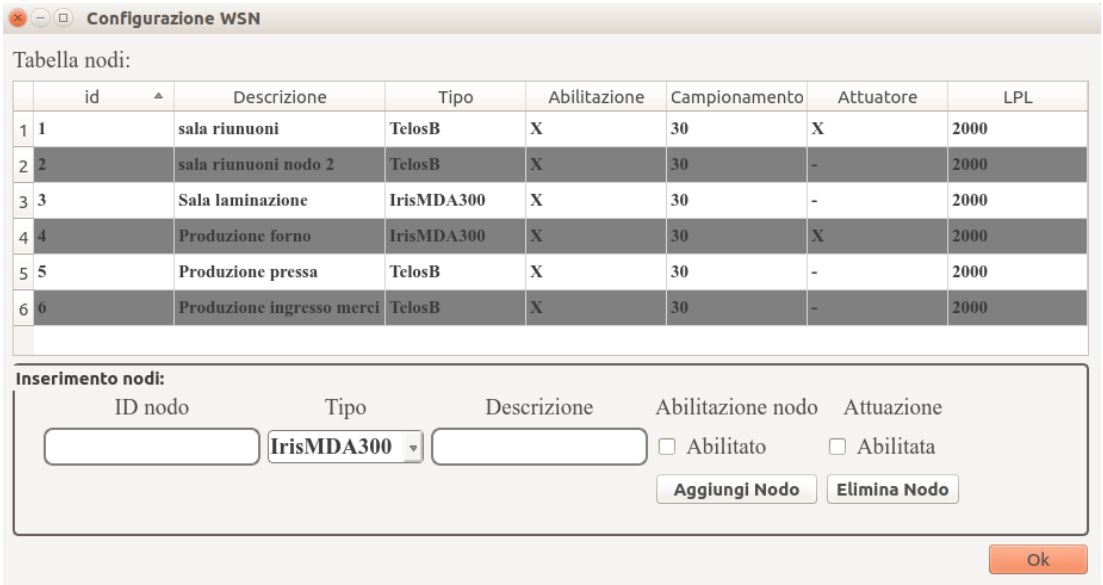


Figura 54 Configurazione nodi della rete WSN.

I parametri che possono essere impostati sono:

- *Periodo di campionamento*: specifica il tempo di campionamento delle grandezze fisiche misurate dal nodo sensore (temperatura, umidità, ecc.); per semplicità tale periodo è lo stesso per tutte le grandezze acquisite.

- *Periodo LPL*: è il massimo intervallo di tempo in cui il transceiver del nodo rimane in stato di inattività (sleep)
- *Periodo Trasmissione*: specifica ogni quanto verrà trasmesso un pacchetto (in pratica il rapporto fra tale periodo e il periodo di acquisizione determina il numero di misure trasmesse per ogni singolo pacchetto).
- *Attuazione*: questo campo è un flag che specifica che per tale nodo sono previste una o più leggi di attuazione (da configurare successivamente).
- *Abilitazione*: questo campo è un flag che specifica se un nodo dovrà essere o meno visualizzato e abilitato al reale funzionamento. In tal modo è possibile definire delle configurazioni virtuali dei nodi da richiamare all'occorrenza.

Chiusa l'interfaccia di *ConfigurazioneWSN* compariranno nell'area di disegno delle icone rappresentanti i nodi sensori precedentemente abilitati (si veda la Figura 55). I nodi rilevati appaiono allineati in alto nell'area di disegno ed è possibile procedere al loro posizionamento.

Figura 55 Nodi rilevati automaticamente in fase di configurazione.

Ogni icona relativa ai nodi sensori mostra (dall'alto verso il basso) l'ID del nodo sensore, un campo testuale per la visualizzazione di messaggi e/o valori e le informazioni sullo stato della batteria e sulla potenza del segnale ricevuto dal nodo. Le icone possono apparire come illustrato in Figura 56 a seconda dello stato di funzionamento. In tale fase, non essendo ancora avviata la rete, l'icona appare come in Figura 56-a) indicando che la fase di rilevamento automatico dei nodi si è conclusa ma la rete non è ancora avviata, mentre al completamento della procedura di configurazione ovvero in uno stato di funzionamento normale il nodo appare come in Figura 56-e). In Figura 56 sono riportate le icone relative agli atri possibili stati di funzionamento, in particolare la Figura 56-c) mostra l'icona di un nodo "inattivo per un breve periodo" (perdita di 5 pacchetti consecutivi) e la Figura 56-d) mostra lo stato di "inattivo per un lungo periodo" (perdita di 10 pacchetti consecutivi). Infine, in Figura 56-b è mostrato lo stato di attesa pacchetto che indica l'avvio della rete e l'attesa del primo pacchetto contenente i dati acquisiti dai sensori.

Figura 56 Stati possibili dei nodi sensore.

A questo punto è possibile posizionare opportunamente tali icone nell'area di disegno in modo da rispecchiare la posizione reale del nodo sensore nell'ambiente fisico da monitorare/controllare.

Configurazione attuatori: premendo il pulsante *Leggi di attuazione* è possibile specificare per ogni attuatore una legge di funzionamento.

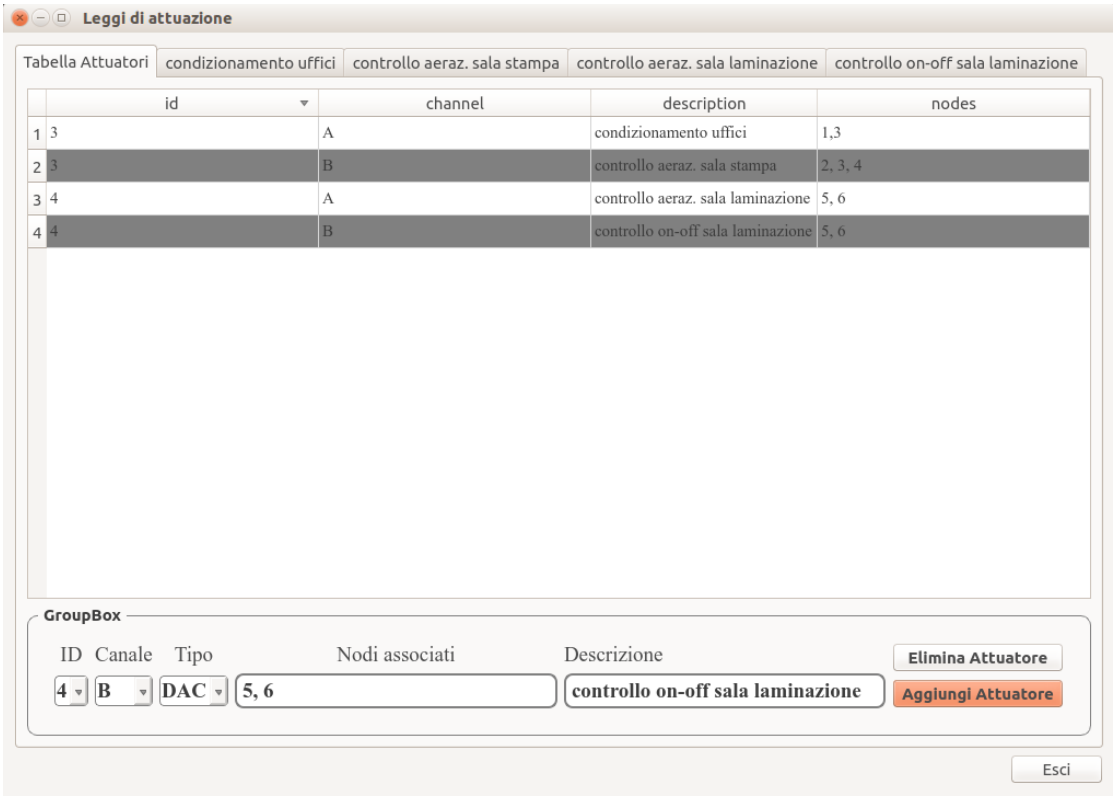


Figura 57 Impostazione leggi di attuazione e tabella degli attuatori.

La configurazione avviene mediante una tabella (detta *Tabella degli attuatori*) in cui per ogni nodo è possibile dare un nome logico all'attuatore al fine di una sua più semplice identificazione. Inoltre, per ogni attuatore occorre indicare il Tipo di attuatore (es. DAC o Relè), il Tipo di legge di attuazione e l'elenco dei nodi che concorrono alla legge di attuazione (si veda Figura 57).

Allo stato attuale sono state previste cinque modalità di funzionamento per gli attuatori.

- controllo manuale
- controllo ON/OFF temporizzato
- controllo a catena aperta con segnale di riferimento
- controllo proporzionale a catena chiusa
- controllo PID a catena chiusa

Il controllo manuale prevede semplicemente l'invio di un valore desiderato ad un attuatore.

Ad esempio in Figura 58 è riportato un esempio di attuazione in cui l'inclinazione di una griglia di aerazione viene impostata al valore desiderato di 45°.

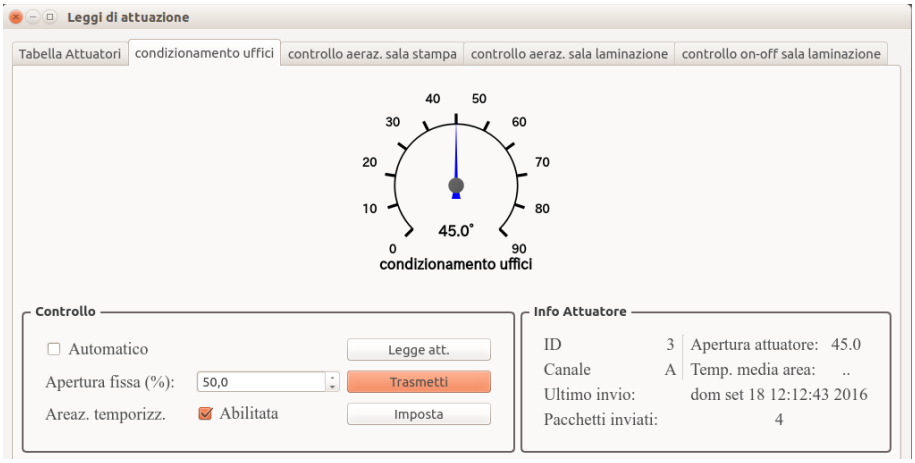


Figura 58 Impostazione leggi di controllo.

Il controllo ON/OFF temporizzato prevede l'invio ad un attuatore di un valore di regolazione con una periodicità desiderata. Nella Figura 59 è mostrato un esempio di applicazione della legge di attuazione suddetta nel caso specifico di controllo di un sistema di aerazione. Come è possibile osservare, il software permette di specificare il valore desiderato dell'angolo di apertura di una griglia di aerazione (Apertura), per quanto tempo debba avvenire l'aerazione (Durata), e l'intervallo di tempo fra due aperture successive (campo "Aera ogni"). È inoltre possibile specificare una fascia oraria in cui tale legge di attuazione verrà applicata.

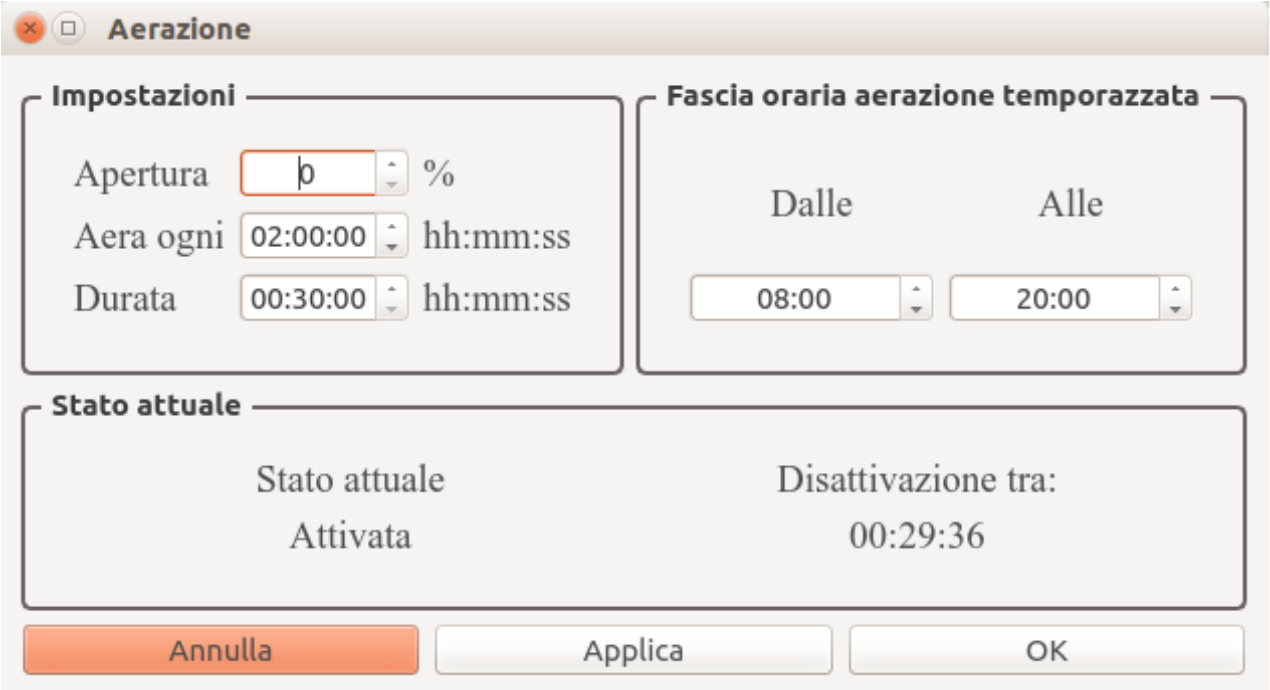


Figura 59 Aerazione temporizzata.

Il controllo a catena aperta con segnale di riferimento generalizza il controllo precedente permettendo di specificare un valore di attuazione differente per ogni fascia oraria si veda ad esempio la Figura 60.

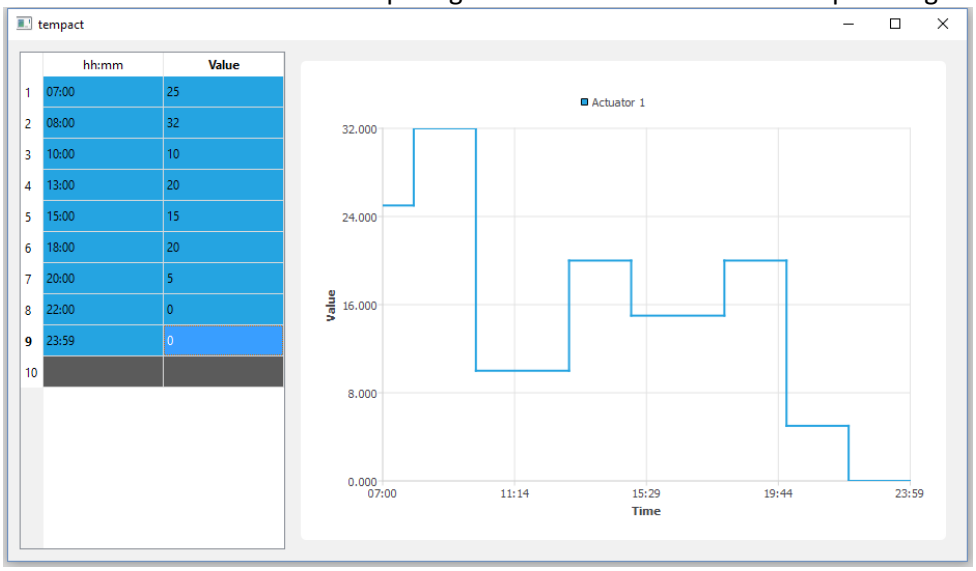


Figura 60 Controllo a catena aperta.

Il controllo proporzionale a catena chiusa prevede la possibilità di combinare le misure prelevate da diversi nodi sensori, tramite media aritmetica o pesata, e confrontare il valore ottenuto con un valore di riferimento. La differenza fra questi due valori, comunemente nota come segnale errore nella teoria dei

controlli automatici, viene fornita ad una legge lineare a tratti (piecewise linear) che può essere impostata graficamente. Anche il valore di riferimento è impostato dall'utente.

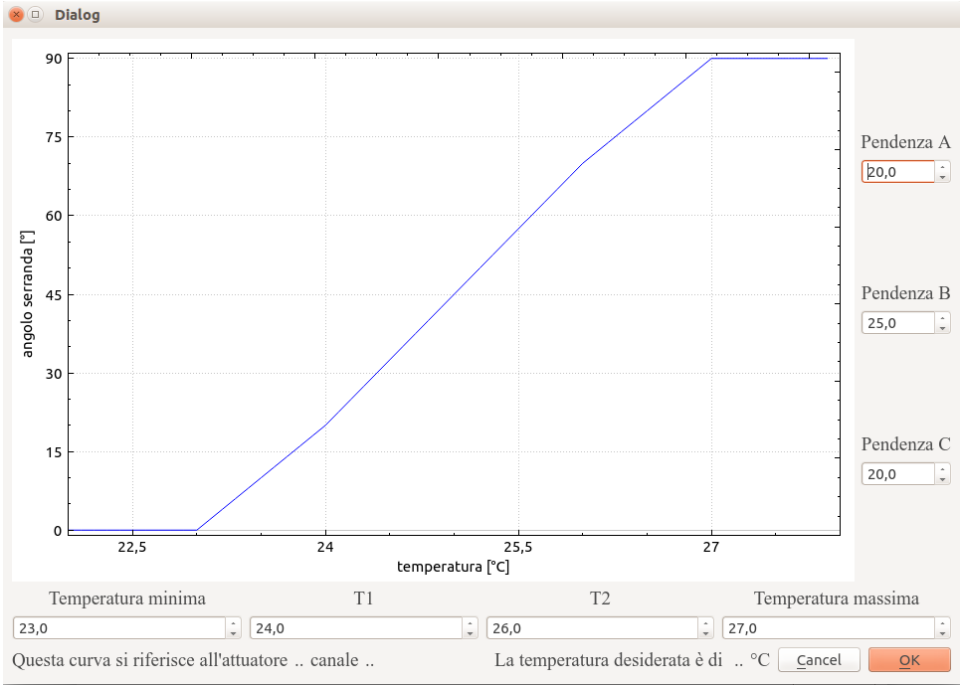


Figura 61 Controllo a catena chiusa.

Infine il controllo PID (Proportional-Integral-Derivative) a catena chiusa prevede che il segnale errore venga utilizzato per una legge di regolazione del tipo

$$y[n] = K_p e[n] + K_i \sum_k e[k] + K_d(e[n] - e[n-1])$$

I valori delle costanti K_p, K_i e K_d possono essere specificate dall'interfaccia, come mostrato in Figura 62.

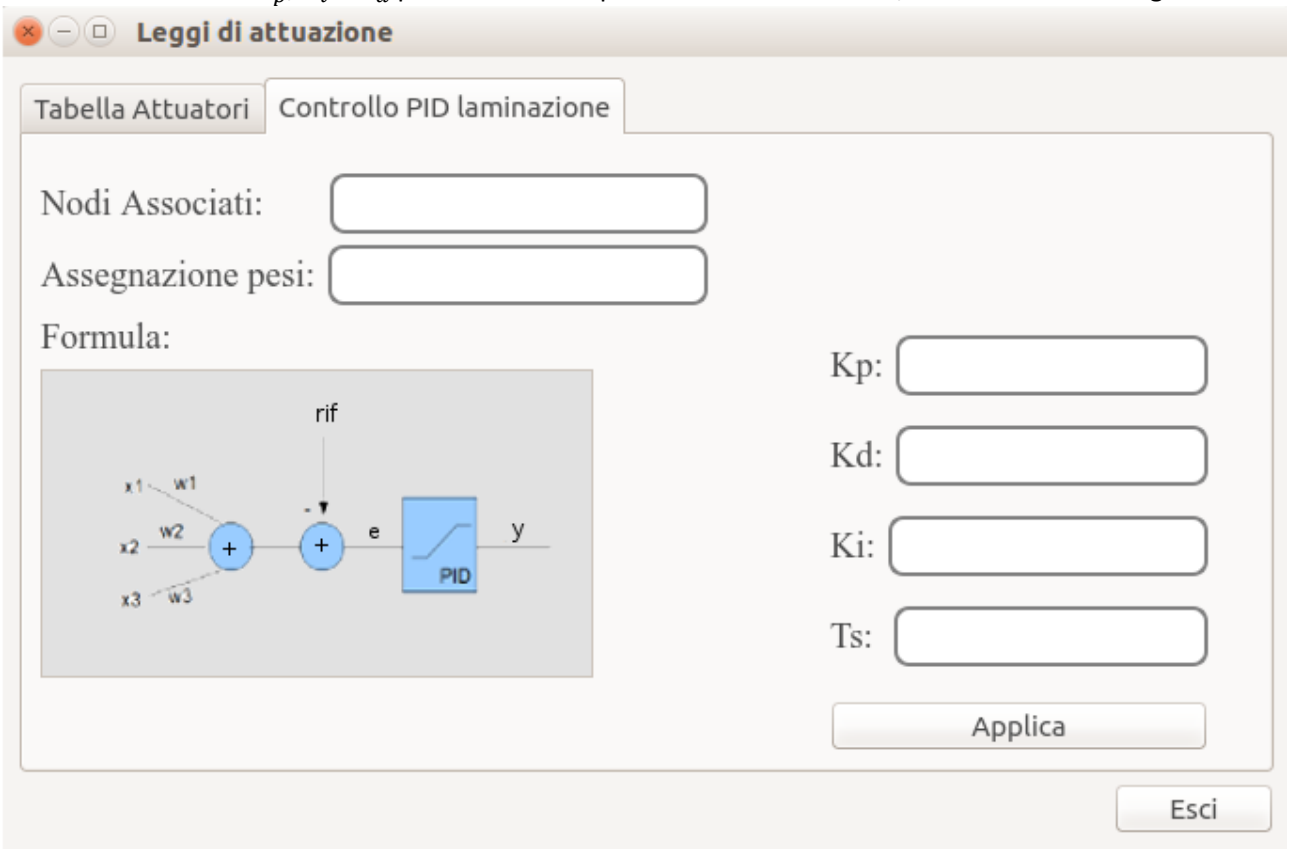


Figura 62 Controllo PID a catena chiusa.

Avvio rete: La configurazione dei sensori e degli attuatori è resa operativa premendo il pulsante *Avvio Rete*. A seguito di tale operazione lo stato dei nodi cambia, ciò è segnalato dal fatto che le icone cambiano colore mostrando il messaggio "Attesa Pacchetto", si veda la Figura 56-b) ad indicare che il Software attende il primo pacchetto da parte del nodo. All'arrivo del primo pacchetto verrà mostrato il valore delle grandezze fisiche acquisite e l'icona del nodo apparirà come in Figura 56-e), aggiornando in tempo reale i valori delle grandezze fisiche acquisite mostrate nell'area centrale del nodo.

Fase III – Raccolta, visualizzazione ed elaborazione dati
Durante tale fase i nodi sensori inviano i pacchetti al nodo sink il quale li inoltra al gateway che provvede ad estrarre i dati e a memorizzarli nel database MySQL precedentemente descritto.
Il software EasyWSN può accedere al database per permettere una più efficace visualizzazione ed elaborazione dei dati acquisiti.
Ad esempio selezionando un nodo nell'area di disegno è possibile ottenere informazioni relative all'ultimo valore acquisito dai sensori, la potenza del segnale, il livello di tensione della batteria, la percentuale di pacchetti persi e un timestamp dell'ultimo pacchetto ricevuto (si veda la Figura 63).

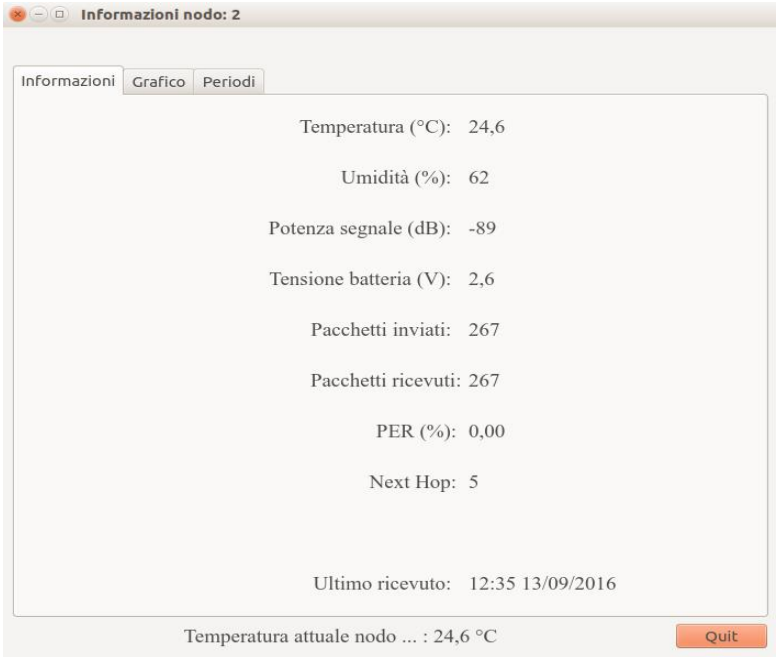


Figura 63 Informazioni principali relative ad ogni nodo.

Il software permette poi la visualizzazione dei dati acquisti mediante diagrammi temporali.
In Figura 64 è illustrato un esempio di visualizzazione dei dati acquisiti in tempo reale relativi a misure di umidità e temperatura.

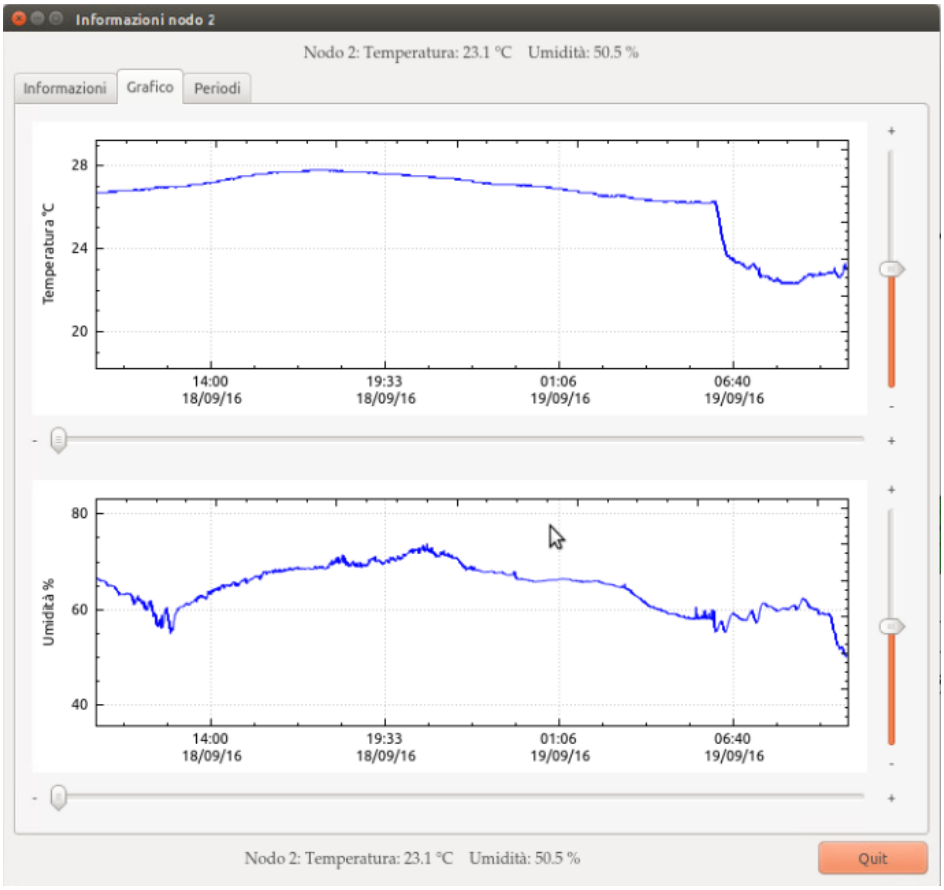


Figura 64 Grafici delle grandezze fisiche acquisite.

Il software permette anche di visualizzare i dati di più nodi contemporaneamente, come illustrato nella Figura 65. In tal caso è sufficiente selezionare nell'area di disegno i nodi e le grandezze desiderate.

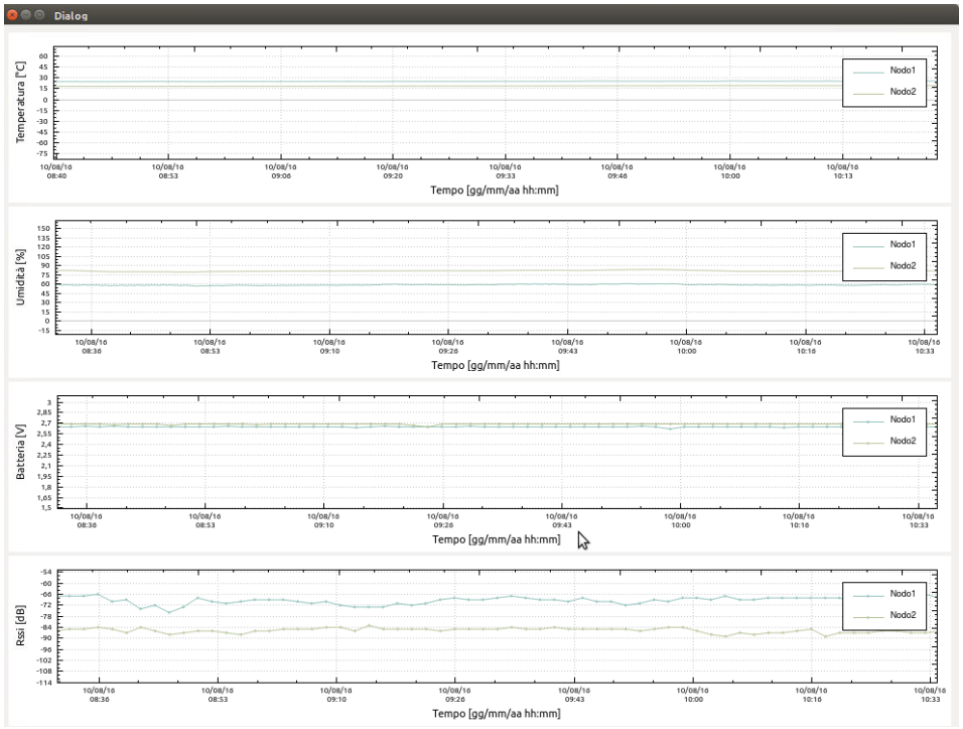


Figura 65 Grafici delle grandezze e dei livelli di tensione della batteria e della potenza.

Il software EasyWSN permette anche di estrarre informazioni statistiche e di riportarle sotto forma di grafici a barre. Ad esempio è possibile visualizzare mediante diagrammi a barre i valori medi, minimi e massimi per ogni grandezza fisica e per ogni nodo. A tale funzionalità si accede premendo il pulsante Statistiche, che, selezionato il nodo o i nodi da esaminare e il periodo di analisi desiderato, mostra i valori calcolati (si veda la Figura 66).

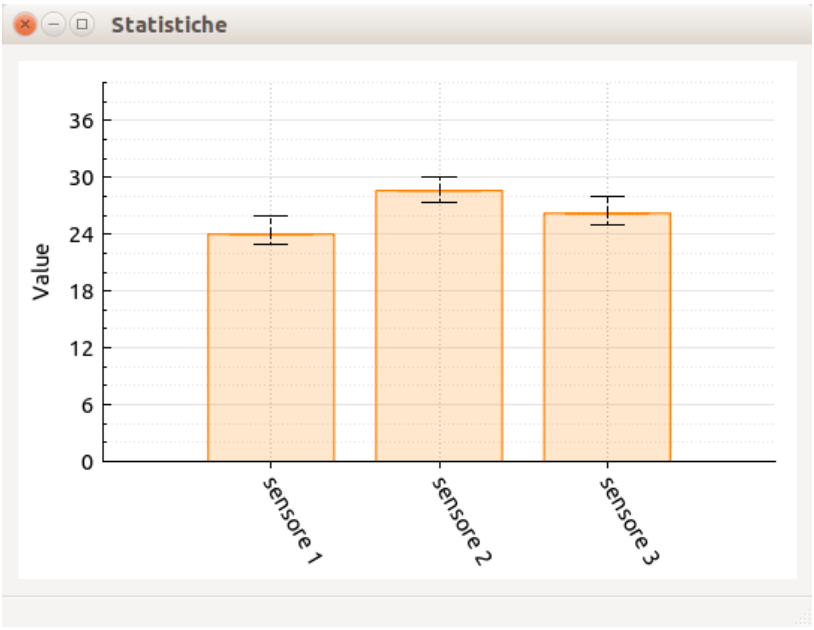


Figura 66 Statistiche – valori medi massimi e minimi delle grandezze acquisite.

Il Software EasyWSN integra inoltre le funzionalità necessarie per comunicare con i nodi mediante il protocollo M2M COAP.

Premendo sul pulsante M2M nella toolbar il software fornisce una interfaccia grafica che permette di eseguire tutte le funzionalità previste dal protocollo COAP, secondo lo standard RFC7252.

Tale interfaccia prescinde dal normale funzionamento dell'acquisizione dati, che continua in background in modo automatico e senza intervento dell'utente.

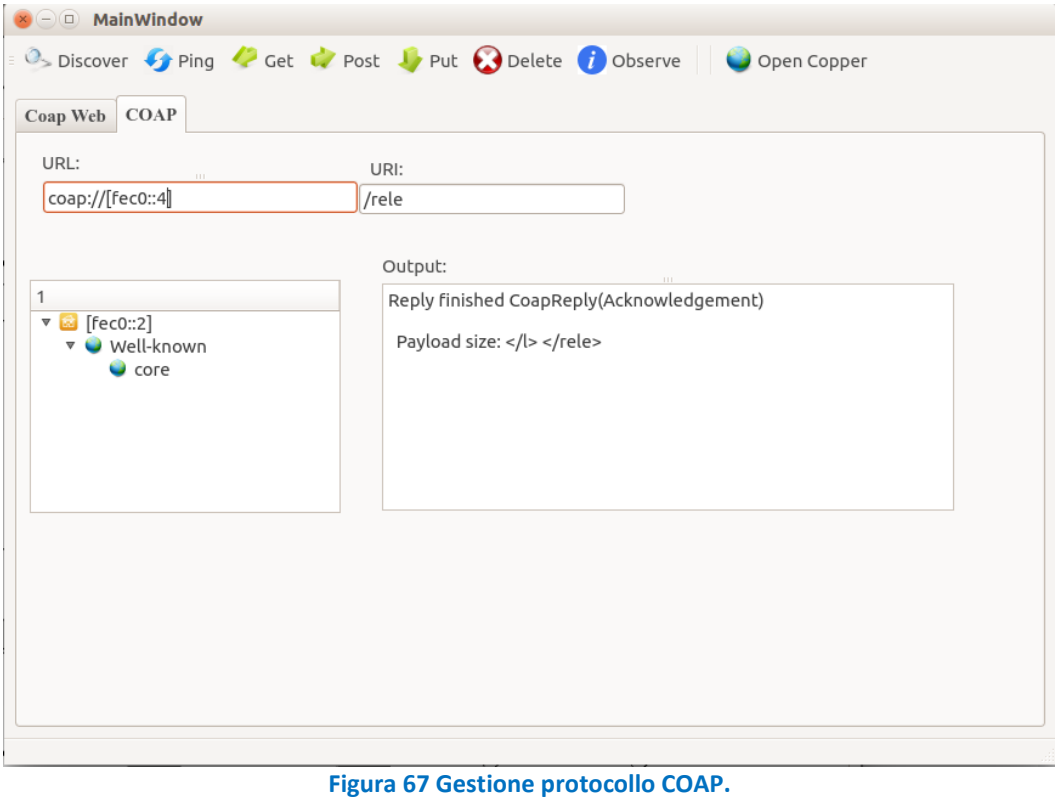


Figura 67 Gestione protocollo COAP.

Come mostrato in Figura 67, per poter effettuare operazioni ed interrogare uno dei nodi della rete, è necessario dapprima inserire l'indirizzo IPv6 del nodo nel campo URL. Una toolbar nella parte superiore permette a tal punto di interrogare il nodo e di eseguire le seguenti operazioni:

- *discover*: esegue una richiesta di "discovery" al nodo. La richiesta consiste in una operazione di GET all'indirizzo "./well-known/core". Se il software riceve una risposta valida viene visualizzato un diagramma ad albero contenente l'elenco delle risorse del nodo (sotto forma di indirizzi URL), come illustrato in Figura 67. Tale elenco è mostrato nell'area testuale *Output* dell'interfaccia grafica. Nel caso in cui non si ottenga risposta dal server, oppure si ottenga una risposta non valida, nella stessa area di Output viene visualizzato un messaggio d'errore relativo all'evento che lo ha causato.
- *get*: la richiesta di get serve ad ottenere l'informazione sulla risorsa identificata dalla URI. Ad esempio, è possibile ottenere informazioni sullo stato di un relè di un nodo con indirizzo fec0::4, specificando l'indirizzo nel campo URL, selezionando la risorsa nel campo URI e premendo il pulsante GET. Il messaggio di risposta viene visualizzato nel campo Output.
- *ping*: invia un pacchetto ICMPv6 ed aspetta la risposta dal nodo. Il nodo risponde con un pacchetto con il codice 2.05 ed il payload vuoto, come previsto dal protocollo COAP. All'arrivo della risposta al ping è mostrato nell'area di Output il messaggio "Reply finished CoapReply (Ack)" insieme al tempo di risposta. La mancata risposta viene visualizzata con un messaggio d'errore.
- *put:* il messaggio put è utilizzato per inviare valori ad una risorsa. A titolo di esempio è possibile pilotare l'accensione di un relè del nodo fec0::4 specificando l'indirizzo del nodo nel campo URL e selezionando la risorsa nel campo URI seguita dal valore da inviare alla risorsa (es. 0=off, 1=on).
- *delete:* il messaggio delete serve a chiedere che la risorsa identificata dall'URI venga cancellata. L'avvenuta cancellazione viene mostrata da un messaggio nel campo Output
- *post:* permette di aggiungere una risorsa
- *observe*: permette di selezionare una risorsa e di ottenere messaggi ogni qual volta vi sia una variazione del valore della risorsa stessa.
- *open Copper:* richiama il plug-in "Copper" di Firefox. Tale funzionalità è stata inserita ai fini di debugging per verificare il corretto funzionamento dell'applicazione sviluppata. In particolare, è possibile confrontare il risultato delle richieste eseguite col software sviluppato con il risultato ottenuto dal plugin Copper di Firefox.

Gestione degli utenti

All'avvio del software appare una finestra di login (Figura 68) in modo da impedire l'accesso al software di gestione ad utenti non autorizzati. Inizialmente il login è possibile solo all'Amministratore. L'utente Amministratore ha poi la possibilità di creare e gestire i permessi dei vari utenti, mediante l'interfaccia mostrata in Figura 69.

In particolare l'amministratore può creare nuovi utenti assegnandoli ad uno dei seguenti Gruppi:

- *Amministratore*: ha pieno accesso a tutte le funzionalità del software, in particolare può configurare la rete e aggiungere altri utenti ed aggiungere e/o eliminare nodi della rete;
- *Operatore*: non ha la possibilità di aggiungere altri utenti, né la possibilità di configurare la rete. L'utente Operatore è però in grado di cambiare i parametri di configurazione dei nodi, quali periodo di campionamento, periodo di trasmissione, periodo LPL, ecc.; può eliminare e/o spostare i nodi dall'area di disegno; può definire leggi di attuazione; può visualizzare i grafici sia real-time che storici delle grandezze acquisite e richiedere statistiche;
- *Utente standard*: non può modificare i parametri dei nodi o le leggi di attuazione, né cancellare e/o spostare nodi; può visualizzare lo stato dei nodi e ottenere grafici sia real-time che storici delle grandezze acquisite oltre che richiedere statistiche;
- *Ospite*: ha solo accesso ai dati acquisiti per ottenere informazioni statistiche, ma non può visualizzare l'area di disegno e/o monitorare lo stato dei nodi (Figura 56).

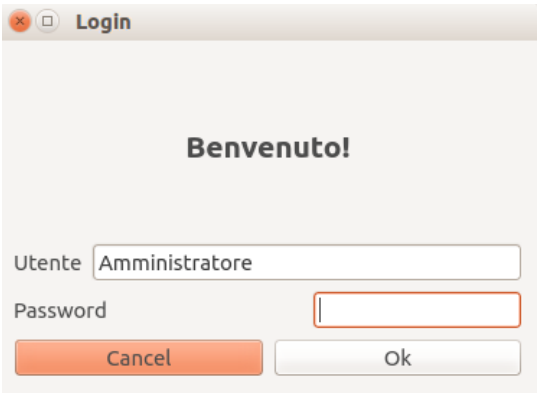


Figura 68 Login utenti.

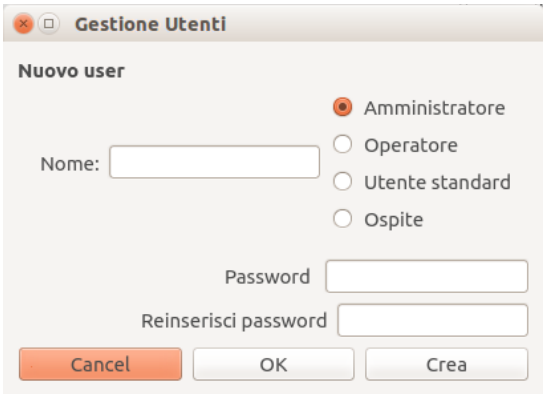


Figura 69 Creazione nuovi utenti.

Le funzionalità del software vengono abilitate o inibite in base al tipo di utente, in particolare alcuni pulsanti della toolbar principale sono disponibili solo all'Amministratore.
Il software permette di cambiare utente in ogni momento, senza interrompere l'esecuzione, premendo sul pulsante della toolbar principale che individua l'utente corrente (si veda la Figura 51).

11 Conclusioni e attività future

Il presente elaborato ha illustrato le attività svolte presso il Dipartimento di Ingegneria dell'Università di Messina finalizzate al progetto e sviluppo di una architettura per reti di sensori e attuatori per progetti M2M.

Di seguito sono elencati i principali risultati raggiunti:

- sono stati individuati i requisiti e le specifiche tecniche per gli scenari applicativi di interesse del progetto (smart industries, smart metering and smart grid)
- sono stati analizzati i diversi nodi sensori presenti sul mercato ed i protocolli di comunicazione per WSN individuando quelli più congeniali per progetti M2M; in particolare è stato definito uno stack protocollare completo, dal livello fisico al livello applicativo, basato interamente su protocolli aperti (802.15.4, 6LowPAN, IPv6, RPL e COAP);
- sono state analizzate le problematiche inerenti la sicurezza nelle reti di sensori individuando le tecniche compatibili con le limitate risorse a disposizione dei nodi sensori;
- sono state analizzate le tecniche di risparmio energetico atte a minimizzare il consumo dei nodi sensori al fine ultimo di massimizzarne il tempo di vita. In particolare è stato investigato l'impatto delle tecniche di duty-cycling e di compressione sui consumi energetici dei nodi sensori, sono stati inoltre individuati i parametri di configurazione atti a massimizzare l'efficienza delle tecniche suddette ed è stato sviluppato un algoritmo di compressione compatibile con le limitate risorse dei nodi e in grado di ridurre del 75% la quantità di dati da trasmettere;
- è stato sviluppato un modello analitico che permette di determinare l'impatto dei parametri di configurazione dei nodi sui principali indici prestazionali della rete (latenza, throughput, efficienza energetica, affidabilità); il modello suddetto è stato validato con misure sperimentali. Mediante il modello sono stati analizzati diversi casi di studio ed individuati i parametri di configurazione per i diversi scenari applicativi. In particolare è stata verificata la possibilità di raggiungere una autonomia dei nodi dell'ordine dei 15 anni per applicazioni di smart metering e di ottenere tempi di risposta nell'ordine dei secondi in reti multi-hop finalizzate ad applicazioni di controllo industriale.
- è stata realizzata una interfaccia utente flessibile ed efficiente che permette di semplificare il set-up e la configurazione delle reti di sensori e che fornisce un front-end per l'acquisizione e l'elaborazione dei dati;
- sono stati realizzati e testati appositi firmware per i nodi sensori al fine di permettere comunicazioni M2M basate sul protocollo COAP;
- sono state effettuate diverse misure sperimentali atte a validare la possibilità di utilizzare l'architettura proposta in ambienti industriali;
- è stato identificato il modo di interfacciare la reti di sensori e attuatori con comuni dispositivi portatili (smartphone e tablet). Ciò permette l'accesso alle risorse della rete agli utenti autorizzati senza che sia necessaria l'installazione di software specifico ma solo l'utilizzo di un comune browser.

Si ritiene opportuno sottolineare che non poche sono state le problematiche tecniche che è stato necessario superare per raggiungere i risultati suddetti, principalmente legate alle limitate risorse dei nodi disponibili in commercio.

Inoltre diverse sono state le competenze in ambito ICT necessarie per realizzare e validare il sistema, dall'elettronica all'informatica passando per le telecomunicazioni e i sistemi di controllo automatici. Solo un efficace coordinamento e il lavoro di squadra fra le unità di ricerca presso UniMe ed ENEA ha reso possibile raggiungere i risultati suddetti nei tempi previsti.

Nonostante i risultati raggiunti si ritiene però che diverse siano le attività ancora necessarie per poter trasformare l'architettura proposta da un prototipo di laboratorio ad un prodotto commerciale.

In particolare occorre evidenziare come attualmente il rapporto costi-prestazioni dei nodi sensori presenti sul mercato non sia ottimale e che la progettazione e realizzazione di nuovi nodi sensori a basso costo con una adeguata capacità computazionale risulti essere di fondamentale importanza al fine di una vasta applicazione delle tecnologie suddette in scenari industriali.

Si ritiene inoltre necessario uno studio mirato a migliorare l'affidabilità nelle reti di sensori. L'attività svolta e in particolare il modello sviluppato hanno infatti evidenziato i principali limiti delle tecniche di trasmissione e dei protocolli attualmente adottati nelle reti di sensori che, essendo basati su meccanismi di ritrasmissione (ARQ), permettono di garantire una maggiore affidabilità solo a scapito di una maggiore latenza di rete (ovvero maggiori ritardi); tali limiti possono essere almeno in parte superati mediante l'utilizzo di tecniche di codifica di canale (FEC) e di codifica di rete (network coding) che permetterebbero di ridurre le ritrasmissioni e quindi di aumentare l'efficienza di rete in termini di affidabilità, throughput, latenza e consumi energetici.

Altra questione rilevante non affrontata in questa fase del progetto è l'interfacciamento dell'architettura di rete proposta con sistemi di metering commerciali. Uno studio in tal senso permetterebbe di trasformare sistemi di metering convenzionali (in particolare i comuni contatori relativi alle utenze di acqua e gas) in sistemi di smart metering.

Infine si ritiene occorra la validazione sul campo dell'architettura proposta con una adeguata campagna di misure finalizzata alla previsione del comportamento energetico di edifici e di processi industriali mediante ad esempio l'uso di modelli inversi.

12 Riferimenti bibliografici

[1] I. F. Akyildiz e M. C. Vuran, “Wireless Sensor Networks”, 2010.

[2] O. Hersent, D. Boswarthick e O. Elloumi, The Internet of Things: Key Applications and Protocols, 2 a cura di, Wiley, 2012.

[3] “Introduction to Software Engineering/Process/Methodology”, 2015. [Online]. Available: https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Methodology.

[4] United States Department of Defense, “DOD-STD-2167A, MILITARY STANDARD: DEFENSE SYSTEM SOFTWARE DEVELOPMENT”, 29 Febbraio 1988.

[5] R. Beudert, L. Juergensen e J. Weiland, “Understanding smart machines: How they will shape the future”, Schneider-Electric, 2015.

[6] “The Internet of Things Architecture (IoT-A) - SOTA report on existing integration frameworks/architectures for WSN, RFID and other emerging IoT related Technologies”, Nicola Bui, 2011.

[7] D. Rotondi, A. Galipò, S. Piccione (TXT), H. Huth, A. M. Houyou, J. W. Walewski (Siemens), C. Kloukinas (City), H. Trsek (inIT) e J. Claessens (EMIC), “IoT@Work D1.3 - Final Requirements and Reference Architecture”, Siemens, 2013.

[8] F. Krief, Green Networking, Wiley, 2012, p. 296.

[9] “TIM”, [Online]. Available: https://www.tim.it/.

[10] “International Society of Automation (ISA)”, [Online]. Available: http://www.isa.org/intech/201504web.

[11] Union for the coordination of transmission of electricity (UCT), “Final report of the Disturbances”, 2006.

[12] Y. Liu, “Wireless Sensor Network Applications in Smart Grid: Recent Trends and Challenges”, *International Journal of Distributed Sensor Networks (IJDSN),* vol. 8, n. 9, 2012.

[13] Terna, “Allegato 6, Rev. 03, Criteri di telecontrollo e di acquisizione dati”, Luglio 2010.

[14] Parlamento europeo e del Consiglio, “Direttiva 2012/27/UE”, 25 ottobre 2012.

[15] CEN-CENELEC-ETSI, “Functional reference architecture for communications in smart metering systems”, 2011.

[16] “Sistemi di comunicazione per contatori e di lettura a distanza dei contatori Parte 4: Lettura senza fili dei contatori (lettura via radio dei contatori per il funzionamento nella banda SRD da 868 MHz a 870 MHz)”, CEI UNI EN 13757-4, 2013.

[17] “Conference Report Disruptive Civil Technologies”, in *National Intelligence Council*, 2008.

[18] J. Granjal, E. Monteiro e J. Sa Silva, “Security of Internet of Things: A Survey of Existing Protocols and Open Research Issues”, in *IEEE Communications Surveys and Tutorials 17.3*, 2015.

[19] I. Stojmenovic, “Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems”, *IEEE Internet of Things Journal,* vol. 1, pp. 122-128, 2014.

[20] A. Kumar, A. Abdelhadi e C. C., “An online delay efficient packet scheduler for M2M traffic in industrial automation”, in *2016 Annual IEEE Systems Conference (SysCon)*, 2016.

[21] Y. Zhang, R. Yu e S. Xie, “Home M2M networks: Architectures, standards and QoS improvements”, *IEEE Communication Magazine,* vol. 49, 2011.

[22] S. H. Sutar, R. Koul e R. Suryavanshi, “Integration of Smart Phone and IOT for development of smart public transportation system”, in *2016 International Conference on Internet of Things and Applications (IOTA)*, 2016.

[23] Z. Fan, R. J. Haines e P. Kulkarni, "M2M communications for E-health and smart grid: an industry and standard perspective", *IEEE Wireless Communications,* vol. 21, n. 1, pp. 62-69, 2014.

[24] M. P. Shopov, "An M2M solution for smart metering in electrical power systems", in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016.

[25] I. J. Shin, D. S. Eom e B. K. Song, "The CoAP-based M2M gateway for distribution automation system using DNP3.0 in smart grid environment", in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2015.

[26] N. Mukudu, N. Ventura, J. Mwangama, A. Elmangoush, R. Steinke e T. Magedanz, "Prototyping Smart City applications over large scale M2M testbed", in *2016 IST-Africa Week Conference*, 2016.

[27] ETSI, "Machine-to-Machine communications (M2M); Functional Architecture", 2013.

[28] ETSI, "Machine to Machine communications (M2M): Use cases of Automotive Applications in M2M capable networks", 2013.

[29] ETSI, "Digital cellular telecommunications system (Phase 2+) (GSM) Universal Mobile Telecommunications System (UMTS); LTE; Service requirements for Machine Type Communications (MTC)", 2016.

[30] "TR-50 - M2M - Smart Device Communications", [Online]. Available: http://www.tiaonline.org/all-standards/committees/tr-50.

[31] "oneM2M", [Online]. Available: http://www.onem2m.org/.

[32] R. Minerva, A. Biru e D. -. T. I. S. Rotondi, "Towards a Definition of the Internet of Things (IoT)", IEEE, 2015.

[33] "CASAGRAS Final Report: RFID and the Inclusive Model for the Internet of Things", EU FP7 Project CASAGRAS, 2009.

[34] "Internet of Things – Architecture", [Online]. Available: http://www.iot-a.eu/public/public-documents/copy_of_d1.2.

[35] "CERP-IoT", [Online]. Available: http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf.

[36] "iCore", [Online]. Available: http://www.iot-icore.eu.

[37] "AIM", [Online]. Available: http://www.ict-aim.eu/.

[38] P. Carlos e A. Ana, "Towards Efficient Mobile M2M Communications: Survey and Open Challenges", *Molecular Diversity Preservation International (MDPI),* vol. 14, n. 10, pp. 19582-19608, 2014.

[39] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures,* UNIVERSITY OF CALIFORNIA, IRVINE, 2000.

[40] W3C, "SOAP Specifications", 29 Marzo 2014. [Online]. Available: http://www.w3.org/TR/soap/.

[41] W3C, "EXI Specifications", 11 Febbraio 2014. [Online]. Available: https://www.w3.org/TR/exi/.

[42] A. Foster, "A Comparison Between DDS, AMQP, MQTT, JMS, REST, CoAP and XMPP", PrismTech, 2015.

[43] Z. Shelby, K. Hartke e C. Bormann, "RFC7252: The Constrained Application Protocol (CoAP)", Giugno 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252.

[44] C. Bormann, A. P. Castellani e Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes", *IEEE Internet Computing,* vol. 16, n. 2, pp. 62-67, 2012.

[45] IETF, "Datagram Transport Layer Security Ver. 1,2", 2012.

[46] CoRE Working Group, "Draft - Transport of COAP over SMS, USSD and GPRS", 2013.

[47] Electronic Industries Association. Engineering Dept, "EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange", Washington, 1696.

[48] TIA/EIA STANDARD, "Electrical Characteristics of Balanced Voltage Digital Interface Circuits", 1994.

[49] EIA Standard, "Electronic Industries Association (1983). Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems", 1983.

[50] A. Stanford-Clark e H. L. Truong, "MQTT For Sensor Networks (MQTT-SN)", International Business Machines Corporation (IBM), 2013.

[51] E. G. Davis, A. Calveras e I. Demirkol, "Improving Packet Delivery Performance of Publish/Subscribe Protocols in Wireless Sensor Networks", *MDPI,* vol. 13, n. 1, pp. 648-680, 2013.

[52] "MEMSIC: Wireless Sensor Networks", [Online]. Available: http://www.memsic.com/wireless-sensor-networks/.

[53] "The Sensor Network Museumtm - Tmote Sky", [Online]. Available: http://www.snm.ethz.ch/Projects/TmoteSky.

[54] "CM5000", [Online]. Available: http://www.advanticsys.com/wiki/index.php?title=CM5000.

[55] "ScatterWeb", [Online]. Available: http://cst.mi.fuberlin.de/projects/ScatterWeb/index.html.

[56] L. F. W. Van Hoesel, S. O. Dulman, P. J. M. Havinga e H. J. Kip, "Design of a low-power testbed for Wireless Sensor Networks and verification", University of Twente, 2003.

[57] "SCHIMMER", [Online]. Available: http://www.shimmer-research.com/tag/sensor.

[58] "Tinynode", [Online]. Available: http://www.tinynode.com/, 2011.

[59] "TinyNode on-line Data Sheet", [Online]. Available: http://www.tinynode.com/?q=system/files/Tinynode%20G4%20gateway_product%20sheet_20160701_0.pdf.

[60] "BTnode Platform", [Online]. Available: http://www.btnode.ethz.ch/.

[61] "National Center of Competence in Research (NCCR)", [Online]. Available: http://www.mics.org/.

[62] "The Smart-Its Project", [Online]. Available: http://www.smart-its.org/.

[63] "Z-Accel Demonstration Kit", [Online]. Available: http://www.ti.com/tool/ez430-rf2480.

[64] "Libelium, Waspmote", [Online]. Available: http://www.libelium.com/products/waspmote/.

[65] "NI WSN-3202", [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/it/nid/207088.

[66] "OpenMote Technologies", [Online]. Available: http://www.openmote.com/.

[67] "Zolertia Online Resources and documentation", [Online]. Available: https://github.com/Zolertia/Resources/wiki.

[68] S. Gajjar, M. Sarkar, N. Choksi e K. S. Dasgupta, "Comparative analysis of wireless sensor network motes", in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, Noida, 2014.

[69] M. O. Farooq e T. Kunz, "Wireless Sensor Networks Testbeds and State-of-the-Art Multimedia Sensor Nodes", 2014.

[70] MEMSIC, "Imote2", [Online]. Available: http://www.devisersoftware.com/uploads/flink/201432013242170.pdf.

[71] A. Rowe, A. Goode, D. Goel e I. Nourbakhsh, "CMUcam3: An Open ProgrammableEmbedded Vision Sensor", 2007.

[72] "Sun SPOT", [Online]. Available: http://www.sunspotworld.com.

[73] S. Hengstler, D. Prashanth, S. Fong e H. Aghajan, "MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance", *In IPSN 2007: Proceedings of the 6th international conference on Information processing in sensor networks,* 2007.

[74] R. Kleihorst, A. Abbo, B. Schueler e A. Danilin, "Camera Mote with a High-Performance Parallel Processor for Real-Time Frame-Based Video Processing", in *First ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, 2007.

[75] "Cyclops", [Online]. Available: http://www.cyclopsgear.com/.

[76] M. Johnson, M. Healy, P. van de Ven, M. J. Hayes, J. Nelson, T. Newe e E. Lewis, "A comparative review of wireless sensor network mote technologies", in *IEEE Sensors*, Christchurch, 2009.

[77] D. De Guglielmo, S. Brienza e G. Anastasi, "IEEE 802.15.4e: a Survey", *Elsevier Computer Communications,* 2016.

[78] IEEE, "IEEE 802.15 WPAN™ Task Group 4 (TG4)", [Online]. Available: www.ieee802.org/15/pub/TG4.html.

[79] "ZigBee Alliance", [Online]. Available: http://www.zigbee.org.

[80] "ZigBee Specification", [Online]. Available: ftp://ftp1.digi.com/support/documentation/html/manuals/ZigBee/Introduction/zigbee.htm.

[81] N. Kushalnagar, G. Montenegro e C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", 2007.

[82] "SNMP Research International, Inc.", [Online]. Available: http://www.snmp.com/snmpv3/.

[83] E. Toscano e L. Lo Bello, "Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios", in *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, Lemgo, 2012.

[84] G. Mulligan, "The 6LoWPAN Architecture", in *Proc. of the 4th Worckshop on Embedded Networked Sensors*, Cork, Ireland, 2007.

[85] HART Communitation Foundation, "WirelessHART", [Online]. Available: http://it.hartcomm.org/hcp/tech/wihart/wireless_overview.html.

[86] Hart Communication Foundation, "HART", [Online]. Available: http://it.hartcomm.org/.

[87] M. Nixon e T. X. Round Rock, "A Comparison of WirelessHART™ and ISA100. 11a.", *Whitepaper, Emerson Process Management,* pp. 1-36, 2012.

[88] International Society of Automation (ISA), "Standard: ISA 100.11A", [Online]. Available: http://standards.globalspec.com/std/1413497/isa-100-11a.

[89] "Embedded Wireless - MiWi Protocol", [Online]. Available: http://www.microchip.com/design-centers/wireless-connectivity/embedded-wireless/miwi-protocol.

[90] R. Steigmann e J. Endresen, "Introduction to WISA - Wireless Interface for Sensors and Actuators", ABB, 2006.

[91] "ANT Basic", [Online]. Available: https://www.thisisant.com/developer/ant/ant-basics.

[92] "SimpliciTI Compliant Protocol Stack", [Online]. Available: http://www.ti.com/tool/simpliciti.

[93] "Z-Wave", [Online]. Available: http://www.z-wave.com/.

[94] EnOcean, "EnOcean Serial Protocol 3 (ESP3)", 2016.

[95] "KNX Association", [Online]. Available: https://www.knx.org/knx-en/index.php.

[96] "Bluetooth Low Energy (BLE)", [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy.

[97] "BLE 4.2: adopted specifications", [Online]. Available: https://www.bluetooth.com/specifications/adopted-specifications.

[98] S. Raza, P. Misra, Z. He e T. Voigt, "Bluetooth smart: An enabling technology for the Internet of Things", in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Abu Dhabi, 2015.

[99] C. Gomez, J. Oller e J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology", *Sensors,* vol. 12, n. 9, p. 11734, 2012.

[100] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby e C. Gomez, "IPv6 over Bluetooth Low Energy", Universitat Politecnica de Catalunya, 2015.

[101] "Thread", [Online]. Available: https://www.threadgroup.org.

[102] K. Gravogl, J. Haase e C. Grimm, "Choosing the best wireless protocol for typical applications", in *ARCS - 24th International Conference on Architecture of Computing Systems*, Como, Italy, 2011.

[103] T. Lennvall e S. H. F. Svensson, "A comparison of WirelessHART and ZigBee for industrial applications", in *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*, Dresden, 2008.

[104] F. Labeau, A. Agarwal e B. Agba, "Comparative Study of Wireless Sensor NetworkStandards for application in Electrical Substations", in *Computing, Communication and Security (ICCCS), 2015 International Conference on*, Pamplemousses, 2015.

[105] C. E. Perkins e P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *SIGCOMM Comput. Commun. Rev.,* 1994.

[106] T. Winter, P. Thubert e R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks draft-ietf-roll-rpl-10", 2010.

[107] "CTP: Collection Tree Protocol", [Online]. Available: https://sing.stanford.edu/gnawali/ctp/.

[108] C. Perkins, E. Royer e S. Das, "Ad hoc On-demand Distance Vector (AODV) Routing".

[109] D. B. Johnson, D. A. Maltz e Y.-C. Hu, "The dynamic source routing (DSR) protocol for mobile ad hoc network", in *IETF MANET Working Group*, 2004.

[110] T. Yang, M. Ikeda, G. DeMarco e L. Barolli, "Performance Behavior of AODV, DSR and DSDV Protocols for Different Radio Models in Ad-Hoc Sensor Networks", in *in Proc. Int. Conf on Parallel Processing Workshops*, 2007.

[111] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne e T. Voigt, "Cross- level sensor network simulation with COOJA", in *in Proc. 31st IEEE Conf. on Local Computer Networks*, 2006.

[112] I. E. Radoi, "Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks", in *ICWMC: The Eighth International Conference on Wireless and Mobile Communications*, 2012.

[113] D. Wang, Z. Tao, J. Zhang e A. Abouzeid, "RPL Based Routing for Advanced Metering Infrastructure in Smart Grid", MITSUBISHI ELECTRIC RESEARCH LABORATORIES, 2010.

[114] "SENSEI: 1st Semantic Web 3.0 Standard for the Internet of Things (IoT)", [Online]. Available: http://www.sensei-iot.org/.

[115] "InterDigital", [Online]. Available: http://www.interdigital.com/.

[116] A. Reddy, A. P. Kumar, D. Janakiram e G. Kumar, "Operating Systems for Wireless Sensor Networks: A Survey Technical Report", *International Journal of Sensor Networks,* vol. 15, pp. 236-255, 2009.

[117] "TinyOS", [Online]. Available: http://www.tinyos.net/.

[118] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer e D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation,* vol. 38, pp. 1-11, 2003.

[119] W. McCartney e N. Sridhar, "Abstractions for safe concurrent programming in networked embedded systems", in *Sensys 2006 Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, 2006.

[120] The TinyOS Alliance, *TinyOS 2.1 Adding Threads and Memory Protection to TinyOS,* SenSys 2008 Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, 2008.

[121] M. Harvan, *Connecting Wireless Sensor Networks to theInternet – a 6lowpan Implementation forTinyOS 2.0 Master's Thesis,* Bremen, Germany: University Bremen, 2007.

[122] T. Vu Chien, H. N. Chan e T. N. Huu, "A comparative study on operating system for Wireless Sensor Networks", in *ICACSIS International Conference on Advanced Computer Science and Information System*, 2011.

[123] A. Dunkels, G. O. e T. Voigt, "Contiki – a Lightweight and Flexible Operating System for

TinyNetworked Sensors", in *In Proceedings of the First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, 2004.

[124] T. Reusing, "Comparison of perating SystemsTinyOS and Contiki", in *Proceedings of the Seminar Sensor Nodes – Operation, Network and Application*, 2012.

[125] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson e R. Han, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms", *Mobile Networks and Applications,* vol. 10, n. 4, pp. 563-579, 2005.

[126] Q. Cao, T. Abdelzaner e J. Stancovic, "The LiteOS Operative System: Towards Unix-like Abstractions for Wireless Sensor Networks", in *IPSN 2008 Information Processing in Sensor Networks*, 2008.

[127] "Java.net – Squawck Development Wiki", [Online]. Available: http://java.net/projects/squawck/pages/SquawckDevelopment.

[128] D. a. C. C. Simon, D. Cleal, J. Daniels e D. White, "JavaTM on the Bare Metal of Wireless Sensor Devices: The Squawk Java Virtual Machine", in *Proceedings of the 2Nd International Conference on Virtual Execution Environments*, {Ottawa, Ontario, Canada, 2006.

[129] N. D. N. Shaylor e W. R. Bush, "A Java Virtual Machine Architecture for Very Small Devices", in *ACM SIGPLAN Conference on Language, Compiler and Tool Support for Embedded Systems*, 2003.

[130] "RIOT GitHub Wiki", [Online]. Available: https://github.com/RIOT/wiki/introduction.

[131] E. Baccelli, O. Hahm, M. Gunes, M. Wahlish e T. Schmidt, *RIOT OS: Towards the Internet of Things,* IEEE INFOCOMM, 2013.

[132] "RIOT Home Page", [Online]. Available: http:// www.riot-os.org.

[133] "Open WSN", [Online]. Available: https://openwsn.atlassian.net/wiki/pages/viewpage.action?pageid=688187.

[134] T. Watteynw, "OpenWSN: a Standard-Based low-power Wireless Development Enviroment", *Transactions on Emerging Telecommunications Technologies,* 2012.

[135] L. Atzori, G. Morabito e A. Iera, "The Internet of Things: A survey", *Computer Networks,* p. 2787–2805, 2010.

[136] V. Rathod e M. Metha, "Security in Wireless Sensor Network: A survey", *GANPAT UNIVERSITY JOURNAL OF ENGINEERING & TECHNOLOGY,* 2011.

[137] F. Melakessou, T. Cholez, J. François, M. Palattella e A. Panchenko, "Distributed IPv6-based Security, Privacy, Authentication and QoS / IoT6 Deliverable D2.2 / Universal Integration of the Internet of Things through an IPv6-based Service Oriented Architecture enabling heterogeneous components interoperability", 2013.

[138] D. Harkins e D. Carrel, "RFC2409: The Internet Key Exchange (IKE)", CISCO Sistem, 1998.

[139] S. Kent, "RFC4302 IP Authentication Header", 2005.

[140] E. Rescorla e N. Modadugu, "RFC6347: Datagram Transport Layer Security Version 1.2", 2012.

[141] K. Chelli, "Security Issues in Wireless Sensor Networks: Attacks and Countermeasures", in *Proceedings of the World Congress on Engineering 2015 Vol I*, London, U.K., 2015.

[142] D. Martins e H. Guyennet, "Wireless Sensor Network Attacks and Security Mechanisms: A Short Survey", *IEEE,* 2010.

[143] A. S. Sastry, S. Sulthana e S. Vagdevi, "Security Threats in Wireless Sensor Networks in Each Layer", *International Journal of Advanced Networking and Applications, Vol. 04 Issue 04,,* pp. 1657-1661, 2013.

[144] S. Kaplantzis, "Security Models for Wireless Sensor Networks", 2006.

[145] C. Karlof e D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures", *Ad Hoc Networks Journal, Vol.1, Issue 2-3,* pp. 293-315, 2003.

[146] Y. Sun, Z. Han e K. J. R. Liu, "Defense of Trust Management Vulnerabilities in Distributed Networks",

IEEE Communications Magazine, Vol 46, Issue 2, pp. 112-119, 2008.

[147] Y. Yu, K. Li, W. Zhou e P. Li, "Trust mechanisms in wireless sensor networks: attack analysis and countermeasures", *Journal of Network and Computer Applications, Elsevier,* 2011.

[148] W. Xu e et al, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks", in *MobiHoc '05: Proc. 6th ACM Int. Symp. Mobile Ad Hoc Net. and Comp*, 2005.

[149] W. Xu, W. Trappe e Y. Zhang, "Channel Surfing: Defending Wireless Sensor Networks from Interference", in *Proc. Of Information Processing in Sensor Networks*, 2007.

[150] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang e A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks", in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, Rome, 2001.

[151] A. Woo e D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks", in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom*, Rome, 2001.

[152] R. David, R. Midkiff e S. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses", *IEEE Pervasive Computing, Vol. 7, No. 1,* pp. 74-81, 2008.

[153] B. Parno, A. Perrig e V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks", in *Proceedings of the IEEE Symposium on Security and Privacy (S&P'05)*, 2005.

[154] Y. Xiao, V. Rayi, B. Sun, X. Du, F. Hu e M. Galloway, "A survey of key management schemes in wireless sensor networks", *Computer Communications 30(11-12),* p. 2314–2341, 2007.

[155] A. Jain, K. Kant e M. R. Tripathy, "Security Solutions for Wireless Sensor Networks", in *Second International Conference on Advanced Computing & Communication Technologies*, 2012.

[156] D. E. Burgner e A. Luay, "Wahsheh Security of Wireless Sensor Networks", in *Eighth International Conference on Information Technology: New Generations*, 2011.

[157] S. Zhu, S. Setia e S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks", in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003.

[158] B. Lai, S. Kim e I. Verbauwhede, "Scalable session key construction protocol for wireless sensor networks", in *Proceedings of the IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES '02)*, 2002.

[159] R. Blom, "An optimal class of symmetric key generation systems", in *Proceedings of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, 1985.

[160] C. Blundo, A. Santix, A. Herzberg, S. Kutten, U. Vaccaro e M. Yung, "Perfectly-secure key distribution for dynamic conferences", in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Crypto-logy*, Germany, 1992.

[161] D. Liu e P. Ning, "Location-based pairwise key establishments for static sensor networks", in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (CCS '03)*, 2003.

[162] L. Eschenauer e V. Gligor, "A key-management scheme for distributed sensor networks", in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002.

[163] R. Anderson, H. Chan e A. Perrig, "Key infection: Smart trust for smart dust", in *Proceedings of the 12thIEEE International IEEE International Conference on Network Protocols (ICNP '04)*, 2004.

[164] N. Gura, A. Patel, W. Arvinderpal, E. Hans e S. Chang Shantz, ""Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs", Sun Microsystems Laboratories,", in *Cryptographic Hardware and Embedded Systems*, 2004, pp. 119-132.

[165] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn e P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology", in *Proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks SASN'04*, 2004.

[166] D. J. Malan, M. Welsh e M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based

on Elliptic Curve Cryptography", 2007.

[167] H. Wang, B. Sheng, C. C. Tan e Q. Li, "Comparing Symmetric-key and Public-key based Security Schemes in Sensor Networks: A Case Study of User Access Control", in *The 28th International Conference on Distributed Computing Systems ICDCS '08*, USA, 2008.

[168] H. Wang e Q. Li, "Efficient implementation of public key cryptosystems on mote sensors", in *Proceedings of the International Conference on Information and Communication Security (ICICS '06)*, 2006.

[169] J. Girao, D. Westhoff, E. Mykletun e T. Araki, "TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks", *Elsevier Ad Hoc Journal,* p. 1073–1089, 2007.

[170] L. B. Oliveira, M. Scott, J. Lopez e L. Dahab, "TinyPBC: Pairings for Authenticated Identity Based Non-Interactive Key Distribution in Sensor Networks", *Computer Communications,* p. 485–493, 2011.

[171] A. Barki, A. Bouabdallah, S. Gharout e J. Traoré, "M2M Security: Challenges and Solutions", *IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 2,* 2016.

[172] ETSI, "Machine-to-Machine communications (M2M); MIA, DIA and mid interfaces", 2014.

[173] P. Jayaraman, N. Com, R. Lopez, Y. Ohba e M. Parthasarathy, "Protocol for carrying authentication for network access (PANA) framework, RFC 5193", 2008.

[174] I. Cha, Y. Shah, A. Schmidt, A. Leicher e M. Meyerstein, "Trust in M2M communication,", *IEEE Veh. Technol. Mag., vol. 4, no. 3,,* p. 69–75, 2009.

[175] W. Trappe, "The challenges facing physical layer security,", *IEEE Commun. Mag., vol. 53, no. 6,* p. 16–20, 2015.

[176] N. Lawson, "Side-channel attacks on cryptographic software,", *IEEE Sec.,* p. 65–68, 2009.

[177] GSMA, "GSMA embedded SIM specification-A single,common and global specification to accelerate growth in M2M", in *Remote M2M Provisionning*, 2014.

[178] S. Yu, S. Guo e I. Stojmenovic, "Fool me if you can: Mimicking attacks and anti-attacks in cyberspace,", *IEEE Trans. Comput., vol. 64, no. 1,* pp. 139-151, 2015.

[179] M. Jawurek, . F. Kerschbaum e G. Danezis, "SoK: Privacy technologies for smart grids—A survey of options", 2012.

[180] W. Yu, "False Data Injection Attacks in Smart Grid: Challenges and Solutions", 2012.

[181] A. Olivereau, Y. B. Saied e M. Laurent, "5th Int. Conf. New Technol. Mobility Sec. (NTMS)", in *A distributed approach for secure M2M communications*, 2012.

[182] H. Hussen, G. Tizazu, M. Ting, T. Lee, Y. Choi e K.-H. Kim, "Sakes: Secure authentication and key establishment scheme for M2M communication in the IP-based wireless sensor network (6l0 WPAN)", in *5th Int. Conf. Ubiq. Future Netw. (ICUFN)*, 2013.

[183] H. Nicanfar, P. Jokar, K. Beznosov e V. Leung, "Efficient authentication and key management mechanisms for smart grid communications", *IEEE Syst. J., vol. 8, no. 2,* p. 629–640, 2014.

[184] Y. Li, "Design of a key establishment protocol for smart home energy management system", in *5th Int. Conf. Comput. Intell. Commun. Syst. Netw. (CICSyN)*, 2013.

[185] J. Kamto, L. Qian, J. Fuller e J. Attia, "Light-weight key distribution and management for advanced metering infrastructure", in *IEEE GLOBECOM Workshops*, 2011.

[186] I. Doh, J. Lim, S. Li e K. Chae, "Key establishment and management for secure cellular machine-to-machine communication", in *7th Int. Conf. Innov. Mobile Internet Serv. Ubiq. Comput. (IMIS)*, 2013.

[187] W. Zhang e G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach", in *24th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, 2005.

[188] A. Bartoli, J. Hernondez-Serrano, O. Leon, A. Kountouris e D. Barthel, "Energy-efficient physical layer packet authenticator for machine-to-machine networks", *Emerging Telecommun. Technol., vol. 24,*

no. 4,, p. 401–412, 2013.

[189] H. Nicanfar, P. Jokar e L. Leung, "Smart grid authentication and key management for unicast and multicast communications", in *IEEE PES Innov. Smart Grid Technol. Asia (ISGT)*, 2011.

[190] S. Chen e M. Ma, "A dynamic-encryption authentication scheme for M2M security in cyber physical systems", in *IEEE Global Commun. Conf. (GLOBECOM)*, 2013.

[191] M. Nabeel, S. Kerr, X. Ding e E. Bertino, "Authentication and key management for advanced metering infrastructures utilizing physically unclonable functions", in *IEEE 3rd Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2012.

[192] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing", in *Advances in Cryptology — CRYPTO '91*, 1992, pp. 129-140.

[193] C. Schnorr, "Efficient identification and signatures for smart cards", in *Advances in Cryptology — EUROCRYPT '89*, 1990, pp. 688-689.

[194] J. Cao, M. Ma e H. Li, "A group-based authentication and key agreement for MTC in LTE networks", in *IEEE Global Commun. Conf. (GLOBECOM)*, 2012.

[195] C. Lai, H. Li, R. Lu, R. Jiang e X. Shen, "LGTH: A lightweight group authentication protocol for machine-type communication in LTE networks", in *IEEE Global Commun. Conf. (GLOBECOM)*, 2013.

[196] C. Lai, H. Li, X. Li e J. Cao, "A novel group access authentication and key agreement protocol for machine-type communication", *Trans. Emerging Telecommun. Technol., vol. 26, no. 3,,* 2015 .

[197] C. Lai, H. Li, R. Lu, R. Jiang e X. Shen, "SEGR: A secure and efficient group roaming scheme for machine to machine communications between 3GPP and WiMAX networks", in *IEEE Int. Conf. Commun. (ICC)*, 2014.

[198] H. Nicanfar, S. Hosseininezhad, P. TalebiFard e V. Leung, "Robust privacy-preserving authentication scheme for communication between electric vehicle as power energy storage and power stations", in *IEEE INFOCOM*, 2013.

[199] M. Au, J. Liu, J. Fang, Z. Jiang, W. Susilo e J. Zhou, "A new payment system for enhancing location privacy of electric vehicles", *IEEE Trans. Veh. Technol., vol. 63, no. 1,* pp. 3-18, 2014.

[200] M. Au, W. Susilo e Y. Mu, "Constant-size dynamic K-TAA", in *Security and Cryptography for Networks*, 2006, p. 111–125.

[201] S. Goldwasser, S. Micali e C. Rackoff, "The knowledge complexity of interactive proof systems", *SIAM J. Comput., vol. 18, no. 1,* p. 186–208, 1989.

[202] R. Popa, H. Balakrishnan e A. Blumberg, "VPriv: Protecting privacy in location-based vehicular services", in *18th Conf. USENIX Sec. Symp. (SSYM'09)*, Berkeley, 2009.

[203] X. Chen, G. Lenzini, S. Mauw e J. Pang, "A group signature based electronic toll pricing system", in *7th Int. Conf. Availability Reliab. Sec. (ARES)*, 2012.

[204] C. Troncoso, G. Danezis, E. Kosta, J. Balasch e B. Preneel, "PriPAYD: Privacy-friendly pay-as-you-drive insurance,", *IEEE Trans. Dependable Secure Comput., vol. 8, no. 5,* p. 742–755, 2011.

[205] G. Kalogridis, C. Efthymiou, S. Denic, T. Lewis e R. Cepeda, "Privacy for smart meters: Towards undetectable appliance load signatures", in *1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2010.

[206] D. Varodayan e A. Khisti, "Smart meter privacy using a rechargeable battery: Minimizing the rate of information leakage", in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2011.

[207] B. Adiga, P. Balamuralidhar, M. Rajan, R. Shastry e V. Shivraj, "An identity based encryption using elliptic curve cryptography for secure M2M communication", in *1st Int. Conf. Sec. Internet Things SECURIT*, Kollam, 2012.

[208] G. Frey e H.-G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves", *Math. Comput., vol. 62,* p. 865–874, 1994.

[209] J.-R. Shih e et al., "Securing M2M with post-quantum public-key cryptography", *IEEE J. Emerging Sel. Topics Circuits Syst., vol. 3, no. 1,* p. 106–116, 2013.

[210] J. Hoffstein, J. Pipher e J. Silverman, "NTRU: A ring-based public key cryptosystem", in *Algorithmic Number Theory*, 1998, p. 267–288.

[211] B.-Y. Yang, J.-M. Chen e Y.-H. Chen, "TTS: High-speed signatures on a low-cost smart card", in *Cryptographic Hardware and Embedded Systems*, 2004, p. 371–385.

[212] L. Marin, A. Jara e A. Skarmeta, "Shifting primes: Optimizing elliptic curve cryptography for smart things", in *6th Int. Conf. Innov. Mobile Internet Serv. Ubiq. Comput. (IMIS)*, 2012.

[213] R. Lu, X. Li, X. Liang, X. Shen e X. Lin, "GRS: The green, reliability, and security of emerging machine to machine communications", *IEEE Commun. Mag., vol. 49, no. 4,* pp. 28-35, 2011.

[214] W. Ren, L. Yu, L. Ma e Y. Ren, "RISE: A reliable and secure scheme for wireless machine to machine communications", *Tsinghua Sci. Technol., vol. 18,* pp. 100-117, 2013.

[215] G. Anastasi, E. Ancillotti, M. Conti e A. Passarella, "Experimental Analysis of TCP Performance in Static Multi-hop Ad Hoc Networks", in *Mobile Ad Hoc Networks: from Theory to Reality*, Nova Science Publisher, 2007.

[216] C. Busch, M. Magdon-Ismail, F. Sivrikaya e B. Yener, "Contention-Free MAC Protocols for Wireless Sensor Networks", in *Distributed Computing*, Springer, 2004, pp. 245-259.

[217] I. Demirkol, C. Ersoy e F. Alagoz, "F. Alagoz - MAC protocols for wireless sensor networks: a survey", 2006.

[218] V. Rajendran, J. Garcia-Luna-Aceves e K. Obraczka, "Energy-Efficient, Application-Aware Medium Access for Sensor Networks", in *MASS-IEEE*, 2005.

[219] L. van Hoesel e P. Havinga, "A Lightweight Medium Access Protocol (LMAC) for Wireless SensorNetworks".

[220] W. L. Lee, A. Datta e Cardell-Oliver, "RFlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks", 2008.

[221] R. Soua, E. Livolant e P. Minet, "An Adaptive Strategy for an Optimized Collision - Free Slot Assignment in Multichannel Wireless Sensor Networks", 2013.

[222] J. Polastre, J. Hill e D. Culler, "Versatile Low Power Media Accessfor Wireless Sensor Networks", 2004.

[223] D. Moss e P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking", 2008.

[224] M. Doudou, D. Djenouri e N. Badache, "Survey on Latency Issues of Asynchronous MAC Protocols in Delay-Sensitive Wireless Sensor Networks", *IEEE Communications Surveys & Tutorials,* vol. 15, n. 2, pp. 528-550, 2013.

[225] M. Buettner, G. V. Yee, E. Anderson e R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks", in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, Boulder, Colorado, USA, 2006.

[226] "MAC protocols in ContikiOS", [Online]. Available: http://anrg.usc.edu/contiki/index.php/MAC_protocols_in_ContikiOS.

[227] "MathWorks", [Online]. Available: https://it.mathworks.com/.

[228] C. Wang, Q. Yin, W. Wang, J. Zhang e H. Liu, "A simple energy efficient transceiver for IEEE 802.15.4", in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, Paris, 2012.

[229] H. Nyquist, "Certain topics in telegraph transmission theory", *Trans. AIEE,* vol. 47, p. 617–644, 1928.

[230] G. Campobello, O. Giordano, A. Segreto e S. Serrano, "Comparison of local lossless compression algorithms for Wireless Sensor Networks", *Journal of Network and Computer Applications,* vol. 47, pp. 23-31, 2015.

[231] R. Xuejun e F. Dingyi, "A normal distribution encoding algorithm for slowly varying data compression

in wireless sensor networks", in *International Conference on Wireless Communications Networking and Mobile Computing*, 2010.

[232] M. V. F. Marcelloni, "A simple algorithm for data compression in wireless sensor networks", *IEEE Communications Letters,* vol. 12, n. 6, pp. 411-413, 2008.

[233] Y. Liang e W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission", *ACM SIGCOMM Computer Communication Review,* vol. 40, pp. 12-18, 2012.

[234] D. Slepian e J. K. Wolf, "Noiseless coding of correlated information sources", *IEEE Transactions on information Theory,* vol. 19, pp. 471-480, 1973.

[235] E. Candès e M. Wakin, "An Introduction To Compressive Sampling", *IEEE Signal Processing Magazine,* vol. 21, 2008.

[236] T. Ho e D. S. Lun, Network Coding: An Introduction.

[237] G. Campobello, A. Segreto e S. Serrano, "Data Gathering Techniques for Wireless Sensor Networks: A Comparison", *International Journal of Distributed Sensor Networks,* vol. 2016, 2016.

[238] L. Lesser, "Exploring the birthday problem with spreadsheets", *The Mathematics Teacher,* vol. 92, n. 5, pp. 407-411, 1999.

[239] "Copper", [Online]. Available: https://addons.mozilla.org/it/firefox/addon/copper-270430/.

[240] "Aneska", [Online]. Available: https://play.google.com/store/apps/details?id=pl.sixpinetrees.aneska&hl=it.

[241] RFC3920, "Extensible Messaging and Presence Protocol (XMPP): Core", Ottobre 2014. [Online]. Available: http://www.ietf.org/rfc/rfc3920.

[242] S. Swapna Kumar, M. Nanda Kunarm, V. S. Sheeba e K. R. Kashwan, "Power Efficient Dynamc MAC Protocol (D-MAC) for Wireless Sensor Networks", 2012.

[243] X. Shi e G. Stromberg, "SyncWUF: An Ultra Low-Power MAC Protocol for Wireless Sensor Networks", *in IEEE Transactions on Mobile Computing,* vol. 6, n. 1, pp. 115 - 125, 2007.

[244] S. Li, K. Choi e K. Chae, "An enhanced measurement transmission scheme for privacy protection in smart grid", in *Int. Conf. Inf. Netw. (ICOIN)*, 2013.

[245] E. Daniel e A. Luay, "Wahsheh "Security of Wireless Sensor Networks", in *Eighth International Conference on Information Technology: New Generations*, 2011.

[246] W. P. a. S. N. McCartney, "Abstractions for Safe Concurrent Programming in Networked Embedded Systems", in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems - SenSys 2006*, Boulder, Colorado, USA, 2006.

Giuseppe Campobello nasce a Messina, Italia, nel 1975.
Riceve la Laurea in Ingegneria Elettronica (summa cum laude) e il Dottorato di Ricerca (Ph.D.) in "Tecnologie Avanzate per l'Ingegneria dell'Informazione" presso l'Università degli Studi di Messina, rispettivamente nel 2000 e nel 2004.

Dal 2004 al 2006 è stato titolare di contratti finalizzati ad attività di ricerca e professore a contratto della Facoltà di Ingegneria e della Facoltà di Scienze MM.FF.NN. dell'Università di Messina.

Nel 2006 è stato altresì assegnista di ricerca presso il Dipartimento di Matematica e Informatica dell'Università di Catania.

Nel dicembre del 2006 vince il concorso per Ricercatore Universitario del settore ING-INF/03 (Telecomunicazioni) presso l'Università degli Studi di Messina.

Attualmente afferisce al Dipartimento di Ingegneria dell'Università di Messina dove è Ricercatore Confermato a tempo pieno oltre che Professore Aggregato del settore Telecomunicazioni e responsabile del laboratorio di Comunicazioni Wireless.

L'attività di ricerca, svolta sia in ambito universitario che in collaborazione con aziende ed enti di ricerca, si inquadra principalmente nell'ambito delle reti di telecomunicazioni, dell'elaborazione numerica dei segnali e dell'elettronica digitale applicata alle telecomunicazioni. In particolare l'attività di ricerca più recente è focalizzata sulle reti di sensori wireless.

È autore di oltre trenta articoli scientifici apparsi su riviste internazionali o presentati a conferenze internazionali e revisore di diverse riviste internazionali della IEEE e della Elsevier.

È inoltre membro del Consiglio Scientifico del Gruppo Telecomunicazioni e Tecnologie dell'Informazione (GTTI) e del Microwave Engineering Center for Space Applications (MECSA).

Nicola Donato ha conseguito la laurea in Ingegneria Elettronica all'Università degli Studi di Messina ed il dottorato di ricerca all'Università degli Studi di Palermo. Attualmente è professore Associato di Elettronica presso il dipartimento di Ingegneria dell'Università di Messina. E' responsabile del laboratorio di Elettronica dei Sensori e dei Sistemi di Trasduzione (LESST). I suoi principali campi di ricerca riguardano la caratterizzazione e modellistica circuitale di dispositivi per applicazioni a microonde (0.05-40 GHz), la progettazione e realizzazione di sistemi di misura, lo sviluppo e caratterizzazione elettrica di sensori, la realizzazione di software per l'interfacciamento di strumentazione avanzata, lo studio delle prestazioni elettriche e relativa modellizzazione circuitale di sensori di gas a film sottile (resistivi, capacitivi, BAW, SAW) e lo sviluppo e caratterizzazione di dispositivi organici ad eterogiunzione fotosensibili e celle fotovoltaiche ibride organico/inorganico.

È coautore di più di 120 pubblicazioni scientifiche, (H-index 18, cit. 1026, fonte Scopus). E' referee di numerose riviste internazionali quali IEEE Trans. IM e Sensors and Actuators; è inoltre componente dell'editorial board del Hindawi Journal of Sensors.

È componente del consiglio direttivo del MECSA (Microwave Engineering Center for Space Applications) e responsabile dell'unità di ricerca GMEE (Associazione Gruppo Misure Elettriche ed Elettroniche) di Messina.

Antonino Segreto nasce a messina, Italia nel 1979.

Cosegue la Laurea quinquennale in Ingegneria Elettronica nel 2011 e il Dottorato di Ricerca (Ph.D) in "Tecnologie avanzate per l'Optoelettronica, la Fotonica e Modellizzazione Elettromagnetica" nel 2015, entrambe presso l'Università degli Studi di Messina.

Dal 2015 ad oggi è collaboratore di ricerca presso il Dipartimento di Ingegneria dell'Università degli Studi di Messina.

L'attività di ricerca riguarda principalmente le reti di telecomunicazioni, l'elaborazione dei segnali e le reti di sensori wireless.

Maria-Anna Segreto nasce a Messina nel 1975.

Consegue la laurea quinquennale in Ingegneria Civile-Edile nel 2005 con una tesi dal titolo "Sviluppo delle tecnologie per componenti edilizi innovativi opachi e trasparenti. Progettazione energetica di una scuola d'arte nell'area del Monte di Pietà a Messina." Dal 2005 ricercatrice presso ENEA, attualmente Responsabile Scientifico del laboratorio di Ricerca Industriale LAERTE, si occupa principalmente in studi in ambito di efficientamento energetico in ambito civile e nei processi industriali. Membro della Commissione Ambiente dei Dottori Commercialisti di Bologna, membro del Comitato Tecnico Scientifico per la redazione del Piano Energetico Regionale in Emilia-Romagna, membro del Nucleo di Valutazione per i bandi sull'efficienza energetica per la Regione Emilia-Romagna. Ha svolto, inoltre, attività di valutazione delle proposte per l'accesso al meccanismo dei Certificati Bianchi ed ha all'attivo numerose partecipazioni a Progetti Europei in ambito energetico.

Salvatore Serrano è nato a Catania (Italia) nel 1972. Ha conseguito rispettivamente la laurea quinquennale in Ingegneria Informatica nel 1999 e il dottorato di ricerca in Ingegneria Informatica e delle Telecomunicazioni nel 2003 presso l'Università degli Studi di Catania. Dal 2005 al 2007 ha lavorato come assegnista di ricerca presso il Dipartimento di Ingegneria Informatica e delle Telecomunicazioni dell'Università degli Studi di Catania occupandosi di riconoscimento dello stato emotivo degli individui attraverso l'analisi del segnale vocale inserito in un'attività di ricerca supportata da Telecom Italia Mobile (TIM).

Attualmente è ricercatore a tempo indeterminato e professore aggregato del settore telecomunicazioni presso il Dipartimento di Ingegneria dell'Università degli Studi di Messina dove è anche responsabile del laboratorio "Telecomunicazioni". L'attività di ricerca riguarda l'area dell'elaborazione del segnale (codifica e riconoscimento della voce, elaborazione di segnali biomedicali, elaborazione del segnale per le telecomunicazioni) e le reti di telecomunicazioni (wireless mesh network, voice over IP e wireless sensor network). È autore di oltre quaranta articoli scientifici apparsi su riviste internazionali o presentati a conferenze internazionali, revisore di diverse riviste internazionali e fa parte dell'Editorial Board della rivista International Journal of Distributed Sensor Networks (IJDSN).

Simone Beozzo, nato il 04/12/1982 a Rovereto (TN). Diploma di Perito Industriale e Capotecnico in Elettronica e Comunicazioni presso l'Istituto Tecnico Industriale Aldini Valeriani di Bologna. Dal 16/04/2012 ad oggi lavora per l'Agenzia Nazionale per le Nuove tecnologie, l'Energia e lo sviluppo economico sostenibile (ENEA) presso la sede di Bologna come Tecnico di Laboratorio (CTER) nel Laboratorio Laerte per lo sviluppo di software e dispositivi elettronici per la ricerca in ambito di efficientamento energetico. Dal 2003 al 2008 Per la Carlo Gavazzi - Sensors Division come tecnico progettista con ruolo di Progettazione ed Industrializzazione per sensori elettrici e di sicurezza. Dal 2001 al 2003 presso Innovatech Srl come ricerca e sviluppo per nuovi prodotti ed impiantistica nel settore domotico. Possiede esperienza nella progettazione e realizzazione di circuiti elettrici analogici e digitali e nella realizzazione di componentistiche meccaniche con utensili di precisione. Utilizzo di programmi di grafica, di progettazione CAD ed editing Audio – Video.

Roberto Guida nasce a Palermo nel 2002 consegue il diploma di perito industriale capotecnico con specializzazione in elettronica e telecomunicazioni presso l'ITIS A. Volta di. Attualmente iscritto al corso di laurea in Ingegneria Elettronica e Telecomunicazioni presso l'Università degli studi di Bologna. Dal 01/04/2005 al 15/04/2012 presta servizio come Collaboratore Tecnico Enti di Ricerca presso la sezione di Palermo INGV per lo sviluppo di dispositivi elettronici per il monitoraggio geochimico in aree vulcaniche e sismiche. Dal 16/04/2012 ad oggi presta servizio come CTER presso ENEA – Laboratorio Laerte per lo sviluppo di software e dispositivi elettronici per la ricerca in ambito di efficientamento energetico. Ha esperienza nella progettazione e realizzazione di circuiti elettronici analogici e digitali, nella programmazione di microcontrollori e sistemi embedded. Ha una conoscenza avanzata dei principali linguaggi di programmazione di alto e basso livello (C, C++, Java, assembler, etc…) e nella programmazione di logiche programmabili (VHDL, Verilog). Ha una conoscenza avanzata dei sistemi operativi Linux e Windows e nella creazione di siti e applicativi web (HTML5, CSS, Javascript, PHP, MySQL, etc…).