Indice generale

1.Introduzione....1
1.1.Sistemi Real-Time....1
1.2.Sistema embedded....2
1.3.“Internet of Things” e Wireless Sensor Network....3
1.3.1.Campi di applicazione....4
1.4.Motivazione....5
1.5.IEEE 802.15.4....7
1.5.1.Tipi di dispositivi e ruoli....7
1.5.2.Tipologia di rete....8
1.5.3.Archichettura del protocollo....9
1.5.4.Modalità di accesso al mezzo....11
1.6.IETF....12
1.6.1.Ipv6 over Low power WPAN....12
1.6.2.Politica di gestione Indirizzi....12
1.6.3.Compressione Header....13
1.7.Routing per reti opportunistiche....13
1.7.1.Funzionamento AODV....14
1.7.2.TinyAODV o AODV per IEEE 802.15.4 Networks....16
2.Specifica dei Requisiti....17
2.1.Descrizione generale del sistema....17
2.2.Requisiti funzionali....18
2.2.1.Addressing....18
2.2.2.Completamente connesso....18
2.2.3.Discovery....19
2.3.Requisiti non funzionali....19
2.3.1.Scalabilità....19
2.3.2.Eterogeneità....19
2.3.3.Estensibilità....20
2.3.4.Fault tolerance....20
2.3.5.Vincoli sull'implementazione....20
3.Ambiente di Sviluppo....21
3.1.Piattaforma Hardware....21
3.1.1.FLEX Embedded board....22

3.1.2.Architettura modulare....22
3.1.3.Microcontrollore dsPIC33FJ256MC710....25
3.2.Ambiente Software....28
3.2.1. ERIKA Enterprise....29
3.2.2.µWireless....32
3.3.Strumenti di debugging....33
3.3.1.Chipcon CC2420 Packet Sniffer....33
3.3.2.DAQ....35
4.Progettazione del Sistema....37
4.1.Definizione del problema....37
4.2.Archittettura del sistema....38
4.3.Topologia....39
4.4.Addressing....41
4.5.Neighbor discovery....43
4.6.Routing....45
4.7.Strutture dati....47
4.7.1.Strutture dati del LoWPAN Node....47
4.7.2.Strutture dati del LoWPAN Edge Router....48
5.Implementazione....49
5.1.Vincoli di implementazione....49
5.2.Il Tree di Erika....50
5.3.Implementazione dei moduli....51
5.3.1.Implementazione del modulo Network....53
5.3.2.Implementazione del modulo UDP....54
5.3.3.Implementazione del modulo AODV....54
5.4.Implementazione delle strutture dati....54
5.5.Implementazione applicazioni dimostrative....55
6.Risultati sperimentali....56
6.1.Installazione sperimentale....56
6.1.1.Hardware setup....56
6.1.2.Round trip daelay....57
6.1.3.Route formation time....58
6.2.Attività sperimentali....58
6.2.1.Addressing....59
6.2.2.Routing....61

6.2.3.Misura del Route formation time.....64
6.2.4.Misura del round trip delay.....65
Conclusioni.....67

Indice delle illustrazioni

Figura 1.1: Rigidità dei vincoli temporali in relazione alla criticità dell'applicazione.....2
Figura 1.2: Sistema complesso e Real-Time.....6
Figura 1.3: Topologie a Stella e Peer-to-Peer (Mesh e Cluster-Tree).....9
Figura 1.4: Lo stack di IEEE 802.15.4.....10
Figura 1.5: Invio messaggi RREQ nell'algoritmo AODV.....15
Figura 1.6: Invio messaggio RREP nell'algoritmo AODV.....16
Figura 2.1: Struttura generale del sistema.....18
Figura 3.1: Archichettura modulare della FLEX Board.....22
Figura 3.2: FLEX Base Boards.....24
Figura 3.3: FLEX Demo Board.....25
Figura 3.4: Schema generale del dsPIC®33 DSC.....27
Figura 3.5: Archichettura di ERIKA RTOS.....30
Figura 3.6: Screenshot del framework Eclipse con il plugin RT-Druid.....32
Figura 3.7: Archichettura stratificata del sistema di comunicazione µWireless compliant con lo standard IEEE 802.15.4 basato su Erika.....33
Ifigura 3.8: La board Texas Instruments CC2420 EB con l'estensione CC2420EM.....34
Figura 3.9: Interfaccia grafica del Chipcon CC2420 Packer Sniffer.....34
Figura 3.10: DAQ.....36
Figura 4.1: Rappresentazione dello stack µWireless.....39
Figura 4.2: Simple LoWPaN.....40
Figura 4.3: Extended LoWPAN.....40
Figura 4.4: Indirizzo IPv6.....41
Figura 4.5: Stateless Address Autoconfiguration.....42
Figura 4.6: Esempio suffisso dell'indirizzo IPv6 ottenuto con la procedura Stateless Adress Autoconfiguration.....43
Figura 4.7: Sequenza dei messaggi Router Solicitation e Router Advertisement.....44
Figura 4.8: Sequenza messaggi Node Registration e Node Confirmation.....45
Figura 5.1: Tree di ERIKA.....50
Figura 5.2: Collocazione dei file .c e .h 52
Figura 5.3: Esempio di definizione delle funzioni.....52
Figura 5.4: Esempio di struttura dati.....55
Figura 5.5: Esempio di definizione di una tabella.....55
Figura 6.1: Configurazione del sistema in fase di test.....57

Figura 6.2: Sequenza messaggi in fase di start-up....59
Figura 6.3: Router Solicitation con header compression....60
Figura 6.4: Router Solicitation (RS)....60
Figura 6.5: Router Advertisement (RA)....61
Figura 6.6: Node Registration (NR)....61
Figura 6.7: Node Confirmation (NC)....61
Figura 6.8: Sequenza messaggi AODV....62
Figura 6.9: RREQ inviato dal nodo sorgente....62
Figura 6.10: RREQ inviato da un nodo della rete....62
Figura 6.11: RREP inviato dal destinatario....63
Figura 6.12: RREP inviato da un nodo della rete....63
Figura 6.13: Tempi per creare un percorso....64
Figura 6.14: Messaggio inviato nel test di Round Trip....65
Figura 6.15: Tempi di Round Trip....66

Capitolo 1

1. Introduzione

1.1. Sistemi Real-Time

I sistemi real-time sono quei sistemi di calcolo la cui correttezza di funzionamento non dipende soltanto dalla validità dei risultati ottenuti ma anche dal tempo in cui i risultati sono prodotti [1]. Questi sistemi vengono utilizzati solitamente in ambito industriale (controllo di processo, pilotaggio di robot, trasferimento di dati nelle telecomunicazioni) o comunque dove sia necessario ottenere una risposta dal sistema entro un tempo prefissato.

I sistemi real-time non devono essere necessariamente veloci: non è importante l'intervallo di tempo in cui il sistema deve reagire; l'importante è che risponda entro un tempo massimo determinato. In altre parole il sistema deve essere prevedibile. Per prevedibilità si intende la capacità di determinare in anticipo (ad esempio all'istante di creazione) se uno o più processi potranno essere completati entro i proprio vincoli temporali.

I task di questi sistemi rappresentano le attività che devono essere completate entro le loro deadline. Essi sono rilasciati periodicamente, task periodici, o possono essere stimoli dell'ambiente esterno, task aperiodici, e quest'ultimi devono essere processati in maniera concorrente con i primi. La modalità con cui il sistema gestisce la programmazione concorrente è detta politica di schedulazione e può essere basata su priorità assegnate a ciascun attività.

La criticità della specifica applicazione determina quanto rigorosamente dovranno essere

rispettati i vincoli temporali (fig. 1.1).

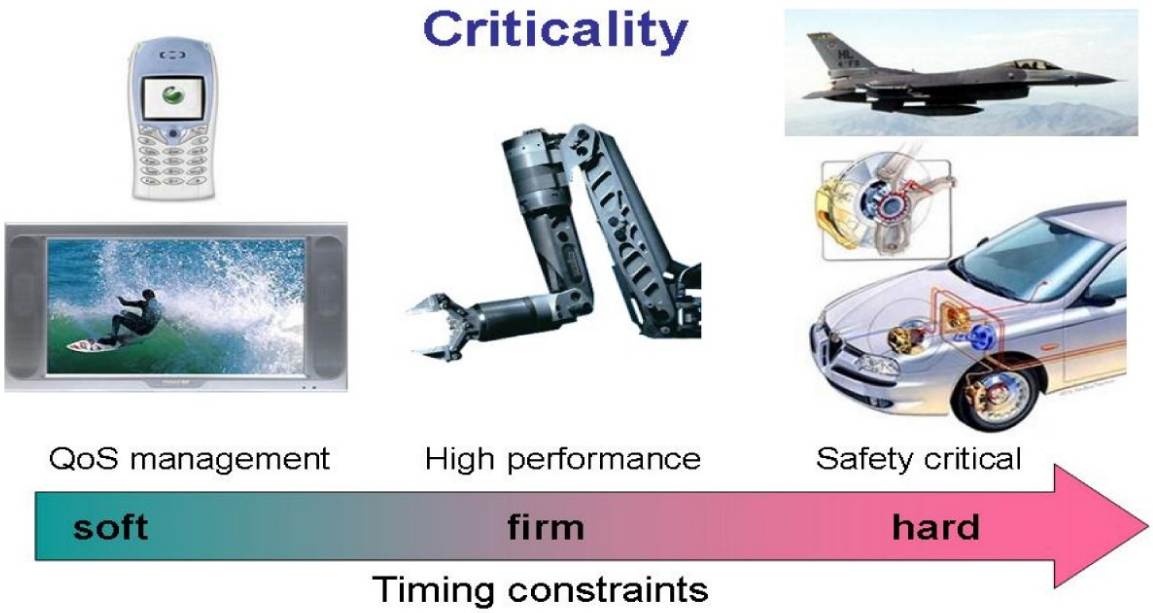


Figura 1.1: Rigidità dei vincoli temporali in relazione alla criticità dell'applicazione

1.2. Sistema embedded

I sistemi embedded sono tutti quei sistemi di elaborazione che non rientrano nella categoria *general-purpose*. I sistemi embedded sono circuiti integrati di elementi elettronici ed unità di calcolo, e contrariamente ai computer generici, ha dei compiti conosciuti già durante lo sviluppo, che eseguirà grazie ad una combinazione hardware/software specificamente studiata per tale applicazione.

Tali sistemi possono essere realizzati con pure tecniche elettroniche oppure mappando alcune funzionalità richieste in software. Seguendo questa linea, l'utilizzo di microcontrollori programmabili ha esteso il dominio delle applicazioni dei sistemi embedded e la loro flessibilità applicativa. L'incremento delle prestazione dei dispositivi, nonchè la loro miniaturizzazione, fa sì che oggi i sistemi basati su microcontrollori siano una delle soluzioni principali in molti settori dell'industria (automotive, controllo industriale) e del mercato consumer (cellulari, lettori MP3). Nel 2008 sono state vendute oltre 4 miliardi di unità di sistemi embedded; le previsioni parlano

di 16 miliardi di unità in funzione entro il 2010 e 40 miliardi entro il 2020[2].

1.3. “Internet of Things” e Wireless Sensor Network

L'espressione "Internet of Things" indica il collegamento di internet al mondo reale composto da oggetti e da luoghi concreti. Il concetto dell'”Internet delle cose” è attribuito all'Auto-ID Center, fondato nel 1999[3]. I sensori associati ad un oggetto possono identificarlo univocamente e raccogliere informazioni in tempo reale su parametri del suo ambiente come: temperatura, localizzazione, pressione, rumore, luce, umidità. Questo nuovo scenario tecnologico unito al progresso delle tecnologie wireless, come le WSN, offre l'opportunità di sviluppare una serie di applicazioni innovative “context-based” che si prevede negli anni futuri entreranno nella nostra società nei più svariati campi applicativi. Se consideriamo la quantità e la varietà degli oggetti da gestire possiamo capire i problemi da risolvere per la raccolta ed instradamento dei dati. Sicuramente le tecnologie wireless saranno una componente predominante nella rete di raccolta, per la facilità di installazione e la qualità raggiunta dai vari standard di comunicazione.

Una Wireless Sensor Network (WSN) è una rete wireless costituita da un insieme di dispositivi autonomi dotati di sensori distribuiti nell'ambiente, che permettono di rilevare fenomeni fisici o monitorare condizioni ambientali. I nodi di una WSN sono sistemi embedded dotati di radio transceiver, o altro tipo di apparato wireless, un piccolo microcontrollore e di strumenti di misurazione. Le reti di questi dispositivi costituiscono l'infrastruttura per la raccolta, fusione e aggregazione dei dati immagazzinati nei nodi sensori. Un aspetto molto interessante di questa tecnologia è la possibilità di implementare un meccanismo multi-hop per il routing su reti di vasta area.

Uno dei principali vantaggi delle WSN è la versatilità, ovvero sono in grado di supportare applicazioni molto differenti tra loro. Per questo sono particolarmente adatte a diversi campi applicativi, fra cui il controllo della salute del cittadino, la sicurezza, il controllo ambientale e del traffico, oltre a quelle in ambito militare, le quali hanno un elevato contenuto di innovazione. Altri vantaggi delle WSN rispetto ai tradizionali sistemi di misurazione sono la pervasività del monitoraggio e un'intrinseca tolleranza ai guasti.

1.3.1. Campi di applicazione

I campi di applicazione di una rete di sensori sono molteplici. Ciò è dovuto al fatto che i sensori possono essere di diverso tipo e quindi capaci di rilevare grandezze fisiche diverse come temperatura, umidità, pressione, luce, ma anche capaci di rilevare il movimento di veicoli, inquinamento, livello di rumore e molto altro. Di conseguenza le applicazioni in cui si possono usare le reti di sensori sono molteplici e si possono classificare in militari, ambientali, medico-sanitarie, domestiche e commerciali.

APPLICAZIONI MILITARI

In questo ambito nasce la ricerca sulle sensor network[4]. La facilità e la rapidità di distribuzione, l'auto-organizzazione e la tolleranza ai guasti sono caratteristiche adatte per applicazioni militari. L'impiego principale è il rilevamento ed il tracciamento di unità nemiche. È possibile usare una rete di sensori anche per effettuare la stima dei danni di una battaglia oppure per il riconoscimento della tipologia d’attacco (nucleare, biologico o chimico).

APPLICAZIONI AMBIENTALI

In questo ambito le WSN sono usate soprattutto per il monitoraggio di disastri ambientali, rilevando prontamente eventuali incendi, inondazioni[5], [6] e in quest’ultimo caso prevenendole[7]. Altri utilizzi si ritrovano nell'agricoltura di precisione, dove è possibile monitorare il livello dei pesticidi nell'acqua, il livello di erosione del terreno e il grado di inquinamento nell'aria[8].

APPLICAZIONI MEDICO-SANITARIE

L’uso di queste applicazioni nel settore medico-sanitario sono rivolte all'amministrazione ospedaliera sia essa relativa ai pazienti che ai medici. Si occupano della rintracciabilità, il monitoraggio di dati fisiologici riguardanti l'uomo e forniscono un'interfaccia per le persone affette da handicap.

APPLICAZIONI DOMESTICHE

Inserendo sensori nei vari elettrodomestici (forno, frigorifero, termostato, etc.), l’uso delle WSN consiste nell’automazione della casa. Questi nodi possono interagire l'uno con l'altro e anche con reti esterne tramite l'utilizzo di internet, così da poter permettere la gestione anche da distanze remote. Una tecnologia in questo ambito è l'Ambient Assisted Living (AAL)[9].

APPLICAZIONI COMMERCIALI

Tra i principali utilizzi delle WSN in questo ambito, troviamo il rilevamento della posizione e del movimento di veicoli (car tracking)[10], è possibile monitorare la posizione di un'auto in movimento o monitorare il traffico; il controllo dell'ambiente in ufficio, gestione del riscaldamento o dell'aria condizionata; gestione del magazzino, quantità di merce in giacenza o esaurimento scorte di una data merce (Inventory System)[11].

1.4. Motivazione

Come già accennato, la "Internet of the Things" prevede la possibilità di interagire con dispositivi embedded dislocati su vaste aree geografiche e connessi "in rete".

Per realizzare questa sfida le tecnologie ad oggi disponibili su "sistemi embedded" devono essere estese per quello che riguarda la risorse ed i servizi di rete in maniera da realizzare una rete pervasiva completamente connessa.

La comunità scientifica è al momento impegnata nella standardizzazione dei servizi e delle interfacce per la realizzazione di una architettura di rete di dimensione planetaria (in particolare per i sistemi embedded): questo sforzo si concretizza nelle iniziative della Internet Engineering Task Force volte all'approvazione dei documenti RFC, descritte più avanti (cfr. +).

Chiaramente lo sforzo deve essere diretto a fruire della tecnologia e degli standard esistenti di modo da rendere operativi e fruibili taluni servizi ipotizzati quali il trasferimento dei dati in una topologia Peer to Peer che rendono necessaria la codifica del Livello 3 (Networking) dello stack di comunicazione.

Nel 2003 ha avuto luogo la standardizzazione dei livelli fisici e MAC per una Wireless Personal Area Network; lo standard, approvato da IEEE e denominato 802.15.4, prevede che le reti WSN permettano una connessione P2P (attraverso una topologia mesh o gerarchica) anche se non specifica come questa debba essere implementata.

Un primo tentativo di realizzazione delle componenti a livello Network ed Application è avvenuta nell'anno 2004 da parte di un gruppo di aziende consorziatesi sotto il nome di Zigbee[12], con l'obiettivo di definire un insieme di protocolli di comunicazione ad alto livello che utilizzano piccole antenne digitali e dispositivi microcontrollore a bassa potenza. Zigbee non recepisce gli obiettivi dell'IPv6 e dunque sembra un approccio parallelo per la "Internet of the Things" rispetto a quello proposto dall' IETF.

In questa tesi, partendo dai documenti dell'IETF si intende realizzare uno stack di

comunicazione compatibile con il livello MAC IEEE 802.15.4 di modo da supportare la topologia mesh e permettere la comunicazione P2P. La confluenza di know-how e tecnica per l'implementazione di reti WSN con sistemi operativi e protocolli di rete aventi caratteristiche di tempo reale permette di realizzare sistemi distribuiti topologicamente complessi come mostrato pittoricamente in Figura 1.2.

La soluzione proposta in questa tesi trova un'implementazione esistente di riferimento nell'ambito del progetto Contiki portato avanti dallo Swedish Institute of Computer Science[13]. In quell'ambito lo stack IPv6 è stato provato sperimentalmente sui MAC Ethernet (cablato) e IEEE 802.15.4 (wireless).

Questo software non gira su piattaforme MicroChip © , piattaforme sulle quali, di contro, gira il sistema operativo ERIKA ed i protocolli di comunicazione real-time conformi allo standard IEEE 802.15.4.

A valle del lavoro di tesi, avendo parallelamente completato il porting del sistema operativo su piattaforme condivise con Contiki (quale il TI MSP430, alla base della board TelosB commercializzata da CrossBow ®), sarà possibile una campagna di misura sperimentale mirata alla valutazione comparativa delle funzionalità e delle performance implementate sull'IPv6.

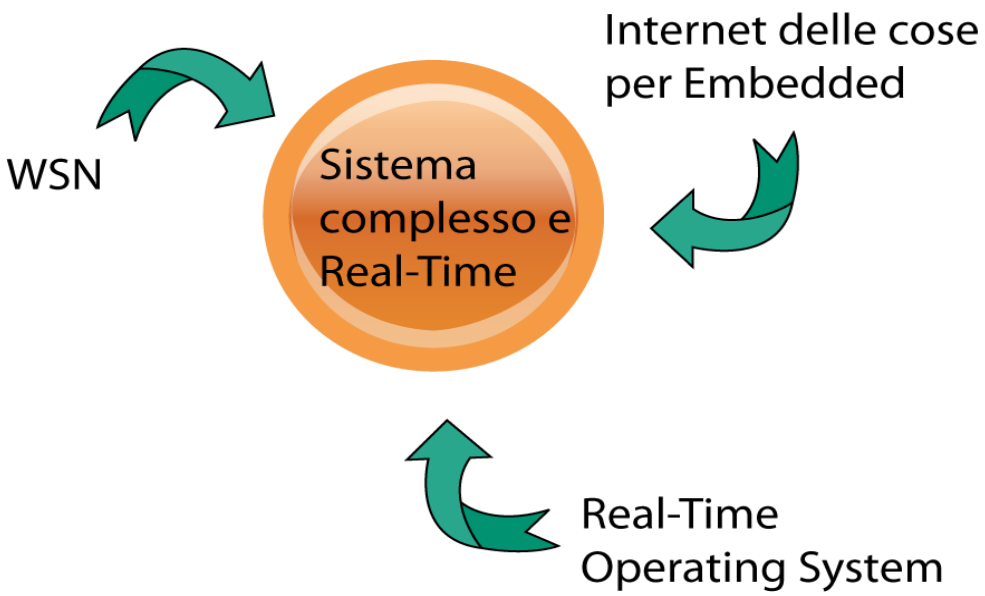


Figura 1.2: Sistema complesso e Real-Time

1.5. IEEE 802.15.4

IEEE 802.15.4 viene approvato nel 2003[14] (ed aggiornato nel 2006) e definisce il protocollo di trasmissione a basso livello tramite comunicazione radio tra diversi dispositivi di una WPAN (Wireless Personal Area Network) a basso rate trasmissivo (LR-Low Rate). Lo standard definisce in particolare le specifiche del livello fisico ed del livello MAC (Medium Access Control) con lo scopo di garantire una modalità di connessione wireless a basso data-rate tra i vari dispositivi che necessitano di un basso consumo di potenza così da garantire una lunga durata delle batterie a bordo e che operano in uno spazio operativo dell'ordine di qualche decina di metri. Una Lr-Low Rate è una rete di comunicazione semplice e a basso costo, selettivamente orientata verso applicazioni a basso consumo ed a "throughput" non elevato, principalmente caratterizzata da:

- Data-rate di 250 kbit/s, 100 kbit/s, 40 kbit/s e 20 kbit/s
- Tipologia di rete a stella o "peer-to-peer"
- Indirizzo a 64 bit o a 16 bit
- Allocazione opzionale di guaranteed time slots (GTSs)
- Accesso al canale in modalità CSMA-CA
- Completa definizione del protocollo per il trasferimento dei dati
- Basso consumo di energia
- 16 canali nella banda attorno a 2.4GHz, 10 canali nella banda attorno a 915MHz, un canale ad 868MHz.

1.5.1. Tipi di dispositivi e ruoli

Lo standard 802.15.4 definisce due ruoli per i nodi di una WPAN: device e coordinator o PAN coordinator. I primi sono semplici nodi sensori con lo scopo di rilevare le misurazioni nell'ambiente in cui è collocato e di inviare il risultato di tali misurazioni ai secondi. I coordinator o PAN coordinator sono responsabili della gestione della rete/sottorete WPAN, e sono più complessi rispetto ai device. Nello standard vengono definiti due tipi di nodi: Reduced Function Device (RFD) e Full Function Device (FFD). Gli RFD che possono operare nella rete solo come Device, dispongono solo di un sottoinsieme delle funzionalità MAC e quindi possono comunicare solo con altri RFD. Gli FFD possono operare nella rete come PAN Coordinator, come semplice Coordinator oppure come Device e dispongono di tutte le primitive MAC e

quindi possono comunicare sia con altri FFD sia con dispositivi RFD.

1.5.2. Tipologia di rete

Una WPAN è costituita da un minimo di due dispositivi. In ciascuna rete uno solo di essi può configurarsi con PAN Coordinator ed ha il compito di iniziare, gestire e terminare la comunicazione tra i diversi nodi. Una LR-WPAN può configurarsi in due topologie: topologia a stella e topologia peer-to-peer. Nella topologia a stella ciascun dispositivo può comunicare solo con il coordinatore, il quale controlla e gestice la rete. Un device ha associato una applicazione ed è o il punto iniziale o il punto terminale di una comunicazione. Il coordinator è tipicamente collegato ad alimentazione fissa a differenza degli altri dispositivi che sono dotati di batteria.

Una topologia peer-to-peer differisce dalla precedente in quanto ciascun dispositivo può comunicare direttamente con un altro interno alla rete, a patto che questo rientri nella sua area operativa di copertura, senza ricorrere alla mediazione del coordinatore.

Questa topologia si presta bene alla formazione di reti di comunicazione decisamente complesse che coinvolgono potenzialmente un elevato numero di dispositivi. Esempi di applicazioni sono il controllo ed il monitoraggio industriale ed ambientale, applicazioni domotiche.

Un esempio di rete complessa fondata su una topologia peer-to-peer è il cluster-tree. Ogni dispositivo della rete può funzionare da coordinatore, fornendo servizi ad altri dispositivi coordinatori, anche al di fuori della propria sfera di influenza. Però, soltanto uno di questi coordinatori "locali" può configurarsi come coordinatore PAN, il quale è tipicamente dotato di maggiori risorse anche a livello computazionale.

Una rete peer-to-peer può permettere la comunicazione multi-hop tra qualunque device della rete. Queste funzioni, tuttavia, non sono parte dello standard, ma devono essere fornite dai livelli superiori dello stack. In Figura 1.3 sono illustrate le varie tipologie di rete.

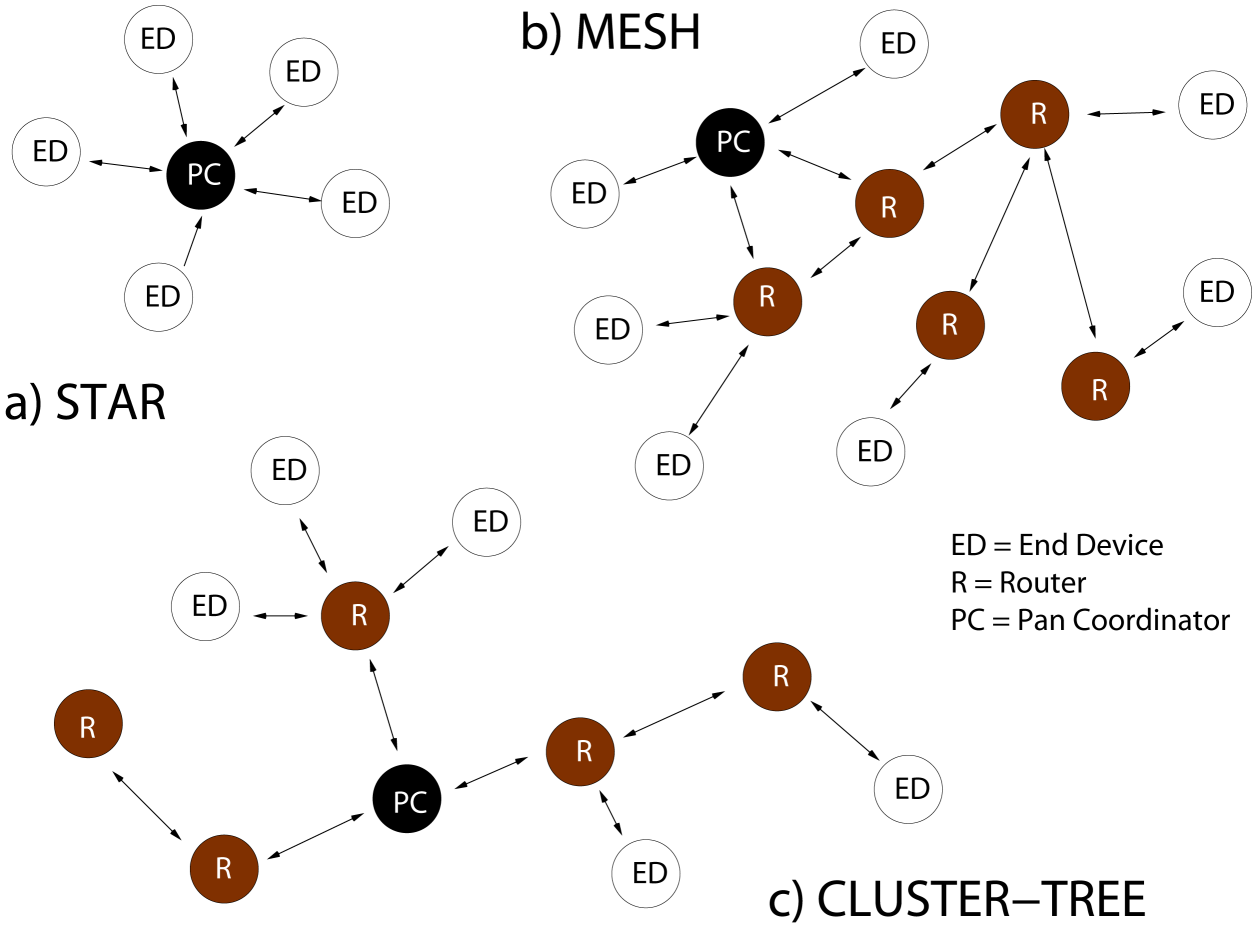


Figura 1.3: Topologie a Stella e Peer-to-Peer (Mesh e Cluster-Tree)

1.5.3. Archichettura del protocollo

I livelli definiti dallo standard 802.15.4 sono basati sulla pila ISO/OSI e sono solamente i primi due: il livello fisico (PHY) ed il livello datalink (MAC). L'interazione con i livelli superiori è prevista tramite un sub-layer LLC (Logical Link Control) ed un SSCS (Service Specific Convergence Sublayer). Il primo definisce i tipi di servizio ottenibili per un trasferimento dati mentre il secondo esegue le azioni necessarie per richiedere la trasmissione dati in base al servizio richiesto. In Figura 1.4 è illustrata l'archichettura del protocollo.

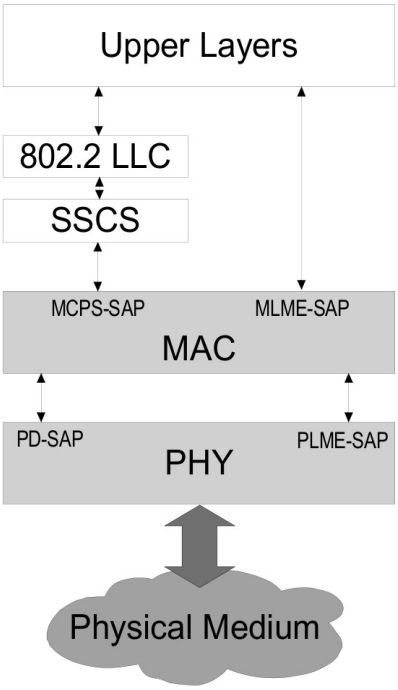


Figura 1.4: Lo stack di IEEE 802.15.4

IL LIVELLO FISICO

Il livello fisico fornisce due differenti servizi: quello dati e quello gestione. Le caratteristiche principali implementate a questo livello sono l'attivazione e la disattivazione del trasmettitore radio, il rilevamente dell'energia (ED), l'indicazione della qualità del collegamento (LQI), la selezione del canale, la stima della disponibilità del canale (CCA) e la trasmissione e ricezione di pacchetti sul mezzo fisico (canale radio).

I bitrate offerti dal PHY sono mostrati in tabella 1.1.

Data Rate	Symbol Rate	Banda	Canali	Modulazione
20 Kbps	20 Ksyms	868 – 868.6 Mhz	1	BPSK
40 Kbps	40 Ksyms	905 928 Mhz	10	BPSK
250 Kbps	62.5 Ksyms	2.4 – 2.483 Ghz ISM	16	O-QPSK

Tabella 1.1: PHY Bitrates

IL LIVELLO MAC

I principali compiti del livello MAC sono il coordinamento dell'accesso al mezzo da parte del tranciver, creazione ed instradamento dei pacchetti, generazione e riconoscimento dell'indirizzo, verifica del numero di sequenza dei pacchetti.

Le principali funzioni del livello MAC sono implementate in software a differenza di quanto avviene per il livello fisico e scritte generalmente in linguaggio C.

Il livello MAC definisce 4 possibili tipi di frame:

- Frame di dati, costituito al massimo da 128 bytes è utilizzato per tutti i trasferimenti di dati;
- Frame ACK, fornisce la conferma che il pacchetto inviato è stato ricevuto correttamente;
- Frame di comando MAC, fornisce un meccanismo per il controllo e la configurazione da remoto dei nodi client;
- Frame di beacon, utilizzato dal coordinator per definire la struttura del superframe (se utilizzato) e per permettere ai device di localizzare la rete.

1.5.4. Modalità di accesso al mezzo

Dato che il mezzo radio è condiviso da tutti i dispositivi, è necessario disporre di qualche metodo per la trasmissione, onde evitare che due dispositivi inviino pacchetti contemporaneamente. Esistono due tecniche utilizzate: il beacon-enabled ed il non beacon-enabled.

La prima è basata su una struttura a superframe che si ripete periodicamente e grazie alla quale i nodi sono in grado di sincronizzarsi con il coordinator e quindi di disciplinare la comunicazione tra i vari nodi della rete.

La seconda non prevede l'uso di un superframe di sincronizzazione della comunicazione, pertanto l'accesso al canale avviene in modo completamente asincrono.

In entrambe le tecniche i device si devono associare al nodo coordinator e quest'ultimo deve:

- mantenere aggiornata la lista dei device associati;
- inviare periodicamente il frame beacon se in beacon mode;
- scambiare pacchetti dati con i deice o con altri coordinator.

1.6. IETF

La Internet Engineering Task Force[15] è una comunità aperta di tecnici, specialisti e ricercatori interessati all'evoluzione tecnica e tecnologica di internet. L'obbiettivo dell'IETF è quello di migliorare internet attraverso la creazione di documenti tecnici, di alta qualità a supporto della progettazione, uso, e gestione di internet.

La differenza di questo ente rispetto ad altri è la sua struttura aperta: il lavoro viene svolto da gruppi di lavoro (working groups) che operano soprattutto tramite Mailing list, aperte alla partecipazione di chiunque sia interessato. Ogni gruppo si occupa di uno specifico argomento e sono organizzati in aree. I gruppi di lavoro producono documenti denominati RFC (Request For Comments) che vengono sottoposti alla IESG (Internet Engineering Steering Group) per il loro avanzamento a standard ufficiale.

Di interesse per questa tesi sono i working group "Ipv6 over Low power WPAN (6lowpan)"[16] ed il working group “Mobile Ad-hoc Networks (MANET)”[17].

1.6.1. Ipv6 over Low power WPAN

Lo scopo principale di questo working group è di definire il trasporto Ipv6 sullo standard IEEE 802.15.4. Il gruppo di lavoro ha completato due RFC:

- "Ipv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919;
- "Transmission of Ipv6 Packets over IEEE 802.15.4 Networks, RFC 4944.

Nel primo vengono definiti i requisiti, le motivazioni, i problemi e gli obiettivi legati alla proposta di standard che viene specificata nel secondo RFC. In quest'ultimo si definiscono i dettagli tecnici e i formati dei pacchetti utilizzati nel 6LoWPAN. Inoltre sono stati pubblicati diversi draft, verranno elencati quelli di interessi per questa tesi:

- "Compression Format for IPv6 Datagrams in 6LoWPAN Networks";
- "Design and Application Spaces for 6LoWPANs";
- "6LoWPAN Neighbor Discovery".

1.6.2. Politica di gestione Indirizzi

L'IEEE 802.15.4 definisce due tipologie di indirizzi: indirizzi estesi IEEE EUI-64 bit, univoci

globalmente, o indirizzi short a 16 bit, univoci soltanto ell'interno della PAN. Emtrambi i tipi sono supportati dal 6LoWPAN.

Per utilizzare indirizzi short a livello di rete, dato che essi sono disponibili soltanto dopo l'assegnamento da parte del coordinatore della PAN e la loro validità è limitata unicamente alla validità dell'associazione, il 6LoWPAN impone che l'utilizzo venga effettuato sotto precise condizioni ed entro certi limiti.

In 6LoWPAN si assume che una PAN sia mappata all'interno di uno specifico link Ipv6: questo comporta l'utilizzo di un prefisso comune per i dispositivi di rete.

Questa assunzione ha importanti conseguenze nella gestione degli indirizzi per tre aspetti:

- gestione della comunicazione multicast;
- autoconfigurazione della rete;
- creazione e manipolazione indirizzi link local.

1.6.3. Compressione Header

Nel caso più sfavorevole in un frame IEEE 802.15.4 siano disponibili per i livelli superiori soltanto 81 byte. Considerando che l'header Ipv6 occupa da solo 40 byte, rimarrebbero soltanto 41 byte per i dati dei livelli superiori. Nel caso venisse usato anche il protocollo UDP dovremmo considerare un ulteriore header di 8 byte e, di conseguenza, soltanto 33 byte per il livello applicativo.

La proposta di standard definisce un meccanismo di compressione degli header,allo scopo di eliminare le informazioni ridondanti, prevedendo un'integrazione tra il livello 3, 6LoWPAN, e il livello 2, MAC 802.15.4, della stack protocollare.

1.7. Routing per reti opportunistiche

Come già accennato, una WSN può ricoprire vaste aree e di conseguenza può essere composta da un gran numero di dispositivi. Questo porta a dover risolvere il problema dell'instradamento dei pacchetti al fine di ottenere un sistema completamente connesso. È impensabile ottenere questa caratteristica con collegamenti a singolo hop, per questo le WSN si stanno indirizzando verso tecniche multi-hop opportunistiche.

Con il termine reti opportunistiche viene identificato un particolare tipo di reti mobili in grado

di fornire una insfrastruttura ad hoc sfruttando link radio a corto raggio all'interno di un'area. Le reti opportunistiche stabiliscono connessioni temporanee tra i nodi per il forwarding dei pacchetti utilizzando l'opportuinità di connessione offerta da un'altra stazione all'interno del raggio di comunicazione: per questo sono in genere tolleranti ai guasti eventuali sul singolo link.

In questo scenario i classici algoritmi di routing basati su IP non sono funzionali. Infatti i suddetti algoritmi hanno come assunzione principale l'assegnamento in maniera gerarchica di indirizzi ai vari nodi e l'utilizzo di tabelle indicanti il prossimo hop dove indirizzare il pacchetto in base all'indirizzo del destinatario trovato. Questo richiede, oltre ad una struttura molto stabile e gerarchizzata, grandi quantità di memoria per le tabelle di routing. Nelle reti di sensori tutte queste condizioni non valgono e si deve riuscire a fare routing di pacchetti in un contesto dove i nodi possono essere disposti in maniera casuale e in grandi quantità, senza che abbiano grandi disponibilità di memoria. Un protocollo adatto alle reti di sensori è l'"Ad-Hoc On-Demand Vector" (AODV).

AODV è un algoritmo di routing per reti ad-hoc mobili, discusso nell'RFC 3561 prodotto dal working group "Mobile Ad-hoc Networks" dell'IETF, ed è un derivato per reti ad-hoc del protocollo Distance Vector. Quest'ultimo è un algoritmo di routing dinamico che tiene conto della lunghezza dei vari percorsi.

A differenza dei protocolli di routing più comuni in internet che individuano tutti i nodi ed i percorsi della rete indipendentemente dal loro uso (proactive), l'AODV ricerca i percorsi nella rete solo su richiesta (reactive).

1.7.1. Funzionamento AODV

L'algoritomo AODV è costituito da due processi: il Route Discover, permette di reperire le informazioni di routing per raggiungere una certa destinazione tramite lo scambio di messaggi, ed il Route Maintenance che ha il compito di mantenere valide le informazioni di routing.

In AODV la rete rimane completamente silente finchè non è richiesta una connessione per l'inoltro di un pacchetto di dati. Il protocollo definisce i seguenti pacchetti per cercare i percorsi sulla rete:

- Route request (RREQ)
- Route reply (RREP)
- Route error (RERR)

L'invio di questi pacchetti avviene tramite il protocollo UDP.

Per garantire l'assenza di cicli nei percorsi utilizzati ogni nodo possiede un numero di sequenza che cresce nel tempo. L'algoritmo di routing è distribuito in tutti i nodi della rete, infatti ogni componente memorizza un suo indice dei percorsi, che contiene l'indirizzo IP del prossimo nodo in direzione della destinazione (next hop), il suo numero di sequenza e la distanza complessiva indicata in salti (hops).

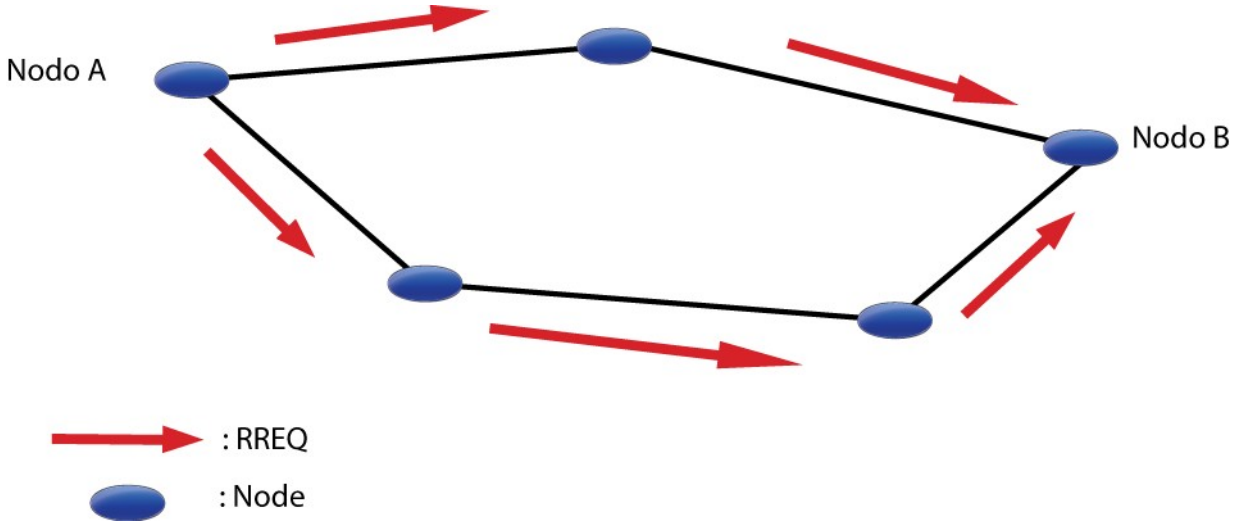


Figura 1.5: Invio messaggi RREQ nell'algoritmo AODV

Quando un nodo sorgente richiede una connessione ad un nodo destinatario di cui non ha informazioni di routing, il nodo sorgente invia una RREQ in broadcast generando un'esplosione di messaggi che vengono inoltrati attraverso tutta la rete, come illustrato in Figura 1.5. La RREQ conterrà oltre all'indirizzo della sorgente del destinatario, il Sequence Number della sorgente ed un valore di zero per il campo Hop Count. Ricevuta una RREQ un nodo esegue le seguenti azioni:

- controlla che non sia un duplicato;
- crea o aggiorna le informazioni di routing per la sorgente. L'aggiornamento viene eseguito solo se il Sequence Number della RREQ è maggiore dal nodo:
  - il Next Hop sarà il nodo da cui è arrivata la RREQ
  - l'Hop Count sarà impostato al valore della RREQ + 1
- se il nodo è il destinatario o conosce un percorso per raggiungere il destinatario risponde con una RREP

Ricevuta una RREP un nodo esegue le seguenti azioni:

- creare o aggiornare le informazioni di routing per la destinazione;
- se vengono ricevute più RREP per la stessa destinazione si sceglieranno le informazioni della RREP con un Hop Count minore.

Esempio di invio messaggio RREP è illustrato in Figura 1.6.

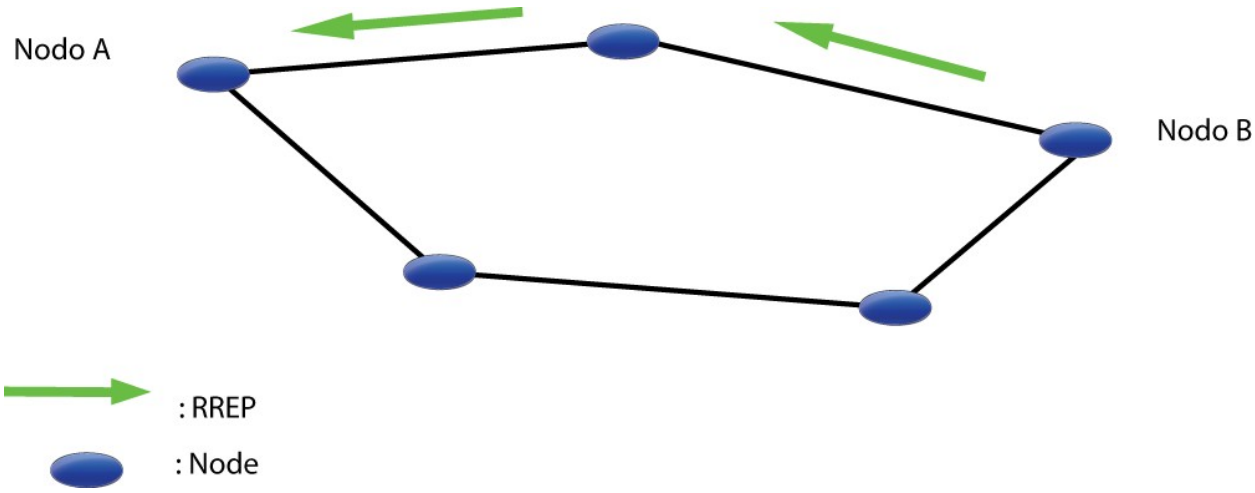


Figura 1.6: Invio messaggio RREP nell'algoritmo AODV

1.7.2. TinyAODV o AODV per IEEE 802.15.4 Networks

TinyAodv, discusso nell'internet draft “AODV for IEEE 802.15.4 Networks”, è una versione semplificata dell'algoritmo di routing "Ad-Hoc On-Demand Vector". Rispetto all'AODV introduce delle semplificazioni:

- i messaggi di RREP vengono generati solo dal nodo destinazione, e non da altri nodi eventualmente in possesso di un percorso per raggiungere la destinazione;
- viene usata solo una metrica "hop count";
- l'indirizzo per identificare la sorgente o il destinatario occupa meno spazio.

TinyAODV è stato qui proposto in quanto sarà adottato come protocollo di routig nel testbed adottato.

Capitolo 2

2. Specifica dei Requisiti

Seguendo le più comuni tecniche di ingegneria del software, la fase iniziale del lavoro è stata la raccolta e l'analisi dei requisiti. Nel seguito vengono riportati i principali requisiti individuati, suddivisi in funzionali e non funzionali.

2.1. Descrizione generale del sistema

Il progetto prevede la realizzazione di un set di applicazioni dimostrative ed un set di librerie con la finalità di avere un sistema distribuito con una topologia dinamica e complessa. Per dinamicità si intende la capacità del sistema ad adattarsi a cambiamenti della rete, come il variare del numero dei nodi o la variazione della posizione di un nodo. Per complessità, invece, si intende la possibilità di configurare la rete con tipologie più articolate rispetto alla semplice stella, previste dallo standard IEEE 802.15.4 ed implementabili solo a fronte della codifica di opportuni servizi di livello Network. Esempi di configurazioni sono reti mesh o reti ad albero.

Di seguito verranno esaminati i vari requisiti, funzionali e non, raccolti in fase iniziale. In Figura 2.1 è riportata la struttura generale del sistema.

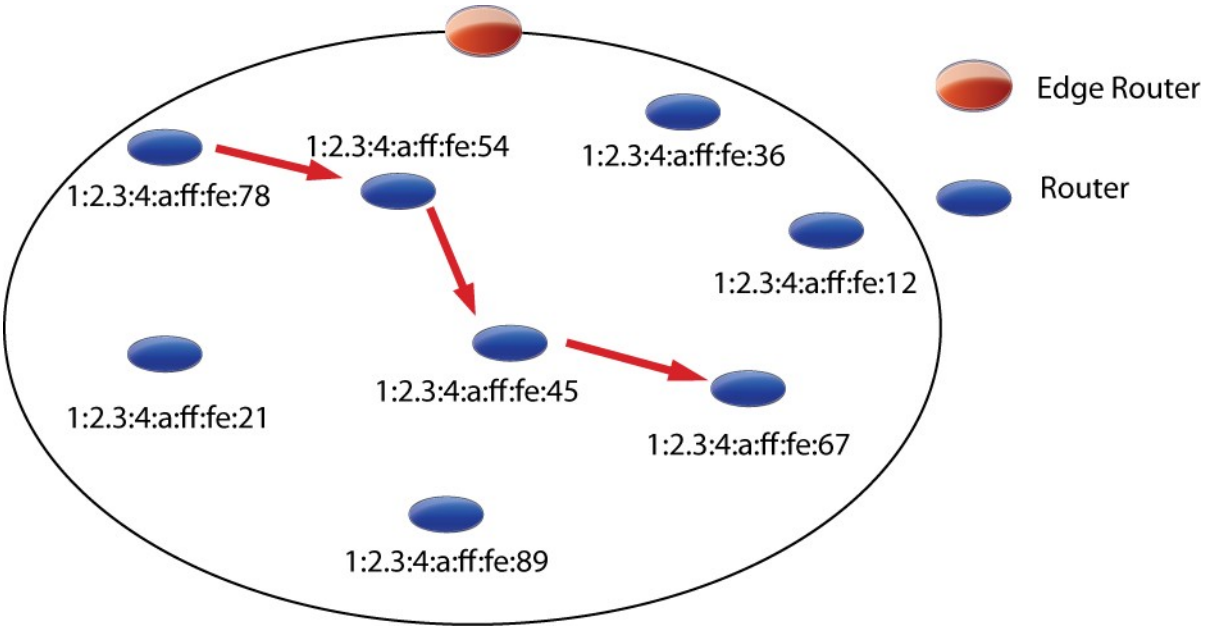


Figura 2.1: Struttura generale del sistema

2.2. Requisiti funzionali

I requisiti funzionali definiscono quali sono i servizi che il sistema dovrà fornire, ovvero quali sono le sue funzioni. Date le caratteristiche sopra citate, il sistema dovrà essere indirizzabile, assegnazione di un indirizzo di rete ad ogni nodo, e completamente connesso. Inoltre si dovrà fornire il servizio di "discovery", ovvero acquisizione dei parametri della rete, individuazione dei nodi vicini ed infine creazione di percorsi per inviare dati tramite il multi-hop.

2.2.1. Addressing

Ogni nodo del sistema per poter far parte della rete dovrà avere un indirizzo. Tale indirizzo, assegnato in fase di start-up, deve avere la caratteristica di unicità per poter identificare in modo univoco il dispositivo. Questo dato sarà presente nei dati inviati dal nodo, in qualità di indirizzo sorgente, e nei dati inviati ad esso, in qualità di indirizzo destinatario.

2.2.2. Completamente connesso

Tutti i dispositivi collegati alla rete dovranno avere la possibilità si comunicare con un qualsiasi altro dispositivo. Ciò non significa che ogni nodo ha un collegamento diretto con gli

altri nodi della rete, bensì si dovrà fornire un servizio per l'inoltro dei messaggi.

2.2.3. Discovery

In fase di start-up un dispositvo, oltre ad acquisire l'indirizzo di rete, deve configurare i vari parametri che caratterizzano la rete a cui si sta collegando. Quindi il sistema dovrà permettere l'acquisizione dei valori caratterizzanti la rete. Inoltre, si dovrà fornire un servizio che permetta la localizzazione dei nodi, in quanto essa non è fissata a priori. La comunicazione tra due dispositivi può avvenire o in modo diretto, singolo hop, o in modo indiretto, multi-hop. Nel secondo caso i percorsi che può effettuare un pacchetto sono molteplici, si dovrà scegliere la strada secondo una determinata metrica (es. numero minimo di hop).

2.3. Requisiti non funzionali

I requisiti non funzionali non sono collegati direttamente con le funzioni implementate dal sistema, ma piuttosto alle modalità operative e di gestione. Definiscono vincoli sullo sviluppo del sistema.

Oltre ai requisiti non funzionali che caraterizzano un sistema distribuito, quali la scalabilità, eterogeneità , estensibilità, fault tolerance, si aggiungono i requisiti riguardanti il software e l'hardware.

2.3.1. Scalabilità

In una WSN per scalabilità si intende la possibilità di aumentare il numero dei nodi senza che ciò comporti un eccessivo deterioramento delle prestazioni generali. Per prestazioni generali del sistema si intende l'utilizzo della memoria, il traffico dei pacchetti in rete, l'overhead di ogni singolo pacchetto. Ad esempio un nodo tiene traccia solo dei percorsi dei destinatari con cui deve comunicare.

2.3.2. Eterogeneità

Un sistema distribuito è detto eterogeneo se contiene componenti diverse in hardware e software. Il sistema dovrà essere realizzato in maniera tale da poter essere utilizzato su piattaforme diverse, consentendone anche un upgrade.

2.3.3. Estensibilità

Per estensibilità si intende la possibilità di inserire nuove funzionalità nel sistema senza doverlo reingegnerizzare. L'architettura dell'applicativo dovrà essere a blocchi, separati ed indipenti. Ciò permetterà l'aggiunta di estensioni, come la frammentazione dei pacchetti.

2.3.4. Fault tolerance

Fault tolerance o tollerana ai guasti è la capacità del sistema di non subire fallimenti anche in presenza di guasti. L'applicativo, in particolare, deve continaure a garantire i servizi offerti anche in caso di limitate failure locali.

2.3.5. Vincoli sull'implementazione

Per l'implementazione del sistema sono stati imposti vincoli ben precisi:

- il sistema dovrà essere scritto per il micro-kernel real-time ERIKA enterprise (cfr. +)
- utilizzo della libreria µWireless che implementa lo standard IEEE 802.15.4
- il sistema dovrà essere in grado di funzionare utilizzando come hardware per i nodi della WSN le schede FLEX (cfr. +).

Capitolo 3

3. Ambiente di Sviluppo

Questo capitolo si propone come obiettivo quello di introdurre l'architettura hardware e software utilizzata nel corso di questo lavoro. Si è scelto di non entrare troppo nel dettaglio nella descrizione dei componenti e dei protocolli, ma di fornire una panoramica necessaria per affrontare le problematiche implementative. Per quanto riguarda l'hardware si tratterà della scheda di sviluppo utilizzata, mentre per quanto riguarda il software si darà una definizione degli elemnti base del sistema operativo real-time Erika e della libreria µWireless. Infine si descriveranno in breve gli strumenti utilizzati per il debugging.

3.1. Piattaforma Hardware

I sistemi embedded devono necessariamente essere caratterizzati da una forte correlazione tra software ed hardware, obbligando il programmatore/progettista ad avere una chiara visione delle funzionalità e proprietà di entrambe le entità.

Per realizzare applicazioni su microcontrollore è necessario disporre di sistema hardware di supporto che prende il nome di evaluation board o scheda di sviluppo.

3.1.1. FLEX Embedded board

FLEX è una scheda embedded, il cui nucleo è rappresentato da microcontrollori della famiglia dsPIC®DSC di Microchip [18], frutto della collaborazione tra due aziende italiane: Evidence [19] ed Embedded Solutions [20]. Incarna a pieno le caratteristiche richieste da una evaluation board, in quanto permette lo sviluppo ed il testing immediato di applicazioni, consentendo l'incremento delle funzionalità fornite tramite moduli di espansione. Questo è dovuto a due caratteristiche fondamentali del sistema:

- architettura hardware modulare, unita ad una buona robustezza della parte elettronica;
- supporto del kernel real-time Erika Enterprise e del suo ambiente di sviluppo.

3.1.2. Architettura modulare

Nel dettaglio l'architettura FLEX prevede la presenza di una "scheda madre" principale (base board) e di un insieme di "schede figlie" (daughter boards), che una volta connesse possono dare origine a sistemi complessi ad elevato contenuto tecnologico. La Figura 3.1 riassume simbolicamente l'architettura.

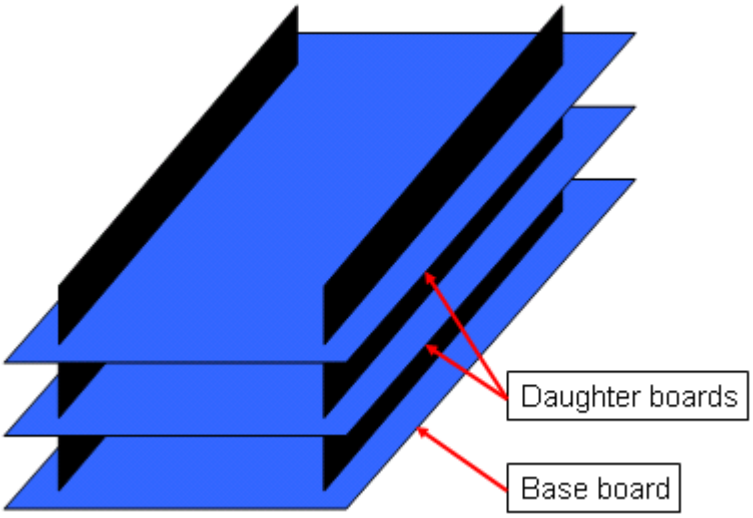


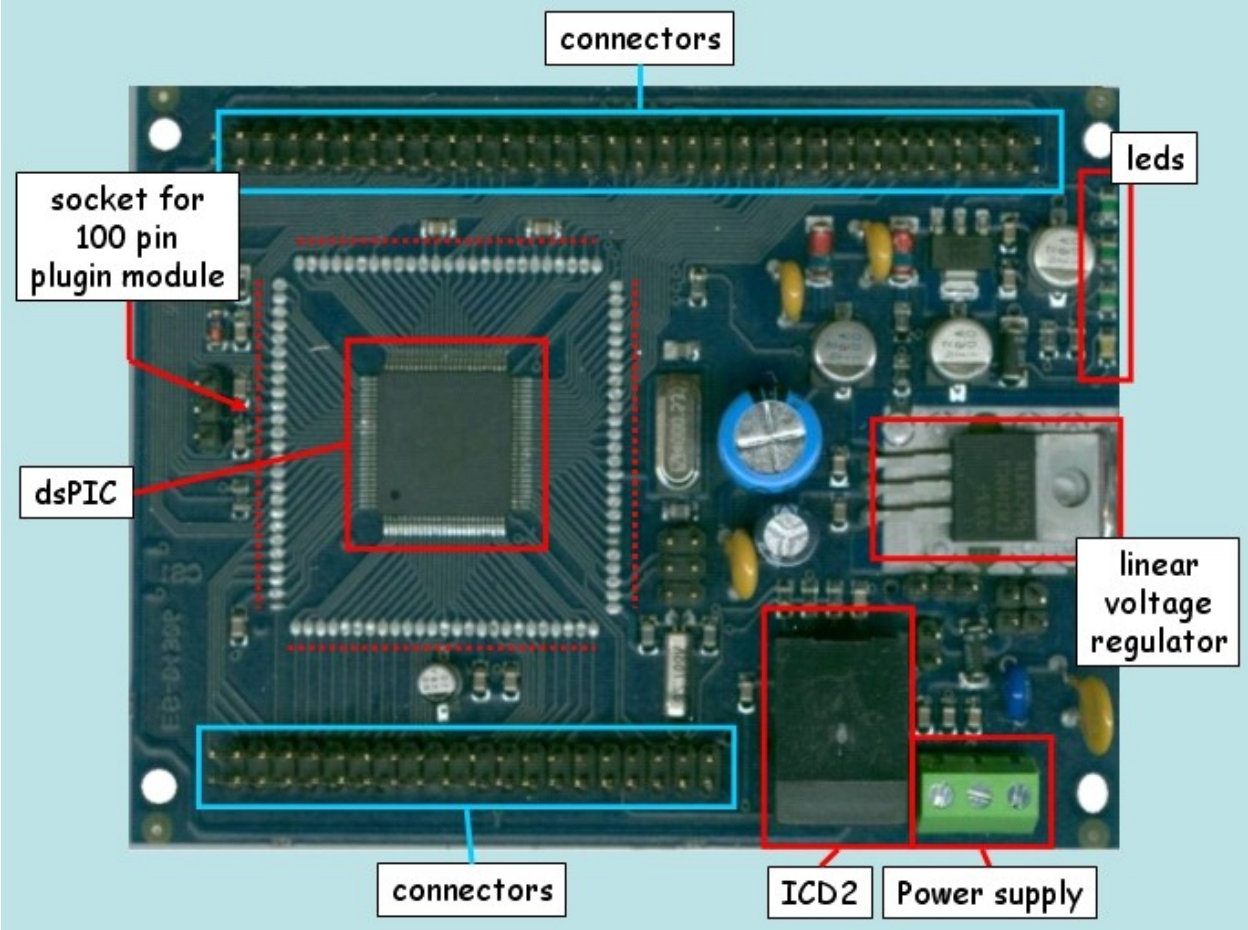
Figura 3.1: Archichettura modulare della FLEX Board

Base board

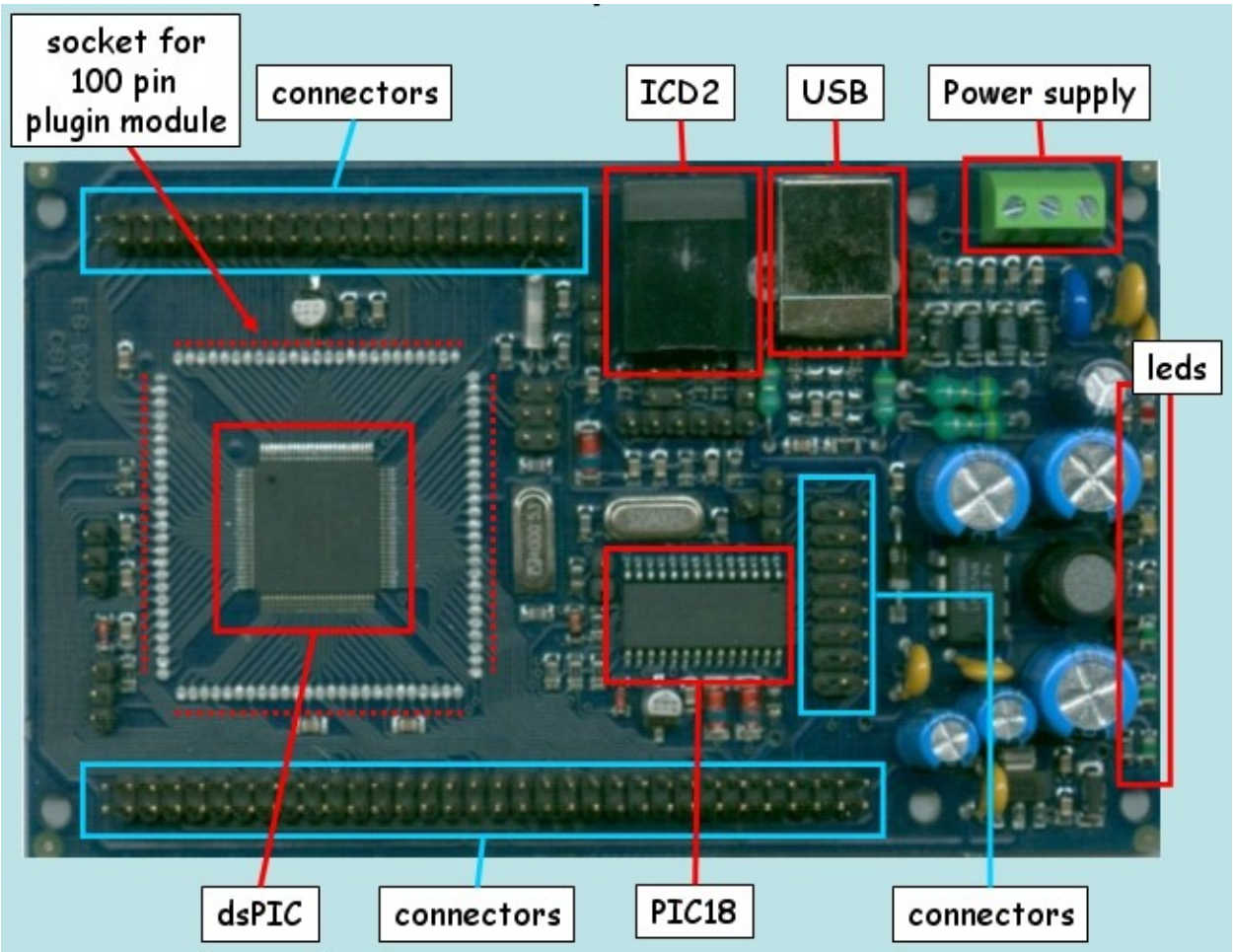
La FLEX base gestisce l'alimentazione e la programmazione del processore dsPIC33FJ256MC710 esportandone inoltre i 100 pin a due blocchi di connettori doppia fila di

passo 2.54 mm. Il blocco dei connettori riporta anche una tensione a 3,3 V, una a 5 V ed il riferimento di massa. Questa scelta produttiva consente l'interfacciamento con molti dispositivi esterni a bassa potenza senza richiedere l'uso di ulteriori blocchi di alimentazione.

Il prodotto è fornito in due versioni: FLEX Light e FLEX Full (fig 3.2) che si differenziano per dimensioni meccaniche e tipologia di alimentazione (di tipo lineare la prima con range di funzionamento 9-12 V, di tipo switching la seconda con range 9 -36 V). Da ricordare inoltre la presenza di led controllabili tramite settaggio di appositi registri, caratteristica che si è più volte dimostrata indispensabile in fase di debug del software.



(a) FLEX Light



(b) FLEX Full

Figura 3.2: FLEX Base Boards

Daughter board

Le schede figlie sono prive di processore che si interfacciano alla base board tramite i connettori precedentemente introdotti, ampliando notevolmente le funzionalità del sistema. Più daughter board possono essere impilate offrendo una personalizzazione del prodotto praticamente infinita.

In questo lavoro di tesi, nello specifico, sono state utilizzate varie FLEX Demo Board (fig. 3.3) le quali aggiungono alla base board interessanti funzionalità:

- un LCD (16 caratteri x 2 linee);
- un set di 4 bottoni;
- un set di 8 LED;
- un connettore per transceiver 802.15.4;

- un socket per moduli Multibus;
- 2 DAC (con risoluzione a 12 bit);
- un accelerometro a 3 assi;
- un buzzer;
- un potenziometro;
- un sensore di temperatura;
- un sensore di luminosità,
- un transceiver ad infrarossi.

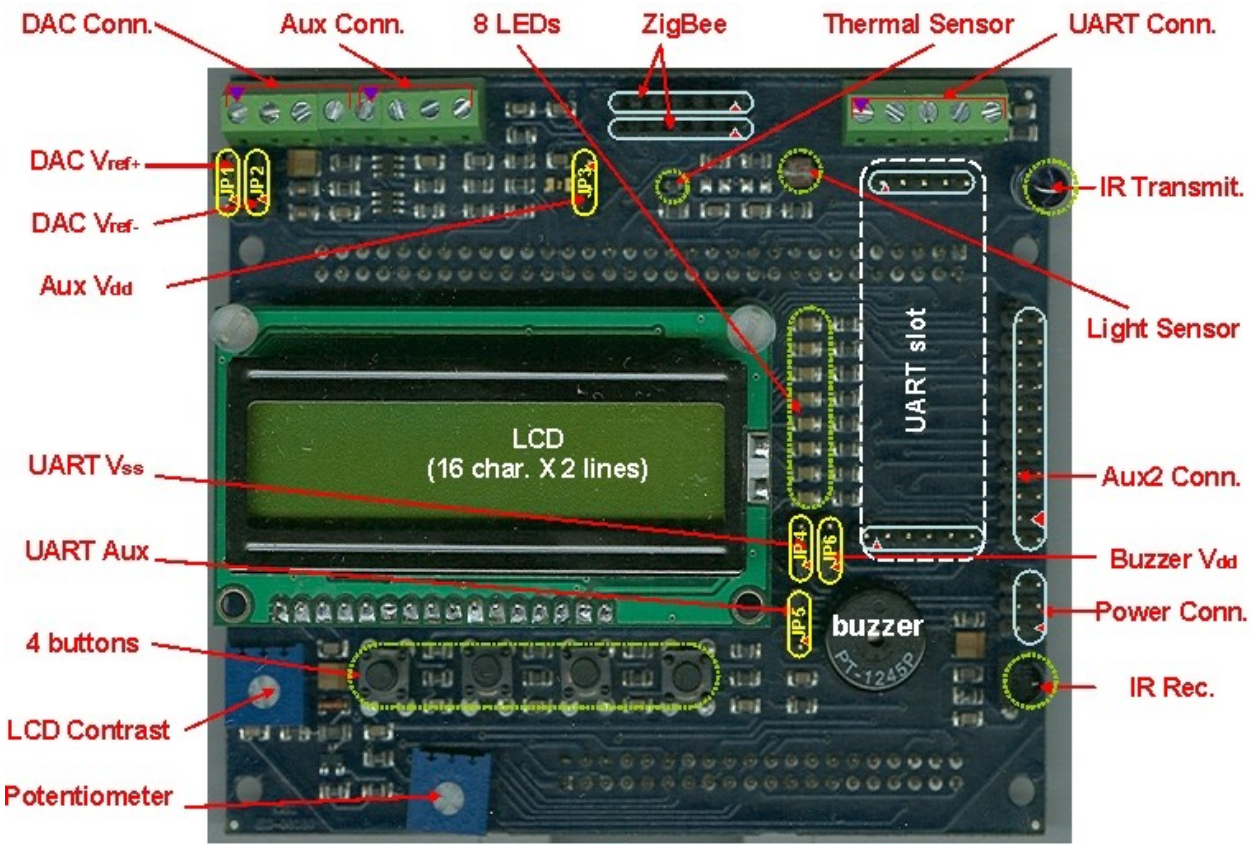


Figura 3.3: FLEX Demo Board

3.1.3. Microcontrollore dsPIC33FJ256MC710

Il microcontrollore della FLEX board è il dsPIC33FJ256MC710 della famiglia dspic DSC (Digital Signal Controller) di Microchip[18].

Esso si base su di un'architettura a 16 bit che integra alcune importanti funzionalità di un unità

DSP e che grazie a questa caratteristica è in grado di fornire prestazioni di tutto rispetto nell'elaborazione di segnali digitali. La presenza di un elevato numero di pin digitali (fino ad 85) e la presenza già a livello di silicio di numerose periferiche, fanno di questo prodotto un potente strumento nelle reti di controllo.

La Figura 3.4, estratta dal manuale del dispositivo, illustra le principali caratteristiche del dispositivo.

L'architettura dell'MCU è di tipo Harvard modificata, si ha in sostanza che il bus per le istruzioni sono a 24 bit mentre i bus dati sono a 16 bit; è da sottolineare l'utilizzo di due bus dati per accedere parallelamente a due zone di memoria. La memoria presenta la seguente struttura:

- Nei primi 2 KB vengono mappati in memoria gli Special Function Register (SFR) ossia i registri utilizzati dalla CPU e dalle periferiche. Dopo di essi si trovano 30 KB di SRAM che sono suddivisi in X Data RAM (accessibile dal relativo X Data Bus), Y Data RAM (accessibile dal relativo Y Data Bus), DMA RAM (2K di memoria dual-port dedicata al DMA). Tramite l'uso della Program Space Visibility (PSV) è possibile mappare una parte della memoria codice (di programma) nella memoria dati.
- La memoria per il codice è di tipo flash ed ha una dimensione di 256KB.

La CPU è composta da una componente MCU e da una DSP. La frequenza di clock del processore può essere variata tramite il settaggio di appositi divisori che agiscono su delle PLL, generando una velocità massima di 40 Mhz. Per tale frequenza il costruttore dichiara un valore di potenza computazionale pari a 40MIPS.

E' inoltre da sottolineare la presenza di un controllore DMA ad 8 canali.

Periferiche

Le periferiche principali presenti sul dsPIC33FJ256MC710 sono:

- 9 Timers a 16 bit (con prescaler programmabile), di cui 8 possono essere utilizzati in combinazione per dare origine a timers a 32 bit;
- 8 canali di Input Capture e 8 canali di Output Compare;
- 2 SPI, 2 I2C, 2 UART, 2 Enhanced CAN (ECAN);
- 8 canali Motor Control PWM;
- 1 modulo Quadrature Encoder Interface;
- 2 moduli ADC.

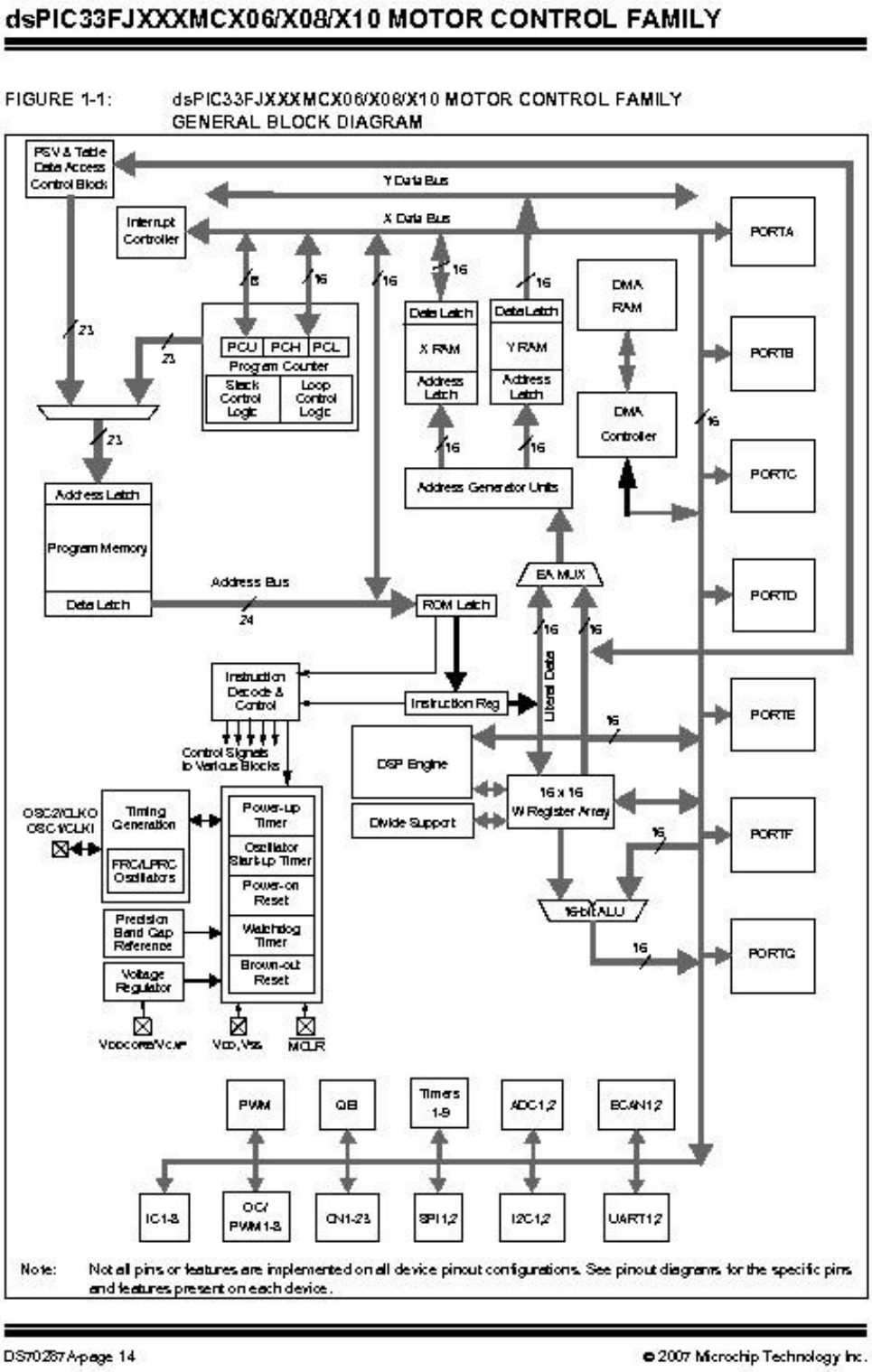


Figura 3.4: Schema generale del dsPIC®33 DSC

Supporto per Programmazione/Debugging

I microcontrollori dsPIC®33 integrano delle interfacce per la programmazione e la diagnostica del dispositivo stesso. In particolare sono messe a disposizione tre modalità:

- In-Circuit Serial Programming (ICSP)
- Enhanced ICSP;
- Joint Test Action Group (JTAG).

La prima è una tecnologia proprietaria di Microchip sviluppata per consentire la programmazione dei propri dispositivi. Per questo motivo essa è integrata direttamente nel core del microcontrollore, dove è implementata come macchina a stati che regola la scrittura in memoria. La scrittura della memoria richiede in ogni caso hardware e software esterni.

E' inoltre far notare come l'ICSP gestisca un canale per il debugging sul chip (in-circuit debugging), funzione sfruttabile solo se in possesso di un dispositivo esterno appropriato che trasferisca nel microcontrollore il SW specifico per il debugging. MPLAB ICD2 è un programmatore distribuito dalla Microchip che tramite il software MPLAB IDE (presente al momento solo per Windows) riesce a svolgere queste operazioni.

3.2. Ambiente Software

Nel lavoro di implementazione e debugging del sistema tre componenti software si sono dimostrati fondamentali, MPLAB C30, MPLAD IDE ed ERIKA. I primi due strumenti vengono forniti dalla MICROCHIP, azienda produttrice del microprocessore dsPIC33FJ256MC710, mentre l'ultimo è distribuito dalla Evidence S.R.L. Nel proseguo verrà fornita una descrizione delle caratteristiche principali di questi strumenti.

MPLAB IDE

Microchip distribuisce un ambiente integrato di sviluppo, MPLAB IDE, caratterizzato da un elevato numero di funzionalità. Vengono forniti una gestione dei progetti, una gestione dei files ed un loro editing ed infine una potente modalità di debugging. Risulta infatti possibile vedere sia il valore assunto dalle variabili (non a run-time) che vedere l'intero contenuto della memoria.

COMPILATORE C30

E' il compilatore utilizzato per sviluppare su processori Microchip della famiglia dsPIC®33. E' una versione del compilatore GCC della Free Software Foundation ottimizzata ed adattata all'architettura specifica di questi processori.

Alcune delle differenze sostanziali rispetto all'ANSI-C sono:

- specifica di attributi per le variabili;
- specifica di attributi per le funzioni;
- definizione di funzioni inline;
- associazione di variabili a specifici registri.

Sono inoltre presenti nella tool-chain di programmazione, anche un compilatore per assembler (MPLAB ASM30 Suite) ed linker specifico.

Il compilatore è distribuito anche in versione "student" liberamente scaricabile dal sito di Microchip[18]. Per maggiori informazioni consultare il manuale di riferimento.

3.2.1. ERIKA Enterprise

Embedded Real Time Kernel Architecture[21] è un sistema operativo real-time (RTOS) caratterizzato da un occupazione di memoria minimale, studiato appositamente per soluzioni embedded su piccoli microcontrollori. E' caratterizzato da una API aderente a quelle proposte dal consorzio OSEK/VDX[22]. E' distribuito da Evidence s.r.l.[19] in doppia licenza (commerciale e GPL). A corredo del kernel viene fornito RT-Druid, il quale fornisce un ambiente di sviluppo per il sistema operativo caratterizzato dalla presenza di strumenti per la scrittura, la compilazione e l'analisi dell'applicazione.

Per questo particolare progetto si è scelto di utilizzare ERIKA Enterprise (EE) con licenza GPL, di cui è doveroso fornire le principali caratteristiche:

- politiche di scheduling Fixed Priority (FP) ed Earliest Deadline;
- multitasting di tipo preemptive e non-preemptive;
- gestione della condivisione dello stack;
- gestione delle risorse condivise (Resource) con Immediate Priority Ceiling;
- attivazione periodica dei task mediante allarmi (Alarm);
- semafori generalizzati (Semaphore);
- quattro classi di conformità OSEK.

Architettura del Kernel

L'architettura di EE, mostrata in Figura 3.5, si compone di due livelli: il Kernel Layer e l'Hardware Abstraction Layer (HAL).

Il livello del Kernel implementa la gestione dei task e le strategia di schedulazione. Esporta per il livello di applicazione una serie di RTOS API per Tasks, Alarms, Resources e Semaphores.

L'HAL gestisce la parte del sistema operativo dipendente dall'hardware, ad esempio la gestione delle interruzioni e dei cambi di contesto. Al momento le architetture per cui è presente il supporto sono:

- Microchip dsPIC
- Atmel AVR5
- Altera NIOS II
- ARM7TDMI
- Tricore
- Motorola PPC
- Hitachi H8
- ST10

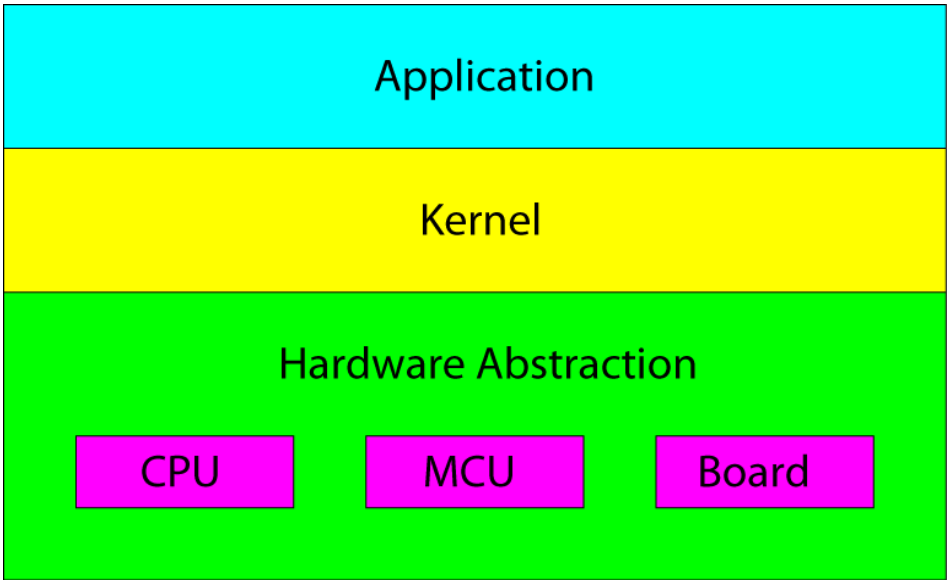


Figura 3.5: Archichettura di ERIKA RTOS

Standard OSEK

Come già introdotto in precedenza, ERIKA è coerente a quattro classi di conformità OSEK[23]. Per meglio comprendere questa importante proprietà è necessario definire brevemente lo standard OSEK.

Lo standard OSEK (Offene Systeme und deren schnittstellen fur die Elektronik im Kraftfahrzeug - open system and corresponding interfaces for automotive electronics) è un progetto nato da un consorzio di industrie automobilistiche con il fine di sviluppare sistemi di controllo distribuiti nei veicoli. Il consorzio prende il nome di OSEK/VDX (dove VDX sta per Vehicle Distributed eXecutive) e si occupa della definizione di un insieme di API per sistemi operativi real-time (OSEK) e di un sistema per la gestione di rete (VDX).

Le applicazioni tipiche dello standard riguardano sistemi di controllo caratterizzati da vincoli temporali molto stretti ed orientati a produzione di massa. Ne deriva che OSEK presenta ottimizzazioni nell'uso della memoria, delle risorse e sulle prestazioni finali. Per riassumere, in quattro punti, le caratteristiche principali dello standard, abbiamo:

- scalabilità (supporto per molte architetture HW);
- portabilità (interfaccia ISO/ANSI-C);
- configurabilità (OSEK Implementation Language – OIL);
- allocazione statica (elementi del kernel allocati a tempo di compilazione).

RT-Druid

RT-Druid rappresenta in sostanza l'ambiente di sviluppo per EE ed è implementato come un set di plugin per il framework Eclipse[24] (fig. 3.6). Esso si compone di una componente per la generazione del codice e di un analizzatore di schedulabilità.

Per quanto riguarda il code generator, ne esiste anche una versione stand-alone indipendente da Eclipse. Nello specifico RT-Druid è un ``compilatore" per OIL (dello standard OSEK) che, in base alle specifiche definite nella configurazione del sistema, genera la struttura necessaria alla compilazione di ERIKA (crea makefiles, inizializza le strutture dati, ecc). Sono inoltre fornite a corredo una serie di esempi di applicazioni (template project) pronte per essere compilate ed eseguite sulla FLEX board.

Il plugin per l'analisi della schedulabilità consente la modellizzazione, l'analisi e la simulazione del comportamento temporale del sistema embedded che si vuole progettare. Questo tool è disponibile solo nelle versioni commerciali del sistema operativo.

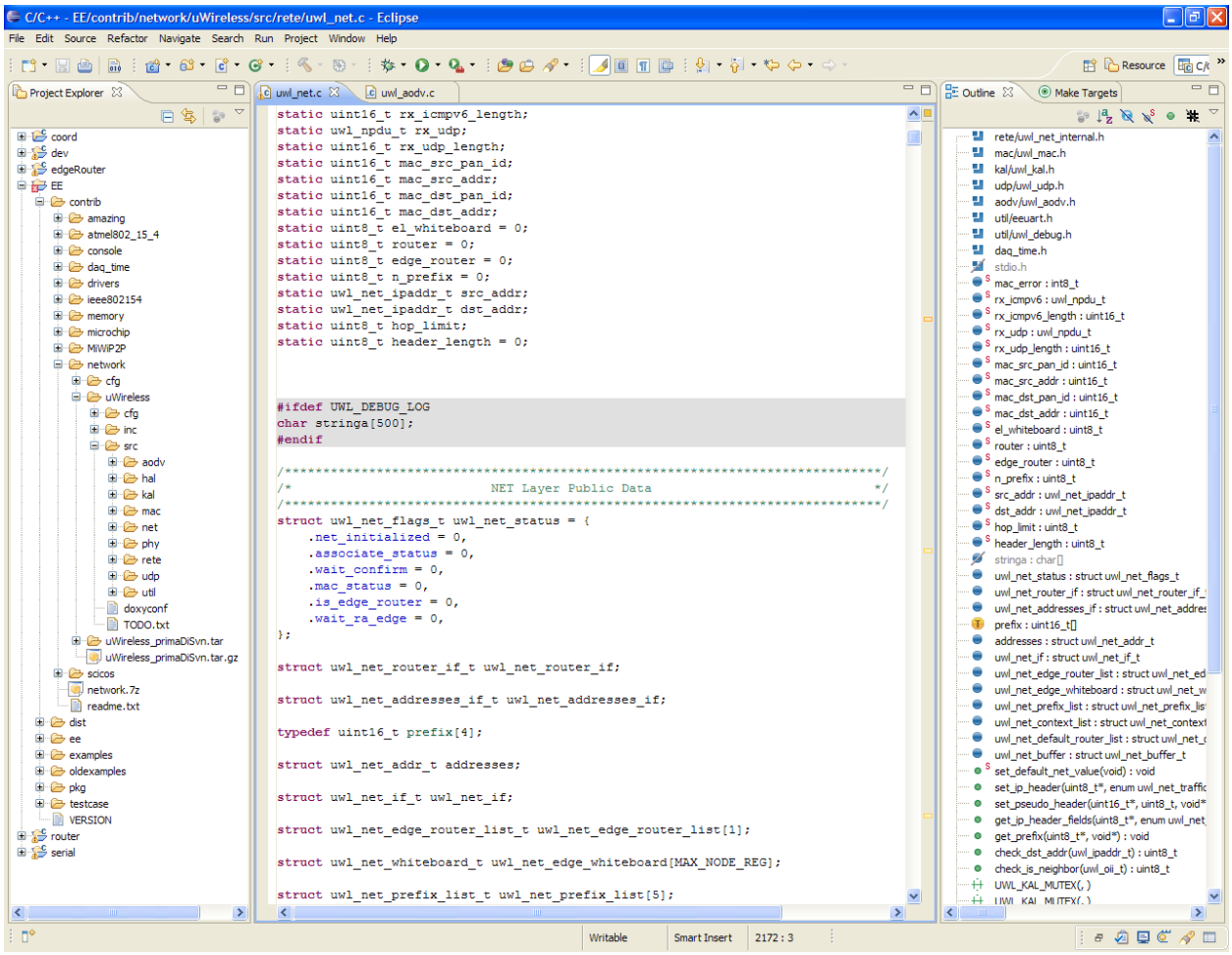


Figura 3.6: Screenshot del framework Eclipse con il plugin RT-Druid

3.2.2. µWireless

µWireless è un'implementazione dello stack 802.15.4 per Erika Enterprise sviluppato dal laboratorio Retis della Scuola Superiore Sant'Anna. Oltre ad avere un meccanismo per l'associazione dinamica dei dispositivi, include il supporto alla modalità beacon-enabled con GTS. Si riportano le funzionalità principali esportate al livello applicativo da µWireless:

- possibilità di configurare la durata del superframe e dei timeslot tramite i parametri BO e SO;
- invio in modalità CSMA/CA o GTS;
- possibilità di sincronizzare l'applicazione con gli eventi della rete (ricezione dei un frame, inizio e dine del superframe o del singolo time slot, et.) tramite la tecnica delle callback;
- possibilità di impostare il payload del beacon frame.

Le funzionalità dello stack sono implementate all'interno di appositi task che vengono attivati per mezzo di allarmi periodici. Come si può vedere in Figura 3.7 µWireless fa uso delle primitive software messe a disposizione da Erika.

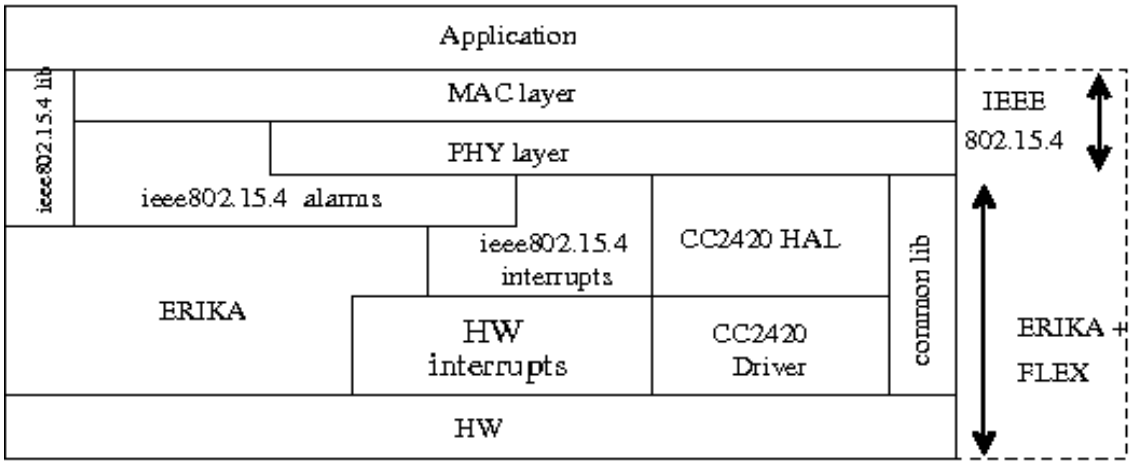


Figura 3.7: Archichettura stratificata del sistema di comunicazione µWireless compliant con lo standard IEEE 802.15.4 basato su Erika

3.3. Strumenti di debugging

Lo sviluppo del sistema è stato supportato da diversi strumenti di debugging. Due di questi, la Flex Demo Board e la coppia MPLAB IDE/MPLAB ICD2 sono stati già presentati. Nei paragrafi seguenti verranno presentati il CC2420 Packet Sniffer, utilizzato per l'analisi del traffico di rete, ed il DAQ.

3.3.1. Chipcon CC2420 Packet Sniffer

L'analizzatore di rete Chipcon CC2420 Packet Sniffer per IEEE 802.15.4 [25] permette di catturare i frame IEEE802.1.4 per mezzo della scheda Texas Instruments CC2420ED dotata dell'estensione CC2420EM (fig. 3.8).

Ifigura 3.8: La board Texas Instruments CC2420 EB con l'estensione CC2420EM

La sua interfaccia grafica (fig. 3.9) mette a disposizione le seguenti funzionalità:

- elenco dei pacchetti ricevuti;
- interpretazione dei pacchetti con evidenziazione dei differenti campi;
- filtraggio dei pacchetti;
- lista dei device presenti nella rete.

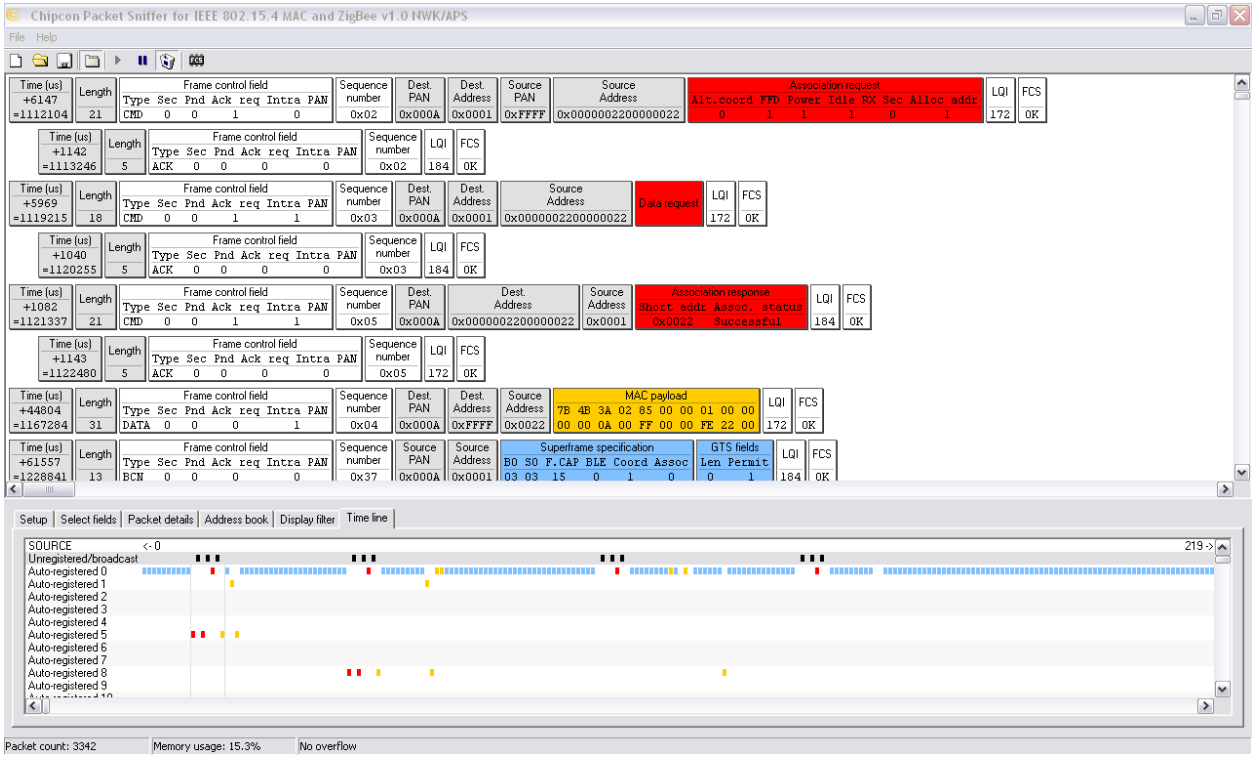


Figura 3.9: Interfaccia grafica del Chipcon CC2420 Packer Sniffer

3.3.2. DAQ

XML-based Data AcQuisition (DAQ), sviluppato dal gruppo di lavoro Wireless Sensor Network del Retis Lab della Scuola Superiore Sant'Anna, è un sistema di acquisizione dati che permette il salvataggio dei valori generati da un dispositivo embedded. Questa operazione è di fondamentale importanza nello sviluppo di applicazioni embedded complesso in quanto si ha la possibilità di valutare le prestazioni del firmware realizzato. L'idea innovativa alla base di questo tool è quella di essere indipendente dal firmware che si intende testare, infatti il DAQ viene inizializzato con un file XML contenente la sorgente dei dati (file, porta seriale, USB, ecc.) ed il loro formato.

Come mostrato in figura + il file XML viene processato dal metodo XMLParser della classe CXML per ottenere le informazioni relative alla formattazione dei dati in input con cui inizializzare le strutture SDef e Metadata. La classe CSerial contiene un buffer a basso livello per ricevere i dati in ingresso dal dispositivo embedded e inviarli alla classe CFormatter che gestirà la sua serializzazione tramite le classi del software ROOT.

ROOT è un framework standard “de facto” per il data processing in High Energy Physics, tramite la sua classe TTree vengono istanziate delle n-tuple in cui ciascuno dei record può avere una struttura più complessa (una classe). Questi TTree possono essere serializzati e salvati su disco su contenitori simili a quelli di un filesystem (TFile). I TFile possono essere, in fase di post processing, analizzati attraverso le librerie di ROOT utilizzabili su piattaforme Linux e DOS in quanto viene creato all'interno di un TFile un file system indipendente dalla piattaforma.

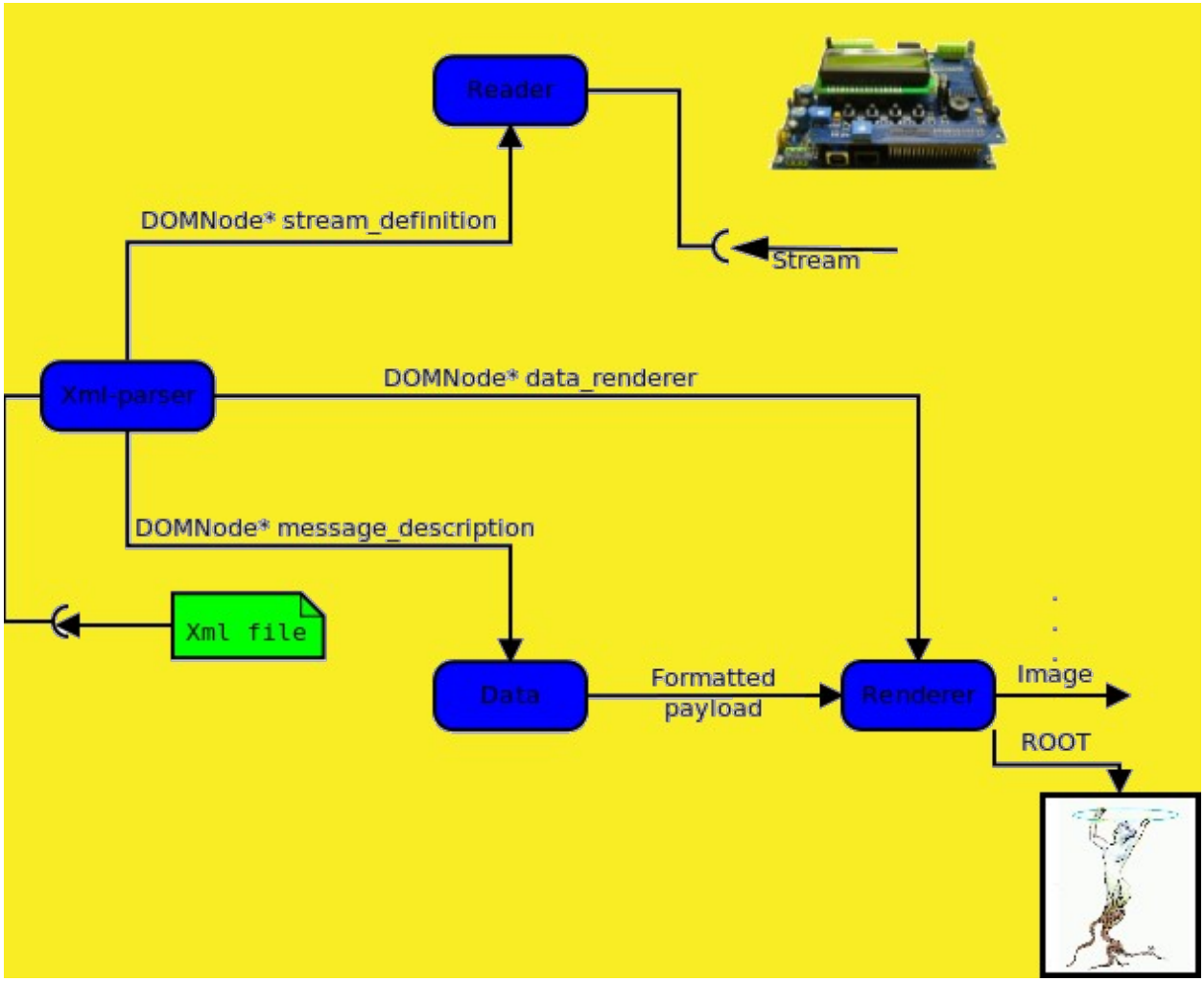


Figura 3.10: DAQ

Capitolo 4

4. Progettazione del Sistema

In questo capitolo inizialmente verranno descritte le varie fasi della progettazione del sistema e le scelte effettuate. In seguito si descriverà come l'applicativo implementa le funzionalità richieste.

4.1. Definizione del problema

La progettazione del sistema è fortemente influenzata dai requisiti non funzionali, in particolar modo dal requisito di scalabilità. Come visto (cfr. +) l'implementazione del sistema deve essere effettuata su dispositivi hardware con ridotte capacità, sia di memoria che di calcolo computazionale. Questi vincoli, però, non devono deteriorare le prestazioni generali della rete, come i tempi di risposta, ivi inclusi nella comunicazione P2P. Per prima cosa si sono studiati gli standard prodotti relativi a questa materia per capirne se era possibile la loro implementazione. Mentre per l'addressing ed il discovery l'adozione degli RFC del working group “ Ipv6 over Low power WPAN” è stata dettata dal fatto che al momento dell'inizio del lavoro fosse lo standard più completo che rispettasse tutti i requisiti del sistema, per quanto riguarda il routing c'erano più opzioni. Si è deciso di adottare l'algoritmo AODV, oltre al fatto che rende la rete completamente connessa e scalabile, per la sua semplicità di implementazione.

Il secondo passo è stato quello di studiare il funzionamento della libreria µWireless, che

implementa lo standard IEEE 802.15.4. Questa fase di studio si è incentrata soprattutto sulla comunicazione tra il livello MAC ed il livello superiore. Per comunicazione si intendono i metodi implementati, con i relativi parametri, per lo scambio di informazioni tra i due livelli.

Una volta acquisite queste conoscenze si è potuto definire l'architettura del sistema, rispettando il requisito di estensibilità. Infine sono state individuate le varie componenti software da implementare, nonchè la loro collocazione all'interno dell'architettura.

4.2. Archittettura del sistema

L'architettura della libreria µWireless, che contiene il livello fisico ed il livello MAC, rispetta il requisito di estensibilità, per la sua architettura strutturata a blocchi, separati ed indipendenti. Si è continuato ad adottare questa architettura, arricchendola dei moduli necessari per l'implementazione del sistema. Il livello MAC comunica con il livello superiore attraverso delle funzioni definite dallo standard IEEE 802.15.4.

I moduli implementati sono:

- network, che implementa l'addressing ed il discovery;
- AODV, che implementa il protocollo di routing;
- UDP, semplice livello “User Datagram Protocoll” (RFC 768).

Si è deciso di implementare il protocollo di routing in un modulo a se stante per motivi di convenienza, avendo così la possibilità di integrare il sistema in futuro con algoritmi alternativi.

La modularità è anche caratteristica delle funzioni implementati nei vari moduli. Ad esempio all'interno dello stato network, anche la formazione del datagram è modulare. In futuro si potranno aggiungere nuove funzionalità senza reingegnerizzare.

La Figura 4.1 illustra l'archichettura della libreria µWireless, il modulo AODV è stato posizionato tra il livello NET ed il livello UDP in quanto, come protocollo di routing, sfrutta il livello network ed utilizza il secondo livello per inviare i messaggi di RREQ e RREP.

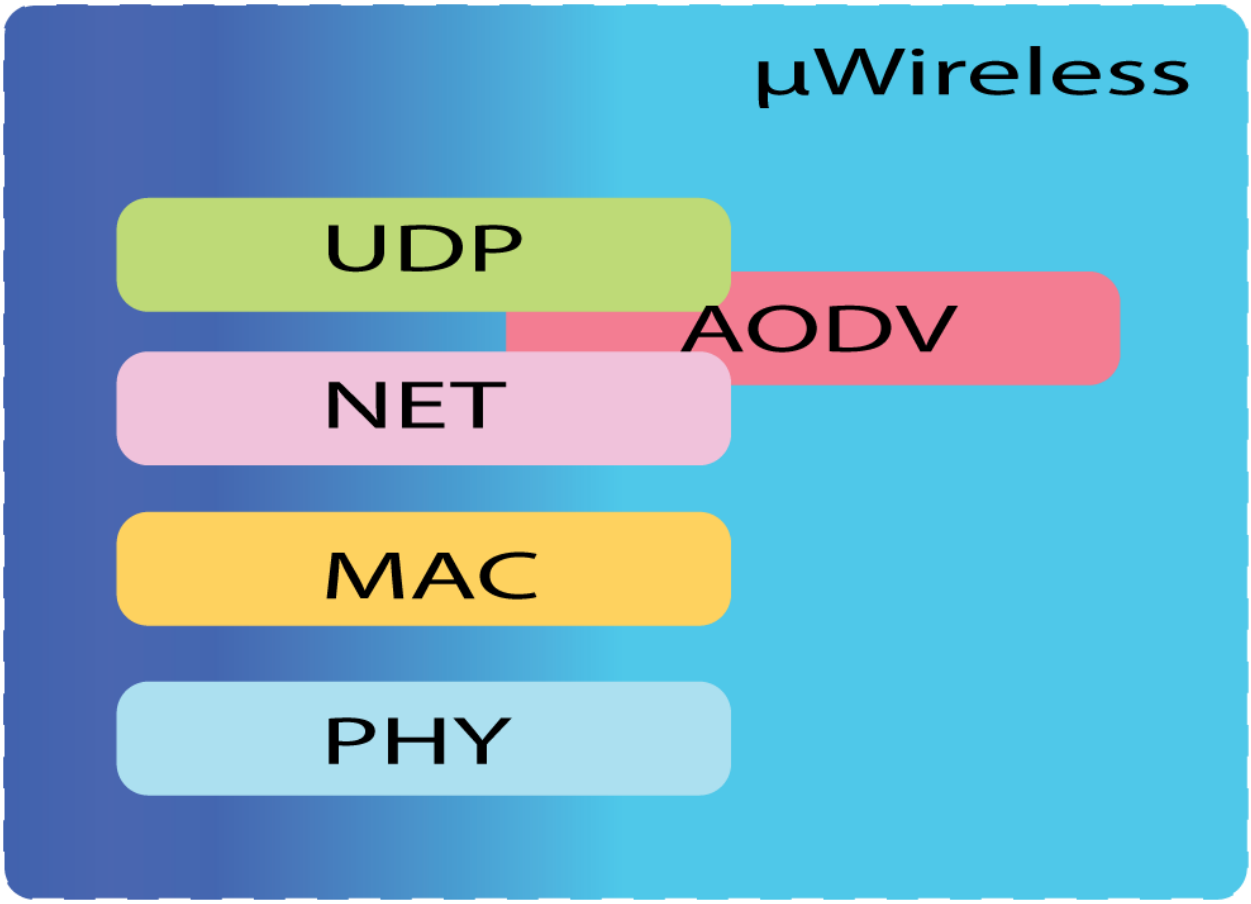


Figura 4.1: Rappresentazione dello stack µWireless

4.3. Topologia

Il draft dell'IETF "6LoWPAN Neighbor Discovery" definisce diversi ruoli per i dispositivi facenti parte di una rete 6LoWPAN:

- LoWPAN Host, nodo che può essere solo sorgente o destinatario di un pacchetto;
- LoWPAN Router, nodo avente il compito di inoltrare i pacchetti tra sorgente e destinatario;
- LoWPAN Edge Router, router avente il compito di interconnettere la rete LoWPAN ad un'altra rete IP. Tiene traccia dei nodi registrati in una tabella denominata Whiteboard.

I primi due vengono chiamati più genericamente LoWPAN Node. Inoltre vengono definite la Simple LoWPAN e la Extended LoWPAN, due tipologie di rete che differiscono per il numero di Edge Router presenti. La prima è costituita da un unico Edge Router e da un insieme di nodi (fig. 4.2), mentre la seconda è composta da più Simple LoWPAN come in Figura 4.3.

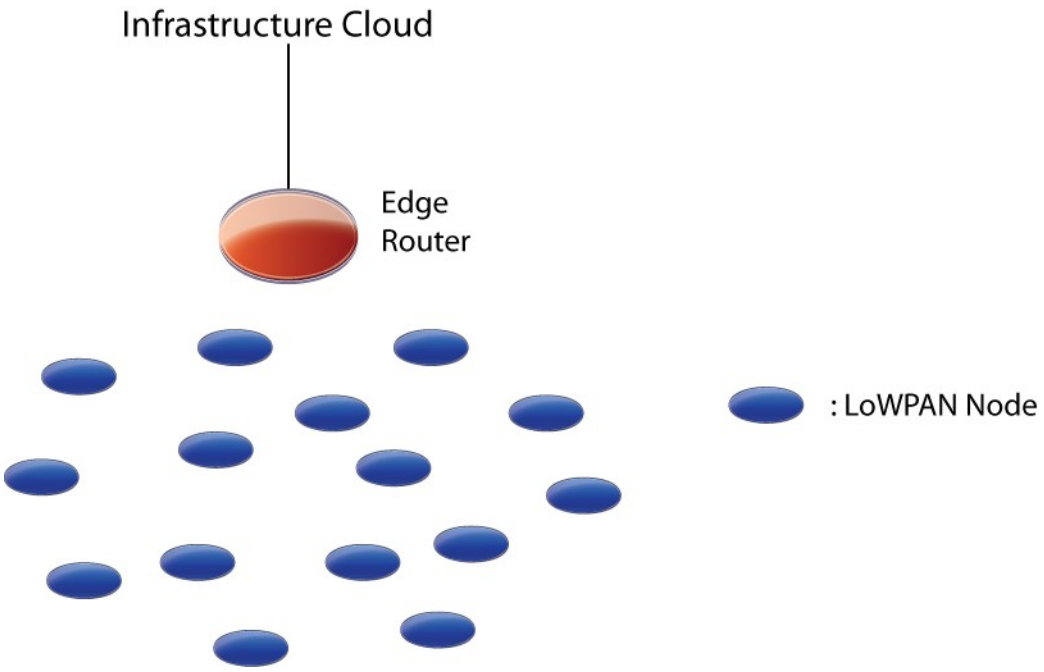


Figura 4.2: Simple LoWPaN

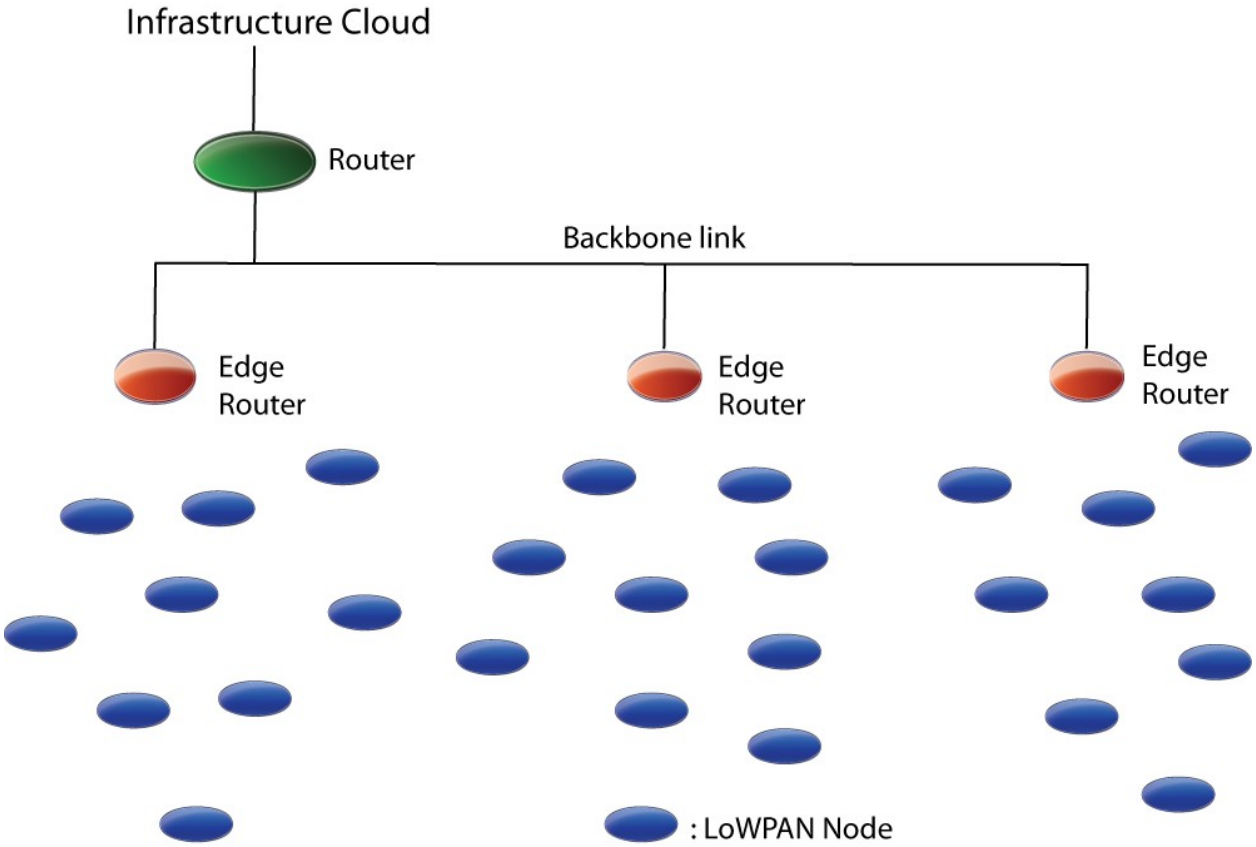


Figura 4.3: Extended LoWPAN

L'implementazione del sistema si è basata sulla prima topologia, con la presenza di più nodi, di tipo LoWPAN Router, e da un Edge Router.

4.4. Addressing

Per compredere meglio la funzionalità di addressing, si farà una breve introduzione sugli indirizzi IPv6 e la loro rappresentazione.

L'indirizzo IPv6 è composto da due parti: suffisso e prefisso (fig. 4.4), ciascuna di 64 bit. Ci sono due possibili forme convenzionali per rappresentarlo:

- la forma preferita è x:x:x:x:x:x:x:x, dove le x sono un numero esadecimale composto da 4 cifre. ABCD:EF01:2345:6789:ABCD:EF01:2345:6789 e 2001:DB8:0:0:8:800:200C:417A sono esempi di questa notazione;
- la seconda forma introduce una compressione degli zeri contigui. 2001:DB8:0:0:8:800:200C:417A si può scrivere 2001:DB8::8:800:200C:417A, FF01:0:0:0:0:0:0:101 si può scrivere FF01::101.

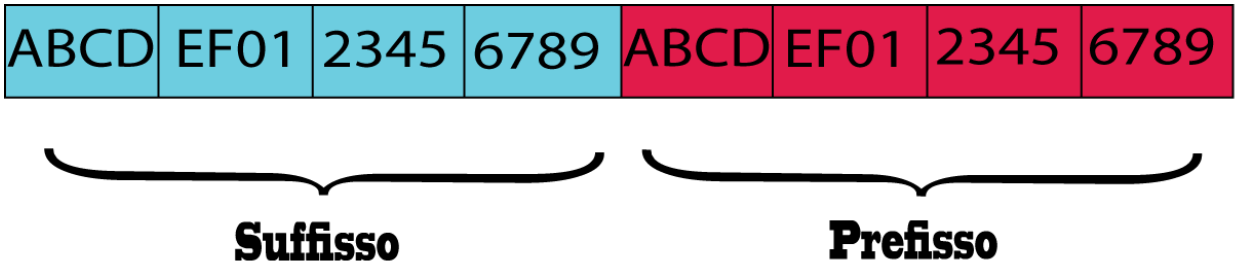


Figura 4.4: Indirizzo IPv6

Per quanto riguarda il prefisso la notazione è la seguente:

indirizzo IPv6/lunghezza del prefisso

dove:

- la prima parte è un indirizzo IPv6 in una notazione sopra citate;
- la seconda parte è un valore decimale e specifica i bit contigui più a sinistra rappresentano il prefisso.

Esempi sono:

- 2001:0DB8:0000:CD30:0000:0000:0000:0000/60;
- 2001:0DB8:0000:CD30:0:0:0:0/60;
- 2001:0DB8::CD30/60.

In base al prefisso si possono identificare diversi tipi di indirizzo:

- Unspecified → ::/128, utilizzato in casi speciali, come ad esempio in fase di registrazione;
- Multicast → FF00::/8, comunicazione a tutti i nodi;
- Link-Local unicast → FE80::/10, utilizzato per comunicare all'interno di una stessa rete;
- Global Unicast, identifica univocamente il nodo.

E' definita subnet tutti i nodi con lo stesso prefisso. Inoltre, in IPv6 viene introdotto il concetto di "interface". Un dispositivo si collega alla rete attraverso la sua inerfaccia, composta dagli indirizzi sopra citati. Si possono avere più interfacce per ogni nodo, una per ogni rete a cui si collega. Ogni interfaccia si differenzia dall'altra in base agli indirizzi Global Unicast.

Le possibilità per ottenere un "interface identifier" sono diverse. Lo standard IPv6 ne definisce tre: attraverso l'uso di un DHCP, manualmente oppure eseguendo la procedura "stateless address autoconfiguration", definita nell'RFC "IPv6 Stateless Address Autoconfiguration" (RFC 4862). In questa tesi si è implementato il terzo modo.

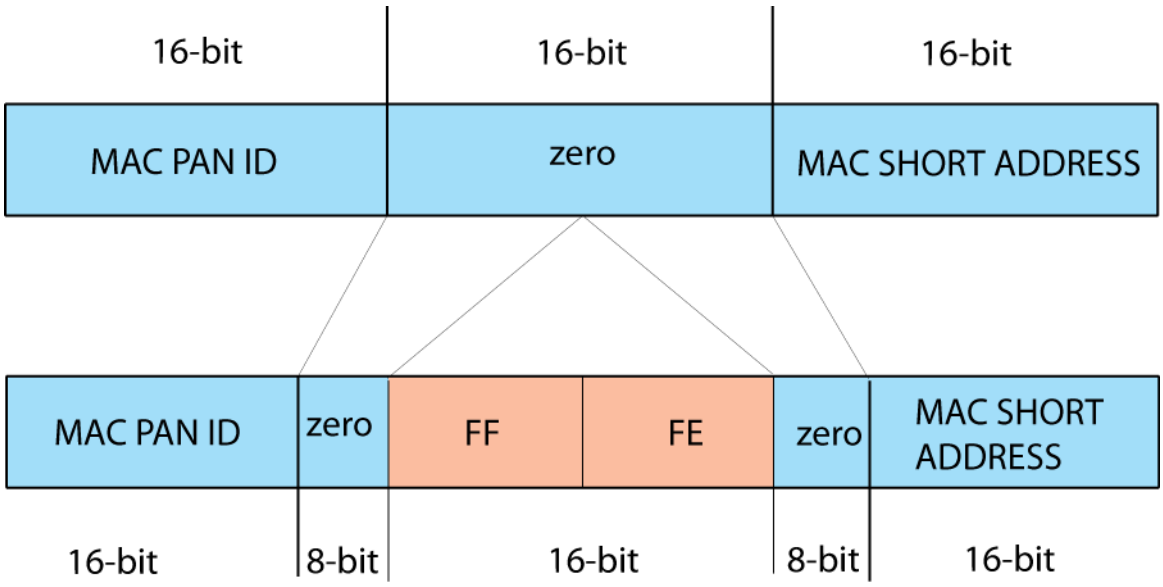


Figura 4.5: Stateless Address Autoconfiguration

Seguendo questo standard, l'ID può essere ottenuto attraverso due modi: o utilizzando il MAC extended address, di 64 bit, oppure utilizzando il MAC short address, di 16 bit, insieme al MAC PAN ID, di 16 bit. Nel secondo caso è necessario implementare delle operazioni che permettano di arrivare ad un suffiso di 64 bit . Il sistema implementerà il secondo modo. Un nodo, all'accensione, dopo aver svolto le operazioni relative all'associazione dinamica e quindi dopo

aver ottenuto un indirizzo MAC short e conoscendo già il MAC PAN ID, creerà il suo “Interface Idenfier. Il primo passo è quello di creare uno pseudo indirizzo di 48 bit attraverso la concatenazione dell'indirizzo della PAN, indirizzo short del device e 16-bit di valore zero. Una volta creato questo indirizzo di 48-bit si aggiungono i valori “FF:FE” per ottenere un indirizzo a 64-bit. L'intero procedimento è illustato in Figura 4.5, un esempio di un suffisso di 64-bit è illustarato in Figura 4.6.

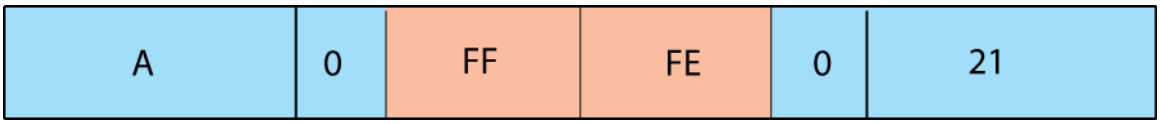


Figura 4.6: Esempio suffisso dell'indirizzo IPv6 ottenuto con la procedura Stateless Adress Autoconfiguration

Completati questi primi passi, il nodo ha associato un suffisso e potrà quindi formare il suo indirizzo link-local, composto dal prefisso FE80::/10 e dal suffisso.

Per quanto riguarda gli indirizzi multicast, lo standard ne definisce alcuni oltre alla possibilità di crearne riferiti ad un gruppo di nodi, si è deciso di implementarne uno solo: FF00::2. Questo indirizzo si riferisce a tutti i nodi facenti parte della subnet.

L'ultimo passo da completare per quanto riguarda l'addressing è la creazione di un indirizzo Global Unicast (per accedere al nodo univocamente da Internet). Per far ciò, il nodo si mette in ascolto dei messaggi inviati dagli altri router presenti nella rete. Ricevuto e processato il messaggio, il dispositivo viene a conoscenza del prefisso della subnet di cui fa parte. A questo punto il nodo è in grado di creare il suo indirizzo globale composto dal prefisso della rete e dal suffisso creato in precedenza.

Infine il nodo avvia la procedura di registrazione con l'Edge Router.

L'Edge Router definisce il suo indirizzo link-local esattamente allo stesso modo come un qualsiasi altro nodo della LoWPAN.

4.5. Neighbor discovery

Il neighbor discovery consiste in due fasi:

- registrazione dell'indirizzo del nodo sulla tabella dell'Edge Router;
- “scoperta dei vicini”.

Lo standard definisce quattro messaggi di tipo ICMPv6 per portare a termine le fasi appena citate:

- Router Solicitation – RS, con lo scopo di sollecitare un Router Advertisement;
- Router Advertisment – RA, con lo scopo di pubblicizzare i parametri della rete e la presenza del router;
- Node Registration – NR, messaggio diretto all'Edge Router per richiedere la registrazione;
- Node Confirmation – NC, messaggio di conferma avvenuta registrazione.

Vengono definite anche delle opzioni per i singoli messaggi.

Un nodo per prima cosa esegue la procedura "stateless address autoconfiguration" creando il suo link-local unicast (cfr. +). Per connettersi alla rete LoWPAN, il dispositivo, trasmette un messaggio di tipo RS e attende in risposta un messaggio di tipo RA dall'Edge Router o da un LoWPAN Router (fig. 4.7).

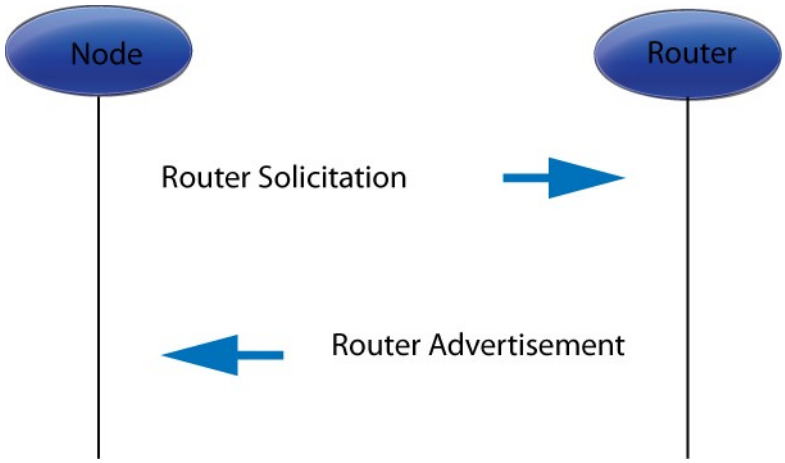


Figura 4.7: Sequenza dei messaggi Router Solicitation e Router Advertisement

La ricezione dei messaggi Router Advertisement da parte dei nodi vicini ha un duplice scopo: il primo per completare la procedura di autoconfigurazione degli indirizzi, mentre il secondo è quello di venire a conoscenza dei nodi vicini con cui può scambiarsi messaggi attraverso un singolo hop. Per quanto riguarda il primo scopo, se nel messaggio è presente il prefisso valido della subnet a cui si sta connettendo, il nodo può formare il suo Global Unicast completando la procedura "stateless address autoconfiguration".

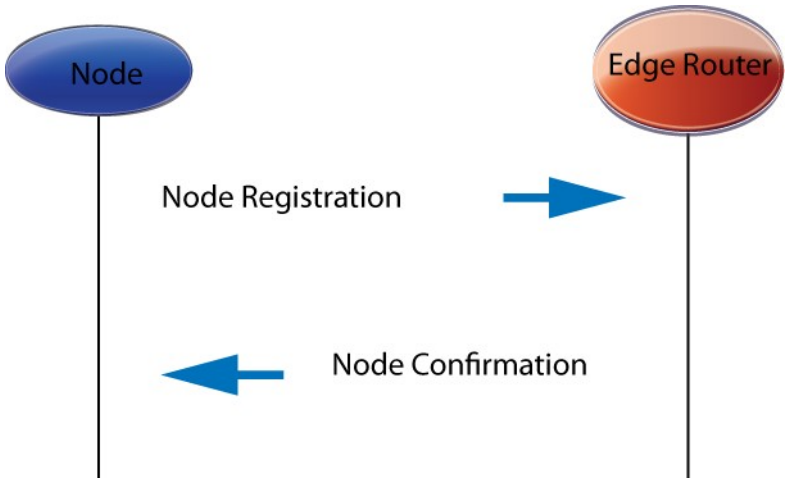


Figura 4.8: Sequenza messaggi Node Registration e Node Confirmation

In seguito il nodo tenterà di eseguire la registrazione con l'Edge Router della rete. La registrazione avviene attraverso l'invio di un messaggio NR. Quest'ultimo conterrà l'indirizzo con il quale si vuole registrare nella Whiteboard. L'Edge Router, dopo aver processato il messaggio NR ed aver rilevato che non ci siano indirizzi duplicati, risponde con un NC. In Figura 4.8 viene mostrato questo scambio di messaggi.

A questo punto il dispositivo è in grado di utilizzare pienamente la LoWPAN, ad esempio può inviare e ricevere pacchetti. Per pubblicizzare la prorpria presenza all'interno della rete, il dispositivo invierà periodicamente dei Router Advertisement.

4.6. Routing

Nell'implementazione dell'algoritmo di routing si deve tener conto che il sistema deve essere completamento connesso, requisito funzionale. Considerato il grande numero di sensori che possono far parte di una WSN, è impensabile, come già detto, poter ottenere questo requisito collegando direttamente tutti i sensori della rete. Per quanto detto la comunicazione tra i dispositivi interni alla rete deve avvenire tramite tecniche di multi-hop. Adottando questa tecnica si hanno anche altri vantaggi quali un risparmio di energia ed una suddivisione del traffico, evitando così un concentramento del traffico in un solo nodo.

Ricordando anche che il sistema deve essere scalabile, si deve minimizzare l'utilizzo della memoria, il traffico dei pacchetti in rete e l'overhead di ogni singolo pacchetto. Il primo obiettivo viene raggiunto grazie all'utilizzo dell'algoritmo AODV. Questo protocollo ha come caratteristica il fatto che si memorizzano solo i percorsi necessari e tutti i nodi del percorso

partecipano all'instradamento del pacchetto. Questo avviene in quanto non viene memorizzato tutti il percorso tra sorgente e destinatario, ma bensì solo l'indirizzo del destinatario ed l'indirizzo del prossimo nodo per raggiungere il destinatario (next-hop). Gli ultimi due obbiettivi sono raggiunti grazie all'utilizzo del TinyAODV con le specifiche già citate (cfr. +).

Nel momento in cui un nodo deve comunicare con un altro, in base all'indirizzo IPv6 del destinatario, può effettuare diverse scelte:

- se l'indirizzo non è un link-local o il prefisso non fa parte della stessa subnet della sorgente, si invia il pacchetto all'Edge Router, il quale penserà ad instradarlo al di fuori della rete;
- se il prefisso è interno alla rete, si possono avere due opzioni:
  - il nodo destinatario è un “neighbor”, e quindi raggiungibile con un singolo hop;
  - il nodo destinatario si trova all'interno della rete, ma non si conosce un percorso per raggiungerlo. In questo caso si chiama l'algoritmo di routing AODV per poter creare un percorso tra sorgente e destinatario.

AODV ha il compito di formare un percorso di instradamento tramite i tipi di messaggi definiti (cfr. +).

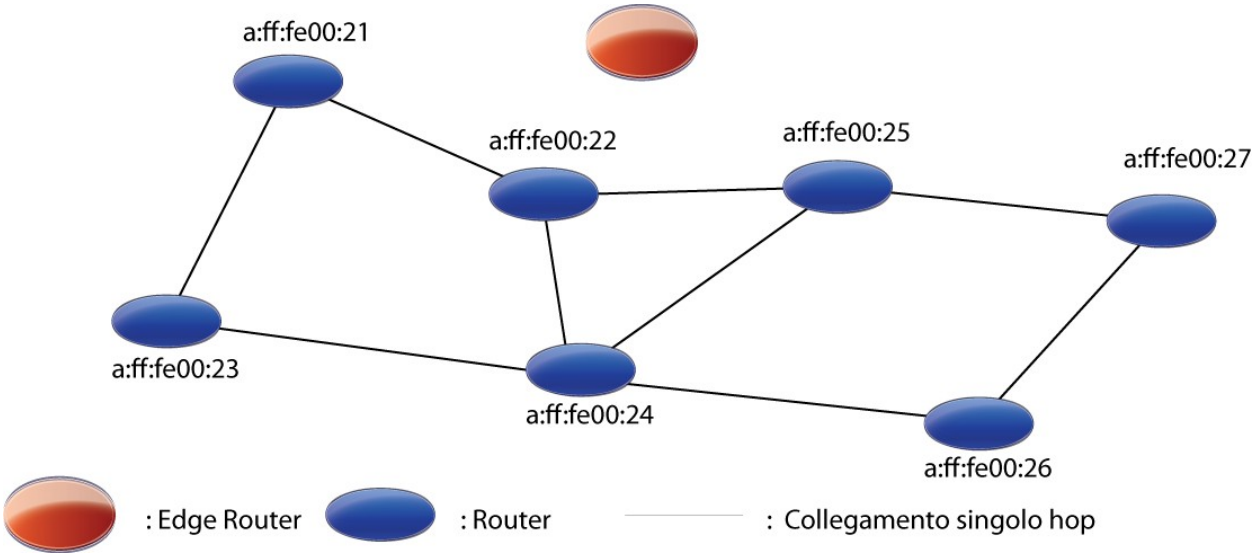


Tabella dei Neighbor
a:ff:fe00:21
a:ff:fe00:24
a:ff:fe00:25

Tabella di Routing	
Destinatario	**Next-Hop**
a:ff:fe00:26	a:ff:fe00:24

Tabella 4.1: Esempio Tabella dei Neighbor e Tabella di Routing relativa al nodo a:ff:fe00.22

Inizialmente il dispositivo sorgente che non conosce un percorso per raggiungere la destinazione, invia dei messaggi RREQs in multicast e si mette in attesa di un messaggio RREP. Questo messaggio, che può essere inviato solo dalla destinazione, viene processato ed inoltrato dai nodi che comporranno il percorso di instradamento. Con processare si intende l'aggiornamento della propria tabella di routing.

Una volta ricevuto il messaggio di RREP il dispositivo sorgente può inviare il messaggio al destinatario.

4.7. Strutture dati

Lo standard definisce delle strutture dati per permettere l'implementazione delle varie operazioni. Nell'implementarle si è tenuto conto del fatto che il sistema deve essere scalabile, più precisamente si è data molta attenzione all'utilizzo della memoria. Ad esmpio nel memorizzare gli indirizzi IPv6 dei nodi vicini, o dell'Edge Router, si è creato un campo di 64-bit anziché di 128-bit. Questo è stato possibile dal presupposto che i nodi facenti parte della stessa subnet possiedono lo stesso prefisso. Gli standard adottati non definiscono vincoli sul come implementare le strutture dati.

Nel seguito verranno descritte le principali strutture dati implementate, suddivise per tipo di dispositivo.

4.7.1. Strutture dati del LoWPAN Node

Innanzitutto un qualsiasi nodo deve implementare delle strutture dati che contengano le proprie informazioni. Una di queste è l'interfaccia, necessaria per il collegamento alla rete. Questa struttura è composta da:

- prefisso della subnet;
- indirrizzo Link-Local;
- indirizzo Global Unicast;
- indirizzo Multicast.

È necessario inoltre memorizzare alcuni parametri relativi al nodo ed i parametri relativi alla rete a cui si sta collegando. Le principali informazioni da memorizzare sono:

- flag indicante se il nodo è di tipo router;
- flag indicante se il nodo può inviare periodicamente messaggi RA;
- intervallo di tempo in cui inviare i messaggi RA;
- MTU utilizzato dalla rete;
- Hop Limit corrente utilizzato dalla rete;
- una stima del tempo di vita del dispositivo.

Alcuni di questi parametri verranno trasmessi nei messaggi Router Advertisement per pubblicizzarli all'interno della rete con lo scopo, ad esempio, di farli conoscere a nuovi dispositivi.

Il draft "6LoWPAN Neighbor Discovery", inoltre, definisce alcune strutture dati per la memorizzazione dei nodi vicini e dell'Egde Router. Per i primi si utilizza una struttura dati che contiene l'"interface identifier" dei dispositivi raggiungibili con un singolo hop con relativo tempo di vita. L'Edge Router viene memorizzato in un'altra struttura, che oltre a contenere l'ID ed il relativo tempo di vita, come quella per i nodi vicini, mantiene anche un campo indicante quando è avvenuta la registrazione sulla Whiteboard.

Infine anche lo standard relativo all'algoritmo di routing definisce una struttura dati, ovvero la tabella di routing. Come già accennato le entrate di questa tabella avranno un campo indicante l'ID del destinatario, l'ID del "next-hop", il numero di hop per raggiungere la meta, utilizzato per la metrica ed un numero di sequenza indicante l'età del percorso. Sempre rispettando il requisito non funzionale di scalabilità e più precisamente il minimizzare l'uso della memoria, nella tabella di routing non sono presenti i nodi vicini, evitando così la ridondanza dei dati.

4.7.2. Strutture dati del LoWPAN Edge Router

Oltre ad implementare le stesse strutture dati che caratterizzano un qualsiasi nodo della 6LoWPAN, deve mantenere una tabella chiamata Whiteboard. Lo scopo di questa struttura è quello di tener memorizzati i nodi facenti parte della subnet, e quindi gli "interface identifier" dei dispositivi registrati. Oltre a questo campo, la tabella conterrà quando è avvenuta la registrazione ed il tempo di vita del nodo.

Capitolo 5

5. Implementazione

In questo capitolo verranno discussi gli aspetti dell'implementazione del sistema, come ad esempio i linguaggi e le tecniche di programmazione impiegati, le librerie utilizzate, i vincoli di cui tener conto, ecc.

5.1. Vincoli di implementazione

Nella scrittura del codice si sono dovuti affrontare i problemi tipici dell'implementazione di software per sistemi embedded, e rispettarne i vincoli. Non è stato possibile scegliere liberamente il linguaggio di programmazione, ma si è dovuto utilizzare il linguaggio C. Inoltre le capacità computazionali e di memoria dei sistemi embedded hanno richiesto una buona ottimizzazione del codice e l'uso di appositi accorgimenti.

A questi vincoli si sono aggiunti quelli dei requisiti non funzionali, come estensibilità ed indipendenza dell'hardware, non facilmente soddisfabili in sistemi embedded. L'utilizzo di un microkernel come ERIKA ha permesso di raggiungere questi obiettivi. Ad esempio ERIKA permette di astrarre l'hardware sottostante e quindi consente di ottenere l'indipendenza da esso.

Nel seguito si vedrà in dettaglio come sono stati implementati i vari moduli.

5.2. Il Tree di Erika

Il Tree di ERIKA è composto da due directory principali: *ee* e *contrib*. La prima contiene il codice relativo al kernel ed i file che permettono l'astrazione dall'archichettura del singolo microcontrollore (AVR5, PIC30, ARM7TDI, NIOS2, etc.). La seconda directory contiene i driver e le librerie di sistema, tra cui quelle relative al networking come µWireless.

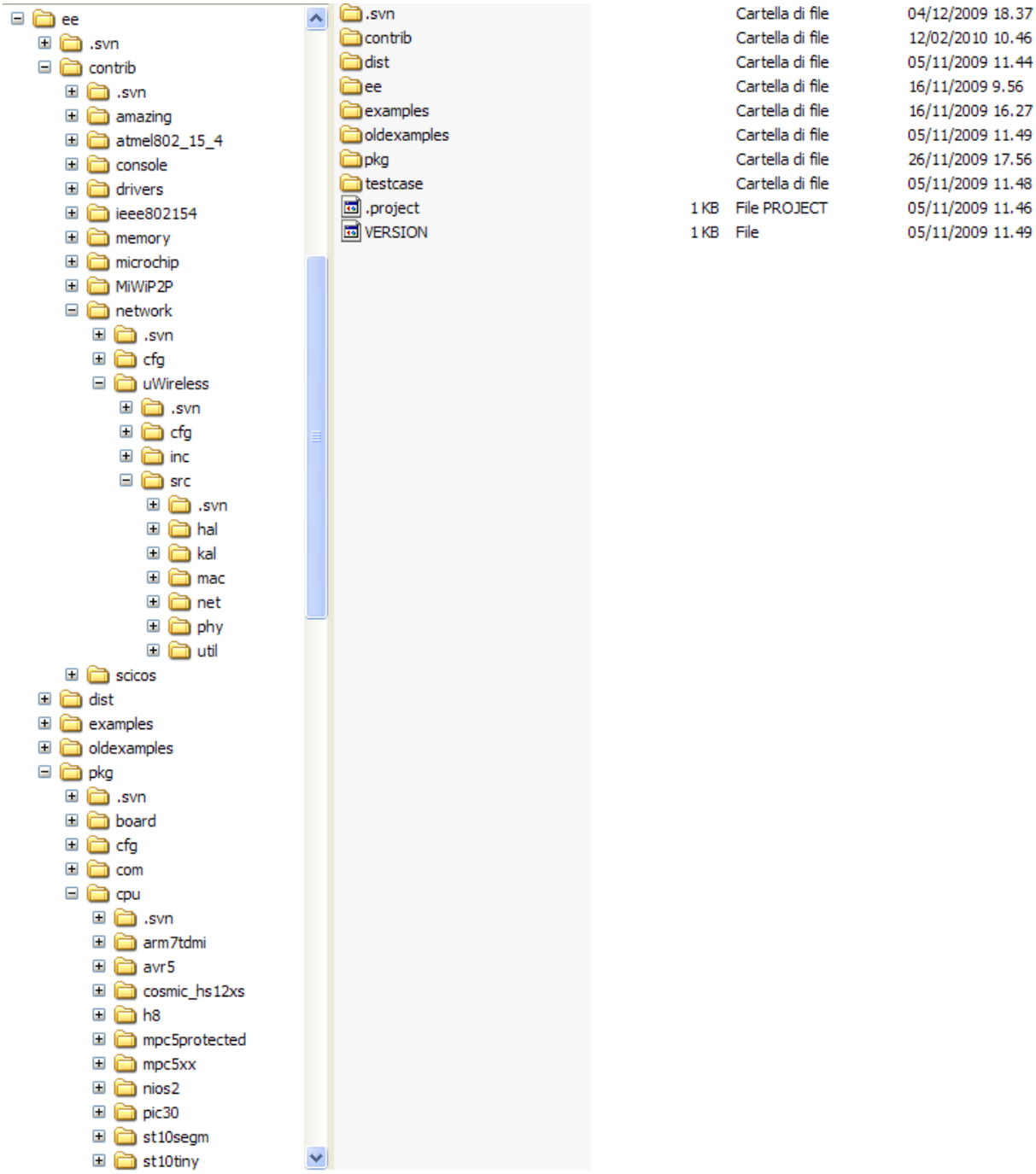


Figura 5.1: Tree di ERIKA

Questa libreria contiene i seguenti moduli:

- hardware abstraction layer, contenuto nella directory hal e che permette l'astrazione dall'hardware;
- kernel abstraction layer, contenuto nella directory kal e che permette l'astrazione dal kernel;
- physic layer, contenuto nella directory phy e che contiene l'implementazione del livello fisico dello standard IEEE 802.15.4;
- MAC layer, contenuto nella directory mac e che contiene l'implementazione del livello MAC dello standard IEEE 802.15.4,
- utility, contenuto nella directory util che contiene degli strumenti per il debugging in fase di progettazione;
- network layer, contenuto nella directory net che contiene una prima e semplice implementazione del livello network.

I file ed i moduli implementati in questa tesi sono stati collocati all'interno della libreria µWireless, con l'aggiunta dei moduli implementati. In Figura 5.1 è illustrato il tree di ERIKA.

5.3. Implementazione dei moduli

Ogni modulo è composto da due tipi di file:

- .c, ovvero file sorgenti delle librerie;
- .h, ovvero file da includere nei primi. Qui vengono definite tutte le strutture dati con visibilità globale e le macro.

La collocazione di questi due tipi di file è illustrato in Figura 5.2.

I file sorgenti hanno al loro interno funzioni che possono essere chiamate al di fuori del modulo in cui sono collocate, e funzioni di servizio con lo scopo di generalizzare operazioni che devono essere ripetute. Le prime seguono una determinata nomenclatura gerarchica. Ad esempio *uwl_net_init_edge_router* (fig. 5.3) indica una funzione con nome init_edge_router della libreria µWireless (uwl), collocata nel modulo Network (net).

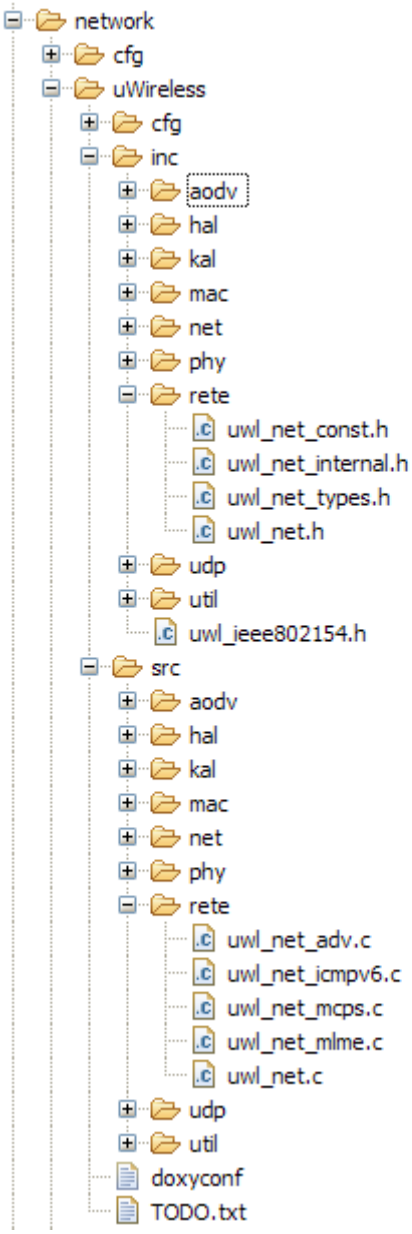


Figura 5.2: Collocazione dei file .c e .h

```
int8_t uwl_net_init_edge_router(uint16_t coordinator_id, uint16_t pan_id,
          uint8_t channel, uint8_t bo, uint8_t so);

void uwl_net_init_router_if(uwl_mac_dev_addr_short_t short_address);

void uwl_net_send_packet(uwl_ipaddr_t dst_addr, uint8_t *payload, uint8_t len,
        enum uwl_net_type_next_header_t next_header, uint8_t hop_limit);
```

Figura 5.3: Esempio di definizione delle funzioni

5.3.1. Implementazione del modulo Network

Lo strato Network dovrà implementare le funzionalità di addressing e di discovery. Sono stati creati i seguenti file sorgente:

- uwl_net.c
- uwl_net_icmpv6.c
- uwl_net_mlme.c
- uwl_net_adv.c
- uwl_net_mcps.c

Nel primo file, *uwl_net.c*, sono contenute le funzioni relative allo start-up dei dispositivi. La funzione chiamata varia a seconda del tipo del nodo: LoWPAN Edge Router o LoWPAN Router. Queste funzioni chiamano al loro interno servizi del livello MAC, come ad esempio *uwl_MLME_START_request* che permette la creazione e l'inizializzazione della PAN. Qui inizia la procedura di “stateless address autoconfiguration”, con la creazione del proprio ID. Vengono inoltre implementate le funzioni per l'invio, il processamento e l'inoltro dei messaggi. L'invio dei dati è stato implementato tramite funzioni che innanzitutto settano l'header con *set_ip_header*, in seguito trasferiscono il pacchetto al livello MAC per essere inviato. La ricezione di un messaggio avviene tramite una callback da parte del livello inferiore. Se il nodo è il destinatario del pacchetto, questo viene processato con l'attivazione di un task. A seconda del tipo di messaggio ricevuto si attiverà un task diverso. Se invece il nodo viene usato solo come hop in una comunicazione multi-hop, il dispositivo si limeterà all'inoltro del messaggio.

I file *uwl_net_mlme.c* e *uwl_net_mcps.c* contengono le implementazioni delle funzioni definite dallo standard IEEE 802.15.4 e che hanno lo scopo di permettere lo scambio di informazioni tra il livello MAC ed il livello Network.

uwl_net_icmpv6.c contiene il codice relativo ai servizi implementati dello standard ICMPv6. Messaggi di questo tipo sono il RS, RA, NR e NC. Sono implementate funzioni che settano i vari valori dei dati da inviare. checksum_icmpv6 è l'algoritmo per il calcolo del checksum da inserire nell'header dei pacchetti ICMPv6.

Infine nel file uwl_net_adv.c sono contenute le funzioni che permettono l'invio dei messaggi periodici RA da parte dei LoWPAN Router. Questo avviene attraverso il settaggio di un allarme collegato ad un task periodico.

5.3.2. Implementazione del modulo UDP

Infine è stato implementato un semplice modulo per l'invio di messaggi tramite il protocollo UDP. Anche in questo caso c'è un solo file sorgente: *uwl_udp.c*. In esso sono contenute delle semplici funzioni per settare l'header del pacchetto. Non è stato implementato nessun tipo di controllo sui valori inviati in un messaggio di tipo UDP.

5.3.3. Implementazione del modulo AODV

Il modulo AODV è relativo all'algoritmo di routing. È stato implementato un solo file sorgente, uwl_aodv.c. Le funzioni principali implementate in questo file sono: *uwl_aodv_next_hop* e *uwl_aodv_find_route*. La prima restituisce il next-hop per raggiungere un destinatario, se presente nella tabella di routing; in caso contravio ritorna zero. La seconda viene chiamata nel momento in cui il risultato della prima funzione è uguale a zero. L'algoritmo di routing è stato implementato così:

- nel momento in cui deve inviare un messaggio, ricerca l'indirizzo del destinatario nella sua tabella dei vicini, solo se non ha un risultato positivo richiama la funzione *uwl_aodv_next_hop*.
- Se anche il risultato della funzione precedente è negativo, allora prima si memorizza il pacchetto da inviare in un buffer e successivamente viene invocata la funzione *uwl_aodv_find_route.*
- Questa funzione implementa l'algoritmo di routing. Viene creato un messaggio di tipo RREQ e inviato in broadcast.
- Appena ricevuto un messaggio di tipo RREP dal destinatario, viene ivocata la funzione di invio dati.

Inoltre in questo modulo sono implementati i metodi per il processamento, aggiornamento delle tabelle di routing, dei messaggi definiti dallo standard. Per l'invio di quest'ultimi si usa il protocollo UDP.

5.4. Implementazione delle strutture dati

Le strutture date sono state implementate tramite l'istruzione struct del C, come si può vedere in figura 5.4. Alcune strutture devono essere tabelle con più entrate. Per far ciò si sono definiti

dei vettori relative ad esse (fig. 5.5).

```
struct uwl_net_whiteboard_t{
    uwl_net_oii_t oii;
    uwl_net_on_t on;
    unsigned primaryFlag : 1;
    uint16_t registrationAge;
    uint16_t lifeTime;
};
```

Figura 5.4: Esempio di struttura dati

```
struct uwl_net_whiteboard_t uwl_net_edge_whiteboard[MAX_NODE_REG];
```

Figura 5.5: Esempio di definizione di una tabella

5.5. Implementazione applicazioni dimostrative

Si sono implementate delle applicazioni dimostrative con lo scopo di testare le librerie implementate. Gli applicativi sono due: una per l'Edge Router e l'altra per i Router. Tutte e due inizializzano il dispositivo ad esse relative.

Il dimostrativo per il router presenta una funzione in più: premendo un pulsante si invia un messagio ad un nodo della rete.

Capitolo 6

6. Risultati sperimentali

In questo capitolo vengono riportate le misure preliminari su un testbed sperimentale di una WSN programmata con le librerie e gli applicativi implementati. Si vogliono provare le funzionalità introdotte nel capitolo 4.

La raccolta dati sul PC è stata effettuata utilizzando eDAQ, in maniera da ottenere la visualizzazione on-line delle misure richieste, nonchè la serializzazione delle stesse sul filesystem utilizzando il framework ROOT, sviluppato dal CERN. I dati acquisiti sono stati poi analizzati off-line utilizzando le funzionalità di ROOT.

È stato utilizzato, inoltre, lo sniffer CC2420EM di pacchetti a R/F compliant con lo strato fisico e MAC dello standard IEEE 802.15.4 allo scopo di debugging e per fornire una serie di istantanee del traffico di rete.

6.1. Installazione sperimentale

6.1.1. Hardware setup

In Figura 6.1 è rappresentata l'installazione sperimentale del sistema. I nodi sono delle FLEX Demo board. La rete è stata configurata staticamente sulla banda a 2.4 Ghz dello standard IEEE 802.15.4: tutti i transceiver CC2420 della rete sono stati sintonizzati su un singolo canale.

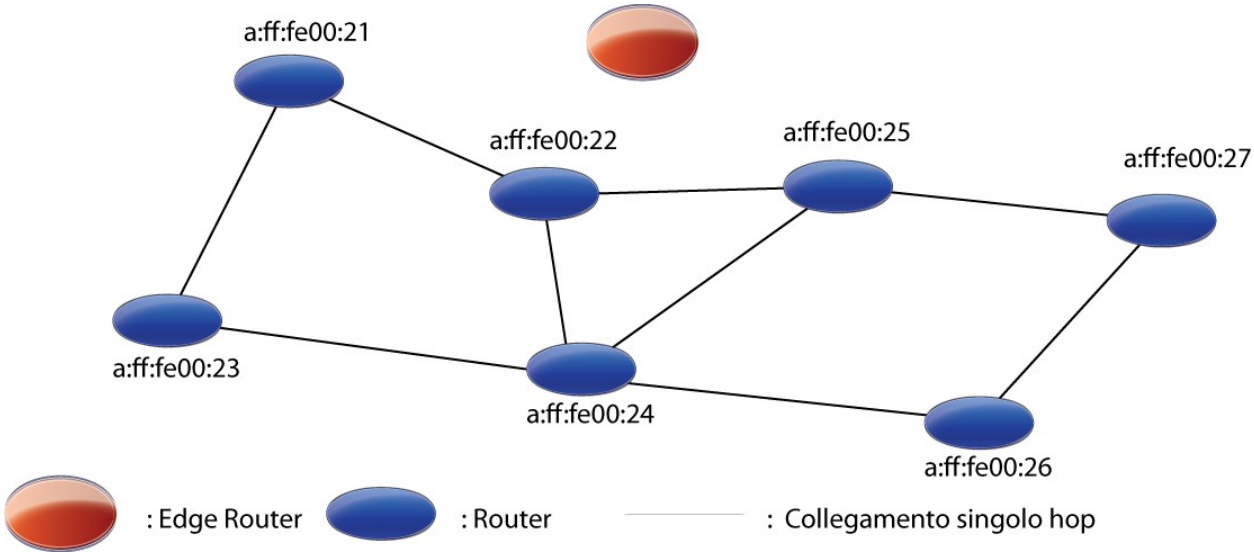


Figura 6.1: Configurazione del sistema in fase di test

I test sono stati effettuati in una stanza e per evitare che i nodi fossero tutti “neighbor”, si è usato uno stratagemma a livello software: con delle semplici istruzioni di condizione, il dispositivo processa solo i messaggi ricevuti da determinati indirizzi.

6.1.2. Round trip daelay

Per "round-trip-delay" si intende il tempo intercorso tra la spedizione di un pacchetto da un nodo e la ricezione da parte dello stesso della risposta del nodo destinatario. Nel caso specifico è stato implementato un semplice dimostratore che permette sia la computazione del round trip delay sia di mostrare gli hop dal pacchetto in andata (per il ritorno il pacchetto seguirà lo stesso percorso). Per effettuare queste misure, il device a:ff:fe00:27 spedirà ogni 2000 ticks (@40MPS) un pacchetto al nodo a:ff:fe00:23 e farà partire un timer che fermerà soltanto alla ricezione della risposta. Tale pacchetto avrà un payload lungo 33 bytes tali che, i primi 8 siano il suffisso del nodo sorgente e tutti gli altri siano 0x00. Ogni nodo intermedio abilitato al forwarding del pacchetto, prima di inoltrarlo al nodo successivo, aggiornerà il payload con il suo suffisso: così facendo utilizzando lo sniffer del livello MAC si potrà capire il percorso seguito dal pacchetto. Il nodo destinatario (a:ff:fe00:23) alla ricezione dei dati invierà un pacchetto di risposta. Al momento della ricezione della risposta il nodo sorgente fermerà il timer e invierà tramite la porta seriale RS-232 il valore del round-trip-delay così ottenuto, che verrà acquisito e salvato tramite eDAQ.

6.1.3. Route formation time

Per "Route formation time" si intende il tempo di servizio (nell'ambito del protocollo AODV) intercorso tra la spedizione del pacchetto RREQ in multicast e la ricezione del pacchetto RREP. Al fine di misurare la distribuzione del route formation time è stato necessario automatizzare il processo di reset della rete, come lo svuotamento del contenuto delle tabelle di routing di tutti i nodi interessati nelle operazioni di instradamento al fine di acquisire statistica sufficiente. In particolare, in questo esercizio, abbiamo seguito la logica seguente:

1) Il nodo sorgente (a:ff:fe00:27) attiva un timer, invia la richiesta in multicast RREQ e aggiorna la sua tabella di routing;
2) I nodi intermedi (in ordine il nodo a:ff:fe00:26 e il nodo a:ff:fe00:24) interpretano la richiesta, la reinoltarno e aggiornano, se necessario, la loro tabella di routing;
3) Il nodo destinatario (a:ff:fe00:23) riceve la richiesta, la interpreta, risponde con il pacchetto RREP e svuota la sua tabella di routing;
4) I nodi intermedi ricevono il pacchetto RRP, lo interpretano, lo reinoltrano e svuotano la loro tabella di routing;
5) Il nodo sorgente riceve il pacchetto RREP, lo interpreta, ferma il timer, spedisce il valore del route formation time via bus seriare RS-232 al PC, svuota la sua tabella di routing e ritorna allo step 1.

I dati sul PC vengono acquisiti da eDAQ e tramite questo visualizzati (sotto forma di istogramma).

6.2. Attività sperimentali

In questa sezione sono riportate le validazioni ed alcuni misure sperimentali effettuate. I test si riferiscono alle seguenti funzionalità:

- addressing, cioè la fase di start-up, in cui il nodo inizializza la sua interfaccia con i relativi indirizzi;
- creazione di un percorso multi-hop per l'invio di pacchetti;
- calcolo del tempo per la creazione del percorso;
- calcolo del tempo di round-trip.

6.2.1. Addressing

Sono proposti una serie di screeshot del traffico di rete per validare l'inizializzazione di un nodo. Come visto nel parafrago 4.4 in fase di start-up si avvia una sequenza di messaggi per l'acquisizione del prefisso della rete e per la registrazione con l'Edge Router.

In Figura 6.2 è riporta la sequenza dei messaggi definiti dal protocollo di registrazione, evidenziando il tipo di messaggio. In seguito verranno esaminati in dettaglio i singoli pacchetti e cioè il Router Solicitation (RS), il Router Advertisement (RA), il Node Registratione (NR) ed il Node Confirmation (NC).

In Figura 6.3 è rappresentato il messaggio di Router Solicitation. In particolare si fa una breve descrizione della compressione dell'header IPv6, ovvero i primi due byte dell'intestazione del pacchetto.

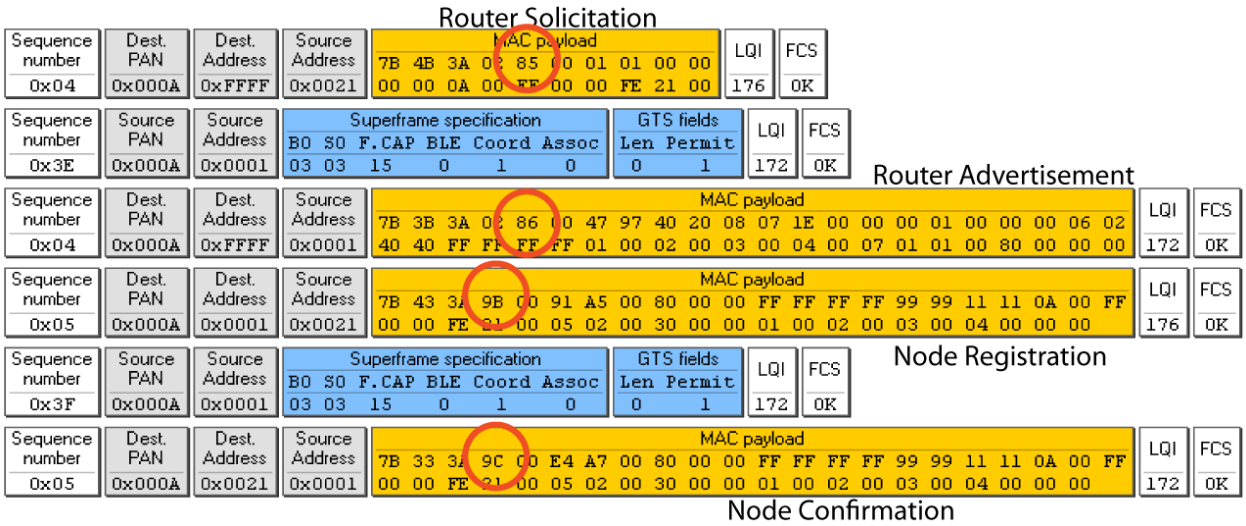
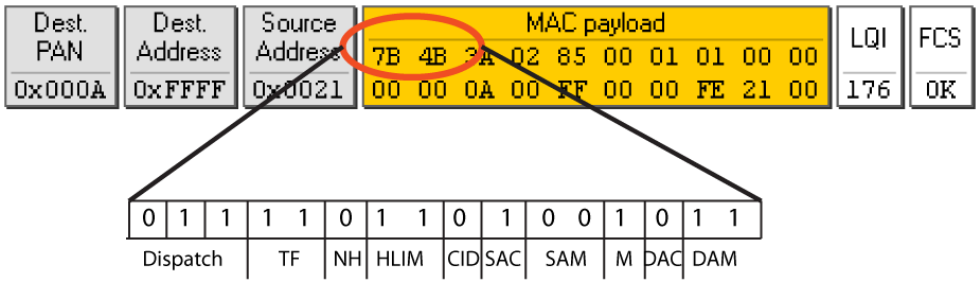


Figura 6.2: Sequenza messaggi in fase di start-up



TF: Traffic Class, Flow Label --> 11: Traffic Class and Flow Label are elided
NH: Next Header --> 0: Full 8 bits for Next Header are carried in-line
HLIM: Hop Limit --> 11: The Hop Limit field is compressed
CID: Context Identifier Extension --> 0: No additional 8-bit Context Identifier Extions is used
SAC: Source Address Compression --> 1: Source address compression uses stateless compression
SAM: Source Address Mode --> 00: The UNSPECIFIED address
M: Multicast Compression --> 1: Destination address is a multicast addresses
DAC: Destination Address Compression --> 0: Destination address compression uses stateless compression
DAM: Destination Address Mode --> 11: The address takes the form FF02: : 00XX

Figura 6.3: Router Solicitation con header compression

Come si può vedere nella Figura 6.4 il suffisso, di 64-bit, creato in fase di start-up segue lo standard Stateless Address Autoconfiguration. L'indirizzo MAC di 16-bit del nodo, dopo l'associazione dinamica, è 0x21 e il PAN ID è 0x0A.

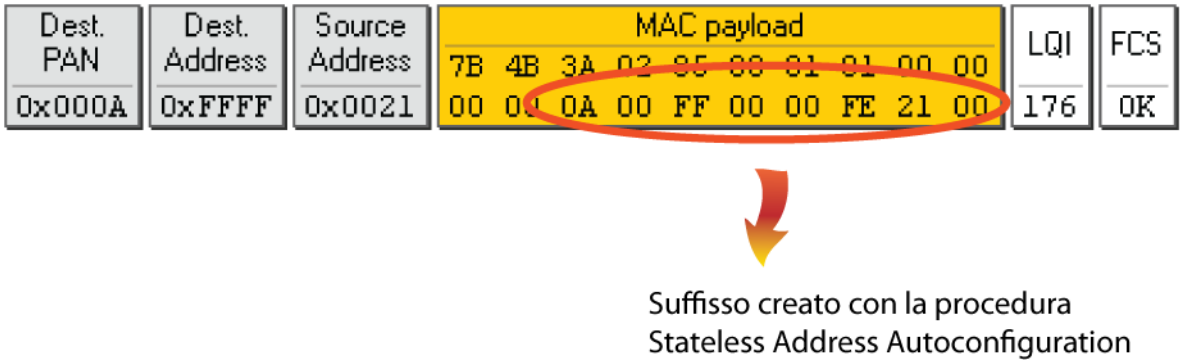


Figura 6.4: Router Solicitation (RS)

Il messaggio Router Solicitation è inviato in multicast, come si può vedere dal campo Dest. Address, a tutti i nodi il quale risponderanno con un Router Advertisement, Figura 6.5. Nel messaggio è evidenziato i valori relativi al prefisso della rete, dato necessario per l'indirizzo Global Unicast.

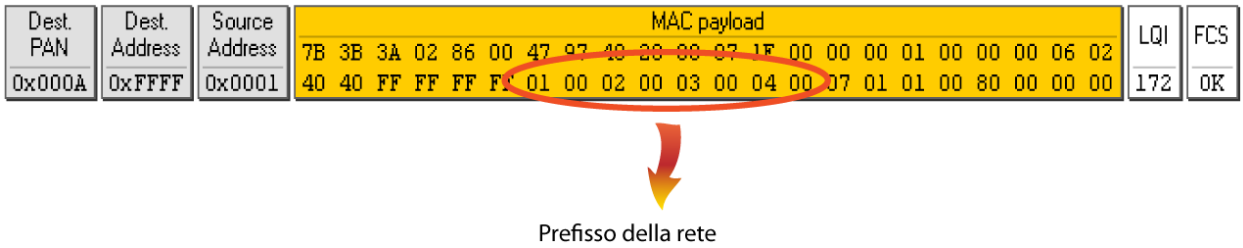


Figura 6.5: Router Advertisement (RA)

Ricevuto il messaggio di RA ed una volta completata la procedura di Stateless Address Autoconfiguration, il dispositivo invia un messaggio di Node Registration all'Edge Router per effettuare la registrazione sulla Whiteboard. Vengono evidenziati nella Figura 6.6 il suffisso ed il prefisso dell'indirizzo del nodo inviante il NR.

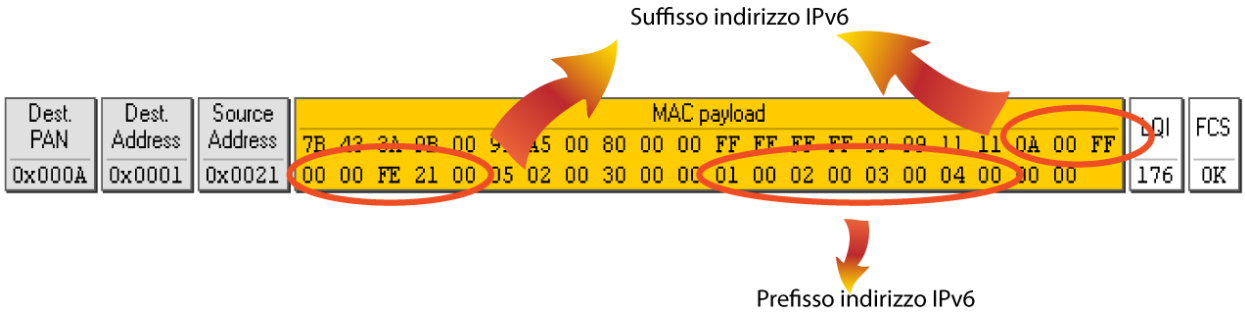


Figura 6.6: Node Registration (NR)

In Figura 6.7 è riportato il messaggio di Node Confirmation inviato dall'Edge Router ed indica la registrazione è andata a buon fine.

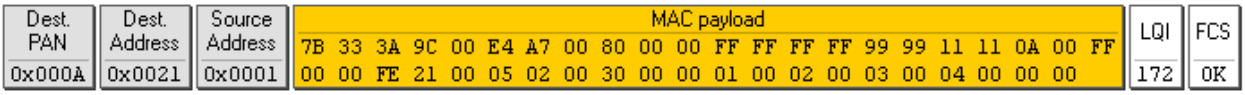


Figura 6.7: Node Confirmation (NC)

6.2.2. Routing

Anche per la validazione del routing verranno proposte una serie di screenshot riguardanti i messaggi scambiati tra i vari nodi della rete per la creazione di un percorso di routing. Inizialmente si presenterà la sequenza dei messaggi del protocollo AODV (fig. 6.8), in seguito verranno esaminati in dettaglio i singoli pacchetti. Infine verrà proposto una tabella per

migliorare la comprensione.

RREQ

Sequence number	Dest. PAN	Dest. Address	Source Address	MAC payload	LQI	FCS
0x06	0x000A	0xFFFF	0x0027	7B 3B 11 02 F4 8E 02 8E 02 10 28 00 00 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	220	OK
0x08	0x000A	0xFFFF	0x0026	78 3B 11 FE 02 F4 8E 02 8E 02 10 28 00 01 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	200	OK
0x08	0x000A	0xFFFF	0x0024	78 3B 11 FD 02 F4 8E 02 8E 02 10 28 00 02 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	196	OK
0x09	0x000A	0x0024	0x0023	7B 33 11 F4 8E 02 8E 02 11 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00 06 00 00 00	112	OK
0x09	0x000A	0xFFFF	0x0022	78 3B 11 FC 02 F4 8E 02 8E 02 10 28 00 03 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	192	OK
0x09	0x000A	0x0026	0x0024	78 33 11 FE F4 8E 02 8E 02 11 00 00 01 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00 06 00 00 00	192	OK
0x09	0x000A	0x0027	0x0026	78 33 11 FD F4 8E 02 8E 02 11 00 00 02 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00 06 00 00 00	192	OK

Figura 6.8: Sequenza messaggi AODV

In Figura 6.9 è illustrato il messaggio RREQ inviato in multicast da parte del nodo iniziatore con suffisso a:ff:fe00:27, contente la richiesta di creazione del percorso per inviare dati al nodo a:ff:fe00:23. Mentre in Figura 6.10 è rappresentato lo stesso messaggio di RREQ, però inviato dal dispositivo a:ff:fe00:24, che è un nodo intermedio.

Dest. PAN	Dest. Address	Source Address	MAC payload	LQI	FCS
0x000A	0xFFFF	0x0027	7B 3B 11 02 F4 8E 02 8E 02 10 28 00 00 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	220	OK

Suffisso del destinatario Suffisso della sorgente

Figura 6.9: RREQ inviato dal nodo sorgente

Dest. PAN	Dest. Address	Source Address	MAC payload	LQI	FCS
0x000A	0xFFFF	0x0024	78 3B 11 FD 02 F4 8E 02 8E 02 10 28 00 02 01 00 00 00 00 00 00 00 01 00 00 00 0A 00 FF 00 00 FE 23 00 0A 00 FF 00 00 FE 27 00	196	OK

Figura 6.10: RREQ inviato da un nodo della rete

In Figura 6.11 si può vedere il messaggio di RREP inviato dal destinatario (a:ff:fe00:23). Il pacchetto viene inviato ad un nodo vicino(a:ff.fe00:24), che farà parte del percorso di instradamento. Infine in Figura 6.12 è illustrato il pacchetto RREP inviato da un nodo della rete, scelto come hop, al dispositivo iniziatore. Si può notare il campo Hop Count che ha un valore di due, ciò indica che dal nodo a:ff:fe00:23 al nodo a:ff:fe00:26 ci sono due hop.

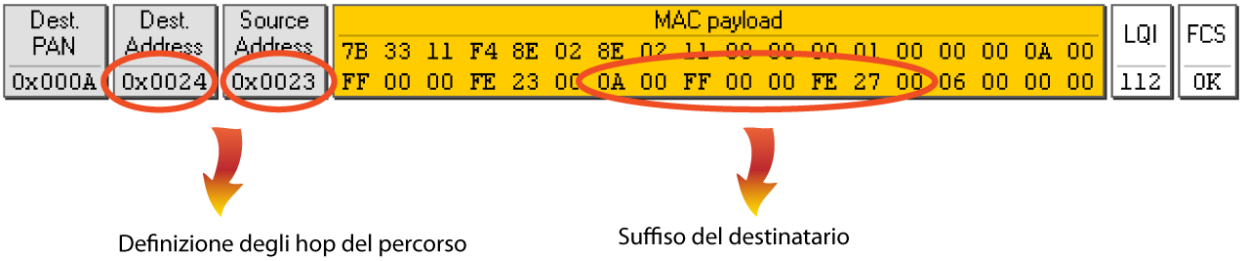


Figura 6.11: RREP inviato dal destinatario

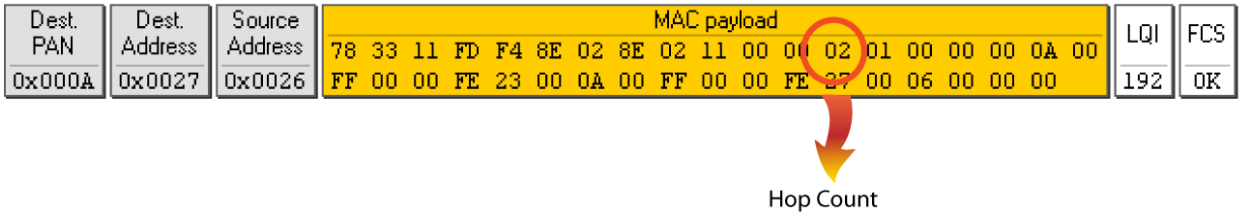


Figura 6.12: RREP inviato da un nodo della rete

Di seguito verrà riportata una tabella semplificativa della sequenza dei messaggi illustrati in Figura 6.8.

Tipo di messaggio	Sorgente MAC	Destinatario MAC	Sorgente IPv6	Destinatario IPv6	Note
RREQ	0x0027	Broadcast	fe80: :a:ff:fe00:27/10	Multicast	
RREQ	0x0026	Broadcast	fe80: :a:ff:fe00:26/10	Multicast	
RREQ	0x0024	Broadcast	fe80: :a:ff:fe00:24/10	Multicast	*
RREP	0x0023	0x0024	fe80: :a:ff:fe00:23/10	fe80: :a:ff:fe00:24/10	**
RREQ	0x0022	Broadcast	fe80: :a:ff:fe00:22/10	Multicast	
RREP	0x0024	0x0026	fe80: :a:ff:fe00:24/10	fe80: :a:ff:fe00:26/10	***
RREP	0x0026	0x0027	fe80: :a:ff:fe00:26/10	fe80: :a:ff:fe00:27/10	****

Tabella 6.2: Sequenza messaggi AODV

* Prima di inviare il messaggio di RREQ, il dispositivo a:ff:fe00:24 salva nella sua tabella di

routing l'indirizzo a:ff:fe00:27 come destinatario e come next-hop a:ff:fe00:26.

** Il dispositivo a:ff:fe00:23 salva nella sua tabella di routing come destinatario a:ff:fe00:27 e come next-hop a:ff:fe00:24 e poi invia il messaggio di RREP.

*** a:ff:fe00:26 salva nella sua tabella di routing il destinatario a:ff:fe00:23 e come next-hop a:ff:fe00:24.

**** Il messaggio di RREP arrivato al nodo sorgente della comunicazione, a:ff:fe00:27, aggiorna la sua tabella di routing inserendo come destinatario a:ff:fe00:23 e come next-hop a:ff:fe00:26.

6.2.3. Misura del Route formation time

Il seguente grafico (fig 6.13) mostra la distribuzione dei tempi necessari alla creazione di un percorso tra un nodo sorgente ed un nodo destinatario utilizzando una comunicazione wireless multi-hop. Chiaramente la distribuzione dei tempi è compatibile con una gaussiana.

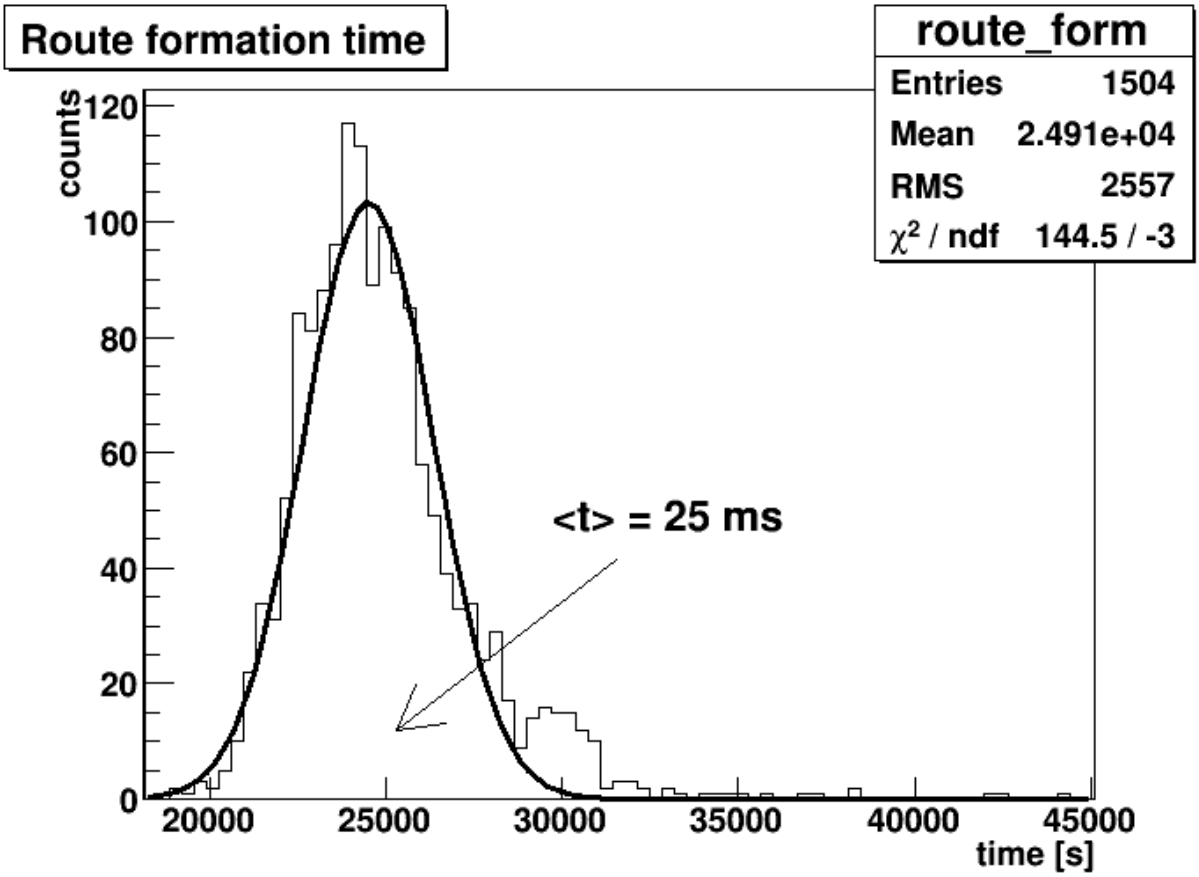


Figura 6.13: Tempi per creare un percorso

Nella fase di creazione del percorso, nel caso della nostra configurazione di rete, vengono inviati 3 pacchetti di 41 byte (RREQ) e 3 pacchetti di 36 byte (RREP). I byte sono relativi solo al pacchetto di livello network. Aggiungendo 11 byte per ogni messaggio inviato relativi al livello MAC e considerando la velocità massima dello standard IEEE 802.15.4 è di 250 kbit/s, possiamo calcolare i tempi di trasmissione con la seguente formula:

numero di byte * 8 bit / 250 kbit/s = tempo di trasmissione in ms.

Da quanto detto l'invio di 3 messaggi RREQ ha un tempo di trasmissione di 4,992 ms e l'invio di 3 messaggi RREP ha un tempo di trasmissione di 4,504. Sommando i due risultati si ha 9,504 ms. Dal grafico si ha una media di 25 ms, ciò ci indica che l'intero stack aggiunge un tempo di elaborazione dei pacchetti all'incirca di 15,5 ms.

Inoltre, dal grafico, si può notare dei valori alla destra della curva: questo è dovuto all'accodamento dei messaggi. Ricordiamo che un nodo deve inviare periodicamente dei Router Advertisement che possono rallentare l'invio di altri pacchetti.

6.2.4. Misura del round trip delay

Nella fase di misura del tempo di round trip delay, ogni nodo del percorso aggiunge al payload del pacchetto il suo suffisso di 64-bit, come illustrato in Figura 6.14. Il messaggio rappresentato è inviato dal nodo a:ff:fe80:24. Prima di essere ricevuto dal nodo a:ff:fe80:24, il pacchetto è stato inviato dal nodo a:ff:fe80:27, nodo sorgente; ricevuto, processato e inviato dal nodo a:ff:fe80:25.

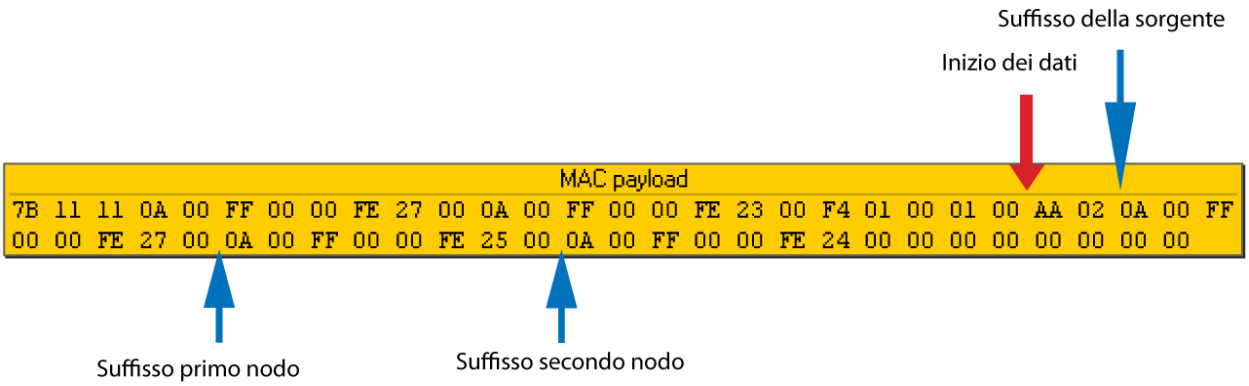


Figura 6.14: Messaggio inviato nel test di Round Trip

Infine in Figura 6.15 mostra la distribuzione, anch'essa gaussiana come la precedente, dei

tempi di Round Trip. La lunghezza dei messaggi, rispetto al caso precedente è diversa: si inviano 3 pacchetti con una lunghezza di 57 byte e 3 pacchetti con una lunghezza di 25 byte. Il tempo trasmissivo totale è di 9,984 ms. Dal grafico si ha una media di 40 ms. In questo caso lo stack aggiunge un tempo computazionale di circa 30 ms. Rispetto al risultato ottenuto con la precedente misurazione è maggiore. Questo è dovuto alle maggiori operazioni effettuate nel Round trip delay, ad esempio come la scrittura nel payload del pacchetto il suffisso di 64-bit.

Anche in questo caso sono presenti valori alla destra della curva, ed anche in questo caso è dovuto all'accodamento dei pacchetti.

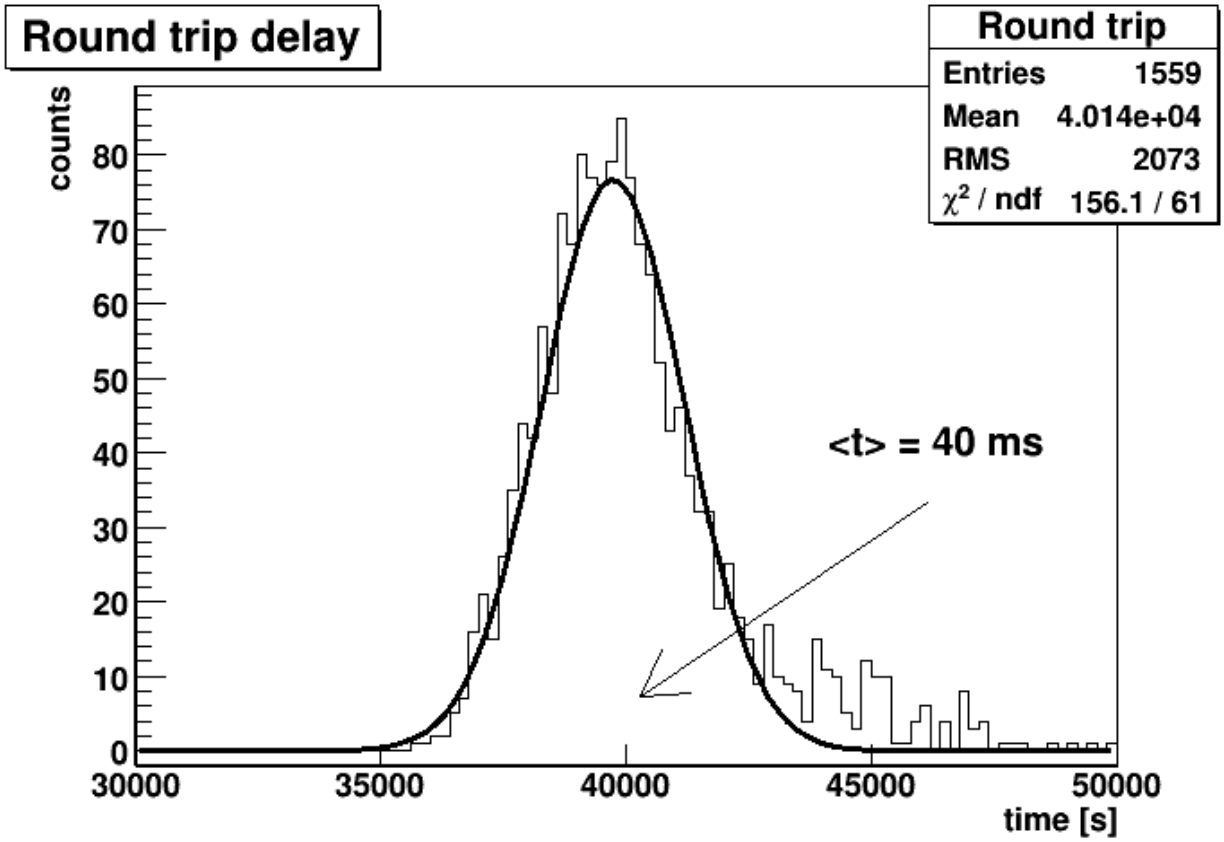


Figura 6.15: Tempi di Round Trip

Conclusioni

In questa tesi è stata presentata l'implementazione di uno stack di comunicazione wireless conforme alle specifiche dell' IPv6. E' stata inoltre dimostrata l'interoperabilità dell'architettura con i protocolli standard per la comunicazione in una rete di sensori.

Per la parte implementativa, si è usata la development board per i microcontrollori dsPIC® a 16 bit della Microchip (scheda FLEX) su cui è stato applicato il transceiver a R/F Texas Instruments CC2420 conforme allo standard IEEE 802.15.4.

Ci si è riferiti al Know-How della Scuola Superiore Sant'Anna in termini di sistemi operativi micro-kernel e di protocolli per la gestione delle risorse con garanzie di tempo reale; in particolare il codice è stato sviluppato sull' ambiente open-source (Eclipse) e freeware (RT-Druid, MPLAB).

Per la validazione delle funzionalità dello stack si è fatto uso dello sniffer CC2420EM della Texas Instruments per il monitoraggio dell'attività di rete e del package eDAQ per la serializzazione e l'analisi dei dati formattati e trasferiti sul disco di un PC.

Il progetto di ricerca IPERMOB della Scuola Superiore Sant'Anna di Pisa prevede di applicare la tecnologia sviluppata in questa tesi su opportuni casi di studio legati alla infomobilità e alla gestione del traffico urbano con una rete di sensori topologicamente complessa, opportunistica e pervasiva.

Il progetto verrà inoltre pubblicizzato ed il codice distribuito su licenza LGPL attraverso il sito web di Evidence srl.

Bibliografia

[1] John A. Stankovic and K. Ramaritham, editors. *Tutorial on Hard Real-Time Systems.* IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[2] ARTEMIS Embedded Computing Systems Initiative. https://www.artemis-ju.eu/.

[3] Auto-ID Labs. http://www.autoidlabs.org/.

[4] Chee yee Chong, Ieee, Srikanta P. Kumar, and Senior Member. Sensor networks: evolution, opportunities, and challenges. In *Proceedings of the IEEE*, pages 1247-1256, 2003.

[5] SCIER - A technological simulation platform to managing natural hazards. Available: http://www.scier.eu/.

[6] WINSOC - Wireless Sensor Network with Self-Organization Capabilities for critical and emergency applications. Available: http://www.winsoc.org/.

[7] ALERT SYSTEM. http://alert.dot.pima.gov/.

[8] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szwczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In WSNA '02: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88-97, New York, NY, USA, 2002. ACM

[9] SensAction-AAL - SENSing and ACTION to support mobility in Ambient Assisted Living. http://www.sensaction-aal.eu/.

[10] SMARTFREIGHT - Smart Freight Transport in Urban Areas. http://www.smartfreight.info/.

[11] CoBIs - Collaborative Business Items. http://www.cobis-online.de/.

[12] ZigBee Alliance. http://www.zigbee.org/.

[13] Contiki Operating System for Embedded Smart Objects. http://www.sics.se/contiki/.

[14] LAN-MAN Standards Commitee of the IEEE Computer Society. *Wireless Medium Access Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs).* IEEE Press, 2003.

[15] IETF - The Internet Engineering Task Force. http://www.ietf.org/.

[16] IPv6 over Low power WPAN (6lowpan). http://www.ietf.org/dyn/wg/charter/6lowpan-charter.html.

[17] Mobile Ad-hoc Networks (MANET). http://www.ietf.org/dyn/wg/charter/manet-charter.html.

[18] Microchip Technology Inc. http://www.microchip.com/.

[19] Evidence srl, Pisa(I). http://www.evidence.eu.com/.

[20] Embedded Solutions srl, Milan(I). http://www.es-online.it/.

[21] Paolo Gai, Giuseppe Lipari, Luca Abeni, Marco di Natale, and Enrico Bini. Archichetture

for a portable open source real-time kernel environment. In *Proceedings of the Second Real-Time Linux Workshop and Hand's on Real-Time Linux Tutorial* November 2000.

[22] OSEK/VDX Portal. http://www.osek-vdx.org/.

[23] Giorgio C. Buttazzo. *Sistemi in Tempo Reale* chapter 12 Sistemi Operativi Esistenti, pages 341-362. Third edition, 2006.

[24] Eclipse Framework. http://www.eclipse.org/.

[25] CC2420DK Development Kit. http://focus.ti.com/docs/toolsw/folders/print/cc2420dk.html.